Hybrid iterated local search algorithm for the vehicle routing problem with lockers

Bruno Oliveira[1*], Artur Pessoa[2†] and Marcos Roboredo[2†]

[1*]Programa de Pós Graduação em Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Pátria, Niterói, 24210240, Rio de Janeiro, Brazil.
[2]Departmento de Engenharia de Produção, Universidade Federal Fluminense, Rua Passo da Pátria, Niterói, 24210240, Rio de Janeiro, Brazil.

*Corresponding author(s). E-mail(s): bruno_mattos@id.uff.br;
Contributing authors: arturpessoa@id.uff.br; mcroboredo@id.uff.br;
†These authors contributed equally to this work.

Abstract

In the Vehicle Routing Problem (VRP) with Lockers, the vertices in a graph are divided into two key subsets: a customer set and a locker set, with lockers serving as alternative delivery points for the customer's parcel. This paper presents a generic VRP with lockers, where a customer's parcel can be delivered either to its home within a time windows or to a nearby locker, where they can pick it up at any time. The objective is to design a set of routes for a fleet of homogeneous vehicles in order to minimize the total travel, vehicle and assignment costs in a such manner that each locker can be visited at most once, the capacities of the vehicles and lockers are not exceeded and the time windows constraints are ensured. To solve this problem. we propose a hybrid metaheuristic that combines Iterated Local Search with a Set Partitioning-like model. Our algorithm uses several classical and specific neighborhood operators. We demonstrate the effectiveness of the proposed methodology by evaluating it on benchmark literature instances from three problems that we show that are encompassed by the generic problem defined in this paper. The results show that we found or improved 431 out of 490 best known solutions.

Keywords: Iterated local search, Vehicle routing problem, Hybrid algorithm, Lockers, Set partitioning

1 Introduction

The use of lockers in last-mile delivery has emerged as an innovative and efficient solution to modern logistical challenges. In an era where the demand for fast and convenient deliveries is growing exponentially, lockers provide a practical and sustainable alternative. Strategically located in easily accessible places such as supermarkets, shopping centers, and public transportation stations, these collection points allow consumers to pick up their parcels at times that best fit their schedules, eliminating the need to be home at the time of delivery. In addition to increasing customer convenience, lockers optimize the vehicle routing process, reducing the number of failed delivery attempts and consequently lowering operational costs and environmental impact. This solution represents a significant evolution in last-mile delivery management, combining technology and logistics to meet the demands of e-commerce and enhance the consumer experience.

In this paper, we present a generic Vehicle Routing Problem (VRP) with lockers. This problem aims to design a set of routes for a fleet of homogeneous vehicles to minimize three types of costs: travel, vehicle, and assignment ones. The routes are designed in a way that each locker is visited at most once, and vehicle and locker capacities constraints, as well time windows ones must be satisfied.

To solve the problem defined in this paper, we propose a hybrid algorithm that combines Iterated Local Search (ILS), Randomized Variable Neighborhood Descent (RVND) and a Set Partitioning-like (SP) model. Our algorithm uses several classical and specific neighborhood operators.

In order to test our approach, we present several computational experiments over benchmark instances from three problems that are encompassed by the proposed problem definition: the Vehicle Routing Problem with Lockers and Time Windows (VRPLTW) (Buzzega and Novellani, 2023); the Vehicle Routing Problem with Private and Shared Delivery Location (VRPPSDL) (Mancini and Gansterer, 2021); the Vehicle Routing Problem with Demand Allocation (VRPDA) (Ghoniem et al., 2013).

The main contributions of this work are

- A generic problem definition that encompasses the VRPLTW, VRPPSDL and VRPDA;
- A hybrid algorithm for the defined generic problem;
- The first heuristic approach for VRPLTW and its variants;
- A comparison between the proposed approach and the current state-of-art metaheuristic approaches for VRPPSDL and VRPDA, proposed by Grabenschweiger et al. (2021) and Reihaneh and Ghoniem (2018);
- The achievement or improvement of 431 out of 490 best known solutions considering the three problems addressed in this paper.

This work is organized as follows. Section 2 presents a general description of a generic VRPs with locker definition and three problems that are encompassed by it. Section 3 presents a brief literature review about the problems addressed in this paper and other VRPs with locker variants. Section 4 presents the proposed hybrid ILS metaheuristic. Section 5 presents the computational experiments and the results of the proposed approach. Finally, 6 summarizes our conclusions.

2 Problem Description

Let $G = (V, E)$ a complete undirected graph with $V = \{0\} \cup V_C \cup V_L$, where the node 0 represents the depot vertex, the set V_C represents the customer vertices, and the set V_L represents the locker vertices. Each edge $e \in E$ is associated with a positive cost c_e and with a positive travel time t_e. Each vertex $v \in V$ is associated with a time windows $[a_v, b_v]$ ($a_0 = 0$). Each customer $i \in V_C$ is also associated with a positive parcel demand q_i to be fulfilled, a service time s_i, and a set $\phi(i) \subset \{i\} \cup V_L$ where $\phi(i)$ represents the set of all possible parcel delivery locations for that customer. For example, if $\phi(i) = \{i, l_1, l_2\}$, then the customer's parcel can be delivered at his home (node i) or at the lockers l_1 or l_2. If the parcel of a customer $i \in V_C$ is delivered at a locker $l \in \phi(i) \cap V_L$, then it generates a assignment cost f_{il}. Each locker $l \in V_L$ is also associated with a capacity Q_l, and a service time s_l. In order to serve the customers demands, an unlimited fleet of homogeneous vehicles with capacity Q is available. The objective is to design a set of routes and customer-locker assignments for the vehicles such that

- Each route starts and ends at the depot node.
- Each node is visited within its time windows (including the depot node).
- Each locker can be visited at most once considering all designed routes.
- The parcel demand q_i of each customer $i \in V_C$ is fully delivered at exactly a single node of $\phi(i)$.
- The sum of parcels delivered by each vehicle does not exceed Q.
- The sum of parcels delivered at a given locker $l \in V_L$ does not exceed Q_l.

The routes are designed in order to minimize the sum of three objective functions: the total traveled cost, the total assignment cost, and the number of used vehicles. These functions are weighted by respectively α_1, α_2 and α_3.

