Optimization Letters (2023) 17:2033–2063
https://doi.org/10.1007/s11590-023-01990-1

ORIGINAL PAPER

A general VNS for the multi-depot open vehicle routing problem with time windows

Sinaide Nunes Bezerra[1] · Sérgio Ricardo de Souza[1] ·
Marcone Jamilson Freitas Souza[2]

Received: 27 December 2021 / Accepted: 6 February 2023 / Published online: 13 March 2023

Abstract

This paper presents an algorithm based on the variable neighborhood search (VNS) metaheuristic, called smart general VNS (SGVNS), to solve the multi-depot open vehicle routing problem with time windows (MDOVRPTW). For the problem, two single-objective approaches are proposed for cost assessment: one for reducing the total distance covered and the other for reducing the total number of vehicles used and, after, the total distance covered. SGVNS involves the perturbation and local search phases. In the perturbation phase, gradual changes are carried out in the neighborhoods to expand the diversification of solutions and escape from local optima. The random combination of specific neighborhood structures is used in the local search to refine the solution generated in the previous phase. As no instances are known in the literature for MDOVRPTW, the computational tests are executed in two groups of classic MDVRPTW instances, involving up to 960 customers, 12 depots, and 120 vehicles. The present study made it possible to investigate cost improvements through the use of the MDOVRPTW model when compared to the MDVRPTW. There was a reduction in the distance covered in all instances evaluated. The total distance covered decreased by 12.07% in one of the reference groups and 10.43% in the other. For the first group, the fleet reduction occurred in 75% of the instances. In the second group, there was a reduction in all instances. It corresponds to – 10.42% and – 24.13% of the total vehicles used in each group, respectively. The SGVNS algorithm proved effective for the two problems for which it was applied, either in reducing the total traveled distance or in reducing the fleet.

Keywords Multi-depot open vehicle routing problem with time windows · Variable neighborhood search · Neighborhood · Metaheuristic

✉ Sinaide Nunes Bezerra
sinaide@hotmail.com

Extended author information available on the last page of the article

1 Introduction

This article addresses the multi-depot open vehicle routing problem with time windows (MDOVRPTW). In this problem, a homogeneous fleet of vehicles must depart from a set of depots and serve a set of consumers within a time window for each one. Unlike traditional vehicle routing problems, the used vehicles are not required to return to the original depots after delivering goods to customers. MDOVRPTW consists of finding the shortest routes and, hence, the objective is to minimize the total distance covered by the vehicles [1, 2]. On the other hand, the current article also addresses a variant of MDOVRPTW, which we named MDOVRPTW*. This variant seeks first to minimize the number of used vehicles and, after, to minimize the total covered distance. Hence, MDOVRPTW* is a problem solved hierarchically, and this methodology considers the addition of a new vehicle as the primary cost to be minimized and, consequently, the total covered distance by these vehicles as the secondary cost. Therefore, it emulates the traditional methodology adopted for solving the vehicle routing problem with time windows (VRPTW). To our knowledge, this variant has not yet been addressed in the literature.

According to Brandão [3], Repoussis et al. [4] and Shen et al. [2], open vehicle routing problems are frequently encountered in the real-world. These problems appear when the companies do not have their own fleet of vehicles for the distribution of goods or, in some cases, the existing fleet is not enough to serve the customers and, therefore, some external vehicles must also be hired. These same characteristics are found in school bus routing problems and train or airplane route planning, as long as the routes start and end at different points. These are some examples that make it more suitable for describing various real-world problems. The number of depots involved is another important characteristic in this type of problem since it is common for companies to have several distribution centers. On the other hand, the time window maintains the adherence of this type of problem to the customers' needs by defining their best service schedules. MDOVRPTW*, by its turn, is in line with the environmental concerns of lesser use of vehicles, especially in the case of contracted fleets and autonomous vehicles, where secondary costs arising from driver hiring and vehicle maintenance are not considered. However, to our knowledge, it has not yet been addressed in the literature. These characteristics make MDOVRPTW and MDOVRPTW* suitable for treating several real problems.

As pointed out by Cordeau et al. [5] and Brandão [3], MDOVRPTW and MDOVRPTW* are extensions of MDOVRP and OVRPTW, which are NP-Hard problems, and, therefore, they are also NP-hard ones. This claim justifies the application of metaheuristics for solving them. In the current article, we apply the smart general variable neighborhood search (SGVNS) developed by Rego and Souza [6] to treat these two problems. SGVNS is a general variable neighborhood search-based algorithm in which the basic variable neighborhood descent (BVND) [7] is the local search method. However, while Rego and Souza [6] used BVND as a local search method, we applied the randomized basic variable neighborhood descent (RBVND or RVND in the nomenclature of Subramanian et al. [8]). Unlike BVND, this procedure does not require a neighborhood order calibration. Furthermore, the best order of neighborhoods can be instance-dependent, requiring calibrating these orders for each instance [8, 9]. This local search strategy has been used successfully for solving various routing problems, as in [10–12]. Finally, Bezerra et al. [13] addressed the multi-depot vehicle routing problem with time windows (MDVRPTW), aiming to minimize the vehicle fleet. This problem is the closed-route version of MDOVRPTW. The authors proposed a smart general variable neighborhood search with adaptive local search (SGVNSALS) to solve this problem. The results showed that this algorithm provided a vehicle fleet average reduction of up to 23.32% of the evaluated instances.

The main contributions of this article are:

(1) The proposition of MDOVRPTW* as a new variant of the class of multi-depot open vehicle routing problems;
(2) The adaptation of the smart general variable neighborhood search (SGVNS) for treating both MDOVRPTW and MDOVRPTW*;
(3) The development of an efficient local search method for SGVNS;
(4) An extensive analysis of the results achieved by the proposed algorithm, comparing them with the results with and without fleet reduction.

The remainder of this article is organized as follows. Section 2 introduces MDOVRPTW and MDOVRPTW*. Section 3 presents a literature review on vehicle routing problems related to the ones addressed in this work. Section 4 presents the mathematical formulations for the addressed problems. Section 5 describes the proposed SGVNS-based algorithm for treating these two problems. Section 6 reports the computational results obtained with the SGVNS algorithm, and the statistical analysis of its performance are shown in Sect. 7. Section 8 concludes the article with a general analysis of the addressed problem.

2 Multi-depot open vehicle routing problem with time windows

The multi-depot open vehicle routing problem with time windows (MDOVRPTW) consists of determining a set of routes that minimize the total distance traveled, serving all customers, respecting the vehicle capacity, the maximum duration on the routes, and the time windows in which the customers must be served. The characteristics of this problem are:

(1) The number of depots is greater than one;
(2) Each vehicle starts its route in a depot;
(3) The capacity of each vehicle is known;
(4) The vehicle fleet is homogeneous;

Table 1 Notations used in the description of MDOVRPTW and MDOVRPTW*

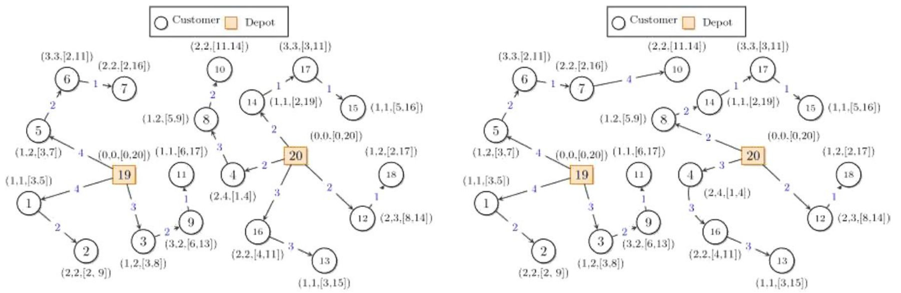
Acronym	Description
α	Weight for the number of vehicles
β	Weight for the distance traveled
$\mathcal{C}$	Set of customers, $\mathcal{C} = \{1, 2, \dots, N\}$
$\mathcal{D}$	Set of depots, $\mathcal{D} = \{N + 1, N + 2, \dots, N + M\}$
$\mathcal{V}$	Set of vertices representing depots and consumers, $\mathcal{V} = \mathcal{D} \cup \mathcal{C}$
$\mathcal{A}$	Set of arcs representing the distances (or times) between a depot and a consumer or between two consumers
$\mathcal{K}$	Set of vehicles
$\mathcal{K}_d$	Set of vehicles of a depot d
d	Depot $\in \mathcal{D}$
q_i	Demand of customer i
h_i	Service time of customer i
e_i	Start of the service window for customer i
l_i	End of the service window for customer i
MT	Maximum time spent en-route
VC	Vehicle capacity
TS_{ki}	Arrival time for vehicle k in customer i
T_k	Spent time by the vehicle k on a route
WU_k	Total waiting time to unload the vehicle k

(5) Each customer is served by only one vehicle;
(6) The total demand for each route cannot exceed the capacity of the vehicle that is on the route;
(7) The customer time window must be met;
(8) Each route has a maximum duration to be covered;
(9) The vehicles do not return to the depot.

In this article, we also address a variant of this problem, named MDOVRPTW*. The solution strategy for this problem consists of building the solution hierarchically in such a way that, first, we seek to minimize the number of vehicles used and, in the sequel, to minimize the total distance traveled. This characteristic is added by the following formulation:

(10) The total number of vehicles used must be minimized, and, after, the total distance traveled must be minimized.

These characteristics allow describing the problems under study. Table 1 summarizes the notations used in this description. These problems, in which N customers are served by M depots, are defined from a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, with $\mathcal{V}$



(a) Example of a solution to MDOVRPTW.   (b) Example of a solution to MDOVRPTW*.

