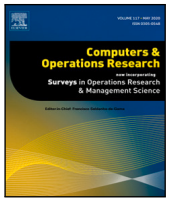
Computers & Operations Research 149 (2023) 106016

Contents lists available at ScienceDirect

Computers and Operations Research

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



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

Sinaide Nunes Bezerra [a], Marcone Jamilson Freitas Souza [b], Sérgio Ricardo de Souza [a,*]

[a] Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), 30510-000, Belo Horizonte, MG, Brazil
[b] Universidade Federal de Ouro Preto (UFOP), Campus Universitário, Morro do Cruzeiro, Ouro Preto, MG, 35400-000, Brazil

ARTICLE INFO

Keywords:
Multi-depot Vehicle Routing Problem with Time Windows
Variable neighborhood search
Neighborhood
Local search

ABSTRACT

This article addresses the Multi-Depot Vehicle Routing Problem with Time Windows with the minimization of the number of used vehicles, denominated as MDVRPTW*. This problem is a variant of the classical MDVRPTW, which only minimizes the total traveled distance. We developed an algorithm named Smart General Variable Neighborhood Search with Adaptive Local Search (SGVNSALS) to solve this problem, and, for comparison purposes, we also implemented a Smart General Variable Neighborhood Search (SGVNS) and a General Variable Neighborhood Search (GVNS) algorithms. The SGVNSALS algorithm alternates the local search engine between two different strategies. In the first strategy, the Randomized Variable Neighborhood Descent method (RVND) performs the local search, and, when applying this strategy, most successful neighborhoods receive a higher score. In the second strategy, the local search method is applied only in a single neighborhood, chosen by a roulette method. Thus, the application of the first local search strategy serves as a learning method for applying the second strategy. To test these algorithms, we use benchmark instances from MDVRPTW involving up to 960 customers, 12 depots, and 120 vehicles. The results show SGVNSALS performance surpassed both SGVNS and GVNS concerning the number of used vehicles and covered distance. As there are no algorithms in the literature dealing with MDVRPTW*, we compared the results from SGVNSALS with those of the best-known solutions concerning these instances for MDVRPTW, where the objective is only to minimize the total distance covered. The results showed that the proposed algorithm reduced the vehicle fleet by 91.18% of the evaluated instances, and the fleet size achieved an average reduction of up to 23.32%. However, there was an average increase of up to 31.48% in total distance traveled in these instances. Finally, the article evaluated the contribution of each neighborhood to the local search and shaking operations of the algorithm, allowing the identification of the neighborhoods that most contribute to a better exploration of the solution space of the problem.

1. Introduction

Several factors, like population growth, globalization, the increasing informatization, the need to decentralize points of sale, the large variety of products, among others, have contributed to the increase in the complexity of the distribution networks of goods and services. In this context, companies that deal with the logistic of goods are subject to continually suit their work in response to different scenarios. These adjustments may be related to the number of distribution centers, the fleet size, the capacity of the vehicles allocated, and the schedule of their activities. Distribution and business logistical processes are fundamental tools to face increasingly competitive markets. This context, aggravated as of 2020 by the enormous impacts of the SARS-Cov-2 pandemic on the entire global economic process, has led to the emergence of research on issues in the distribution processes management, intending to reduce costs by employing shorter routes and fewer vehicles involved.

The Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) is a formulation suitable for modeling logistical situations that meet the described framework. According to Polacek et al. (2004), the first article to effectively describe this problem is Cordeau et al. (2001b). Several works, since then, addressed this problem, the majority in terms of the originally proposed formulation. This formulation considers, as a hypothesis, that all vehicles in the homogeneous fleet must be used in routing. This hypothesis, therefore, imposes that there is no possibility of reducing the vehicle fleet and, consequently, of routes, during the solution process. Thus, this formulation does not represent, in fact, an extension of the Vehicle Routing Problem with

* Corresponding author.
E-mail addresses: sinaide@hotmail.com (S.N. Bezerra), marcone@ufop.edu.br (M.J.F. Souza), sergio@dppg.cefetmg.br (S.R de Souza).

https://doi.org/10.1016/j.cor.2022.106016
Received 27 September 2021; Received in revised form 10 August 2022; Accepted 3 September 2022
Available online 7 September 2022

Time Windows (VRPTW) for the approach of multiple depots. VRPTW, according to Ombuki et al. (2006), is solved hierarchically, firstly minimizing the total number of used vehicles (and, consequently, routes) and, later, minimizing the total distance covered. This solution methodology of VRPTW considers, therefore, as the main cost to be minimized, the addition of a new vehicle, and not the total distance covered by these vehicles.

This article proposes a variant of MDVRPTW as an extension of the Vehicle Routing Problem with Time Windows (VRPTW). In this formulation, which we call MDVRPTW*, a hierarchical solution method is adopted, as the one applied for solving the VRPTW; therefore, this solution method first minimizes the number of vehicles and, following, minimizes the total distance covered. According to Christiaens and Vanden Berghe (2020), the fleet reduction reflects the need for real-world transport companies to accommodate their requests. In this sense, the proposed method seeks the best adaptation to the planning horizon, reducing the costs related to the number of vehicles involved in the goods delivery. For the solution of MDVRPTW*, an algorithm called Smart General Variable Neighborhood Search with Adaptive Local Search (SGVNSALS) is developed, based on the Variable Neighborhood Search (VNS) metaheuristic, proposed by Mladenović and Hansen (1997). Additionally, to compare and validate the results obtained by the SGVNSALS algorithm, we implement the Smart General Variable Neighborhood Search (SGVNS) and the General Variable Neighborhood Search (GVNS), which is one of the main variants of VNS (Hansen et al., 2017, 2019).

This article has as its central objective to solve MDVRPTW* with the application of SGVNSALS. The main contributions are:

(i) the proposition of MDVRPTW* as an extension of VRPTW with multiple depots (MDVRPTW), seeking the reduction of the used vehicle fleet;
(ii) the development of SGVNSALS, a VNS-based hybrid algorithm to solve MDVRPTW*;
(iii) the assessment of neighborhood structures used in SGVNSALS when solving MDVRPTW*.

The report described in this article is structured as follows: the next section presents the complete description of MDVRPTW*. Section 3 shows a bibliographical review of the problem, verifying how different authors deal with the subject under study. Section 4 introduces the proposed SGVNSALS algorithm, describing the neighborhoods, the local search operators, and the shake procedures, as well as describes the implementation of the SGVNS and GVNS algorithms. It details the actions related to changes in the choice of neighborhood structures. Section 5 shows the computational results obtained with the developed algorithm, statistical analysis for the results, and an evaluation concerning the influence of the neighborhoods and the local search operators in the solution procedure. Finally, Section 6 refers to the conclusions involving the general analysis of the work carried out.

2. Multi-Depot Vehicle Routing Problem with Time-Windows

The Multi-Depot Vehicle Routing Problem with Time-Windows (MDVRPTW), according to its original formulation introduced in Cordeau et al. (2001b), consists of determining a set of routes that minimize the total traveled distance, serving all customers, respecting the maximum number of vehicles allocated per depot, 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, as stated by Cordeau et al. (2001b) and Polacek et al. (2004), are:

(i) the number of depots is greater than one;
(ii) each vehicle starts and ends its route in the same depot;
(iii) the capacity of each vehicle is known;
(iv) the vehicle fleet is homogeneous;

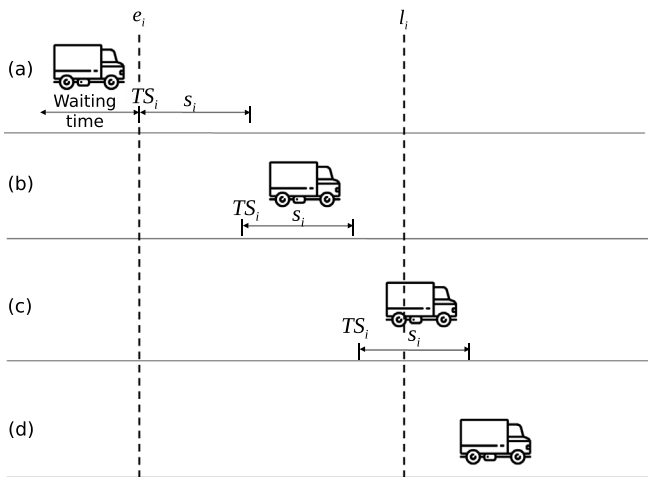


Fig. 1. Time window for customer service.

(v) each customer is served by only one vehicle;
(vi) the total demand for each route cannot exceed the capacity of the vehicle that is on the route;
(vii) the customer's time window must be respected;
(viii) each route has a maximum duration to be covered;
(ix) all vehicles of the fleet are used, without any leftovers, in the routing process.

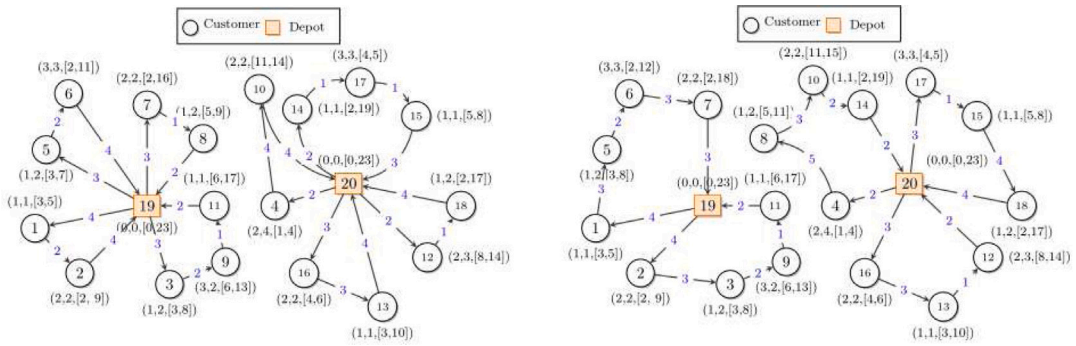
Therefore, as a result of the characteristic (ix) above, the number of existing routes is exactly equal to the number of vehicles in the fleet.

In this article, on the other hand, we address a variant of this problem, which we name MDVRPTW*. This variant is an extension of the solution strategy found in the researches concerning VRPTW to the multi-depot vehicle routing problem case. This solution strategy consists of hierarchically constructing the problem solution so that, first, we minimize the number of used vehicles and, subsequently, minimize the total traveled distance. Consequently, the characteristic (ix) defined above does not apply to the variant here approached since, by minimizing the number of used vehicles, not all vehicles in the fleet will be used in the routing process. The characteristic (ix) is, then, replaced by the following formulation:

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

The MDVRPTW* is an adaptation of the definitions for the MDVRPTW, proposed previously by Cordeau et al. (2001a), Polacek et al. (2004), and Montoya-Torres et al. (2015). This problem, in which N customers are served by M depots, is defined from a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, with $\mathcal{V}$ 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 depots set $\mathcal{D} = \{N + 1, N + 2, \ldots, N + M\}$, 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 $|\mathcal{K}| = |\mathcal{K}_{N+1}| + |\mathcal{K}_{N+2}| + \cdots + |\mathcal{K}_{N+M}|$. 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 also a service time h_i that starts at TS_i, such that $e_i \leq TS_i \leq l_i$. 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 values.

Fig. 1 illustrates the possible situations encountered in the delivery of products regarding the time window for customer service. In this



(a) Example of a solution to MDVRPTW. Note there are eight routes and the number of routes per depot are equal, i.e., $|\mathcal{K}_{19}| = |\mathcal{K}_{20}| = 4$.

(b) Example of a solution to MDVRPTW*. Note there are five routes and and the number of routes per depot are different, i.e., $|\mathcal{K}_{19}| = 2$ and $|\mathcal{K}_{20}| = 3$.

Fig. 2. Example of Solutions to the MDVRPTW and MDVRPTW*.

Table 1
Data for the example shown in Figs. 2(a) and 2(b).

Data for Fig. 2(a)								Data for Fig. 2(b)						
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, 5, 6, 19)	(3, 2, 4)	(0,2, 3, 0)	0	14	(0,1, 3, 0)	4	(19, 1, 5, 6, 7, 19)	(4, 3, 2, 3, 3)	(0,1, 2, 3, 2, 0)	0	23	(0,1, 1, 3, 2, 0)	7
2	(19, 7, 8, 19)	(3, 1, 2)	(0,2, 2, 0)	0	10	(0,2, 1, 0)	3	(19, 2, 3, 9, 11, 19)	(4, 3, 2, 1, 2)	(0,2, 2, 2, 1, 0)	0	19	(0,2, 1, 3, 1, 0)	7
3	(19, 3, 9, 11, 19)	(3, 2, 1, 2)	(0,2, 2, 1, 0)	0	13	(0,1, 3, 1, 0)	5	(20, 4, 8, 10, 14, 20)	(2, 5, 3, 2, 2)	(0,4, 2, 2, 1, 0)	0	23	(0,2, 1, 2, 1, 0)	6
4	(19, 1, 2, 19)	(4, 2, 4)	(0,1, 2, 0)	0	13	(0,1, 2, 0)	3	(20, 17, 15, 18, 20)	(3, 1, 4, 4)	(0,3, 1, 2, 0)	1	19	(0,3, 1, 1, 0)	5
5	(20, 4, 10, 20)	(2, 4, 4)	(0,4, 2, 0)	0	16	(0,2, 2, 0)	4	(20, 16, 13, 12, 20)	(3, 3, 1, 2)	(0,2, 1, 3, 0)	1	16	(0,2, 1, 2, 0)	5
6	(20, 14, 17, 15, 20)	(2, 1, 1, 3)	(0,1, 3, 1, 0)	0	12	(0,1, 3, 1, 0)	5							
7	(20, 12, 18, 20)	(2, 1, 4)	(0,3, 2, 0)	0	12	(0,2, 1, 0)	3							
8	(20, 16, 13, 20)	(3, 3, 4)	(0,2, 1, 0)	0	13	(0,2, 1, 0)	3							
Cost					103							100		

representation, the first three occurrences result in the delivery of the order. In the situation (a), the vehicle arrives at the customer i before e_i, which leads to a waiting time until $TS_i = e_i$. In situations (b) and (c), the vehicle arrives at the customer so that $e_i \leq TS_i \leq l_i$, immediately starting the delivery. In (b), the product is unloaded entirely in the time window interval, and in (c), although $TS_i + h_i > l_i$, there does not prevent the delivery. In situation (d), the vehicle arrives at the customer with $TS_i > l_i$, and the delivery cannot be performed.

To better understand the differences between the MDVRPTW and MDVRPTW* problems, we introduce, in the following, an example, which presents solutions of a small dimension instance for these two treated variations. The approached instance has two depots, which serve eighteen customers. There is a homogeneous fleet of eight vehicles, each one with a capacity of $VC = 7$. The en-route time of each vehicle cannot exceed $MT = 23$. 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 i node. Fig. 2(a) illustrates an MDVRPTW solution for this instance, according to the model proposed by Cordeau et al. (2001b). On the other hand, Fig. 2(b) shows an MDVRPTW* solution for this same instance, as this article defines the MDVRPTW*.

Table 1 shows the data for the two solutions. For each vehicle k, the column Routers shows the used routes; the column WU_{ki}, the waiting time to unload the vehicle k on the customer i; the column 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. Fig. 2 includes data relating to the travel cost c_{ij} between the vertices i and j and the tuple $(q_i,\ h_i, [e_i, l_i])$ for each vertex. Concerning the solution for MDVRPTW, shown in Fig. 2(a), the number of used vehicles is fixed and equal to the size of the fleet itself. In the case of the solution for MDVRPTW*, shown in Fig. 2(a), the smallest fleet that guarantees all the requirements, except the fixed fleet size, is sought. Thus, in MDVRPTW* case, the number of used vehicles decreases to five vehicles. The importance of this variant is to show that, from the same set of depots and customers, it is possible to find compatible solutions in terms of travel costs with a smaller number of vehicles.