This paper focus on three literature problems that are encompassed by the previous definition: the VRPLTW, the VRPPSDL, and the VRPDA. These problems have the following particular characteristics:

- VRPLTW: The assignment cost depends on the distance between the customer and the locker. Lockers and vehicles are both capacitated. The objective function considers only the routing and assignments costs;
- VRPPSDL: The assignment costs are identical. Lockers are capacitated, but the vehicles are not. The objective function considers the assignments, routing and vehicle usage costs;
- VRPDA: The assignment cost depends on the distance between the customer and the locker. There is no time windows constraints and the customers' demand can not be delivered at its home. Vehicles are capacitated, but the lockers are not. The objective function considers assignments and routing costs.

Besides the three previous problems, this paper also deals with three VRPLTW variants: without time windows (VRPL), with a single route (TSPLTW), and with a single route and without time windows (TSPL).

Figure 1 illustrates two distinct solutions for the generic problem in an instance with 4 customers and two lockers. The dashed red lines indicate that a customer is

assigned to a locker while the routes are represented through the directed black arcs. In this figure, the node 0 marked in yellow represents the depot, the circle nodes 1,2,3, and 4 marked in blue represent the customers, and the triangle nodes 5 and 6 represent the lockers. On left, we present a solution where each customer is assigned to one of the two visited lockers while on right, we present a solution where the locker 6 is not visited and the customers 3 and 4 are directly visited at home. According to the VRPDA definition, only the solution on left is feasible

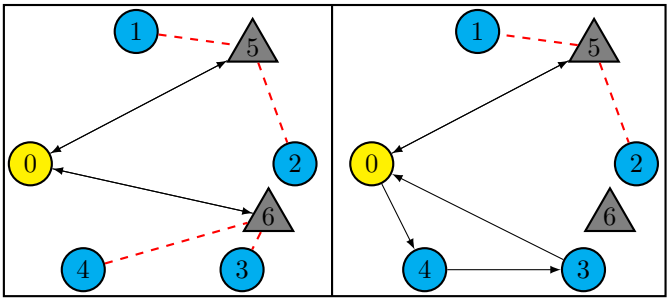

Fig. 1: Example of two solutions of the generic problem.

3 Literature Review

Now, we present a brief literature review about each problem addressed in this paper and about vehicle routing problems with lockers in general. This section is organized as follows. Subsection 3.1 provides a comprehensive summary of the literature on VRPLTW and VRPPSDL. Similarly, subection 3.2 offers a detailed review of the literature on VRPDA. Lastly, Subsection 3.3 presents a thorough literature review on other VRPs with lockers.

3.1 Literature review about VRPLTW and VRPPSDL

Mancini and Gansterer (2021) and Buzzega and Novellani (2023) introduced respectively the VRPPSDL and VRPLTW, that are two very similar problems encompassed by the generic problem proposed in this paper. In these problems, minimal cost routes are designed in a way that each customer's parcel can either to be delivered at a nearby locker or at it's home within a time windows.

The VRPLTW and VRPPSDL share many similarities, such as locker capacities, time window constraints, and the requirement that each locker is visited at most once. Regarding the main differences, it is notable that vehicle capacities are not considered in VRPPSDL, and a fixed cost for the use of vehicles, as well as service times are not considered in VRPLTW.

Mancini and Gansterer (2021) proposed a Mixed Integer Programming (MIP) formulation and two matheuristics for the VRPPSDL. The metaheuristics are based on ILS and Adaptive Large Neighborhood Search (ALNS) algorithms. To solve the VRPLTW, Buzzega and Novellani (2023) proposed three different MIP formulations.

The authors also presented some valid inequalities, leading their approach to a branch-and-cut (B&C), and considered the three VRPLTW variants: VRPL, TSPLTW, and TSPL. In addition to presenting several computational experiments on VRPLTW, VRPL, TSPLTW, and TSPL instances, the authors adapted their B&C algorithm to solve VRPPSDL instances and conducted a comparison between their formulation and that proposed by Mancini and Gansterer (2021). The comparative results indicated that the B&C algorithm exhibited superior computational performance.

Grabenschweiger et al. (2021) introduced a problem related to the VRPLTW, named the Vehicle Routing Problem with Heterogeneous Locker Boxes (VRPHLB). In this scenario, customers' parcels vary in size, necessitating careful consideration of how parcels are to be packed at locker box stations, in addition to route and customer assignment decisions. To address this challenge, the authors developed both a mathematical formulation and a metaheuristic solution. As the VRPHLB is a generalization of the VRPPSDL, Grabenschweiger et al. (2021) also showed that their metaheuristic approach was able to improve or achieve the same quality than all VRPPSDL solutions presented in Mancini and Gansterer (2021).

3.2 Literature review about VRPDA

Ghoniem et al. (2013) proposed the Vehicle Routing Problem with Demand Allocation (VRPDA), which requires optimizing the selection of lockers from candidate locations, assigning customers to these delivery sites, and routing a fleet of vehicles from a central depot to deliver parcels. They proposed a relax-and-fix (Pochet and Wolsey (2006)) heuristic, Column Generation (CG) approach, where the pricing subproblem is solved via MIP formulation, and a MIP formulation.

Solak et al. (2014) proposed two Benders decomposition algorithms for VRPDA. The authors presented computational experiments that show that their algorithms perform better than MIP formulation proposed by Ghoniem et al. (2013).

Reihaneh and Ghoniem (2018) introduced for the VRPDA a multi-start heuristic that iteratively solve a Generalized Assignment Problem (GAP) and a Capacitated Vehicle Routing Problem (CVRP) with perturbations. That algorithm improve several best known solutions up to that point.

Reihaneh and Ghoniem (2019) developed a branch-and-price algorithm for VRPDA, where the pricing subproblems can be seen as a Resource Constrained Shortest Path Problem (RCSP) and are solved via tailored labeling algorithm (Feillet et al. (2004)). Their method found optimal solutions for several instances with up to 25 lockers and 50 customers within one hour of running time.

3.3 Literature review about other VRPs with Lockers

Zhou et al. (2018) proposed the Multi-Depot Two-Echelon Vehicle Routing Problem with Delivery Options. In this scenario, the first echelon routes parcels from the main depot to secondary depots, with parcel delivery taking place in the second echelon. Customers express preferences for receiving parcels at home or at lockers. Lockers

are uncapacitated and can be visited by multiple vehicles. The objective is to minimize total distribution costs. To tackle this problem, the authors developed a genetic algorithm.

