Israel Pereira de Souza

A Reactive GRASP Algorithm for the Multi-depot Vehicle Routing Problem with Time Windows

Vitória, ES

2022

Israel Pereira de Souza

A Reactive GRASP Algorithm for the Multi-depot Vehicle Routing Problem with Time Windows

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Supervisor: Profa. Dr. Maria Claudia Silva Boeres

Vitória, ES

2022

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Souza, Israel Pereira, 1997-
S719r A Reactive GRASP Algorithm for the Multi-depot Vehicle Routing Problem with Time Windows / Israel Pereira Souza. - 2022.
55 f. : il.

Orientadora: Maria Claudia Silva Boeres.
Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Otimização combinatória. 2. Logística. I. Boeres, Maria Claudia Silva. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

To anyone.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

"Se todos fossem, ninguém seria."
(Author)

Abstract

The vehicle routing problem (VRP) is a well know hard to solve problem in literature. In this work, we describe a reactive greedy randomized adaptive search procedures algorithm, for short, reactive GRASP, using a variable neighborhood descent (VND) algorithm as local search procedure to solve the multi-depot vehicle routing problem (MDVRP) and multi-depot vehicle routing problem with time windows (MDVRPTW). This algorithm, called RGRASP+VND, combines four distinct local search procedures and a clustering technique. The Cordeau et al. dataset, a widely well known MDVRP benchmark, is considered for the experimental tests. RGRASP+VND achieves better results on most small instances and a lower average solution cost for all instances on the experimental tests when compared to the earlier GRASP approaches in the MDVRP literature. RGRASP+VND results are also compared with the state-of-the-art of MDVRP and MDVRPTW.

Keywords: Reactive greed randomized adaptive search procedures, Multi-depot vehicle routing problem, City logistics.

Resumo

O problema de roteamento de veículos (PRV) é um problema conhecido na literatura, pela sua elevada complexidade computacional. Neste trabalho descrevemos o algoritmo *Reactive greed randomized adaptive search procedures* (*Reactive GRASP*), usando descendência de vizinhança variável, em inglês *VND*, como procedimento de busca local para resolver o problema de roteamento de veículos com múltiplos depósitos (PRVMD) e o problema de roteamento de veículos com múltiplos depósitos e janelas de tempo (PRVMDJT). Este algoritmo, denominado RGRASP+VND, combina quatro estratégias de busca local e uma técnica de clusterização. O conjunto de instâncias de Cordeau et al., um *benchmark* conhecido para o PRVMD e PRVMDJT, foi utilizado nos testes experimentais. O RGRASP+VND obtém melhores resultados na maioria das instâncias pequenas e a menor média de custo das soluções para todas as instâncias nos testes experimentais comparando com as aplicações anteriores do GRASP na literatura do MDVRP. Os resultados também são comparados com o estado da arte do MDVRP e MDVRPTW.

Palavras-chaves: Reactive greed randomized adaptive search procedures, Problema de roteamento de veículos com múltiplos depósitos e janelas de tempo, Logísticas de cidades.

List of Figures

List of Tables

List of Algorithms

List of abbreviations and acronyms

PRV	Problema de Roteamento de Veículos
PRVMD	Problema de Roteamento de Veículos com Múltiplos Depósitos
PRVMDJT	Problema de Roteamento de Veículos com Múltiplos Depósitos e Janelas de Tempo
VRP	*Vehicle Routing Problem*
VRPTW	*Vehicle Routing Problem with Time Windows*
MDVRP	*Multi-depot Vehicle Routing Problem*
MDVRPTW	*Multi-depot Vehicle Routing Problem with Time Windows*
GRASP	*Greedy Randomized Adaptive Search Procedures*
RGRASP	*Reactive Greedy Randomized Adaptive Search Procedures*

Contents

1 Introduction

The Vehicle Routing Problem (VRP) is a well know logistical model to minimize costs to deliver goods in time to customers against minimal transportation costs. It has a significant impact on logistics systems, optimizing traffic and reducing transportation trajectories, which can directly impact reducing the final cost of the logistical processes.

VRP is an NP-hard [5] problem with several real-world applications. It was first introduced by Dantzig and Ramser [6] in 1959 with the proposal of a mathematical programming formulation and algorithms to model the delivery of gasoline to service stations. In the following, Clarke and Wright [7] in 1964 improved this approach, proposing an effective greedy heuristic. Afterward, the problem was derived into a variety of versions, such as VRP with time windows (VRPTW), multi-depot VRP (MDVRP), multi-depot VRP with time windows (MDVRPTW), VRP with pickup and delivery (VRPPD), VRP with backhauls (VRPB), split delivery VRP (SDVRP), periodic VRP (PVRP), among others. We focus on MDVRP and MDVRPTW.

