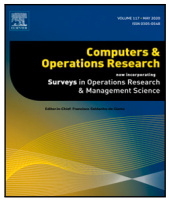
Computers & Operations Research 173 (2025) 106826



Contents lists available at ScienceDirect

Computers and Operations Research

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





Adaptive large neighborhood search for drayage routing problems involving longer combination vehicles

Daniel Bustos-Coral *, Alysson M. Costa

School of Mathematics and Statistics, The University of Melbourne, VIC 3010, Australia

ARTICLE INFO

Dataset link: https://github.com/dbuscor/drp_hcl_instances

Keywords:
Drayage routing
Heterogeneous fleet
Compatibility constraint
Truck load configuration
Longer combination vehicle
Adaptive large neighborhood search

ABSTRACT

Longer combination vehicles (LCVs)—tractor-trailer combinations with two or more trailers—can move more cargo per trip than standard trucks, potentially reducing costs and emissions. Acknowledging this, the drayage industry has recently launched initiatives to increase the adoption of LCVs, aiming to mitigate supply chain issues. This paper aims to support these kinds of initiatives by tackling drayage routing problems involving LCVs and further aspects, such as heterogeneous truck fleets, containers of any size and cargo category, and compatibility constraints based on the containers' sizes and loads. As solution method, we propose an adaptive large neighborhood search (ALNS) heuristic, whose search operators account for aspects such as truck/container compatibility and load configurations specific to each truck. The proposed ALNS heuristic also incorporates a novel acceptance criterion that seeks to escape local optima while allowing for revisiting the current best solution and avoiding search stagnation. Through a series of benchmark tests against a state-of-the-art exact approach, we show that the proposed ALNS heuristic can consistently find high-quality solutions for a wide range of instances while, in some cases, cutting runtimes from hours or even days to a few minutes. We also show that the proposed acceptance criterion enables improved performance. Finally, we use the proposed ALNS heuristic to derive managerial insights into the savings delivered by different types of LCVs in drayage operations.

1. Introduction

Drayage—the short-distance trucking of containers—is vital in intermodal transportation, as it fills the gap between long-haul vehicles and inland facilities. The importance of drayage is accentuated by its high costs, relative to the total intermodal costs. For example, in the case of rail-truck intermodal operations in the US, even conservative estimates show that drayage can account for between 50% and 75% of the total intermodal costs (Resor et al., 2004).

Technologies such as *longer combination vehicles* (LCVs), *i.e.*, semi-trucks pulling two or more trailers/chassis, offer alternatives to tackle the high costs of drayage. LCVs offer cost and emission benefits due to their higher cargo capacity compared to standard trucks, *i.e.*, rigid trucks or semi-trucks pulling a single trailer/chassis. However, the worldwide adoption of LCVs is limited due to infrastructure needs and regulations. Despite this, examples of successful LCV operations can be found in Australia, Canada, and some states of the US.

Particularly in the US, there have been recent initiatives of the drayage industry to adopt LCVs as a means to tackle congestion and driver shortages (Fletcher, 2022). Research on the impact of LCVs on drayage operations (see, *e.g.*, Thompson and Hassall, 2014) is necessary to support these kinds of initiatives. However, this research topic has not garnered much attention, arguably due to the low usage of LCVs worldwide.

In response to this research need, this paper addresses drayage problems where different types of standard trucks and LCVs perform container pickup and delivery services. We are interested in quantifying the savings that different LCV configurations can realize over standard trucks. In particular, throughout this paper we consider the truck configurations shown in Fig. 1, which are based on the US truck classes described by Tunnell (2008).

To tackle the problems of interest, we consider as framework the *drayage routing problem with heterogeneous fleet, compatibility constraints, and truck load configurations* (DRP-HCL), proposed by Bustos-Coral and Costa (2022). The DRP-HCL allows for considering trucks of any type, with different load configurations, and containers of any size and cargo category, with compatibility constraints based on the containers' sizes and cargoes (*e.g.*, only trucks pulling a 53 ft chassis can haul 53 ft containers, only trucks pulling a genset chassis can move reefer containers, and containers carrying incompatible loads cannot be moved together).

* Corresponding author.
E-mail addresses: dbustos@student.unimelb.edu.au (D. Bustos-Coral), alysson.costa@unimelb.edu.au (A.M. Costa).

https://doi.org/10.1016/j.cor.2024.106826
Received 12 September 2023; Received in revised form 24 July 2024; Accepted 27 August 2024
Available online 30 August 2024
0305-0548/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

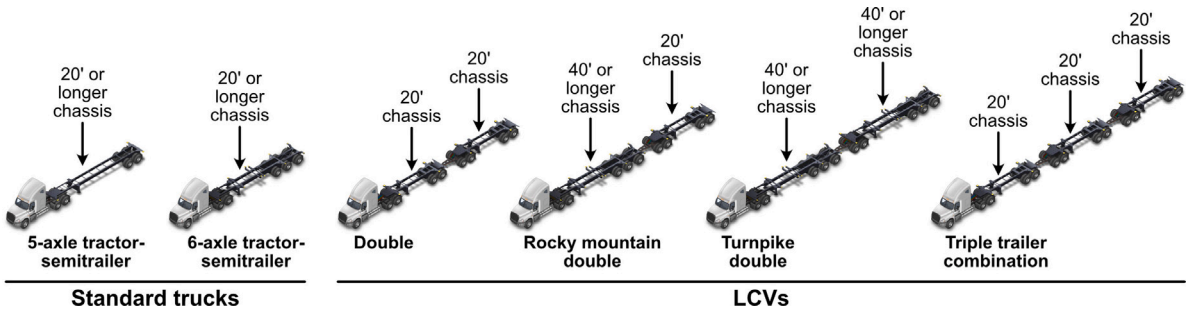


Fig. 1. Considered truck configurations.

To model truck cargo capacities and compatibility constraints, Bustos-Coral and Costa (2022) proposed a state transition logic based on which they modeled different types of standard trucks, which can accommodate up to two containers. In this paper, we extend this logic to model LCVs, which can accommodate three or four containers.

Bustos-Coral and Costa (2022) developed two exact approaches to the DRP-HCL: a compact mixed-integer programming (MIP) formulation and a branch-and-cut algorithm, with the latter being capable of solving larger instances. Computational experiments revealed that the performances of these approaches are significantly influenced by the trucks' cargo capacities: instances involving double-container trucks are generally harder to solve than instances involving only single-container trucks, to the point where many instances involving double-container trucks remained unsolved.

The above-mentioned approaches are generally unsuitable to tackle the DRP-HCL involving LCVs, as these trucks have a larger cargo capacity than standard trucks. Indeed, our experiments indicated that the branch-and-cut algorithm of Bustos-Coral and Costa (2022) can need several days of computation time to solve some (relatively small) instances involving LCVs and 100 requests.

Hence, to solve DRP-HCL instances involving LCVs (and other difficult instances), in this paper we present an *adaptive large neighborhood search* (ALNS) heuristic, based on the destroy-and-repair framework of Ropke and Pisinger (2006). Remarkably, we develop destruction/repair operators tailored to the DRP-HCL, which explicitly account for aspects such as compatibility constraints and load configurations specific to each truck. Furthermore, we propose the *pulsed threshold acceptance* (PTA) criterion, a problem-independent mechanism based on the record-to-record travel (RRT) acceptance criterion (Santini et al., 2018), which aims to escape local optima while preventing search stagnation and allowing for revisiting the current best solution.

Through a comprehensive set of benchmark tests against the branch-and-cut approach of Bustos-Coral and Costa (2022), we show that the proposed ALNS heuristic can consistently find high-quality solutions (we found new best known solutions for several instances). Furthermore, when solving hard instances (especially those involving LCVs), the proposed ALNS heuristic can cut runtimes from hours (or even days) to a few minutes. We also report computational results that show that the PTA criterion can yield better results than the RRT acceptance criterion. Finally, we show how the proposed ALNS heuristic can be used to derive managerial insights into the use of different types of LCVs and standard trucks (those of Fig. 1) for drayage services.

The main contributions of this paper are thus (1) an extension of the state transition logic of Bustos-Coral and Costa (2022) to model LCVs; (2) a state-of-the-art ALNS heuristic for the DRP-HCL that includes tailored operators and can solve instances involving LCVs (and other difficult instances) that cannot be solved in practical times using previously proposed approaches; (3) the PTA criterion, which can outperform the RRT acceptance criterion and, due to its generic nature, has potential for use in any path-based heuristic framework; and (4) an analysis of the performance in drayage operations of various US truck classes, including both LCVs and standard trucks.

The remainder of this paper is organized as follows: Section 2 reviews literature on drayage problems involving LCVs. Section 3 describes the DRP-HCL, focusing on the modeling of LCVs. Section 4 details the proposed ALNS heuristic. Section 5 presents computational results, analyzing the performance of the proposed ALNS heuristic and showing its utility to derive managerial insights into the use of LCVs. Finally, Section 6 concludes the paper.

2. Related work

In this section, we first review the literature on drayage problems involving LCVs. Then, we draw a comparison between the reviewed studies and ours, discussing how our study advances this body of work.

Lai et al. (2015) proposed an integer programming formulation and an adaptive guidance search heuristic for a drayage problem involving moves of 24.5 ft containers. The authors considered standard trucks, capable of accommodating one or two containers, and LCVs, capable of accommodating up to six containers. The authors found that three-container trucks can reduce routing costs by up to 25% with respect to double-container trucks, four-container trucks can reduce routing costs by up to 46% with respect to two-container trucks, and six-container trucks can reduce routing costs by up to 27% with respect to four-container trucks and by up to 79% with respect to single-container trucks.

Vidović et al. (2017) proposed a MIP formulation and a variable neighborhood search (VNS) heuristic for a drayage pickup and delivery problem involving movements of 20/40 ft containers. In this problem, the fleet of trucks is composed of either standard trucks, European modular system (EMS) trucks, or LCVs. Standard trucks can carry one full 20/40 ft container, making up to two stops on their routes; EMS trucks can carry up to two full 20 ft containers, making up to four stops on their routes; and LCVs can carry up to two 40 ft, four 20 ft, or even three 40 ft containers, making up to eight stops on their routes. The authors reported that, when compared to standard trucks, EMS trucks and LCVs can reduce transportation costs by up to 15% and 51%, respectively.

Moghaddam et al. (2020) proposed a MIP formulation for a drayage problem involving movements of 20/40 ft containers and various truck fleets: a homogeneous fleet of 40 ft trucks, a low-capacity fleet composed mostly of 20 ft and 40 ft trucks, a high-capacity fleet composed mostly of LCVs (60 ft and 80 ft trucks), and a balanced fleet evenly composed of 20/40/60/80 ft trucks. The authors reported that the high-capacity fleet achieved the lowest costs, followed (in order) by the balanced fleet, the homogeneous fleet, and the low-capacity fleet.

Zhang et al. (2021) developed a MIP formulation and a simulated annealing-based heuristic for a drayage problem involving a fleet of homogeneous semi-trucks, trailers, and dry containers. The semis can be coupled/uncoupled to one or more trailers, and each trailer can carry a single container. While the semis can travel alone or pulling trailers, the latter cannot travel alone, and require a semi to move them. This problem extends the truck and trailer routing problem (TTRP; Chao, 2002), where semis can be coupled/uncoupled to a single trailer. The

Table 1
Comparison of drayage problems involving LCVs.

Paper	Het. cont.	Container types	Het. trucks	Comp. constr.	Time windows	Solution approach	Size of largest instance
Lai et al. (2015)		24.5 ft containers (f&e)				E/H	50 customers
Vidović et al. (2017)	✓	20/40 ft containers			✓	E/H	193 tasks
Moghaddam et al. (2020)	✓	20/40 ft containers (f&e)	✓	✓(s)	✓	E/H	89 requests
Zhang et al. (2021)		Dry containers (f&e), trailers				E/H	100 orders
Chen et al. (2021)	✓	Any container	✓	✓(s)		E/H	300 requests
Bjelić et al. (2022)	✓	20/40 ft containers (f&e)			✓	E/H	96 tasks
This paper	✓	Any container (f&e)	✓	✓(s&c)	✓	H	1,000 requests

(f&e): full and empty (s): size (s&c): size and cargo E: exact H: heuristic.

authors remarked that two-trailer combinations can yield a reduction in working times of 28% on average when compared to single-trailer combinations.

Chen et al. (2021) proposed a MIP formulation and a VNS heuristic for a drayage problem that can involve different container types (the authors considered 20/40 ft containers) and LCVs (the authors considered trucks that accommodate up to six 20 ft containers). This problem involves compatibility constraints, as small trucks cannot move large containers. The authors noted that the solvability of their formulation is largely affected by the number of truck types considered in a problem instance. On the other hand, their VNS heuristic scales well enough for real-world applications.

Bjelić et al. (2022) developed a VNS heuristic for a dynamic drayage routing problem where trucks move 20/40 ft containers between customer locations and a depot. The authors considered fleets composed of either single-trailer trucks, which can move up to two 20 ft containers or one 40 ft container, or multi-trailer trucks, which can move up to four 20 ft containers or two 40 ft containers. The authors reported that, when multi-trailer trucks are used, travel distances can be reduced by nearly 30% and the number of trips required to serve the requests can be reduced by nearly 40%.

Table 1 compares the reviewed studies with ours based on the following aspects: consideration of heterogeneous containers, types of containers, consideration of heterogeneous trucks, consideration of compatibility constraints (regarding container size or cargo), inclusion of time-window constraints, type of solution approach (exact or heuristic), and the size of the largest problem instance tackled.

Note that the consideration of LCVs in drayage problems is a relatively new trend, as the body of literature on this matter is still small, and most of it has been published recently. Indeed, Chen et al. (2022) note that, in general, the literature on drayage problems has diversified over the last few years, as researchers have been starting to focus on new directions of research. We can argue that the introduction of LCVs in drayage is one such direction. This paper continues this research effort by tackling a problem involving heterogeneous containers and LCVs, as well as compatibility requirements concerning container size (*e.g.*, a 20 ft truck cannot handle a 40 ft container) and cargo (*e.g.*, containers carrying hazardous loads must be moved by special trucks and cannot be moved alongside other types of containers). As Table 1 shows, when it comes to the modeling of trucks and containers, none of the surveyed studies has considered a problem with all the characteristics of the problem that we are tackling.

3. Problem definition

The DRP-HCL concerns the transportation of containers from pickup locations to delivery locations using different types of trucks, with specific fixed and routing costs. The containers can vary in size and cargo category, and the trucks can handle specific types of containers, based on the containers' sizes and contents. In addition, some containers cannot be transported together due to incompatible loads, *e.g.*, food-grade containers should not be transported alongside containers carrying hazardous materials. Pickups and deliveries have separate time windows and service times, and all container movements must be completed within a given planning horizon. The aim is to plan a set of container movements defining an optimal fleet composition and an optimal routing plan. Below, we introduce the notation of the problem and delve into the modeling of LCVs.

3.1. Notation

We assume n drayage requests, with each request involving the movement of a container from a pickup location to a delivery location. Define $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ as a directed graph, with $\mathcal{N} = \{0, 1, \dots, 2n, 2n+1\}$ and $\mathcal{A} = \{(i, j) : i \in \mathcal{N}, j \in \mathcal{N}\}$. Nodes 0 and $2n+1$ represent the start and end depot, respectively. $\mathcal{P} = \{1, \dots, n\}$ is the subset of pickup nodes and $\mathcal{D} = \{n+1, \dots, 2n\}$ is the subset of delivery nodes. Each service request i, $1 \le i \le n$, is picked up at node $i \in \mathcal{P}$ and delivered at node $n+i \in \mathcal{D}$.

To each node in $\mathcal{N}$ there is an associated service description that specifies the service to perform at that node. Let $\mathcal{U} = \{+, -\}$ be the set of service types: $+$ for pickup and $-$ for delivery. Let $\mathcal{V}$ be the set of container types considered in a problem instance. For example, $\mathcal{V} = \{\texttt{e20}', \texttt{f20}', \texttt{40}'\}$ if the instance involves 20 ft containers (empty or full) and 40 ft containers. Note that we distinguish between empty and full containers only when there are trucks whose load capacities vary depending on the containers' contents (for example, trucks capable of hauling three empty 20 ft containers or two full 20 ft containers). When truck load capacities remain the same regardless of the containers' contents, no distinction between empty and full containers is necessary. For each node $i \in \mathcal{P} \cup \mathcal{D}$, the service description is given by $r_i = (u_i, v_i) \in \mathcal{U} \times \mathcal{V}$. Specifically, for each request j, $1 \le j \le n$, the service descriptions of its associated pickup and delivery nodes have the following pattern: $u_j = +$, $u_{n+j} = -$, and $v_j = v_{n+j}$; that is, the same container that is picked up at node j is delivered at node $n+j$. For example, if $r_j = (+, \texttt{e20}')$, then $r_{n+j} = (-, \texttt{e20}')$. For the nodes representing the start and end depots, the service descriptions are given, respectively, by $r_0 = \texttt{start}$ and $r_{2n+1} = \texttt{end}$. The set of all service descriptions is given by $\mathcal{R} = (\mathcal{U} \times \mathcal{V}) \cup \{\texttt{start}, \texttt{end}\}$.

Each node $i \in \mathcal{N}$ has a time window $[e_i, l_i]$ and a service duration s_i. For $i \in \mathcal{P} \cup \mathcal{D}$, e_i and l_i are respectively the earliest and latest times for starting service at the node, and $s_i > 0$. For the depot nodes, the time windows $[e_0, l_0] = [e_{2n+1}, l_{2n+1}]$ represent the planning horizon, and $s_0 = s_{2n+1} = 0$.