Tilk et al. (2021) introduced the Vehicle Routing Problem with Delivery Options (VRPDO), where customers select a priority level for various locations where they can retrieve their deliveries. A higher priority indicates a lower preference for delivery at that location. The total priority level across a set of routes must not exceed a specified threshold. To address this challenge, they proposed a branch-cut-and-price (BCP) algorithm featuring non-robust and robust cuts.

Vincent et al. (2022) introduced the Vehicle Routing Problem with Parcel Lockers (VRPPL), where customers have the flexibility to choose their preferred delivery location: at home, at a locker, or at any designated location. Unlike the problems addressed in this paper, in the VRPPL, a locker can be visited multiple times, and the objective is to minimize the total cost of routes only. In order to solve this problem, the authors developed both a MIP formulation and a Simulated Annealing (SA) algorithm.

4 Iterated Local Search algorithm

In this section, we describe the proposed ILS algorithm. This algorithm uses a Randomized Variable Neighborhood Descent (RVND) algorithm in the local search phase and a Set Partitioning-like (SP) model to refine the solutions, leading to an ILS-RVND-SP algorithm.

The choice of an ILS approach to solve the generic model is due to its success in solving other VRP variants, as we can see in Subramanian et al. (2013), Penna et al. (2019), and Osorio-Mora et al. (2023). The implementation of a MIP model to refine solutions were also used by Subramanian et al. (2013), Pham et al. (2022) (a Set Covering model was implemented) and many others.

The proposed ILS-SP algorithm with multiple restarts follows the structure described by the Algorithm 1, where the function $f(.)$ computes the total cost for a given solution.

Preliminary results on the VRPLTW and other variants showed that solving the SP formulation at the end of the ILS-SP leads to worst solutions than solving during the execution of the ILS. Then, we opted to solve the SP after each $\lfloor \delta \times iter_{max} \rfloor$ iterations without improvement, where δ is a parameter set to 0.3. This approach gives better results as it gives more opportunity to the ILS to explore the space near to a good solution found by the SP model. We adopted this approach for all variants except for the VRPDA, for which it is better to solve the SP formulation only at the end of the ILS-SP execution.

The construction of the initial solution is described in Subsection 4.1. The operators and the local search algorithm is described in Subsection 4.2. The perturbations procedures are presented in Subsection 4.4. Finally, the SP-like formulation is presented in Subsection 4.5.

Algorithm 1 ILS-SP

```
1: procedure ILS-SP(restarts, iter_max)
2:     P ← ∅
3:     for i = 1 : restarts
4:         s ← InitialSolution()
5:         s ← LocalSearch(s)
6:         P ← P ∪ s
7:         iter ← 0
8:         while iter < iter_max
9:             s' ← Perturbation(s)
10:            s' ← LocalSearch(s')
11:            P ← P ∪ s'
12:            if This Problem ≠ VRPDA and (iter mod ⌊δ × iter_max⌋) = 0 then
13:                s' ← SetPartitioning(P)
14:            end if
15:            if f(s') < f(s) then
16:                s ← s'
17:                iter ← 0
18:            else
19:                iter ← iter + 1
20:            end if
21:        end while
22:    end for
23:    s ← SetPartitioning(P)
24: end procedure
```

4.1 Initial Solution

The initial solution is constructed by randomly selecting customers to be assigned. Once the assignments are fixed, the problem becomes a classical Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) or becomes a classical CVRP when time windows constraints are not considered. To solve the resulting CVRPTW or CVRP instance, we use the Hybrid Genetic Search (HGS) Algorithm proposed by respectively Wouda et al. (2024) and Vidal (2022).

Note that not necessarily all lockers have a customer assigned to them initially. In other words it could exist some lockers that are not visited by any solution route. We refer them as "closed" lockers. For each closed locker, we maintain the best position where it should be inserted in the solution if a customer becomes assigned to it during the local search. This position is initialized according to the smallest increase in the travel cost. This information is then stored for later use.

4.2 Local Search Phase

In the local search phase, we implement 12 different neighborhood operators that can be separated into three classes: inter-route operators, intra-route operators, and

assignment operators. The way we deal with time windows constraints is described in Subsection 4.6.

We employ two very classical inter route operators: the Swap(1,1) and Shift(1,0), where:

- Swap(1,1) (M_1): Exchange two nodes from two different routes;
- Shift(1,0) (M_2): Move a node from one route to another one.

The intra-route operators used are: 2-opt, Exchange, Reinsertion and Or-opt2, where:

- 2-opt (M_3): A section of a route is inverted;
- Exchange (M_4): The position of two nodes in the same route is exchanged;
- Reinsertion (M_5): A node is reinserted in another position of the same route;
- Or-opt2 (M_6): Two consecutive nodes are reinserted in another position of the same route.

Whenever a movement is accepted, we recompute the best position for each closed locker.

The specific assignment operators used are the so-called Assign$_a$(1, 0), Assign$_h$(1, 0), Assign$_a$(2, 0), Assign$_h$(2, 0), Unassign(1), and Assig(1,1), Assig(2,1), where:

- Assign$_h$(1, 0) (M_7): A customer c previously visited at home is assigned to a locker l. If the locker is close, we insert it in its best position.
- Assign$_a$(1, 0) (M_7'): A customer c previously assigned to a locker l_1 is assigned to a locker l_2. If l_1 becomes closed, we store its current position as the best one. If l_2 is closed, we insert it in its best position.
- Assign$_h$(2, 0) (M_8): 2 customers that are visited at home are assigned simultaneously to a locker. If the locker is closed, we insert it in its best position.
- Assign$_a$(2, 0) (M_8'): 2 customers that are assigned to a given locker l_1 are assigned to another locker l_2. If l_1 becomes closed, then its current position is stored as its best position. If l_2 is closed, we insert it in its best position.
- Unassign(1) (M_9): A customer c that is assigned to a locker l is now visited at home and it is inserted in the best position among all the solution routes. If l becomes closed, then its current position is stored as its best position.
- Assig(1,1) (M_{10}): A customer c_1 assigned to a locker l_1 and a customer c_2 assigned to a locker l_2 are reassigned to l_2 and l_1, respectively;
- Assig(2,1) (M_{11}): Consider the customers c_1 and c_2 that are assigned to the locker l_1 and the customer c_3 that is assigned to the locker l_2. Then, c_1 and c_2 are reassgined to the locker l_2, while customer c_3 is reassigned to the locker l_1.

According to the VRPDA definition, the customers can not be visited directly by the vehicles. Therefore, the operators Assign$_h$(1, 0), Assign$_h$(2, 0), and Unassign(1) are not applied to this problem.

