A three-phase algorithm for the three-dimensional loading vehicle routing problem with split pickups and time windows

Emeline Leloup[a,*], Célia Paquay[a], Thierry Pironet[a], José Fernando Oliveira[b]

[a] *QuantOM, HEC - Management School of the University of Liège, Rue Louvrex 14, 4000, Liège, Belgium*

[b] *INESC TEC, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal*

Abstract

In a survey of Belgian logistics service providers, the efficiency of first-mile pickup operations was identified as a key area for improvement, given the increasing number of returns in e-commerce, which has a significant impact on traffic congestion, carbon emissions, energy consumption and operational costs. However, the complexity of first-mile pickup operations, resulting from the small number of parcels to be collected at each pickup location, customer time windows, and the need to efficiently accommodate the highly heterogeneous cargo inside the vans, has hindered the development of real-world solution approaches. This article tackles this operational problem as a vehicle routing problem with time windows, time-dependent travel durations, and split pickups and integrates practical 3D container loading constraints such as vertical and horizontal stability as well as a more realistic reachability constraint to replace the classical "Last In First Out" (LIFO) constraint. To solve it, we propose a three-phase heuristic based on a savings constructive heuristic, an extreme point concept for the loading aspect and a General Variable Neighborhood Search as an improvement phase for both routing and packing. Numerical experiments are conducted to assess the performance of the algorithm on benchmark instances and new instances are tested to validate the positive managerial impacts on cost when allowing split pickups and on driver working duration when extending customer time windows. In addition, we show the impacts of considering the reachability constraint on cost and of the variation of speed during peak hours on schedule feasibility.

Keywords: Routing, Packing, Split collection, 3D loading, Variable Neighborhood Search

*Corresponding author

Email addresses: emeline.leloup@uliege.be (Emeline Leloup), cpaquay@uliege.be (Célia Paquay), thierry.pironet@uliege.be (Thierry Pironet), jfo@fe.up.pt (José Fernando Oliveira)

1. Introduction

Electronic commerce, or e-commerce, is rapidly expanding around the world (retail e-commerce sales amounted to $5,311 billion worldwide in 2022 up from $4,248 billion in 2020) (Statista, 2023). Consumers tend to prefer shopping online as it enables them to choose from a variety of offers from a large number of retailers with price transparency from the comfort of their homes. In 2023, retail e-commerce sales represented 19.4% of total retail sales (Statista, 2024).

As a consequence, the quantity freight increases and generates some drawbacks. In particular, customers are also returning products at a growing rate, from 25% up to 40% depending on the industry sectors and regions in the world (Zhang et al., 2021). This trend requires efficient reverse logistics to manage these product returns, which is becoming an important issue for several reasons. First, in terms of sustainability, this return logistics, associated with unsatisfied customers, consumes energy, generates carbon dioxide emissions, and increases traffic congestion (Zhang et al., 2021). Then, return logistics represents huge costs for retailers and logistics service providers while efficient return flow management could strengthen their competitiveness and market share (Ambilkar et al., 2022). Finally, Ramanathan (2011) showed that handling product returns plays an important role in shaping customer loyalty.

A key step in reverse logistics is the collection and transportation of returned products from drop-off points. This step corresponds to the first-mile collection problem. Whereas the literature on last-mile delivery problem is extensive (e.g., VRP), the first-mile pickup problem has received little attention so far, even though these two closely related problems are not similar and have their own characteristics (see Giménez-Palacios et al., 2022; Zhang et al., 2021).

In this paper, we investigate the first-mile pickup of three-dimensional parcels dropped off by customers at different locations in an urban area. Hence, we face a combination of two well-known $\mathcal{NP}$-hard problems, namely, the Container Loading Problem and the Capacitated Vehicle Routing Problem, typically referred to as 3L-CVRP in the literature (Gendreau et al., 2006). It is crucial to design a solution method that simultaneously addresses the routing problem and the three-dimensional loading problem. Indeed, as explained in Côté et al. (2017), in many real-world freight transportation applications, not only the weight but also the dimensions of the items should be considered when designing the routes and related loading patterns since the box positions may have an impact on the feasibility of the packing. Therefore, a solution should consist of a list of routes,

each with an associated schedule and a loading pattern, i.e., a formal description of the position of each box in the assigned vehicle.

In order to identify the real-life issues faced by practitioners, a survey was conducted among some Belgian logistics service providers regarding their picking operations. It seems that they are mostly dealing with vehicle routing problems with time windows corresponding to regular opening hours and they often allow split pickups, i.e., a customer can be visited several times in a single day. Based on their limited homogeneous vehicle fleet, they collect the customer parcels by providing vehicle routes with tailored schedules and loading patterns while minimizing the global transportation distance. On the one hand, each schedule must comply with the customers' time windows, the drivers' maximum working duration (driving, loading, and waiting durations), and the depot's opening hours. On the other hand, each loading pattern must be valid at each customer location, i.e., it must satisfy a set of constraints (namely, geometric, vertical stability, orientation, and reachability constraints). Therefore, the problem can be considered as a three-dimensional loading vehicle routing problem with split pickups and time windows (3L-SPVRP-TW).

Ignoring traffic conditions and assuming constant travel durations throughout the day could lead to schedule failures, as it has been confirmed by the survey, especially when considering time constraints such as customer time windows or depot opening hours (e.g., Ichoua et al., 2003; Eglese et al., 2006; Kok et al., 2012). To get closer to a real-life context, we consider time-dependent travel times which assume that the travel duration between two locations varies over the day according to a time-dependent pattern, as delays occur during peak hours. The interest in considering time-dependent travel times has grown, as evidenced by the literature and a recent literature review on the subject (Adamo et al., 2024).

Since this problem is computationally hard to solve to optimality for realistic instances, in this work, we propose a three-phase heuristic to quickly obtain good-quality solutions. The first phase consists of building an initial solution. We compare adaptations of the sweep algorithm of Gillett & Miller (1974) and the savings heuristic of Clarke & Wright (1964). The second phase is an attempt to eliminate routes and is applied when the initial solution requires more vehicles than the carrier's fleet, hereafter called rented vehicles. As a third phase, a general variable neighborhood search (GVNS) is applied to improve the solution in terms of the transportation costs, based on the travel distance and the fixed costs associated with rented vehicles. The five neighborhoods are based on typical routing operators: crossover and relocate or swap within a single route (intra) or between

two routes (inter). In the three phases, we use the Extreme Points (EPs) defined in Crainic et al. (2008) and extended in Paquay et al. (2018) to construct or modify the loading patterns.

The main contribution of this work is threefold. To solve the 3L-SPVRP-TW, we design a three-phase heuristic that takes into account practical aspects leading to more realistic solutions, such as the time-dependency in the trip durations, as well as the concept of box reachability when loading the vehicle. Moreover, each of the three phrases is adapted to consider split pickups if this option is profitable for the service provider. In particular, we modify the well-known saving, sweep, and nearest neighbor heuristics to this purpose. Finally, the combination of these elements enables us to derive useful managerial insights for service providers offering a first-mile transportation service. In particular, we analyze the potential benefits of allowing split pickups and of extending the time windows on costs, vehicle loading rate as well as on driver working duration. We also assess the impact of considering reachability constraint instead of sequential loading constraint and of accounting for time-dependent travel durations.

The paper is organized as follows: Section 2 describes the first-mile pickup problem. Section 3 reviews the literature related to the 3L-CVRP. Section 4 describes the solution representation and feasibility, while Section 5 depicts the three-phase heuristic. Section 6 presents the instances and computational results, including managerial insights. Finally, Section 7 draws conclusions and suggests future research directions.

2. First-mile pickup problem description

A logistics service provider (LSP) owns a limited homogeneous fleet of small vehicles with a maximum weight capacity and loading space with given dimensions (i.e., length, width, and height). To satisfy the requests, the LSP can rent extra vehicles facing a rental cost. The rented vehicles are of the same type as the LSP vehicles.

This LSP has a number N of customers requiring a various number of cuboid boxes to be collected and transported from their locations to the central depot. Each box is also characterized by a length, which is the largest dimension of the box, a width, a height, and a weight. Each customer can be visited daily by at most two vehicles. All customer requests must be satisfied.

We assume that this information is available at the beginning of the day and is not subject to unexpected changes during the day, i.e., the LSP knows with certainty customer locations, time

windows, service times and box characteristics. More specifically, the service time s_i is fixed per customer i, regardless of the number of boxes. Due to the small number of boxes per customer, the loading time is considered negligible compared to the time for parking, calling the customer, going to the loading area, and opening the door of the vehicle.

Travel durations are assumed to be time-dependent, i.e., they vary throughout the day according to a predefined deterministic pattern. In this work, we consider the variations resulting from predictable or recurring events such as congestion during peak hours. Hence, the travel durations are computed based on the Euclidean distances between two locations by using the algorithm from Ichoua et al. (2003) in which the working day is divided into time slots, each with a given constant speed that can be different from one time slot to another. More precisely, the travel duration between locations i and j can be computed knowing the departure time D_i from location i as $t_{ij}(D_i)$ or knowing the arrival time A_j at location j as $t'_{ij}(A_j)$ (see Section 1 of the Supplementary Material). An important feature of this approach is that it respects the First-In First-Out (FIFO) property. The FIFO property guarantees that if a vehicle v leaves a location i for a location j at a given time, any other vehicle v' leaving the location i for location j after vehicle v will arrive at location j later than vehicle v. In Eglese et al. (2006) and Schilde et al. (2014), historical data is used to determine the number and size of the time slots. To avoid being case-specific, we choose to consider five slots as in Kok et al. (2012). We set two peak periods from 7:30AM to 9AM and from 4PM to 6PM. Off-peak periods take place before (from 6AM), between, and after (until 11PM) these two peak periods. As mentioned in Kok et al. (2012), this model has the drawback of considering sudden speed changes, which can be improved by increasing the number of time slots.

To minimize transportation costs, the LSP decides which vehicles leave the depot and, for those vehicles, determines their route, that is, the sequence of locations each vehicle will visit. He also schedules the operations, that is, the time when each driver arrives, serves a customer and then leaves. Moreover, at each loading point of each route, the LSP extends the associated loading pattern by the boxes to be newly positioned inside the vehicle. We assume that the cargo is not rearranged during the collection process, meaning that each box remains in the same location once loaded. The transportation costs are given as a linear combination of the total travel distance and the total rental cost of the extra vehicles, i.e.

$$\text{cost of the solution} = \text{total travel distance} + \text{rental cost} \times \text{number of rented vehicles}$$

The rental cost will be fixed as in Koch et al. (2018), i.e., 10 $\times$ maximum distance between two locations.

To apply these decisions in real life, the routes, schedules, and loading patterns should satisfy several sets of constraints. We specify hereunder the constraints considered in this model.

The first set of constraints relates to the **routing**:

- *One route per vehicle and driver:* vehicles start from and return to a single central depot at most once, resulting in a single trip per vehicle and related driver.
- *Multiple visits per customer:* each customer should be served by a maximum of two vehicles.
- *Consistency of arrival, departure and service times*: the arrival time at location $j+1$ is later than the departure time at location j, denoted D_j, plus the travel time $t_{j(j+1)}(D_j)$. Picking operations start after arrival at a location and end before the departure from that location.
- *Depot and customer time windows:* routes take place during the scheduling horizon, i.e., the opening hours of the depot $[a_0, b_0]$. The picking operations start and end within the customers' time windows $[a_i, b_i], i = 1, \ldots, N$. We assume here that the whole pickup operations should lie in the customer time windows since they correspond to opening hours of the businesses. If a vehicle arrives before the starting time a_i, it has to wait until a_i, and the waiting time is added to the duration of the route.
- *Maximum route duration:* each route duration (driving, loading, and waiting durations) does not exceed the maximum driver working duration T. Compulsory breaks are not taken into account as only small-sized vehicles are used in an urban context. Overtime is not allowed.