The mathematical formulation for the problem under analysis, developed from the one presented in Li et al. (2016), is given by:

$$\min\ \beta \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} c_{ij}x_{kij} + \alpha \sum_{k\in\mathcal{K}} y_k, \tag{1}$$

Subject to:

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}} x_{kij} = \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}} x_{kji} = 1,\ \forall i \in C, \tag{2}$$

$$\sum_{d\in D}\sum_{j\in C} x_{kdj} = \sum_{d\in D}\sum_{i\in C} x_{kid} \leq 1,\ \forall k \in \mathcal{K}, \tag{3}$$

$$\sum_{i\in\mathcal{L}}\sum_{j\in\mathcal{L}} x_{kij} \leq |\mathcal{L}| - 1,\ \forall \mathcal{L} \subseteq C, k \in \mathcal{K}, \tag{4}$$

$$\sum_{i\in D}\sum_{j\in D} x_{kij} = 0,\ \forall k \in \mathcal{K}, \tag{5}$$

$$x_{kij}(TS_{ki} + h_i + c_{ij} - TS_{kj}) \leq 0,\ \forall k \in \mathcal{K}, \forall i, j \in \mathcal{V}, \tag{6}$$

$$e_i \leq TS_{ki} \leq l_i,\ \forall i \in C, \forall k \in \mathcal{K}, \tag{7}$$

$$\sum_{i\in\mathcal{V}}\sum_{j\in C} q_j x_{kij} \leq VC,\ \forall k \in \mathcal{K}, \tag{8}$$

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} (c_{ij} + h_j)x_{kij} \leq MT,\ \forall k \in \mathcal{K}, \tag{9}$$

$$y_k \geq \frac{\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} x_{kij}}{|N|^2},\ \forall k \in \mathcal{K}, \tag{10}$$

$$x_{kij}, y_k \in \{0,1\}, \quad \forall\, i,j \in \mathcal{V}, \forall k \in \mathcal{K}. \tag{11}$$

Expression (1) represents the objective function of the problem, to be minimized. This objective function is given by the weighted sum between the travel cost between customers and the total number of involved vehicles. The α and β weighting factors show the relative importance of each of the two cost parts. Constraints (2) ensure that each consumer is visited by exactly one vehicle. Constraints (3) indicate that each vehicle starts and returns to the same depot. Constraints (4) guarantee that the graph is connected, as well as the subtours elimination. Restrictions (5) impose that the vehicle cannot travel directly from depot i to depot j. Constraints (6) imply the order of visits to the nodes, because 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 (7) guarantee the occurrence of the i customer service within the time window $[e_i, l_i]$. Constraints (8) ensure that the load will not exceed the capacity of the vehicle k. Constraints (9) ensure that the total duration of the route is at most equal to T. Constraints (10) indicate which vehicles will be used, and, finally, Constraints (11) define the decision variables x and y as binary.

3. Literature review

The studies found in the literature are only related to MDVRPTW, in which the focus is on minimizing the total distance traveled by vehicles. This section, in consequence, addresses the main works treating this problem to allow the reader to understand the scenario for introducing the MDVRPTW* variation and its proposed solution procedure.

MDVRPTW was originally introduced in Cordeau et al. (2001b). This work presented a solution using the Tabu Search (TS) metaheuristic, also including the first set of instances for this problem. This set of instances continues to be used as a reference for experiments with this problem. In Cordeau et al. (2004) the authors improved this original solution proposal that they had presented, producing new results, and, among them, five still remain as the best obtained for the respective instances.

Polacek et al. (2004) proposed a VNS-based algorithm, with acceptance criteria for the best solutions, together with the choice of neighborhoods in the local search, using, for the computational experiments, the same group of instances introduced in Cordeau et al. (2001b). The evaluations carried out showed that this proposed algorithm showed competitive results to the TS algorithm presented by Cordeau et al. (2001b) concerning the quality of the solution obtained and the computational effort required. Polacek et al. (2008) showed a parallel version of the VNS-based algorithm introduced in Polacek et al. (2004), using new intensification and diversification strategies through cooperation adaptation schemes. The results showed that all best-known results in the literature for the instances of Cordeau et al. (2001b) were found. Additionally, this algorithm provided new best results in 11 out of the 20 instances.

Tansini and Viera (2006) presented heuristics for clustering customers to depots, using, as a reference, time windows and distances in assessing the proximity between customers and depots. The method used greedy assignments and allowed a reduction between 5% and 6% for the distance covered in the tested instances.

Ting and Chen (2008), in turn, introduced a hybrid metaheuristic involving Multiple Ant Colony System (MACS) and Simulated Annealing (SA). This hybrid metaheuristic obtained three new best-known results for the instances presented in Cordeau et al. (2001b). Cordeau and Maischberger (2012) showed a parallel hybrid metaheuristic, combining Iterated Local Search (ILS) and Tabu Search (TS) to promote further exploration of the solution space of the problem. This algorithm led to results similar to those found in Polacek et al. (2008), however, achieving six new best-known results, with small improvements to the previous values.

Tamashiro et al. (2010) presented a Tabu Search approach combined with an extended saving method for solving the MDVRPTW. The proposed algorithm consists of two phases. The first phase is a Tabu Search method, applied for solving the customer assignment problem and, at the same time, to lead to neighborhood reduction. The second phase uses the saving method for constructing the routes for the depots.

Noori and Ghannadpour (2012) proposed a three-level hybrid metaheuristic to solve the MDVRPTW. In the first level, the "Cluster-first, Route second" method is applied, associating customers to depots; in the second level, for each cluster, a Genetic Algorithm (GA) is used to optimize the routes; and, in the third phase, the solutions are refined with the Tabu Search metaheuristic. The results obtained by this hybrid metaheuristic reached those shown in Polacek et al. (2004).

Vidal et al. (2013a,b) proposed a hybrid genetic algorithm with adaptive diversity management (Hybrid Genetic Search with Advanced Diversity Control — HGSADC) to solve a wide range of problems derived from VRPTW. Regarding the MDVRPTW, this algorithm not only achieved all the best-known solutions of the set of instances from Cordeau et al. (2001b) but also added nine new best-known solutions. Additionally, the authors proposed a new set of instances, as an extension with large dimensions for the set from Cordeau et al. (2001b).

Montoya-Torres et al. (2015) presented a literature review on the MDVRP, evaluating articles published until 2014. According to the authors, 88.44% addressed this problem considering a single objective, and 25% of these were related to the MDVRPTW. Several variations of MDVRPTW are listed, without, however, showing authors that addressed the MDVRPTW* variant treated here. Among these variants, we highlight the proposition from Li et al. (2016). In this article, the authors treated the Multi-depot vehicle routing problem with time windows under shared depot resources (MDVRPTWSDR), in which the vehicles leave a certain depot but can return to a different depot. A hybrid genetic algorithm with adaptive local search (HGA_ALS) is proposed for solving this problem, using, for performing the computational experiments, ten out twenty instances from Cordeau et al. (2001b). Wang et al. (2019b) represents an important continuity of the approach from Li et al. (2016), including the environmental issue as fundamental for future research on this class of problems.

Hesam Sadati et al. (2021) presented an interesting hybrid implementation of VNS with Tabu Search, named Variable Tabu Neighborhood Search (VTNS), for solving the Multi-Depot Vehicle Routing Problems (MDVRP), MDVRP with Time Windows (MDVRPTW), and Multi-Depot Open Vehicle Routing Problem (MDOVRP). The proposed algorithm included, in the intensification phase, a granular local search procedure, and, in the intensification phase, a tabu shaking procedure. The use of tabu search allows working with infeasible solutions for escaping from local optima. The results were very competitive concerning the state-of-the-art algorithms in the literature. However, the authors did not work with the large instances from Vidal et al. (2013b), Lahrichi et al. (2015) for testing the algorithm.

On the other hand, methods based on neighborhood exploration have been widely applied to solve vehicle routing problems. Belloso et al. (2019) proposed a multistart biased-randomized heuristic to solve the fleet size and mix vehicle routing problem with backhauls (FSMVRPB). The results show a competitive approach that determined 20 new solutions for the 36 evaluated instances. Rezgui et al. (2019) showed a adapted General Variable Neighborhood Search (GVNS) for solving the fleet size and mix vehicle routing problem with time windows using electric modular vehicles. This work adapted the instances from Solomon (1987) to evaluate the proposed method. Ren et al. (2020) addressed the bi-objective mixed-energy green vehicle routing problem with time windows (B-MFGVRPTW). The first objective was to minimize pollutant emissions by minimizing the use of gasoline and oil vehicles. The second objective was to minimize the total delay time to maximize customer satisfaction by penalizing the deliveries performed out of the time windows. For solving this problem, this work applied

a multi-objective version of the GVNS metaheuristic on the instances from Solomon, building the Pareto frontiers solutions.

Karakostas et al. (2020a) introduced fleet-size and mix pollution location-inventory-routing problem with just-in-time replenishment policy and capacity planning. This article extends the strategic-level decisions of a classic location-inventory-routing problem (LIRP), considering capacity selection decisions and heterogeneous fleet composition. The authors proposed an algorithm based on the GVNS metaheuristic with six different local search strategies that use the Variable Neighborhood Descent method, applying it for solving a set of instances proposed by them and comparing it with the solution via CPLEX. Karakostas et al. (2020b) proposed three versions of GVNS for the solution of a supply distribution structure in the care of immunotherapy patients.

Masmoudi et al. (2021) introduced the Fleet Size and Mixed Vehicle Routing Problem with Synchronized Visits (FSM-VRPS), i.e., a Vehicle Routing Problem with Synchronized Visits involving a mixed fleet size of vehicles composed of heterogeneous passenger cars and bikes. The article proposed a metaheuristic based on Multi-Start Adaptive Large Neighborhood Search with Threshold Accepting (MS-ALNS-TA) algorithm for solving a set of instances based on benchmark VRPS instances from the literature. The modified adaptive local neighborhood search improved the intensification and diversification mechanisms of the standard ALNS algorithm during the search process.

Another variant of impact to be highlighted is the one involving a heterogeneous fleet, as Rabbouch et al. (2018) and Abdallah and Ennigrou (2020). To conclude this literary review, it is worth noting that the articles treating the multi-objective formulation of MDVRPTW, as Wang et al. (2019a) and Li et al. (2019), has reached a large audience recently, mainly because it allows for the rapid insertion of environmental issues.

Finally, although several and significant variations of MDVRPTW have been addressed in the literature, the MDVRPTW* formulation we have presented here has not been previously treated, at least according to the authors' knowledge of the current article. This formulation is the research gap that we investigate in this article.

4. *Smart* GVNS with Adaptive Local Search (SGVNSALS)

For treating the MDVRPTW*, we propose an algorithm based on the General Variable Neighborhood Search metaheuristic (GVNS) (Hansen et al., 2017). Named SGVNSALS, an acronym for Smart General Variable Neighborhood Search with Adaptive Local Search, this algorithm uses the "smart" version of the GVNS method developed in Rego and Souza (2019) and an adaptive local search. While in the algorithm of Rego and Souza (2019) the local searches are applied following the Variable Neighborhood Descent (VND) (Mladenović and Hansen, 1997) method's structure, in the algorithm proposed here we apply adaptive local searches according to the local search's success in each neighborhood in previous iterations. To describe the proposed algorithm, we use throughout this section the notations introduced in Table 2. Additionally, to better understand the impact of this proposed algorithm in the solution of MDVRPTW*, we also implemented a variation using only the "smart" mechanism and performing the local search using VND, which we named SGVNS, and the classic GVNS algorithm. Section 4.8 explains how this is done.

The use of local search adaptation mechanisms in VNS has been increasing to solve different problems. Among the works that use this procedure, we highlight Li and Tian (2016), Todosijević et al. (2016) and Karakostas et al. (2020b, 2022). Todosijević et al. (2016) proposed an adaptive general variable neighborhood search for solving the unit commitment with a simple adaptive success principle in the shaking procedure. Karakostas et al. (2020b) introduced two adaptive shaking procedures, also with a simple success principle but, on the other hand, with a re-ordering mechanism for the neighborhood operators. Karakostas et al. (2022) employed similar re-ordering mechanisms. Li and Tian (2016) proposed a self-adaptive algorithm where the sequence of the neighborhoods is determined automatically by the algorithm from its search history.

Table 2
Notations used in the description of the SGVNSALS 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 MDVRPTW*
v	Index representing one of the neighborhood structures
$\bar{q}$	Average of demands
$\bar{s}$	Average of customers service times
α	Weight for the number of vehicles
β	Weight for the distance traveled
$\phi(k)$	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 $(i,rd1d2)$
$\mathcal{N}$	Set of neighborhood structures
$\mathcal{R}$	Set of lists that represent the routes of the depots
$\mathcal{U}$	Set of used vehicles

This section is organized as follows. Initially, we describe the working principle of a VNS based algorithm in Section 4.1. In Section 4.2, we show how to represent a solution. Section 4.3 presents the procedure for generating an initial solution to the problem. Section 4.4 describes the function used for evaluating the solution of MDVRPTW*. Section 4.5 describes the neighborhood structures used to explore the solution space of the problem. Section 4.6 details the shaking procedure and Section 4.7 describes the local search method. Finally, Section 4.8 presents the proposed SGVNSALS, SGVNS, and GVNS algorithms.

4.1. Variable Neighborhood Search

Proposed by Mladenović and Hansen (1997), Variable Neighborhood Search (VNS) is a metaheuristic that explores the solution space through systematic changes of neighborhood structures.

Typically, a basic VNS algorithm alternately executes a local search procedure and a shake procedure until a predefined stopping criterion is met. The local search procedure is applied to refine a solution, guiding it to a local minimum. In turn, the shake procedure promotes diversification to guide the search towards other basins of attraction. Shaking operators consist of random moves applied to a given solution s to guide it to a perturbed solution s'. According to Hansen et al. (2017, 2019), perturbations are applied gradually, given by the neighborhood changes and by the number of times the structure is applied to the solution.

Among the VNS variants, General Variable Neighborhood Search (GVNS) (Hansen et al., 2010, 2017, 2019) uses Variable Neighborhood Descent (VND) as its local search method. VND is a descent method that realizes change of neighborhoods in a deterministic way to improve a solution. It has the advantage of returning a local optimum concerning all used neighborhoods. This is an important property because a local optimum concerning several neighborhoods is more likely to be a global optimum than the solution generated as a local optimum for just one neighborhood.

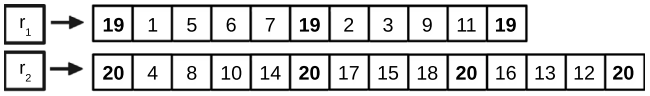


Fig. 3. Example of a solution with two depots. The data are from the example shown in Fig. 2(b).

4.2. Solution representation

In SGVNSALS, a solution s is given by a set of lists $\mathcal{R} = \{r_1, r_2, \ldots, r_{|D|}\}$, in which $|D| \geq 2$ refers to the amount of depots, and r to the routes associated with each one. Fig. 3 shows the solution representation for the scenario described in Section 2 (Fig. 2(b)). In this figure, r_1 and r_2 represent the routes associated with depots 19 and 20, respectively.