Fig. 1 Example of solutions to MDVRPTW and MDVRPTW*

being a set of $(N + M)$ vertices and $\mathcal{A}$ a set of arcs. Therefore, the set $\mathcal{V}$ results from the union of two subsets, namely, the set of customers to be served, represented by $\mathcal{C} = \{1, 2, \ldots, N\}$, and the set of depots $\mathcal{D} = \{N + 1, N + 2, \ldots, N + M\}$, so that $\mathcal{V} = \mathcal{D} \cup \mathcal{C}$ and $\mathcal{D} \cap \mathcal{C} = \emptyset$. For the set $\mathcal{K}$ of vehicles, there is a subset $\mathcal{K}_d \subset \mathcal{K}$ for each depot $d \in \mathcal{D}$, where $\mid \mathcal{K} \mid = \mid \mathcal{K}_{N+1} \mid + \mid \mathcal{K}_{N+2} \mid + \cdots + \mid \mathcal{K}_{N+M} \mid$. The vehicle fleet is homogeneous so that all vehicles $k \in \mathcal{K}$ have the same capacity VC, maximum duration time of the route MT, and utilization cost α. For each consumer $i \in \mathcal{C}$, there is a positive demand q_i, which must be met in a time window $[e_i, l_i]$. There is a service time h_i that starts at the arrival time for vehicle k in customer i, TS_{ki}, such that $e_i \leq TS_{ki} \leq l_i$. For any depot $d \in \mathcal{D}$, both demand q_d and service time h_d have zero value, that is, $q_d = h_d = 0$. Each arc $(i,j) \in \mathcal{A}$ is associated with a non-negative cost c_{ij}, which can represent both the distance between the respective nodes, or the travel time, or other measurement value.

Figures 1a and 1b illustrate solutions for the MDOVRPTW and MDOVRPTW*, respectively. In these examples, the fleet of eight vehicles is divided equally among the depots. The approached instance has two depots, which serve eighteen customers. There is a homogeneous fleet of eight vehicles, each one with a capacity of $VC = 8$. The en-route time of each vehicle cannot exceed $MT = 20$. Each arc has a cost c_{ij}, given by the travel time between the node i and the node j. A tuple $(q_i, h_i, [e_i, l_i])$ is associated with the node i, being, respectively, q_i the demand, h_i the service time, and $[e_i, l_i]$ the time window for this node i.

Table 2 presents the data for the examples shown in Fig. 1. For each vehicle k, the column "Routes" shows the routes created; the column c_{ij} represents the cost between the vertices i and j; h_i represents the service time spent on the customer i; WU_k, the waiting time to unload the vehicle k; T_k represents the total time spent by the vehicle k on the route; Q_k represents the total cargo carried on the route k.

Note that the solution for MDOVRPTW shown in Fig. 1a has 7 routes, while the solution for MDOVRPTW*, shown in Fig. 1b, has 6 routes.

Table 2 Data for the examples shown in Fig. 1

Data for the MDOVRPT (Fig. 1a)								Data for the MDOVRPT* (Fig. 1b)						
k	Routes	c_{ij}	h_i	WU_k	T_k	q_i	Q_k	Routes	c_{ij}	h_i	WU_k	T_k	q_i	Q_k
1	(19, 1, 2)	(4, 2)	(0, 1, 2)	0	9	(0, 1, 2)	3	(19, 1, 2)	(4, 2)	(0, 1, 2)	0	9	(0, 1, 2)	3
2	(19, 5, 6, 7)	(4, 2, 1)	(0, 2, 3, 2)	0	14	(0, 1, 3, 2)	6	(19, 5, 6, 7, 10)	(4, 2, 1, 4)	(0, 2, 3, 2, 2)	0	20	(0, 1, 3, 2, 2)	8
3	(19, 3, 9, 11)	(3, 2, 1)	(0, 2, 2, 1)	0	11	(0, 1, 3, 1)	5	(19, 3, 9, 11)	(3, 2, 1)	(0, 2, 2, 1)	0	11	(0, 1, 3, 1)	5
4	(20, 4, 8, 10)	(2, 3, 2)	(0, 4, 2, 2)	0	15	(0, 2, 1, 2)	5	(20, 8, 14, 17, 15)	(2, 2, 1, 1)	(0, 2, 1, 3, 1)	3	16	(0, 1, 1, 3, 1)	6
5	(20, 14, 17, 15)	(2, 1, 1)	(0, 1, 3, 1)	0	9	(0, 1, 3, 1)	5	(20, 12, 18)	(2, 1)	(0, 3,2)	0	8	(0, 2, 1)	3
6	(20, 12, 18)	(2, 1)	(0, 3, 2)	0	8	(0, 2, 1)	3	(20, 4, 16, 13)	(3, 3, 3)	(0, 4, 2, 1)	0	16	(0, 2, 1)	3
7	(20, 16, 13)	(3, 3)	(0, 2, 1)	0	9	(0, 2, 1)	3							

Table 3 Problems, authors and solution methods for open vehicle routing problems

Problem	Authors	Algorithm/method
OVRP	[15]	Tabu search algorithm (TSA)
	[16]	Gravitational emulation local search algorithm (GELS)
OVRPTW	[4]	Greedy look-ahead
	[17]	VNS
	[18]	VNS with reinforcement learning (VNS-RLS)
MDOVRP	[19]	Multiple neighborhood search hybridized with a tabu search (MNS-TS)
	[20]	Hybrid adaptive large neighbourhood search algorithm (HALNS)
	[3]	Memory-based iterated local search algorithm (MBILSA)
	[1]	Mathematical programming
MDOVRPTW (variants)	[21]	Hybrid genetic algorithm with adaptive local search
	[2]	Gravitational emulation local search algorithm (GELS)

3 Literature review

The open vehicle routing problem (OVRP) was first addressed by Schrage [14]. Since then, much research has emerged into a variety of OVRPs. Table 3 reports some studies in open VRP. This table lists the problem, the author(s), and the algorithms or methods proposed in each study.

Brandão [15] developed a tabu search algorithm (TSA) to solve the OVRP. Hosseinabadi et al. [16] proposed the gravitational emulation local search algorithm (GELS) to solve the OVRP. In both studies [15, 16], the objective was to reduce the traveling time, distance, and the number of vehicles used.

The OVRP with Time Windows (OVRPTW) was addressed by Repoussis et al. [4]. The authors developed a heuristic to solve the OVRPTW by utilizing a greedy look-ahead solution framework for customer selection and route insertion. In Redi et al. [17], an improved variable neighborhood search (VNS) algorithm is proposed to solve the problem. The VNS features a route construction mechanism to ensure that customers with earlier time windows are served first. The proposed VNS was tested on 12 R1 and 8 RC1 datasets of classical VRPTW instances [22]. In recent work, Chen et al. [18] solved a real-life container transportation problem. They developed a variable neighborhood search algorithm with Reinforcement Learning (VNS-RLS). The algorithm was applied in 15 instances from the real-life Ningbo Port dataset.

The MDOVRP was introduced by Tarantilis et al. [23]. Soto et al. [19] proposed the multiple neighborhood search algorithm hybridized with a Tabu Search (MNS-TS) for solving this problem. The authors introduced some ejection chain techniques to reduce the size of the neighborhoods proposed. In Lahyani et al. [20], a hybrid algorithm based on a hybrid adaptive large neighborhood search (HALNS) heuristic was developed. The method combines the power of an ALNS algorithm to perform a diverse and thorough search of the solution space with local search improvement procedures to intensify the search for good and promising solutions. Brandão [3] presented the memory-based iterated local search algorithm (MBILSA). The moves performed during the local search are recalled, and this historical search information is then used to dene the moves executed inside the perturbation procedures. Lalla-Ruiz and Mes [1] proposed a two-index-based mathematical formulation for the MDOVRP. They also assessed the contribution of alternative constraints for handling flow conservation and sub-tours.

A variant of the MDOVRPTW under shared depot resources was addressed by Li et al. [21] as the combination of MDVRPTW and OVRP. In this problem, the vehicles may end at the depot nearest to the last customer. The problem was solved with a hybrid genetic algorithm with an adaptive local search. In Shen et al. [2], the authors solved the MDOVRPTW, which considers low-carbon trading policies. The low-carbon MDOVRPTW model was constructed with minimum total costs, which include the driver's salary, penalty costs, fuel costs, and carbon emissions trading costs. The solution was obtained through the particle swarm optimization (PSO) and tabu search (TS) algorithms.

In terms of techniques for the solution of routing-oriented problems, there are many in the literature; however, there is a large number of recent studies applying variable neighborhood search-based methods, as in studies of Smiti et al. [24], Bezerra et al. [25], Derbel et al. [26], Sanchez-Oro et al. [27], Ren et al. [28], Karakostas et al. [29], Barrero et al. [30], and Karakostas and Sifaleras [31]. The good performance of these methods in solving routing problems justifies our proposal to deal with the problems under study.

It is important to highlight that to the best of our knowledge, no research addresses the open VRP with multi-depots and time windows, with the primary objective to reduce the number of vehicles, as the proposed MDOVRPTW*.

4 Mathematical formulations

The mathematical formulations for MDOVRPTW and MDOVRPTW* are based on those presented in Li et al. [21], Brandão [3], and Repoussis et al. [4]. Table 1 shows the used notations in these formulations. Initially, the following decision variables are defined:

$$x_{kij} = \begin{cases} 1, & \text{if vehicle } k \text{ visits consumer } j \text{ after consumer } i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$y_k = \begin{cases} 1, & \text{if vehicle } k \text{ is used;} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

$$z_{jk} = \begin{cases} 1, & \text{if customer } j \text{ is the last customer to be visited by vehicle } k; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Considering that $c_{id} = 0$, $\forall\ i \in \mathcal{C}$, $d \in \mathcal{D}$, i.e., the return distance of a vehicle to the depot is null, the formulation for MDOVRPTW is given by:

$$(MDOVRPTW)\ \min \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} x_{kij}\ \ (4) \tag{4}$$

subject to:

$$\sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{V}} x_{kji} = 1, \quad \forall i \in \mathcal{C}, \tag{5}$$

$$\sum_{j \in \mathcal{V}} x_{kji} = \sum_{j \in \mathcal{V}} x_{kij} + z_{ik}, \quad \forall i \in \mathcal{C}, \forall k \in \mathcal{K} \tag{6}$$

$$\sum_{j \in \mathcal{C}} x_{kdj} = \sum_{j \in \mathcal{C}} z_{jk}, \quad \forall k \in \mathcal{K}, \forall d \in \mathcal{D} \tag{7}$$

$$\sum_{d \in \mathcal{D}} \sum_{j \in \mathcal{C}} x_{kdj} \leq 1, \quad \forall k \in \mathcal{K}, \tag{8}$$

$$\sum_{d \in \mathcal{D}} \sum_{i \in \mathcal{C}} x_{kid} = 0, \quad \forall k \in \mathcal{K}, \tag{9}$$

$$\sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{L}} x_{kij} \leq \mid \mathcal{L} \mid -1, \quad \forall \mathcal{L} \subseteq \mathcal{C}, k \in \mathcal{K}, \tag{10}$$

$$\sum_{i \in \mathcal{D}} \sum_{j \in \mathcal{D}} x_{kij} = 0, \quad \forall k \in \mathcal{K}, \tag{11}$$

$$x_{kij}(TS_{ki} + h_i + c_{ij} - TS_{kj}) \leq 0, \quad \forall k \in \mathcal{K}, \forall i, j \in \mathcal{V}, \tag{12}$$

$$e_i \leq TS_{ki} \leq l_i, \quad \forall i \in \mathcal{C}, \forall k \in \mathcal{K}, \tag{13}$$