Similarly to the MDVRP, the MDVRPTW solving methodologies can be classified into two approaches, which are: the route-first cluster-second, where a single giant tour is built considering all customers and then partitioned into a set of feasible vehicles [8]; and the cluster-first route-second, where the customers are organized into clusters, which are associated to the depots and then, the whole route is built considering the clustered depots [9]. Examples of heuristics that follow the cluster-first route-second approach for the MDVRP are: the multi-phase modified shuffled frog leaping algorithm [10] and a tabu search heuristic with variable cluster grouping [11]. The route-first cluster-second approach is adopted by heuristics as an evolutionary algorithm for the VRP [12], a memetic algorithm for the VRPTW [13], among others. This work concerns the proposal of a heuristic for the MDVRP and MDVRPTW adopting the cluster-first route-second approach.

Recent literature on MDVRP heuristics involves: a novel two-phase method approach [14]; a biased-randomized variable neighborhood search [15]; a 2-opt guided discrete antlion optimization [16]; a harmony search [17] and an enhanced intelligent water drops algorithm [18]. A GRASP algorithm was used to solve the single truck and trailer routing problem with satellite depots (STTRPSD) [2], of which MDVRP is a special case. Recent relevant literature on MDVRPTW involves: a multi-phase modified shuffled frog leaping [10]; an improved ACO [19] and POPMUSIC [20].

In this work, we describe a Reactive GRASP algorithm for the MDVRP and MDVRPTW (RGRASP+VND) combining four distinct local search procedures, two of them presented here, and a clustering technique. The contributions of this master

thesis include two local search procedures and a Reactive GRASP application study for the MDVRP and MDVRPTW variants. It also studies the application of two literature clustering methods for both variants, carrying out an extensive parameter calibration of Solomon Insertion Heuristic I1 [21], and the RGRASP+VND algorithm to solve MDVRP and MDVRPTW variants. For purposes of comparison, a literature review was conducted to identify the state-of-the-art algorithms for the MDVRP. For the computational experiments and results analysis, we consider as input, the well-known Cordeau et al. dataset for the MDVRP [4] and MDVRPTW [22]. State-of-the-art algorithms are identified and compared to RGRASP+VND and the results are discussed in Section 7.2.

The remainder of this master thesis is structured as follows: Section 2 presents the theoretical referential of the VRP. Section 3 describes the problem with its mathematical formulation. Section 4 presents MDVRP and MDVRPTW solving approaches, which are the route-first cluster-second and cluster-first route-second, and discusses two clustering techniques. Section 6 describes the RGRASP+VND algorithm for the MDVRP and MDVRPTW. The analysis of the results obtained from the computational experiments is presented in Section 7, which is organized with the algorithm parameters tuning (Section 7.1), followed by the comparison of the state-of-the-art algorithms (Section 7.2). Section 8 concludes this work.

A contribution of this work to the literature was reported in a paper accepted and published on *The 22nd International Conference on Computational Science and Its Applications*, held on 4-7 July in Malaga, Spain.

2 Theoretical referential

2.1 The VRP variants

The VRP is one of the most important, and studied, combinatorial optimization problem [23]. A large number of real-world applications have shown that the use of computerized procedures for distribution process planning offers a substantial cost decrease (generally 5% to 20%) in the global transportation costs, which can significantly impact the global economic system. The transportation process involves all stages of production and distribution systems and represents generally from 10% to 20% of the final cost of the goods [23].

The VRP was first introduced by Dantzig and Ramser [6] in 1959 with the proposal of a mathematical programming formulation and algorithms to model the delivery of gasoline to service stations. In the following, Clarke and Wright [7] in 1964 improved the VRP approach, proposing an effective greedy heuristic. Afterward, the problem was derived into a variety of versions. In the following, we describe several VRP variants. Many are described in the survey [1]. We also have used other references to explain variants not included in the survey or for further explanation of those which are included in this work. They are described as follows:

- CVRP: the basic VRP, known as the Capacitated VRP. The CVRP goal is to find the set of routes for a fleet of identical vehicles with an overall minimum route cost that serves all the customer's demands [1].
- VRPTW: the VRP with Time Windows establishes that each vehicle delivers the goods to the customers within a specific time interval. There are two types of time windows: the VRP with soft time window (VRPSTW), in which violating the time windows is allowed but associated with penalty costs; and VRP with hard time window (VRPHTW or VRPTW) in which the time windows do not allow any delay [1].
- PDVRP/VRPPD: the Pickup and Delivery VRP or VRP with Pickup and Delivery defines several goods that need to be moved from certain pickup locations to other delivery locations. The Pickup and Delivery Problem (PDP) can be classified into VRP with backhauls (VRPB), VRP with Clustered Backhauls (VRPCB), VRP with mixed backhauls (VRPMB), VRP with divisible deliveries and pickups (VRPDDP) and VRP with simultaneous pickups and deliveries (VRPSPD) [1].
- HVRP: the Heterogeneous VRP (HVRP) assumes the vehicles have different capaci-

ties) [1].