Figures 3, 4 5 and 6 illustrate the assignment operators, given the solution in Figure 2.

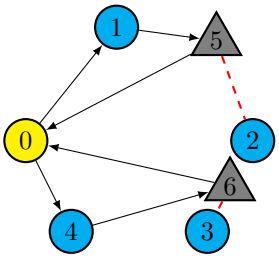

Fig. 2: Solution to be explored.

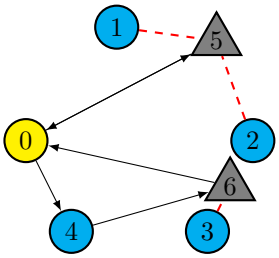

Fig. 3: Assig$_h(1, 0)$: Customer 1 is assgined to locker 5.

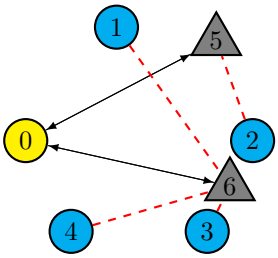

Fig. 4: Assig$_h(2, 0)$: Customer 1 and 4 are assigned to locker 6.

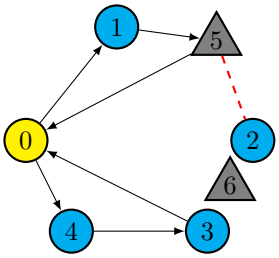

Fig. 5: Unassig(1): Customer 3 is unassigned from locker 6, which becomes closed.

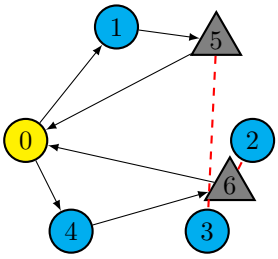

Fig. 6: Assig(1,1): Lockers 5 and 6 have its assignments exchanged.

All operators described before use the best improvement strategy and are explored exhaustively. Only feasible moves with respect to vehicle and locker capacity are considered.

The operators are used in an RVND algorithm that randomly selects an operator from the list of operators, and, if the selected operator do not improve the current solution, this operator is removed from the list and another one is chosen. If the operator improves the current solution, the list of operators is "repopulated" with the

ones that were removed and another operator is chosen at random. This procedure goes on until the list becomes empty. Algorithm 2 describes our RVND algorithm.

Algorithm 2 RVND

```
1:  procedure RVND(solution)
2:      s ← solution
3:      NG ← {M_1, M_2, ... M_11}                    ▷ List of operators.
4:      while |NG| > 0
5:          ng = random(NG)
6:          s' ← ng(s)                               ▷ Apply the operator.
7:          if f(s') < f(s) then
8:              s ← s'
9:              NG ← {M_1, M_2, ... M_11}            ▷ Repopulate the list of operators
10:         else
11:             NG ← NG \ {ng}                       ▷ Remove ng operator
12:         end if
13:     end while
14:     return s
15: end procedure
```

4.3 Acceleration mechanisms

To improve the performance of the algorithm, we employ two techniques to accelerate it, Memoization Move Descriptor (MMD) and Memoization Solution Descriptor (MSD) as described in Malheiros et al. (2021).

The MMD works as follows: For a given solution and a local search operator, it stores information about the best improving move or the absence of one. Before starting a local search operator, the algorithm checks if the solution has already been explored by it. If it has, MMD returns the stored information. If not, the search proceeds normally, and the resulting information is stored for future reference.

The MSD extends the MMD by storing the best solution found after a restart. If this solution is encountered again in another restart, the search is stopped, and the algorithm restarts. Similarly, the solution found by the RVND is stored, and if this solution is found again by a local search operator, the RVND is stopped.

To efficiently use these mechanisms, we employ hash functions and efficient data structures for storage.

4.4 Perturbations procedures

The perturbation procedure aims to escape from a local optimum and to permit the algorithm to explore different solutions. To do so, for each ILS iteration, we randomly apply one of the following four perturbation procedures:

- RandomAssign(1,0) (P_1): Performs randoms Assig$_h$(1,0) and Assig$_a$(1,0);

- RandomSwap(1,1) (P_2): Performs randoms Swap(1,1);
- RandomShift(1,0) (P_3): Performs randoms Shift(1,0);
- DestroyLocker (P_4): Select a random locker l from a solution route r with more than two customers assigned to it. The locker l becomes closed and the customers that assigned to it are then inserted at the end of route r. For VRPDA instances, the customers are randomly assigned to other lockers.

If it's not possible to perform P_1, P_2 or P_3 due to the capacities constraints, we perturb the solution with a random 2-opt.

4.5 Set Partitioning-like model

Although we use several local search and perturbation procedures, the proposed algorithm may have difficulty finding good solutions in some instances, since in the generic model, there are two types of decisions: routing and assignment. To address this issue, we implement a Set Partitioning model. To use such a model, we store the local optima solutions found by the RVND algorithm in a pool, $\mathcal{P}$, and solve the SP model.

The SP-like model uses a binary variable λ_r that is equal to 1 if the route $r \in \mathcal{P}$ is used in the solution and 0 otherwise. Additionally, there are two constants: c_r, which gives the cost of the route r, and β_{lr}, which is equal to 1 if the locker l is visited by the route r and 0 otherwise. Similarly, γ_{ir}, which is equal to 1 if the customers i is visited or assigned in the route r and 0 otherwise. The formulation follows.

$$\text{Min} \sum_{r \in \mathcal{P}} c_r \lambda_r \tag{1a}$$

$$\text{s.t.} \sum_{r \in \mathcal{P}} \gamma_{ir} \lambda_r = 1, \quad i \in V_C \tag{1b}$$

$$\sum_{r \in \mathcal{P}} \beta_{lr} \lambda_r \leq 1, \quad l \in V_L \tag{1c}$$

$$\lambda_r \in \{0, 1\}, \quad r \in \mathcal{P} \tag{1d}$$

The objective function (1a) minimizes the cost of the solution. Constraints (1b) guarantee that each customer is visited or assigned exactly once. Constraints (1c) ensure that the locker is visited at most once. (1d) are the domain constraint of the variables λ.

4.6 Dealing with time windows

To solve the variants with time windows constraints, we employ the approach proposed by Vidal et al. (2013) to compute the time window violation in constant time.