The second set of constraints arises from the **loading** problem encountered at each customer location (Bortfeldt & Wäscher, 2013):

- *Geometric* constraints: boxes cannot overlap and lie completely inside the vehicle.
- *Weight limit* constraints: the maximum weight capacity of the vehicle is respected. As vehicles are small, specific constraints related to truck legislation such as axle weight limits (a particular case of the weight distribution constraints) do not need to be taken into account.
- *Box orientation*: boxes are packed in such a way that their edges are orthogonal or parallel to the edges of the vehicle (referred to as orthogonality constraints). To avoid the boxes from

opening up, the vertical orientation ("This side up!") of the boxes is maintained, leading to only a possible 90-degree rotation of the boxes over the horizontal plane.

- *Cargo stability* to reduce box damages and potential injuries due to a falling box. *Vertical stability* (or static stability) prevents boxes from falling onto the vehicle floor or onto other boxes. In this work, vertical stability is ensured if the box is either on the floor or if its four corners are supported by other boxes, as shown in Figure 1. *Horizontal stability* (or dynamic stability) prevents the boxes from moving significantly while the vehicle is moving. In this research, we specify a minimum contact area for each corner when a corner of a box is supported by another box, as shown by the gray surfaces in Figure 1. The larger the contact area, the greater the frictional forces, and thus the more horizontally stable the packing. However, this approach can eventually be improved by adding shrinking foils or by using straps or filler material such as foam pieces (Bortfeldt & Wäscher, 2013).

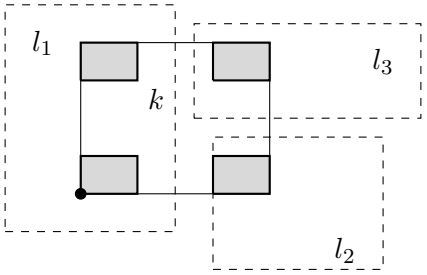

Figure 1: Example of four corners of a box k supported by boxes l_1, l_2 and l_3 (dashed lines), with the minimum contact area in gray (top view)

- *Sequential loading* (or multi-drop or LIFO) constraints in order to avoid unloading/reloading efforts (e.g., when a customer is visited, his boxes cannot be packed below those of previous customers in the route). In this work, we use the concept of *reachability* introduced in Junqueira et al. (2012) and Ceschia et al. (2013). This concept states that a box is "reachable" if the distance between its front face and the driver (who is loading it) is smaller or equal to a predefined length, typically the driver's arm. This prevents the driver from standing on items to reach other items during unloading operations and substitutes the LIFO policy.

In this work, we do not consider fragility constraints since our survey revealed that LSPs require their customers to provide secure shipping packaging to prevent damage to the boxes.

3. State of the art of 3L-CVRP

The three-dimensional loading capacitated vehicle routing problem (3L-CVRP) was first studied in Gendreau et al. (2006) and has received much attention since then. A first literature review is proposed in Iori & Martello (2010) regarding the first 36 papers on both two and three-dimensional loading CVRP and related problems (e.g., multi-compartment, multi-pile). A few years later, Pollaris et al. (2015) present an updated review, covering 76 more recent articles. The authors describe the main characteristics of VRP, based on Toth & Vigo (2002), and of CLP, based on Bortfeldt & Wäscher (2013). Let also mention Vega-Mejía et al. (2019) that present a systematic literature review of the integrated problem from a sustainability perspective. Since 2015, about 100 new articles have been published on the topic. In the following, we focus our analysis on the articles that tackle the problems closest to ours.

3L-CVRP. Gendreau et al. (2006) introduced the 3L-CVRP for a homogeneous fleet of capacitated vehicles. They developed a tabu search algorithm that deals with geometric, 90-degree rotations, fragility, vertical stability and sequential loading constraints. The 3L-CVRP is $\mathcal{NP}$-hard, thus exact approaches struggle to solve large instances (e.g., the mathematical model of Junqueira et al. (2013) solved instances up to 15 customers and 32 boxes). Therefore, a significant part of the literature has proposed heuristic approaches to solve the 3L-CVRP (e.g., ant colony optimization algorithm as in Fuellerer et al. (2010), tree search algorithm as in Bortfeldt (2012), evolutionary local search as in Zhang et al. (2015)).

3L-CVRP with time windows. In logistics, customers may require to be served during a given time window. Moura & Oliveira (2009) studied the 3L-CVRP variant considering time windows (3L-VRP-TW). In their article, they proposed two resolution methods: one based on a sequential approach and one based on a hierarchical approach. In the hierarchical approach, vehicle routing is the main problem and container loading is the sub-problem, while in the sequential approach, the two problems are treated at the same level, but the sets of constraints related to the single visit and the sequential loading are relaxed. Bortfeldt & Homberger (2013) presented a packing first, routing second heuristic to solve the 3L-VRP-TW that outperforms the approaches of Moura & Oliveira (2009). To solve the 3L-VRP-TW, Zhang et al. (2017) proposed a tabu search combined with an artificial bee colony algorithm. Moura (2019) presented a mathematical formulation for the 3L-VRP-TW that considers heterogeneous vehicles, but it does not take into account the stability (neither

vertical nor horizontal), the potential fragility of the boxes, or the sequential loading constraints. The results obtained with this exact method, with a maximum computation time of 12 hours, can be used to compare the performance of heuristic approaches. Moura (2019) also proposed a hybrid approach consisting in grouping the customers according to their location and solving a smaller 3L-VRP-TW. Krebs et al. (2021) presented an adaptive large neighborhood search for the routing part and a Deepest-Bottom-Left-Fill (DBLF) algorithm for the packing part of the 3L-VRP-TW. The DBLF algorithm, presented by Karabulut & İnceoğlu (2005), loads boxes one at a time by packing the current box into the deepest, bottom-most, leftmost location. Krebs et al. (2023) extended their work by presenting three variants for the DBLF and comparing their performance in terms of travel distance and average runtime. Chi & He (2023) addressed the 3L-VRP-TW from a pickup point of view and developed a branch-and-price (B&P) algorithm to solve it. Giménez-Palacios et al. (2022) are the first to tackle dynamism in a three-dimensional loading VRP with pickups, meaning that routes are modified based on incoming real-time information. The initial plan is obtained in two phases: first a genetic algorithm to minimize the number of vehicles, and then a local search algorithm to minimize the total travel distance.

3L-CVRP with split loads. Occasionally, a mix between 3L-CVRP and split-delivery VRP is studied. The problem is called the three-dimensional loading Vehicle Routing Problem with split deliveries (3L-SDVRP). Ceschia et al. (2013) proposed a simulated annealing and a large-neighborhood search for the 3L-SDVRP without time windows but including the reachability constraint. However, their results do not prove any advantage of allowing split deliveries. Bortfeldt & Yi (2020) considered two variants of the 3L-SDVRP: one where split is forced, i.e., a customer's request is split only if it cannot be loaded in a single vehicle with their tailor-made packing algorithm, and one where split is optional, i.e., one vehicle could contain the total request of a customer, but a split is made if it is profitable. To solve both variants, they presented a packing first, routing second approach. A genetic algorithm and constructive heuristics are used to solve the packing problem, while a local search approach is developed to solve the routing problem. Contrary to Ceschia et al. (2013), they proved that the travel distance and thus the cost can be saved by considering split deliveries. Chen et al. (2020) considered a 3L-SDVRP with time windows and proposed a tabu search algorithm for the routing part and a DBLF algorithm for the packing part. Allowing split delivery enables to respect the time constraints and reduces on average the total travel distance by 3%.

A summary is presented in Table 1. Moreover, information about the benchmark instances proposed

in those articles can be found in Table A.4.

Table 1: Review of the three-dimensional loading VRP

	Authors	Objective function	Homogeneous fleet	Time Windows	Time dependency	Split	Geometric and weight capacity	Vertical stability (Area=A, corner=C)	Horizontal stability (partially=P)	Box orientation (90^0-rotations=H)	Sequential loading	Fragility	Solution method
3L-CVRP	Gendreau et al. (2006)	min costs	X				X	A		H	X	X	Tabu search
	Fuellerer et al. (2010)	min costs	X				X	A			X	X	Ant colony
	Bortfeldt (2012)	min total distance	X				X	A		H	X	X	Hybrid algorithm
	Junqueira et al. (2013)	min costs	X				X	A			X	X	Exact
	Zhang et al. (2015)	min fuel consumption	X				X	A		H	X	X	Evolutionary algorithm
3L-VRP-TW	Moura & Oliveira (2009)	min # routes (1) min total distance (2)	X	X		X	X	A	X	H			Sequential
	Moura & Oliveira (2009)	min # routes (1) min total distance (2)	X	X			X	A	X	H	X		Hierarchical
	Bortfeldt & Homberger (2013)	min # routes min total distance	X	X			X	A		X	X	X	Packing first routing second
	Zhang et al. (2017)	min total distance	X	X			X	A		H	X	X	Hybrid algorithm
	Moura (2019)	min distance min # vehicles		X			X			X			Hybrid algorithm Matheuristic
	Krebs et al. (2021)	min total distance min # vehicles	X	X			X	X		H	X	X	ALNS
	Giménez-Palacios et al. (2022)	min # veh (1) min total distance (2)		X			X	A		X	X		Genetic algorithm & local search
	Krebs et al. (2023)	min total distance	X	X			X	A		H	X	X	ALNS
	Chi & He (2023)	min costs	X	X			X	A		H	X	X	B&P
3L-SDVRP	Ceschia et al. (2013)	min costs				X	X	A		H	X	X	Local search
	Bortfeldt & Yi (2020)	min # routes (1) min total distance (2)	X			X	X	A		H	X	X	Hybrid algorithm
	Chen et al. (2020)	min # routes (1) min total distance (2)	X	X		X	X	A			X		Tabu search
	This work	min costs	X	X	X	X	X	C	P	H	X		Three-phase

This state of the art highlights some gaps in the literature. Firstly, most of the articles focus on the delivery of the parcels and not on their collection. In three dimensions, this has an impact on some constraints as highlighted in Chi & He (2023). Among these, to prevent reloading efforts, the sequential loading and stability constraints must be adapted since previously loaded boxes cannot be stacked on the boxes to be loaded later. Secondly, we can observe that most of the papers do not consider the possibility of splitting the collection or delivery, which makes the problem even more challenging. Finally, none of them integrates the time-dependent travel durations.

4. Solution representation and feasibility

A solution to the 3L-SPVRP-TW is a list of routes $\mathcal{R}$, where a route is an ordered sequence of customers $R = (0 - r_1 - \cdots - r_i - \cdots - r_n - 0)$, 0 being the depot, r_i the i^{th} customer visited in route R, and n the number of customers in route R. Each route has an associated schedule of activities and a loading pattern for the loaded boxes of the visited customers.

A solution is feasible if all customers have their boxes collected by at most two vehicles and if all routes are feasible. A given route is declared as feasible if there exists a feasible schedule (Section 4.1) and if a feasible loading pattern can be generated for this route using our packing heuristic (Section 4.2). Schedule feasibility is checked first due to its short computation time.

4.1. Schedule generation and feasibility check

For a given route, a schedule is defined for each visited customer i by three time instants: the vehicle's arrival time A_i, the start service time S_i (box collection), and the departure time D_i.

A feasible schedule should satisfy the consistency of arrival, departure and service times, the depot and customer time windows, as well as the maximum route duration as explained in Section 2.