- MDVRP: the Multi-depot VRP considers that a company may have several depots from which it can serve its customers [1]. The fleet of vehicles now must serve several depots instead of only one and each vehicle must start and end at the same depot [24]. There are multiple bases (depots) from where vehicles are starting. Depending on the situation, vehicles are either obligated to return to the starting base (fixed) or they have no such limitation (non-fixed) [25].
- MDVRPI: the Multi-depot VRP with Inter-Depot Routes specifies that vehicles might be replenished at intermediate depots along their route [26].
- PVRP: the Periodic VRP extends the planning horizon to several days and customers may be visited more than once [1]. The Periodic VRP is used when planning is constructed over a certain period and deliveries to the customer can occur on different days [27].
- SDVRP: the Split-delivery VRP specifies that the demands of customers are allowed to be split and delivered using several vehicles on different routes [1].
- SVRP: the Stochastic VRP has one or several components of the problem which follow a random probability distribution. The SVRP can be divided into the VRP with stochastic demand (VRPSD), the VRP with stochastic customers (VRPSC), the VRP with stochastic demands and customers (VRPSDC), and the VRP with stochastic travel and service times (VRPSTS) [1].
- OVRP: the Open VRP specifies that vehicles do not necessarily return to the original depot after completing their delivery services. If they do, they must visit the same customers in the reverse order [1].
- TDVRP: the Time-dependent VRP has the travel speeds (times) assumed to depend on the time of travel when planning vehicle routing [1].
- Other less commonly VRP variants are Green VRP (G-VRP), Dynamic VRP (DVRP), VRP with Loading Constraints (VRPLC), Generalized VRP (GVRP), Multi Trip VRP (MT-VRP), Truck and Trailer routing problem (TTRP), Multi-compartment VRP (MCVRP), Fuzzy VRP, Site-dependent VRP (SD-VRP), among others.

In this work, we focus to solve the MDVRP and MDVRPTW.

2.2 Metaheuristics approach

Metaheuristics are widely recognized in the literature as efficient approaches for many hard optimization problems [28]. The metaheuristics can be grouped into two categories: single-solution based and population-based [1].

The population-based metaheuristics include evolutionary and populational-based metaheuristics, that maintain a group of solutions and usually represent a process in nature. Their evolutive process generally uses more than one population member to create another solution candidate.

The single-solution-based metaheuristics, except for simulated annealing that describes a physics process, usually are more algorithmic approaches and do not describe a process in nature. Their optimization processes are generally based on the information of the solution being built.

Figure 1 shows a classification tree, reproduced from 1, considering many metaheuristic algorithms.