In our approach, we do not apply a penalty cost for time window violations. Instead, we adopt a hierarchical approach. Given a solution s with time window violation τ and a solution s' with time window violation τ', we use the following condition to accept a movement or solution, where $f(.)$ is the cost function:

$$\begin{cases} \tau' < \tau, & \text{accept} \\ \tau' = \tau \wedge f(s') < f(s), & \text{accept,} \\ otherwise & \text{reject} \end{cases}$$

5 Computational Experiments

In this section, we present the results of our algorithm applied to the VRPLTW, VRPPSDL and VRPDA literature benchmark instances respectively used by Buzzega and Novellani (2023), Mancini and Gansterer (2021) and Ghoniem et al. (2013). For VRPLTW, we follow the literature and consider more three variants: without time windows (VRPL), with single route (TSPLTW) and with single route and without time windows (TSPL).

The tables in this section summarize the best and average results. However, we provide an online supplementary material with detailed results for each instance.

For each problem, the algorithm was run five times with five different seeds.

The algorithm was implemented in Julia, version 1.10, and run on an Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz with Ubuntu 18.04. The solver used to solve the SP models was CPLEX 12.10. The parameters of the HGS algorithms used for the construction of the initial solution are present in appendix A.

5.1 Results on VRPLTW instances

Buzzega and Novellani (2023) generated VRPLTW instances by adapting 40 instances proposed in Dumas et al. (1995) and Gendreau et al. (1998) for the traveling salesman problem with time windows (TSPLTW) problem. The instances are divided in two sets: a set with 20 instances with time windows of amplitude of 20 and a set of 20 instances with time windows of amplitude 100. Each one of these two sets is is divided into four groups of five instances each, each one with the same number of customers $|V_C| = 20, 40, 60, 100$ and with the same number of lockers $|V_L|$ which is equal to ten percent of the number of customers ($|V_L| = 0.1 \times |V_C|$). The (x, y) coordinate of each customer is given in the original TSPLTW instance and the $|V_L|$ lockers are included in the instance by adding the first $|V_L|$ vertices of the following set of coordinates: $(25.0, 25.0)$, $(12.5, 12.5)$, $(37.5, 37.5)$, $(12.5, 25.0)$, $(37.5, 12.5)$, $(12.5, 25.0)$, $(25.0, 37.5)$, $(37.5, 25.0)$, $(25.0, 12.5)$, and $(0, 0)$. The time windows for the depot and for the customers are given in the original TSPLTW instance. There is no time windows for the lockers, that is, $[a_l, b_l] = [0, +\infty], \forall l \in V_L$. Let d_{ij} be the euclidean distance between two nodes i and j. For each edge e, the travel cost c_e and travel time t_e are given by $c_e = t_e = d_{ij}$. For each locker l and each customer i, the assignment cost f_{il} is given by $f_{il} = d_{il}$. For each customer $i \in V_C$, the parcel demand is unitary ($q_i = 1$) and the value of R for the sets $\phi(i)$ is equal to 15. The capacity of each locker l is equal to 5 ($Q_l = 5, \forall l \in V_L$) while the capacity of each vehicle is given by $Q = \frac{|V_C|}{2}$. For the objective function parameters, the values considered are $\alpha_1 = 1$ and $\alpha_2 = 0.5$. Recall that in VRPLTW the usage of vehicles have no cost ($\alpha_3 = 0$) and there is no service time for customers and lockers ($s_i = 0, \forall i \in V_C \cup V_L$). For the variants without time windows and/or

single route, we use the same instances relaxing the time windows and the vehicle capacity constraints, respectively.

5.1.1 Parameters Tuning

We tested the ILS-RVND-SP with fifteen different configurations, summarized on Table 1.

Parameter	Tested values
$restarts$	{5, 10, 15}
$iter_{max}$	$\{\|V_C\|, 3\|V_C\|, 5\|V_C\|, 7\|V_C\|, 10\|V_C\|\}$

Table 1: Tested values for ILS-RVND-SP parameters

Table 2 presents two average statistics: the average percentage difference between the best known solution and the average solution of the proposed algorithm (Column *Avg. gap(%)*) and the average running time in seconds spent by our algorithm (Column *Avg. time(s)*) considering five runs for all the 40 VRPLTW instances and all 40 VRPL instances. The best known solutions used in this table were provided by the authors of the paper Buzzega and Novellani (2023) for the VRPLTW and variants instances.

Among the configurations, we selected $(15, 7|V_C|)$ for our experiments, as it offers a good trade-off between solution quality and running time for both VRPL and VRPLTW.

	VRPL		VRPLTW	
$(restarts, iter_{max})$	Avg. gap (%)	Avg. time (s)	Avg. gap (%)	Avg. time (s)
(5, $1\|V_C\|$)	0.74	4.78	1.41	10.27
(5, $3\|V_C\|$)	0.42	12.91	0.68	29.14
(5, $5\|V_C\|$)	0.28	21.23	0.50	47.45
(5, $7\|V_C\|$)	0.19	29.15	0.27	65.64
(5, $10\|V_C\|$)	0.05	41.95	0.25	90.35
(10, $1\|V_C\|$)	0.33	8.1	0.93	18.45
(10, $3\|V_C\|$)	0.06	31.62	0.43	31.62
(10, $5\|V_C\|$)	-0.05	44.3	0.25	89.36
(10, $7\|V_C\|$)	-0.04	51	0.05	111.6
(10, $10\|V_C\|$)	-0.10	76.07	0.08	179.13
(15, $1\|V_C\|$)	0.14	11.2	0.69	24.78
(15, $3\|V_C\|$)	-0.04	31.67	0.23	69.89
(15, $5\|V_C\|$)	-0.05	51.8	0.05	116.29
(15, $7\|V_C\|$)	-0.12	71.91	-0.03	162.21
(15, $10\|V_C\|$)	-0.17	102.9	-0.04	234.06

Table 2: Summary informations for the tested parameterization.

5.1.2 Comparison with the best known solutions

Tables 3 - 6 present the results of our ILS-RVND-SP algorithm where the instances are grouped according the number of customers. The column $|V_C|$ denotes the number of customers of the corresponding group. The column *#Inst* indicates the number of instances of the corresponding group. In the column *Run*, we report the *Best* run (the one that achieved the best solution), and the *Average* run (the average result of all five runs).

The column *#BKS* displays the number of instances where the best known solution was either found or improved. The column *Avg. gap(%)* shows the average percentage gap between the best known solutions and the solutions found by our algorithm. The column *Std. Dev. gap(%)* presents the standard deviation of the gap. The column *#gap* ≤ 1 lists the number of instances where the gap was at most 1%. The *Avg. time* column presents the average runtime of the algorithm.