The determination of a feasible schedule for a given route is based on the concepts of earliest/latest arrival/departure times of Savelsbergh (1992) and Mitrović-Minić & Laporte (2004). For each customer i, $i \in \{1, ..., n\}$, in the route, the earliest arrival time $\underline{A}_i$ at customer i, i.e., the earliest time the driver can reach customer i after visiting all customers preceding i within their time windows, and the earliest departure time $\underline{D}_i$ from customer i, i.e., the earliest time the driver may leave customer i knowing $\underline{A}_i$, are computed via a forward pass as detailed in (1). Similarly, the latest departure time $\overline{D}_i$ from customer i, i.e., the latest possible time to leave customer i in order to be on time at all following customers and at the depot, and the latest arrival time $\overline{A}_i$ at customer i, i.e., the latest time to reach customer i knowing $\overline{D}_i$, are computed via a backward pass as detailed in (2).

$$(1) \quad \begin{cases} \underline{D}_0 = a_0 & \\ \underline{A}_i = \underline{D}_{i-1} + t_{(i-1)i}(\underline{D}_{i-1}) & i = 1, ..., n \\ \underline{D}_i = \max\{\underline{A}_i, a_i\} + s_i & i = 1, ..., n \\ \underline{A}_0 = \underline{D}_n + t_{n0}(\underline{D}_n) & \end{cases} \qquad (2) \quad \begin{cases} \overline{A}_0 = b_0 & \\ \overline{D}_n = \overline{A}_0 - t'_{n0}(\overline{A}_0) & \\ \overline{A}_i = \min\{\overline{D}_i, b_i\} - s_i & i = 1, ..., n \\ \overline{D}_i = \overline{A}_{i+1} - t'_{i(i+1)}(\overline{A}_{i+1}) & i = 0, ..., n-1 \end{cases}$$

A feasible schedule with respect to depot and customer time windows, and time consistency can

then be found, by fixing the arrival and start service times to $\max\{\underline{A}_i, a_i\}$ and the departure time to $\underline{D}_i$, as long as:

$$\max\{\underline{A}_i, a_i\} \leq \overline{A}_i \quad \text{and} \quad \underline{D}_i \leq \min\{\overline{D}_i, b_i\} \quad i = 0, ..., n \tag{3}$$

where $X \leq Y$ means that X is earlier than Y.

The iterative approach of Mitrović-Minić & Laporte (2004) allows to consider time-dependent transportation durations but cannot consider a maximum route duration constraint. Therefore, based on the earliest/latest arrival/departure times, we identify the two extreme schedules that are likely to minimize the route duration by removing the slack at the end or beginning of the day and providing a tight schedule over the day. If the duration of both of these schedules exceeds the maximum duration, this route is discarded. These two extreme schedules correspond to two policies, i.e., a strategy to determine arrival A_i, start service S_i and departure D_i times: either we decide to visit and serve the last customer of the route as early as possible and then to serve all previous customers of the route as late as possible on this basis, as detailed in (4), or we decide to visit the first customer as late as possible and then to serve and leave as early as possible, as detailed in (5).

$$(4) \quad \begin{cases} S_n = A_n = \max\{\underline{A}_n; a_n\} & \\ D_n = S_n + s_n & \\ D_i = A_{i+1} - t'_{i(i+1)}(A_{i+1}) & i = 1, ..., n-1 \\ S_i = A_i = \min\{D_i; b_i\} - s_i & i = 1, ..., n-1 \end{cases}$$

$$(5) \quad \begin{cases} S_1 = A_1 = \overline{A}_1 & \\ D_i = S_i + s_i & i = 1, ..., n \\ A_i = D_{i-1} + t_{(i-1)i}(D_{i-1}) & i = 2, ..., n \\ S_i = \max\{a_i; A_i\} & i = 2, ..., n \end{cases}$$

In both policies, to avoid waiting time, the departure time from the depot is such that we arrive at the start service time of the first customer in the route $D_0 = S_1 - t'_{01}(S_1)$ and the arrival time at the depot at the end of the route is such that we leave the last customer at his departure time $A_0 = D_n + t_{n0}(D_n)$.

In the policy (5), the driver waits before providing service (drive and wait strategy), while in the policy (4), the driver waits at the current destination before departing and arrive directly on time to serve the next customer (wait and drive strategy).

Finally, we compute the route duration $A_0 - D_0$ for both policies and check whether this duration is smaller or equal to the maximum duration T. In order to determine one specific schedule, the policy leading to the smallest route duration is selected.

Figure 2 summarizes the process to check the existence of a feasible schedule for a given route R.

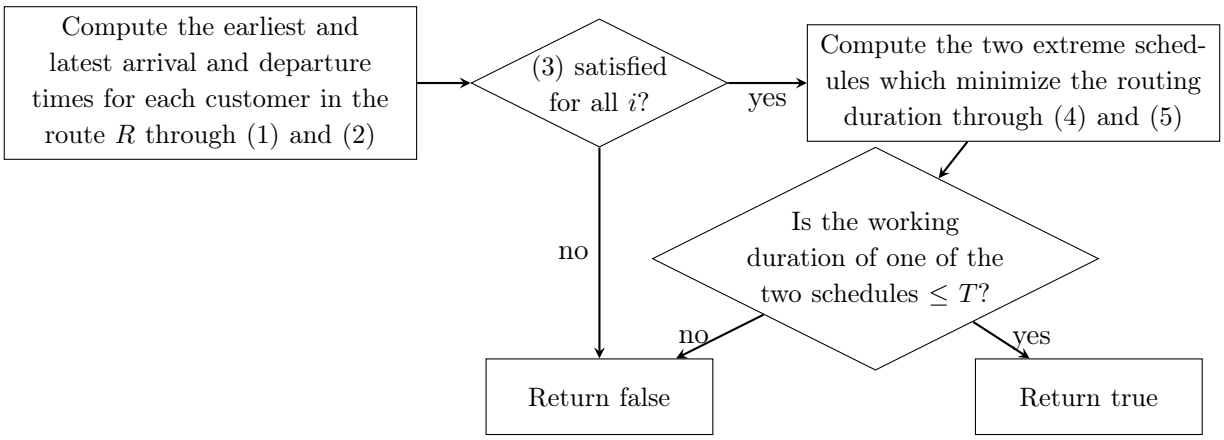

Figure 2: Checking the existence of a feasible schedule for a route R - `checkExistenceFeasibleSchedule(`R`)`

4.2. Loading feasibility

If the sequence of customers visited in a route is feasible with respect to routing constraints, we then check whether all the boxes could be accommodated in the vehicle, given the sequence of customers visited as we do not allow repacking along the way. We thus design a packing algorithm aiming to provide a feasible packing, if possible. Before creating the loading pattern associated to a route, we first need to decide the loading sequence of the boxes of a given customer.

4.2.1. Loading sequence

In pickup operations, the boxes of the customers are loaded customer by customer according to the route sequence. We then need to define a loading sequence of the boxes for each customer since testing the loading of all possible permutations of boxes is time-consuming. For each customer's boxes, we consider two possible sorting operators. On the one hand, the *sortSurfLength* operator sorts first by the largest surface of the bottom face first, and as a tiebreaker, by the largest length and then by the largest width. On the other hand, the *sortRotSurf* operator first sorts boxes that cannot be rotated due to their dimension (box length larger than vehicle width), and if tied, by the largest surface of the bottom face and then by the largest length. If all boxes can be rotated, then the two operators provide the same sequence.

The box loading sequence given by the customer sequence eases checking the reachability as well as the cargo stability constraint (all boxes that support a given box of customer j must also belong to j or to customers that are visited before customer j).

In the constructive heuristics (see Sections 5.2.1, 5.2.2 and 5.2.3), each customer has an associated

sorting operator for his boxes. The loading is first tested with the *sortSurfLength* operator and without splitting the request. If no feasible packing can be found, the *sortRotSurf* operator is applied without split. If there is still no feasible loading pattern, meaning that the whole set of boxes of a customer cannot be loaded in the vehicle using our packing heuristic, then that customer's request is split, and the vehicle visits that customer and loads as many boxes as possible using the *sortSurfLength* sort operator. The remaining boxes then become a request to be served with the parameter *alreadySplit* to "true", where the parameter *alreadySplit* is associated with a customer and indicates whether his request is already split into two subsets and thus, cannot be split anymore. This parameter is defined in more detail in Section 5.

4.2.2. Loading pattern construction and feasibility check

So far, we have a given sequence of customers to be visited, as well as a sequence of boxes to pack for each of those customers.

To describe the loading pattern inside a vehicle, we introduce a coordinate system and use the front (smallest x-value) left (smallest y-value) bottom (smallest z-value) corner (x_k, y_k, z_k) and the rear right top corner (x'_k, y'_k, z'_k) to describe the precise location of the box k, as shown in Figure 3.

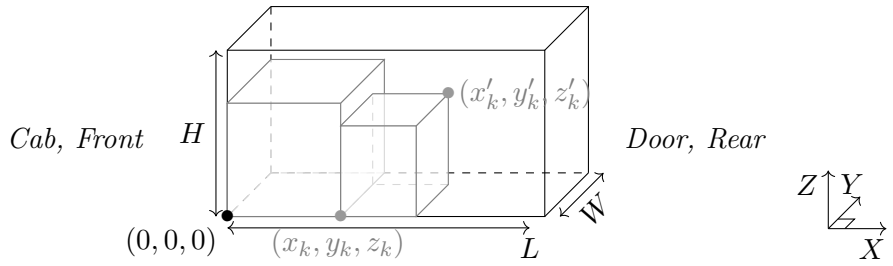

Figure 3: The coordinate system associated with a vehicle and the coordinates of a box k

In the following, we explain first how to generate the potential locations for the front left bottom corner of a box, how to identify those that are suitable for a given box and a given orientation, and finally, how to select the best suitable location for the corner in the loading pattern.

Potential locations generation. We generate the potential locations as in Paquay et al. (2018), who extended the Extreme Point (EP)-based method developed in Crainic et al. (2008) to pack boxes inside a three-dimensional container. This method is independent of the particular packing problem addressed and is widely used in the literature (e.g., Bortfeldt (2012); Gzara et al. (2020); Meliani et al. (2022)). The EPs are the relevant potential locations for the front left bottom corner of a box

to be loaded, exploiting the free residual space inside a vehicle.

Suitable locations identification. This list of EPs provides potential locations for accommodating the new box to be packed, but not all of them would lead to a feasible packing. Therefore, an EP is declared suitable for a box with a specific orientation if the loading constraints, namely, the geometric, vertical stability, orientation, and reachability constraints defined in Section 2, are satisfied. Regarding the latter, we defined reachability in the case of multiple boxes per customer by considering as obstacles only the boxes of previous customers and the boxes of the same customer that are located below. In Figure 4, the reachability of box C considers only the blue box from the first customer and box A, since box B could be loaded after box C of the second customer. For the orange box from the potential third customer visited in the route, all boxes should be considered to determine the reachability. If the distance is larger than the driver's arm, then this box cannot be loaded in this vehicle and the orange customer cannot be visited in the route (provided there is no other place for his box).

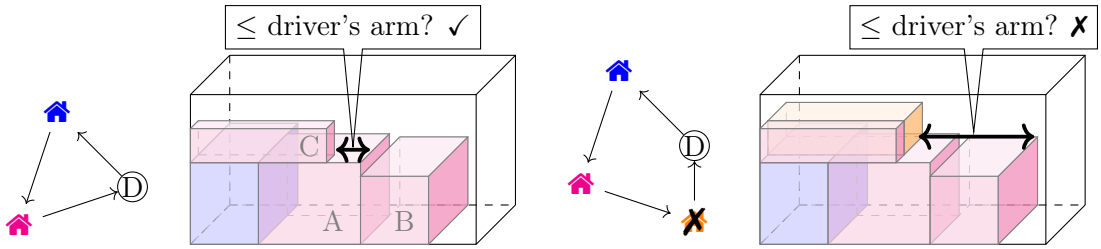

Figure 4: On the left-hand side, the vehicle can accommodate the boxes of the blue and the pink customers while respecting the reachability notion. On the right-hand side, the customer in orange cannot be added to the route since the position to load his box is not reachable.

Best location and orientation selection. Thus, we end up with a list of suitable EPs and their associated orientation for the new box among which we still need to select the best one. In this work, we considered the Top-Deepest-Left-Fill (TDLF) heuristic rule, which loads boxes one at a time by packing the current box in the top-most, deepest, leftmost location from the loader's perspective. This selection criterion is the most promising one as it tends to make the packing as compact as possible at the rear of the loading space (corresponding to the front of the vehicle), which will increase the number of EPs that are likely to satisfy the reachability constraint. This choice is statistically verified in Section 6.4.2. In case of a tie on the EP location, meaning that the box can lie in the same location regardless of its orientation, then we choose the orientation with the box length along the X-axis (Figure 3).

Potential locations update. Finally, we place the front left bottom corner of this box according to the "best" EP, "best" orientation, remove the EPs covered by the box, and generate the new EPs.