Let $\mathcal{F} = \{1, \dots, F\}$ be a set of truck types. The fixed cost associated with using a truck of type $f \in \mathcal{F}$ is given by E^f. The routing cost that a truck of type f incurs when it traverses an arc $(i, j) \in \mathcal{A}$ is given by $c^f_{i,j}$. The travel time needed by a truck $f \in \mathcal{F}$ to traverse an arc $(i, j) \in \mathcal{A}$ is given by $\tau_{i,j}$; this time is assumed to be truck independent. For a truck type $f \in \mathcal{F}$, we assume that costs $c^f_{i,j}$ depend on the travel times $\tau_{i,j}$, for $(i, j) \in \mathcal{A}$.

Each truck type $f \in \mathcal{F}$ is compatible with specific containers and has a load capacity that depends on the containers' sizes/contents (*e.g.*, a truck could be loaded with either one 40 ft container, three empty 20 ft containers, two full 20 ft containers, or one 40 ft container and one 20 ft container). We delve into the modeling of the trucks' capacities and compatibility constraints in Section 3.2.

The objective of the DRP-HCL is to route a fleet of trucks (of types $f \in \mathcal{F}$) to serve the drayage requests of graph $\mathcal{G}$ while minimizing

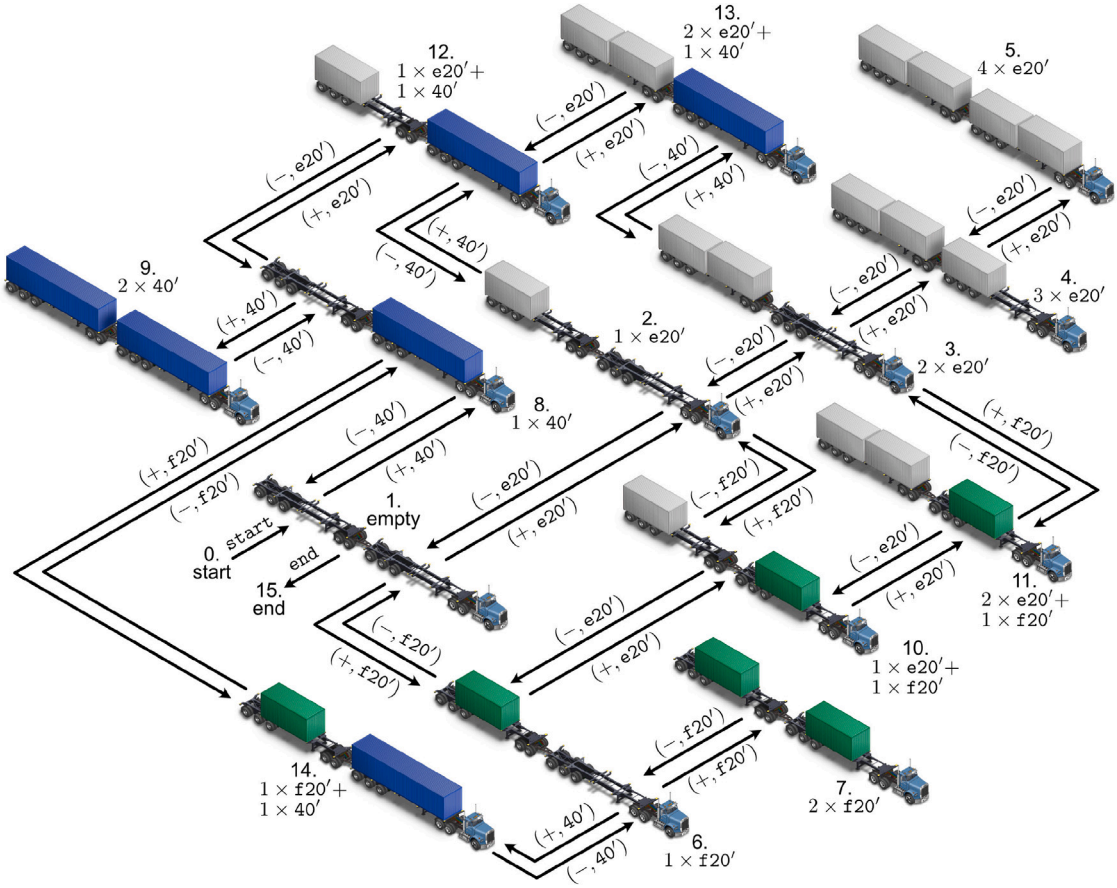


Fig. 2. State transition graph for turnpike doubles that haul 20/40 ft dry containers.

all fixed and routing costs. All trucks must depart from and return to the depot, all requests must be served exactly once, and all time-window constraints must be met. Appendix A presents a compact MIP formulation for the problem.

3.2. Modeling the trucks' load capacities and compatibility constraints

To model the trucks' load capacities and compatibility constraints, we use the state transition logic described by Bustos-Coral and Costa (2022). Such logic consists of defining a state transition graph for each truck type $f \in \mathcal{F}$, where the nodes represent possible load configurations, and the arcs represent the changes in a truck's load as it picks up and delivers containers.

Specifically, we first define a set $\Theta_f = \{0, 1, \ldots, m_f - 1, m_f\}$, which contains the possible states for a type f truck. States $1, \ldots, m_f - 1$ represent the load configurations the truck can have along its route. States 0 and m_f are the start and final states of the truck, respectively.

Define a subset $\Sigma_f \subset (\Theta_f \times \mathcal{R})$ containing ordered pairs $(h, r) \in (\Theta_f \times \mathcal{R})$ such that the service described by r can be undertaken by a type f truck in state h. To determine the state transitions for a type f truck, we define a transition function $q_f : \Sigma_f \to \Theta_f$; $(h, r) \mapsto q_f(h, r)$.

The state transition graph for a type f truck is composed of nodes representing truck states $h \in \Theta_f$ and arcs representing transitions (h, h') such that $q_f(h, r) = h'$, with $(h, r) \in \Sigma_f$ and $h' \in \Theta_f$.

Bustos-Coral and Costa (2022) used this state transition logic to model standard trucks. The idea is here extended to consider LCVs. For example, consider Fig. 2, which shows the state transition graph for turnpike doubles that haul 20/40 ft dry containers, say $f = 1$. The node set of this graph is $\Theta_1 = \{0, 1, \ldots, 15\}$; nodes $1, \ldots, 14$ represent load configurations—*e.g.*, node 7, labeled as $2 \times \mathtt{f20'}$, indicates that the truck is carrying two full 20 ft dry containers. Nodes 0 and 15 represent the truck's start and final states, respectively. Each arc is labeled with a service description $r \in \mathcal{R}$, with $\mathcal{R} = (\{+, -\} \times \{\mathtt{e20'}, \mathtt{f20'}, \mathtt{40'}\}) \cup \{\mathtt{start}, \mathtt{end}\}$. Each arc's service description indicates the service a truck must perform to transition from the node at the tail of the arc to the node at the head of the arc. In this case, the arcs are defined by the following transition function:

$$
\begin{aligned}
&q_1(0, \mathtt{start}) = 1, && q_1(1, (+, \mathtt{e20'})) = 2,\\
&q_1(2, (+, \mathtt{e20'})) = 3, && q_1(3, (+, \mathtt{e20'})) = 4,\\
&q_1(4, (+, \mathtt{e20'})) = 5, && q_1(5, (-, \mathtt{e20'})) = 4,\\
&q_1(4, (-, \mathtt{e20'})) = 3, && q_1(3, (-, \mathtt{e20'})) = 2,\\
&q_1(2, (-, \mathtt{e20'})) = 1, && q_1(1, (+, \mathtt{f20'})) = 6,\\
&q_1(6, (+, \mathtt{f20'})) = 7, && q_1(7, (-, \mathtt{f20'})) = 6,\\
&q_1(6, (-, \mathtt{f20'})) = 1, && q_1(1, (+, \mathtt{40'})) = 8,\\
&q_1(8, (+, \mathtt{40'})) = 9, && q_1(9, (-, \mathtt{40'})) = 8,\\
&q_1(8, (-, \mathtt{40'})) = 1, && q_1(2, (+, \mathtt{f20'})) = 10,\\
&q_1(10, (-, \mathtt{f20'})) = 2, && q_1(6, (+, \mathtt{e20'})) = 10,\\
&q_1(10, (-, \mathtt{e20'})) = 6, && q_1(3, (+, \mathtt{f20'})) = 11,\\
&q_1(11, (-, \mathtt{f20'})) = 3, && q_1(10, (+, \mathtt{e20'})) = 11,\\
&q_1(11, (-, \mathtt{e20'})) = 10, && q_1(2, (+, \mathtt{40'})) = 12,\\
&q_1(12, (-, \mathtt{40'})) = 2, && q_1(8, (+, \mathtt{e20'})) = 12,\\
&q_1(12, (-, \mathtt{e20'})) = 8, && q_1(3, (+, \mathtt{40'})) = 13,\\
&q_1(13, (-, \mathtt{40'})) = 3, && q_1(12, (+, \mathtt{e20'})) = 13,\\
&q_1(13, (-, \mathtt{e20'})) = 12, && q_1(6, (+, \mathtt{40'})) = 14,\\
&q_1(14, (-, \mathtt{40'})) = 6, && q_1(8, (+, \mathtt{f20'})) = 14,\\
&q_1(14, (-, \mathtt{f20'})) = 8, && q_1(1, \mathtt{end}) = 15.
\end{aligned}
$$

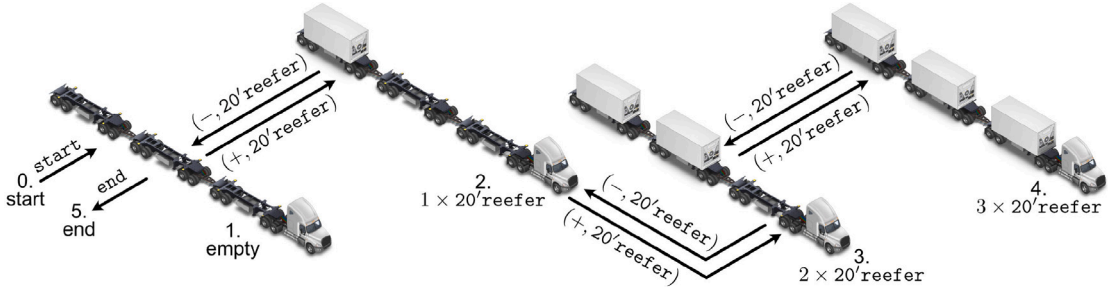


Fig. 3. State transition graph for triple trailer combinations that haul 20 ft reefer containers.

By defining appropriate state transition graphs, we can model different truck types, with specific compatible containers and load configurations. For example, Fig. 3 shows the state transition graph for triple trailer combinations compatible with 20 ft reefer containers.

As the previous examples show, the trucks' state transition graphs implicitly model load capacity and compatibility constraints. Load capacity constraints are enforced because the states/transitions define how many containers a truck can carry, depending on the containers' sizes/contents. In addition, the states/transitions forbid empty trucks from performing deliveries. Compatibility constraints between trucks and containers are enforced because only trucks with compatible states can haul certain containers. In addition, compatibility constraints between containers can be modeled such that trucks do not move incompatible containers together.

3.3. Example

For illustration purposes, Fig. 4 shows a solution to a DRP-HCL instance indicating how trucks transition between states as they pick up and deliver containers. This instance involves movements of 20/40 ft dry containers and 20 ft reefer containers. The time units are minutes, and the start and end depots are represented by nodes situated at the same location. The set of truck types is $\mathcal{F} = \{1, 2\}$, where 1 represents turnpike doubles (Fig. 2) and 2 represents triple trailer combinations (Fig. 3). The trucks' fixed and routing costs are the following: $E^1 = 79$, $E^2 = 88$, and $c^1_{i,j} = c^2_{i,j} = 1.9\tau_{i,j}, (i,j) \in \mathcal{A}$. The fixed costs correspond to daily costs and the routing costs are given on a per-minute basis. All costs are in USD and are calculated as described in Online Appendix 1.

4. ALNS heuristic for the DRP-HCL

ALNS is a search algorithm proposed by Ropke and Pisinger (2006) that extends the large neighborhood search (LNS) algorithm proposed by Shaw (1998). LNS starts from an initial solution from which it creates new solutions by iteratively applying a destruction operator followed by a repair operator: The destruction operator takes a complete solution and removes a part of it, rendering the solution incomplete. The repair operator takes an incomplete solution and reconstructs it into a complete solution. ALNS extends LNS by considering a collection of destruction and repair operators, which are selected based on their previous performances.

Algorithm 1 describes the general framework of ALNS. This procedure takes as input an initial solution, S_{initial}, and a stopping criterion, *stop*. Line 2 initializes the best and current solutions. The loop on lines 3–10 performs the search. Line 4 selects a destruction operator, op_{destroy}, and a repair operator, op_{repair}. Next, line 5 creates a candidate solution, $S_{\text{candidate}}$, by applying the selected destruction operator followed by the selected repair operator. Line 7 updates the best and current solutions whenever a new best solution is found. Alternatively, the current solution may be updated by replacing it with an alternative solution, seeking to modify the search trajectory (line 9). Line 10 updates the scores considered in the selection of the destruction and repair operators. Finally, line 11 returns the best solution found.

Algorithm 1 General framework of ALNS

```
1: function ALNS(S_initial, stop)
2:     S_best ← S_initial, S_current ← S_initial
3:     while not stop do
4:         op_destroy, op_repair ← SELECT-OPS()
5:         S_candidate ← DESTROY-AND-REPAIR(S_current, op_destroy, op_repair)
6:         if COST(S_candidate) < COST(S_best) then
7:             S_best ← S_candidate, S_current ← S_candidate
8:         else if S_current meets update criteria then
9:             S_current ← S_alternative
10:        ASSIGN-SCORES(op_destroy, op_repair)
11:    return S_best
```

In the following, we present our ALNS heuristic for the DRP-HCL. Section 4.1 describes the destruction operators. Section 4.2 describes a repair operator. Section 4.3 describes the scoring and selection of the destruction and repair operators. Section 4.4 describes the acceptance criterion, which determines how the current solution is updated throughout the search. Finally, Section 4.5 presents the main body of the ALNS heuristic.

4.1. Destruction operators

This section introduces the destruction operators used in our ALNS heuristic. First, we discuss the ideas of compatibility sets and similarity measures in Section 4.1.1. We use these ideas to develop two destruction operators, namely Shaw removal, discussed in Section 4.1.2, and spatial proximity removal, discussed in Section 4.1.3. We also use a random destruction operator, which is discussed in Section 4.1.4.

4.1.1. Compatibility sets and similarity measures

As Shaw (1998) points out, when destroying a solution, it is convenient to remove similar requests, as such requests may be easily exchanged when reinserting them into the solution, which can eventually lead to cost savings. By contrast, if very different requests are removed, they may end up being reinserted into their original locations, as they are not interchangeable, and no cost savings are obtained.

In the DRP-HCL, there can be requests that are not at all compatible. For example, some requests may be compatible with specific truck types. These requests cannot be exchanged with requests incompatible with such truck types. To avoid removing requests that are not at all compatible, we define the idea of "compatibility sets". Each compatibility set contains requests that can eventually be served by the same truck, *i.e.*, there is at least one truck type compatible with all the requests in a compatibility set. Thus, we can assume that the requests belonging to a compatibility set are similar regarding truck compatibility (and thus they can be removed together). To define the compatibility sets associated with a problem instance, we establish, for each truck type $f \in \mathcal{F}$, a set with all its compatible requests (we

Node data

i	r_i	e_i	s_i	l_i
0	start	0	10	400
1	(+,e20′)	30	10	60
2	(+,e20′)	50	10	80
3	(+,40′)	80	10	110
4	(+,f20′)	170	10	200
5	(+,20′reefer)	30	10	70
6	(+,20′reefer)	80	10	110
7	(+,20′reefer)	110	10	140
8	(−,e20′)	280	10	340
9	(−,e20′)	210	10	240
10	(−,40′)	120	10	150
11	(−,f20′)	220	10	270
12	(−,20′reefer)	210	10	250
13	(−,20′reefer)	150	10	180
14	(−,20′reefer)	180	10	210
15	end	0	10	400

Solution cost

	Cost (USD)		
	Fixed	Routing	F + R
Route 1 (turnpike double)	79.0	532.0	611.0
Route 2 (triple trailer)	88.0	399.0	487.0
Total	167.0	931.0	1,098.0

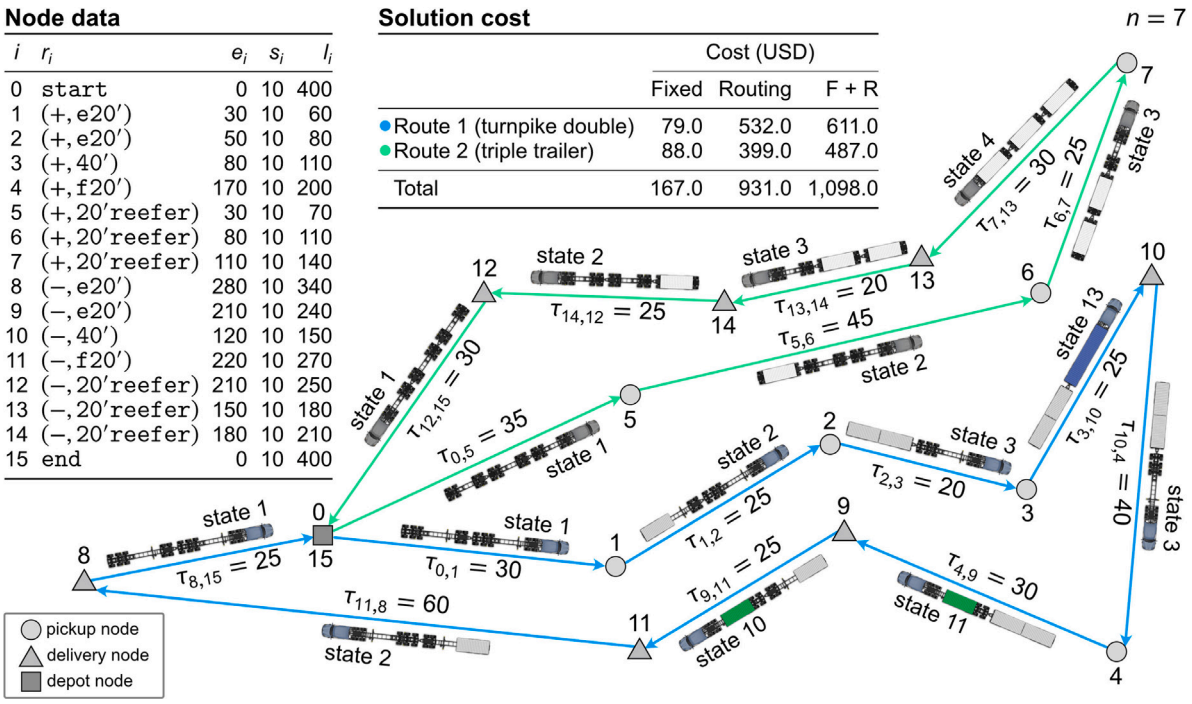


Fig. 4. Example of a solution to a DRP-HCL instance.

only consider sets containing at least two requests). If two or more truck types are compatible with the same requests, we define only one compatibility set associated with such truck types.