4.3. Construction of the initial solution

We use the technique of first grouping and then routing, inspired by the algorithm proposed by Gillett and Johnson (1976), to construct an initial solution. The procedure for generating clusters is shown in Algorithm 1, and, that for generating routes, in Algorithm 2. At lines 8–18 of Algorithm 1, we determine the two closest depots to each customer. Then, we assign the customers to one of these depots (lines 22 to 36). These two depots, called $d1$ and $d2$, are identified in lines 12 and 14 and they have distances $c_{d1,i}$ and $c_{d2,i}$, respectively, to each customer i. The proximity relationships between each customer i and its depots $d1$ and $d2$ are calculated in line 15. The pair $(i,\ c_{d1,i}/c_{d2,i})$ is assigned to the set $\mathcal{HR}$ (line 16). From this rate, $\mathcal{HR}$ is sorted in ascending order (line 19). Unlike Gillett and Johnson (1976), the distribution between depots occurs in a balanced way. That is, for each depot $d \in \mathcal{D}'$ are designated $\lceil |C|/|D| \rceil$ customers (lines 28 to 34). Therefore, in the order given by $\mathcal{HR}$, each pair formed by the customer $i \in C'$, and its respective depot $d1$ (or $d2$), is assigned to the set $\mathcal{HD}$. The assigned customers are removed from the set C' in line 37. In turn, the depots that have reached their capacity limit are also removed (lines 39 to 46). This procedure is repeated until all customers are assigned to a depot. Figs. 4(a) and 4(b) exemplify the assigning procedure with eighteen customers distributed to two depots.

After the assignment, the routes are constructed in a partially greedy way, respecting the customers' windows, the capacity, and maximum route time of the vehicles, as shown in Algorithm 2. The customer i and the depot d, with $(i,d) \in \mathcal{HD}$, are chosen from a previously fixed value $\lambda \in [0,1]$. A real random value between 0 and 1 is chosen. If this value is less than or equal to λ, then we choose any pair $(i,d) \in \mathcal{HD}$ (line 5). Otherwise, we select the first element of the set $\mathcal{HD}$. From empirical tests, we adopt $\lambda = 0.6$. At line 10, the subset $r_d \subset \mathcal{R}$ represents the routes of the depot d. After the composition of the routes by the customers assigned to the depots, the algorithm is finalized in line 13, returning the solution set $\mathcal{R}$ of routes. For all customers to be served, the number of vehicles used can be greater than $|\mathcal{K}|$. Fig. 4(c) shows a representation of the routes for the example in Fig. 3.

4.4. Evaluation function

As in Ombuki et al. (2006), which studied the solution of VRPTW, we evaluate a solution s by a function $F(s)$ that represents a weighted sum of three objective functions to be minimized. The function $F(s)$ is calculated according to:

$$F(s) = \beta L + \alpha|\mathcal{U}| + \varphi \tag{12}$$

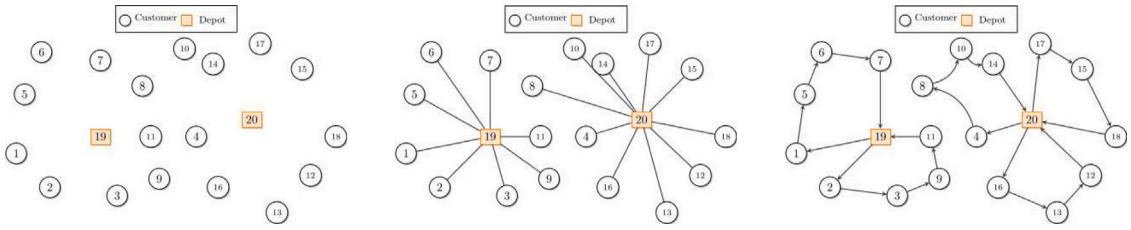
The first two objective functions seek to reduce the total distance traveled (represented by L) by all vehicles and the number of used vehicles (named by $|\mathcal{U}|$ so that $|\mathcal{U}| \leq |\mathcal{K}|$), respectively. The last objective function seeks to minimize the violations (φ) that may occur due to applying the moves to explore the solution space of the problem. These violations can occur due to the excess number of vehicles available in a depot, vehicle overload, customer service delays, and extrapolation of the route's maximum duration.

Algorithm 1: CreateClusters(C, D)

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

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

1: $\mathcal{R} \leftarrow \emptyset$; {Solution set of the MDVRPTW*}
2: **while** $|\mathcal{HD}| > 0$ **do**
3: $\quad j \leftarrow 1$;
4: $\quad$**if** $rand(0,1) \leq \lambda$ **then**
5: $\quad\quad j \leftarrow\ rand(|\mathcal{HD}|)$;
6: $\quad$**end if**
7: $\quad$Let i be the client of the j-th position of $\mathcal{HD} \mid i \in C$;
8: $\quad$Let d be the depot associated with the client i of the j-th position of $\mathcal{HD} \mid d \in \mathcal{D}$;
9: $\quad$Let r_d be the list associated with the depot $d \mid r_d \subset \mathcal{R}$;
10: $\quad r_d \leftarrow r_d \cup \{i\}$; {Compose the routes of the depot d respecting the time windows of the customers, the maximum capacities of the vehicles and the duration of these routes};
11: $\quad\mathcal{HD} \leftarrow \mathcal{HD} \setminus \{(i,d)\}$;
12: **end while**
13: **return** $\mathcal{R}$;



(a) Customer distribution. (b) Depot clustering. (c) Routing.

Fig. 4. Steps for generating an initial solution.

The three objective functions of Eq. (12) are weighted by factors β, α, and φ, that reflect the importance of each objective. As proposed in Ombuki et al. (2006), we used $\beta = 0.001$ and α is calculated by:

$$\alpha = \frac{1}{N} \sum_{i,j \in C}^{N} c_{ij} \tag{13}$$

According to Eq. (13), the weight α represents the average distance between each pair of customers. Finally, the weight φ is calculated according to:

$$\varphi = \sum_{k \in U} \phi(k) \tag{14}$$

where $\phi(k)$ represents the penalization for not respecting the constraints imposed on the route of each vehicle k, being calculated as:

$$\phi(k) = \omega^{Q} \max\left\{0, Q_k - VC\right\} + \omega^{T} \max\left\{0, T_k - MT\right\} + \omega^{TW}\ TW_k \tag{15}$$

In Eq. (15), 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 the 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 weight values are calculated from α, the average demands $\bar{q}$, and the average service times $\bar{h}$ for consumers, as defined below:

$$\omega^{Q} = \frac{\alpha}{\bar{q}}\ , \qquad \omega^{T} = \omega^{TW} = \frac{\alpha}{\bar{h}}$$

The use of penalties for constraint violations is a strategy commonly used in the literature due to the high degree of difficulty in exploring the solution space of problems like VRPTW or variants using only feasible solutions. The algorithms of Polacek et al. (2008) and Vidal et al. (2013a) for solving MDVRPTW are examples of applying this strategy.

4.5. Local search operators

To explore the MDVRPTW* solution space, we use neighborhood operators that apply moves in the same depot in different routes (inter-routes); in the same route (intra-route); or between routes from different depots (inter-depots).

The intra-route operators used are *Swap*, *Reinsertion*, *Or-opt2*, *2-opt*, and *3-opt*. As inter-route operators, we apply the *Swap(1,1)* and *Shift(1,0)* moves. Finally, as inter-depot operators, we apply the same moves that define the inter-route operator, plus the *Swap(2,2)* move. We also apply the *Destruction and Reconstruction* operator, which consists of redistributing customers from the shortest route to the other routes. Such operators are widely used in the literature, for example, in Polacek et al. (2004), Pisinger and Ropke (2007), Subramanian et al. (2010, 2013), Vidal et al. (2012, 2013a, 2014), Bezerra et al. (2018), and Christiaens and Vanden Berghe (2020). They are described below.

Intra-route operators

The Intra-route operators are described as:

(a) $\mathcal{N}_1$ (*Swap*): it consists of swapping two customers in the same route;
(b) $\mathcal{N}_2$ (*Reinsertion*): A customer is removed and reinserted in another position in the same route;
(c) $\mathcal{N}_3$ (*Or-opt2*): two consecutive customers are removed and reinserted in another position in the same route;
(d) $\mathcal{N}_4$ (*2-Opt*): two non-adjacent edges are deleted and two others are added to generate a new route;
(e) $\mathcal{N}_5$ (*3-Opt*): three edges are excluded and all possibilities of exchange between them are tested to generate new routes.

Inter-route operators

The Inter-route operators are defined as:

(a) $\mathcal{N}_6$ (*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;
(b) $\mathcal{N}_7$ (*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

The Inter-depots operators are described as:

(a) $\mathcal{N}_8$ (*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;
(b) $\mathcal{N}_9$ (*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;
(c) $\mathcal{N}_{10}$ (*Swap(2, 2)-InterDepot*): it consists of swapping two adjacent customers (v_j, v_{j+1}) from a route r_k with two other adjacent customers (v_t, v_{t+1}) belonging to another route r_l from another depot.

Destruction and reconstruction operator

The Destruction and Reconstruction operator is given by:

(a) $\mathcal{N}_{11}$ (*Eliminates Smallest Route*): It consists of eliminating the shortest route in the solution: the one with the lowest number of customers and reinserting its customers in the other routes.

The set of all local search operators is named $\mathcal{LSO}$, that is, $\mathcal{LSO} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{11}\}$.

4.6. Shake procedure

We use the *Shift(1,0)-InterDepot*, *Swap(1,1)-InterDepot*, and *Eliminates Smallest Route* operators to form the set $\mathcal{PO} = \{\mathcal{N}_8, \mathcal{N}_9, \mathcal{N}_{11}\}$ of shake operators. For the *Shift(1,0)-InterDepot* move, a customer is chosen at random and transferred from the route to another one. In the *Swap(1,1)-InterDepot* move, two customers of different routes are swapped. In the *Eliminates Smallest Route* move, the smallest route is removed, and the customers are assigned to other routes. If it is not

possible to link any of the customers to the routes, a new route is created in the depot with the least number of vehicles. In all three moves, the time window, the vehicle's capacity, and the vehicle's route's maximum duration must be respected.

The shake procedure is presented by Algorithm 3. In this procedure, p represents the level of shake to be applied to the vth perturbation neighborhood of the solution s.

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

1: **for** $(i = 1;\ i \leq p;\ i++)$ **do**
2: Generate a random neighbor $s' \in \mathcal{PO}_v(s)$;
3: $s \leftarrow s'$
4: **end for**
5: **return** s';

4.7. Proposed local search

We propose an adaptive local search method to explore the solution space of the problem. Named ALS, Algorithm 4 shows how it works.

Algorithm 4: ALS(s, s', $\mathcal{N}$, $success$, $itLS$, $maxLS1$)

1: **if** $itLS < maxLS1$ **then**
2: $s'' \leftarrow$ RVND(s, s', $\mathcal{N}$, $success$);
3: **else**
4: $s'' \leftarrow$ SingleLocalSearch(s, s', $\mathcal{N}$, $success$);
5: **end if**
6: **return** s'';

Observe by Algorithm 4 that the perturbed solution s' can be refined either by the RVND method (Souza et al., 2010) (line 2) or by a simple local search (line 4). It is refined by the RVND method only if the proposed algorithm's current iteration is less than *maxLS1*. Thus, the RVND method is applied during *maxLS1* iterations of the SGVNSALS algorithm. The advantage of the RVND method is that it does not require the calibration of the neighborhoods' exploration orders. Like the VND method, it ensures that the returned solution is a local optimum for all the used neighborhoods. It is important to highlight that VND is a deterministic method, as Todosijević et al. (2016), Hansen et al. (2017, 2019), Karakostas et al. (2020b) pointed out, in which the exploration order of the neighborhoods is defined in advance, as shown by Algorithm 6. In the RVND method, on the other hand, the exploration order of the neighborhoods is randomly defined, as line 1 of Algorithm 7 shows.

Algorithm 5: UpdateSuccess(s, s', s'', v, $success$)

1: **if** $f(s'') < f(s)$ **then**
2: $success[v] \leftarrow success[v] + 15$;
3: **else if** $f(s'') < f(s')$ **then**
4: $success[v] \leftarrow success[v] + 5$;
5: **end if**

Algorithm 6: VND(s, s', $\mathcal{N}$)

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

During the application of the RVND (see its pseudocode in Algorithm 7), we store in the *success* variable the neighborhoods that were most successful in this improvement phase. The scores are calculated as follows: if the solution s'' returned by the local search in the neighborhood v improves only the perturbed solution s', that is, if $f(s) < f(s'') < f(s')$, then the *success* variable is increased by 5 units. However, if the local search also improves the best solution s generated so far, that is, $f(s'') < f(s') < f(s)$, then the *success* variable is increased by 15 units. Algorithm 5 shows how the *success* variable is updated when it receives the shake solution s', the improved solution s'' by the local search in neighborhood v, and the best solution s so far. As the local search with the RVND method is computationally costly, after *maxLS1* iterations of the SGVNSALS algorithm, we apply only one local search to the shake solution s'. In this phase, we selected a local search using the roulette method based on each neighborhood's success variable during the application phase of the RVND method. All neighborhoods have a chance to be chosen; however, those with the highest values for the *success* variable have a higher chance. Algorithm 8 shows how the improvement phase works in this second phase of local search. We affirm that our local search is adaptive because it learns from the application of the first phase based on the RVND method.

Algorithm 7: RVND(s, s', $\mathcal{N}$, $success$)

1: $\mathcal{N}' \leftarrow rand(\mathcal{N})$;
2: $v \leftarrow 1$;
3: **while** $v \leq |\mathcal{N}'|$ **do**
4: $s'' \leftarrow \arg\min_{z \in \mathcal{N}'_v(s')} f(z)$; {Best neighbor of $\mathcal{N}'_v(s')$}
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: UpdateSuccess(s, s', s'', v, $success$);
12: **end while**
13: **return** s';

Algorithm 8: SingleLocalSearch(s, s', $\mathcal{N}$, $success$)

1: $v \leftarrow Roullette(success)$;
2: $s'' \leftarrow \arg\min_{z \in \mathcal{N}_v(s')} f(z)$; {Best neighbor of $\mathcal{N}_v(s')$}
3: UpdateSuccess(s, s', s'', v, $success$);
4: **return** s'';

4.8. Proposed algorithms

Three algorithms are implemented here for solving MDVRPTW*. The Smart General Variable Neighborhood Search with Adaptive Local Search (SGVNSALS) algorithm is the method we proposed in this article for solving this problem. For performing a comparison with it, we include an implementation of the Smart General Variable Neighborhood Search (SGVNS) method, introduced in Rego and Souza (2019). However, for conciseness purposes, we presented an unified pseudo-code, addressing the SGVNSALS and SGVNS algorithms jointly. Algorithm 9 introduces this pseudo-code. The value of the variable als at line 12 defines which algorithm will be performed for solving the problem. If it is active (als value is true), the ALS search is triggered, and SGVNSALS is executed (line 13); otherwise, the VND search is triggered, and SGVNS is run (line 15). The third algorithm implemented is the General Variable Neighborhood Search (GVNS) method (Hansen et al., 2017, 2019), which is the most classic variant of VNS, used here as a benchmark for evaluating the behavior of the SGVNSALS proposed method. Algorithm 10 shows the GVNS procedure implemented.