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{C}} q_j x_{kij} \leq VC, \quad \forall k \in \mathcal{K}, \tag{14}$$

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} (c_{ij} + h_j) x_{kij} \leq MT, \quad \forall k \in \mathcal{K}, \tag{15}$$

$$x_{kij}, z_{ik} \in \{0, 1\}, \quad \forall\, i,j \in \mathcal{V}, \forall k \in \mathcal{K}. \tag{16}$$

In turn, the mathematical formulation of MDOVRPTW* is given by:

$$(MDOVRPTW^{*}) \quad \min\, \beta \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} c_{ij}x_{kij} + \alpha \sum_{k\in\mathcal{K}} y_k, \tag{17}$$

subject to:

$$\text{Constraints (4)–(14)} \tag{18}$$

$$y_k \geq \frac{\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} x_{kij}}{\mid N \mid^2}, \quad \forall k \in \mathcal{K}, \tag{19}$$

$$y_k \in \{0, 1\}, \quad \forall k \in \mathcal{K}. \tag{20}$$

Constraints (5) ensure that each customer i is visited by exactly one vehicle k. Constraints (6) guarantee that if a vehicle k arrives at customer i, this vehicle must leave this customer or, then, this customer will be the last one on the route. Constraints (7) assure that if a vehicle starts the route at a given depot, exactly one customer will be the last customer on the route. Constraints (8) guarantee that a vehicle will leave a depot, while constraints (9) assure that it will not return to it or another depot. These constraints define the routing condition as open. Constraints (10) assure that the graph is connected, as well as the sub-tours elimination. Constraints (11) establish that the vehicle cannot travel directly from depot i to depot j. Constraints (12) present the order of visits to the nodes, once if the vehicle k travels directly from the node i to the node j, then the moment of arrival TS_{kj} at the node j must be equal to $(TS_{ki} + h_i + c_{ij})$. Constraints (13) guarantee the occurrence of the service on customer i within the time window $[e_i, l_i]$. Constraints (14) ensure that the load will not exceed the capacity of the vehicle k. Constraints (15) define that the total duration of the route is at most equal to MT. Constraints (19) show which vehicles will be used, and, finally, Constraints (16) and (20) define the binary domain of the decision variables x, z, and y.

Expression (4) is the objective function of MDOVRPTW. This function represents the total covered distance, once c_{ij} is the distance between the customers i and j, and must be minimized. It is the focus of this problem. On the other hand, Expression (17) shows the objective function of MDOVRPTW*, also to be minimized. This objective function is given by the weighted sum between the cost associated with the distance between customers and the total number of used vehicles. The weighting factors α and β show the relative importance of each of the two objectives to achieve the desired solution. The values that these weighting factors take for each problem are described in Sect. 5.2.

5 Smart general variable search algorithm

The general variable neighborhood search (GVNS) [7, 32] is a metaheuristic that performs perturbation and local search procedures based on systematic neighborhood changes until reaching a predefined stopping criterion. Perturbations are random moves gradually applied in a solution s to guide the search toward other basins of attraction. Each perturbed solution s' is refined through the Basic Variable Neighborhood Descent (BVND) [7] local search method, guiding the search to a local minimum. The method returns the best solution found after the repeated application of these two procedures.

In this paper, we propose an algorithm named SGVNS, an acronym for Smart GVNS. The SGVNS algorithm uses the smart version of the GVNS method developed in [6]. Reinsma et al. [33] introduced the adjective "smart" to refer to how to increase the level of perturbation in an iterated local search (ILS) algorithm. Traditionally, there is an increase in the perturbation level in a classic ILS algorithm whenever there is no improvement in the current solution. However, this way of changing the perturbation level can lead to a loss in the quality of the final solution due to the hasty way of leaving the current search region. As perturbation occurs randomly, other choices of perturbed solutions within the same current search region can lead to better solutions through local search. Based on this observation, the decision to increase the perturbation level in the smart strategy occurs only after a certain number of local search applications without improvement in the quality of the current solution. Thus, this strategy allows a better investigation to be carried out in a specific region of the solution space, enabling a more precise intensification during the algorithm's execution. On the other hand, while Rego and Souza [6] apply the BVND as local search method, we use the randomized basic variable neighborhood descent (RBVND) method [8, 34].

To describe the proposed algorithm, we use throughout this section the notations described in Table 4 in addition to the notation introduced in Table 1.

This section is organized as follows. In Sect. 5.1, we show how to represent a solution. Section 5.2 describes the function used for evaluating the solution of MDVRPTW and MDOVRPTW*. Section 5.3 presents the procedure for generating an initial solution to the problem. Section 5.4 shows the neighborhood structures used to explore the problem's solution space. Section 5.5 details the perturbation procedure, and Sect. 5.6 describes the local search method. Finally, Sect. 5.7 presents the proposed SGVNS algorithm.

5.1 Solution representation

We represent a solution of the two problems through a set of lists $\mathcal{R} = \left\{ r_1^{N+1}, r_2^{N+1}, \ldots, r_m^{N+1}, r_1^{N+2}, r_2^{N+2}, \ldots, r_n^{N+2}, r_1^{N+|\mathcal{D}|}, r_2^{N+|\mathcal{D}|}, \ldots, r_t^{N+|\mathcal{D}|} \right\}$. In this set, r^{N+1}, r^{N+2}, and $r^{N+|\mathcal{D}|}$ refer to the set of routes of each depot (from $N+1$ to $N+ \mid \mathcal{D} \rvert$). In turn, m, n, $t \geq 1$ refer to the number of routes in each depot. For the example of Fig. 1a, $r_1^{19} = \{19,\ 1,\ 2\}$, $r_2^{19} = \{19,\ 5,\ 6,\ 7\}$, $r_3^{19} = \{19,\ 3,\ 9,\ 11\}$ represent the routes of

Table 4 Notations used in the description of the SGVNS algorithm

Acronym	Description
L	Total distance traveled
$F(s)$	Cost of solution s
c_{di}	Travel cost between a customer i and a depot d
r_d	List with routes of the depot d
$rd1d2$	Ratio of customer proximity to the two closest depots
s	Solution of the MDOVRPTW*
v	Index representing one of the neighborhood structures
$\bar{q}$	Average of demands
$\bar{s}$	Average of customers' service times
$\phi(k)$	Cost for violating the constraints of the vehicle k
φ	Total amount of infeasibility of the solution due to non-compliance with vehicle constraints
ω^Q	Penalty factor for overload vehicle
ω^T	Penalty factor for overduration
ω^{TW}	Penalty factor for delay
ψ	Correction factor for single thread rating between used processors
$\mathcal{A}'$	Set of pairs (i, d) of customers i assigned to the closest depot d
$\mathcal{B}'$	Set of pairs (i, d) of customers i assigned to the second closest depot d
$\mathcal{HD}$	Set of pairs (i, d) of customers i assigned to the depot d
$\mathcal{HR}$	Set of pairs $(i, rd1d2)$
$\mathcal{PO}$	Set of perturbation operators
$\mathcal{LSO}$	Set of local search operators
$\mathcal{R}$	Set of lists that represent the routes of the depots
$\mathcal{U}$	Set of vehicles used

depot 19, and $r_1^{20} = \{20, 4, 8, 10\}$, $r_2^{20} = \{20, 14, 17, 15\}$, $r_3^{20} = \{20, 12, 18\}$, $r_4^{20} = \{20, 16, 13\}$ the routes for depot 20.

5.2 Evaluation functions

As mentioned before, MDOVRPTW and MDOVRPTW* have different goals. In the case of MDOVRPTW, the objective is to reduce the total traveled distance using the total number of vehicles available. On the other hand, for MDOVRPTW*, the primary objective is to reduce the vehicle fleet, i.e., the number of used vehicles, and the secondary one is to reduce the total traveled distance. The evaluation functions of each problem are presented and discussed below. These evaluation functions add, to the objective functions of the mathematical formulations described in Sect. 4, a parcel of infeasibility to deal with the generation of infeasible solutions by the proposed algorithm during the exploration of the solution space of the problems.

5.2.1 Evaluation function to fleet reduction

As in Ombuki et al. [35], which addresses VRPTW, we evaluate a solution s for MDOVRPTW* by a evaluation function $F(s)$ that represents a weighted sum of three objective functions to be minimized:

$$F(s) = \beta L + \alpha \mid \mathcal{U} \mid + \varphi \tag{21}$$

The first objective function (L) seeks to reduce the total traveled distance by all used vehicles; the second objective function ($\mid \mathcal{U} \mid \leq \mid \mathcal{K} \mid$) tries to reduce the number of used vehicles. The last objective function (φ) seeks to minimize the violations that may occur due to applying the moves to explore the problem's solution space. These violations can occur due to the vehicle overload, customer service delays, and extrapolation of the route's maximum duration.