The idea is to remove requests belonging to the same compatibility set, as we know that such requests are similar to some extent. However, note that the requests belonging to the same compatibility set can still be differentiated based on the similarity of their containers (according to truck load capacity and compatibility constraints) and their spatial/temporal characteristics.

First, let us discuss how two requests i and j, belonging to the same compatibility set, can be differentiated based on the similarity of their containers. Note that two requests i and j are fully interchangeable, regarding their containers, if container types $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ are compatible with the same truck types, and each of these truck types can carry the same number of v_i containers as v_j containers. On the other hand, requests i and j differ if v_i and v_j containers are not compatible with the same truck types, or if there are truck types compatible with both v_i and v_j containers, but that can carry a different number of v_i containers than v_j containers.

To quantify the extent to which two requests i and j, belonging to the same compatibility set, differ based on their containers, we define a *container dissimilarity measure*. Let $Q_f(v_i)$ be the number of v_i containers a truck of type $f \in \mathcal{F}$ can carry at a time (if truck type f is not compatible with v_i containers, then $Q_f(v_i) = 0$). For a truck type f compatible with v_i or v_j containers, *i.e.*, $Q_f(v_i) > 0$ or $Q_f(v_j) > 0$, we define the difference in load capacity of truck type f regarding requests i and j as

$$J_f(i,j) = \frac{|Q_f(v_i) - Q_f(v_j)|}{\max\{Q_f(v_i), Q_f(v_j)\}}. \tag{1}$$

Note that $J_f(i,j) = 0$ when $Q_f(v_i) = Q_f(v_j)$; $0 < J_f(i,j) < 1$ when $Q_f(v_i) \neq Q_f(v_j)$, with $Q_f(v_i) > 0$ and $Q_f(v_j) > 0$; and $J_f(i,j) = 1$ when $Q_f(v_i) = 0$ and $Q_f(v_j) > 0$ or $Q_f(v_i) > 0$ and $Q_f(v_j) = 0$.

Let $\mathcal{F}_i \subseteq \mathcal{F}$ be the set of truck types compatible with a request i. The following function defines the container dissimilarity between two requests i and j, belonging to the same compatibility set:

$$\delta_c(i,j) = \frac{1}{|\mathcal{F}_i \cup \mathcal{F}_j|} \sum_{f \in \mathcal{F}_i \cup \mathcal{F}_j} J_f(i,j). \tag{2}$$

Note that $0 \leq \delta_c(i,j) \leq 1$. The smaller the value of $\delta_c(i,j)$, the more similar requests i and j are regarding truck load capacity and truck compatibility. The largest the value of $\delta_c(i,j)$, the more different requests i and j are regarding truck load capacity and truck compatibility.

We have developed an approach to measure the dissimilarity between requests based on their associated containers, while preventing incompatible requests from being removed together. This is a development specific to the DRP-HCL, as other approaches to measure request similarity in routing problems are limited to compare quantities of goods demanded, which are continuous values, without factoring in the vehicles' load capacities and without ensuring that incompatible requests are not removed together (if compatibility constraints are involved); see, *e.g.*, Ropke and Pisinger (2006), François et al. (2016), and Friedrich and Elbert (2022).

Now, we define the dissimilarity between requests based on their spatial/temporal characteristics. For this part, we consider the relatedness measure described by Ropke and Pisinger (2006) and define the spatial/temporal dissimilarity between two requests i and j as follows: Let $p(i)$ and $d(i)$ represent the pickup and delivery locations of a request i, respectively. The spatial dissimilarity between two requests i and j is given by

$$\delta_s(i,j) = \tau_{p(i),p(j)} + \tau_{d(i),d(j)}, \tag{3}$$

where $\tau_{i,j}$ represents the travel time between nodes i and j. Meanwhile, the temporal dissimilarity between two requests i and j is given by

$$\delta_t(i,j) = |B_{p(i)} - B_{p(j)}| + |B_{d(i)} - B_{d(j)}|, \tag{4}$$

where B_i represents the time at which service starts at node i in the solution being considered.

4.1.2. Shaw removal

This destruction operator is based on the Shaw removal heuristic of Ropke and Pisinger (2006). The idea of this operator is to remove requests belonging to the same compatibility set by considering the spatial, temporal, and container dissimilarity measures previously defined.

Specifically, the dissimilarity between two requests i and j, belonging to the same compatibility set, is calculated as follows:

$$\delta(i,j) = \mathrm{w}_1 \cdot \delta_{\mathrm{s}}(i,j) + \mathrm{w}_2 \cdot \delta_{\mathrm{t}}(i,j) + \mathrm{w}_3 \cdot \delta_{\mathrm{c}}(i,j). \quad (5)$$

In the above dissimilarity measure, w_1, w_2, and w_3 are non-negative weights. In addition, the values of $\delta_{\mathrm{s}}(i,j)$ and $\delta_{\mathrm{t}}(i,j)$ are normalized such that $0 \leq \delta(i,j) \leq \mathrm{w}_1 + \mathrm{w}_2 + \mathrm{w}_3$. The lower the value of $\delta(i,j)$, the more similar requests i and j are.

Algorithm 2 presents the pseudocode for the Shaw removal operator. This procedure takes as input a solution, S; a fraction of requests to remove, $0 \leq \phi_{\mathrm{s}} \leq 1$; and the weights used to calculate the dissimilarity between requests, $\mathrm{w}_1 \geq 0, \mathrm{w}_2 \geq 0,$ and $\mathrm{w}_3 \geq 0$. Line 2 randomly selects a compatibility set $\mathcal{L}_{\text{compatible}}$. Line 3 randomly selects a request ℓ from the chosen compatibility set. The set of requests to remove is initialized with request ℓ in line 4. The loop on lines 5–9 fills the set with other requests to remove. At each iteration, a request ℓ' is randomly selected from the set of requests to remove. Then, from the unselected requests in compatibility set $\mathcal{L}_{\text{compatible}}$, a new request to be removed, ℓ^*, is selected based on its dissimilarity with respect to request ℓ'. Finally, line 10 removes the selected requests from solution S.

4.1.3. Spatial proximity removal

This destruction operator is based on the adjacent string removal operator proposed by Christiaens and Vanden Berghe (2020). The main idea of this procedure is to remove spatially close requests, belonging to close routes, considering as stopping criterion a number of routes affected by the removals. In this case, the requests to remove belong to the same compatibility set, and the spatial distance between them is calculated by using Eq. (3).

Algorithm 3 shows the pseudocode for the spatial proximity removal operator. This procedure takes as input a solution, S, and a fraction of routes to affect, $0 \leq \phi_{\mathrm{sp}} \leq 1$. Line 2 randomly selects a compatibility set $\mathcal{L}_{\text{compatible}}$. Line 3 defines $\mathcal{H}_{\text{compatible}}$, the set of routes visiting the requests of the chosen compatibility set. In line 4, a seed request, ℓ, is randomly selected from the chosen compatibility set. Line 5 defines a set of requests to remove, $\mathcal{L}_{\text{remove}}$, and a set of affected routes, $\mathcal{H}_{\text{affected}}$. Line 6 establishes the number of routes that will be affected, *num-routes*. The loop on lines 7–14 selects the remaining requests to remove considering *num-routes* as stopping criterion. For a request $\ell \in \mathcal{L}_{\text{compatible}}$, function Adjacent returns a list of requests $\ell' \in \mathcal{L}_{\text{compatible}}$ ordered in ascending order according to their spatial distance to request ℓ. The first element of said list is request ℓ itself. At each iteration, the loop on lines 7–14 selects a request ℓ' from the list returned by function Adjacent, identifies the route that serves said request, ℓ'_{route}, and checks if the request and its route have already been considered. If not, a number of compatible requests equal to *num-req* is selected to be removed from route ℓ'_{route}. These requests include request ℓ' and other requests selected using weighted random selection, with higher probability for the requests closer to request ℓ'. This selection is performed by function Choose on line 13. In line 15, the selected requests are removed from the solution.

Algorithm 2 Shaw removal for the DRP-HCL

1: **function** Shaw-Removal($S, \phi_{\mathrm{s}}, \mathrm{w}_1, \mathrm{w}_2, \mathrm{w}_3$)
2: $\quad\mathcal{L}_{\text{compatible}} \leftarrow$ Random-Choice(Compatibility-Sets(S))
3: $\quad\ell \leftarrow$ Random-Choice($\mathcal{L}_{\text{compatible}}$)
4: $\quad\mathcal{L}_{\text{remove}} \leftarrow \{\ell\}$
5: $\quad$**while** $|\mathcal{L}_{\text{remove}}| <$ Ceil($\phi_{\mathrm{s}} \cdot |\mathcal{L}_{\text{compatible}}|$) **do**
6: $\quad\quad\ell' \leftarrow$ Random-Choice($\mathcal{L}_{\text{remove}}$)
7: $\quad\quad\mathcal{L}_{\text{remaining}} \leftarrow \mathcal{L}_{\text{compatible}} \setminus \mathcal{L}_{\text{remove}}$
8: $\quad\quad$choose a request $\ell^* \in \mathcal{L}_{\text{remaining}}$ using weighted random selection with probability proportional to $\delta(\ell', \ell'')^{-1}$ for each request $\ell'' \in \mathcal{L}_{\text{remaining}}$; the dissimilarity is computed with w_1, w_2, and w_3
9: $\quad\quad\mathcal{L}_{\text{remove}} \leftarrow \mathcal{L}_{\text{remove}} \cup \{\ell^*\}$
10: $\quad$Remove-Requests($S, \mathcal{L}_{\text{remove}}$)

Algorithm 3 Spatial proximity removal for the DRP-HCL

1: **function** Spatial-Proximity-Removal(S, ϕ_{sp})
2: $\quad\mathcal{L}_{\text{compatible}} \leftarrow$ Random-Choice(Compatibility-Sets(S))
3: $\quad\mathcal{H}_{\text{compatible}} \leftarrow$ Filter-Routes($S, \mathcal{L}_{\text{compatible}}$)
4: $\quad\ell \leftarrow$ Random-Choice($\mathcal{L}_{\text{compatible}}$)
5: $\quad\mathcal{L}_{\text{remove}} \leftarrow \{\}, \mathcal{H}_{\text{affected}} \leftarrow \{\}$
6: $\quad$*num-routes* $\leftarrow$ Ceil($\phi_{\mathrm{sp}} \cdot |\mathcal{H}_{\text{compatible}}|$)
7: $\quad$**for** $\ell' \in$ Adjacent($\ell, \mathcal{L}_{\text{compatible}}$) **do**
8: $\quad\quad$**if** $|\mathcal{H}_{\text{affected}}| <$ *num-routes* **then**
9: $\quad\quad\quad\ell'_{\text{route}} \leftarrow$ Get-Route($\ell', \mathcal{H}_{\text{compatible}}$)
10: $\quad\quad\quad$**if** $\ell' \notin \mathcal{L}_{\text{remove}}$ **and** $\ell'_{\text{route}} \notin \mathcal{H}_{\text{affected}}$ **then**
11: $\quad\quad\quad\quad$*total-req* $\leftarrow$ Count-Requests($\ell'_{\text{route}}, \mathcal{L}_{\text{compatible}}$)
12: $\quad\quad\quad\quad$*num-req* $\leftarrow$ Discrete-Uniform-Random(1, *total-req*)
13: $\quad\quad\quad\quad\mathcal{L}_{\text{remove}} \leftarrow \mathcal{L}_{\text{remove}} \cup$ Choose($\ell'_{\text{route}}, \ell'$, *num-req*, $\mathcal{L}_{\text{compatible}}$)
14: $\quad\quad\quad\quad\mathcal{H}_{\text{affected}} \leftarrow \mathcal{H}_{\text{affected}} \cup \{\ell'_{\text{route}}\}$
15: $\quad$Remove-Requests($S, \mathcal{L}_{\text{remove}}$)

4.1.4. Random removal

This destruction operator randomly selects requests from a solution and removes them regardless of their similarity, as shown in Algorithm 4. This procedure takes as input a solution, S, and a fraction of requests to remove, $0 \leq \phi_{\mathrm{r}} \leq 1$. Line 2 sets the number of requests to remove, *num-req*. Line 3 randomly selects a number of requests equal to *num-req* from the solution. Finally, line 4 removes the selected requests from the solution.

Algorithm 4 Random removal for the DRP-HCL

1: **function** Random-Removal(S, ϕ_{r})
2: $\quad$*num-req* $\leftarrow$ Ceil($\phi_{\mathrm{r}} \cdot |$Requests($S$)$|$)
3: $\quad\mathcal{L}_{\text{remove}} \leftarrow$ Random-Choice(Requests(S), *num-req*)
4: $\quad$Remove-Requests($S, \mathcal{L}_{\text{remove}}$)

4.2. Repair operator: random regret insertion

This operator is based on the regret insertion procedure described by Ropke and Pisinger (2006). The main idea of this procedure is to prioritize the available insertions that, if not implemented, would lead to the greatest regret. The proposed operator introduces two sources of randomness. First, some insertions are randomly omitted, similarly as done in the greedy insertion with blinks procedure described by Christiaens and Vanden Berghe (2020). Second, when it is necessary to create a new route to serve a request, there is a randomness component in the selection of the truck type assigned to the new route.

Algorithm 5 shows the pseudocode for the random regret insertion operator, defined by function Random-Regret. This function takes as input an incomplete solution, S; the number of insertions considered in calculating the regret of an insertion, $\kappa \geq 0$; and a randomness factor for the insertions, $0 \leq \varsigma \leq 1$. The loop on lines 24–32 inserts the requests missing in solution S until no unrouted request remains. Line 25 finds all routes compatible with the requests yet to be routed. Line 26 finds map *ins*, which maps each request $\ell \in$ Unrouted(S) to a list of valid insertions, where each insertion corresponds to the best insertion in one of the routes compatible with request ℓ or the creation of a new route. We detail the operation of function Compute-Insertions below. Line 27 finds set $\mathcal{I}$, which contains the unrouted requests for which the regret cannot be calculated due to the lack of enough valid insertions. If set $\mathcal{I}$ is not empty, line 29 selects the request from this set with the fewest valid insertions and the lowest insertion cost (these selection criteria are considered in lexicographic order). On the other hand, when set $\mathcal{I}$ is empty, *i.e.*, it is possible to calculate the regret for all unrouted requests, line 31 selects the unrouted request with the largest regret (the regret function is defined by ε_1), and the lowest insertion cost (these selection criteria are also considered in lexicographic order). When $\kappa = 0$, no regret is calculated—the request to insert is the