Algorithm 9 presents the pseudocode of the SGVNSALS method proposed for solving MDVRPTW*. In lines 1 and 2, an initial solution is built according to Section 4.3. Lines 4 to 7 set the initial shaking level, the index for the first neighborhood, the iteration counter without improvement, and the iteration counter for applying the local search

Algorithm 9: SGVNSALS($\mathcal{C}$, $\mathcal{D}$, $\mathcal{N}$, $\mathcal{PO}$, $iterMax$, $maxTime$, $maxLevel$, $pLS1$, $pLS2$, als)

```
1: HD ← CreateClusters(C, D);
2: s ← CreateRoutes(HD, C, D);
3: success ← InitializeSucess();
4: p ← 1;                          {Initial shake level}
5: v ← 1;                          {Index for the first neighborhood}
6: iter ← 0;                       {Iteration counter without improvement}
7: itLS ← 0;                       {Counter for the local search}
8: maxLS1 ← pLS1 · iterMax; {Number of the RVND iterations}
9: maxLS2 ← (pLS1 + pLS2) · iterMax; {Number of the SingleLocalSearch iterations}
10: while iter < iterMax and time < maxTime do
11:    s' ← Shake(s, PO, v, p);
12:    if als then
13:       s'' ← ALS(s, s', N, success, itLS, maxLS1);
14:    else
15:       s'' ← VND(s, s', N);
16:    end if
17:    if f(s'') < f(s) then
18:       s ← s'';
19:       v ← 1;                          {Return to the first neighborhood}
20:       p ← 1;                          {Return to the first shake level}
21:       iter ← 0                        {Reset the iteration counter without improvement};
22:    else
23:       p ← p + 1;                      {Increase the shake level}
24:       iter ← iter + 1                 {Increase the iteration counter without improvement};
25:    end if
26:    if p > maxLevel then
27:       v ← v + 1;                      {Move to the next neighborhood}
28:       p ← 1                           {Return to the first shake level};
29:    end if
30:    if v > |PO| then
31:       v ← 1;          {Return to the first neighborhood}
32:       p ← 1;          {Return to the first shake level}
33:    end if
34:    itLS ← itLS + 1;
35:    if itLS ≥ maxLS2 then
36:       itLS ← 0;
37:    end if
38: end while
39: return s;
```

Algorithm 10: GVNS($\mathcal{C}$, $\mathcal{D}$, $\mathcal{N}$, $\mathcal{PO}$, $iterMax$, $maxTime$)

```
1: HD ← CreateClusters(C, D);
2: s ← CreateRoutes(HD, C, D);
3: v ← 1;                          {Index for the first neighborhood}
4: iter ← 0;                       {Iteration counter without improvement}
5: while iter < iterMax and time < maxTime do
6:    Generate a random neighbor s' ∈ PO_v(s);
7:    s'' ← VND(s, s', N);
8:    if f(s'') < f(s) then
9:       s ← s'';
10:      v ← 1;                          {Return to the first neighborhood}
11:      iter ← 0                        {Reset the iteration counter without improvement};
12:   else
13:      iter ← iter + 1                 {Increase the iteration counter without improvement};
14:   end if
15:   if v > |PO| then
16:      v ← 1;          {Return to the first neighborhood}
17:   end if
18: end while
19: return s;
```

procedure, respectively. On lines 8 and 9, we set the number of RVND and ALS iterations, respectively. On lines 10 to 38, the algorithm starts a loop that finishes when the number of iterations without improvement reaches its maximum value or when the runtime meets its maximum duration. In line 11, the solution s is perturbed by applying p random moves in the vth neighborhood $\mathcal{PO}$, as shown in Section 4.6.

Between lines 12 and 16, we apply the local search procedure described in Section 4.7. Then, if the improved solution s'' is better than the current solution, we update the current solution, return to the first neighborhood and to the first shake level, and reset the iteration counter without improvement in lines 18 to 21, respectively. Otherwise, we increase the shake level and the iteration counter in lines 23 to 24, respectively. We move to the next neighborhood only if the shake level meets its maximum level, i.e., *maxLevel*. When it occurs, we reset the shake level to its minimum value, i.e., $p = 1$. When we reach the last shake neighborhood, we return to the first shake neighborhood and the first shake level in lines 31 and 32, respectively.

In line 34, we increment *itLS*, that is, the number of applications of the local search procedure. In the next line, if this number exceeds *maxLS2* iterations, then *itLS* is restarted. In this way, we apply a new round of *maxLS1* iterations with the RVND method and (*maxLS2* - *maxLS1*) iterations using the SingleLocalSearch method, both described in Section 4.7. This cycle is repeated during the execution of the SGVNSALS, alternating between RVND and SingleLocalSearch procedures as local search methods. The proposed algorithm returns, in line 39, the best solution s found during the search.

Regarding the GVNS procedure shown in Algorithm 10, the performed implementation corresponds to removing, from Algorithm 9, the adaptation and intelligent neighborhood choice mechanisms. Thus, it becomes a standard implementation of this classic procedure for the solution of MDVRPTW*.

5. Computational results

The SGVNSALS, SGVNS and GVNS algorithms, shown in Algorithms 9 and 10, were 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.

Two sets of instances, named Group I and Group II, were used as benchmark in this article. They were used in Cordeau et al. (2001a) and Vidal et al. (2013a), respectively, and are available at http://neo.lcc.uma.es/vrp/vrp-instances/ and https://w1.cirrelt.ca/~vidalt/en/VRP-resources.html, respectively. Both groups have a homogeneous fleet per depot. Their characteristics are described in Tables 8 and 12, respectively. In these tables, the first five columns refer to the number of customers, depots, vehicles per depot, and the total of vehicles available, respectively. Group I contains 20 instances involving 48 to 288 customers, 4 to 6 depots, vehicles with capacity ranging from 150 to 200 and route duration ranging from 400 to 500. The first ten instances of this group have narrow time windows, while the last ten ones have wide windows. The 28 instances presented in Group II contain 360 to 960 customers, 4 to 12 depots, and vehicles available per depot between 4 and 26.

This section is organized as follows. Section 5.1 shows the procedures for tuning the parameters. Sections 5.2 and 5.3 show the computational results concerning the application of Algorithms 9 and 10 to Groups I and II of instances, respectively. Section 5.4 presents a study about the influence of the local search and shake operators on the found results.

5.1. Parameter tuning

We used the Iterated Racing for Automatic Algorithm Configuration (IRACE) (López-Ibáñez et al., 2016) for tuning the SGVNSALS algorithm parameters.

Initially, we selected 30% of the Group I instances to calibrate the parameters of the proposed algorithm. These instances were chosen because they contain the different characteristics of all instances of the group. Table 3 shows the six chosen instances, ordered by the values of $\rho = |\mathcal{C}| \times |\mathcal{K}_d| \times |D|$ and grouped by tight and wide time windows, respectively.

Table 4 shows the proposed algorithm's parameters, the values tested, and those returned by IRACE after tuning. The parameter *iterMax* was fixed at the maximum value established because, in empirical tests, values above this value required high times with insignificant improvements in the results. The value ranges tested for the *pLS1* and *pLS2* parameters values were selected from empirical observations,

Table 3
Characteristics of the instances used for tuning with IRACE.

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

in which the best results were obtained between 20% and 40% of *iterMax*. For each value of the maximum number of iterations without improvement, the percentages were verified, and, as values returned by IRACE, *pLS1* and *pLS2* resulted in 30% of the best value of *iterMax*.

5.2. Group I results

Table 5 reports the results achieved by the SGVNSALS, SGVNS, and GVNS algorithms when solving MDVRPTW* for the Group I of instances. For each algorithm, Columns $|\mathcal{U}|$ and "Distance" show the best results found by the algorithms in 30 runs with 60 minutes per instance. Column "Time" reports the average runtime found by these algorithms in minutes. Line "Percent" presents the difference between the results, considering SGNVSALS as a baseline.

For Group I, when comparing the three algorithms, this table shows that SGVNSALS behaved better than SGVNS and GVNS in the three items listed. It showed a reduction of 13.99% in the number of available vehicles, against a reduction of 13.10% obtained by SGVNS and 13.39 obtained by GVNS, which used 3 and 2 more vehicles, respectively, than SGVNSALS. Furthermore, when using SGVNS, there was an increase of 0.76% in the total traveled distance and 8.59% in the runtime; concerning GVNS, there was an increase of 4.61% in the total traveled distance and 32.85 in the runtime.

The three algorithms tied in the number of used vehicles in 14 out of the 20 instances analyzed. In 8 out of these 14 tied, however, SGVNSALS presented better results concerning the covered distance; SGVNS won in 4 from these 14; GVNS won in the two remaining instances. Besides, in 7 of these 14 tied instances, SGVNSALS presented better results about runtime; SGVNS won in the other 3, and GVNS won in the four remainings. Additionally, in 3 instances, SGVNSALS overcame the others algorithms about the number of used vehicles; GVNS overcame the others in 2 instances regarding this item; finally, in 1 instance, SGVNSALS and SGVNS tied in the number of used vehicles and overcame GVNS.

The statistical analysis of these results involving the three methods was performed concerning the number of used vehicles and the covered distance. For normalizing the data, the gap between the respective medians and the results shown in Table 5 was considered, in the form:

$$Gap_{vehicle} = \frac{Median_{vehicle} - Number\ of\ used\ vehicles}{Median_{vehicle}} \quad (16)$$

$$Gap_{distance} = \frac{Median_{distance} - Distance}{Median_{distance}} \quad (17)$$

Table 6 shows the p-values obtained for the hypotheses tests concerning randomness, normality, and homoscedasticity.

From the Durbin–Watson and Fligner-Killeen tests, the randomness and homoscedasticity of the data were guaranteed. From the Shapiro–Wilk test, as the normality of the data was not guaranteed, we used the Kruskal–Wallis hypothesis test. The null hypothesis (H_0) considers there are no statistical differences between the results found by the methods and, as an alternative hypothesis (H_1), there is a difference between the found results. Regarding used vehicles, from SGVNSALS × SGVNS and SGVNSALS × GVNS relations, with 95% confidence, there is evidence to refute H_0, i.e., these statistical tests showed there are differences in the found results by the methods concerning the number of used vehicles.

However, when analyzing the p-values for the covered distance, for the case SGVNSALS × SGVNS, there is a strong relationship between the found results and, therefore, there is statistical evidence to accept H_0 and no statistical difference was detected between the results from these methods. For the case SGVNSALS × GVNS, with 95% confidence, there is evidence to refute H_0 and, thus, to accept H_1 and conclude that there are differences in the results about the covered distance.

To deepen the comparison between the three methods, we performed a statistical analysis involving only the instances in which there were ties in the number of used vehicles. Table 7 presents the found results, involving the evaluation regarding the $Gap_{distance}$, for a confidence level of 5%. These results showed that for SGVNSALS × SGVNS and from the Kruskal–Wallis test, there was no statistical evidence demonstrating differences in the values of covered distances. On the other hand, for SGVNSALS × GVNS, the Kruskal–Wallis test allowed us to conclude that there are statistical differences between the results for covered distances.

Table 8 reports the values found by SGVNSALS in comparison with the best-known results in the literature for the Group I of instances, associated with the results for the HGSADC algorithm, described in Vidal et al. (2013a), for solving the classic MDVRPTW. In the literature, we did not find any work that addresses the minimization in the number of used vehicles, and, in this sense, this reference material was presented here only to report the traveled distance. As we described in Section 2, MDVRPTW* prioritizes the minimization of the number of used vehicles ($\mathcal{U}$) and, secondarily, the minimization of the traveled distance (Column *Distance*). Columns six and seven of this table report the total traveled distance and the average runtime, in minutes, respectively, found by the HGSADC algorithm. As in Table 5, Columns $|\mathcal{U}|$ and "Distance" record the best results found by the SGVNSALS algorithm in 30 runs in 60 minutes per instance. The "Time" column records the average execution time. The "Adjusted time" column shows the adjusted runtime of the SGVNSALS algorithm. The adopted procedure for obtaining this adjusted runtime will be explained next. The penultimate column shows the difference between the numbers of used vehicles and available vehicles in each instance. Finally, the last column records the percentage variation in the used vehicle number concerning the HGSADC algorithm, i.e., concerning the fleet size.

As HGSADC and SGVNSALS algorithms were executed in different machines, we determine the Single Thread Rating of the processor for each computer used and apply the Passmark software (https://www.cpubenchmark.net/) for allowing a fairer comparison between the results achieved from them. According to this software, the Single Thread Rating values for the processors used by the HGSADC and SGVNSALS algorithms are 1426 and 1098, respectively. Hence, as the SGVNSALS algorithm was tested on a slower machine, we use a correction factor ψ_1, given by $\psi_1 = 1098/1426 = 0.77$ to adjust its runtime. Thus, the time contained in the "Adjusted time" column records this adjusted runtime in minutes.

From the results of Table 8, it is possible to notice a reduction of approximately 14% in the total number of used vehicles, i.e., 47 used vehicles. This reduction occurs differently for each instance, varying between 0% and 41.67%. The instances with narrow time windows have a greater reduction in the number of used vehicles. In this case, among the ten instances, six present a reduction greater than 20%. On the other hand, the instances with wide time windows have a low reduction in the number of used vehicles, and, in six of the ten instances, it was not possible to obtain a reduction in the number of used vehicles. These characteristics of the achieved solutions show the similarity between MDVRPTW* and the classic VRPTW.

5.3. Group II results

Table 9 shows the results generated by the SGVNSALS, SGVNS, and GVNS algorithms for the Group II of instances. This table maintains the same formatting described for Table 5. Line "Percent" of this table shows the difference between the results, considering SGNVSALS as a baseline.

Table 4
Tested parameter values and results returned by IRACE.

Description	Parameter	Range	Returned value
Maximum number of iterations without improvement	*iterMax*	{300, 400, 500}	500
Maximum level of shake	*maxLevel*	{3, 4, 5, 6, 7}	5
Percentage of *iterMax* used for training	*pLS1*	{0.2, 0.3, 0.4}	0.3
Percentage of *iterMax* used for applying the adaptive local searches	*pLS2*	{0.2, 0.3, 0.4}	0.3

$maxLS1 = pLS1 \cdot iterMax = 0.3 \cdot 500 = 150$ (see line 8 of Algorithm 9)
$maxLS2 = (pLS1 + pLS2) \cdot iterMax = (0.3 + 0.3) \cdot 500 = 300$ (see line 9 of Algorithm 9).

Table 5
Results for MDVRPTW* from the SGVNSALS algorithm versus SGVNS and GVNS algorithms concerning the Group I of instances.

Instances data					SGVNSALS			SGVNS			GVNS		
Name	$\|C\|$	$\|D\|$	$\|\mathcal{K}_d\|$	$\|\mathcal{K}\|$	$\|\mathcal{U}'\|$	Distance	Time	$\|\mathcal{U}'\|$	Distance	Time	$\|\mathcal{U}'\|$	Distance	Time
pr01	48	4	2	8	5	1 190.00	0.26	5	1 198.82	0.40	5	1 197.91	0.35
pr02	96	4	3	12	9	2 082.10	1.48	9	2 029.35	2.53	9	2 021.23	2.75
pr03	144	4	4	16	12	2 725.56	3.79	13	2 676.56	5.42	13	2 758.97	6.64
pr04	192	4	5	20	17	3 558.48	8.20	17	3 816.50	8.66	16	3 848.25	9.51
pr05	240	4	6	24	21	3 789.84	17.58	21	3 769.81	12.91	21	3 899.95	16.37
pr06	288	4	7	28	26	4 651.68	28.93	26	4 897.11	26.91	26	4 929.69	60.00
pr07	72	6	2	12	7	1 527.00	0.68	7	1 639.05	1.18	7	1 734.83	0.53
pr08	144	6	3	18	12	2 511.62	4.41	12	2 452.78	6.96	12	2 732.75	7.42
pr09	216	6	4	24	17	3 431.24	11.04	18	3 287.81	15.07	18	3 389.11	23.79
pr10	288	6	5	30	25	4 671.04	22.66	25	4 580.05	17.38	26	4 532.35	30.94
pr11	48	4	1	4	4	1 032.98	0.19	4	1 012.47	4.08	4	1 005.73	0.44
pr12	96	4	2	8	8	1 602.65	2.04	8	1 592.03	10.82	8	1 664.28	3.87
pr13	144	4	3	12	11	2 274.40	5.21	11	2 280.04	7.74	11	2 865.31	1.98
pr14	192	4	4	16	15	2 721.26	10.25	15	2 745.77	12.99	15	2 793.73	21.78
pr15	240	4	5	20	20	3 171.43	19.23	20	3 280.77	14.40	20	3 325.56	5.92
pr16	288	4	6	24	22	3 808.61	30.09	23	3 950.76	22.21	23	4 079.80	26.55
pr17	72	6	1	6	6	1 253.21	0.59	6	1 255.93	9.33	6	1 254.01	0.50
pr18	144	6	2	12	12	2 040.39	4.49	12	1 976.29	9.36	11	2 348.53	3.57
pr19	216	6	3	18	16	2 681.81	13.72	16	2 658.08	17.20	16	2 825.88	18.35
pr20	288	6	4	24	24	3 864.18	29.86	24	3 905.85	27.60	24	3 899.81	43.96
Sum				336	289	54 589.48	214.70	292	55 005.83	233.15	291	57 107.68	285.21
Percent (%)								+1.04	+0.76	+8.59	+0.69	+4.61	+32.85

Maximum runtime for each instance $maxTime = 60$ min
Runs per instance: 30

Table 6
p-values for hypothesis tests concerning randomness, normality, and homoscedasticity for Group I.