The three objective functions of Eq. (21) are weighted by the values β, α, and φ, respectively, that reflect the relative importance of each objective function for reaching the solution of MDVRPTW*. As proposed in Ombuki et al. [35], we set $\beta = 0.001$. The parameter α is calculated by:

$$\alpha = \frac{1}{N} \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}} c_{ij} \tag{22}$$

According to Eq. (22), the parameter α represents the average distance between each pair of customers. Finally, the weighting factor φ is given by:

$$\varphi = \sum_{k \in \mathcal{U}} \phi(k) \tag{23}$$

where $\phi(k)$ represents the penalization by violating the constraints imposed on the route of each vehicle k, being calculated as:

$$\phi(k) = \omega^{Q} \max \left\{ 0, Q_k - VC \right\} + \omega^{T} \max \left\{ 0, T_k - MT \right\} + \omega^{TW} TW_k \tag{24}$$

In these expression, the parameter ω^{Q} penalizes the vehicle overload, the parameter ω^{T} penalizes the time exceeded in the route duration of vehicle k, and the parameter ω^{TW} penalizes delays in the delivery of products by vehicle k. The delay TW_k is the sum of the times that exceed the final time window of each customer belonging to the vehicle route k. In turn, the weighting values ω^{Q}, ω^{T} and ω^{TW} are calculated from the parameter α, the average demands $\bar{q}$, and the average service times $\bar{h}$ for consumers, as defined below:

$$\omega^{Q} = \frac{\alpha}{\bar{q}} \tag{25}$$

$$\omega^{T} = \omega^{TW} = \frac{\alpha}{\bar{h}} \tag{26}$$

The use of penalties for treating constraint violations is a strategy commonly used in the optimization literature due to the high degree of difficulty in exploring the solution space of problems like VRP and variants using only feasible solutions. The algorithms proposed by Polacek et al. [36] and Vidal et al. [37], for example, use this strategy.

5.2.2 Evaluation function to distance reduction

This section presents the evaluation function adopted for solving MDOVRPTW in order to reduce the total traveled distance by the used vehicles. This evaluation function for a solution s is given by:

$$F'(s) = \beta L + \alpha \Big| |\mathcal{K}| - |\mathcal{U}| \Big| + \varphi \tag{27}$$

As this evaluation function seeks to minimize the total traveled distance, the weighting factor β is set to $\beta = 1$. The second parcel of this evaluation function aims to stimulate the use of the whole vehicle fleet and, hence, the parameter α is calculated according to Eq. (22). The third parcel φ of $F(s)$ remains as described in Eq. (23), since the proposed algorithm also generates infeasible solutions.

5.3 Construction of the initial solution

The procedure for building the initial solution was inspired by the algorithm proposed by [38]. This algorithm first groups the customers for each depot and then generates the routes. This technique is performed here by Algorithms 1 and 2, respectively.

5.3.1 Procedure for generating clusters

This section describes Algorithm 1, which performs the clustering procedure. While customers are not associated with any route, the method finds a depot that can accept them. This is shown in Algorithm 1 in lines 3–47. In lines 8–18, we determine the two depots closest to each customer $i \in \mathcal{C}'$. These depots were identified as $d1$ and $d2$ (lines 11 and 13). The distances between customer i and depots $d1$ and $d2$ were named $c_{d1,i}$ and $c_{d2,i}$, respectively. In line 15, the proximity ratio between each customer i and its depots $d1$ and $d2$ are calculated. The set of pairs $\{i, c_{d1,i}/c_{d2,i}\}$, described in line 16, constitute the set $\mathcal{HR}$. This set is sorted in ascending order from these ratios in line 19. Then, the customers are assigned to one of these depots. This procedure starts at line 22 and finishes in line 36. The distribution between depots occurs in a balanced way (lines 28–34). Therefore, in the order given by $\mathcal{HR}$, each pair formed by the customer $i \in \mathcal{C}'$, and its respective depot $d1$ (or $d2$), is assigned to the sets $\mathcal{HD}$ and $\mathcal{C}''$. In the new composition of set $\mathcal{C}'$, all assigned customers are removed (line 37). In turn, the depots that have reached their capacity limit are also removed (lines 39–46). The algorithm returns the set $\mathcal{HD}$ with pairs of customers and depots.

Algorithm 1 CreateClusters($\mathcal{C}$, $\mathcal{D}$)

```
 1: $\mathcal{C}' \leftarrow \mathcal{C}$; $\mathcal{D}' \leftarrow \mathcal{D}$;    ▷ Copies of the sets: customers and depots
 2: $\mathcal{HD} \leftarrow \varnothing$;    ▷ Set of pairs $(i,\ d)$ of customers $i$ assigned to depots $d$
 3: while $\mid \mathcal{C}' \mid > 0$ do
 4:     $\mathcal{A}' \leftarrow \varnothing$;    ▷ Set of pairs $(i, d)$, where $d$ is the closest depot to customer $i$
 5:     $\mathcal{B}' \leftarrow \varnothing$; ▷ Set of pairs $(i, d)$, where $d$ is the second closest depot to customer $i$
 6:     $\mathcal{HR} \leftarrow \varnothing$; ▷ Set of pairs $(i,\ rd1d2)$, where $rd1d2$ is the ratio of proximity of
    customer $i$ to the two closest depots
 7:     $j \leftarrow 1$;
 8:     while $j \leq \mid \mathcal{C}' \mid$ do
 9:         Let $i$ be the customer of the $j$-th position of $\mathcal{C}' \mid i \in \mathcal{C}'$;
10:         $c_{di} \leftarrow$ distance between the customer $i$ and each depot $d \in \mathcal{D}'$;
11:         $d1 = \arg\min_{d}\ \{c_{di}\},\ \forall d \in \mathcal{D}'$;
12:         $\mathcal{A}' \leftarrow \mathcal{A}' \cup \{(i, d1)\}$;
13:         $d2 = \arg\min_{d}\ \{c_{di}\},\ \forall d \in \mathcal{D}',\quad d1 \neq d2$;
14:         $\mathcal{B}' \leftarrow \mathcal{B}' \cup \{(i, d2)\}$;
15:         $rd1d2 = c_{d1,i}/c_{d2,i}$;
16:         $\mathcal{HR} \leftarrow \mathcal{HR} \cup \{(i, rd1d2)\}$;
17:         $j \leftarrow j + 1$;
18:     end while
19:     Sort $\mathcal{HR}$ in ascending order, according to the value of $rd1d2$;
20:     $j \leftarrow 1$;
21:     $\mathcal{C}'' \leftarrow \varnothing$;    ▷ Set of customers assigned to some depot
22:     while $j \leq \mid \mathcal{C}' \mid$ do
23:         Let $i$ be the customer of the $j$-th position of $\mathcal{HR} \mid i \in \mathcal{C}'$;
24:         Let $d1$ be the depot associated with the client $i \mid (i, d1) \in \mathcal{A}'$;
25:         Let $d2$ be the depot associated with the customer $i \mid (i, d2) \in \mathcal{B}'$;
26:         Let $td$ be the total of customers in $\mathcal{HD}$ assigned to depot $d1$;
27:         Let $td'$ be the total of customers in $\mathcal{HD}$ assigned to depot $d2$;
28:         if $td < \lceil \mid \mathcal{C} \mid / \mid \mathcal{D} \mid \rceil$ then
29:             $\mathcal{HD} \leftarrow \mathcal{HD} \cup \{(i,\ d1)\}$;
30:             $\mathcal{C}'' \leftarrow \mathcal{C}'' \cup \{i\}$
31:         else if $td' < \lceil \mid \mathcal{C} \mid / \mid \mathcal{D} \mid \rceil$ then
32:             $\mathcal{HD} \leftarrow \mathcal{HD} \cup \{(i,\ d2)\}$;
33:             $\mathcal{C}'' \leftarrow \mathcal{C}'' \cup \{i\}$
34:         end if
35:         $j \leftarrow j + 1$;
36:     end while
37:     $\mathcal{C}' \leftarrow \mathcal{C}' \setminus \mathcal{C}''$;
38:     $j \leftarrow 1$;
39:     while $j \leq \mid \mathcal{D}' \mid$ do
40:         Let $d$ be the depot of the $j$-th position of $\mathcal{D}' \mid d \in \mathcal{D}$;
41:         Let $td$ be the total of customers in $\mathcal{HD}$ assigned to depot $d$
42:         if $td \geq \lceil \mid \mathcal{C} \mid / \mid \mathcal{D} \mid \rceil$ then
43:             $\mathcal{D}' \leftarrow \mathcal{D}' \setminus \{d\}$;
44:         end if
45:         $j \leftarrow j + 1$;
46:     end while
47: end while
48: return $\mathcal{HD}$;
```

5.3.2 Algorithm for generating routes

Algorithm 2 shows the procedure to construct the routes. The routes are constructed respecting the vehicles' capacity, the maximum route time, and the customers' time windows. These routes are constructed in a partially greedy way. The customer i and the depot d, with $\{i, d\} \in \mathcal{HD}$, are chosen from a previously fixed value $\lambda \in [0, 1]$. If a real random value is less than or equal to λ (line 4), then a customer $j \in \mathcal{HD}$ is chosen (line 5). Otherwise, the jth element is the first of the set $\mathcal{HD}$ (line 3). From empirical tests, the value of λ was fixed in 0.6. In lines 7–8, the customer and depot are defined. The routes are represented in a subset $r_d \subset \mathcal{R}$ that receives the customer i, as shown in line 10. In line 11, the pair $\{i, d\}$ is dropped from $\mathcal{HD}$. All customers must be served. This imposition can occur that the number of vehicles used can be greater than $| \mathcal{K} |$. The algorithm is finalized in line 13, returning the set $\mathcal{R}$ of routes.

Algorithm 2 : CreateRoutes($\mathcal{HD}$, $\mathcal{C}$, $\mathcal{D}$)

```
1:  $\mathcal{R} \leftarrow \varnothing$;                                   ▷ Solution set of the MDVRPTW*
2:  while $| \mathcal{HD} | > 0$ do
3:      $j \leftarrow 1$;
4:      if $rand(0,1) \leq \lambda$ then
5:          $j \leftarrow\ rand(| \mathcal{HD} |)$;
6:      end if
7:      Let $i$ be the customer of the $j$th position of $\mathcal{HD} \mid i \in \mathcal{C}$;
8:      Let $d$ be the depot associated with the customer $i$ of the $j$th position of
    $\mathcal{HD} \mid d \in \mathcal{D}$;
9:      Let $r_d$ be the list associated with the depot $d \mid r_d \subset \mathcal{R}$;
10:     $r_d \leftarrow r_d \cup \{i\}$;                        ▷ Compose the routes of the depot $d$ ;
11:     $\mathcal{HD} \leftarrow \mathcal{HD} \setminus \{(i, d)\}$;
12: end while
13: return $\mathcal{R}$;
```

5.4 Neighborhoods

To explore the MDOVRPTW and MDOVRPTW* solution spaces, we use neighborhood operators that apply moves in the same depot on different routes (inter-routes), in the same route (intra-route), or between routes from different depots (inter-depots). These neighborhood operators are described in the following.