As its simple to adapt the ILS-RVND-SP algorithm to solve the single vehicle variant (forbidding inter-route movements), we also test it on these instances.

Table 3 summarizes the results obtained for the TSPL with the following parameterization ($restarts$, $iter_{max} = (10, 3|V_C|)$). It can be seen that, among the best runs, the algorithm found or improved the best known solution in 34 out of 40 instances. Additionally, none of the instances across all run types exhibited a gap larger than 1%, demonstrating the algorithm's effectiveness.

$\|V_C\|$	#Inst	Run	#BKS	Avg. gap (%)	Std. Dev. gap (%)	#gap $\leq$ 1%	Avg. time (s)
20	10	Best	10	0.00	0.00	10	0.54
		Average	-	0.00	0.00	10	
40	10	Best	10	0.00	0.00	10	5.00
		Average	-	0.00	0.00	10	
60	10	Best	8	0.08	0.22	10	22.97
		Average	-	0.22	0.22	10	
100	10	Best	6	0.06	0.09	10	331.83
		Average	-	0.28	0.22	10	

Table 3: Results for the TSPL.

In Table 4, results for the TSPLTW with the following parameterization ($restarts$, $iter_{max}$) $= (10, 5|V_C|)$ are presented. The algorithm improved several instances with 100 customers and found or improved the best-known solution in 34 out of 39 instances, with an average of improvement of 3.83%! The algorithm also performs very well when considering the average of the solutions found, with only 3 instances having a gap bigger than 1%.

For the VRPL (Table 5), the algorithm found or improved 38 out of 40 best-known solutions in the best run, with almost all instances having a gap smaller than 1% in a reasonable amount of time. We can also see that, on average, the algorithm also performs very well, with only 1 instance having a gap bigger than 1%.

Table 6 provides the results for the VRPLTW. Despite the added complexity of time windows, the algorithm found or improved 31 out of the 40 best-known solutions.

$\|V_C\|$	#Inst	Run	#BKS	Avg. gap (%)	Std. Dev. gap (%)	#gap $\leq$ 1%	Avg. time (s)
20	10	Best	10	0.00	0.00	10	1.64
		Average	-	0.11	0.34	10	
40	10	Best	9	0.02	0.07	10	7.23
		Average	-	0.07	0.13	10	
60	10	Best	7	0.04	0.13	10	25.48
		Average	-	0.46	0.55	8	
100	9	Best	8	-3.83	5.60	9	207.19
		Average	-	-3.06	5.68	9	

Table 4: Results for the TSPLTW.

$\|V_C\|$	#Inst	Run	#BKS	Avg. gap (%)	Std. Dev. gap (%)	#gap $\leq$ 1%	Avg. time (s)
20	10	Best	10	0.00	0.00	10	0.89
		Average	-	0.00	0.00	10	
40	10	Best	10	0.00	0.00	10	8.82
		Average	-	0.21	0.45	10	
60	10	Best	8	0.05	0.45	9	37.10
		Average	-	0.31	0.47	9	
100	10	Best	10	-1.37	1.42	10	240.83
		Average	-	-1.01	1.32	10	

Table 5: Results for the VRPL.

Even in cases where the best-known solution was not achieved, the average gap remains low, the algorithm found solutions with a gap smaller than 1% on all instances, considering its best run. When considering the average of the solutions, only 5 instances got a gap bigger than 1%.

$\|V_C\|$	#Inst	Run	#BKS	Avg. gap (%)	Std. Dev. gap (%)	#gap $\leq$ 1%	Avg. time (s)
20	10	Best	10	0.00	0.00	10	2.75
		Average	-	0.06	0.19	10	
40	10	Best	8	0.04	0.12	10	20.22
		Average	-	0.27	0.33	10	
60	10	Best	5	0.25	0.37	10	74.89
		Average	-	0.70	0.59	7	
100	10	Best	8	-1.98	2.70	10	550.96
		Average	-	-1.14	2.60	8	

Table 6: Results for the VRPLTW.

5.2 Results on VRPPSDL instances

The instances are separated into three sets, each instance containing five lockers. The number of customers are 25, 50, or 75. Customers were randomly placed within a predefined area represented as a 10x10 km square. The depot is situated in the southern part

of this area, with lockers strategically positioned in the southeast, southwest, northeast, northwest, and center. For all instances, its considered a time horizon equivalent to 720 minutes. This time horizon is divided into 12 slots, each lasting 60 minutes.

For each edge e, the travel cost c_e and travel time t_e are given by $c_e = t_e = 3 \times d_{ij}$. Each customer has a service time of 5, and each locker has a fixed service time of 10. For each locker l and each customer i, the assignment cost id fixed to five ($f_{il} = 5$). For each customer $i \in V_C$, the parcel demand is unitary ($q_i = 1$) and the value of R for the sets $\phi(i)$ is equal to 5. Among the lockers, four have the same capacity, while one locker has a larger capacity, enabling routes that only visit lockers to be feasible. For the objective function parameters, the values considered are $\alpha_1 = \alpha_2 = \alpha_3 = 1$. Recall that in VRPPSDL there is no capacity for the vehicles ($Q = +\infty$).

5.2.1 Comparison with the best known solutions

Table 7 shows the results of the proposed algorithm on the instances proposed by Mancini and Gansterer (2021) where the instances are grouped according to the number of customers. Each best known solution is the best solution found by the methods proposed by Mancini and Gansterer (2021), Grabenschweiger et al. (2021) and Buzzega and Novellani (2023). The column labels are the same of tables 3 - 6. Due to the similarity between the problems, the parameters used for VRPPSDL instances are the same used for the VRPLTW ones. Besides the fact that the algorithm had a different performance when compared with the VRPLTW instances, the algorithm found good solutions in a small amount of time. Only 5 best-known solution was not achieved by the algorithm when considering the best run, and only 2 instances got a gap bigger than 1% on the average.

$\lvert V_C \rvert$	#Inst	Run	#BKS	Avg. gap (%)	Std. Dev. gap (%)	#gap ≤1%	Avg. time (s)
25	10	Best	10	0.00	0.00	10	6.02
		Average	-	0.08	0.15	10	
50	10	Best	10	0.00	0.00	10	36.64
		Average	-	0.09	0.22	10	
75	10	Best	5	0.25	0.34	10	165.75
		Average	-	0.60	0.38	8	

Table 7: Results on the VRPPSDL instances.