SGVNSALS *versus*	Performance measure	Test			
		Durbin–Watson	Shapiro–Wilk	Fligner-Killeen	Kruskal–Wallis
SGVNS	$Gap_{vehicle}$	1.95	4.67 e−38	0.06	0.02
	$Gap_{distance}$	2.02	5.01 e−18	0.29	0.97
GVNS	$Gap_{vehicle}$	1.95	4.32 e−31	0.19	2.34 e−19
	$Gap_{distance}$	1.94	5.26 e−15	0.18	9.96 e−26

Table 7
p-values for hypothesis tests concerning randomness, normality, and homoscedasticity for Group I, considering only the situations in which there was a tie in the number of used vehicles.

SGVNSALS *versus*	Performance measure	Test			
		Durbin–Watson	Shapiro–Wilk	Fligner-Killeen	Kruskal–Wallis
SGVNS	$Gap_{distance}$	1.98	7.04 e−17	0.76	0.90
GVNS	$Gap_{distance}$	2.01	4.69 e−14	0.15	7.44 e−16

The results show that SGVNSALS overcame SGVNS and GVNS in the three evaluated items. From the total fleet (1960 vehicles), SGVNSALS provided a reduction of 23.32% (457 vehicles) in the number of used vehicles, from a reduction of 22.35% (438 vehicles) obtained by SGVNS and 21.07% (413 vehicles) by GVNS.

Considering the results from SGVNSALS as a reference, there was an increase of 2.80% in the traveled distance found using SGVNS and a 36.25% decrease in runtime; using GVNS, the result for traveled distance was 6.58% higher, and there was a 57.61% decrease in runtime. There were no ties for this evaluated item involving only SGVNS and GVNS. In 13 of the 28 instances, SGVNSALS outperformed the other algorithms concerning the number of used vehicles; in these same 13 instances, the total distance covered is the smallest among those obtained by the three algorithms; SGVNS outperformed the others in 3 instances, and, in the same way, generated the shortest covered distance; and GVNS only outperformed the others in 1 instance. In the remaining 11 instances, there were ties in the number of used vehicles. In these 11 instances where there was a tie, SGVNSALS generated the shortest traveled distance in 6 instances; SGVNS in 4; and GVNS in only one. In a single instance (pr21b), there was a tie in the number of used vehicles between the three algorithms.

Furthermore, there was a tie in the number of used vehicles between SGVNSALS and SGVNS in 9 of these 11 instances; in two others, there was a tie between SGVNSALS and GVNS. There were no ties for this item involving only SGVNS and GVNS. In terms of runtime, in 26 of the 28 instances, GVNS had the shortest runtime; SGVNSALS had the

Table 8
Results for MDVRPTW* from the SGVNSALS algorithm concerning Group I instances compared with the results for MDVRPTW from the HGSADC algorithm.

Instances data					HGSADC		SGVNSALS					
Name	$\|C\|$	$\|D\|$	$\|\mathcal{K}_d\|$	$\|\mathcal{K}\|$	Distance	Time	$\|\mathcal{U}\|$	Distance	Time	Adjusted time	$\|\mathcal{U}\| - \|\mathcal{K}\|$	Percent
pr01	48	4	2	8	1 074.12	0.31	5	1 190.00	0.26	0.20	−3	−37.50%
pr02	96	4	3	12	1 762.21	1.15	9	2 082.10	1.48	1.14	−3	−25.00%
pr03	144	4	4	16	2 373.65	1.75	12	2 725.56	3.79	2.92	−4	−25.00%
pr04	192	4	5	20	2 815.48	5.89	17	3 558.48	8.20	6.31	−3	−15.00%
pr05	240	4	6	24	2 964.65	8.68	21	3 789.84	17.58	13.53	−3	−12.50%
pr06	288	4	7	28	3 588.78	13.43	26	4 651.68	28.93	22.27	−2	−7.14%
pr07	72	6	2	12	1 418.22	0.51	7	1 527.00	0.68	0.52	−5	−41.67%
pr08	144	6	3	18	2 096.73	2.39	12	2 511.62	4.41	3.39	−6	−33.33%
pr09	216	6	4	24	2 712.56	5.20	17	3 431.24	11.04	8.50	−7	−29.17%
pr10	288	6	5	30	3 465.92	15.22	25	4 671.04	22.66	17.44	−5	−16.67%
pr11	48	4	1	4	1 005.73	0.51	4	1 032.98	0.19	0.15	0	0.00%
pr12	96	4	2	8	1 464.5	1.68	8	1 602.65	2.04	1.57	0	0.00%
pr13	144	4	3	12	2 001.83	2.94	11	2 274.40	5.21	4.01	−1	−8.33%
pr14	192	4	4	16	2 195.33	6.55	15	2 721.26	10.25	7.90	−1	−6.25%
pr15	240	4	5	20	2 433.15	12.56	20	3 171.43	19.23	14.81	0	0.00%
pr16	288	4	6	24	2 836.67	15.97	22	3 808.61	30.09	23.17	−2	−8.33%
pr17	72	6	1	6	1 236.24	1.05	6	1 253.21	0.59	0.45	0	0.00%
pr18	144	6	2	12	1 788.18	3.30	12	2 040.39	4.49	3.46	0	0.00%
pr19	216	6	3	18	2 261.08	8.59	16	2 681.81	13.72	10.56	−2	−11.11%
pr20	288	6	4	24	2 993.31	22.18	24	3 864.18	29.86	22.99	0	0.00%
Sum				336	44 488.34	129.86	289	54 589.48	214.68	165.30	−47	
Percent							−13.99	+22.71%		+27.29%	−13.99%	

Maximum runtime for each instance $maxTime = 60$ min.
Adjusted time: Adjusted time to the processor used by the SGVNSALS algorithm.
Runs per instance: 30.

Table 9
Results for MDVRPTW* from the SGVNSALS algorithm versus SGVNS and GVNS algorithms concerning the Group II of instances.

Instance	$\|C\|$	$\|D\|$	$\|\mathcal{K}_d\|$	$\|\mathcal{K}\|$	SGVNSALS			SGVNS			GVNS		
					$\|\mathcal{U}\|$	Distance	Time	$\|\mathcal{U}\|$	Distance	Time	$\|\mathcal{U}\|$	Distance	Time
pr11a	360	4	10	40	34	8 367.96	26.52	34	8 726.24	25.88	36	8 965.56	13.31
pr12a	480	4	13	52	43	10 203.06	79.13	43	10 787.73	54.83	45	11 300.13	19.22
pr13a	600	4	16	64	52	12 050.67	166.16	54	12 861.05	56.68	54	13 161.64	53.30
pr14a	720	4	19	76	63	14 430.54	257.14	64	15 272.39	99.36	66	15 673.55	62.52
pr15a	840	4	22	88	77	17 298.62	266.35	76	16 861.08	155.03	80	18 407.24	91.22
pr16a	960	4	26	104	87	19 526.10	300.00	90	20 129.48	167.49	93	21 590.78	130.60
pr17a	360	6	7	42	33	8 071.20	29.81	32	7 909.23	35.81	33	8 425.72	16.31
pr18a	520	6	10	60	47	10 905.42	67.13	48	11 027.97	45.25	50	11 840.88	29.97
pr19a	700	6	13	78	63	14 193.26	216.85	64	15 267.10	105.01	65	15 564.61	46.69
pr20a	880	6	16	96	75	15 825.91	276.89	76	16 251.23	151.12	78	17 093.54	105.50
pr21a	420	12	4	48	36	8 222.74	29.20	36	8 161.57	77.92	37	8 143.01	75.39
pr22a	600	12	6	72	49	10 648.65	128.15	48	10 438.98	135.31	51	12 185.14	55.44
pr23a	780	12	8	96	64	13 746.12	183.50	64	13 745.38	138.61	65	14 089.12	97.91
pr24a	960	12	10	120	84	16 890.61	242.67	85	17 322.22	147.39	85	17 868.87	96.16
pr11b	360	4	8	32	27	6 068.24	29.91	27	5 959.04	24.34	28	6 280.37	16.97
pr12b	480	4	11	44	36	7 739.38	39.11	36	7 945.88	35.91	35	8 195.71	34.23
pr13b	600	4	14	56	44	9 409.75	95.18	44	9 919.54	77.03	45	9 821.87	49.86
pr14b	720	4	17	68	54	11 283.40	244.13	56	11 910.17	92.98	56	12 753.49	65.50
pr15b	840	4	20	80	66	13 948.48	256.86	66	14 101.22	134.17	67	14 412.66	94.59
pr16b	960	4	23	92	75	15 364.09	290.66	76	15 871.36	170.54	75	15 290.21	134.04
pr17b	360	6	6	36	26	5 839.02	36.87	28	5 995.21	27.23	28	6 328.40	16.52
pr18b	520	6	9	54	39	8 061.21	65.19	40	8 388.09	57.77	40	8 073.65	29.44
pr19b	700	6	12	72	52	10 662.51	166.90	54	11 676.04	69.08	55	11 819.53	56.60
pr20b	880	6	15	90	69	13 743.12	264.70	71	14 309.46	115.42	70	14 739.72	94.74
pr21b	420	12	4	48	30	6 252.96	40.45	30	6 138.25	113.45	30	6 396.91	64.98
pr22b	600	12	6	72	43	8 567.72	72.62	43	9 386.33	104.15	44	9 016.95	69.82
pr23b	780	12	7	84	58	11 495.28	131.89	59	11 643.89	135.59	59	12 329.20	63.13
pr24b	960	12	8	96	77	14 398.34	202.32	78	14 264.83	128.28	77	14 711.38	98.89
Sum				1960	1503	323 214.36	4206.29	1522	332 270.96	2681.63	1547	344 479.84	1782.86
Percent								1.26%	2.80%	−36.25	2.93%	6.58%	−57.61%

Maximum runtime for each instance $maxTime = 300$ min.
Runs per instance: 10.

shortest runtime in the remaining two. The analysis concerning runtime for SGVNS and GVNS clearly indicates a premature convergence of these algorithms and the success of the adaptive and the "smart" procedures of SGVNSALS, a fact demonstrated by the results obtained.

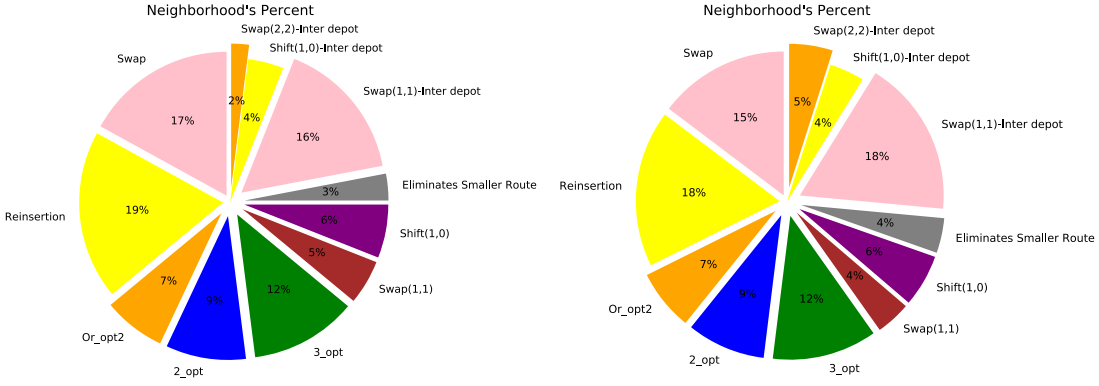
The statistical analysis of the results for Group II of instances regarding the application of SGVNSALS, SGVNS, and GVNS was realized considering the number of used vehicles and the covered distance. Table 10 presents the obtained p-values for the hypotheses tests for randomness, normality, and homoscedasticity. As occurred in the statistical analysis of the results for Group I of instances, the randomness, and homoscedasticity of the data were guaranteed from the Durbin–Watson and Fligner-Killeen tests. The Shapiro–Wilk test showed that

Table 10
p-values for hypothesis tests concerning randomness, normality, and homoscedasticity for Group II.

SGVNSALS *versus*	Performance measure	Test			
		Durbin–Watson	Shapiro–Wilk	Fligner-Killeen	Kruskal–Wallis
SGVNS	$Gap_{vehicle}$	1.95	1.02 e−09	0.01	6.35 e−21
	$Gap_{distance}$	2.08	0.09	1.05 e−06	1.36 e−37
GVNS	$Gap_{vehicle}$	1.92	5.60 e−08	1.43 e−07	1.52 e−57
	$Gap_{distance}$	1.94	3.76 e−05	4.31 e−07	4.40 e−73

Table 11
p-values for hypothesis tests concerning randomness, normality, and homoscedasticity for Group I, considering only the situations in which there was a tie in the number of used vehicles.

SGVNSALS *versus*	Performance measure	Test			
		Durbin–Watson	Shapiro–Wilk	Fligner-Killeen	Kruskal–Wallis
SGVNS	$Gap_{distance}$	2.09	0.02	1.90 e−03	4.01 e−09
GVNS	$Gap_{distance}$	2.21	0.12	0.05	2.90 e−09



(a) Contribution in Group I. (b) Contribution in Group II.

Fig. 5. Percentage of contribution of each local search operator.

the normality of the data was not guaranteed, and, thus, the Kruskal–Wallis hypothesis test was used to identify statistical differences between the results. The null hypothesis (H_0) considers there are no statistical differences between the results found by the methods, and, as an alternative hypothesis (H_1), there is a difference between the found results. Based on the results obtained with the Kruskal–Wallis test, there is statistical evidence to refute the H_0 hypothesis in all cases, which allowed us to conclude that there are statistical differences between the results found by the three methods.

As we have done for Group I of instances, we performed a statistical analysis involving only the instances in which there were ties in the number of used vehicles for Group II of instances. Table 11 describes the found results, evaluating the $Gap_{distance}$, for a confidence level of 5%. These results showed that for SGVNSALS × SGVNS and for SGVNSALS × GVNS, the Kruskal–Wallis test led us to conclude that there are statistical differences between the results for covered distances.

Table 12 compares the results from SGVNSALS with the best-known results in the literature for the Group II of instances, provided for the HGSADC algorithm, described in Vidal et al. (2013b), for solving the classic MDVRPTW. This table maintains the same structure described for Table 8, including the column "Adjusted time", which represents the adjusted runtime due to the difference in the used processors. The results found for this group differ from those for Group I. For all instances, there was a reduction in the number of used vehicles, with the smallest reduction being 12.50% (occurring in instance pr15a) and the biggest reduction being 40.28% (occurring in instance pr22b). The total reduction for the used vehicles was 23.32% (457 vehicles).