Algorithm 5 Random regret insertion for the DRP-HCL

1: **function** COMPUTE-INSERTIONS($\mathcal{L}_{\text{unrouted}}, \mathcal{H}_{\text{compatible}}, \varsigma$)
2: let ins be a map mapping each request $\ell \in \mathcal{L}_{\text{unrouted}}$ to a list of insertions
3: **for** each request $\ell \in \mathcal{L}_{\text{unrouted}}$ **do**
4: let $\ell_{\text{insertions}}$ be a new list
5: **for** each route $\ell_{\text{route}} \in \{$ $g \in \mathcal{H}_{\text{compatible}}$: route g is compatible with request ℓ $\}$ **do**
6: $i_{\text{best}} \leftarrow$ NIL
7: **for** $i \in$ VALID-INSERTIONS($\ell, \ell_{\text{route}}$) **do**
8: **if** CONTINUOUS-UNIFORM-RANDOM(0.0, 1.0) $< \varsigma$ **then**
9: **if** $i_{\text{best}} ==$ NIL **or** COST(i) < COST(i_{best}) **then**
10: $i_{\text{best}} \leftarrow i$
11: add i_{best} to list $\ell_{\text{insertions}}$
12: $\mathcal{F}_{\text{compatible}} \leftarrow$ COMPATIBLE-TRUCK-TYPES(ℓ)
13: **if** $|\mathcal{F}_{\text{compatible}}| == 1$ **then**
14: let $f_{\text{new}} \in \mathcal{F}_{\text{compatible}}$ be the only compatible truck type
15: **else**
16: randomly select an integer $k \in [1, |\mathcal{F}_{\text{compatible}}|]$ with probability proportional to its value
17: let $f_{\text{new}} \in \mathcal{F}_{\text{compatible}}$ be the truck type with the k^{th} largest value of $\chi(f) = (\chi_1(f), \chi_2(f), \chi_3(f))$, with $\chi_1(f) =$ COMPATIBILITY($f, \mathcal{L}_{\text{unrouted}}$), $\chi_2(f) =$ LOAD-CAPACITY($f, \mathcal{L}_{\text{unrouted}}$), and $\chi_3(f) = -$COST(f, ℓ)
18: define an insertion i_{new}, which indicates the creation of a new route assigned to a truck of type f_{new} to serve request ℓ
19: add insertion i_{new} to list $\ell_{\text{insertions}}$
20: sort list $\ell_{\text{insertions}}$ by ascending insertion cost, NIL insertions have cost ∞
21: insert list $\ell_{\text{insertions}}$ into map ins with key ℓ
22: **return** ins
23: **function** RANDOM-REGRET(S, κ, ς)
24: **while** |UNROUTED(S)| > 0 **do**
25: $\mathcal{H}_{\text{compatible}} \leftarrow$ COMPATIBLE-ROUTES(UNROUTED(S))
26: $ins \leftarrow$ COMPUTE-INSERTIONS(UNROUTED(S), $\mathcal{H}_{\text{compatible}}, \varsigma$)
27: $\mathcal{I} \leftarrow \{$ $\ell \in$ UNROUTED(S) : there are fewer than $\kappa + 1$ insertions j in $ins[\ell]$ with COST(j) < ∞ $\}$
28: **if** $|\mathcal{I}| > 0$ **then**
29: let $\ell^* \in \mathcal{I}$ be the request that minimizes $\vartheta(\ell) = (\vartheta_1(\ell), \vartheta_2(\ell))$, with $\vartheta_1(\ell) = ins[\ell].count$(insertions j with COST(j) < ∞) and $\vartheta_2(\ell) =$ COST($ins[\ell][0]$)
30: **else**
31: let $\ell^* \in$ UNROUTED(S) be the request that maximizes $\epsilon(\ell) = (\epsilon_1(\ell), \epsilon_2(\ell))$, with $\epsilon_1(\ell) = \sum_{j=0}^{\kappa}$[COST($ins[\ell][j]$) − COST($ins[\ell][0]$)] and $\epsilon_2(\ell) = -$COST($ins[\ell][0]$)
32: INSERT-REQUEST($S, ins[\ell^*][0]$)

one with the lowest insertion cost. On the other hand, when $\kappa \geq 1$, the regret function calculates the cost difference between a request's best insertion ($ins[\ell][0]$) and its κ next best insertions ($ins[\ell][j]$, for $j = 1, \ldots, \kappa$)—the idea is that we would regret not inserting the request whose best insertion produces the most cost savings with respect to its κ next best insertions. Finally, the request selected in either line 29 or 31 is inserted in line 32.

Now we delve into the operation of function COMPUTE-INSERTIONS, used in line 26. This function takes as input a set of unrouted requests, $\mathcal{L}_{\text{unrouted}}$; a set of routes compatible with the unrouted requests, $\mathcal{H}_{\text{compatible}}$; and the parameter that controls the randomness of the insertions, ς. The output of this function is a map, ins, mapping each unrouted request to a list of its possible insertions, sorted in ascending order based on their costs.

Map ins is initialized in line 2 and populated by the loop on lines 3–21. For each request ℓ, the best insertion on each of its compatible routes is calculated by the loop on lines 5–11, where function VALID-INSERTIONS calculates all feasible insertions of a request ℓ on a route ℓ_{route}. An insertion is feasible if it meets precedence, time-window, load capacity, and compatibility constraints. In particular, we check load capacity and compatibility constraints as follows: Assume that a request ℓ is inserted into a route $\ell_{\text{route}} = (a_0, a_1, \ldots, a_M)$, where $a_0 = 0$ and $a_M = 2n + 1$. Route ℓ_{route} is traveled by a type f truck. The truck is in the empty state at node a_0. To check route ℓ_{route} against load capacity and compatibility constraints, we check if, for nodes a_{k-1} and $a_k, k = 1, \ldots, M$, there are corresponding states $h_{a_{k-1}}, h_{a_k} \in \Theta_f$ such that $q_f(h_{a_{k-1}}, r_{a_k}) = h_{a_k}$, with h_{a_M} being the truck's final state. The route is feasible regarding load capacity and compatibility constraints only if this condition holds. Note that some insertions on a route ℓ_{route} may be omitted based on the parameter ς. Note also that the output of function VALID-INSERTIONS can be cached to speed up the solution process.

The insertion of each request ℓ in a new route is also considered (lines 12–19): First, a set of truck types compatible with request ℓ is defined in line 12. If the request is compatible with only one truck type, such a truck type is selected to create the new route. Otherwise, when there are two or more truck types compatible with the request, the truck type of the new route is selected using weighted random selection, with higher probability for the truck types compatible with more unrouted requests, with larger cargo capacity for moving the containers associated with the unrouted requests, and with lower fixed and routing costs. These criteria are considered in lexicographic order (line 17). We prioritize trucks compatible with more unrouted requests and with larger cargo capacity because these trucks can enable more efficient routing, even if they have higher fixed and routing costs.

Once the list of insertions of a request ℓ has been filled, line 20 sorts the list by ascending insertion cost. Then, line 21 adds the list to map ins. Finally, line 22 returns map ins.

4.3. Operator scoring and selection

The mechanism used to score and select the destruction and repair operators follows the ideas of the original ALNS algorithm (Ropke and Pisinger, 2006). Given a pool of possible destruction operators (in this case, Shaw, spatial proximity, and random removal) and repair operators (in this case, random regret insertion with $\kappa = 0, 1, 2, 3$), the operators to be applied at each iteration are chosen randomly with a bias given to the best performing ones (at the start of the search, no bias exists). The selection weights associated with operator choice are built iteratively, given past success. More specifically, a score to each operator is assigned according to the quality of solution $S_{\text{candidate}}$ with respect to the current best solution S_{best}. The idea is to reward the operators that find better solutions, so that they are more likely to be selected in later iterations.

Let $\alpha_3 \geq \alpha_2 \geq \alpha_1$ be the possible scores to assign to an operator. Also, let COST(S) be the cost of a solution S, and let thr be an acceptance threshold. If COST($S_{\text{candidate}}$) < COST(S_{best}), the operators are assigned a score α_3. Else, if $\frac{\text{COST}(S_{\text{candidate}}) - \text{COST}(S_{\text{best}})}{\text{COST}(S_{\text{candidate}})} < thr$, the operators are assigned a score α_2. Otherwise, the operators are assigned a score α_1.

The operators' accumulated scores are used to update the selection weights after every γ solutions are found. The scores are then reset to zero for the next series of γ solutions. To describe how the weights are updated, we define the following elements:

- γ_i: the ith series of γ solutions found during the search.
- ω_i^{op}: the weight of operator op during the series of solutions γ_i.
- $\hat{\alpha}_i^{op}$: the total score that operator op obtained in creating the series of solutions γ_i.
- μ_i^{op}: the number of times that operator op was used to create the series of solutions γ_i.
- $\bar{\alpha}_i^{op} = \hat{\alpha}_i^{op} / \mu_i^{op}$: the mean score of operator op in the series of solutions γ_i ($\bar{\alpha}_i^{op} = 0$ if $\mu_i^{op} = 0$).
- $0 \leq \beta \leq 1$: a forgetting factor that decides the extent to which the previous weight affects the value of the new weight.

The weight of operator op for the series of solutions γ_{i+1} is given by $\omega_{i+1}^{op} = \omega_i^{op}(1 - \beta) + \beta \bar{\alpha}_i^{op}$.

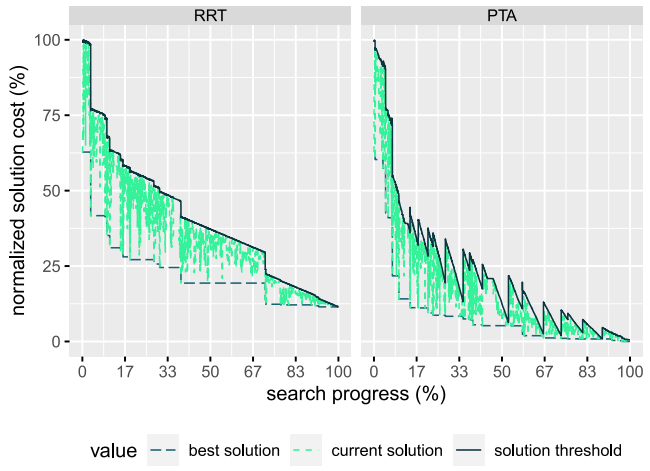


Fig. 5. ALNS using RRT vs. PTA.

4.4. Acceptance criterion: PTA

As any path-based metaheuristic, the proposed method works by moving from a current solution to neighbor solutions (which in this case are found with the destruction and repair operators). To determine how the search moves from a current solution to another, we propose the PTA criterion, a new general threshold-based acceptance criterion built on the following principles:

1. The search always moves to any new best solution (as in the general framework of ALNS described in Algorithm 1).
2. The search moves to neighbor solutions that are within the acceptance threshold (thr) of the current best solution; this is a means to escape local optima.
3. The acceptance threshold decreases steadily as the search consecutively finds solutions that do not improve upon the best solution—we refer to these solutions as "non-improving solutions".
4. When a new best solution is found or a maximum number of consecutive non-improving solutions is reached, the search is "shaken" by increasing the acceptance threshold. The margin by which the acceptance threshold increases is reduced as the search progresses.
5. If a solution with a cost within the acceptance threshold is not found, the search continues from the current best solution.

The name of the proposed acceptance criterion, pulsed threshold acceptance, stems from the fact that the value of the acceptance threshold increases and decreases repeatedly over time, resembling a pulsed signal.

PTA is based on RRT, one of the acceptance criteria for ALNS reviewed by Santini et al. (2018). RRT also seeks to escape local optima by accepting newly found solutions if their costs are within an acceptance threshold of the current best solution. This threshold is constantly reduced as the search progresses (in this case, we consider the version of RRT where the threshold linearly decreases until it reaches zero). Santini et al. (2018) recommended the use of RRT, as it commonly outperformed the other criteria they reviewed. However, RRT has the following drawbacks:

1. The search can be prone to long periods of stagnation when the acceptance threshold is high, as the pressure for improvement is low. This issue is only corrected in late stages of the search, as the acceptance threshold decreases. Note that one could set a small acceptance threshold from the beginning of the search, but this would reduce the ability of the algorithm to explore the search space.
2. Opportunities to make improvements directly on the current best solution may be missed because, after finding such a solution, the search may continue exploring other (worse) solutions, and the current best solution may never be found again.

PTA amends these issues. As the acceptance threshold alternates between high and low values, pressure for improvement is induced throughout the whole search, not only at its latest stages. Furthermore, the search is forced to revisit the current best solution when a newly found solution has a cost exceeding the acceptance threshold. Fig. 5 exemplifies these aspects. This figure shows the solutions that two ALNS procedures found for a sample DRP-HCL instance. All configurations used to generate both figures were the same, with the only difference being the use of RRT or PTA.

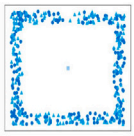
4.5. Main body of the proposed ALNS heuristic

Algorithm 6 describes the proposed ALNS heuristic that incorporates the PTA criterion. From now on, we refer to this heuristic as ALNS-PTA. The inputs of the heuristic are an initial solution, S_{initial}; a maximum runtime value, $runtime_{\max}$; a maximum number of consecutive non-improving solutions after which the search is shaken, $cnis_{\max}$; and a maximum acceptance threshold, $thr_{\max}$. In this case, S_{initial} is a naïve solution composed of n different routes, each serving a single request (the type of the truck assigned to each route is selected at random when there are two or more truck types compatible with the route).

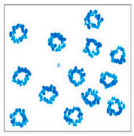
Line 2 initializes the best and current solutions, line 3 initializes the acceptance threshold (thr), line 4 initializes the counter of consecutive non-improving solutions ($cnis$), and line 5 starts recording the procedure's runtime. The loop on lines 6–25 performs the search using the PTA criterion; the search is conducted until the maximum runtime is reached. Line 7 selects a destruction operator, op_{destroy}, and a repair operator, op_{repair}, using weighted random selection, as previously described. Line 8 creates a candidate solution, $S_{\text{candidate}}$, by applying operator op_{destroy} followed by operator op_{repair} to the current solution. In line 9, the cost of the best solution is saved; this value is used later to update the values of the acceptance threshold and the $cnis$

Algorithm 6 ALNS-PTA

1: **function** ALNS-PTA($S_{\text{initial}}, runtime_{\max}, cnis_{\max}, thr_{\max}$)
2: $S_{\text{best}} \leftarrow S_{\text{initial}}, S_{\text{current}} \leftarrow S_{\text{initial}}$
3: $thr \leftarrow thr_{\max}$
4: $cnis \leftarrow 0$
5: $runtime \leftarrow$ NOW()
6: **while** $runtime.elapsed() < runtime_{\max}$ **do**
7: $op_{\text{destroy}}, op_{\text{repair}} \leftarrow$ SELECT-OPS()
8: $S_{\text{candidate}} \leftarrow$ DESTROY-AND-REPAIR($S_{\text{current}}, op_{\text{destroy}}, op_{\text{repair}}$)
9: $prior\text{-}best \leftarrow$ COST(S_{best})
10: **if** COST($S_{\text{candidate}}$) < COST(S_{best}) **then**
11: $S_{\text{best}} \leftarrow S_{\text{candidate}}, S_{\text{current}} \leftarrow S_{\text{candidate}}$
12: **else**
13: **if** $\frac{\text{COST}(S_{\text{candidate}}) - \text{COST}(S_{\text{best}})}{\text{COST}(S_{\text{candidate}})} < thr$ **then**
14: $S_{\text{current}} \leftarrow S_{\text{candidate}}$
15: **if** $\frac{\text{COST}(S_{\text{current}}) - \text{COST}(S_{\text{best}})}{\text{COST}(S_{\text{current}})} > thr$ **then**
16: $S_{\text{current}} \leftarrow S_{\text{best}}$
17: **if** COST(S_{best}) < $prior\text{-}best$ **then**
18: $thr \leftarrow thr_{\max} \cdot \left(1.0 - \frac{\min\{runtime.elapsed(), runtime_{\max}\}}{runtime_{\max}}\right), cnis \leftarrow 0$
19: **else**
20: $cnis \leftarrow cnis + 1$
21: **if** $cnis == cnis_{\max}$ **then**
22: $thr \leftarrow thr_{\max} \cdot \left(1.0 - \frac{\min\{runtime.elapsed(), runtime_{\max}\}}{runtime_{\max}}\right), cnis \leftarrow 0$
23: **else**
24: $thr \leftarrow thr - \frac{thr}{cnis_{\max} - cnis}$
25: ASSIGN-SCORES($op_{\text{destroy}}, op_{\text{repair}}$)
26: **return** S_{best}



C1: The pickup and delivery locations of each request are located on opposite sides of the region. Thus, long distances are expected between the pickup and delivery locations of each request. The depot is located at the center of the region.



C2: The pickup and delivery locations of each request are clustered together, and there is a truck type compatible with all the requests of each cluster. Thus, the distances between pickup and delivery locations are expected to be short. The depot is randomly located.



R: All locations are randomly distributed (including the depot).

Fig. 6. Geographical distributions of locations of benchmark set 1.

counter. Line 11 updates the best and current solutions whenever a new best solution is found. In line 14, a candidate solution is accepted as current solution if its cost is within the acceptance threshold with respect to the best solution. In line 16, the current solution is replaced with the best solution when the cost of the former with respect to the latter is over the acceptance threshold. Whenever a new best solution is found, line 18 increases the acceptance threshold and resets the $cnis$ counter. Otherwise, line 20 increases the $cnis$ counter by one. When the limit of consecutive non-improving solutions is reached, line 22 shakes the search by increasing the acceptance threshold and resetting the $cnis$ counter. While the limit of consecutive non-improving solutions is reached, line 24 gradually reduces the acceptance threshold until it reaches zero. Note that, in lines 18 and 22, the acceptance threshold is increased by progressively smaller amounts as the search progresses. By the end of the search, the amount of increase is close to zero. Line 25 assigns scores to operators op_{destroy} and op_{repair} based on the quality of solution $S_{\text{candidate}}$, as previously discussed. Finally, line 26 returns the best solution found.

5. Computational results

5.1. Hardware and software

We implemented the ALNS-PTA heuristic in the Rust programming language, version 1.65.0. As a baseline for comparison, we considered the results of the branch-and-cut algorithm of Bustos-Coral and Costa (2022), to which we refer to as *fragment-based approach*. We implemented this algorithm using Python 3.9 with the Gurobi 10.0 API, and used it with the same parameter settings specified in Bustos-Coral and Costa (2022). We conducted all experiments on The University of Melbourne's Spartan high-performance computing system, which has a physical partition with 14 nodes featuring Intel Xeon Gold 6154 CPUs @ 3.00 GHz, with 1519 GB RAM per node, and 79 nodes featuring Intel Xeon Gold 6254 CPUs @ 3.10 GHz, with 710 GB RAM per node. We conducted all experiments allocating a maximum of 256 GB RAM. Each experiment run was assigned to one of the available compute nodes. Note that the Intel Xeon Gold 6154 and 6254 have similar specifications and performance (for details, see PassMark Software, 2024).