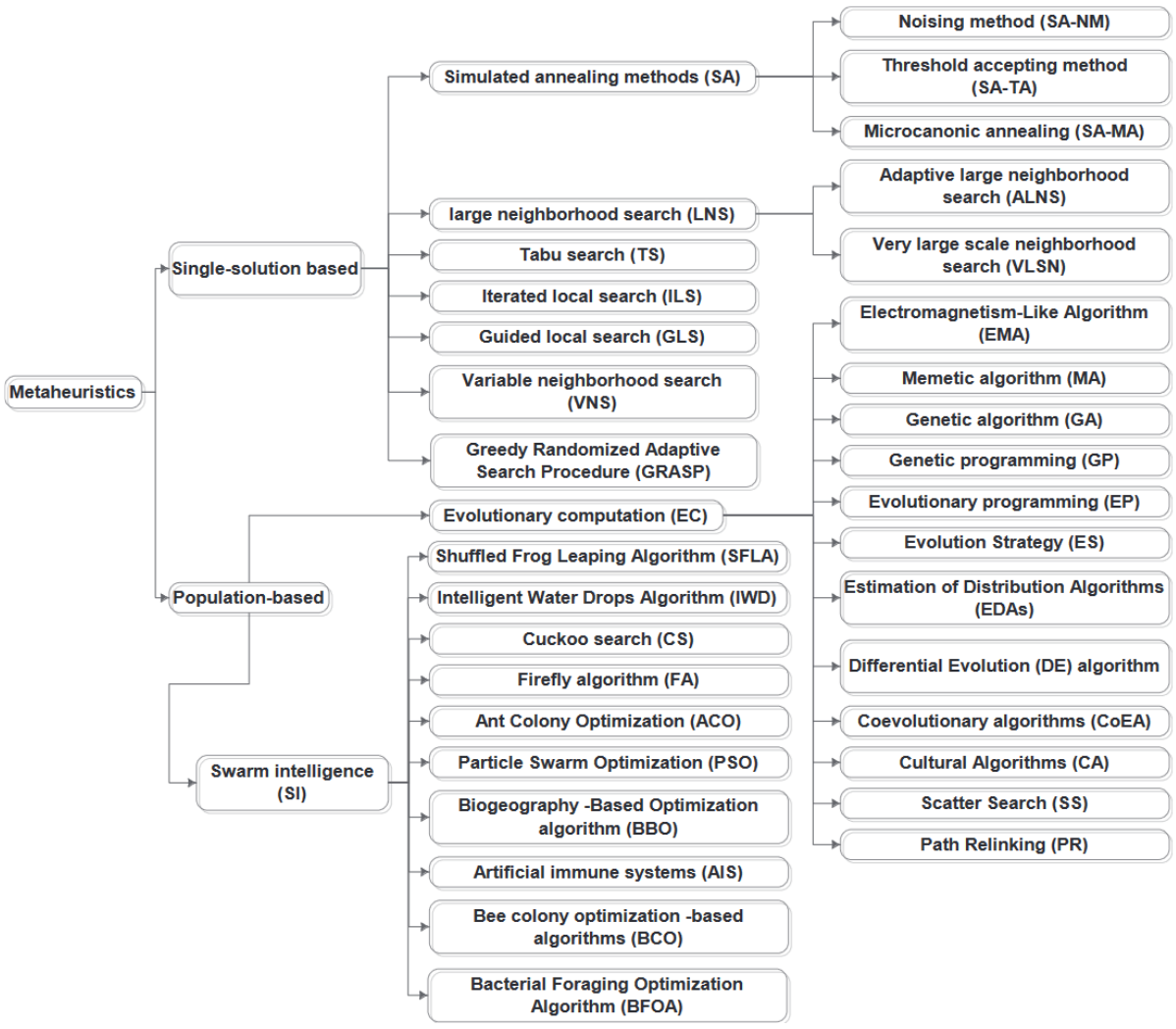


Figure 1 – Classification tree of metaheuristic algorithms [1].

3 Problems description

This chapter reproduces in Section 3.1 and 3.2 the Multi-depot Vehicle Routing Problem and the Multi-depot Vehicle Routing Problem with Time Windows description and the mathematical formulation presented in [29] and [19].

3.1 The Multi-depot Vehicle Routing Problem

The Multi-depot Vehicle Routing Problem (MDVRP) can be defined using an undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_N, v_{N+1}, v_{N+2} \ldots, v_{N+M}\}$ is the vertex set, $D = \{D_1, D_2, \ldots, D_M\} = \{v_{N+1}, v_{N+2}, \ldots, v_{N+M}\}$ is the depot set and $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is the edge set. The vertices v_1 to v_N represent the clients or customers and v_{N+1} to v_{N+M}, the depots. Each customer v_i has an associated demand q_i, $i = 1, \ldots, N$. The depots have no associated demand. Moreover, each edge $(v_i, v_j) \in E$ is associated to d_{ij}, which is the Euclidean distance computed between vertices i and j. A fleet of up to K vehicles of capacity Q_k are located in each of the M depots. The MDVRP goal is to minimize a set of vehicle routes such that:

- each vehicle starts and ends at the same depot;
- each customer is served exactly once by only one vehicle;
- a total load of a vehicle k, for $k = 1, \ldots, R$, does not exceed its maximum capacity Q_k;
- the number of vehicles used by a depot does not exceed the maximum number of vehicles K.

The input parameters and decision variables considered for the mathematical formulation are described in the following:

N total number of customers.

M total number of depots.

K maximum number of vehicles for a depot.

R number of vehicles used by the M depots.

Q_k maximum capacity of a vehicle $k = 1, \ldots, R$.

T_k maximum distance allowed for a route from a vehicle $k = 1, \ldots, R$.

q_i demand of a customer v_i, $i = 1, \ldots, N$.

d_{ij} Euclidean distance between vertices v_i and v_j.

Decision variables:

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ travels from } v_i \text{ to } v_j. \\ 0, & \text{otherwise.} \end{cases}$$

The mathematical formulation proposed in [29] is reproduced in the following:

Minimize:

$$Cost = \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^{R} d_{ij} x_{ijk} \tag{3.1}$$

$$\sum_{i=1}^{N+M} \sum_{k=1}^{R} x_{ijk} = 1, \quad j = 1, 2, \ldots, N \tag{3.2}$$

$$\sum_{j=1}^{N+M} \sum_{k=1}^{R} x_{ijk} = 1, \quad i = 1, 2, \ldots, N \tag{3.3}$$

$$\sum_{i=1}^{N+M} x_{ihk} - \sum_{j=1}^{N+M} x_{hjk} = 0, \quad k = 1, 2, \ldots, R, \;\; h = 1, 2, \ldots, N + M \tag{3.4}$$

$$\sum_{i=1}^{N+M} q_i \sum_{j=1}^{N+M} x_{ijk} \leq Q_k, \quad k = 1, 2, \ldots, R \tag{3.5}$$

$$\sum_{i=1}^{N+M} \sum_{j=1}^{N+M} d_{ij} x_{ijk} \leq T_k, \quad k = 1, 2, \ldots, R \tag{3.6}$$

$$\sum_{i=N+1}^{N+M} \sum_{j=1}^{N} x_{ijk} \leq 1, \quad k = 1, 2, \ldots, R \tag{3.7}$$

$$\sum_{j=N+1}^{N+M} \sum_{i=1}^{N} x_{ijk} \leq 1, \quad k = 1, 2, \ldots, R \tag{3.8}$$

$$y_i - y_j + (M + N) x_{ijk} \leq N + M - 1 \text{ for } 1 \leq i \neq j \leq N \text{ and } 1 \leq k \leq R \tag{3.9}$$

The objective function (3.1) minimizes the total distance traveled by the vehicles. Constraints (3.2) and (3.3) guarantee that each customer is served by one vehicle. The route continuity is represented by constraint (3.4). Constraint (3.5) ensures that the vehicle capacities are respected and the constraint (3.6) ensures that the vehicle maximum distance is respected. Constraints (3.7) and (3.8) verify vehicle availability and subtour elimination is checked by constraints (3.9).