In the following, a loading pattern is defined as a list of boxes, each with the position of its front left bottom and rear right top corners, as well as the list of extreme points generated so far.

Algorithm 1 shows how to load a box in a loading space and determine the best EP and orientation for this box. Moreover, it ensures that loading constraints are met by testing whether an EP is suitable.

Algorithm 1: Attempt to add a box k to a loading pattern P – `attemptToAddOneBox`(P,k)

Input: A loading pattern P, where $\mathcal{E}$ denotes the list of EPs, and a box k to be loaded
Output: The updated loading pattern if P can accommodate k, initial P otherwise
Sort the list of EPs $\mathcal{E}$ by Top-Deepest-Left-Fill;
$l \leftarrow 1$;
$P' \leftarrow P$;
while $l \leq$ *number of EPs in* $\mathcal{E}$ **do**
  Let e be the l^{th} EP in $\mathcal{E}$;
  if *the EP e is not suitable for any orientations of box k* **then**
    $l \leftarrow l + 1$;
  else
    if *the EP e is suitable for the box k for both orientations* **then**
      Let the orientation of k be the one with the box length along the X-axis;
    else
      Let the orientation of k be the one for which the EP is suitable;
    end
    Update P by placing the front left bottom corner of box k at location e and updating the list of EPs $\mathcal{E}$;
    return P;
  end
end
return P';

5. Three-phase heuristic

As shown in Figure 5, the proposed heuristic method is based on three distinct phases: (I) a constructive phase to generate an initial solution (Section 5.2), (II) a route elimination procedure to reduce the number of rented vehicles, if any (Section 5.3), and (III) a general variable neighborhood search (GVNS) to minimize the transportation costs (Section 5.4).

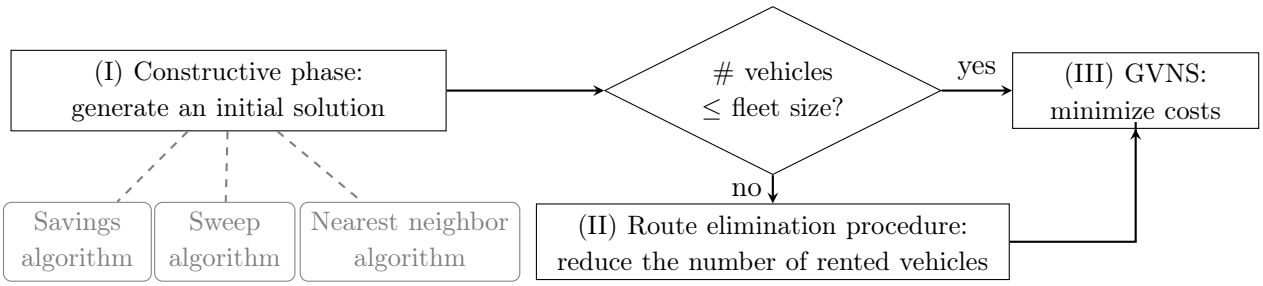

Figure 5: Diagram of the three-phase heuristic

Throughout these three phases, only feasible routes with respect to the routing and loading constraints as described in Section 4 are taken into account in the solutions.

5.1. Customer insertion

In the three phases, the heuristics will try, at one point of the process, to insert a customer at a given position in a route, with all or only a subset of his boxes. We thus need to ensure the feasibility of the routing and the loading, as explained in the previous section. The following algorithms describe how we deal with these insertions.

Algorithm 2 tries to insert a customer at a given position of a given route. It employs the procedure described in Figure 2 to ascertain the feasibility of a schedule when customer i is inserted into route R at position *pos*. As a reminder, this procedure accounts for the maximum working duration and time dependency. The parameter *splitAllowed* is set to "true" if the insertion attempt occurs at the end of the route to avoid high computation time due to repacking of subsequent customers, and if the request has not already been split earlier in the optimization process (parameter *alreadySplit*, introduced in Section 4.2.1). The split is valid if the subset of boxes that cannot be accommodated in the current vehicle can be accommodated in an empty vehicle. A copy of the customer i with the subset of boxes that cannot be accommodated in the current vehicle, called i' in the Algorithm 2, is then added to the list of customers to be inserted in the solution later. To avoid unnecessary packing calculations, if the request cannot be split, we first check the maximum weight capacity constraint.

Algorithm 2: Attempt to insert fully or partially customer i at position pos in route R - `attemptToInsert`$(R, i, pos, alreadySplit_i)$

Input: A route R (that is, the associated list $\mathcal{L}$ of customers and the loading pattern P), a customer i to be inserted at position pos in R, and a boolean $alreadySplit_i$ stating if the request of customer i was already split

Output: The route R with customer i inserted if feasible, initial route R otherwise, and a potential new customer i' with the subset of boxes of customer i that cannot be accommodated in the current vehicle

```
boolean splitAllowed;
if pos=|L| + 1 and alreadySplit_i is false then
    splitAllowed ← true;
else
    splitAllowed ← false;
    if weight of boxes from (L ∪ i) > maximum weight capacity then
        return (R, ∅)
    end
end
Let R' be the route R with the customer i inserted in position pos;
if checkExistenceFeasibleSchedule(R') is true then                    // see Figure 2
    P' ← attemptToLoad(R, i, pos, splitAllowed) ;                     // see Algorithm 3
    if splitAllowed is false then
        if all boxes of i are in the loading pattern P' then
            Insert customer i with all his boxes to the route R at position pos, update
             schedule information, and update the loading pattern (P ← P');
        end
    else
        if at least one box of customer i is in the loading pattern P' then
            Create a copy i' of customer i with the boxes of i that are not loaded;
            alreadySplit_i' ← true ;
            P'' ← attemptToLoad(∅, i', 0, false) ;                    // see Algorithm 3
            if all boxes of customer i' are in the loading pattern P'' then
                Insert customer (i\i') in route R at position pos, update schedule
                 information, and update the loading pattern (P ← P');
                alreadySplit_(i\i') ← true;
                return (R, i');
            end
        end
    end
end
return (R, ∅);
```

Algorithm 2 relies on Algorithm 3 to generate a loading pattern, which incorporates the loading constraints through Algorithm 1 from Section 4.2.2. Algorithm 3 iteratively loads one box after another (from the customer i and also from subsequent customers in the route) until either all boxes are loaded, or a part of boxes from the customer i, if his request can be split, and all boxes of

subsequent customers are loaded. The attempt fails if one box of a subsequent customer cannot be loaded, or a box of customer i cannot be loaded and his request cannot be split.

Algorithm 3: Attempt to generate a loading pattern after insertion of customer i in route R at position *pos* – `attemptToLoad`$(R, i, pos, splitAllowed)$

Input: A route R (that is, the associated list $\mathcal{L}$ of customers and the loading pattern P), a customer i to be inserted at position *pos* in R, and a parameter *splitAllowed* indicating whether the request from customer i can be split

Output: The updated loading pattern P' if i can be inserted in route R, initial P otherwise

Let P' be the loading pattern after visiting the customer at position $pos - 1$, if any, in route R;

Let K be the sequence of boxes from customer i and subsequent customers in the route after position *pos*, sorted according to the route and box loading sequences (Section 4.2.1) ;

for *each box k in K* **do**

  $P'' \leftarrow$ `attemptToAddOneBox`(P', k) ; `// see Algorithm 1`

  if P'' *is* P' **then** `// box` k `is not in the loading pattern`

    if *box k is from a subsequent customer or splitAllowed is false* **then**

      return P;

    end

  else `// box` k `is in the loading pattern`

    $P' \leftarrow P''$;

  end

end

return P';

5.2. Constructive heuristics

While reviewing the heuristics for the Vehicle Routing Problem (VRP), Laporte et al. (2000) proposed to classify the constructive heuristics, used to build an initial solution, into two main categories: either merging routes via a savings criterion (e.g., the savings heuristic of Clarke and Wright) or inserting customers into routes via an insertion cost (e.g., the sweep algorithm, the nearest neighbor algorithm).

In this section, we describe how we design the savings algorithm (Section 5.2.1), the sweep algorithm (Section 5.2.2), and the nearest neighbor algorithm (Section 5.2.3) to fit our specific problem, in particular the possibility to split the requests. These algorithms have the advantages of being intuitive, easy to implement, and fast (Toth & Vigo (2014)). They all assume that the number of vehicles is unlimited. If the solution found requires more vehicles than the fleet size, one or more extra vehicles are rented and a route elimination procedure is then applied to the complete initial solution as explained in Section 5.3.

5.2.1. Savings algorithm

In Clarke & Wright (1964), the authors presented a savings algorithm, applicable to non-symmetric cost functions, for assigning customers to vehicles and generating routes. Two versions of the algorithm exist: a parallel one in which several routes are generated simultaneously and a sequential one in which routes are generated one after the other, i.e., a route is created and extended as long as the constraints are met, then a new route is created and so on (denoted respectively Parallel savings and Sequential savings in Figure B.13). Laporte et al. (2000) showed experimentally that the parallel version outperforms the sequential one, which is also verified in our case in Section 6.4.1.

Thanks to its simplicity, this algorithm has been widely used for decades to solve vehicle routing problems and some of its extensions (see Rand, 2009, for a survey), and to provide an initial solution to other $\mathcal{NP}$-hard problems such as the 3L-CVRP (see Gendreau et al., 2006; Zhang et al., 2015; Krebs et al., 2023).

In this work, when generating the trivial routes, i.e., depot-customer-depot, the loading is tested as explained in Section 4.2. When a request is split, both subrequests have the same sorting operator, that is, the *sortSurfLength* operator. Each customer thus has an associated sorting operator for his boxes based on the trivial route that he will keep along the next phases of the heuristic.

The savings heuristic is then applied, i.e., the merging of routes using Algorithm 2, based on the largest savings computed on the travel distances. When trying to merge two routes R_1 and R_2, we consider two cases: either the second route R_2 visits only one customer or it visits more than one customer. In the former, the customer from the route R_2 is inserted at the end of the route R_1 if the resulting route is feasible. In the latter, we try to insert the customers one by one by setting the *alreadySplit* parameter to "true" since we do not allow split in that case.

5.2.2. Sweep algorithm

Gillett & Miller (1974) presented the sweep algorithm, which consists of representing the depot at the origin of a polar coordinate system and the pickup nodes with polar coordinates (radius r and polar angle θ). Sorted by increasing polar angle, customers are added one by one to a route until adding the next customer exceeds the maximum weight capacity of the vehicle. Then, this route is closed, a new one is opened, and the process is repeated. As suggested by the authors, this solution may be improved in terms of total distance by solving the traveling salesman problem for each route. This method belongs to the cluster-first routing-second algorithms and is commonly used in routing

problems (see Dondo & Cerdá, 2013; Thammano & Rungwachira, 2021; Vangipurapu & Govada, 2021).

As it is based on a geometrical insight but not simultaneously considering space and time, cluster-first route-second algorithms may request additional vehicles while taking into account time windows since the schedule may not be feasible (see left-hand side of Figure 6). Consequently, we adapt this generic algorithm by sweeping over all customers before closing the route and opening the next one (see right-hand side of Figure 6). Each time, we try to insert a customer at the end of the route under construction using Algorithm 2.

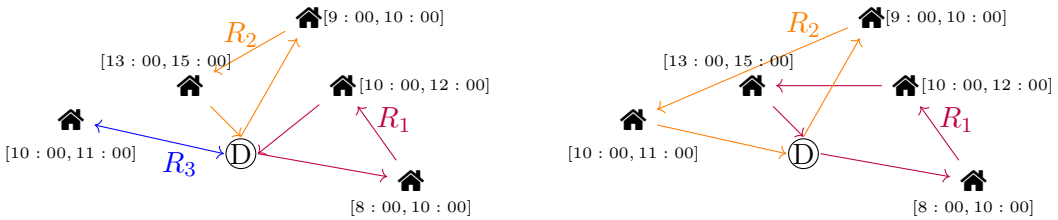

Figure 6: On the left-hand side, the classical application of the sweep algorithm requires 3 vehicles. On the right-hand side, the adaptation by sweeping over all customers before closing the route allows the reduce the number of routes to 2.