5.2. Description of the problem instances

To test the ALNS-PTA heuristic, we developed a benchmark set involving standard trucks and LCVs. This benchmark set, to which we refer to as benchmark set 1, is detailed in Section 5.2.1. We also tested the ALNS-PTA heuristic on instances of Bustos-Coral and Costa (2022), which involve only standard trucks. These instances, to which we refer to as benchmark set 2, are detailed in Section 5.2.2. Both benchmark sets are available at https://github.com/dbuscor/drp_hcl_instances.

5.2.1. Benchmark set 1

This benchmark set contains 120 instances featuring different geographical distributions of locations, time-window lengths, and truck fleets. These instances involve 25, 50, 75, and 100 requests.

As to the spatial distribution of locations, the instances are classified into classes C1, C2, and R, as detailed in Fig. 6. We assume all locations to lie in a square area of 60 mi × 60 mi in C1 instances, 90 mi × 90 mi in R instances, and 120 mi × 120 mi in C2 instances. In addition, we assume that, for any arc $(i, j) \in \mathcal{A}$, the travel time $\tau_{i,j}$—given in minutes—is equal to the Euclidean distance between i and j.

Regarding the time windows, the instances are classified into classes W1 and W2. In the former, all pickups and deliveries have 60-minute time windows, while in the latter, all pickups and deliveries have 120-minute time windows. In all cases, the service times are equal to 30 min, and the planning horizon lasts 600 min. The time windows are distributed over time as in the instances of Bustos-Coral and Costa (2022).

The instances involve the truck types described in Table 2 (the trucks' fixed costs are per day and the routing costs are per minute, with all costs being in USD). The modeling of the trucks, including the calculation of the fixed and routing costs, is detailed in Online Appendix 1.

Considering the trucks listed in Table 2, we defined different fleets of trucks and containers, based on which the instances are classified into classes T1, T2, T3, T4, and T5, as detailed in Fig. 7. This figure shows the types of trucks and containers considered in each class, with arrows representing the compatibility between trucks and containers (each arrow is labeled with the maximum number of containers of a specific type that a truck can carry at a time).

Note that, in T1 and T2 instances, there is full truck/container compatibility, as all trucks are compatible with all containers. However, the trucks' load capacities vary between T1 and T2 instances. In T1 instances, trucks have a small cargo capacity, as they can carry only one container at a time. Meanwhile, in T2 instances, trucks have a large cargo capacity, as they can carry up to four containers at a time. Instances T3 feature medium truck/container compatibility. Each truck type is expected to be compatible with about half of the containers, and each container type can be moved by two of the four truck types. The trucks have varying load capacities. While some can only transport a single container at a time, others can carry two, three, or even four containers simultaneously. In T4 and T5 instances, the truck/container compatibility is low, as there are five container types and five truck types, with each truck type being compatible with only one container type. In particular, in T5 instances, we consider containers used to move different classes of hazmat loads. Even if empty, these containers are incompatible and must be kept separate. Finally, note that the trucks' load capacities differ between T4 and T5 instances, with T4 instances featuring trucks with a small cargo capacity and T5 instances featuring trucks with a large cargo capacity.

Table 2
Truck types considered in benchmark set 1.

Truck type	Description	E^f	$c^f_{i,j}, (i,j) \in \mathcal{A}$
5A–20'	Five-axle tractor-semitrailer consisting of a semi-truck and a 20' 2-axle chassis.	18	$1.7\tau_{i,j}$
5A–40'	Five-axle tractor-semitrailer consisting of a semi-truck and a 40' 2-axle chassis.	17	$1.7\tau_{i,j}$
5A-53'	Five-axle tractor-semitrailer consisting of a semi-truck and a 53' 2-axle chassis.	26	$1.7\tau_{i,j}$
6A–20/40'	Six-axle tractor-semitrailer consisting of a semi-truck and a 20/40' 3-axle chassis.	31	$1.7\tau_{i,j}$
TD–20/40'	Turnpike double consisting of a semi-truck and two 20/40' 3-axle chassis.	79	$1.9\tau_{i,j}$
TTC–20'	Triple trailer combination consisting of a semi-truck and three 20' 2-axle chassis.	88	$1.9\tau_{i,j}$

A 20' 2-axle chassis carries one 20 ft container, a 40' 2-axle chassis carries one 40 ft container, a 53' 2-axle chassis carries one 53 ft container, and a 20/40' 3-axle chassis carries one 20/40 ft container or two empty 20 ft containers.

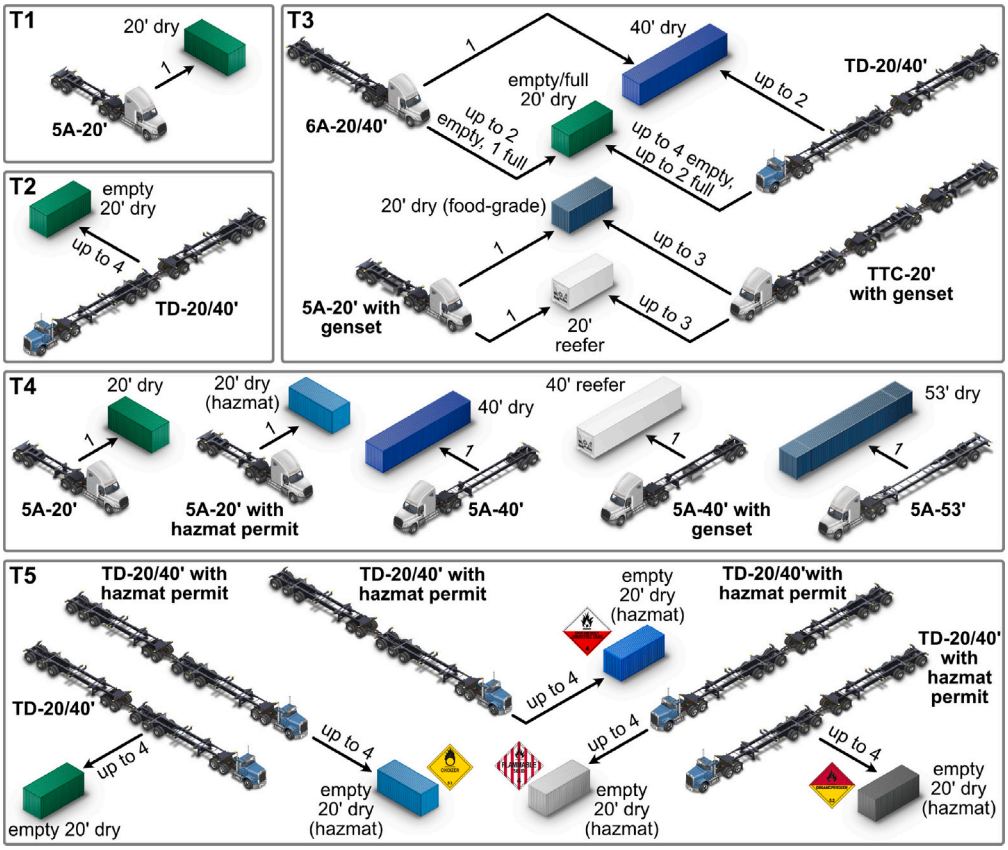


Fig. 7. Description of the fleets considered in benchmark set 1.

5.2.2. Benchmark set 2

This benchmark set is composed of the 100-, 200-, 400-, 600-, 800-, and 1,000-request instances of Bustos-Coral and Costa (2022). In total, there are 450 instances. Similar to the instances of benchmark set 1, the instances of this benchmark set combine different geographical distributions of locations, truck fleets, and time-window lengths.

Regarding the geographical distribution of locations, the instances are classified into classes C1, where the requests have distant pickup and delivery locations; C2, where the requests' pickup and delivery locations are clustered nearby; R, where the requests' pickup and delivery locations are randomly distributed; RC1, a combination of the C1 and R classes; and RC2, a combination of the C2 and R classes.

Regarding the truck fleets, it should be noted that, in these instances, the containers can either be moved only one at a time—which restricts the routing possibilities—or in a 2x20' load configuration—which allows for more flexible routing. In addition, there are different levels of truck/container compatibility (the fraction of containers compatible with a truck type to total containers, averaged across the truck types). Considering these aspects, the instances are classified into classes F1, where truck/container compatibility is very high, and all containers can be moved only one at a time; F2, where truck/container compatibility is very high, and a very high percentage of containers can be moved in a 2x20' load configuration; F3, where truck/container compatibility is high, and a high percentage of containers can be moved in a 2x20' load configuration; F4, where truck/container compatibility is medium, and a low percentage of containers can be moved in a 2x20' load configuration; and F5, where truck/container compatibility is low, and a very low percentage of containers can be moved in a 2x20' load configuration.

Based on the time windows, the instances are classified into classes W1, where the requests' time windows are short; W2, where the requests' time windows are long; and W3, where some requests have short time windows and others have long time windows.

5.3. Parameter tuning

To tune the proposed ALNS heuristic, we used the automatic configuration tool irace (López-Ibáñez et al., 2016). To use this tool, it is necessary to specify a set of parameters with their ranges of values, a set of training instances, and a training budget (*e.g.*, a maximum number of algorithm executions). The irace tool iteratively samples parameter configurations and tests them against the training instances, discarding

Table 3
Results of the parameter tuning process.

	Parameter	Description	Type	Range	Best configurations					
					1	2	3	4	5	6
Core parameters	$thr_{\max}$	Maximum acceptance threshold.	Real	[0.005, 0.05]	0.011	0.007	0.011	0.007	0.009	0.016
	γ	Number of solutions after which the selection weights are updated.	Integer	[10, 100], step 10	30	30	30	40	20	40
	α_3	Best score assigned to the search operators.	Real	[0.0, 10.0]	9.33	9.21	9.13	8.67	9.92	8.86
	α_2	Intermediate score assigned to the search operators.	Real	[0.0, 10.0]	5.052	4.962	7.737	4.482	3.681	7.711
	α_1	Worst score assigned to the search operators.	Real	[0.0, 10.0]	1.148	2.068	1.131	0.382	2.132	0.427
	β	Forgetting factor.	Real	[0.1, 1.0]	0.796	0.706	0.525	0.561	0.490	0.496
Destruction operators	Use random removal				✓	✓	✗	✓	✓	✗
	ϕ_r	Fraction of requests to remove.	Real	[0.05, 0.5]	0.115	0.126	N/A	0.195	0.080	N/A
	Use spatial proximity removal				✓	✓	✗	✓	✓	✗
	ϕ_{sp}	Fraction of routes to affect.	Real	[0.05, 0.5]	0.144	0.116	N/A	0.091	0.195	N/A
	Use Shaw removal				✓	✓	✓	✓	✓	✓
	ϕ_s	Fraction of requests to remove.	Real	[0.05, 0.5]	0.10	0.07	0.13	0.18	0.17	0.14
	w_1	Weight of spatial dissimilarity.	Real	[0.0, 10.0]	7.84	8.06	6.22	7.60	7.33	6.42
	w_2	Weight of temporal dissimilarity.	Real	[0.0, 10.0]	4.94	5.33	0.12	6.39	5.67	0.24
	w_3	Weight of container dissimilarity.	Real	[0.0, 10.0]	1.36	1.63	4.74	3.06	5.53	5.06
Repair operators	Use random regret insertion with $\kappa = 0$ comparisons				✗	✗	✗	✗	✗	✗
	Use random regret insertion with $\kappa = 1$ comparisons				✓	✓	✓	✓	✓	✓
	Use random regret insertion with $\kappa = 2$ comparisons				✗	✗	✗	✗	✗	✗
	Use random regret insertion with $\kappa = 3$ comparisons				✗	✗	✗	✗	✗	✗
	ς	Randomness factor.	Real	[0.9, 1.0]	0.977	0.981	0.989	0.987	0.957	0.987
Accept. criteria	Use PTA or RRT				PTA	PTA	PTA	PTA	PTA	PTA
	$cnis'_{\max}$	Maximum number of consecutive non-improving solutions.	Integer	[120, 1,200], step 120	360	480	360	360	720	240

Note that $cnis_{\max} = 100/n \cdot cnis'_{\max}$; 120 is the least common denominator of the fractions $100/n$ with $n = 25, 50, 75, 100, 200, 400, 600, 800, 1{,}000$ (according to the numbers of requests we consider in the other sets of experiments).

the configurations that are statistically worse than others. In the end, irace returns a list with the best found configurations, ordered by mean performance.

Considering the findings of François et al. (2016), we used the irace tool not only for parameter tuning, but also to decide which destruction/repair operators to include in the search, based on their performances. For each destruction/repair operator, we considered a Boolean parameter indicating the inclusion of the operator in the weighted selection mechanism of the ALNS procedure. The final values of such parameters were determined by the irace tool. As noted by François et al. (2016), this a priori selection of operators complements the role of the ALNS selection scheme and can lead to better results, as destruction/repair operators that are likely to underperform are not even considered in the search, thus preventing possible waste of search time.

In addition to using the irace tool for parameter tuning and an a priori selection of search operators, we also used it to validate the efficacy of the PTA criterion, compared to the RRT acceptance criterion. To this aim, we included a parameter to toggle between PTA and RRT and let irace decide the final value of the parameter, in line with the approach used by Friedrich and Elbert (2022) to compare the performances of different acceptance criteria for ALNS.

To avoid introducing bias in the results, and following the recommendation of François et al. (2016), we used the irace tool with a set of training instances that does not contain any of the instances involved in the other sets of experiments. In this case, we considered 10 training instances, with different numbers of requests (varying between 100 and 1,000), and encompassing all the geographical distributions of locations, time-window lengths, and truck fleets described in Section 5.2.1. We set a training budget of 20,000 algorithm executions, with a maximum runtime (per run) varying between 3 min and 30 min, depending on the number of requests of the instance being solved. The configurations were compared with Friedman tests at a 95% confidence level. Note that irace can perform parallel algorithm executions, so the tuning process can be completed faster.

Table 3 shows the best overall parameter configurations found by irace. The configurations are ordered by mean performance, with the first configuration being the best overall configuration.

The best overall configuration uses all three destruction operators and a single repair operator, which indicates that the heuristic's adaptive component enables improved performance, although it is only needed for the destruction phases. Note that, from the destruction operators, Shaw removal has the most positive impact on the performance of the ALNS heuristic, as it is enabled in every one of the best overall configurations. The use of spatial proximity removal and random removal is also beneficial, as both operators are enabled in most of the best overall configurations (in particular, they are enabled in the best two configurations). From these results, we can argue that the problem-specific dissimilarity measure used in Shaw removal is indeed effective, and the performance of the proposed ALNS approach is heavily reliant on it. Indeed, it is still possible to achieve good results by using only Shaw removal. Regarding the repair operators, we can see that in every one of the best overall configurations, the only repair operator used is random regret with a single comparison. This indicates that, to obtain the best results from the reinsertion process, it is sufficient to compare the cost of the best insertion with that of the second-best insertion.

Regarding the acceptance criteria, we can see that PTA is used in every one of the best overall configurations. Indeed, as Table 4 shows, if we consider all the iterations performed by irace, we see that PTA started to dominate RRT in the best configurations from the second iteration onward. Thus, we can argue that PTA is indeed more effective than RRT. In the following section, we discuss results of further experiments that confirm the dominance of PTA over RRT.

To conduct the experiments described in Sections 5.5 and 5.6, we set the ALNS parameters according to the first configuration of Table 3, as it was the best overall configuration found by irace.

Table 4
Acceptance criteria in the best configurations found by irace.

Configuration	Iteration										
	1	2	3	4	5	6	7	8	9	10	11
1st best	RRT	PTA	PTA	RRT	PTA	PTA	PTA	PTA	PTA	PTA	PTA
2nd best	RRT	PTA	PTA	PTA	PTA	PTA	PTA	PTA	PTA	PTA	PTA
3rd best	RRT	RRT	RRT	PTA	PTA	PTA	PTA	PTA	PTA	PTA	PTA
4th best	RRT	PTA	RRT	PTA	PTA	PTA	PTA	PTA	PTA	PTA	PTA
5th best	PTA	PTA	PTA	PTA	PTA	RRT	PTA	PTA	PTA	PTA	PTA
6th best	RRT	PTA	PTA	RRT	RRT	PTA	PTA	PTA	PTA	PTA	PTA

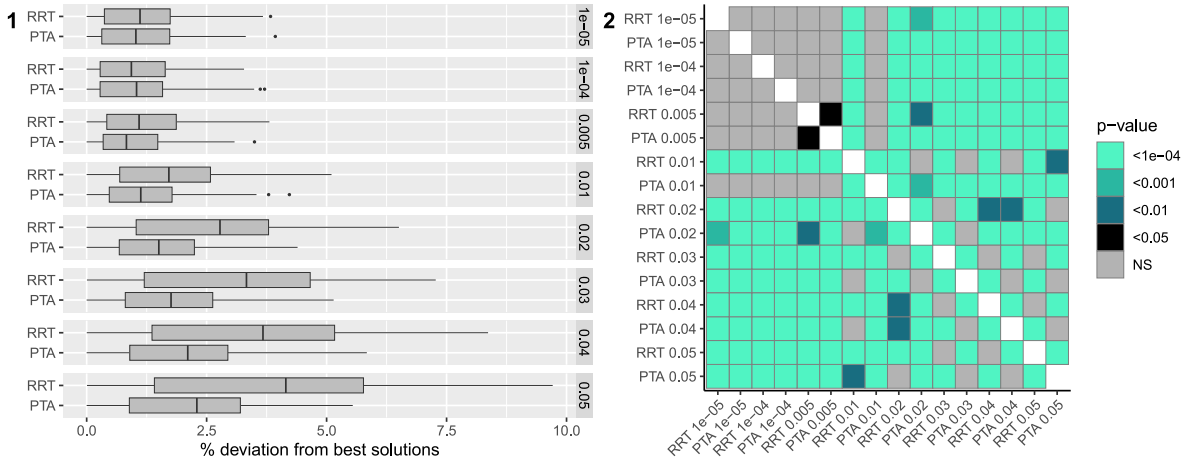


Fig. 8. PTA vs. RRT.