3.2 The Multi-depot Vehicle Routing Problem with Time Windows

The Multi-depot Vehicle Routing Problem with Time Windows (MDVRPTW) can be defined using a directed graph $G = (V, E_{TW})$, where $V = \{v_1, v_2, \ldots, v_N, v_{N+1}, v_{N+2} \ldots, v_{N+M}\}$ is the vertex set, $D = \{D_1, D_2, \ldots, D_M\} = \{v_{N+1}, v_{N+2}, \ldots, v_{N+M}\}$ is the depot set and $E_{TW} = \{(v_i, v_j) : v_i, v_j \in V\}$ is the arcs set. The vertices v_1 to v_N represent the clients or customers and v_{N+1} to v_{N+M}, the depots. Each customer $v_i, i \in [1, N]$ has an associated demand q_i, a service time s_i and a time windows $[e_i, l_i]$ where e_i is the earliest time of service that v_i can initiate and l_i is the latest time of service that v_i can initiate. Depots have no associated demand or service time and their time windows represents their operating time. Moreover, the distances d_{ij} and travel times t_{ij} are associated with each arc $(v_i, v_j) \in E_{TW}$, which is the Euclidean distance computed between vertices v_i and v_j. A fleet of up to K vehicles of capacity Q_k and a non-negative maximum route duration T_k are located in each of the M depots. The MDVRPTW goal is to minimize a set of vehicle routes such that:

- each vehicle starts and ends at the same depot;
- each customer is served exactly once by only one vehicle;
- a total load of a vehicle k, for $k = 1, \ldots, R$, does not exceed its maximum capacity Q_k;
- the number of vehicles used by a depot does not exceed the maximum number of vehicles K;
- the service beginning time b_i for each customer c_i lies in its time windows $[e_i, l_i]$;
- each route starts and ends within the time windows of its depot.

The MDVRPTW is similar to MDVRP and can be defined by using the MDVRP mathematical formulation described in Section 3.1 also adding the time windows constraints, which were presented in [19] and are described in the following. These constraints require the graph G to be a directed graph, considering that the traveling time between vertices can be different considering the direction (t_{ij} can differ by t_{ji}).

Parameters:

s_i service time of vertex v_i.

t_{ij} travelling time from vertex v_i to vertex v_j.

e_i earliest service time of vertex v_i.

l_i latest service time of vertex v_i.

b_i service starting time for customer v_i.

Constraints:

$$b_{N+1} = b_{N+2} = \cdots = b_{N+M} = 0, \tag{3.10}$$

$$\sum_{i=1}^{N+M} \sum_{k=1}^{R} x_{ijk}(b_i + s_i + t_{ij}) \leq b_j, \quad j = 1, 2, \ldots, N \tag{3.11}$$

$$e_i \leq b_i \leq l_i, \quad i = 1, 2, \ldots, N \tag{3.12}$$

The constraint 3.10 ensures that each depot starting time is zero. Constraint 3.11 guarantee that the service starting time of customer v_j occurs after serving the customer v_i, for each arc (v_i, v_j), $i, j \in [1, N + M]$ and $i \neq j$. Constraint 3.12 guarantee the service starting time for each customer respects its time windows.

4 Solution construction

In this section, we present the solution construction approaches. Section 4.1 presents the clustering techniques. Section 4.2 presents the constructive algorithm adopted for this work and a literature study of constructive algorithms for the VRPTW.

4.1 Clustering approaches

Similar to the MDVRP, heuristics for solving the MDVRPTW can be classified into cluster-first route-second or route-first cluster-second approaches. In the first approach, the vertices are clustered considering a depot as the center of each cluster. The routes are then determined by properly sequencing the customers in each cluster [9]. In the second approach, a single giant route is built considering all vertices, and then it is split into a set of feasible routes [8].

Several clustering methods can be found in the literature. A classical is K-means [30]. K-means partitions a set of points into a given number of clusters, assigning them to a center or centroid determined by the nearest distance mean.

Algorithm 1: K-means pseudocode

Input: X, k.
Output: Data points with cluster memberships
1 **do**
2 Initialize the k centroids randomly
3 Associate each data point in X with the nearest centroid. (This will divide the data points into k clusters)
4 Recalculate the position of the centroids
5 **while** *a cluster was updated*;

Algorithm 1 reproduces the K-means pseudocode present in [31]. It receives as input the set of data points X (which are the vertices in the VRP) and the number of clusters k. Although the K-means description given in algorithm 1 can perform the while loop many times, it executes only once in the VRP application. In the VRP application, the center of each cluster is defined as the depot vertice and for this reason, the centers are not updated.