A second adaptation of the sweep is obtained by setting the polar axis to the location of the earliest remaining customer, i.e., the customer whose time window starts the earliest and, in case of a tie, whose time window range is the shortest, and then considering the remaining customers by increasing polar angle.

In these two adaptations of the sweep algorithms (denoted Sweep #1 and Sweep #2 in Figure B.13), the sequence of operators used to sort the boxes when loading them to the vehicle is the one explained in Section 4.2.1.

These two adaptations focus only on the clustering part of the sweep algorithm. The improvement of the transportation distance (routing-second) is addressed in the GVNS described in Section 5.4.

5.2.3. Nearest neighbor algorithm

The nearest neighbor algorithm (denoted Nearest Neighbor in Figure B.13) avoids grouping spatially dispersed customers, as the sweep tends to do, by taking distances into account when generating the routes.

In this work, we adapt the nearest neighbor algorithm by sequentially starting each route with the earliest remaining customer and then trying to insert the nearest unvisited customers one by one

using Algorithm 2 until no other customer can be added to the route. The sequence of operators to sort the boxes when loading them into the vehicle is the one explained in Section 4.2.1.

5.3. Route elimination procedure

The set of routes provided at the end of the constructive phase may exceed the LSP's fleet size, resulting in rental costs. Therefore, a route elimination procedure is applied to reduce the number of rented vehicles required until either the number of routes equals the fleet size or we have tried to modify each route and no more reduction in the number of vehicles is possible.

Routes are first sorted by non-decreasing number of visited customers to maximize the chance of reassigning all customers to other routes. We try to empty routes one at a time, starting from the last customer, to keep the packing feasible. For each customer to be removed, we attempt to reassign it to another route from the first to the last position using Algorithm 2. The routes that could accommodate the customers to be reassigned are considered by decreasing number of customers (we try to fill in the most loaded vehicles). As a first step, we try to reassign whole customer requests, i.e., without splitting them. At the end of this step, if there are still rented vehicles, then the process is repeated, while allowing split.

At the end of this phase, we keep the solution that minimizes the number of vehicles and, in case of a tie, the total travel distance.

5.4. General Variable Neighborhood Search

The third phase aims at reducing the total cost by applying a General Variable Neighborhood Search (GVNS), which is described in Section 2 of the Supplementary Material. The algorithm is composed of two phases: a shaking phase in which neighbors are randomly selected and a variable neighborhood descent (VND) phase in which the best neighbor is selected. The GVNS is a particular case of the Variable Neighborhood Search, which was introduced by Mladenović & Hansen (1997). Hansen et al. (2019) presented variants and applications. This metaheuristic and its variants have been used, alone or hybridized, inter alia, to solve VRP (see Kytöjoki et al., 2007) and VRPTW (see Bräysy, 2003; Ronconi & Manguino, 2022). They have the advantage of requiring few parameters to be tuned (the initial solution $\mathcal{R}$, the number of neighborhoods in the shaking and in the VND phases, and parameters related to stopping conditions) and of producing good-quality solutions by alternating between an intensification phase, by improving the solution, and a diversification phase, by selecting a random neighbor (shaking phase).

In this work, the stopping conditions consist in repeating the process up to a maximum number of iterations, given a maximum time and a maximum number of iterations without improvement.

We consider the same sets of neighborhoods for the shaking phase and the VND. We consider five moves, displayed in Table 2: two intra-route moves and three inter-route moves. One common feature of these five neighborhoods that makes them appropriate for satisfying the customers' time windows is that each move does not require reversing the direction of any part of the route. In order to determine in which sequence to apply the neighborhoods in the shaking and VND phases, the software package `irace` developed and presented by López-Ibáñez et al. (2016) was used. `irace` provides an automatic configuration tool for tuning optimization algorithms, that is, automatically finding the most appropriate combination of values for the parameters of an algorithm given a set of instances of a problem. The advantage of using `irace` is its relevance compared to random manual tuning. After some tests, `irace` was not able to disregard some configurations. Consequently, we randomly selected one. The sequence of moves is: intra-swap, crossover, inter-swap, inter-relocate and intra-relocate.

Table 2: Moves

Name	**Description**
Intra-relocate	Relocate one customer to another place in the same route
Intra-swap	Exchange the positions of two customers from the same route
Inter-relocate	Relocate one customer within another route
Inter-swap	Exchange the positions of two customers from different routes
Crossover	Exchange the terminal segment of two different routes

The existence of a feasible schedule is checked via Figure 2 and the construction of a loading pattern is achieved by loading boxes, starting with the customer's boxes at the smallest route modification position.

In the process, a route may become empty. In this case, the vehicle is no longer needed and the rental cost, if any, is reduced. In addition, applying the crossover operator to an empty route would result in splitting the other route into two routes.

Some preconditions exist to do some moves in order to obtain a feasible solution. First, there are typical preconditions such as the necessity of having at least two customers in a route to perform an intra-move. Second, as we are considering split, we need to check that the customer that will be moved is not already in the other route or will also be moved (inter-swap) or is the consecutive customer, leading to gathering his requests. In the case of a customer visited twice in a row on the same route, we gather his boxes and sort them according to the *sortSurfLength* operator described

in Section 4.2.1.

6. Computational experiments

This section first describes the instance sets used to validate the three-phase heuristic and its variants and then presents the computational experiments and their results. The instances of our problem (Section 6.1.2) as well as results are available on Leloup et al. (2024).

All tests were performed on a laptop with 32 GB of RAM and an Intel core i7 with 2 CPUs running 64-bit Windows 10 Pro. The codes were implemented in Java.

6.1. Instances

We consider two instance sets: the first one to assess the effectiveness of the heuristic presented in this work, the second one with the characteristics of the problem at hand to obtain some managerial insights.

6.1.1. Validation instances

Although the main objective of our work is to solve the 3L-SPVRP-TW, we first test the three-phase algorithm to assess whether it provides good-quality solutions compared to the closest problem solved in the literature, which is the 3L-VRP-TW addressed in Krebs et al. (2023). For this purpose, we use the 585 instances from Krebs et al. (2021) for which a feasible loading pattern exists for each customer in an empty vehicle with the packing heuristic described in Krebs et al. (2023) and with the packing heuristic described in Section 4.2.

6.1.2. Instances for managerial decisions

To build the practical instances, we selected the 240 instances from Krebs et al. (2021) with 60 or 100 customers and a wide scheduling horizon (from 6AM to 10:40PM). We have 120 instances with 60 customers and 120 instances with 100 customers. In both categories, half of the instances have a total number of boxes of 200 and the other half of 400. Those 60 instances are split into three sets of instances of same size with box types equal to 3, 10, and 100 respectively. Out of these 20 instances, 10 have small (S) box sizes and 10 have large (L) box sizes.

We adapted those instances to match the characteristics of the problem we face by adding real features extracted from the survey.

First, we densified customer locations to account for time dependency and to preserve the possibility of visiting each customer. We assume that vehicles travel at a reduced speed during peak hours: 30km/h from 7:30AM to 9AM and from 4PM to 6PM, and 50km/h otherwise (as opposed to a constant speed of 60km/h in Krebs et al. (2021)). Time-dependent travel durations are then computed based on the algorithm proposed in Ichoua et al. (2003) and we consider the Euclidean distances.

Moreover, the customers are divided into companies with wide opening hours (from 7AM to 7PM) and individuals with narrow time windows (from 7AM to 10AM, from 10AM to 1PM, from 1PM to 4PM or from 4PM to 7PM). Each instance has 60% of companies and 40% of individuals evenly distributed across the 4 shifts.

Furthermore, since small-sized vehicles are used in an urban context, the dimensions of the loading space are rescaled and set as follows in all instances: the length is 420cm, the width is 210cm and the height is 230cm. The weight capacity of the vehicles is set to 1200kg, which corresponds to the payload of a vehicle with a volume between 20m^3 and 23m^3. The working duration is limited to 540 minutes per driver per day.

The box dimensions are scaled to fit into the loading space vehicles. Hence, each dimension is divided by the corresponding vehicle dimension in Krebs' data set and then multiplied by the dimension of our vehicle, as shown for the length in Equation (6). We applied the same transformation to the weight of the boxes.

$$\text{new box length} = \text{old box length} \times \frac{\text{length of our vehicle (=420)}}{\text{length of the vehicle in Krebs' data}} \tag{6}$$

Finally, the arm length is set to 50cm as in Ceschia et al. (2013) and the minimum contact area for one box to support another is set to 5cm × 5cm.

6.2. GVNS parameter tuning

The number of neighborhoods in both the shaking and the VND phases is fixed at 5.

Based on some preliminary tests, the maximum number of iterations without improvement was set to 1000, and the total number of iterations to 5000. The computation time limit is set to 3600 seconds as in Krebs et al. (2023).

6.3. Effectiveness of the three-phase heuristic

To assess the competitiveness of the solutions produced by the three-phase heuristic, we test the algorithm on the validation instances from Krebs et al. (2021) and compare the results with the ALNS developed in Krebs et al. (2023). To ensure a fair comparison, we modify our algorithm to fit their problem characteristics. Specifically, we remove the possibility of splitting customer requests, and remove the maximum working duration constraint as well as the time dependency by assuming a constant speed of 60km/h. Regarding the loading aspects, Krebs et al. (2023) define stability as a supporting area of at least 75% of the base area of an item, and consider LIFO instead of reachability. To be as close as possible to their problem definition, we select the best extreme point according to the Deepest-Bottom-Left-Fill (DBLF). Finally, we consider the parallel version of the savings algorithm as the constructive heuristic, since it outperforms the other constructive heuristics developed in this work, as will be statistically verified in Section 6.4.1. Moreover, Krebs et al. (2023) also started their heuristic with an initial solution obtained by the savings algorithm. Similarly to their article, we tested each instance 15 times.

After the first phase of our algorithm, none of the instances requires more vehicles than the fleet size. Thus, we never enter the route elimination procedure. Since the objective in Krebs et al. (2023) is to minimize the total travel distance without exceeding the fleet size (no rented vehicles allowed), and thanks to the previous remark, we can compare the solutions in terms of total travel distance.

Statistical tests comparing the averages over the 15 runs allow us to ascertain that, at a 5% significance level, there is no statistically significant difference between the ALNS of Krebs et al. (2023) and our approach for 136 out of 585 instances (23.25%). The three-phase heuristic significantly outperforms the ALNS for 25 instances (4.27%) and provides solutions with a travel distance within 7% on average compared to those obtained with the ALNS for the 424 remaining instances (72.48%). Table 1 in Section 3 of the Supplementary Material summarizes the average travel distance per instance over the 15 runs from the ALNS of Krebs et al. (2023) and our approach. These results show that our algorithm is capable of delivering competitive results, even though it was not designed to solve this specific problem.

The relative standard deviation between the final solution and the initial solution over the 15 runs[1] is about 4.74% at maximum and 0.68% on average, showing a significant robustness of the approach.

[1]Computed as the standard deviation of the relative improvement of the solution ((initial solution - final solution) / initial solution), starting from the same initial solution in the 15 runs.

We therefore decided to test each instance only once in the following experiments.

6.4. Computational results and analyses for the 3L-SPVRP-TW

In the following, we first determine the most efficient constructive heuristic in terms of cost and computation time. Then, we identify the best strategy for selecting the EP when packing the boxes. Next, we present the results, that is, the total travel distance and the time for each practical instance and we study the stopping criteria of the GVNS. Finally, we provide some managerial insights for LSPs.

6.4.1. Analyses of the first two phases of the algorithm

Statistical tests conclude that the parallel version of the savings algorithm significantly outperforms the others in terms of cost (details are provided in Appendix B). Moreover, all constructive heuristics take less than two seconds to solve each instance. Therefore, the parallel version of the savings algorithm is selected for generating the initial solution of the three-phase heuristic.

After the constructive phase with the parallel version of the savings algorithm, 16 instances out of 240 go through the route elimination procedure. The second step of this procedure, that is, the repetition of the process while allowing split, is never applied since the fleet size is no longer exceeded.