5.4. PTA vs. RRT

The results of irace indicate that PTA is capable of consistently overcoming RRT. To confirm such results, we conducted an additional set of experiments where we compared the performances of PTA and RRT across different values of the maximum acceptance threshold. In these experiments, we considered 20 instances (other than the training instances input to irace), with different numbers of requests, geographical distributions of locations, time-window lengths, and truck fleets. We evaluated the following values for the maximum acceptance threshold: 1×10^{-5}, 1×10^{-4}, 0.005, 0.01, 0.02, 0.03, 0.04, and 0.05. Using the proposed ALNS heuristic, we solved each instance 30 times for each combination of acceptance criterion and value of maximum acceptance threshold, keeping all other parameters according to the best configuration found by irace (the first configuration of Table 3). The results are presented in Fig. 8.1, which shows the distributions of percentage deviations from the best found solutions. For a given instance, let S and S^* be, respectively, the cost of one of the instance's solutions and the cost of the best instance's solution (found across all experiments). Then, the percentage deviation of cost S from cost S^* is given by $100 \cdot (S - S^*)/S^*$.

We used Kruskal–Wallis and Dunn's tests (with Bonferroni correction) to evaluate the differences between the combinations of acceptance criteria and maximum acceptance thresholds—assumptions of parametric analysis of variance (ANOVA) were not met, even after transformations. The Kruskal–Wallis test was significant ($p < 1 \times 10^{-4}$). The results of the Dunn's post-hoc tests are presented in Fig. 8.2; different significance levels are shown, with NS indicating not significant ($p \geq 0.05$).

We can make the following remarks on the obtained results.

1. PTA produced better solutions than RRT across the 0.005–0.05 range of maximum acceptance thresholds. The performance improvement produced by PTA over RRT is more evident at large values of the threshold (PTA reduced the mean percentage deviation from the best found solutions by approximately 40% for thresholds in the range 0.02–0.05). For smaller values of the threshold, the performance improvements produced by PTA over RRT are less pronounced (for thresholds equal to 0.01 and 0.005, PTA reduced the mean percentage deviation from the best found solutions by approximately 33% and 19%, respectively). On the other hand, PTA and RRT performed similarly at values of the maximum acceptance threshold equal to 1×10^{-4} and 1×10^{-5}.
2. Overall, significantly better solutions were obtained with PTA and a maximum acceptance threshold less or equal to 0.01, and with RRT and a maximum acceptance threshold less or equal to 0.005. All best found solutions—25 in total—but one were obtained with such parameter configurations (for some instances, the best found solution was produced repeatedly). Of the 25 best found solutions, 18 were obtained using PTA and 7 using RRT.
3. Solutions of comparable quality were obtained with PTA and a maximum acceptance threshold less or equal to 0.01 and with RRT and a maximum acceptance threshold less or equal to 0.005. Still, the best overall solutions, across all experiments, were obtained with PTA and a maximum acceptance threshold equal to 0.005; with values other than that, the performance of PTA decreased to different extents. Similarly, RRT performed the best with a maximum acceptance threshold equal to 1×10^{-4}, with its performance decreasing to different extents with values other than that.
4. PTA made the ALNS heuristic much more stable and robust to the value of the maximum acceptance threshold, as the performance hit resulting from using larger values of the threshold is much more marked when using RRT than when using PTA (*e.g.*, note that PTA with a threshold in the range 0.02-0.04 performed similarly to RRT with a threshold equal to 0.01).

The obtained results confirm that PTA enables improved performance over RRT, allowing to find better solutions across different values of the maximum acceptance threshold (with the performance gains being more marked at higher values), and making the search more robust to the value of such a parameter. The mechanisms implemented in PTA to amend the shortcomings of RRT work as intended.

Table 5
Results on benchmark set 1.

Class	Fragments		ALNS-PTA		Avg. cost gap		Avg. runtime		
	Optimal	Timeout solution	Optimal	≥ Frag.	Best (%)	Avg. (%)	Fragments (s)	ALNS-PTA (s)	$\frac{\text{Fragments}}{\text{ALNS-PTA}}$
Geographical distribution of locations									
C1	40	0	31	9	0.1	0.4	497.7	112.5	4.4
C2	38	2	31	9	0.2	0.5	39,509.8	112.5	351.2
R	40	0	32	8	0.07	0.3	69.3	112.5	0.6
Container and truck types									
T1	24	0	24	0	0.0	0.003	46.07	112.5	0.4
T2	23	1	13	11	0.2	0.9	26,105.9	112.5	232.05
T3	24	0	14	10	0.2	0.7	19,704.2	112.5	175.1
T4	24	0	24	0	0.0	0.0	4.5	112.5	0.04
T5	23	1	19	5	0.3	0.4	20,933.9	112.5	186.08
Time windows									
W1	60	0	51	9	0.03	0.2	66.2	112.5	0.6
W2	58	2	43	17	0.2	0.6	26,651.7	112.5	236.9
Number of requests									
25	30	0	29	1	0.1	0.2	1.8	45.0	0.04
50	30	0	26	4	0.05	0.3	127.2	90.0	1.4
75	30	0	22	8	0.09	0.4	16,503.4	135.0	122.2
100	28	2	17	13	0.2	0.7	36,803.4	180.0	204.5
All	118	2	94	26	0.1	0.4	13,358.9	112.5	118.7

5.5. Performance of the ALNS-PTA heuristic

We first conducted a set of experiments on benchmark set 1, aiming to compare the solutions of the ALNS-PTA heuristic as much as possible to optimal solutions found with the fragment-based approach of Bustos-Coral and Costa (2022). For the latter, we set a maximum runtime of five days for each instance. Note that, in these experiments, we excluded instances involving LCVs and more than 100 requests because the fragment-based approach could require even more than five days to solve some of such instances. Thus, trying to obtain some exact solutions to compare with would be impractical.

Then, we conducted a set of experiments on benchmark set 2, aiming to benchmark the ALNS-PTA heuristic against the fragment-based approach on larger instances. In this case, we set the maximum runtime for the fragment-based approach as Bustos-Coral and Costa (2022) did: one hour for the fragment generation phase and two hours for the Gurobi solver (aborting the solution process for instances for which the fragment generation phase did not complete within one hour).

In both sets of experiments, we performed 30 independent runs of the ALNS-PTA heuristic on each instance. The maximum runtime for each run was set to $1.8n$ s, where n is the number of requests of the instance being solved. Note that the ALNS-PTA heuristic uses the Rayon library for multithreaded operations, in this case being possible to spawn up to 72 threads (the number of logical cores available in any of the computing nodes we used). Hence, to make fairer comparisons, for the fragment-based approach we report only the runtimes of the Gurobi solver, which is multithreaded, and disregard the time spent in the fragment generation phase, which is single-threaded. As Bustos-Coral and Costa (2022) note, the time spent in the fragment generation phase could be significant when solving some instances. So, in some cases, the analyses could be indeed biased in favor of the fragment-based approach.

Finally, considering all runs of the ALNS-PTA heuristic across benchmark sets 1 and 2, we evaluated its consistency—the level of variation in the quality of the solutions—and convergence speed—how quickly the cost of a solution decreases as the search progresses.

The rest of this section is organized as follows: Sections 5.5.1 and 5.5.2 discuss the results of the experiments on benchmark sets 1 and 2, respectively. Section 5.5.3 discusses the consistency and convergence speed of the ALNS-PTA heuristic.

5.5.1. Results on benchmark set 1

Table 5 summarizes the results on the 120 instances of benchmark set 1. The same results are presented multiple times in the table, with different grouping of the instances: according to their geographical distribution of locations, fleet type, time-window setting, and number of requests. The bottom of the table displays a summary of the results for all instances. For the fragment-based approach, the table shows the counts of proven optimal solutions and feasible solutions returned upon reaching the runtime limit. For the ALNS-PTA heuristic, the table shows the counts of optimal solutions and feasible solutions with a cost greater or equal to the cost obtained by the fragment-based approach. The table also averages the smallest (*i.e.*, best) and average cost gaps achieved by the ALNS-PTA heuristic across 30 solutions for each instance. For a given instance, the cost gap for each of the solutions found by the ALNS-PTA heuristic is given by $100 \cdot (S - S^*)/S^*$, where S is the cost of the solution found by the ALNS-PTA heuristic and S^* is the cost of the corresponding solution found by the fragment-based approach. Finally, the table shows the ratios of the average runtimes of the fragment-based approach to the average runtimes of the ALNS-PTA heuristic (ratios greater than one indicate that the ALNS-PTA heuristic was faster than the fragment-based approach; ratios less than one indicate the opposite). The detailed solutions for this benchmark set are available in Online Appendix 2.

The fragment-based approach solved 118 instances to proven optimality, while it could not prove optimality for two instances within the allowed runtime of five days. Meanwhile, the ALNS-PTA heuristic found optimal solutions for 94 out of the 120 instances. For the remaining instances, it achieved marginal cost gaps (commonly of less than 1%) in relation to the solutions found by the fragment-based approach. These results reflect the effectiveness of the ALNS-PTA heuristic to solve DRP-HCL instances with up to 100 requests, standard trucks and LCVs, and different spatio-temporal characteristics.

Note that the performance of both the ALNS-PTA heuristic (in terms of solution quality) and the fragment-based approach (in terms of runtime) was better on instances involving single-container trucks, few requests, short time windows, and low truck/container compatibility. These features are *more restrictive*, as they reduce the number of feasible routes that can exist. By contrast, both methods showed decreased performance on instances involving LCVs, many requests, long time windows, and high truck/container compatibility. These features are *less restrictive*, as they increase the number of feasible routes that can exist. Note that the solutions of the ALNS-PTA heuristic tended to

Table 6
Results on benchmark set 2.

Class	Fragments			ALNS-PTA			Avg. cost gap		Avg. runtime		
	Optimal	Timeout solution	Timeout unsolved	Optimal	≥ Frag.	< Frag.	Best (%)	Avg. (%)	Fragments (s)	ALNS-PTA (s)	$\frac{\text{Fragments}}{\text{ALNS-PTA}}$
Geographical distribution of locations											
C1	83	1	6	25	59	6	0.5	0.7	386.5	930.0	0.4
C2	36	15	39	14	29	47	−3.1	−2.7	3,348.6	930.0	3.6
R	66	5	19	18	50	22	−0.4	−0.07	1,307.2	930.0	1.4
RC1	79	0	11	25	54	11	0.5	0.7	379.9	930.0	0.4
RC2	49	14	27	14	44	32	−1.4	−1.08	2,916.2	930.0	3.1
Container and truck types											
F1	76	14	0	23	55	12	−3.7	−3.5	1,672.5	930.0	1.8
F2	35	6	49	4	34	52	0.4	0.8	2,481.5	930.0	2.7
F3	54	2	34	10	46	34	1.08	1.5	1,170.9	930.0	1.3
F4	67	7	16	10	63	17	0.8	1.1	1,450.9	930.0	1.6
F5	81	6	3	49	38	3	0.07	0.2	1,035.5	930.0	1.1
Time windows											
W1	128	5	17	41	92	17	0.7	0.9	808.3	930.0	0.9
W2	78	21	51	21	67	62	−2.4	−2.05	2,567.7	930.0	2.8
W3	107	9	34	34	77	39	−0.4	−0.1	1,287.3	930.0	1.4
Number of requests											
100	73	2	0	51	22	2	0.005	0.2	563.7	180.0	3.1
200	59	5	11	20	42	13	−0.1	0.2	1,511.5	360.0	4.2
400	59	7	9	13	52	10	0.1	0.5	1,465.5	720.0	2.04
600	48	6	21	9	43	23	−0.09	0.2	1,770.1	1,080.0	1.6
800	41	6	28	3	40	32	−1.04	−0.8	1,880.5	1,440.0	1.3
1,000	33	9	33	0	37	38	−3.3	−3.06	2,258.7	1,800.0	1.3
All	313	35	102	96	236	118	−0.5	−0.3	1,487.1	930.0	1.6

be somewhat worse than those of the fragment-based approach on instances combining less restrictive features. Nevertheless, this fact is clearly offset by the much faster runtimes of the ALNS-PTA heuristic. Moreover, unlike the fragment-based approach, the performance of the ALNS-PTA heuristic was not significantly affected by the distances between the requests' pickup and delivery points (long distances can be more restrictive and short distances can be less restrictive).

To explain the quality differences in the solutions of the ALNS-PTA heuristic, note that, when solving instances combining more restrictive features, there can be fewer feasible insertion locations to consider when repairing a solution, which can ease the solution process. On the other hand, when solving instances combining less restrictive features, there can be more feasible insertion locations to consider when repairing a solution, which can hinder the solution process.

Overall, the takeaway from these experiments is that the ALNS-PTA heuristic can find high-quality solutions. However, its use is not always recommended. The decision on what solution method should be used comes down to the runtimes of the fragment-based approach. Generally, the fragment-based approach can find proven optimal solutions very quickly for instances combining more restrictive features. In contrast, this approach can require several hours (or even days) to solve some instances combining less restrictive features (such as LCVs)—it is in these cases where it may be better to opt for the ALNS-PTA heuristic, as runtimes can be cut to three minutes or less. Ultimately, the ALNS-PTA heuristic complements the fragment-based approach to solve instances where the use of the latter becomes impractical in terms of solving times.

To validate our observations above, we developed a machine learning (ML) approach that assists in deciding between using the ALNS-PTA heuristic or the fragment-based approach, based on the instances' characteristics. This ML approach indicated that the truck cargo capacity, time-window length, and number of requests are key factors to consider when deciding on the solution method to use. In addition, the ML approach confirmed that the ALNS-PTA heuristic is better to solve instances involving less restrictive features (particularly LCVs), while the fragment-based approach is better to solve instances involving more restrictive features. For details, see Online Appendix 3.

5.5.2. Results on benchmark set 2

Table 6 summarizes the results obtained on benchmark set 2 following a format similar to that of Table 5. In this case, the fragment-based approach found no solution for some instances within the given time limit. So, for the fragment-based approach, the table shows the counts of proven optimal solutions, feasible solutions returned upon reaching the runtime limit, and instances unsolved within the time limit. For the ALNS-PTA heuristic, the table shows the counts of optimal solutions, feasible solutions with a cost greater or equal to the cost obtained by the fragment-based approach, and feasible solutions with a cost lower than that obtained by the fragment-based approach (which includes solutions to instances not solved with the fragment-based approach). The table also compares the cost gaps and runtimes as done in Table 5 (in this case, considering only the instances for which the fragment-based approach found a solution). Note that we considered the solutions of the fragment-based approach provided by Bustos-Coral and Costa (2022) when they were better than the corresponding solutions found in our experiments. The detailed solutions for this benchmark set are available in Online Appendix 2.

On benchmark set 2, the ALNS-PTA heuristic found optimal solutions for 96 out of 450 instances. Furthermore, it found better solutions than the fragment-based approach for 16 instances (with cost reductions of up to around 44%) and solved 102 instances for which the fragment-based approach could not find any solution within the given time limit. All of this while achieving marginal cost gaps (commonly of less than 0.5%) on instances where the fragment-based approach found better solutions. These results show the effectiveness of the ALNS-PTA heuristic to solve larger-scale instances.

Note that, for the instances of this benchmark set, the following features are more restrictive: few containers that can be moved in a 2x20' load configuration, low truck/container compatibility, short time windows, few requests, and requests with distant pickup and delivery points. Meanwhile, the following features are less restrictive: many containers that can be moved in a 2x20' load configuration, high truck/container compatibility, long time windows, many requests, and requests with closely located pickup and delivery points.

Overall, the results obtained for benchmark set 2 were in accordance with the results obtained for benchmark set 1. The ALNS-PTA heuristic

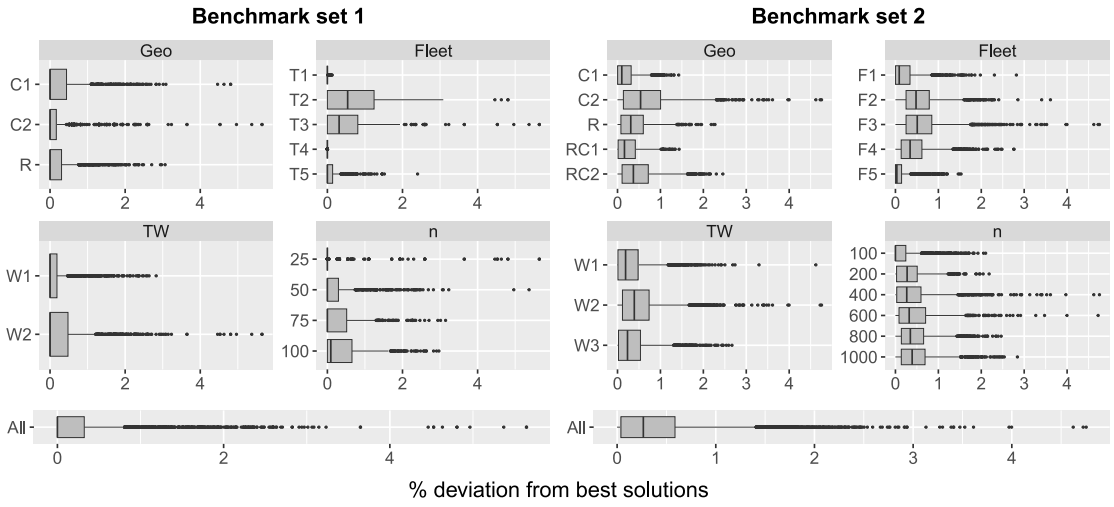


Fig. 9. Consistency of the ALNS-PTA heuristic.

and the fragment-based approach performed better on instances combining more restrictive features and worse on instances combining less restrictive features. (Note that in this case, the geographical distribution of locations did affect the performance of the ALNS-PTA heuristic.) Furthermore, the fragment-based approach proved to be more effective to solve instances combining more restrictive features, as it could find proven optimal solutions relatively quickly, while the ALNS-PTA heuristic was more suitable to solve instances combining less restrictive features.