A group of different clustering methods has been described by Giosa et al. [32] to solve the MDVRP. One of them, the Urgencies clustering method, ranks all customers considering an urgency, and the customer with the most urgency is assigned first. In the same work, Giosa et al. [32] also present a parallel and simplified approach for the

Urgencies clustering method. The parallel approach considers all depots for the customer urgency evaluation and the simplified approach considers only two depots. The customer with the highest urgency is assigned to its closest depot. Equation (4.1) calculates the urgency u_c of a customer v_c, $c = 1, 2, \ldots, N$, considering the Urgencies clustering method with parallel approach. Function `distance` calculates the Euclidean distance between two vertices. D' is the closest depot to the customer v_c. The customer c with the highest value of u_c is assigned to its closest depot.

$$u_c = \left(\sum_{i=1}^{M} \texttt{distance}(v_c, D_i) \right) - \texttt{distance}(v_c, D') \tag{4.1}$$

Algorithm 2 presents the Urgencies clustering method pseudocode. It receives as input the set of customer $C = V \backslash D$, the depots set D, the number of depots M, the number of vehicles of each depot K, the set of vehicle capacities Q, and returns the clustered solution, where each customer is assigned to a depot. The urgencies algorithm starts defining the maximum amount of demand each depot can attend, which is calculated by the sum of the capacity of each vehicle Q_k of the depot (line 2). Until there are customers which were not clustered yet, the algorithm calculates the urgency for all customer not yet clustered, performed by the function `Urgency` defined in Equation 4.1 that evaluate the customer urgency with a parallel approach (line 9). The customer with the highest urgency value is obtained (line 10) and assigned to its closest depot, which has free demand to support it (line 11), and deduct the customer demand q_p from the free demand of depot D_q, identified as $DepotFreeDemands[q]$ (line 12).

Algorithm 2: Urgencies with parallel approach pseudocode

Input: C, D, M, Q, K.
Output: Customers with depot memberships

```
1  for i ← 1 to M do
2  |  DepotFreeDemands[i] ← ∑_{k=1}^{K} Q_k
3  end for
4  while all customers was not clustered do
5  |  Urgencies ← ∅
6  |  foreach unrouted customer c_i do
7  |  |  u_i ← Urgency(c_i, D, M)
8  |  |  Urgencies ← Urgencies∪u_i
9  |  end foreach
10 |  c_p ← get customer with the highest urgency value in Urgencies
11 |  Assign c_p to its closest depot D_q, with free load to support c_p
12 |  DepotFreeDemands[q] ← DepotFreeDemands[q] - q_p
13 end while
```

Figure 2 shows a solution example plot generated with the cluster-first route-second approach for the instance `pr01`, available in the Cordeau et al. dataset [4]. Clustering

solutions obtained by the methods K-means and Urgencies are plotted in Figure 3. The clusters are highlighted in different colors. We can see, for this instance, that the clustering solutions are quite similar, differing only by 1 additional customer in the red and green clusters, when considering the K-means solution, against 2 additional customers in the blue cluster, when considering the Urgencies solution. Each cluster represents a VRP/VRPTW instance subproblem and the union of all subproblems composes an MDVRP/MDVRPTW instance problem constructed with the cluster-first route-second approach.

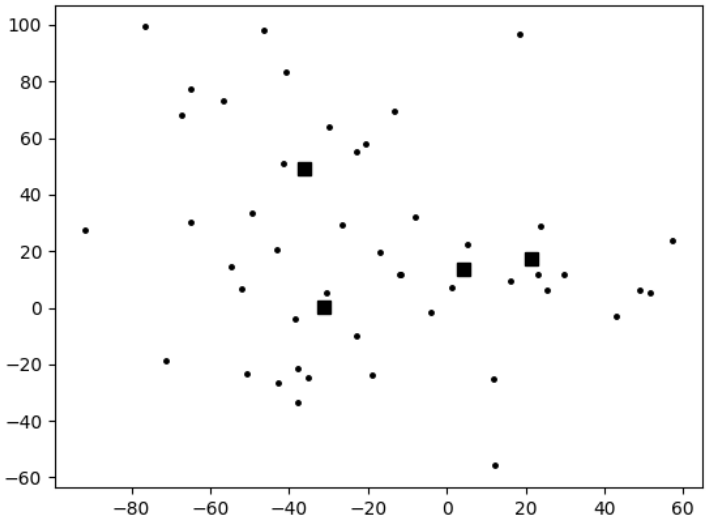
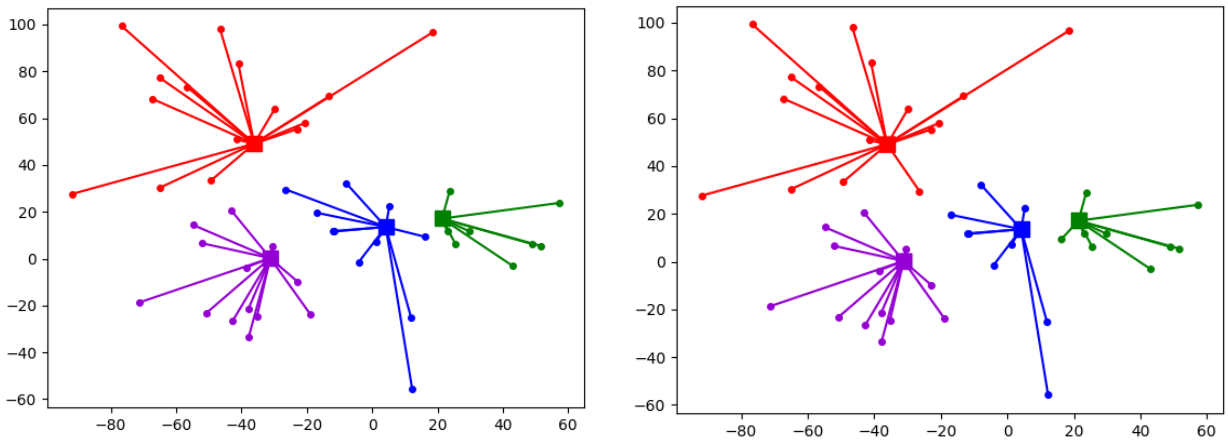