There is a greater percentage reduction in the vehicle number concerning the total fleet from Group I to Group II (13.99% against 23.32%). In Group II, however, the group of narrow time windows instances has a greater reduction than that achieved by the group of wide time windows instances. For the narrow time windows instances, four of the 14 instances have a greater reduction than the general average reduction; for the wide time windows instances, eight out of 14 instances have a reduction greater than the average. Hence, for Group II, there is a smaller usage of vehicles for the wide time windows instances than for the tight windows instances.

5.4. Influence of the local search and shake operators

In this section, we analyze the influence of the different local search and shake operators on the performance of the SGVNSALS algorithm. To evaluate the behavior of the local search operators, we compute the score that each one received, as described in Section 4.7. Fig. 5 summarizes the percentage relative to each one obtained from this score. Fig. 5(a) presents the percentages relative to the test results in the instances of Group I. In turn, we show the results of the instances of Group II in Fig. 5(b).

From the results obtained, we can verify that the algorithm's behavior in the two groups of instances is the same. Furthermore, the *Reinsertion*, *Swap*, *Swap(1,1)-Interdepot*, and *3-Opt* local search operators have the most influence to reduce the size of the vehicle fleet in both groups. Together, these operators represent 64% of the total score distributed for Group I and 63% for Group II.

In Section 5.4.1, we analyze the algorithm's behavior when eliminating neighborhood operators with less contribution in the local search procedure. In turn, in Section 5.4.2, we analyze the algorithm's behavior, modifying the shake operators. Finally, in Section 5.4.3, we analyze the algorithm's behavior by eliminating the operators with the smallest contributions both in the local search and in the shake phase.

Table 12
Results for MDVRPTW* from the SGVNSALS algorithm concerning Group II instances compared with the results for MDVRPTW from the HGSADC algorithm.

Instances data					HGSADC		SGVNSALS					
Name	$\|C\|$	$\|D\|$	$\|\mathcal{K}_d\|$	$\|\mathcal{K}\|$	Distance	Time	$\|\mathcal{U}'\|$	Distance	Time	Adjusted time	$\|\mathcal{U}'\| - \|\mathcal{K}\|$	Variation
pr11a	360	4	10	40	6 720.71	16.81	34	8 367.96	26.52	20.42	−6	−15.00%
pr12a	480	4	13	52	8 179.80	30.00	43	10 203.06	79.13	60.93	−9	−17.31%
pr13a	600	4	16	64	9 667.20	54.85	52	12 050.67	166.16	127.94	−12	−18.75%
pr14a	720	4	19	76	11 124.01	65.65	63	14 430.54	257.14	197.99	−13	−17.11%
pr15a	840	4	22	88	13 013.97	132.44	77	17 298.62	266.35	205.09	−11	−12.50%
pr16a	960	4	26	104	14 299.87	133.63	87	19 526.10	300.00	231.00	−17	−16.35%
pr17a	360	6	7	42	6 304.30	17.23	33	8 071.20	29.81	22.95	−9	−21.43%
pr18a	520	6	10	60	8 308.32	44.25	47	10 905.42	67.13	51.69	−13	−21.67%
pr19a	700	6	13	78	10 677.61	74.42	63	14 193.26	216.85	166.97	−15	−19.23%
pr20a	880	6	16	96	11 963.91	107.37	75	15 825.91	276.89	213.20	−21	−21.88%
pr21a	420	12	4	48	6 260.53	28.00	36	8 222.74	29.20	22.48	−12	−25.00%
pr22a	600	12	6	72	7 985.37	76.05	49	10 648.65	128.15	98.67	−23	−31.94%
pr23a	780	12	8	96	9 937.43	137.72	64	13 746.12	183.50	141.29	−32	−33.33%
pr24a	960	12	10	120	11 923.72	197.17	84	16 890.61	242.67	186.85	−36	−30.00%
pr11b	360	4	8	32	4 839.44	18.04	27	6 068.24	29.91	23.03	−5	−15.63%
pr12b	480	4	11	44	6 063.26	29.09	36	7 739.38	39.11	30.11	−8	−18.18%
pr13b	600	4	14	56	7 254.17	70.99	44	9 409.75	95.18	73.29	−12	−21.43%
pr14b	720	4	17	68	8 732.29	98.92	54	11 283.40	244.13	187.98	−14	−20.59%
pr15b	840	4	20	80	10 439.72	129.48	66	13 948.48	256.86	197.78	−14	−17.50%
pr16b	960	4	23	92	11 483.22	170.31	75	15 364.09	290.66	223.80	−17	−18.48%
pr17b	360	6	6	36	4 806.01	15.78	26	5 839.02	36.87	28.39	−10	−27.78%
pr18b	520	6	9	54	6 526.72	39.45	39	8 061.21	65.19	50.20	−15	−27.78%
pr19b	700	6	12	72	8 227.25	80.55	52	10 662.51	166.90	128.51	−20	−27.78%
pr20b	880	6	15	90	10 325.80	150.74	69	13 743.12	264.70	203.82	−21	−23.33%
pr21b	420	12	4	48	4 866.57	36.75	30	6 252.96	40.45	31.15	−18	−37.50%
pr22b	600	12	6	72	6 488.50	73.28	43	8 567.72	72.62	55.92	−29	−40.28%
pr23b	780	12	7	84	8 523.41	163.99	58	11 495.28	131.89	101.55	−26	−30.95%
pr24b	960	12	8	96	10 890.08	298.51	77	14 398.34	202.32	155.78	−19	−19.79%
Sum				1960	245 833.19	2491.47	1503	323 214.36	4206.31	3238.78	−457	
Percent								+31.48%		+29.99%	−23.32%	

Maximum runtime per instance *maxTime*: 300 min
Runs per instance: 10

Due to the similar behavior of the algorithm in the two groups of instances, we chose to execute it only in the instances of Group I. We run the algorithm with the same parameter configuration already established and with 30 executions per instance. We performed the tests on the CEFET-MG cluster, which contains AMD Opteron 6376 processors, 2.30 GHz, and Linux operating system 64 bits, Single Thread Rating equal to 1165 and correction factor $\psi_2 = 1165/1426 = 0.82$ when compared to the reference machine used to calculate ψ_1.

5.4.1. Influence of the operators with less contribution in the local search

In this subsection, we investigate the influence of the operators that have a contribution of less than 5% in Group I results, that is, *Swap(2,2), Eliminates Smaller Route, and Shift(1,0)* local search operators. For this, we run the proposed algorithm eliminating the two and three neighborhood operators that contributed least to the algorithm results. As we are only analyzing the influence of local search operators, we kept the same three shake operators, that is, *Shift(1,0)-InterDepot*, *Swap(1,1)-InterDepot* and *Eliminates Smaller Route*.

Therefore, in addition to the original set with the eleven local search operators described in Section 4.5, named $\mathcal{LSO}_{11}$, we also analyzed the performance of the proposed algorithm using two other sets of neighborhood operators, $\mathcal{LSO}_9$ and $\mathcal{LSO}_8$, as local search procedures. The $\mathcal{LSO}_9$ and $\mathcal{LSO}_8$ sets have the nine and eight most influential local search operators in the proposed algorithm results, respectively. Table 13 presents the neighborhoods that are part of each set.

In Table 13, IR represents the set formed by all intra-route operators $\mathcal{N}_1$ to $\mathcal{N}_5$ described in Section 4.5, that is, $IR = \{$*Swap*, *Reinsertion*, *Or-opt2*, *2-Opt*, *3-Opt*$\}$. In turn, $\mathcal{LSO}_{11}$ is the set of all neighborhood operators described in Section 4.5. $\mathcal{LSO}_9$ is the set of local search operators resulting from the elimination of *Shift(1,0)-InterDepot* and *Swap(2,2)-InterDepot* operators from the $\mathcal{LSO}_{11}$ set. Finally, the $\mathcal{LSO}_8$ set excludes the *Eliminates Smaller Route* operator from $\mathcal{LSO}_9$.

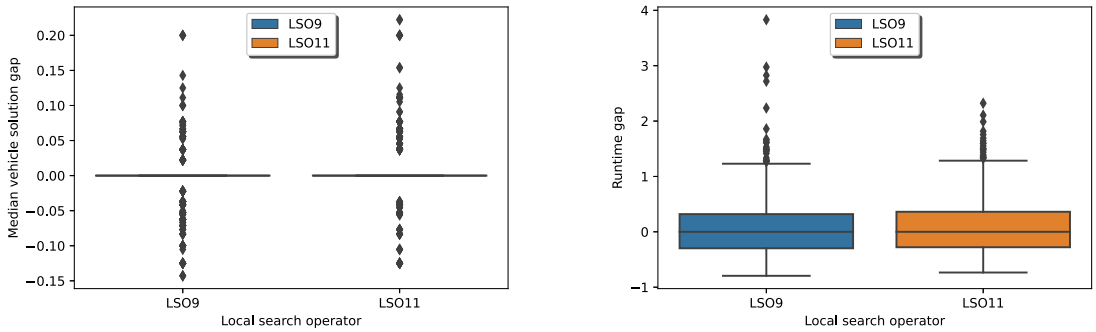
In the $\mathcal{LSO}_8$ set, 6.50% of the solutions were infeasible. Thus, we discard the algorithm results in this set. Table 14 reports the results of the proposed algorithm using the $\mathcal{LSO}_9$ and $\mathcal{LSO}_{11}$ sets of local search operators in Group I instances. In this table, we maintain the same nomenclature defined in Table 8.

Fig. 6 illustrates the boxplots of the SGVNSALS algorithm results in Group I instances using the $\mathcal{LSO}_9$ and $\mathcal{LSO}_{11}$ local search operators. In Figs. 6(a) and 6(b), the ordinate axes represent the gaps of the average solution value concerning vehicles and the runtime in 30 executions of the proposed algorithm in the instances of Group I, respectively.

To verify whether there is a statistical difference of the proposed algorithm in these two sets of local search operators at the significance level of 5%, we consider as a null hypothesis (H_0) the absence of statistical difference between the results and as an alternative hypothesis (H_1) the existence of difference in the results.

As the normality of the samples was not guaranteed, we applied the Kruskal–Wallis test. Table 15 reports the p-values of the tests on the two performance measures analyzed.

Based on the Kruskal–Wallis test, with 95% confidence, there is no statistical evidence to refute the null hypothesis H_0 for the runtime gap as a performance measure. Thus, there is no significant difference in runtime when using the $\mathcal{LSO}_9$ and $\mathcal{LSO}_{11}$ operator sets. On the other hand, regarding the gap in the number of vehicles, the p-value found was equal to $6.43\,e{-}07$. So, with 95% of confidence, there is a statistical difference in the SGVNSALS algorithm results when considering the $\mathcal{LSO}_9$ and $\mathcal{LSO}_{11}$ operator sets. Thus, as the SGVNSALS algorithm with $\mathcal{LSO}_{11}$ operators presents a smaller gap concerning the number of vehicles, it outperforms the version that uses $\mathcal{LSO}_9$ operators.



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

Fig. 6. Boxplots of the SGVNSALS algorithm results in Group I instances using $\mathcal{LSO}_9$ and $\mathcal{LSO}_{11}$ local search operators.

Table 13
Subsets of operators used to analyze the influence of the local search operators.

Set	Inter-Route			Inter-Depot			
	IR $\mathcal{N}_1 - \mathcal{N}_5$	*Shift(1, 0)* $\mathcal{N}_6$	*Swap(1,1)* $\mathcal{N}_7$	*Shift(1, 0)* $\mathcal{N}_8$	*Swap(1,1)* $\mathcal{N}_9$	*Swap(2,2)* $\mathcal{N}_{10}$	*Eliminates Smaller Route* $\mathcal{N}_{11}$
$\mathcal{LSO}_{11}$	✓	✓	✓	✓	✓	✓	✓
$\mathcal{LSO}_9$	✓	✓	✓	✓	✓	–	–
$\mathcal{LSO}_8$	✓	✓	✓	–	✓	–	–

Table 14
Results of the proposed algorithm using $\mathcal{LSO}_9$ and $\mathcal{LSO}_{11}$ sets of local search operators in Group I instances.

Algorithm					HGSADC		SGVNSALS						
Set of local search operators					–		$\mathcal{LSO}_9$				$\mathcal{LSO}_{11}$		
Instance	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{K}_d\rvert$	$\lvert\mathcal{K}\rvert$	Distance	Time	$\lvert\mathcal{U}\rvert$	Distance	Time	Adjusted time	$\lvert\mathcal{U}\rvert$	Distance	Time
pr01	48	4	2	8	1 074.12	0.31	5	1 221.36	0.30	0.24	5	1 190.00	0.20
pr02	96	4	3	12	1 762.21	1.15	9	1 967.90	1.64	1.34	9	2 082.10	1.14
pr03	144	4	4	16	2 373.65	1.75	12	2 631.95	3.72	3.04	12	2 725.56	2.92
pr04	192	4	5	20	2 815.48	5.89	17	3 527.35	6.81	5.56	17	3 558.48	6.31
pr05	240	4	6	24	2 964.65	8.68	21	3 757.40	14.16	11.57	21	3 789.84	13.53
pr06	288	4	7	28	3 588.78	13.43	26	4 619.54	25.33	20.69	26	4 651.68	22.27
pr07	72	6	2	12	1 418.22	0.51	7	1 554.67	0.68	0.55	7	1 527.00	0.52
pr08	144	6	3	18	2 096.73	2.39	12	2 584.57	4.03	3.29	12	2 511.62	3.39
pr09	216	6	4	24	2 712.56	5.20	17	3 637.90	10.50	8.58	17	3 431.24	8.50
pr10	288	6	5	30	3 465.92	15.22	26	4 478.62	17.60	14.38	25	4 671.04	17.44
pr11	48	4	1	4	1 005.73	0.51	4	1 005.73	0.25	0.20	4	1 032.98	0.15
pr12	96	4	2	8	1 464.50	1.68	8	1 631.58	2.14	1.75	8	1 602.65	1.57
pr13	144	4	3	12	2 001.83	2.94	11	2 229.73	3.88	3.17	11	2 274.40	4.01
pr14	192	4	4	16	2 195.33	6.55	15	2 709.21	9.80	8.01	15	2 721.26	7.90
pr15	240	4	5	20	2 433.15	12.56	19	3 293.20	15.42	12.59	20	3 171.43	14.81
pr16	288	4	6	24	2 836.67	15.97	23	3 684.07	21.84	17.84	22	3 808.61	23.17
pr17	72	6	1	6	1 236.24	1.05	6	1 250.64	0.69	0.57	6	1 253.21	0.45
pr18	144	6	2	12	1 788.18	3.30	12	1 942.50	4.42	3.61	12	2 040.39	3.46
pr19	216	6	3	18	2 261.08	8.59	16	2 591.18	12.11	9.90	16	2 681.81	10.56
pr20	288	6	4	24	2 993.31	22.18	24	3 785.40	24.26	19.82	24	3 864.18	22.99
Sum				336	44 488.34	129.86	290	54 104.50		140.70	289	54 589.48	165.30
Percent							−13.69%	21.62%		12.97%	−13.99%	22.71%	27.29%

Maximum runtime per instance (*maxTime*): 60 min
Executions per instance: 30

Table 15
p-values for randomness, normality and homoscedasticity tests.