6.4.2. Criterion for the selection of the best EP

A paired two-sample t-test shows that considering a TDLF packing instead of a DBLF performs better (p-value=0.0034). This is probably due to the reachability constraint that is more likely to be satisfied when the boxes are stacked on top of each other. For this reason, we retain the TDLF packing in the following.

6.4.3. Total travel distance and computational time

Table 3 shows the total travel distance, which corresponds to the total cost since there is no rented vehicle as mentioned in Section 6.4.1, and the total time (in seconds) for the 240 practical instances.

Table 3: Total travel distance and computational time for the three-phase heuristic on the practical instances

# customers	Total # boxes	# box types	Box size	Instance	Total travel distance	Time (sec)	Instance	Total travel distance	Time (sec)
60	200	3	S	16	307.04	127.19	46	334.00	265.35
60	200	3	S	17	332.93	1209.46	47	342.96	628.86
60	200	3	S	18	313.83	273.68	48	323.91	119.34

# customers	Total # boxes	# box types	Box size	Instance	Total travel distance	Time (sec)	Instance	Total travel distance	Time (sec)
60	200	3	S	19	324.91	169.79	49	319.90	603.85
60	200	3	S	20	290.57	260.94	50	309.06	781.12
60	200	3	L	1	434.62	152.00	31	716.83	273.01
60	200	3	L	2	429.49	369.25	32	698.16	266.45
60	200	3	L	3	767.29	318.99	33	523.52	255.93
60	200	3	L	4	682.32	247.85	34	631.82	137.94
60	200	3	L	5	670.24	191.79	35	743.40	181.56
60	200	10	S	21	347.31	134.08	51	302.39	185.29
60	200	10	S	22	274.35	161.90	52	319.28	274.33
60	200	10	S	23	326.80	374.59	53	321.50	950.34
60	200	10	S	24	356.59	112.87	54	337.28	197.77
60	200	10	S	25	323.75	819.24	55	329.84	791.77
60	200	10	L	6	823.87	261.16	36	679.51	364.32
60	200	10	L	7	847.14	273.47	37	798.98	244.77
60	200	10	L	8	834.16	230.00	38	586.25	327.86
60	200	10	L	9	611.37	230.84	39	882.22	318.24
60	200	10	L	10	787.85	347.92	40	833.23	321.73
60	200	100	S	26	320.86	433.81	56	318.22	275.39
60	200	100	S	27	316.03	419.63	57	355.03	1282.13
60	200	100	S	28	318.97	301.96	58	335.17	288.35
60	200	100	S	29	335.08	345.32	59	318.34	1480.35
60	200	100	S	30	347.31	300.13	60	334.73	327.39
60	200	100	L	11	827.19	548.60	41	837.35	305.20
60	200	100	L	12	905.25	284.50	42	734.19	330.26
60	200	100	L	13	869.13	412.53	43	848.95	372.43
60	200	100	L	14	897.04	506.77	44	905.36	291.67
60	200	100	L	15	795.06	434.75	45	883.56	307.99
60	400	3	S	136	366.82	610.38	166	312.37	742.40
60	400	3	S	137	356.90	429.68	167	372.31	634.70
60	400	3	S	138	334.35	354.71	168	328.36	578.84
60	400	3	S	139	364.74	382.62	169	375.59	614.89
60	400	3	S	140	455.71	789.20	170	309.03	1167.74
60	400	3	L	121	1070.93	317.99	151	817.14	779.60
60	400	3	L	122	1057.64	709.57	152	740.14	209.94
60	400	3	L	123	844.41	991.47	153	737.50	552.80
60	400	3	L	124	1523.35	810.04	154	885.64	427.65
60	400	3	L	125	995.92	816.04	155	1109.31	495.05
60	400	10	S	141	336.44	1337.32	171	327.73	2715.86
60	400	10	S	142	378.58	1231.12	172	375.00	510.37
60	400	10	S	143	341.97	439.63	173	378.03	698.43
60	400	10	S	144	319.84	3079.91	174	383.22	724.10
60	400	10	S	145	358.55	1382.92	175	346.53	2104.79
60	400	10	L	126	1244.51	915.10	156	1283.96	980.92
60	400	10	L	127	1080.33	1172.58	157	1473.30	874.35
60	400	10	L	128	1108.14	779.44	158	1150.22	727.20
60	400	10	L	129	776.94	1542.24	159	1326.75	1069.37
60	400	10	L	130	1200.21	818.93	160	1599.90	788.92
60	400	100	S	146	412.97	1563.70	176	365.09	1210.66
60	400	100	S	147	390.43	1677.71	177	387.07	1516.08
60	400	100	S	148	410.41	1160.64	178	404.90	877.16
60	400	100	S	149	386.05	1669.06	179	389.63	1273.37
60	400	100	S	150	394.19	1650.28	180	390.17	1352.83
60	400	100	L	131	1607.45	1772.01	161	1270.64	1107.98
60	400	100	L	132	959.22	1035.32	162	1495.23	1269.36
60	400	100	L	133	1407.91	1260.44	163	1456.98	1023.76
60	400	100	L	134	1117.86	936.45	164	1308.92	1118.19

# customers	Total # boxes	# box types	Box size	Instance	Total travel distance	Time (sec)	Instance	Total travel distance	Time (sec)
60	400	100	L	135	1426.55	1202.06	165	1441.65	1121.94
100	200	3	S	76	446.66	247.22	106	412.93	220.02
100	200	3	S	77	433.18	1669.83	107	454.12	942.22
100	200	3	S	78	451.26	337.70	108	447.38	209.77
100	200	3	S	79	450.91	317.85	109	458.28	1138.87
100	200	3	S	80	430.51	322.41	110	423.48	288.21
100	200	3	L	61	519.70	301.97	91	603.25	389.28
100	200	3	L	62	754.62	475.69	92	605.26	368.14
100	200	3	L	63	700.68	376.51	93	486.78	547.91
100	200	3	L	64	761.62	350.62	94	568.74	291.58
100	200	3	L	65	591.18	295.12	95	871.27	794.71
100	200	10	S	81	459.53	1173.34	111	447.87	230.59
100	200	10	S	82	423.16	559.72	112	442.64	303.03
100	200	10	S	83	432.40	506.33	113	451.54	451.94
100	200	10	S	84	443.65	346.94	114	428.02	434.90
100	200	10	S	85	448.44	706.80	115	448.85	419.62
100	200	10	L	66	669.79	339.41	96	829.90	444.58
100	200	10	L	67	816.98	323.19	97	846.44	531.32
100	200	10	L	68	764.63	636.11	98	872.51	614.09
100	200	10	L	69	901.39	492.76	99	892.48	394.21
100	200	10	L	70	702.80	570.20	100	948.62	527.06
100	200	100	S	86	430.04	398.74	116	426.64	286.01
100	200	100	S	87	454.23	1046.23	117	425.16	286.98
100	200	100	S	88	438.73	614.18	118	454.37	1401.11
100	200	100	S	89	447.45	732.84	119	447.12	259.49
100	200	100	S	90	453.56	368.46	120	446.63	418.79
100	200	100	L	71	928.23	675.85	101	993.80	541.76
100	200	100	L	72	939.69	941.75	102	956.66	466.59
100	200	100	L	73	948.81	858.70	103	935.18	800.57
100	200	100	L	74	973.76	536.03	104	961.91	329.75
100	200	100	L	75	886.64	561.93	105	901.90	413.71
100	400	3	S	196	436.01	1078.43	226	442.43	556.52
100	400	3	S	197	446.28	799.97	227	497.88	500.54
100	400	3	S	198	427.38	669.40	228	477.72	336.49
100	400	3	S	199	464.40	399.02	229	481.18	354.06
100	400	3	S	200	453.15	676.84	230	440.61	375.48
100	400	3	L	181	1008.16	794.92	211	1114.46	872.89
100	400	3	L	182	931.52	2031.64	212	1302.26	911.73
100	400	3	L	183	1702.51	1318.24	213	1126.46	1278.12
100	400	3	L	184	1249.40	748.75	214	987.18	1574.39
100	400	3	L	185	1110.67	732.68	215	907.44	1062.10
100	400	10	S	201	453.06	1067.19	231	445.83	574.12
100	400	10	S	202	457.69	533.73	232	461.68	1266.00
100	400	10	S	203	458.42	596.86	233	490.02	357.55
100	400	10	S	204	490.08	1105.93	234	453.32	473.87
100	400	10	S	205	480.37	381.27	235	478.64	631.35
100	400	10	L	186	1667.79	2365.52	216	1219.96	1106.87
100	400	10	L	187	1381.24	1382.49	217	1252.14	935.36
100	400	10	L	188	1126.33	1618.94	218	1028.66	1309.73
100	400	10	L	189	1325.20	1527.88	219	1259.69	1022.28
100	400	10	L	190	1328.89	1607.57	220	1650.53	1975.44
100	400	100	S	206	437.78	814.23	236	485.20	1659.65
100	400	100	S	207	483.48	1085.74	237	491.72	2080.09
100	400	100	S	208	463.89	1695.40	238	496.30	773.27
100	400	100	S	209	477.34	1274.06	239	464.78	2047.71
100	400	100	S	210	494.19	843.21	240	476.97	1324.31
100	400	100	L	191	1357.04	1175.59	221	1578.07	1367.33

# customers	Total # boxes	# box types	Box size	Instance	Total travel distance	Time (sec)	Instance	Total travel distance	Time (sec)
100	400	100	L	192	1636.51	2856.15	222	1591.38	1315.24
100	400	100	L	193	1556.99	2219.38	223	1492.09	2196.22
100	400	100	L	194	1491.34	2449.46	224	1582.94	1945.38
100	400	100	L	195	1456.36	1276.27	225	1554.00	2030.42

As expected, instances with large boxes have a higher total travel distance (average=1017.64) than those with small boxes (average=400.48).

Taking a closer look at the added value of the GVNS, we observe that 175 out of 240 instances use split after the first two phases. This decreases to 109 instances after the GVNS, indicating that the GVNS tends to gather split requests to reduce the travel distance.

The three-phase heuristic mainly spends computation time within the GVNS (less than one second for the first two phases). However, for 75% of the instances, the GVNS stops in less than 19 minutes. Moreover, a 66% increase in the number of customers results in a 22% increase in computation time on average.

6.4.4. Stopping criteria

Out of the 240 instances, we observed that 229 (95.42%) runs ended because no improvement was observed over 1000 iterations, 11 (4.58%) instances due to the total maximum number of iterations of 5000 iterations, and finally none instance due to the time limit of 3600 seconds. Thus, the computation time is therefore not restrictive and the heuristic has a sufficient number of iterations to explore the neighborhoods.

6.4.5. Managerial insights

In this section, we analyze the potential benefits to LSPs of allowing split pickups and of extending the range of the time windows for individual customers. We also assess the impact of considering reachability constraint instead of sequential loading constraint and of accounting for time-dependent travel durations.

Usefulness of split. Its benefit is first determined by comparing the cost of the solution when split is allowed or not. Then, the impact of the number of customers and the impact of the heterogeneity of the boxes on the split are analyzed.

Without split, 26 out of 240 instances have no initial solution using our packing heuristic since it does not find a loading pattern for at least one customer in a single vehicle in these instances.

For 7 instances, the algorithm ends with a solution at the same cost. A paired two-sample t-test proves that, on average, allowing split is cheaper than considering only single visit to the customers (p-value=2.42×10^{-4}).

Figure 7 shows that the LSP visits at least one customer twice in more than 40% of the cases. Figure 8 shows that the greater the number of box types (i.e. the heterogeneity of the boxes), the larger the number of split requests. We can also observe that when the average number of boxes per customer increases (from the left to the right), the number of split customers increases as well. Moreover, split only occurs for instances with large box sizes.