Figure 2 – Plotted MDVRP pr01 instance from Cordeau et al. set (1997) (points are customers and squares are depots).



(a) (b)

Figure 3 – (a) Plotted K-means clustering method on pr01 (b) Plotted Urgencies clustering method on pr01.

In this work, we consider the cluster-first route-second approach as a solution strategy for both problem variants. In this work, we compare the K-means clustering method and the Urgencies clustering method with the parallel approach [32] applied to

the cluster-first route-second approach, which is discussed in Section 7.1. Both clustering methods were implemented considering a maximum capacity for the depot, which is the sum of the vehicle capacities in the depot. If a chosen customer to be assigned to a cluster has a demand that surpasses the cluster depot capacity, it is assigned instead to the next closest cluster with a capacity to support it.

4.2 Constructive algorithm

The literature presents several constructive methods. To select the constructive heuristic for the VRPTW (the route-second phase) a literature review was performed. The search string: *(("constructive") AND ("algorithm" OR "heuristic")) AND ("Vehicle Routing Problem with Time Windows" OR "VRPTW")* was used on Scopus database up to 2022. We have observed that the Solomon Insertion Heuristic I1 [21], Solomon Sweep Algorithm [21] and Clarke-Wright savings algorithm [7] were the most used constructive algorithms in this review.

Clarke-Wright savings algorithm was not proposed for time window constraints. However, the literature presents adaptations for the algorithm to include time window constraints. Solomon Insertion Heuristic I1 and Solomon Sweep Algorithm, both consider time window constraints. Solomon's paper [21] presents a comparison of the constructive solution in different datasets. There is not a clear winner considering the Insertion Heuristic I1 and the Sweep Algorithm. However, considering the higher amount of uses of Solomon Insertion Heuristic I1 in the literature review, it was the adopted constructive method for this algorithm.

Solomon Insertion Heuristic I1 [21] is a known constructive algorithm in the literature. It starts by initializing a route with three vertices. The first and last vertices are the depot and the second vertice is chosen as the initialization criteria. There are three initialization criteria: (i) the farthest unrouted customer (FC); (ii) the unrouted customer with the earliest deadline (CTW); (iii) the unrouted customer with the minimum equally weighted combination of direct route-time and distance (CTWD).

Solomon Insertion Heuristic I1 has four input parameters: $\alpha_1, \alpha_2 \in [0, 1]$, $\lambda, \mu \geq 0$. To select a new customer to be part of the solution, five major equations are evaluated, that select the customer according to the input parameters. The algorithm evaluates the insertion of an unrouted customer u between two subsequent customers c_i and c_j in the route being built. Equation (4.2) presents the c_1 evaluation, that considers the α_1 and α_2 input parameters, which modifies the importance of distance and time windows, respectively, in the evaluation. For each insertion between positions $p-1$ and p in the route, the equation evaluates c_{1e} and returns the feasible insertion with the lowest cost. Equation (4.4) and (4.5) are used in the c_{1e} evaluation. Equation (4.4) calculates the

savings. It also has the μ input parameter that modifies the savings evaluation. When $\mu = 1$, it calculates the savings similarly to the Clarke-Wright savings algorithm [7]. Equation (4.5) calculates the time windows increase after insertion of customer c_u, where b_{j_u} is the initiation time of customer c_j, when u is inserted, minus the initiation time b_j of customer c_j. Equation (4.6) presents the c_2 evaluation, that considers the λ parameter, which increases the importance of customers more distant of the depot.