Performance measure	Test			
	Durbin–Watson	Shapiro–Wilk	Fligner-Killeen	Kruskal–Wallis
$Gap_{vehicle}$	1.90	7.06 e−36	0.85	6.43 e−07
$Gap_{runtime}$	2.03	4.48 e−26	0.99	0.48

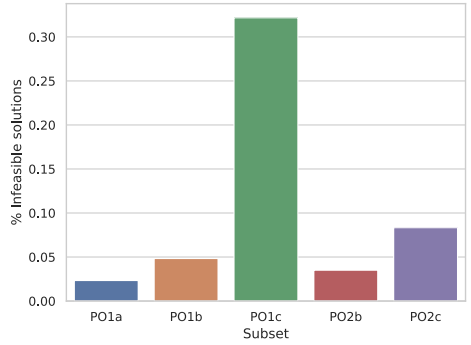
5.4.2. Influence of the shake operators

In this subsection, we analyze the influence of the shake operators, keeping all eleven local search operators used by SGVNSALS. Table 16 lists the subsets formed by all possible combinations of these shake operators. The $\mathcal{PO}_3$ set is listed only to inform the original shake operators used by SGVNSALS. In each subset, we eliminate one or two operators from the $\mathcal{PO}_3$ set.

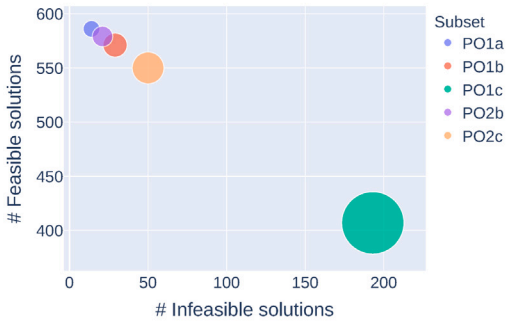
Fig. 7(a) shows the percentage of infeasible solutions when the SGVNSALS algorithm uses the $\mathcal{PO}_{1a}$, $\mathcal{PO}_{1b}$, $\mathcal{PO}_{1c}$, $\mathcal{PO}_{2b}$, and $\mathcal{PO}_{2c}$ subsets as shake operators. In turn, Fig. 7 illustrates, as a bubble

Table 16
Subsets of operators used to analyze the influence of the shake operators.

Subsets	Local search operators	Shake operators		
	$\mathcal{N}_1 - \mathcal{N}_{11}$	$\mathcal{N}_8$: *Shift(1, 0)-InterDepot*	$\mathcal{N}_9$: *Swap(1,1)-InterDepot*	$\mathcal{N}_{11}$: *Eliminates Smaller Route*
$\mathcal{PO}_3$	✓	✓	✓	✓
$\mathcal{PO}_{1a}$	✓	✓	–	–
$\mathcal{PO}_{1b}$	✓	–	✓	–
$\mathcal{PO}_{1c}$	✓	–	–	✓
$\mathcal{PO}_{2a}$	✓	✓	✓	–
$\mathcal{PO}_{2b}$	✓	✓	–	✓
$\mathcal{PO}_{2c}$	✓	–	✓	✓



(a) Percentage of infeasible solutions generated by SGVNSALS when using the $\mathcal{PO}_{1a}$, $\mathcal{PO}_{1b}$, $\mathcal{PO}_{1c}$, $\mathcal{PO}_{2b}$, and $\mathcal{PO}_{2c}$ subsets.



(b) Bubble chart concerning the number of feasible and infeasible solutions found by SGVNSALS using the $\mathcal{PO}_{1a}$, $\mathcal{PO}_{1b}$, $\mathcal{PO}_{1c}$, $\mathcal{PO}_{2b}$, and $\mathcal{PO}_{2c}$ subsets.

Fig. 7. Graphs computing the number of infeasibilities found in $\mathcal{PO}_a - \mathcal{PO}_{2c}$ subsets.

chart, the number of infeasible and feasible solutions generated in 30 executions of the proposed algorithm in the 20 instances with these subsets of shake operators.

Fig. 7(a) shows that the proposed algorithm has the worst performance in the $\mathcal{PO}_{1c}$ subset, with more than 30% of infeasible solutions generated in the tests performed. These results show that the combination of the *Shift(1, 0)-InterDepot* and *Swap(1,1)-InterDepot* shake operators provide the necessary diversity for the SGVNSALS algorithm in the search for feasible solutions for the MDVRPTW*. Using the shake operators described in the $\mathcal{PO}_{2a}$ subset, all solutions generated by the SGVNSALS algorithm were feasible. Therefore, we report in Table 17 only the results generated by the proposed algorithm with this subset of shake operators. The nomenclature is the same as used previously.

5.4.3. Influence of simultaneous elimination of low-contribution operators

In this subsection, we analyze the influence of eliminating the local search and shake operators with a low contribution in the results.

Table 18 presents the configurations of the local search and shake operators, eliminating the operators *Shift(1,0)-InterDepot* and *Eliminates Smaller Route*, which had a contribution smaller than 5% according to Fig. 5(a).

Fig. 8 shows the bubble chart relative to the number of feasible and infeasible solutions generated by the proposed algorithm using the local search and shake operators marked in Table 18.

As we can see in Fig. 8, the proposed algorithm with these local search and shake generates infeasible solutions in all cases. This result shows that these operators are also essential for the search process of the proposed algorithm.

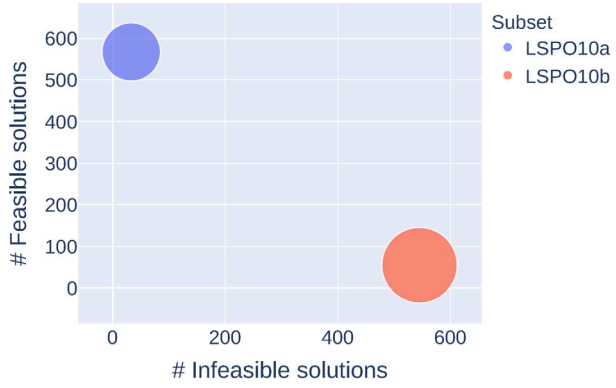


Fig. 8. Bubble chart concerning the number of feasible and infeasible solutions found by the proposed algorithm using the operators defined by the $\mathcal{LSPO}_{10a} - \mathcal{LSPO}_{10b}$ subsets.

5.5. Influence of the α and β parameters of the evaluation function (12)

This section evaluates the influence of parameters α, which weights the total number of used vehicles, and β, which weights the total traveled distance, of the evaluation function (12), on the results found by the SGVNSALS algorithm in the solution of the proposed MDVRPTW*. As stated in Section 4.4, the defined values for these parameters for the results shown in Tables 5–17 imply minimizing the total number of used vehicles in the solution. Thus, the value of parameter α is defined by Expression (12), and parameter β is set to $\beta = 0.001$.

Tables 19 and 20 show the results obtained when setting $\alpha = 0$ and $\beta = 1$. For these values, the minimization of the total number of used vehicles is disregarded, and the evaluation function leads only to the minimization of the total traveled distance. In this sense, the MDVRPTW* problem is no longer addressed, and the shown solutions, therefore, refer to the classical MDVRPTW. In these tables, columns 1–5 define the addressed instance data. Columns 6–7 reproduce the solutions found by the HGSADC algorithm, placed in Vidal et al. (2013b), for Groups I and II of instances. The remaining (columns 8–11) present the found results from the performed computational experiments for $\alpha = 0$ and $\beta = 1$ regarding the SGNVSALS algorithm.

Table 17
SGVNSALS results with the $\mathcal{PO}_{2a}$ subset.

Instance	HGSADC			$\mathcal{PO}_{2a}$			
	$\lvert\mathcal{K}\rvert$	Distance	Time	$\lvert\mathcal{U}\rvert$	Distance	Time	Adjusted time
pr01	8	1 074.12	0.31	5	1 223.19	0.39	0.32
pr02	12	1 762.21	1.15	9	2 053.28	2.52	2.06
pr03	16	2 373.65	1.75	12	2 733.25	5.63	4.60
pr04	20	2 815.48	5.89	17	3 515.26	14.82	12.11
pr05	24	2 964.65	8.68	21	3 722.81	23.86	19.49
pr06	28	3 588.78	13.43	26	4 764.20	39.41	32.20
pr07	12	1 418.22	0.51	7	1 605.97	1.12	0.92
pr08	18	2 096.73	2.39	12	2 443.41	6.92	5.66
pr09	24	2 712.56	5.20	17	3 310.29	16.58	13.54
pr10	30	3 465.92	15.22	25	4 661.87	34.75	28.39
pr11	4	1 005.73	0.51	4	1 005.73	0.43	0.35
pr12	8	1 464.50	1.68	8	1 618.91	3.01	2.46
pr13	12	2 001.83	2.94	11	2 432.16	6.10	4.98
pr14	16	2 195.33	6.55	14	2 736.13	14.14	11.55
pr15	20	2 433.15	12.56	20	3 168.53	24.51	20.02
pr16	24	2 836.67	15.97	23	3 631.15	37.99	31.04
pr17	6	1 236.24	1.05	6	1 262.39	1.10	0.90
pr18	12	1 788.18	3.30	12	2 063.66	7.75	6.33
pr19	18	2 261.08	8.59	16	2 760.53	20.05	16.38
pr20	24	2 993.31	22.18	24	3 801.72	35.80	29.24
Sum	336	44 488.34	129.86	289	54 514.44	296.88	242.54
Percentage				−13.99%	22.54%	128.61%	86.77%

Table 18
Subsets of operators used to analyze the influence of the Local search and Shake operators.

Subset	Local search operators							Shake operators		
	$\mathcal{N}_1 - \mathcal{N}_5$	$\mathcal{N}_6$	$\mathcal{N}_7$	$\mathcal{N}_8$	$\mathcal{N}_9$	$\mathcal{N}_{10}$	$\mathcal{N}_{11}$	$\mathcal{N}_8$	$\mathcal{N}_9$	$\mathcal{N}_{11}$
$\mathcal{LSPO}_{10a}$	✓	✓	✓	✓	✓	✓	–	✓	✓	–
$\mathcal{LSPO}_{10b}$	✓	✓	✓	–	✓	✓	✓	–	✓	✓

From these tables, we conclude that the SGVNSALS algorithm, as a metaheuristic procedure, is tightly coupled to the MDVRPTW* solution and, therefore, does not behave as a procedure aimed at determining quality solutions for the classical MDVRPTW. Only in the pr11 instance of Group I did the SGVNSALS algorithm obtain a total distance traveled equal to that found by the HGSADC algorithm, using the total number of vehicles in the fleet, however with a higher computational time, after the adjustment caused by the difference between the processors.

Another feature of the solution found by SGVNSALS is, despite the values of α and β, the savings in the number of vehicles used concerning the available fleet in 9 of the 20 instances for Group I and in all of Group II. Once again, it shows the strict adherence of the procedure to the MDVRPTW* solution because, even with the unfavorable weighting in the evaluation function, it still determines results closer to the MDVRPTW* than to the classic MDVRPTW. Except for instance pr11 already mentioned, in all other instances of Groups I and II, the total distance covered is higher than the values determined by the HGSADC algorithm.

Another interesting comparison is between the values shown in Tables 8 and 9 ($\alpha = 0$ and β) and those found in Tables 19 and 20 ($\alpha = 0$ and $\beta = 1$). For Group I (Tables 8 and 19), in 5 of the 20 instances, there is a tie in the total number of used vehicles; in the others, the number of used vehicles shown in Table 19 is higher than that shown in Table 8. In four of these five instances, the values of total traveled distance in Table 19 are lower than those shown in Table 8, i.e., in terms of the MDVRPTW*, these solutions have better quality than those shown in Table 8. Once again, this fact proves the strict adherence of the SGVNSALS algorithm to the MDVRPTW* solution. For Group II (Tables 9 and 20), in five of the 28 instances, there is a tie in the total number of used vehicles; among these five, in two of these, the total distance covered in Table 20 is lower than that shown in Table 9. Again, these two solutions have better quality than the associated ones in Table 9.

6. General discussions and conclusions

This article addressed the Multi-Depot Vehicle Routing Problem with Time Windows and the minimization of the number of used vehicles. We called this problem as MDVRPTW* once this is a variation of the classical MDVRPTW, which addresses only the minimization of the total distance traveled. As this problem is NP-hard, we developed an algorithm named Smart General Variable Neighborhood Search with Adaptive Local Search (SGVNSALS) for solving it. The proposed algorithm used two different local search strategies, which were applied alternately. The local search was performed with the Randomized Variable Neighborhood Descent (RVND) method in the first strategy, using 11 classic moves from the literature. When applying this strategy, most successful neighborhoods received a higher score. The second strategy refined a solution in a single neighborhood, chosen by a roulette method. In this way, neighborhoods with the highest score were more likely to be selected. Thus, the first local search strategy served as a learning method for the second strategy. In the SGVNSALS algorithm, the shake level was increased only after a maximum number of iterations without improving the current solution. Consequently, the algorithm more adequately explored the region of the solution space in which the current solution was located.

To test the algorithm, we used 48 benchmark instances of MDVRPTW involving up to 960 customers, 12 depots, and 120 vehicles. We compared SGVNSALS with the classic variant GVNS, in which the Variable Neighborhood Descent (VND) algorithm is applied as the local search, and with the SGVNS algorithm. When comparing the results found by these three methods concerning the number of used vehicles, there was a tie in 54.17% of the instances, and SGVNSALS had a superior performance in 33.33% of the evaluated cases. According to the definition of MDVRPTW*, the second criteria to define the solution quality is the covered distance. Regarding the instances with the tie in the number of used vehicles, SGVNSALS obtained better results in 53.85% of the instances. Therefore, this evaluation allowed us to conclude that the SGVNSALS algorithm had better behavior than the SGVNS and the GVNS algorithms for solving the benchmark instances. The main reason for this superiority is the nonpremature convergence provided by the use of the adaptive and the "*smart*" procedures. The

Table 19
Results from the SGVNSALS algorithm for $\alpha = 0$ and $\beta = 1$, for Group I instances, compared with the results from the HGSADC algorithm.

Instances data					HGSADC		SGVNSALS			
Name	$\|C\|$	$\|D\|$	$\|\mathcal{K}_d\|$	$\|\mathcal{K}\|$	Distance	Time	$\|\mathcal{U}\|$	Distance	Time	Adjusted time
pr01	48	4	2	8	1 074.12	0.31	7	1 149.41	0.43	0.33
pr02	96	4	3	12	1 762.21	1.15	11	1 945.82	2.15	1.65
pr03	144	4	4	16	2 373.65	1.75	13	2 748.80	4.39	3.38
pr04	192	4	5	20	2 815.48	5.89	18	3 554.20	9.56	7.36
pr05	240	4	6	24	2 964.65	8.68	23	3 842.26	18.59	14.32
pr06	288	4	7	28	3 588.78	13.43	28	4 750.90	29.47	22.69
pr07	72	6	2	12	1 418.22	0.51	9	1 511.92	1.08	0.83
pr08	144	6	3	18	2 096.73	2.39	15	2 468.18	5.63	4.34
pr09	216	6	4	24	2 712.56	5.20	20	3 381.65	13.44	10.35
pr10	288	6	5	30	3 465.92	15.22	29	4 757.72	24.58	18.93
pr11	48	4	1	4	1 005.73	0.51	4	1 005.73	3.56	2.74
pr12	96	4	2	8	1 464.50	1.68	8	1 575.79	9.17	7.06
pr13	144	4	3	12	2 001.83	2.94	12	2 175.18	13.61	10.48
pr14	192	4	4	16	2 195.33	6.55	16	2 703.54	16.55	12.75
pr15	240	4	5	20	2 433.15	12.56	20	3 122.11	17.23	13.27
pr16	288	4	6	24	2 836.67	15.97	24	3 776.93	31.94	24.60
pr17	72	6	1	6	1 236.24	1.05	6	1 250.56	4.67	3.59
pr18	144	6	2	12	1 788.18	3.30	12	1 954.29	11.70	9.01
pr19	216	6	3	18	2 261.08	8.59	18	2 645.77	17.30	13.32
pr20	288	6	4	24	2 993.31	22.18	24	3 900.87	23.97	18.46
Sum				336	44 488.34	129.86	317	54 221.63	259.02	199.44

Maximum runtime for each instance $maxTime = 60$ min.
Runs per instance: 30.