5.2.2 Comparison with the best literature heuristic algorithm

To the best of our knowledge, the best heuristic algorithm for the VRPPSDL is the metaheuristic proposed by Grabenschweiger et al. (2021). Table 8 compares the results obtained by their and our algorithms, using the best run results. To do so, we convert their time using factor obtained through comparison of our and the literature processor at the site cpubenchmark.com. The instances are again grouped according to number of customers. Column $|V_C|$ shows the number of customers in each instance in the corresponding group. Column $gap_M\,(\%)$ shows the average gap between their solution and ours. Column T_M/T is the ratio of their average running time and ours.

$\|V_C\|$	gap_M (%)	T_M/T
25	0.00	3.79
50	0.00	4.10
75	-0.08	3.05

Table 8: Comparison with Grabenschweiger et al. (2021)

Our algorithm outperforms the one developed by Grabenschweiger et al. (2021) in terms of solution quality and running time, as our algorithm can find the same solution and improve some instances in much less time.

5.3 Results on VRPDA instances

Customer and lockers coordinates were randomly generated using a uniform distribution within a two-dimensional Euclidean space. These locations are situated 25 to 75 miles from the depot at the origin, $(0, 0)$. Customer demand was also randomly assigned, ranging from 1 to 5 parcels, with the following probabilities: $P(q_i = 1) = 0.5$, $P(q_i = 2) = 0.2$, $P(q_i = 3) = 0.1$, $P(q_i = 4) = 0.1$, $P(q_i = 5) = 0.1$. The lockers have no capacities and the vehicle capacity is set to 25.

In the set of VRPDA instances proposed by Ghoniem et al. (2013), the number of lockers varies from 10 to 50 while the number of customers varies from 20 to 90. The vehicle capacity is $Q = 25$, $\forall l \in V_L$. The customer's demand are integer numbers varying from 1 to 5. For each edge e, the travel cost c_e is given by $c_e = \lfloor d_e \rfloor$. The cost of assigning a customer to a locker is given by $f_{(i,j)} = \lfloor d_{(i,j)} \rfloor$. For the objective function, the tested values for the parameter α_1 are $\alpha_1 = 0.25, 0.50, 0.75$, while the value of the partamenter α_2 is given by $\alpha_2 = 1 - \alpha_1$. Remember that in VRPDA the usage of vehicles have no cost ($\alpha_3 = 0$), time windows constraints is not considered, and there is no capacity for the lockers.

Since the triangular inequality for the travel costs does not hold in the VRPDA instances used in this paper, a locker without any assignment could be visited by some route at the optimal solution. An intuitive way to deal with this possibility would be to compute the shortest path for each pair of nodes. However, it would not ensure one of the constraints of the problem, which is that each locker can be visited at most once considering all the routes. In this context, we do not compute the shortest path between the nodes and we introduce an additional local search operator called InsertEmptyLocker specific for the VRPDA instances. This operator involves inserting an closed locker into some solution route without any assignment.

5.3.1 Parameter Tuning

We tested six different parameterizations, where $restarts \in \{20, 50, 100\}$ and $iter_{max} \in \{50, 100\}$. Table 9 shows some statistics for each combination of parameters for each value of α_1. Column $(restarts, iter)$ is the parameter configuration. Column *Avg. gap (%)* shows the average gap between the best solution found by Reihaneh and

Ghoniem (2018) and the ILS-RVND-SP algorithm. Column *Avg. time (s)* shows the average running time of ILS-RVND-SP.

	$\alpha_1 = 0.25$		$\alpha_1 = 0.5$		$\alpha_1 = 0.75$	
$(restarts, iter_{max})$	Avg. gap (%)	Avg. time (s)	Avg. gap (%)	Avg. time (s)	Avg. gap (%)	Avg. time (s)
(20, 50)	0.30	2.07	-0.31	2.10	0.34	1.98
(20, 100)	0.08	4.21	-0.52	3.98	-0.08	3.63
(50, 50)	0.09	5.01	-0.51	4.84	-0.16	4.77
(50, 100)	-0.09	9.51	-0.64	9.16	-0.40	9.04
(100, 50)	-0.04	9.37	-0.63	9.41	-0.42	9.52
(100, 100)	-0.15	18.06	-0.72	18.11	-0.57	20.35

Table 9: Statistics for each combination of parameters.

Parameterization $(100, 100)$ consistently yields the lowest gap across all values of α_1, making it a good choice despite being the most time-intensive configuration.

5.3.2 Comparison with literature

Table 10 shows the results of our algorithm on VRPDA instances. The instances are grouped according to the number of lockers. The column V_L shows the number of lockers on the tested instances in the corresponding group where, for each value of V_L, there is 20 instances with different number of customers $|V_C|$. Column gap_b *(%)* is the average gap between the best solutions found by Reihaneh and Ghoniem (2018) and our algorithm, while column gap_a *(%)* shows the average gap between the best solution of literature algorithm and our average solution. Columns *T(s)* and *Lit.T(s)* present average running time of our and literature algorithms respectively, where the literature time were adjusted using a factor equal to 0.77 that was obtained through a comparison of the processors used by the literature and by us in the site www.cpubenchmark.com.

We can observe from Table 10 that the proposed ILS-RVND-SP algorithm outperforms the metaheuristic proposed by Reihaneh and Ghoniem (2018) in both solution quality and runtime. The average solution provided by the ILS-RVND-SP is better than the best solution found by Reihaneh and Ghoniem (2018). Moreover, when comparing the best solutions generated by our algorithm, the improvement becomes even more evident. In addition to improving several solutions, our algorithm is also more than twice as fast, on average.

Table 11 presents for each value of the parameter α_1, the number of instances on which our algorithm found, based on the best run, a better solution (row *#win*), the same solution (row *#tie*) and a worst solution (*#loss*).

Observing Table 11, we can note that our algorithm finds better or equal solutions on 269 out of 300 compared instances.