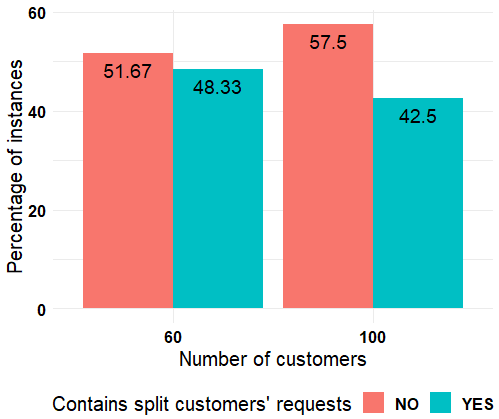

Figure 7: Distribution of the percentage of instances with at least one customer visited twice based on the number of customers

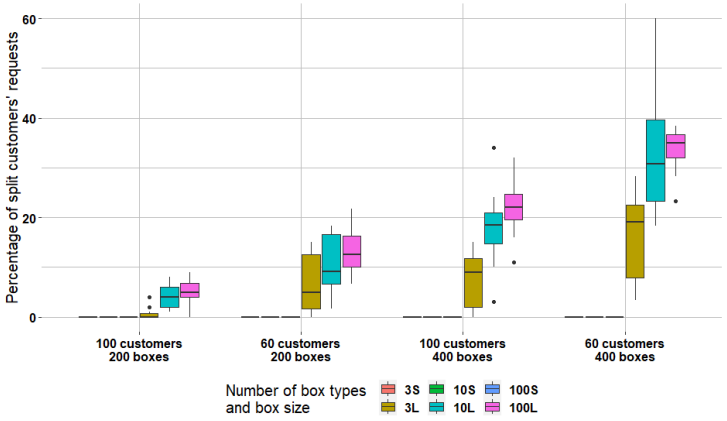

Figure 8: Percentage of customers visited twice based on the number of customers, the total number of boxes, the number of box types, and the box size

Impact of the range of the time windows for individual customers. Individual customers are distributed over 4 shifts: two in the morning, and two in the afternoon. To determine the impact of the range of the time windows, the morning (resp. afternoon) shifts are extended from 7AM to 1PM (resp. from 1PM to 7PM).

First, we compare the **cost** for the LSP as well as the **number of required vehicles**. The results show that there are benefits for the LSP in allowing only two extended shifts. Indeed, paired two-sample t-tests conclude that extending the time windows for individual customers is significantly cheaper (p-value = 3.26×10^{-31}) with a reduction in the travel distance of up to 85km and requires significantly fewer vehicles (p-value = 8.93×10^{-13}) with a reduction of up to 4 vehicles.

Figure 9 shows that extending shifts allows the vehicle's **loading rate** to increase, as the histogram related to extended shifts is more left-skewed than with the 4 initial shifts. However, some vehicles

have a high loading rate while others do not. We can observe in Figure 10 that when the number of box types increases, the loading rate decreases. Thus, the heterogeneity of the boxes makes it more difficult to satisfy the loading constraints and reduces the loading rate. Moreover, we observe smaller loading rate when the box size is small. Nevertheless, some vehicles from instances with a small number of box types and large box size show a low loading rate. By taking a look at the loading patterns (Figure C.14), even though the loading rate of some vehicles is low, some boxes are close to the vehicle's door, preventing more boxes from being added due to the reachability constraint. This implies that a new vehicle must be used to visit the remaining customers. However, some solutions have vehicles with a high loading rate, which shows that our packing heuristic is able to provide compact loading patterns.

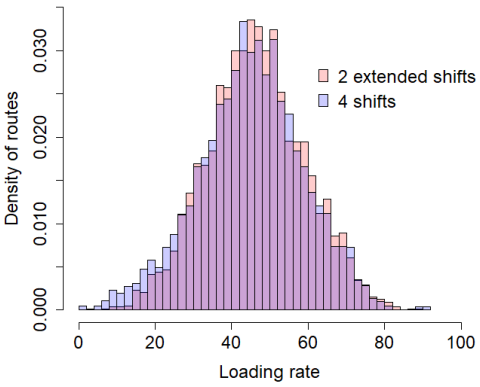

Figure 9: Histogram of the loading rate depending on the range of the time windows

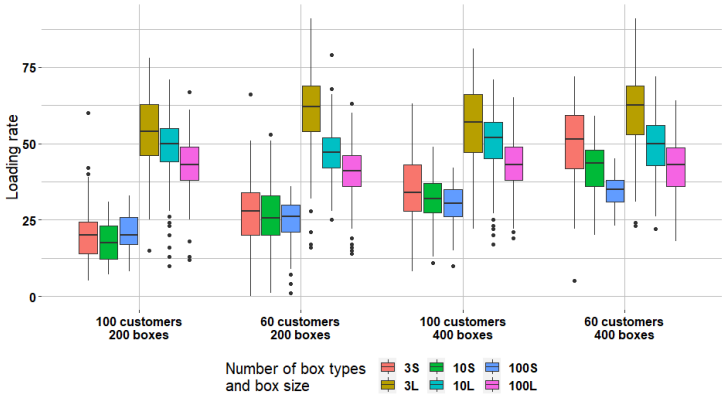

Figure 10: Loading rate based on the number of customers, the total number of boxes, the number of box types, and the box size

Finally, we observe the impact of extending the time windows on the drivers' **working durations**. Figure 11 shows that the drivers spend less time outside the depot when the shifts are extended. If we observe the distribution of the working time among the different tasks that a driver performs on Figure 12, we identify that, thanks to the extended time windows, the drivers can directly serve the customers, thus reducing the waiting time.

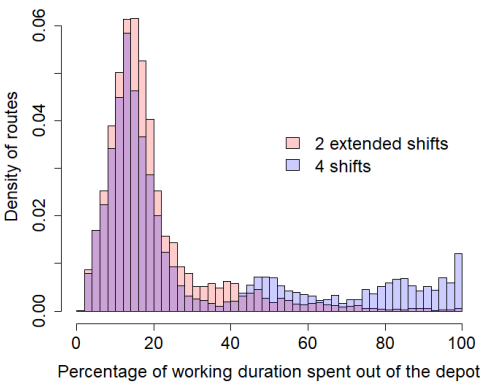

Figure 11: Distribution of the working duration spent outside the depot depending on the range of the time windows for individual customers

Figure 12: Distribution of the working duration among the driver's tasks depending on the range of the time windows for individual customers

Impact of the reachability constraint versus the sequential loading constraint. Statistical tests show that considering the reachability constraint significantly increases the cost (p-value$=9.93 \times 10^{-19}$) as well as the number of vehicles (p-value$=5.7 \times 10^{-17}$). Consequently, the filling rate of the vehicles is reduced, as illustrated in Figure C.15. Despite this significant increase in cost and number of vehicles, it is crucial to consider the reachability constraint to identify feasible solutions in practice. Moreover, there is no significant difference in computational time when considering the reachability constraint instead of the sequential loading constraint (p-value$=0.5672$).

Impact of the time-dependent travel durations. To assess its impact, we compare three off-peak/peak speed ratios: 5/3 (basic scenario described in Section 6.1.2), 5/4 and 5/5 (travel speed constant over the day). Although there is no significant difference in cost when the ratio goes from 5/4 to 5/5, there is a significant decrease when the ratio goes from 5/3 to 5/4 or to 5/5 (p-value< 0.0005). Thus, it appears that a large reduction in speed during peak hours compared to off-peak hours could result in higher costs for the LSP. Moreover, statistical tests show that the average time a driver spends out of the depot tends to increase as the off-peak/peak speed ratio increases. More specifically, the average waiting duration and the average service duration seem to remain the same (p-value$= 0.7274$ and p-value$= 0.09462$), while the average travel duration increases (p-values$< 1.34 \times 10^{-21}$). Thus, it is important to consider time-dependent travel durations in order to provide a feasible schedule associated with the route and to arrive at the customer's location on time. Even if the speed during peak hours cannot be controlled by the drivers, it would be crucial for the LSP to have a precise estimation of this speed to ensure the feasibility of the schedules.

7. Conclusions

In this paper, based on a survey of practitioners in parcel collection, we address the integration of vehicle routing and container loading problems including on the one hand for routing: capacity, customer time windows, driver working duration, split pickups and time-dependent travel durations constraints and on the other hand, for packing: horizontal and vertical stability, and parcel reachability constraints. This reachability notion is more accurate and realistic than the classical LIFO constraint. The literature review highlights the contribution of dealing with this enriched model among the current literature which usually focuses on parcel delivery for the classic 3L-CVRP.

In order to solve efficiently this problem, a three-phase heuristic has been developed. It includes an initial solution based on a constructive procedure based on the savings algorithm after comparing several other options. If needed, the second phase tends to reduce the number of vehicles used in the initial solution to the carrier fleet size to avoid extra costs for renting vehicles. The third phase using a GVNS procedure including five neighborhoods improves the solution. Tests performed on instances from a benchmark set proved the efficiency in performance (i.e., computation time and solution value) of this algorithm. New instances including all the features of our practical context are proposed and solved for various numbers of customers, number of boxes and box heterogeneity. As the computational performances are preserved, this allowed us to test various assumptions about our problem constraints and global parameters. For instance, managerial insights show that allowing split pickups and extending customer time windows is economical. Furthermore, the number of drivers to hire is reduced when the range of time windows is extended. Moreover, we notice that the LSP should investigate the multi-trip option as the packing constraints lead to filling the vehicle in a much shorter time than the drivers' working duration. We show the impacts of considering the reachability constraint on cost and the variation of speed during peak hours on schedule feasibility.

As a perspective for future work, to get one step closer to reality, some additional elements might be taken into account. One important fact resides in the assumption that the full information taken into account when planning is known and certain. For example, most transportation service providers do not know the exact dimensions of the boxes in advance, and so the adaptation of the packing might be left to the driver. This may lead to reloading efforts or sending a new vehicle to meet the request, or even not collecting the boxes. Some new customers may pop up on-line or current customers may cancel their planned requests or add unforeseen extra boxes. This would lead to an on-line adaptation of the current off-line decision process.

Acknowledgments

The authors would like to thank Manuel López-Ibáñez for his help in using the `irace` package.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used Deepl Write in order to check the spelling of the document. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Appendix A. Benchmark instances of the three-dimensional loading VRP

Table A.4: Relevant benchmark instances of the three-dimensional loading VRP

Authors	# instances	existence of TW	# customers	# boxes	# box types
Gendreau et al. (2006)	27 generated 5 real		15-100	1-3 per customer	26-199
Moura & Oliveira (2009)	46 generated	✓	25	30-80 per customer 50-100 per customer	1-5
Bortfeldt & Homberger (2013)	120 generated	✓	100,200,400,600,800,1000	5000-50000	5,10
Ceschia et al. (2013)	13 real		11-129	254-8060	9-97
Zhang et al. (2017)	27 generated	✓	15-100	1-3 per customer	26-199
Bortfeldt & Yi (2020)	20 generated		50-199	3377-14230	3, 20
Krebs et al. (2021)	600 generated	✓	20, 60, 100	200, 400	3, 10, 100

Appendix B. Computational results for the 3L-SPVRP-TW: constructive heuristics

Figure B.13 shows the distribution of the total transportation costs (distance and rented vehicles) after the first phase depending on the constructive heuristics. The lines with stars show the tests (a Friedman test followed by a Wilcoxon test, adjusted via Bonferroni) with a p-value close to 0, where the alternative hypothesis is that the heuristic on the left provides a solution with a strictly smaller cost than the one on the right.

We can see that the parallel version of the savings algorithm significantly outperforms all others in terms of cost.