5.5.3. Consistency and convergence speed

To assess the consistency of the ALNS-PTA heuristic, we calculated the percentage deviations from the best found solutions. Considering a solution with cost S for a particular instance, the percentage deviation from the best found solution is given by $100 \cdot (S - S^*)/S^*$, where S^* is the cost of the best solution found across 30 runs for the instance in question. Fig. 9 shows the distributions of percentage deviations from the best found solutions across all experiments on benchmark sets 1 and 2, aggregating the data based on the instances' features.

Overall, the costs of the solutions were generally within 1% of the cost of the corresponding best found solution. Larger cost gaps were still less than 3%, with only a few exceptions. Note that the obtained solutions were generally more consistent for instances with more restrictive features and less consistent for instances with less restrictive features. We can also analyze the coefficients of variation of the solutions' costs for each instance. (It is possible to calculate these coefficients based on the data provided in Online Appendix 2.) For benchmark set 1, the coefficients of variation were in the range 0% to 1.66%; for benchmark set 2, they were in the range 0% to 1.08%. These results offer evidence that the ALNS-PTA heuristic can produce high-quality solutions consistently for a wide range of DRP-HCL instances.

To assess the convergence speed of the ALNS-PTA heuristic, in each experiment we recorded the cost of the current best solution, S_{best}, at 101 equally distributed timestamps during the search process, with the first cost being that of the naïve initial solution, and the last cost being that of the final solution. Based on these data, we computed the percentage deviations from the final solutions. Specifically, let S be the cost of solution S_{best} at a given timestamp, and let S' be the cost of the final solution that is returned at the end of the search. Then, at the considered timestamp, the percentage deviation from the final solution is given by $100 \cdot (S - S')/S'$. Fig. 10 shows the mean percentage deviations from the final solutions across all experiments on benchmark sets 1 and 2, aggregating the data based on the instances' features (note that, at timestamp x, approximately x% of the maximum runtime has been used).

Overall, the ALNS-PTA heuristic converged faster to the final solutions on instances with more restrictive features and slower for instances with less restrictive features. Note that solutions with a cost within 1% of the final solutions were found very early in the search for some instances with more restrictive features. The same was achieved only at the very end of the search for some instances with less restrictive features. Based on these results, we could even reduce the maximum allowed runtimes for some instances (*e.g.*, instances with 75 requests or fewer) and increase them for others (*e.g.*, instances with 400 requests or more).

5.6. Using the ALNS-PTA heuristic in fleet decision-making

In this section, we show how the ALNS-PTA heuristic can assist decision-making in realistic scenarios of drayage routing involving LCVs—assuming that no restrictions limit the operation of these trucks. The aim is to determine if the larger load capacities of LCVs lead to savings that compensate for the higher costs of these trucks. In this case, we consider a scenario concerning drayage of 20/40 ft containers by trucks of the types described in Table 7. (All costs are in USD. The fixed costs are per day and the routing costs are per minute. See Online Appendix 1 for details on the modeling of the trucks, including the calculation of these costs.)

We consider seven truck fleets: The first fleet is heterogeneous, composed of all truck types described in Table 7 (we refer to this fleet as "ALL"). The second fleet is also heterogeneous, composed of 5A-20' and 5A-40' trucks (we refer to this fleet as "5A-20'&5A-40' "). The remaining fleets are homogeneous, composed of either 6A-20/40', RMD-40'+20', RMD-20/40'+20', TD-20/40', or TD-20/40'+40' trucks (these fleets are named after the truck types that compose them).

We also consider two types of drayage movements, inter-carrier and shuttle-pier movements, described in Fig. 11. These drayage movements are based on the types of drayage described by the Intermodal Association of North America (2021). Note that, in terms of area of service, inter-carrier and shuttle-pier movements correspond to regional and local operations, respectively. It should be noted that we assume a planning horizon of 600 min and mixed time windows of 60 and 120 min in all cases.

To tackle the scenario under consideration, we conducted computational experiments involving the following steps: First, we defined six

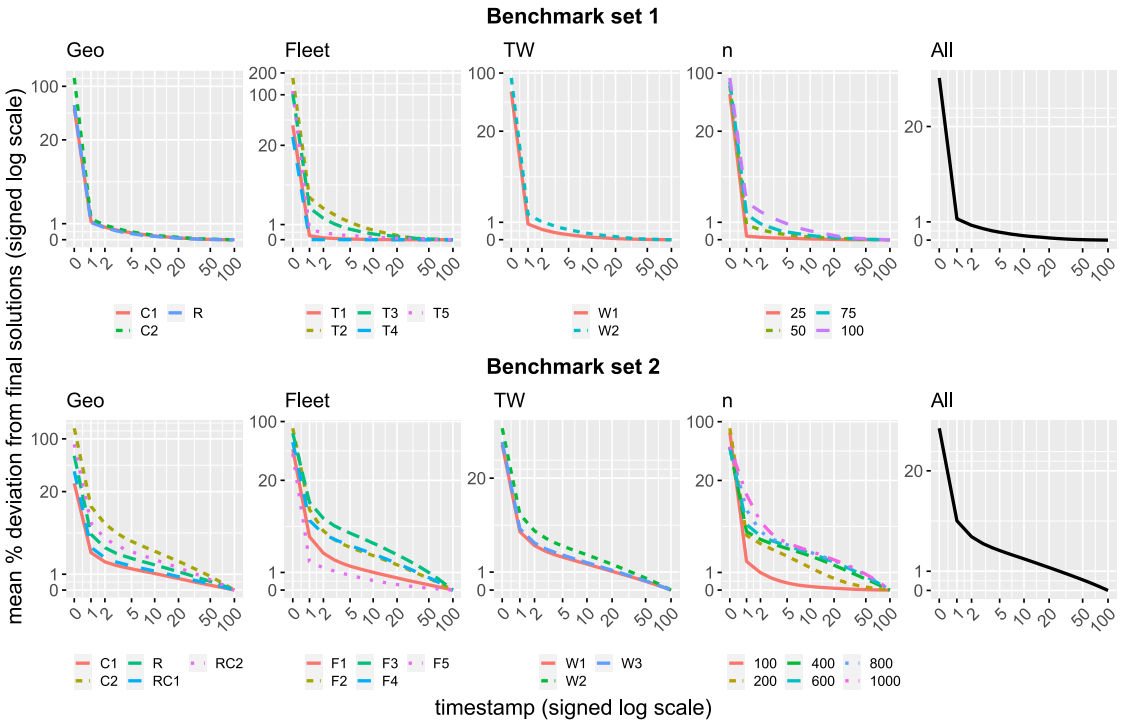
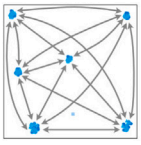


Fig. 10. Convergence speed of the ALNS-PTA heuristic.

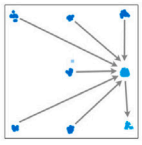
Table 7
Truck types used to move 20/40 ft containers.

Truck type	Description	E^f	$c^f_{i,j}, (i,j) \in \mathcal{A}$
5A–20'	Five-axle tractor-semitrailer consisting of a semi-truck and a 20' 2-axle chassis.	18	$1.7\tau_{i,j}$
5A–40'	Five-axle tractor-semitrailer consisting of a semi-truck and a 40' 2-axle chassis.	17	$1.7\tau_{i,j}$
6A–20/40'	Six-axle tractor-semitrailer consisting of a semi-truck and a 20/40' 3-axle chassis.	31	$1.7\tau_{i,j}$
D–20'	Double consisting of a semi-truck and two 20' 2-axle chassis.	53	$1.7\tau_{i,j}$
RMD–40'+20'	Rocky mountain double consisting of a semi-truck, a 40' 2-axle chassis, and a 20' 2-axle chassis.	53	$2.0\tau_{i,j}$
RMD–20/40'+20'	Rocky mountain double consisting of a semi-truck, a 20/40' 3-axle chassis, and a 20' 2-axle chassis.	66	$2.0\tau_{i,j}$
TD–20/40'	Turnpike double consisting of a semi-truck and two 20/40' 3-axle chassis.	79	$1.9\tau_{i,j}$
TD–40'	Turnpike double consisting of a semi-truck and two 40' 2-axle chassis.	53	$1.9\tau_{i,j}$
TD–20/40'+40'	Turnpike double consisting of a semi-truck, a 20/40' 3-axle chassis, and a 40' 2-axle chassis.	66	$1.9\tau_{i,j}$
TTC–20'	Triple trailer combination consisting of a semi-truck and three 20' 2-axle chassis.	88	$1.9\tau_{i,j}$

A 20' 2-axle chassis carries one 20 ft container, a 40' 2-axle chassis carries one 40 ft container, and a 20/40' 3-axle chassis carries one 20/40 ft container or two empty 20 ft containers.



Inter-carrier: Containers are moved between different carriers' facilities to continue with their journey. For example, containers are transferred from sea to rail, rail to sea, or rail to rail. These movements can involve long distances.



Shuttle-pier: Containers are moved to a temporary storage place from different shipping hubs that are overcapacity and can no longer accommodate more containers (shuttle drayage movements). In addition, containers are moved from the temporary storage place to a pier for further transportation (pier drayage movements). These movements involve short distances.

Fig. 11. Types of drayage movements.

sub-scenarios characterized by a type of drayage movement and a number of requests, as follows: inter-carrier movements with 100 requests, inter-carrier movements with 200 requests, inter-carrier movements with 400 requests, shuttle-pier movements with 100 requests, shuttle-pier movements with 200 requests, and shuttle-pier movements with 400 requests. For each sub-scenario, we generated 30 graphs of requests $\mathcal{G}$. Then, from each graph $\mathcal{G}$, we derived and solved alternative DRP-HCL instances, with the same requests as those of graph $\mathcal{G}$, but with different truck fleets. To solve each instance, we conducted five heuristic runs and chose the best solution from among those runs. The results are presented in Fig. 12, which shows the distributions of the costs and the aggregated numbers of trucks, across all experiments.

A three-way ANOVA test (with log-transformed costs) revealed that the main effects for type of drayage movement ($F_{(1,\ 1,218)} = 16{,}922.39$, $p < 0.001$), number of requests ($F_{(2,\ 1,218)} = 24{,}360.96$, $p < 0.001$), and fleet type ($F_{(6,\ 1,218)} = 783.36$, $p < 0.001$) were all significant.[1] The interactions between type of drayage movement and number of requests ($F_{(2,\ 1,218)} = 35.41$, $p < 0.001$) and type of drayage movement and fleet type ($F_{(6,\ 1,218)} = 2.62$, $p = 0.016$) were also significant. Tukey's post-hoc tests yielded the following results: Across all sub-scenarios, no significant cost differences were found between fleets 6A-20/40' and RMD-40'+20' (all p-values > 0.88), as well as between fleets RMD-20/40'+20' and TD-20/40'+40' (p-value = 0.08 for the inter-carrier

[1] The notation $F_{(bdf,wdf)} = fs$, $p < sl$, reads as follows: bdf is the between-groups degrees of freedom, wdf is the within-groups degrees of freedom, fs is the value of the F statistic, and sl is the significance level.

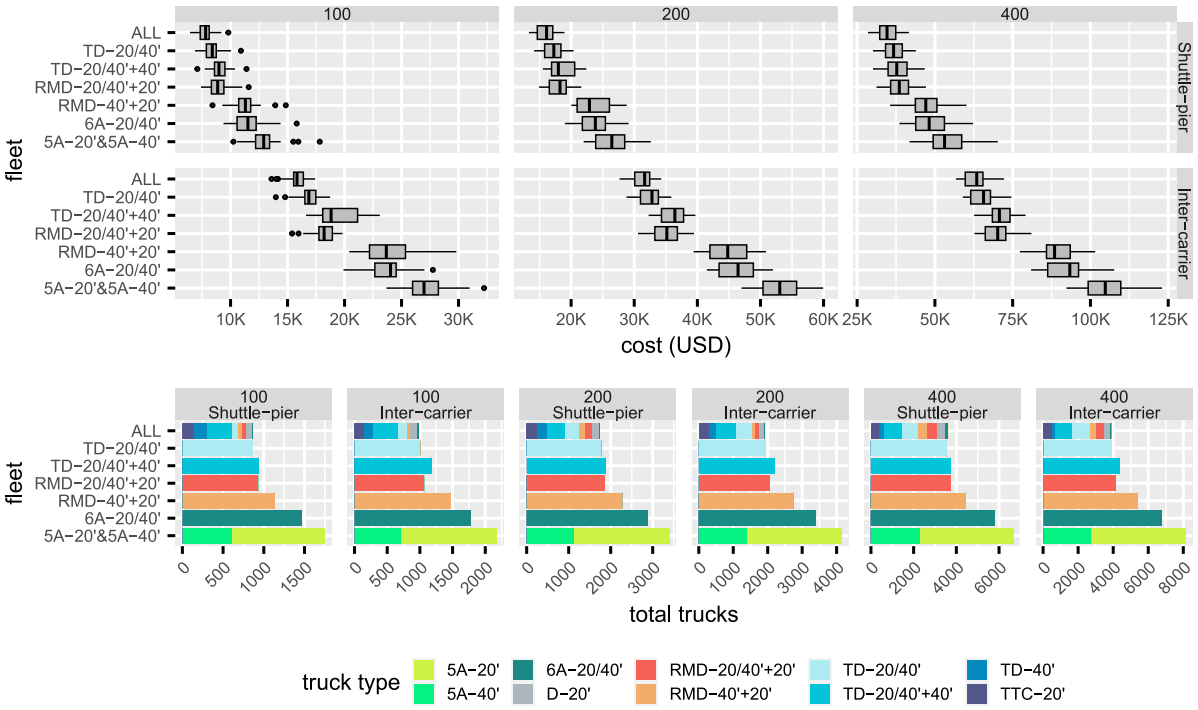


Fig. 12. Results: drayage of 20/40 ft containers.

sub-scenario with 100 requests, all other p-values > 0.74). Across all shuttle-pier sub-scenarios, no significant cost differences were found between fleets RMD-20/40'+20' and TD-20/40' (all p-values > 0.11). In the shuttle-pier sub-scenario involving 400 requests, no significant cost difference was found between fleets TD-20/40'+40' and TD-20/40' (p-value = 0.7). Across all inter-carrier sub-scenarios and in the shuttle-pier sub-scenario involving 400 requests, no significant cost differences were found between fleets TD-20/40' and ALL (all p-values > 0.13). All remaining comparisons between the costs incurred by the fleets, across all sub-scenarios, were significant (all p-values < 0.05). Below, we comment on these results.

The fleet composed of 5-axle trucks was clearly outperformed by all the other fleets, indicating that the operation of trucks with higher cargo capacity was always more convenient. The higher costs associated with trucks of higher cargo capacity were always outweighed by the obtained cost savings.

Across all sub-scenarios, the fleet composed of 6A-20/40' trucks and the fleet composed of RMD-40'+20' trucks achieved similar costs. Note that, although RMD-40'+20' trucks are more expensive to acquire and run than 6A-20/40' trucks, it is necessary to purchase fewer RMD-40'+20' trucks than 6A-20/40' trucks to serve the same number of requests, which balances out the costs of both truck types. However, fleets 6A-20/40' and RMD-40'+20' were clearly outperformed by fleets RMD-20/40'+20', TD-20/40'+40', TD-20/40', and ALL across all sub-scenarios, indicating that the fleet composed of RMD-40'+20' trucks was the only fleet of LCVs that did not offer significant cost savings compared to the fleet of 6-axle trucks.

When analyzing the results of the fleet composed of RMD-20/40'+20' trucks, we can see that, generally, no significant cost differences were found between this fleet and the fleet composed of TD-20/40'+40' trucks. This is an expected result, as both fleets involve similar fixed and routing costs. In addition, note that when the drayage movements involved short distances, no significant cost differences were found between the fleet composed of RMD-20/40'+20' trucks and the fleet composed of TD-20/40' trucks. However, the latter achieved significant cost savings relative to the former when the drayage movements involved longer distances. Note also that, across all sub-scenarios, the fleet composed of all truck types realized significant cost savings relative to the fleet of RMD-20/40'+20' trucks.

As to the fleets involving turnpike doubles, we can see that the fleet composed of TD-20/40' trucks consistently achieved significant cost savings when compared to the fleet composed of TD-20/40'+40' trucks (although this did not happen in the shuttle-pier sub-scenario involving 400 requests). Similarly, the fleet composed of all truck types realized significant cost savings in relation to the fleet composed of TD-20/40'+40' trucks across all sub-scenarios. Moreover, note that the fleet composed of all truck types achieved significant cost savings relative to the fleet composed of TD-20/40' trucks in the shuttle-pier sub-scenarios involving 100 and 200 requests. In the remaining sub-scenarios, no significant cost differences were found between these fleets.