	$\alpha_1 = 0.25$				$\alpha_1 = 0.50$				$\alpha_1 = 0.75$			
$\|V_L\|$	$gap_b(\%)$	$gap_a(\%)$	T(s)	Lit. T(s)	$gap_b(\%)$	$gap_a(\%)$	T(s)	Lit. T(s)	$gap_b(\%)$	$gap_a(\%)$	T(s)	Lit. T(s)
10	-0.05	-0.05	7.18	3.36	0.00	0.00	6.70	4.51	-0.04	-0.03	6.21	4.12
25	-0.02	0.04	9.44	19.87	-0.01	0.02	9.40	23.32	0.08	0.20	8.36	13.07
35	-0.30	-0.19	18.43	39.97	-0.77	-0.63	18.64	47.72	-0.25	-0.09	17.67	42.37
40	-0.09	0.06	23.82	52.12	-1.77	-1.62	23.76	59.83	-1.15	-0.86	22.83	61.16
50	-0.29	-0.09	31.45	70.72	-1.04	-0.85	32.06	78.26	-1.47	-1.16	33.33	69.64
Average	-0.15	-0.05	18.06	37.20	-0.72	-0.62	18.11	42.72	-0.57	-0.39	17.68	38.07

Table 10: Comparison with literature.

	$\alpha_1 = 0.25$	$\alpha_1 = 0.50$	$\alpha_1 = 0.75$
#win	39	52	41
#tie	44	41	52
#loss	17	7	7

Table 11: Number of better, equal and worst solutions.

6 Conclusion

In this paper, we presented a generic problem that generalizes the Vehicle Routing Problem with Lockers (Buzzega and Novellani, 2023) Vehicle Routing Problem with Private and Shared Delivery Location (Mancini and Gansterer, 2021) and Vehicle Routing Problem with Demand Allocation (Ghoniem et al., 2013). To solve it, we developed a hybrid algorithm, that combines Iterated Local Search, Randomized Variable Neighborhood Descent and a Set Partitioning-like model. The algorithm can find or improve 431 out of 490 best known solutions. It also outperforms the existing metaheuristic for the VRPPSDL and VRPDA, in terms of solution quality and running time.

For future researches, we aim to apply this algorithm to other variants, develop more sophisticated local search operators, to exploit the structure of the problem. We also aim to develop an exact algorithm, based on branch-cut-and-price algorithm.

A Parameters of the HGS algorithm

The following tables show the parameters of the HGS algorithm. They were chosen in such a way that the algorithm becomes fast and still leads to high-quality solutions.

For the CVRP instance constructed for the TSPL/VRPL initial solution, the following parameters were used, while the remaining were set to default values. For a more detailed explanation of those parameters, we refer to Vidal (2022).

Parameter	Chosen value
nbIter	10
μ	3
λ	10

Table 12: Parameters to solve the resulting CVRP instance.

For the VRPTW instances that results from the TSPLTW/VRPLTW initial solution construction, the following parameters were used:

For a more detailed explanation of those parameters, we refer to Wouda et al. (2024).

Parameter	Chosen value
nbIter	1000
NoImprovement	20
Penalty Updates	10
μ	10
λ	40

Table 13: Parameters to solve the results VRPTW instance.

References

Buzzega, G., Novellani, S.: Last mile deliveries with lockers: Formulations and algorithms. Soft Computing **27**(18), 12843–12861 (2023)

Dumas, Y., Desrosiers, J., Gelinas, E., Solomon, M.M.: An optimal algorithm for the traveling salesman problem with time windows. Operations research **43**(2), 367–371 (1995)

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks: An International Journal **44**(3), 216–229 (2004)

Grabenschweiger, J., Doerner, K.F., Hartl, R.F., Savelsbergh, M.W.: The vehicle routing problem with heterogeneous locker boxes. Central European Journal of Operations Research **29**, 113–142 (2021)

Gendreau, M., Hertz, A., Laporte, G., Stan, M.: A generalized insertion heuristic for the traveling salesman problem with time windows. Operations Research **46**(3), 330–335 (1998)

Ghoniem, A., Scherrer, C.R., Solak, S.: A specialized column generation approach for a vehicle routing problem with demand allocation. Journal of the Operational Research Society **64**(1), 114–124 (2013)

Mancini, S., Gansterer, M.: Vehicle routing with private and shared delivery locations. Computers & Operations Research **133**, 105361 (2021)

Malheiros, I., Ramalho, R., Passeti, B., Bulhões, T., Subramanian, A.: A hybrid algorithm for the multi-depot heterogeneous dial-a-ride problem. Computers & Operations Research **129**, 105196 (2021)

Osorio-Mora, A., Escobar, J.W., Toth, P.: An iterated local search algorithm for latency vehicle routing problems with multiple depots. Computers & Operations Research **158**, 106293 (2023)

Pham, Q.A., Hà, M.H., Vu, D.M., Nguyen, H.H.: A hybrid genetic algorithm for the vehicle routing problem with roaming delivery locations. Proceedings of the

International Conference on Automated Planning and Scheduling **32**(1), 297–306 (2022) https://doi.org/10.1609/icaps.v32i1.19813

Penna, P.H.V., Subramanian, A., Ochi, L.S., Vidal, T., Prins, C.: A hybrid heuristic for a broad class of vehicle routing problems with heterogeneous fleet. Annals of Operations Research **273**, 5–74 (2019)

Pochet, Y., Wolsey, L.A.: Production Planning by Mixed Integer Programming vol. 149. Springer, ??? (2006)

Reihaneh, M., Ghoniem, A.: A multi-start optimization-based heuristic for a food bank distribution problem. Journal of the operational research society **69**(5), 691–706 (2018)

Reihaneh, M., Ghoniem, A.: A branch-and-price algorithm for a vehicle routing with demand allocation problem. European Journal of Operational Research **272**(2), 523–538 (2019)

Solak, S., Scherrer, C., Ghoniem, A.: The stop-and-drop problem in nonprofit food distribution networks. Annals of Operations Research **221**, 407–426 (2014)

Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Computers & Operations Research **40**(10), 2519–2531 (2013)

Tilk, C., Olkis, K., Irnich, S.: The last-mile vehicle routing problem with delivery options. Or Spectrum **43**(4), 877–904 (2021)

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Computers & operations research **40**(1), 475–489 (2013)

Vidal, T.: Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. Computers & Operations Research **140**, 105643 (2022)

Vincent, F.Y., Susanto, H., Jodiawan, P., Ho, T.-W., Lin, S.-W., Huang, Y.-T.: A simulated annealing algorithm for the vehicle routing problem with parcel lockers. IEEE Access **10**, 20764–20782 (2022)

Wouda, N.A., Lan, L., Kool, W.: Pyvrp: A high-performance vrp solver package. INFORMS Journal on Computing (2024)

Zhou, L., Baldacci, R., Vigo, D., Wang, X.: A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. European Journal of Operational Research **265**(2), 765–778 (2018)