5.4.1 Perturbation operators

Three perturbation mechanisms $\mathcal{PO} = \{\mathcal{PO}_1, \mathcal{PO}_2, \mathcal{PO}_3\}$ were implemented:

(1) $\mathcal{PO}_1$ (*Eliminates route*) it consists of eliminating a route $r^d \in \mathcal{R}$ from a depot d and transferring its customers to other routes from the same depot or to other depot d'. All choices are made at random. All insertions of customers into routes need to be feasible.

(2) $\mathcal{PO}_2$ (*ShiftDepot*) a route of the set r^{d_1} of routes from the depot d_1 is transferred to another depot d_2, where $d_2 \neq d_1$. The depot d_1 must have the greatest number of routes (at least two routes) and d_2 the smallest one. If all depots have the same number of routes, d_1 and d_2 are randomly chosen.

(3) $\mathcal{PO}_3$ (*SwapDepot*) it consists of swapping a route of the set r^{d_1} of routes from the depot d_1 with a route of the set r^{d_2} of routes of the depot d_2.

5.4.2 Local search operators

We used Swap, Reinsertion, and Permutation operators, widely applied in the literature on VRP and variants [8, 25, 37, 39–44]. They are described below.

Intra-route operators:

(1) $\mathcal{LSO}_1$ (*Swap*) it consists of swapping two customers in the same route.

(2) $\mathcal{LSO}_2$ (*Reinsertion*) a customer is removed and reinserted in another position in the same route.

(3) $\mathcal{LSO}_3$ (*Or-opt2*) two consecutive customers are removed and reinserted in another position in the same route.

(4) $\mathcal{LSO}_4$ (*2-Opt*) two non-adjacent edges are deleted and two others are added to generate a new route.

(5) $\mathcal{LSO}_5$ (*3-Opt*) three edges are excluded and all possibilities of exchange between them are tested to generate new routes.

Inter-routes operators:

(1) $\mathcal{LSO}_6$ (*Swap(1,1)*) it consists of swapping a customer v_j from one route r_k with a customer v_t from another route r_l belonging to the same depot.

(2) $\mathcal{LSO}_7$ (*Shift(1,0)*) it consists of transferring a customer v_j from a route r_k to another route r_l belonging to the same depot.

Inter-depots operators:

(1) $\mathcal{LSO}_8$ (*Swap(1,1)-InterDepot*) it consists of swapping a customer v_j from one route r_k with a customer v_t from another route r_l belonging to another depot.

(2) $\mathcal{LSO}_9$ (*Shift(1,0)-InterDepot*) it consists of transferring a customer v_j from a route r_k to another route r_l belonging to another depot.

The set of all local search operators is $\mathcal{LSO} = \{\mathcal{LSO}_1, \ldots, \mathcal{LSO}_9\}$.

5.5 Perturbation procedure

Algorithm 3 presents the perturbation procedure. In this algorithm, $\mathcal{PO}$ is the set of perturbation operators described in Sect. 5.5, $\mathcal{PO}_v$ is the vth operator that generates the perturbation solution, and p is the number of times that this operator is applied to the solution s. In line 6, the procedure returns the perturbed solution s_p.

Algorithm 3 : Perturbation (s, $\mathcal{PO}$, v, p)

```
$s_p \leftarrow s$;
for ($i = 1$;   $i \leq p$;   $i++$) do
    Generate a random neighbor $s' \in \mathcal{PO}_v(s_p)$;
    $s_p \leftarrow s'$;
end for
return $s_p$;
```

5.6 Local search

We used the randomized basic variable neighborhood descent (RBVND) algorithm [34] as the local search method. RBVND is a variation of the basic variable neighborhood descent (BVND) method [32]. Instead of using a predefined order of neighborhoods to explore the solution space, it uses a random order at each call of the local search method. More specifically, whenever it is not possible to improve the current solution in a specific neighborhood, RBVND randomly selects another neighborhood to continue the search in the problem's solution space. At the end of the method, it returns a local optimum concerning all explored neighborhoods.

Algorithm 4 : RBVND(s, $\mathcal{LSO}$)

```
1:  $\mathcal{LSO}' \leftarrow rand(\mathcal{LSO})$;
2:  $v \leftarrow 1$;
3:  while $v \leq | \mathcal{LSO}' |$ do
4:      $s' \leftarrow \arg \min_{z \in \mathcal{LSO}'_v(s)} f(z)$;        ▷ Best neighbor of $\mathcal{LSO}'_v(s)$
5:      if $f(s') < f(s)$ then
6:          $s \leftarrow s'$;
7:          $v \leftarrow 1$;
8:      else
9:          $v \leftarrow v + 1$;
10:     end if
11: end while
12: return $s$;
```

Algorithm 4 presents the pseudocode of RBVND. In this algorithm, the set $\mathcal{LSO}'$ defines the neighborhood structures that will be used, which are randomly ordered from the set $\mathcal{LSO}$ (line 1). At each iteration (lines 3–11), the neighborhood $\mathcal{LSO}'_v \in \mathcal{LSO}'$ is applied to the s solution, generating the neighbor s'. In case of improvement of the current solution (line 5), it is updated, and the index v returns to the first neighborhood of $\mathcal{LSO}'$; otherwise, the search moves to the next neighborhood. The algorithm ends when it reaches the last neighborhood structure, and there is no improvement in the current solution. The refined solution is returned in line 12.

5.7 Proposed algorithm

Algorithm 5 shows the pseudo-code of the SGVNS algorithm proposed to solve MDOVRPTW and MDOVRPTW*. Lines 1 and 2 show that an initial solution is

Table 5 Characteristics of the instances used for tuning SGVNS with IRACE

Instance	pr03	pr04	pr06	pr12	pr19	pr16
$\mid \mathcal{C} \mid$	144	192	288	96	216	288
$\mid \mathcal{K}_d \mid$	4	5	7	2	3	6
$\mid \mathcal{D} \mid$	4	4	4	4	6	4
ρ	2304	3840	8064	768	3888	6912

built using the procedures described in Algorithms 1 and 2, respectively. After the solution is created, the initial perturbation level, the index for the first neighborhood, the iteration counter without improvement, and the iteration counter for applying the local search procedure are set in 3, 4, 5, and 6, respectively. The main loop for SGVNS is between lines 7 and 27. The algorithm ends when the number of iterations without improvement reaches its maximum value or the run time meets its maximum duration. At each iteration, the solution s is perturbed according to Algorithm 3, generating a perturbed solution s' through perturbation operations described in Sect. 5.5. In line 9, the local search procedure is applied in s' according to Algorithm 9 described in Sect. 5.6, returning an improved solution s''. If this solution s'' is better than s, then we update the solution s, return to the first neighborhood and to the first perturbation level, and reset the iteration counter without improvement in lines 11–14, respectively. Otherwise, the perturbation level and the iteration counter are increased (lines 16–17). If the perturbation level meets its maximum level, i.e., *maxLevel*, the neighborhood is changed for the next. When it occurs, the perturbation level is changed to its minimum value, i.e., $p = 1$. If the last perturbation neighborhood is reached, it is necessary to return to the first perturbation neighborhood (line 24) and consequently to the first perturbation level (line 25). In line 28, the SGVNS algorithm returns the best solution s found during the search.

Table 6 Tested parameter values and results returned by IRACE

Description	Parameter	Range	Returned value
Maximum number of iterations without improvement	*iterMax*	{50, 100, 200, 300}	{200, 100}
Maximum level of perturbation	*maxLevel*	{3, 4, 5, 6, 7}	5

Algorithm 5 : SGVNS ($\mathcal{C}$, $\mathcal{D}$, $\mathcal{LSO}$, $\mathcal{PO}$, $iterMax$, $maxTime$, $maxLevel$)

```
$\mathcal{HD} \leftarrow$ CreateClusters($\mathcal{C}$, $\mathcal{D}$);
$s \leftarrow$ CreateRoutes($\mathcal{HD}$, $\mathcal{C}$, $\mathcal{D}$);
$p \leftarrow 1$;                                   ▷ Initial perturbation level
$v \leftarrow 1$;                                   ▷ Index for the first neighborhood
$iter \leftarrow 0$;                                ▷ Iteration counter without improvement
$itLS \leftarrow 0$;                                ▷ Counter for the local search
while $iter < iterMax$ or $time < maxTime$ do
    $s' \leftarrow$ Perturbation($s$, $\mathcal{PO}$, $v$, $p$);
    $s'' \leftarrow$ RBVND($s'$, $\mathcal{LSO}$);
    if $f(s'') < f(s)$ then
        $s \leftarrow s''$;
        $v \leftarrow 1$;                           ▷ Return to the first neighborhood
        $p \leftarrow 1$;                           ▷ Return to the first perturbation level
        $iter \leftarrow 0$;                        ▷ Reset the iteration counter without improvement
    else
        $p \leftarrow p + 1$;                       ▷ Increase the perturbation level
        $iter \leftarrow iter + 1$;                 ▷ Increase the iteration counter without improvement;
    end if
    if $p > maxLevel$ then
        $v \leftarrow v + 1$;                       ▷ Move to the next neighborhood
        $p \leftarrow 1$;                           ▷ Return to the first perturbation level
    end if
    if $v > maxNPert$ then
        $v \leftarrow 1$;                           ▷ Return to the first neighborhood
        $p \leftarrow 1$;                           ▷ Return to the first perturbation level
    end if
end while
return $s$;
```

6 Computational results

The SGVNS algorithm was coded in C++ and executed on an Intel Xeon E5620 2.40 GHz × 16 machine with 112 GB of RAM under the Linux operating system 64 bits.