Table 20
Results from the SGVNSALS algorithm for $\alpha = 0$ and $\beta = 1$, for Group II instances, compared with the results from the HGSADC algorithm.

Instances data					HGSADC		SGVNSALS			
Name	$\|C\|$	$\|D\|$	$\|\mathcal{K}_d\|$	$\|\mathcal{K}\|$	Distance	Time	$\|\mathcal{U}\|$	Distance	Time	Adjusted time
pr11a	360	4	10	40	6 720.71	16.81	35	8 666.96	41.48	31.94
pr12a	480	4	13	52	8 179.80	30.00	44	10 657.16	80.71	62.15
pr13a	600	4	16	64	9 667.20	54.85	57	12 806.52	224.13	172.58
pr14a	720	4	19	76	11 124.01	65.65	69	15 148.58	248.01	190.97
pr15a	840	4	22	88	13 013.97	132.44	81	17 786.38	260.32	200.45
pr16a	960	4	26	104	14 299.87	133.63	93	19 905.54	298.69	229.99
pr17a	360	6	7	42	6 304.30	17.23	36	8 303.79	37.02	28.51
pr18a	520	6	10	60	8 308.32	44.25	47	11 133.98	123.72	95.27
pr19a	700	6	13	78	10 677.61	74.42	67	15 050.56	226.66	174.53
pr20a	880	6	16	96	11 963.91	107.37	79	16 751.01	261.67	201.49
pr21a	420	12	4	48	6 260.53	28.00	36	8 198.88	75.07	57.80
pr22a	600	12	6	72	7 985.37	76.05	50	11 367.75	104.05	80.12
pr23a	780	12	8	96	9 937.43	137.72	68	14 047.48	228.94	176.28
pr24a	960	12	10	120	11 923.72	197.17	86	16 938.92	273.98	210.96
pr11b	360	4	8	32	4 839.44	18.04	28	6 111.42	53.12	40.90
pr12b	480	4	11	44	6 063.26	29.09	36	8 000.16	102.69	79.07
pr13b	600	4	14	56	7 254.17	70.99	44	9 483.42	187.45	144.34
pr14b	720	4	17	68	8 732.29	98.92	55	11 658.84	248.26	191.16
pr15b	840	4	20	80	10 439.72	129.48	69	14 005.21	233.27	179.62
pr16b	960	4	23	92	11 483.22	170.31	77	15 589.62	293.73	226.17
pr17b	360	6	6	36	4 806.01	15.78	28	6 211.66	29.04	22.36
pr18b	520	6	9	54	6 526.72	39.45	40	8 201.48	133.66	102.92
pr19b	700	6	12	72	8 227.25	80.55	54	10 958.81	177.08	136.35
pr20b	880	6	15	90	10 325.80	150.74	70	13 948.10	282.31	217.38
pr21b	420	12	4	48	4 866.57	36.75	31	6 242.38	58.22	44.83
pr22b	600	12	6	72	6 488.50	73.28	43	8 323.41	140.75	108.38
pr23b	780	12	7	84	8 523.41	163.99	60	11 148.14	207.45	159.73
pr24b	960	12	8	96	10 890.08	298.51	79	14 300.39	256.69	197.65
Sum				1960	245 833.19	2491.47	1562	330 946.55	4888.17	3763.89

Maximum runtime per instance *maxTime*: 300 min
Runs per instance: 10

results for the larger instances confirmed this statement, once GVNS has a lower runtime than the other algorithms in 26 of the 28 instances, providing, however, a result of lower quality both in the number of used vehicles and in the covered distance. For SGVNS, on the other hand, in 22 of the 28 instances in this group, the runtime is shorter than that of SGVNSALS but generating lower quality solutions.

As there are no algorithms in the literature dealing with MD-VRPTW*, we compared the results provided by the proposed algorithm with those of the best-known solutions for two groups of benchmark instances of MDVRPTW, where the objective is only to minimize the covered distance. The results showed that the proposed algorithm reduced the number of used vehicles by 87.50% of the evaluated

instances. In the first group of instances, there was an average reduction of 13.99% in the number of used vehicles and an average increase of 22.71% in the total covered distance. All instances where the number of used vehicles generated by SGVNSALS is equal to the number of available vehicles in the fleet had a wide time window. This fact indicates that these instances have a fair dimensioning. In the second group of instances, there was an average reduction of 23.32% in the fleet size and an average increase of 31.48% in the total distance traveled. It is important to highlight that SGVNSALS obtained a reduction in fleet size in all instances of this group. Additionally, the shown results in Tables 19 and 20 led to conclude that the SGVNSALS procedure was strongly coupled to the search for the best solution to the proposed MDVRPTW* and was weakly suited for solving the classic MDVRPTW. The search for competitive results for the classic MDVRPTW problem would require a complete reformulation of the SGVNSALS procedure, which is far beyond the scope of this article.

Finally, we also evaluate the contribution of neighborhoods to the algorithm's local search and shake operations. This study enabled the identification of the neighborhoods that further contribute to a better exploration of the problem's solution space and the necessary moves to generate search diversification. This analysis allowed us to understand more clearly the characteristics of the solution to this problem and its solution space. The methodology applied for this analysis can be extended to other classes of optimization problems.

CRediT authorship contribution statement

Sinaide Nunes Bezerra: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Marcone Jamilson Freitas Souza:** Conceptualization, Methodology, Data curation, Writing – original draft, Writing – review & editing, Supervision. **Sérgio Ricardo de Souza:** Conceptualization, Methodology, Data curation, Writing – original draft, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors would like to thank the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil, the Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), Brazil, the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil, the Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Brazil, and the Universidade Federal de Ouro Preto (UFOP), Brazil for supporting the development of the present study. This study was financed in part by the Coordination for the Improvement of Higher Education Personnel (CAPES) - Brazil - Finance Code 001.

Compliance and ethical standards

This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abdallah, M.B., Ennigrou, M., 2020. Hybrid multi-agent approach to solve the multi-depot heterogeneous fleet vehicle routing problem with time window (MD-HFVRPTW). In: Madureira, A.M., Abraham, A., Gandhi, N., Varela, M.L. (Eds.), Hybrid Intelligent Systems, Vol. 923. Springer International Publishing, Cham, pp. 376–386. http://dx.doi.org/10.1007/978-3-030-14347-3_37.

Belloso, J., Juan, A.A., Faulin, J., 2019. An iterative biased-randomized heuristic for the fleet size and mix vehicle-routing problem with backhauls. Int. Trans. Oper. Res. 26, 289–301. http://dx.doi.org/10.1111/itor.12379.

Bezerra, S.N., de Souza, S.R., Souza, M.J.F., 2018. A GVNS algorithm for solving the multi-depot vehicle routing problem. Electron. Notes Discrete Math. 66, 167–174, 5th International Conference on Variable Neighborhood Search.

Christiaens, J., Vanden Berghe, G., 2020. Slack induction by string removals for vehicle routing problems. Transp. Sci. 54 (2), 417–433. http://dx.doi.org/10.1287/trsc.2019.0914.

Cordeau, J.F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Soumis, F., 2001a. VRP with time windows. In: The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, USA, pp. 157–193.

Cordeau, J.F., Laporte, G., Mercier, A., 2001b. A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. 52, 928–936.

Cordeau, J.F., Laporte, G., Mercier, A., 2004. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. J. Oper. Res. Soc. 55, 542–546. http://dx.doi.org/10.1057/palgrave.jors.2601707.

Cordeau, J.-F., Maischberger, M., 2012. A parallel iterated tabu search heuristic for vehicle routing problems. Comput. OR 39, 2033–2050. http://dx.doi.org/10.1016/j.cor.2011.09.021.

Gillett, B.E., Johnson, J.G., 1976. Multi-terminal vehicle-dispatch algorithm. Omega 4 (6), 711–718.

Hansen, P., Mladenović, N., Brimberg, J., Pérez, J.A.M., 2019. Variable neighborhood search. In: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics, Vol. 272. Springer, pp. 57–97.

Hansen, P., Mladenović, N., Moreno Pérez, J.A., 2010. Variable neighbourhood search: methods and applications. Ann. Oper. Res. 175 (1), 367–407.

Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S., 2017. Variable neighborhood search: basics and variants. EURO J. Comput. Optim. 5 (3), 423–454.

Hesam Sadati, M.E., Çatay, B., Aksen, D., 2021. An efficient variable neighborhood search with tabu shaking for a class of multi-depot vehicle routing problems. Comput. Oper. Res. 133, 105269. http://dx.doi.org/10.1016/j.cor.2021.105269.

Karakostas, P., Panoskaltsis, N., Mantalaris, A., Georgiadis, M., 2020a. Optimization of CAR T-cell therapies supply chains. Comput. Chem. Eng. 139, 106913. http://dx.doi.org/10.1016/j.compchemeng.2020.106913.

Karakostas, P., Sifaleras, A., Georgiadis, M., 2020b. Adaptive variable neighborhood search solution methods for the fleet size and mix pollution location-inventory-routing problem. Expert Syst. Appl. 153, 113444. http://dx.doi.org/10.1016/j.eswa.2020.113444.

Karakostas, P., Sifaleras, A., Georgiadis, M.C., 2022. Variable neighborhood search-based solution methods for the pollution location-inventory-routing problem. Optim. Lett. 16 (1), 211–235. http://dx.doi.org/10.1007/s11590-020-01630-y.

Lahrichi, N., Crainic, T.G., Gendreau, M., Rei, W., Crişan, G.C., Vidal, T., 2015. An integrative cooperative search framework for multi-decision-attribute combinatorial optimization: Application to the MDPVRP. European J. Oper. Res. 246 (2), 400–412.

Li, J., Li, Y., Pardalos, P.M., 2016. Multi-depot vehicle routing problem with time windows under shared depot resources. J. Comb. Optim. 31 (2), 515–532. http://dx.doi.org/10.1007/s10878-014-9767-4.

Li, Y., Soleimani, H., Zohal, M., 2019. An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives. J. Cleaner Prod. 227, 1161–1172. http://dx.doi.org/10.1016/j.jclepro.2019.03.185.

Li, K., Tian, H., 2016. A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. Appl. Soft Comput. 43 (C), 469–479. http://dx.doi.org/10.1016/j.asoc.2016.02.040.

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

Masmoudi, M., Hosny, M., Koç, Ç., 2021. The fleet size and mix vehicle routing problem with synchronized visits. Transp. Lett. http://dx.doi.org/10.1080/19427867.2021.1888196.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (11), 1097–1100.

Montoya-Torres, J.R., Franco, J.L., Isaza, S.N., Jiménez, H.F., Herazo-Padilla, N., 2015. A literature review on the vehicle routing problem with multiple depots. Comput. Ind. Eng. 79, 115–129. http://dx.doi.org/10.1016/j.cie.2014.10.029.

Noori, S., Ghannadpour, S.F., 2012. High-level relay hybrid metaheuristic method for multi-depot vehicle routing problem with time windows. J. Math. Model. Algorithms 11 (2), 159–179. http://dx.doi.org/10.1007/s10852-011-9171-3.

Ombuki, B., Ross, B.J., Hanshar, F., 2006. Multi-objective genetic algorithms for vehicle routing problem with time windows. Appl. Intell. 24 (1), 17–30.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435.

Polacek, M., Benkner, S., Doerner, K.F., Hartl, R.F., 2008. A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. Bus. Res. 1 (2), 207–218. http://dx.doi.org/10.1007/BF03343534.

Polacek, M., Hartl, R.F., Doerner, K., Reimann, M., 2004. A variable neighborhood search for the multi depot vehicle routing problem with time windows. J. Heuristics 10 (6), 613–627. http://dx.doi.org/10.1007/s10732-005-5432-5.

Rabbouch, B., Mraihi, R., Saâdaoui, F., 2018. A recent brief survey for the multi depot heterogenous vehicle routing problem with time windows. In: Abraham, A., Muhuri, P.K., Muda, A.K., Gandhi, N. (Eds.), Hybrid Intelligent Systems, Vol. 734. Springer International Publishing, Cham, pp. 147–157. http://dx.doi.org/10.1007/978-3-319-76351-4_15.

Rego, M.F., Souza, M.J.F., 2019. 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. INSTICC SciTePress, pp. 287–295. http://dx.doi.org/10.5220/0007703302870295.

Ren, X., Huang, H., Feng, S., Liang, G., 2020. An improved variable neighborhood search for bi-objective mixed-energy fleet vehicle routing problem. J. Cleaner Prod. 275, 124155. http://dx.doi.org/10.1016/j.jclepro.2020.124155.

Rezgui, D., Chaouachi Siala, J., Aggoune-Mtalaa, W., Bouziri, H., 2019. Application of a variable neighborhood search algorithm to a fleet size and mix vehicle routing problem with electric modular vehicles. Comput. Ind. Eng. 130, 537–550. http://dx.doi.org/10.1016/j.cie.2019.03.001.

Solomon, M.M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. 35 (2), 254–265.

Souza, M., Coelho, I., Ribas, S., Santos, H., Merschmann, L., 2010. A hybrid heuristic algorithm for the open-pit-mining operational planning problem. European J. Oper. Res. 207 (2), 1041–1051.

Subramanian, A., Drummond, L., Bentes, C., Ochi, L., Farias, R., 2010. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Comput. Oper. Res. 37 (11), 1899–1911, Metaheuristics for Logistics and Vehicle Routing.

Subramanian, A., Penna, P.H.V., Ochi, L.S., Souza, M.J.F., 2013. 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, first ed. Omnipax, Curitiba, PR, pp. 165–180. http://dx.doi.org/10.7436/2013.mhpo.11, chapter 11.

Tamashiro, H., Nakamura, M., Okazaki, T., Kang, D., 2010. A tabu search approach combined with an extended saving method for multi-depot vehicle routing problems with time windows. Biomed. Soft Comput. Human Sci. 15, 31–39. http://dx.doi.org/10.24466/ijbschs.15.1_29.

Tansini, L., Viera, O., 2006. New measures of proximity for the assignment algorithms in the MDVRPTW. J. Oper. Res. Soc. 57 (3), 241–249. http://dx.doi.org/10.1057/palgrave.jors.2601979.

Ting, C.-J., Chen, C.-H., 2008. Combination of multiple ant colony system and simulated annealing for the multidepot vehicle-routing problem with time windows. Transp. Res. Rec. 2089, 85–92. http://dx.doi.org/10.3141/2089-11.

Todosijević, R., Mladenović, M., Hanafi, S., Mladenović, N., Crévits, I., 2016. Adaptive general variable neighborhood search heuristics for solving the unit commitment problem. Int. J. Electr. Power Energy Syst. 78, 873–883. http://dx.doi.org/10.1016/j.ijepes.2015.12.031.

Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. 60 (3), 611–624.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013a. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. European J. Oper. Res. 231 (1), 1–21.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013b. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput. Oper. Res. 40 (1), 475–489. http://dx.doi.org/10.1016/j.cor.2012.07.018.

Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2014. Implicit depot assignments and rotations in vehicle routing heuristics. European J. Oper. Res. 237 (1), 15–28. http://dx.doi.org/10.1016/j.ejor.2013.12.044.

Wang, Y., Assogba, K., Fan, J., Xu, M., Liu, Y., Wang, H., 2019b. Multi-depot green vehicle routing problem with shared transportation resource: Integration of time-dependent speed and piecewise penalty cost. J. Cleaner Prod. 232, 12–29. http://dx.doi.org/10.1016/j.jclepro.2019.05.344.

Wang, J., Weng, T., Zhang, Q., 2019a. A two-stage multiobjective evolutionary algorithm for multiobjective multidepot vehicle routing problem with time windows. IEEE Trans. Cybern. 49 (7), 2467–2478. http://dx.doi.org/10.1109/TCYB.2018.2821180.