To quantify the fleets' overall performances, Table 8 summarizes the percentage changes between the fleets' average costs (Cost) and cumulative numbers of trucks (N.T.), calculated across all sub-scenarios (the fleets on the top row of the table are the references for comparison). Overall, two fleets were the most cost-effective: fleets composed of all truck types and fleets composed only of TD-20/40' trucks. When compared to the other fleets, these fleets obtained savings in average costs of up to about 40%, and reduced the fleet size by up to 50%. In this regard, it is interesting to see that, in the fleets composed of all truck types, the most common trucks were turnpike doubles, followed by triple trailer combinations, rocky mountain doubles, and doubles. On the other hand, very few 5-axle and 6-axle trucks were used. Note that, in cases where a fleet composed of all truck types and a fleet of TD-20/40' trucks perform similarly, one should opt for the latter, as a homogeneous fleet would be much easier to manage than a fleet involving various truck types.

By using the ALNS-PTA heuristic, it is possible to derive managerial insights into different scenarios of drayage routing akin to the scenario described in this section. In Online Appendix 4, we follow the same methodology used in this section to create and analyze two additional scenarios. One scenario involves drayage of 40/45/53 ft containers by 5-axle trucks and turnpike doubles. The other scenario involves drayage of 20 ft containers by 5-axle trucks, doubles, and triple trailer combinations.

Table 8
Percentage changes in average costs and cumulative numbers of trucks.

	5A−20'&5A−40'		6A−20/40'		RMD−40'+20'		TD−20/40'+40'		RMD−20/40'+20'		TD−20/40'	
	Cost	N.T.	Cost	N.T.	Cost	N.T.	Cost	N.T.	Cost	N.T.	Cost	N.T.
6A−20/40'	−12	−16										
RMD−40'+20'	−14	−33	−2	−21								
TD−20/40'+40'	−31	−46	−22	−35	−20	−18						
RMD−20/40'+20'	−32	−48	−23	−38	−22	−21	−2	−4				
TD−20/40'	−37	−50	−28	−41	−27	−26	−8	−9	−6	−5		
ALL	−40	−50	−31	−41	−30	−26	−12	−9	−11	−5	−4	−0.1

6. Conclusions

We have proposed an adaptive large neighborhood search (ALNS) heuristic for drayage routing problems involving longer combination vehicles (LCVs) and further aspects, such as heterogeneous truck fleets, containers of any size and cargo category, and compatibility constraints based on the containers' characteristics. The proposed ALNS heuristic incorporates tailored search operators, which consider factors such as compatibility constraints and specific truck load configurations. In developing our ALNS heuristic, we also have introduced the pulsed threshold acceptance (PTA) criterion, which builds upon the record-to-record travel (RRT; Santini et al., 2018) acceptance criterion, with additional mechanisms to avoid search stagnation and to allow for revisiting the current best solution.

Through a series of benchmark tests against the fragment-based algorithm of Bustos-Coral and Costa (2022), the state-of-the-art exact approach for the considered problems, we showed that the proposed ALNS heuristic can consistently find high-quality solutions (we found new best known solutions for several instances) and, when solving instances involving LCVs and other difficult instances, it can cut runtimes from hours or even days to a few minutes. We also showed that PTA can produce better solutions than RRT. Finally, we showed how the proposed ALNS heuristic can be used to derive managerial insights into the performance in drayage operations of different US truck classes, including LCVs and standard trucks. Among the obtained results, we found that fleets comprising different types of LCVs (doubles, rocky mountain doubles, turnpike doubles, and triple trailer combinations) can realize savings of nearly 40% in average costs and reduce fleet size by up to 50%, in relation to fleets of 5-axle tractor semi-trailers.

Future work could aim to solve exactly the considered problems. A possibility is to develop a fragment-based algorithm combining the developments of Rist and Forbes (2021, 2022) and Bustos-Coral and Costa (2022), where the proposed ALNS heuristic is used to obtain primal bounds; such an algorithm should be highly scalable. Future work could also adapt our approach to tackle more complex drayage problems, involving, *e.g.*, tractor-trailers that can be coupled/uncoupled, as in the problem of Zhang et al. (2021). Furthermore, the proposed search operators could be adapted to tackle trucking problems involving the transportation of large pieces of equipment, *e.g.*, cable reels, tractors, or other agriculture equipment, possibility involving compatibility constraints. Finally, future work could also analyze the performance of the PTA criterion when used in heuristic approaches to other problems, with the goal of confirming its efficacy.

CRediT authorship contribution statement

Daniel Bustos-Coral: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Alysson M. Costa:** Writing – review & editing, Supervision, Resources, Conceptualization.

Data availability

Data has been made available on GitHub at https://github.com/dbuscor/drp_hcl_instances.

Acknowledgments

This research was funded by the University of Melbourne through a Melbourne Research Scholarship, awarded to the first author, and by the Australian Government through the Australian Research Council Industrial Transformation Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Project ID IC200100009. We gratefully acknowledge this support.

We would also like to thank the four anonymous reviewers for their valuable comments and suggestions that have helped improve this paper.

Appendix A. Compact MIP formulation for the DRP-HCL

This section presents a compact MIP formulation for the DRP-HCL based on the concepts outlined in Sections 3.1 and 3.2 (this formulation was first introduced by Bustos-Coral and Costa, 2022).

Let $\mathcal{T} = \{1, \dots, T\}$ be a set of trucks and let $f : \mathcal{T} \to \mathcal{F}; t \mapsto f(t)$ be a function mapping each truck to its corresponding type. Furthermore, define the following sets:

- $\Delta_i^f = \{h' \in \Theta_f : q_f(h,r) = h',\ (h,r) \in \Sigma_f, \text{ and } r = r_i\}$, which contains *the possible states in which a truck of type $f \in \mathcal{F}$ can be after visiting node $i \in \mathcal{N}$*; and
- $\Omega_{i,j}^f = \{(h,h') \in \Theta_f^2 : q_f(h,r) = h',\ (h,r) \in \Sigma_f,\ h \in \Delta_i^f, \text{ and } r = r_j\}$, which contains *the transitions (state at node i, state at node j) that a truck of type $f \in \mathcal{F}$ can perform in traversing an arc $(i,j) \in \mathcal{A}$*.

For each arc $(i,j) \in \mathcal{A}$ and truck $t \in \mathcal{T}$, let $x_{i,j}^t$ be a binary variable equal to one if and only if truck t travels arc (i,j), and zero otherwise. For each node $i \in \mathcal{N}$, truck $t \in \mathcal{T}$, and state $h \in \Delta_i^{f(t)}$, let $y_i^{t,h}$ be a binary variable equal to one if and only if, after visiting node i, truck t is in state h, and zero otherwise. For each arc $(i,j) \in \mathcal{A}$, truck $t \in \mathcal{T}$, and state transition $(h,h') \in \Omega_{i,j}^{f(t)}, |\Omega_{i,j}^{f(t)}| > 1$, let $z_{i,j}^{t,h,h'}$ be a binary variable equal to one if and only if truck t travels arc (i,j) making state transition (h,h'), and zero otherwise. Finally, for each node $i \in \mathcal{N}$ and truck $t \in \mathcal{T}$, let B_i^t be the time at which truck t visits node i.

Using the definitions above, we formulate the DRP-HCL as the MIP of Eqs. (A.1) to (A.20).

$$\text{minimize} \quad \sum_{t\in\mathcal{T}} E^{f(t)} \sum_{j\in\mathcal{P}} x_{0,j}^t + \sum_{t\in\mathcal{T}} \sum_{(i,j)\in\mathcal{A}} c_{i,j}^{f(t)} x_{i,j}^t, \tag{A.1}$$

$$\text{subject to:} \quad \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{N}} x_{i,j}^t = 1, \quad \forall i \in \mathcal{P}, \tag{A.2}$$

$$\sum_{j\in\mathcal{N}} x_{i,j}^t - \sum_{j\in\mathcal{N}} x_{n+i,j}^t = 0, \quad \forall i \in \mathcal{P}, t \in \mathcal{T}, \tag{A.3}$$

$$\sum_{j\in\mathcal{N}} x_{0,j}^t = 1, \quad \forall t \in \mathcal{T}, \tag{A.4}$$

$$\sum_{i\in\mathcal{N}} x_{i,2n+1}^t = 1, \quad \forall t \in \mathcal{T}, \tag{A.5}$$

$$\sum_{j\in\mathcal{N}} x_{j,i}^t - \sum_{j\in\mathcal{N}} x_{i,j}^t = 0, \quad \forall i \in \mathcal{P} \cup \mathcal{D}, t \in \mathcal{T}, \tag{A.6}$$

$$B_j^t \geq B_i^t + s_i + \tau_{i,j} - M_{i,j}(1 - x_{i,j}^t), \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, \tag{A.7}$$

$$e_i \leq B_i^t \leq l_i, \quad \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{A.8}$$

$$B_{n+i}^t \geq B_i^t + s_i + \tau_{i,n+i}, \quad \forall i \in \mathcal{P}, t \in \mathcal{T}, \tag{A.9}$$

$$2x^t_{i,j} \le y^{t,h}_i + y^{t,h'}_j + 2(1 - z^{t,h,h'}_{i,j}), \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, (h,h') \in \Omega^{f(t)}_{i,j}, \left|\Omega^{f(t)}_{i,j}\right| > 1, \tag{A.10}$$

$$\sum_{(h,h') \in \Omega^{f(t)}_{i,j}} z^{t,h,h'}_{i,j} = x^t_{i,j}, \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, \left|\Omega^{f(t)}_{i,j}\right| > 1, \tag{A.11}$$

$$2x^t_{i,j} \le y^{t,h}_i + y^{t,h'}_j, \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, (h,h') \in \Omega^{f(t)}_{i,j}, \left|\Omega^{f(t)}_{i,j}\right| = 1, \tag{A.12}$$

$$x^t_{i,j} = 0, \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, \left|\Omega^{f(t)}_{i,j}\right| = 0, \tag{A.13}$$

$$y^{t,h}_0 = 1, \quad \forall t \in \mathcal{T}, h \in \Delta^{f(t)}_0, \tag{A.14}$$

$$y^{t,h}_{2n+1} = 1, \quad \forall t \in \mathcal{T}, h \in \Delta^{f(t)}_{2n+1}, \tag{A.15}$$

$$\sum_{h \in \Delta^{f(t)}_i} y^{t,h}_i = \sum_{j \in \mathcal{N}} x^t_{i,j}, \quad \forall i \in \mathcal{P} \cup \mathcal{D}, t \in \mathcal{T}, \tag{A.16}$$

$$x^t_{i,j} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, \tag{A.17}$$

$$y^{t,h}_i \in \{0,1\}, \quad \forall i \in \mathcal{N}, t \in \mathcal{T}, h \in \Delta^{f(t)}_i, \tag{A.18}$$

$$z^{t,h,h'}_{i,j} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}, (h,h') \in \Omega^{f(t)}_{i,j}, \left|\Omega^{f(t)}_{i,j}\right| > 1, \tag{A.19}$$

$$B^t_i \ge 0, \quad \forall i \in \mathcal{N}, t \in \mathcal{T}. \tag{A.20}$$

The objective function (A.1) minimizes the total fixed and routing costs. Constraints (A.2) and (A.3) guarantee that each request is served exactly once and, for each request, the pickup and delivery nodes are visited by the same truck. Constraints (A.4) and (A.5) ensure, respectively, that each truck leaves and returns to the depot node exactly once (trucks that go straight from the start depot to the end depot incur no cost). Constraints (A.6) guarantee that the truck that arrives at a pickup/delivery node also departs from it. Time-window constraints are defined by (A.7) and (A.8); $M_{i,j}$ are big M values that can be set to $\max\{l_i + s_i + \tau_{i,j} - e_j, 0\}$, $\forall (i,j) \in \mathcal{A}$, as noted by Desaulniers et al. (2014). Constraints (A.9) ensure that, for each request, the pickup node is visited before the delivery node. The state transition a truck t makes in traversing an arc (i,j) is set by constraints (A.10) and (A.11), when more than one state transition is possible, and by constraints (A.12), when only one state transition is possible. Constraints (A.13) discard the arcs that a truck cannot traverse due to the lack of valid state transitions. Constraints (A.14) to (A.16) ensure that a truck is in a unique state after visiting each node along its route (note that $|\Delta^{f(t)}_0| = |\Delta^{f(t)}_{2n+1}| = 1$). The variable domains are set by constraints (A.17) to (A.20). The formulation also includes the following trivial variable fixations: $x^t_{i,i} = 0$, $\forall i \in \mathcal{N}, t \in \mathcal{T}$ and $x^t_{n+i,i} = 0$, $\forall i \in \mathcal{P}, t \in \mathcal{T}$. Note that constraints (A.13) discard the following arcs, for which there are no feasible state transitions: $x^t_{i,0} = 0$, $\forall i \in \mathcal{N}, t \in \mathcal{T}$; $x^t_{2n+1,i} = 0$, $\forall i \in \mathcal{N}, t \in \mathcal{T}$; $x^t_{i,2n+1} = 0$, $\forall i \in \mathcal{P}, t \in \mathcal{T}$; and $x^t_{0,j} = 0$, $\forall j \in \mathcal{D}, t \in \mathcal{T}$.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cor.2024.106826.

References

Bjelić, N., Vidović, M., Popović, D., Ratković, B., 2022. Rolling-horizon approach in solving dynamic multisize multi-trailer container drayage problem. Expert Syst. Appl. 201, 117170.

Bustos-Coral, D., Costa, A.M., 2022. Drayage routing with heterogeneous fleet, compatibility constraints, and truck load configurations. Transp. Res. E Logist. Transp. Rev. 168, 102922.

Chao, I.-M., 2002. A tabu search method for the truck and trailer routing problem. Comput. Oper. Res. 29 (1), 33–51.

Chen, R., Chen, S., Cui, H., Meng, Q., 2021. The container drayage problem for heterogeneous trucks with multiple loads: A revisit. Transp. Res. E Logist. Transp. Rev. 147, 102241.

Chen, R., Meng, Q., Jia, P., 2022. Container port drayage operations and management: Past and future. Transp. Res. E Logist. Transp. Rev. 159, 102633.

Christiaens, J., Vanden Berghe, G., 2020. Slack induction by string removals for vehicle routing problems. Transp. Sci. 54 (2), 417–433.

Desaulniers, G., Madsen, O.B., Ropke, S., 2014. Chapter 5: the vehicle routing problem with time windows. In: Toth, P., Vigo, D. (Eds.), Vehicle Routing: Problems, Methods, and Applications, second ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, pp. 119–159.

Fletcher, N., 2022. Louisiana starts pilot with two-container trucks at ports. https://www.ttnews.com/articles/louisiana-starts-pilot-two-container-trucks-ports. (Accessed 14 February 2023).

François, V., Arda, Y., Crama, Y., Laporte, G., 2016. Large neighborhood search for multi-trip vehicle routing. European J. Oper. Res. 255 (2), 422–441.

Friedrich, C., Elbert, R., 2022. Adaptive large neighborhood search for vehicle routing problems with transshipment facilities arising in city logistics. Comput. Oper. Res. 137, 105491.

Intermodal Association of North America, 2021. Intermodal glossary. URL: https://www.intermodal.org/education/intermodal-glossary. (Accessed 09 April 2023).

Lai, M., Battarra, M., Di Francesco, M., Zuddas, P., 2015. An adaptive guidance metaheuristic for the vehicle routing problem with splits and clustered backhauls. J. Oper. Res. Soc. 66 (7), 1222–1235.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T., 2016. The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58.

Moghaddam, M., Pearce, R.H., Mokhtar, H., Prato, C.G., 2020. A generalised model for container drayage operations with heterogeneous fleet, multi-container sizes and two modes of operation. Transp. Res. E Logist. Transp. Rev. 139, 101973.

PassMark Software, 2024. Intel Xeon Gold 6154 @ 3.00GHz vs Intel Xeon Gold 6254 @ 3.10GHz. URL: https://www.cpubenchmark.net/compare/3127vs3482/Intel-Xeon-Gold-6154-vs-Intel-Xeon-Gold-6254. (Accessed 01 March 2024).

Resor, R.R., Blaze, J.R., Morlok, E.K., 2004. Short-haul rail intermodal: Can it compete with trucks? Transp. Res. Rec. J. Transp. Res. Board 1873 (1), 45–52.

Rist, Y., Forbes, M.A., 2021. A new formulation for the dial-a-ride problem. Transp. Sci. trsc.2021.1044.

Rist, Y., Forbes, M., 2022. A column generation and combinatorial benders decomposition algorithm for the selective dial-a-ride-problem. Comput. Oper. Res. 140, 105649.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.

Santini, A., Ropke, S., Hvattum, L.M., 2018. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. J. Heuristics 24 (5), 783–815.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: Goos, G., Hartmanis, J., van Leeuwen, J., Maher, M., Puget, J.-F. (Eds.), In: Principles and Practice of Constraint Programming — CP98, vol. 1520, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 417–431.

Thompson, R.G., Hassall, K., 2014. Implementing high productivity freight vehicles in urban areas. Procedia Soc. Behav. Sci. 151, 318–332.

Tunnell, M.A., 2008. Energy and Emissions Impacts of Operating Higher Productivity Vehicles: Update 2008 Technical report, American Transportation Research Institute, Arlington, VA.

Vidović, M., Popović, D., Ratković, B., Radivojević, G., 2017. Generalized mixed integer and VNS heuristic approach to solving the multisize containers drayage problem. Int. Trans. Oper. Res. 24 (3), 583–614.

Zhang, R., Wang, D., Wang, J., 2021. Multi-trailer drop-and-pull container drayage problem. IEEE Trans. Intell. Transp. Syst. 22 (9), 5708–5720.