We do not find test instances for MDOVRPTW and MDOVRPTW* in the literature. Hence, we decide to use the MDVRPTW instances from Cordeau et al. [45] and Vidal et al. [37] as test instances for these open routing problems. We named Group I the set described in Cordeau et al. [45], and Group II the instances proposed in Vidal et al. [37]. The first set is available at Networking and Emerging Optimization—NEO,[1] and the second one at Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation—CIRRELT.[2] The distance in Group I and Group II is the Euclidean distance, and it is supposed that the vehicle's travel time between two nodes is equal to the Euclidean distance. The vehicle fleet is homogeneous in both groups.

The characteristics of instances of Group I and Group II are described in Tables 7 and 8, respectively. The first five columns of each table describe the characteristics of the instances, i.e., the number of customers, depots, vehicles per depot, and the total of available vehicles, respectively. Group I has 20 instances, such that the first ten instances have tight time windows, while the last ten ones have wide time windows. Group II contains 28 instances with a large number of customers and depots.

6.1 Parameter tuning

We used the iterated racing for automatic algorithm configuration (IRACE) [46] for tuning the SGVNS algorithm parameters. To calibrate the parameters *iterMax* and *maxLevel*, we selected 30% of the Group I instances. Table 5 shows the six chosen instances, and it is important to highlight that they contain different characteristics representative of the set of instances. We ordered them by the values of $\rho =\mid \mathcal{C} \mid \cdot \mid \mathcal{K}_d \mid \cdot \mid \mathcal{D} \mid$ and grouped them by tight and wide time windows, respectively.

Table 6 reports the range of the tested values and the returned values by IRACE. The parameter *iterMax* was set to 100; above this value, the algorithm required more computational time without significantly improving the results.

6.2 Group I results

Table 7 reports the best results found by SGVNS in instances of Group I for MDOVRPTW and MDOVRPTW* and a comparison of its results with those shown in [37] for MDVRPTW. In this table, the first five columns show the MDVRPTW characteristics. The column "MDVRPTW" shows the best result for each instance from the literature. Column "SGVNS" shows the results found for the MDOVRPTW and MDOVRPTW*. For each instance, the distance, number of vehicles used, and spent time are shown in columns "Distance", "$\mathcal{K}$", and "Time", respectively. The differences concerning the distances and vehicles for both problems are shown in column "Difference between MDOVRPTW* and MDOVRPTW". The line "Sum" of this table shows the total number of available vehicles and the total distance for the MDVRPTW; the total distance, the number of vehicles used, and spent times in all instances for the MDOVRPTW and MDOVRPTW*; the difference between the total distance and the total number of vehicles used for the MDOVRPTW*

[1] http://neo.lcc.uma.es/vrp/vrp-instances/.

[2] https://w1.cirrelt.ca/~vidalt/en/VRP-resources.html.

Table 7 Results of the SGVNS algorithm applied to Group I instances for MDOVRPTW and MDVRPTW*

Instance	$\mid \mathcal{C} \mid$	$\mid \mathcal{D} \mid$	$\mid \mathcal{K}_d \mid$	$\mid \mathcal{K} \mid$	MDVRPTW	MDOVRPTW				MDOVRPTW*				Difference between MDOVRPTW* and MDOVRPTW	
					Distance	Distance	$\mathcal{U}$	Time	Distance reduction (%)	Distance	$\mathcal{U}$	Time	Vehicle reduction (%)	Distance	Vehicle
pr01	48	4	2	8	1074.12	886.48	8	0.51	− 17.47	1105.09	6	0.20	− 25.00	218.61	− 2
pr02	96	4	3	12	1762.21	1388.35	12	2.75	− 21.22	1602.58	10	1.00	− 16.67	214.23	− 2
pr03	144	4	4	16	2373.65	2157.48	16	2.95	− 9.11	2736.44	12	1.82	− 25.00	578.96	− 4
pr04	192	4	5	20	2815.48	2522.99	20	13.28	− 10.39	2644.36	18	5.78	− 10.00	121.37	− 2
pr05	240	4	6	24	2964.65	2639.74	24	14.13	− 10.96	2809.95	22	19.17	− 8.33	170.21	− 2
pr06	288	4	7	28	3588.78	3401.57	28	27.36	− 5.22	3387.00	27	20.08	− 3.57	− 14.57	− 1
pr07	72	6	2	12	1418.22	1151.73	12	0.90	− 18.79	1340.38	9	0.46	− 25.00	188.65	− 3
pr08	144	6	3	18	2096.73	1852.31	18	4.85	− 11.66	2191.60	13	1.32	− 27.78	339.29	− 5
pr09	216	6	4	24	2712.56	2402.34	24	7.47	− 11.44	2935.67	19	3.83	− 20.83	533.33	− 5
pr10	288	6	5	30	3465.92	3072.93	30	32.82	− 11.34	3614.71	27	23.24	− 10.00	541.78	− 3
pr11	48	4	1	4	1005.73	890.86	4	0.50	− 11.42	884.18	4	0.40	–	− 6.68	–
pr12	96	4	2	8	1464.5	1172.55	8	2.79	− 19.94	1188.69	8	2.65	–	16.14	–
pr13	144	4	3	12	2001.83	1717.10	12	6.92	− 14.22	1949.21	11	5.94	− 8.33	232.11	− 1
pr14	192	4	4	16	2195.33	1932.09	16	14.63	− 11.99	2069.50	15	14.27	− 6.25	137.41	− 1
pr15	240	4	5	20	2433.15	2097.23	20	25.64	− 13.81	2137.34	20	27.31	–	40.11	–
pr16	288	4	6	24	2836.67	2640.57	24	48.94	− 6.91	2680.17	23	48.14	− 4.17	39.60	− 1
pr17	72	6	1	6	1236.24	1070.43	6	1.09	− 13.41	1070.72	6	60.00	–	0.29	–
pr18	144	6	2	12	1788.18	1521.80	12	10.84	− 14.90	1757.01	11	10.70	− 8.33	235.21	− 1
pr19	216	6	3	18	2261.08	1905.24	18	17.57	− 15.74	2099.35	16	22.99	− 11.11	194.11	− 2
pr20	288	6	4	24	2993.31	2693.34	24	53.20	− 10.02	2660.01	24	24.58	–	-33.33	–
Sum				336	44488.34	39117.13	336	289.14		42863.96	301	293.88		3746.83	− 35

Table 7 (continued)

Instance	$\mid \mathcal{C} \mid$	$\mid \mathcal{D} \mid$	$\mid \mathcal{K}_d \mid$	$\mid \mathcal{K} \mid$	MDVRPTW	MDOVRPTW				MDOVRPTW*				Difference between MDOVRPTW* and MDOVRPTW	
					Distance	Distance	$\mathcal{U}$	Time	Distance reduction (%)	Distance	$\mathcal{U}$	Time	Vehicle reduction (%)	Distance	Vehicle
Comparison (%) with MDVRPTW						– 12.07	–			– 3.65	– 10.42				
Comparison (%) with MDOVRPTW														– 9.58	– 10.42

Table 8 Results of the SGVNS algorithm applied to Group II instances for MDOVRPTW and MDVRPTW*

Instance	$\|\mathcal{C}\|$	$\|\mathcal{D}\|$	$\|\mathcal{K}_d\|$	$\|\mathcal{K}\|$	MDVRPTW	MDOVRPTW				MDOVRPTW*				Difference (%) of MDOVRPTW* and MDOVRPTW	
					Distance	Distance	Vehicle	Time	Distance reduction (%)	Distance	Vehicle	Time	Vehicle reduction (%)	Distance	Vehicle
pr11a	360	4	10	40	6720.71	5757.87	40	61.97	– 14.33	6117.39	32	32.63	– 20.00	359.52	– 8
pr12a	480	4	13	52	8179.80	7123.59	52	106.04	– 12.91	7500.05	41	68.04	– 21.15	376.46	– 11
pr13a	600	4	16	64	9667.20	8383.24	64	117.75	– 13.28	8506.39	51	161.58	– 20.31	123.15	– 13
pr14a	720	4	19	76	11124.01	9780.85	76	180.00	– 12.07	10809.87	59	180.02	– 22.37	1029.02	– 17
pr15a	840	4	22	88	13013.97	11544.52	88	180.02	– 11.29	12421.66	72	180.03	– 18.18	877.14	– 16
pr16a	960	4	26	104	14299.87	13221.55	104	180.01	– 7.54	13266.47	87	180.03	– 16.35	44.92	– 17
pr17a	360	6	7	42	6304.30	5560.62	42	36.80	– 11.80	6056.34	32	15.02	– 23.81	495.72	– 10
pr18a	520	6	10	60	8308.32	7346.70	60	92.69	– 11.57	7577.97	46	75.73	– 23.33	231.27	– 14
pr19a	700	6	13	78	10677.61	9328.91	78	180.01	– 12.63	9992.12	59	120.96	– 24.36	663.21	– 19
pr20a	880	6	16	96	11963.91	10750.93	96	180.04	– 10.14	11504.57	74	177.95	– 22.92	753.64	– 22
pr21a	420	12	4	48	6260.53	5798.82	48	50.17	– 7.37	6083.25	34	53.07	– 29.17	284.43	– 14
pr22a	600	12	6	72	7985.37	7511.24	72	180.02	– 5.94	8139.98	50	79.99	– 30.56	628.74	– 22
pr23a	780	12	8	96	9937.43	9427.49	96	172.97	– 5.13	10050.26	65	180.01	– 32.29	622.77	– 31
pr24a	960	12	10	120	11923.72	11484.88	119	180.04	– 3.68	12012.55	85	180.00	– 29.17	527.67	– 34
pr11b	360	4	8	32	4839.44	4179.59	32	59.69	– 13.63	4332.77	27	42.46	– 15.63	153.18	– 5
pr12b	480	4	11	44	6063.26	5311.61	44	106.45	– 12.40	6055.14	35	123.94	– 20.45	743.53	– 9
pr13b	600	4	14	56	7254.17	6457.07	56	177.24	– 10.99	6726.56	44	180.00	– 21.43	269.49	– 12
pr14b	720	4	17	68	8732.29	7649.92	68	180.00	– 12.40	7888.94	55	180.03	– 19.12	239.02	– 13
pr15b	840	4	20	80	10439.72	9048.86	80	180.01	– 13.32	9499.00	66	180.03	– 17.50	450.14	– 14
pr16b	960	4	23	92	11483.22	10427.21	91	180.02	– 9.20	10149.21	74	180.02	– 19.57	– 278.00	– 17
pr17b	360	6	6	36	4806.01	4210.63	36	57.08	– 12.39	4590.55	27	67.86	– 25.00	379.92	– 9