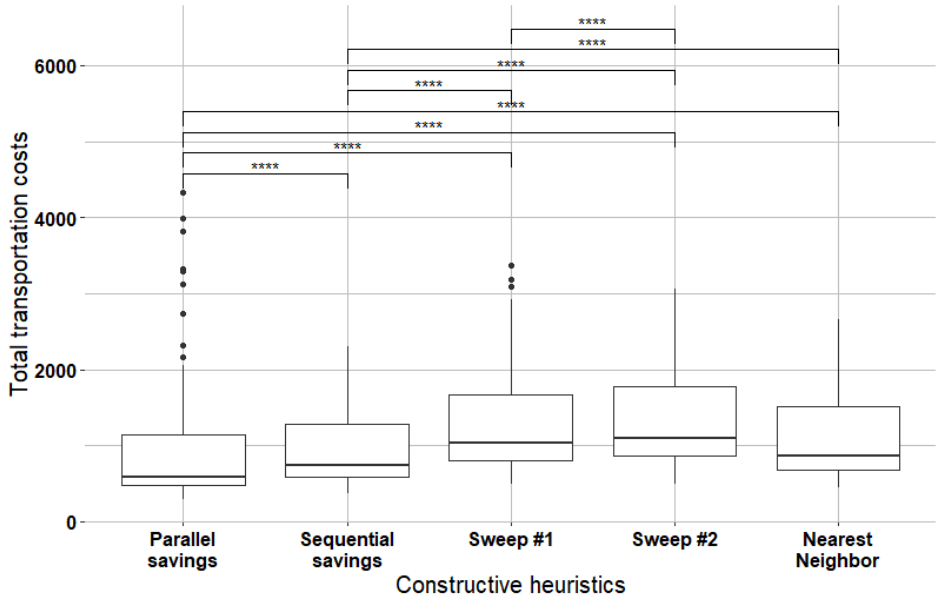

Figure B.13: Comparison of constructive heuristics in terms of cost

Appendix C. Computational results for the 3L-SPVRP-TW: managerial insights

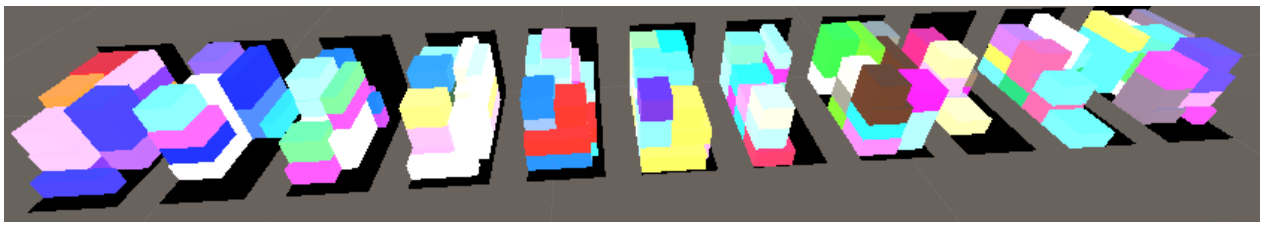

Figure C.14: Visualization of vehicle loading patterns from one instance with a small number of box types and large boxes (boxes of the same color belong to one customer)

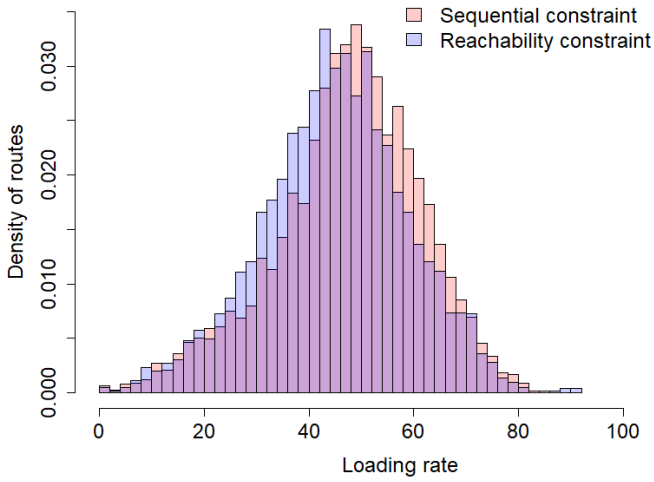

Figure C.15: Histogram of the loading rate depending on the sequential loading constraint

References

Adamo, T., Gendreau, M., Ghiani, G., & Guerriero, E. (2024). A review of recent advances in time-dependent vehicle routing. *European Journal of Operational Research*.

Ambilkar, P., Dohale, V., Gunasekaran, A., & Bilolikar, V. (2022). Product returns management: a comprehensive review and future research agenda. *International Journal of Production Research*, 60(12), 3920–3944.

Bortfeldt, A. (2012). A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. *Computers & Operations Research*, 39(9), 2248–2257.

Bortfeldt, A. & Homberger, J. (2013). Packing first, routing second—a heuristic for the vehicle routing and loading problem. *Computers & Operations Research*, 40(3), 873–885.

Bortfeldt, A. & Wäscher, G. (2013). Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, 229(1), 1–20.

Bortfeldt, A. & Yi, J. (2020). The split delivery vehicle routing problem with three-dimensional loading constraints. *European Journal of Operational Research*, 282(2), 545–558.

Bräysy, O. (2003). A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4), 347–368.

Ceschia, S., Schaerf, A., & Stützle, T. (2013). Local search techniques for a routing-packing problem. *Computers & Industrial Engineering*, 66(4), 1138–1149.

Chen, Z., Yang, M., Guo, Y., Liang, Y., Ding, Y., & Wang, L. (2020). The split delivery vehicle routing problem with three-dimensional loading and time windows constraints. *Sustainability*, 12(17), 6987.

Chi, J. & He, S. (2023). Pickup capacitated vehicle routing problem with three-dimensional loading constraints: Model and algorithms. *Transportation Research Part E: Logistics and Transportation Review*, 176, 103208.

Clarke, G. & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4), 568–581.

Côté, J.-F., Guastaroba, G., & Speranza, M. G. (2017). The value of integrating loading and routing. *European Journal of Operational Research*, 257(1), 89–105.

Crainic, T. G., Perboli, G., & Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing*, 20(3), 368–384.

Dondo, R. & Cerdá, J. (2013). A sweep-heuristic based formulation for the vehicle routing problem with cross-docking. *Computers & Chemical Engineering*, 48, 293–311.

Eglese, R., Maden, W., & Slater, A. (2006). A Road Timetable to aid vehicle routing and scheduling. *Computers & Operations Research*, 33(12), 3508–3519. `https://doi.org/10.1016/j.cor.2005.03.029`

Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2010). Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research*, 201(3), 751–759.

Gendreau, M., Iori, M., Laporte, G., & Martello, S. (2006). A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3), 342–350.

Gillett, B. E. & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2), 340–349.

Giménez-Palacios, I., Parreño, F., Álvarez-Valdés, R., Paquay, C., Oliveira, B. B., Carravilla, M. A., & Oliveira, J. F. (2022). First-mile logistics parcel pickup: Vehicle routing with packing constraints under disruption. *Transportation Research Part E: Logistics and Transportation Review*, 164, 102812.

Gzara, F., Elhedhli, S., & Yildiz, B. C. (2020). The pallet loading problem: Three-dimensional bin packing with practical constraints. *European Journal of Operational Research*, 287(3), 1062–1074.

Hansen, P., Mladenović, N., Brimberg, J., & Pérez, J. A. M. (2019). *Variable neighborhood search*. Springer.

Ichoua, S., Gendreau, M., & Potvin, J. Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2), 379–396. https://doi.org/10.1016/S0377-2217(02)00147-9

Iori, M. & Martello, S. (2010). Routing problems with loading constraints. *Top*, 18(1), 4–27.

Junqueira, L., Morabito, R., & Sato Yamashita, D. (2012). MIP-based approaches for the container loading problem with multi-drop constraints. *Annals of Operations Research*, 199, 51–75.

Junqueira, L., Oliveira, J. F., Carravilla, M. A., & Morabito, R. (2013). An optimization model for the vehicle routing problem with practical three-dimensional loading constraints. *International Transactions in Operational Research*, 20(5), 645–666.

Karabulut, K. & İnceoğlu, M. M. (2005). A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. *Advances in Information Systems: Third International Conference, ADVIS 2004, Izmir, Turkey, October 20-22, 2004. Proceedings 3*, 441–450.

Koch, H., Bortfeldt, A., & Wäscher, G. (2018). A hybrid algorithm for the vehicle routing problem with backhauls, time windows and three-dimensional loading constraints. *OR Spectrum*, 40, 1029–1075.

Kok, A. L., Hans, E. W., & Schutten, J. M. (2012). Vehicle routing under time-dependent travel times: The impact of congestion avoidance. *Computers & Operations Research*, 39(5), 910–918. https://doi.org/10.1016/j.cor.2011.05.027

Krebs, C., Ehmke, J. F., & Koch, H. (2021). Advanced loading constraints for 3D vehicle routing problems. *OR Spectrum*, 43(4), 835–875.

Krebs, C., Ehmke, J. F., & Koch, H. (2023). Effective loading in combined vehicle routing and container loading problems. *Computers & Operations Research*, 149, 105988.

Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & Operations Research*, 34(9), 2743–2757.

Laporte, G., Gendreau, M., Potvin, J.-Y., & Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7(4-5), 285–300.

Leloup, E., Paquay, C., Pironet, T., & Oliveira, J. F. (2024). *Three-dimensional loading vehicle routing problem with split pickups and time windows (3L-SPVRP-TW): instances and results.* Mendeley Data. https://doi.org/10.17632/y9rf4nj7rd.1

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3, 43–58.

Meliani, Y., Hani, Y., Elhaq, S. L., & El Mhamedi, A. (2022). A tabu search based approach for the heterogeneous fleet vehicle routing problem with three-dimensional loading constraints. *Applied Soft Computing*, 126, 109239.

Mitrović-Minić, S. & Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7), 635–655.

Mladenović, N. & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11), 1097–1100.

Moura, A. (2019). A model-based heuristic to the vehicle routing and loading problem. *International Transactions in Operational Research*, 26(3), 888–907.

Moura, A. & Oliveira, J. F. (2009). An integrated approach to the vehicle routing and container loading problems. *OR Spectrum*, 31, 775–800.

Paquay, C., Limbourg, S., & Schyns, M. (2018). A tailored two-phase constructive heuristic for the three-dimensional multiple bin size bin packing problem with transportation constraints. *European Journal of Operational Research*, 267(1), 52–64.

Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., & Limbourg, S. (2015). Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, 37(2), 297–330.

Ramanathan, R. (2011). An empirical analysis on the influence of risk on relationships between handling of product returns and customer loyalty in E-commerce. *International Journal of Production Economics*, 130(2), 255–261.

Rand, G. K. (2009). The life and times of the savings method for vehicle routing problems. *ORiON*, 25(2), 125–145.

Ronconi, D. P. & Manguino, J. L. (2022). Grasp and vns approaches for a vehicle routing problem with step cost functions. *Annals of Operations Research*, 1–26.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing*, 4(2), 146–154.

Schilde, M., Doerner, K. F., & Hartl, R. F. (2014). Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *European Journal of Operational Research*, 238(1), 18–30.

Statista (2023). https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/. Retrieved on October 09, 2024.

Statista (2024). https://www.statista.com/statistics/534123/e-commerce-share-of-retail-sales-worldwide/. Retrieved on October 09, 2024.

Thammano, A. & Rungwachira, P. (2021). Hybrid modified ant system with sweep algorithm and path relinking for the capacitated vehicle routing problem. *Heliyon*, 7(9), e08029.

Toth, P. & Vigo, D. (2002). *The vehicle routing problem*. SIAM.

Toth, P. & Vigo, D. (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Vangipurapu, B. R. & Govada, R. (2021). A construction heuristic for finding an initial solution to a very large-scale capacitated vehicle routing problem. *RAIRO-Operations Research*, 55(4), 2265–2283.

Vega-Mejía, C. A., Montoya-Torres, J. R., & Islam, S. (2019). Consideration of triple bottom line objectives for sustainability in the optimization of vehicle routing and loading operations: a systematic literature review. *Annals of Operations Research*, 273(1), 311–375.

Zhang, D., Cai, S., Ye, F., Si, Y.-W., & Nguyen, T. T. (2017). A hybrid algorithm for a vehicle routing problem with realistic constraints. *Information Sciences*, 394, 167–182.

Zhang, M., Pratap, S., Zhao, Z., Prajapati, D., & Huang, G. Q. (2021). Forward and reverse logistics vehicle routing problems with time horizons in B2C e-commerce logistics. *International Journal of Production Research*, 59(20), 6291–6310.

Zhang, Z., Wei, L., & Lim, A. (2015). An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints. *Transportation Research Part B: Methodological*, 82, 20–35.