Table 8 (continued)

Instance	$\vert\mathcal{C}\vert$	$\vert\mathcal{D}\vert$	$\vert\mathcal{K}_d\vert$	$\vert\mathcal{K}\vert$	MDVRPTW	MDOVRPTW				MDOVRPTW*				Difference (%) of MDOVRPTW* and MDOVRPTW	
					Distance	Distance	Vehicle	Time	Distance reduction (%)	Distance	Vehicle	Time	Vehicle reduction (%)	Distance	Vehicle
pr18b	520	6	9	54	6526.72	5772.21	54	101.88	– 11.56	6086.92	39	119.46	– 27.78	314.71	– 15
pr19b	700	6	12	72	8227.25	7349.44	72	140.55	– 10.67	7247.77	53	180.05	– 26.39	– 101.67	– 19
pr20b	880	6	15	90	10325.80	8866.85	90	180.01	– 14.13	9011.14	69	180.02	– 23.33	144.29	– 21
pr21b	420	12	4	48	4866.57	4790.29	48	19.97	– 1.57	5046.17	30	44.36	– 37.50	255.88	– 18
pr22b	600	12	6	72	6488.50	6320.29	67	59.84	– 2.59	6173.67	44	121.05	– 38.89	– 146.62	– 23
pr23b	780	12	7	84	8523.41	7580.44	79	180.05	– 11.06	7796.45	59	180.05	– 29.76	216.01	– 20
pr24b	960	12	8	96	10890.08	9212.93	94	180.04	– 15.40	10106.99	78	180.05	– 18.75	894.06	– 16
Total				1960	245833.19	220198.55	1946	3701.36		230750.15	1487	3644.44		10551.60	– 459
Comparison (%) with MDVRPTW						– 10.43	– 0.71			– 6.14	– 24.13				
Comparison (%) with MDOVRPTW														4.79	– 23.59

and MDOVRPTW. For Group I, the SGVNS algorithm was executed 30 runs per instance, and the value of the *MaxTime* parameter was set to 60 min.

The results show a reduction of approximately 12.07% in the total distance for MDOVPTW for all evaluated instances when compared with MDVRPTW. In all instances, there was a reduction in the distance traveled. The MDOVRPTW* results showed that the number of vehicles used was reduced by 10.42% on average compared to the total number of available vehicles. Among the 336 vehicles available in Group I, only 301 were used by the SGVNS solution. Even though it is not the aim associated with MDOVRPTW*, the total distance of Group I decreased by 3.65% on average compared with the MDVRPTW solution.

6.3 Group II results

Table 8 shows the best results generated by the SGVNSALS algorithm for Group II. The structure of this table is similar to that adopted for Table 7. As in Sect. 6.2, the results obtained by SGVNS for MDOVRPTW and MDOVRPTW* are compared to those of MDVRPTW shown in Vidal et al. [37]. For Group II, the algorithm was executed 20 runs per instance, and the value of the *MaxTime* parameter was set to 180 min.

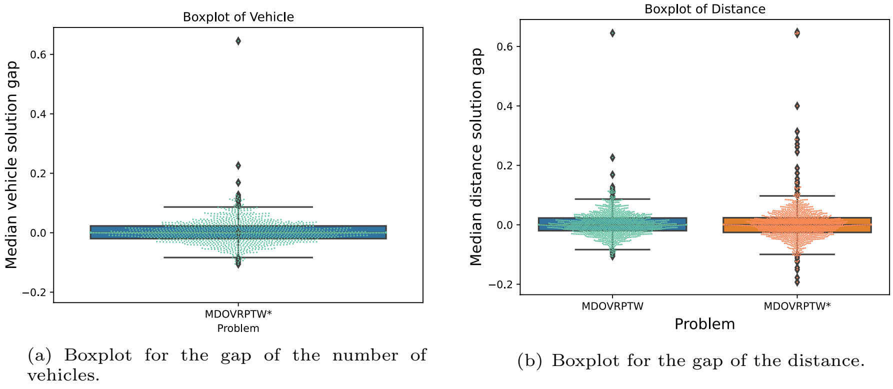
For all instances of Group II, there was also a reduction in the distance and number of used vehicles. For MDOVRPTW, there was a reduction of 10.43% in the total traveled distance compared with the results from MDVRPTW shown in Vidal et al. [37]. Concerning MDOVRPTW*, there was a reduction of 24.13% in the number of used vehicles and 6.14% in the total traveled distance compared with the MDVRPTW results. Additionally, the results from MDOVRPTW* regarding MDOVRPTW achieved a reduction of 23.59% in the number of used vehicles, alongside an increase of 4.79% in the total distance traveled. These results, reached in a challenging set of instances, justify the definition and study of this variant.

7 Analysis of the proposed algorithm

In this section, the results of the SGVNS algorithm are statistically investigated. In Fig. 2a and 2b, the ordinate axis represents the gaps of the value of the median solutions concerning distances and number of vehicles, respectively. The gaps were calculated through:

$$gap\_vehicle = \frac{number\ of\ vehicles}{median\ of\ vehicles} \tag{28}$$

$$gap\_distance = \frac{distance}{median\ of\ distances} \tag{29}$$



(a) Boxplot for the gap of the number of vehicles.

(b) Boxplot for the gap of the distance.

Fig. 2 Boxplots of the SGVNS algorithm results

The results achieved by SGVNS concerning fleet reduction do not show significant variations in its behavior. Figure 2a shows that the distribution occurs around the median. There is no variation for 50% of results between the first and third quartile. The total amplitude was 0.34%. These results show the stability of the SGVNS algorithm concerning fleet reduction.

Figure 2b shows the results for distance deviation. It is possible to verify the stability in the behavior of the proposed algorithm concerning the distances it found. For MDOVRPTW, there is a variation of 4.27% in 50% of the results around the median values, and for MDOVRPTW*, this variation is 4.95%. The distances found by the SGVNS algorithm for MDOVRPTW* have a more significant number of outliers, which indicates the greater variation between these distances. Such variations are not atypical since distance reduction is not prioritized. When analyzing the distances through the errors related to the median values, it is concluded that there are no statistically significant differences between the two problems evaluated.

8 General discussions and conclusions

This article introduced a variant for the MDOVRPTW, called MDOVRPTW*. In the MDOVRPTW, the goal is to reduce the distance traveled, while in MDOVRPTW*, the priority is to reduce the fleet size. An algorithm based on the VNS heuristic, called SGVNS, was developed for treating them. The SGVNS algorithm uses nine neighborhood structures based on insertion, swap, and shift moves as local search operators to refine the incumbent solution and three neighborhood structures as perturbation operators to escape from each local optimum found. As there are no specific instances for the addressed problems, the current article adopted the instances from Cordeau et al. [45] (Group I) and those defined in Vidal et al. [37] (Group II), originally proposed for the MDVRPTW.

The results achieved by SGVNS for the two addressed problems were compared with the best-found results for the MDVRPTW. In Group I, there was a 12.07%

reduction in the total distance traveled for the MDOVRPTW and a 10.42% reduction in fleet size for the MDOVRPTW*. In Group II, the algorithm achieved a 10.43% reduction in total distance traveled for the MDOVRPTW and a 24.13% reduction in fleet size for the MDOVRPTW*. Considering all instances (Groups I and II), there was a reduction in the total distance traveled for all of them and, besides, a fleet reduction in 92.65% of them. These results validate the proposition of MDOVRPTW as an important variant of vehicle routing problems and the proposed algorithm as a technique to solve these addressed problems.

In future work, we will address the challenge of including adaptive procedures in local search structures. Recently, several studies have shown high-quality results in the form of adaptation procedures in choosing neighborhoods like Derbel et al. [26], Karakostas et al. [29, 31], Ren et al. [28], Sanchez-Oro et al. [27], Smiti et al. [24] and in procedures involving machine learning techniques, like Sevaux et al. [47], Talbi [48], Karimi-Mamaghan et al. [49], and Silva et al. [50].

Acknowledgements The authors would like to thank the Brazilian Council of Technological and Scientific Development (CNPq), the Minas Gerais State Research Foundation (FAPEMIG), the Federal Center of Technological Education of Minas Gerais (CEFET-MG) and the Federal University of Ouro Preto (UFOP) for supporting this research. The authors also would like to thank a lot Profa. Elisangela Martins de Sá, for the valuable assistance in formulating the proposed mathematical model.

References

1. Lalla-Ruiz, E., Mes, M.: Mathematical formulations and improvements for the multi-depot open vehicle routing problem. Optim. Lett. **15**, 271–286 (2021)
2. Shen, L., Tao, F., Wang, S.: Multi-depot open vehicle routing problem with time windows based on carbon trading. Int. J. Environ. Res. Public Health **15**(9), 2025 (2018)
3. Brandão, J.: A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem. Eur. J. Oper. Res. **284**(2), 559–571 (2020)
4. Repoussis, P., Tarantilis, C., Ioannou, G.: The open vehicle routing problem with time windows. J. Oper. Res. Soc. **58**, 355–367 (2007)
5. Cordeau, J.F., Laporte, G., Mercier, A.: A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. **52**, 928–936 (2001)
6. Rego, M.F., Souza, M.J.F.: Smart general variable neighborhood search with local search based on mathematical programming for solving the unrelated parallel machine scheduling problem. In: Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019), vol. 2, pp. 287–295. SciTePress, Heraklion, Greece (2019). https://doi.org/10.5220/0007703302870295. INSTICC
7. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO J. Comput. Optim. **5**(3), 423–454 (2017)
8. Subramanian, A., Drummond, L.M.A., Bentes, C., Ochi, L.S., Farias, R.: A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Comput. Oper. Res. **37**(11), 1899–1911 (2010). (**Metaheuristics for Logistics and Vehicle Routing**)
9. Subramanian, A.: Heuristic, exact and hybrid approaches for vehicle routing problems. PhD thesis, Universidade Federal Fluminense, Niterói, Brazil (2012). Available at http://www.ic.uff.br/PosGraduacao/frontend-tesesdissertacoes/download.php?id=532.pdf &tipo=trabalho
10. de Freitas, J.C., Penna, P.H.V.: A variable neighborhood search for flying sidekick traveling salesman problem. Int. Trans. Oper. Res. **27**(1), 267–290 (2020)

11. Coelho, V.N., Grasas, A., Ramalhinho, H., Coelho, I.M., Souza, M.J.F., Cruz, R.C.: An ILS-based algorithm to solve a large-scale real heterogeneous fleet VRP with multi-trips and docking constraints. Eur. J. Oper. Res. **250**(2), 367–376 (2016)
12. 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. Ann. Oper. Res. **275**(1), 5–74 (2013)
13. Bezerra, S.N., Souza, M.J.F., de Souza, S.R.: A variable neighborhood search-based algorithm with adaptive local search for the vehicle routing problem with time windows and multi-depots aiming for vehicle fleet reduction. Comput. Oper. Res. **149**, 106016 (2023). https://doi.org/10.1016/j.cor.2022.106016
14. Schrage, L.: Formulation and structure of more complex/realistic routing and scheduling problems. Networks **11**(2), 229–232 (1981)
15. Brandão, J.: A tabu search algorithm for the open vehicle routing problem. Eur. J. Oper. Res. **157**, 552–564 (2004)
16. Hosseinabadi, A.A.R., Vahidi, J., Balas, V., Mirkamali, S.: OVRP_GELS: solving open vehicle routing problem using the gravitational emulation local search algorithm. Neural Comput. Appl. **29**, 955–968 (2018)
17. Redi, A.A.N.P., Maghfiroh, M.F.N., Yu, V.F.: An improved variable neighborhood search for the open vehicle routing problem with time windows. In: 2013 IEEE International Conference on Industrial Engineering and Engineering Management, Bangkok, Thailand, pp. 1641–1645 (2013)
18. Chen, B., Qu, R., Bai, R., Laesanklang, W.: A variable neighborhood search algorithm with reinforcement learning for a real-life periodic vehicle routing problem with time windows and open routes. RAIRO Oper. Res. **54**(5), 1467–1494 (2019)
19. Soto, M., Sevaux, M., Rossi, A., Reinholz, A.: Multiple neighborhood search, tabu search and ejection chains for the multi-depot open vehicle routing problem. Comput. Ind. Eng. **107**, 211–222 (2017). https://doi.org/10.1016/j.cie.2017.03.022
20. Lahyani, R., Gouguenheim, A.-L., Coelho, L.C.: A hybrid adaptive large neighbourhood search for multi-depot open vehicle routing problems. Int. J. Prod. Res. **57**(22), 6963–6976 (2019). https://doi.org/10.1080/00207543.2019.1572929
21. Li, J., Li, Y., Pardalos, P.M.: Multi-depot vehicle routing problem with time windows under shared depot resources. J. Comb. Optim. **31**(2), 515–532 (2016). https://doi.org/10.1007/s10878-014-9767-4
22. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. **35**(2), 254–265 (1987)
23. Tarantilis, C.D., Kiranoudis, C.T.: Distribution of fresh meat. J. Food Eng. **51**(1), 85–91 (2002). https://doi.org/10.1016/S0260-8774(01)00040-1
24. Smiti, N., Dhiaf, M., Jarboui, B., Hanafi, S.: Skewed general variable neighborhood search for the cumulative capacitated vehicle routing problem. Int. Trans. Oper. Res. **27**(1), 651–664 (2018). https://doi.org/10.1111/itor.12513
25. Bezerra, S.N., de Souza, S.R., Souza, M.J.F.: A GVNS algorithm for solving the multi-depot vehicle routing problem. Electron. Notes Discrete Math. **66**, 167–174 (2018). (**Proceedings of the 5th International Conference on Variable Neighborhood Search**)
26. Derbel, H., Jarboui, B., Bhiri, R.: A skewed general variable neighborhood search algorithm with fixed threshold for the heterogeneous fleet vehicle routing problem. Ann. Oper. Res. **272**, 243–272 (2019). https://doi.org/10.1007/s10479-017-2576-2
27. Sánchez-Oro, J., López-Sánchez, A.D., Colmenar, J.M.: A general variable neighborhood search for solving the multi-objective open vehicle routing problem. J. Heuristics **26**(3), 423–452 (2020). https://doi.org/10.1007/s10732-017-9363-8
28. Ren, X., Huang, H., Feng, S., Liang, G.: An improved variable neighborhood search for bi-objective mixed-energy fleet vehicle routing problem. J. Clean. Prod. **275**, 124155 (2020). https://doi.org/10.1016/j.jclepro.2020.124155
29. Karakostas, P., Sifaleras, A., Georgiadis, M.C.: Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. Expert Syst. Appl. **153**, 113444 (2020). https://doi.org/10.1016/j.eswa.2020.113444
30. Barrero, L., Robledo, F., Romero, P., Viera, R.: A GRASP/VND heuristic for the heterogeneous fleet vehicle routing problem with time windows. In: Mladenovic, N., Sleptchenko, A., Sifaleras, A., Omar, M. (eds.) Proceedings of the 8th International Conference on Variable Neighborhood Search (ICVNS 2021). Lecture Notes in Computer Science, vol. 12559, pp. 152–165. Springer, Cham (2021)

31. Karakostas, P., Sifaleras, A.: A double-adaptive general variable neighborhood search algorithm for the solution of the traveling salesman problem. Appl. Soft Comput. **121**, 108746 (2022). https://doi.org/10.1016/j.asoc.2022.108746
32. Hansen, P., Mladenović, N., Moreno Pérez, J.A.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. **175**(1), 367–407 (2010)
33. Reinsma, J.A., Penna, P.H.V., Souza, M.J.F.: A simple and efficient algorithm for solving the generalized traveling salesman problem (in Portuguese). In: Proceedings of the L Brazilian Symposium of Operations Research. Galoá, Rio de Janeiro, Brazil, SOBRAPO (2018). Available at https://proceedings.science/proceedings/100015/_papers/85522/download/fulltext_file1
34. Souza, M.J.F., Coelho, I.M., Ribas, S., Santos, H.G., Merschmann, L.H.C.: A hybrid heuristic algorithm for the open-pit-mining operational planning problem. Eur. J. Oper. Res. **207**(2), 1041–1051 (2010)
35. Ombuki, B., Ross, B.J., Hanshar, F.: Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl. Intell. **24**(1), 17–30 (2006)
36. Polacek, M., Benkner, S., Doerner, K.F., Hartl, R.F.: A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. Bus. Res. **1**(2), 207–218 (2008). https://doi.org/10.1007/BF03343534
37. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. Eur. J. Oper. Res. **231**(1), 1–21 (2013)
38. Gillett, B.E., Johnson, J.G.: Multi-terminal vehicle-dispatch algorithm. Omega **4**(6), 711–718 (1976)
39. Polacek, M., Hartl, R.F., Doerner, K., Reimann, M.: A variable neighborhood search for the multi depot vehicle routing problem with time windows. J. Heuristics **10**(6), 613–627 (2004). https://doi.org/10.1007/s10732-005-5432-5
40. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Comput. Oper. Res. **34**(8), 2403–2435 (2007)
41. Subramanian, A., Penna, P.H.V., Ochi, L.S., Souza, M.J.F.: Um algoritmo heurístico baseado em Iterated Local Search para problemas de roteamento de veículos. In: Lopes, H.S., de Abreu Rodrigues, L.C., Steiner, M.T.A. (eds.) Meta-Heurísticas em Pesquisa Operacional, 1st edn., pp. 165–180. Omnipax, Curitiba, PR (2013). Chap. 11. Available at http://www.decom.ufop.br/prof/marcone/projects/ppm497-13/MetaheuristicasPesquisaOperacional-Cap11.pdf
42. Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. **60**(3), 611–624 (2012)
43. Vidal, T., Crainic, T.G., Gendreau, M., Prins, C.: Implicit depot assignments and rotations in vehicle routing heuristics. Eur. J. Oper. Res. **237**(1), 15–28 (2014). https://doi.org/10.1016/j.ejor.2013.12.044
44. Christiaens, J., Vanden Berghe, G.: Slack induction by string removals for vehicle routing problems. Transp. Sci. **54**(2), 417–433 (2020). https://doi.org/10.1287/trsc.2019.0914
45. Cordeau, J.F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F.: Vrp with time windows. Chap. 7. In: Toth, P., Vigo, D. (eds.) The Vehicle Routing Problem. Discrete Mathematics and Applications, pp. 157–193. Society for Industrial and Applied Mathematics, Philadelphia (2002)
46. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The IRACE package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016)
47. Sevaux, M., Sörensen, K., Pillay, N.: Adaptive and multilevel metaheuristics. In: Martí, R., Pardalos, P.M., Resende, M.G.C. (eds.) Handbook of Heuristics, pp. 3–21. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-07124-4_16
48. Talbi, E.: Machine learning into metaheuristics: a survey and taxonomy. ACM Comput. Surv. (CSUR) **54**(6), 1–32 (2021). https://doi.org/10.1145/3459664
49. Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M., Talbi, E.-G.: Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: a state-of-the-art. Eur. J. Oper. Res. **296**(2), 393–422 (2022)
50. Silva, M.A.L., de Souza, S.R., Souza, M.J.F., Bazzan, A.L.C.: A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. Expert Syst. Appl. **131**, 148–171 (2019)

Authors and Affiliations

Sinaide Nunes Bezerra[1] · Sérgio Ricardo de Souza[1] · Marcone Jamilson Freitas Souza[2]

Sérgio Ricardo de Souza
sergio@dppg.cefetmg.br

Marcone Jamilson Freitas Souza
marcone@ufop.edu.br

[1] PPGMMC, Federal Center of Technological Education of Minas Gerais (CEFET-MG), Nova Gameleira, Belo Horizonte, MG 30510-000, Brazil

[2] DECOM, Federal University of Ouro Preto (UFOP), Campus Universitário, Morro do Cruzeiro, Ouro Preto, MG 35400-000, Brazil