Algorithm 3 presents the Solomon Insertion Heuristic I1 pseudocode. It initializes the first route (2) with the *init_criteria* chosen for the algorithm. The algorithm runs while there are unrouted customers. To select a new customer to be part of the solution, the algorithm empties the c_1List and evaluates the c_1, using Equation (4.2) for each unrouted customer u and the result is added to the c_1List, that specifies the insertion of the customer u in p_{-1} and p positions in the route, which is feasible and yield the minimum insertion cost. Then, if the c_1List is still empty, the algorithm could not find any feasible insertion for the current route being built. Then the route is added to the *routes* list (11) and a new route is initialized, using the same *init_criteria* (12) and the algorithm returns to the customer selection (13). Otherwise, the algorithm empty the c_2List (15) and for each item in the c_1List, the algorithm calculates the c_2 (17), using Equation (4.6) and then add the result to the c_2List (18). Finally, the item with the highest value is selected in the c_2List (20) and the customer u is inserted between p_{-1} and p positions in the route (21).

$$c_1(i,u,j) = \texttt{MinimumFeasible}[c_{1e}(i_{p-1},u,i_p)], \quad i = 1,2,\ldots,n. \tag{4.2}$$

$$c_{1e}(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{22}(i,u,j), \quad \alpha_1 + \alpha_2 = 1 \tag{4.3}$$

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij}, \quad \mu \geq 0 \tag{4.4}$$

$$c_{12}(i,u,j) = b_{j_u} - b_j \tag{4.5}$$

$$c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j), \quad \lambda \geq 0 \tag{4.6}$$

Algorithm 3: Solomon Insertion Heuristic I1 pseudocode

Input: *init_criteria*, α_1, α_2, λ, μ
Output: *routes*

```
1  routes ← ∅
2  route ← RouteInitialization(init_criteria)
3  while there are unrouted customers do
4  |   RESTART:
5  |   c1List ← ∅
6  |   foreach unrouted customer u do
7  |   |   c1(p−1, u, p) ← CalculateC1(route, u, α1, α2, μ)
8  |   |   c1List ← c1List ∪ c1(p−1, u, p)
9  |   end foreach
10 |   if c1List is empty then
11 |   |   routes ← routes ∪ route
12 |   |   route ← RouteInitialization(init_criteria)
13 |   |   goto RESTART
14 |   end if
15 |   c2List ← ∅
16 |   foreach c1(p−1, u, p) ∈ c1List do
17 |   |   c2(p−1, u, p) ← CalculateC2(c1(p−1, u, p), λ)
18 |   |   c2List ← c2List ∪ c2(p−1, u, p)
19 |   end foreach
20 |   c2(p*−1, u*, p*) ← get the item with the highest value in c2List
21 |   insert u* between p−1 and p in route
22 end while
23 return routes
```

5 Local Search Strategies

Local search procedures used by this algorithm considers four neighborhoods: `2-swap`, `2-opt`, `Drop one point intra-depot`, `Drop one point next depot`. The first two are classical heuristics widely used for optimization problems, the last two algorithms are new local search methods presented in this work.

The `2-swap` and `2-opt` [33, 34] movements perform, respectively, two vertices (customers) swaps and two edges swaps while preserving the tour, both preserving feasibility. According to [35], the neighborhood of a solution s is defined as a function that maps a solution to a set of solutions. The neighborhood search algorithm takes as input an initial solution s and computes the best solution s' in the neighborhood of s. Two classical ways to perform this search are the first improvement and the best improvement. The first improvement enumerates systematically and updates the solution when an enhancement is found and the best improvement only updates the solution with the best solution, that is the solution with the highest enhancement obtained in the neighborhood. In this work, we adopted the best improvement strategy for the `2-opt` and `2-swap` movements.

The `2-opt`, `2-swap` and `Drop one point intra-depot` are intra-depots methods. They perform inside the clusters. `Drop one point next depot`, however, is an inter-depot. In this case, changes are performed considering all clusters of the solution.

Drop One Point Intra-depot For each cluster associated with depot D_i of a solution S, every customer c_i which is clustered to depot D_i is removed, then reinserted in a different route of the depot D_i which the highest enhancement is obtained and the feasibility is respected. Algorithm 4 shows the `Drop one point intra-depot` pseudocode. Notation D_{m,c_i} represents the depot D_m that attends the customer c_i.

Loop from lines 1-9 iterates for each depot D_i of S. Loop of lines 2-8 and 3-7 iterates, respectively, for each route h_l of D_i and all cities c_i in route h_l. Line 5 removes c_i from its current route and reinserts it in a different route that gives the highest enhancement to S, forbidding infeasible moves. This movement is only performed if it improves the solution cost, checked in line 6.

Drop One Point Next Depot The `Drop one point next depot` algorithm seeks to adjust the clusters of the solution, for this reason, it's an inter-depot algorithm. For each city c_i, it's next closest depot D_{next,c_i} is calculated, differently than the one c_i is currently allocated. Then if an improvement is obtained, c_i is removed from S and viably reinserted on the depot D_{next,c_i} where the highest enhancement is obtained. The Algorithm 5 shows

Algorithm 4: Drop one point intra-depot

```
   Input: S
   Output: S
1  foreach depot D_i ∈ S do
2      foreach route h_ℓ ∈ D_i do
3          foreach c_i on route h_ℓ do
4              do
5                  S ← remove and viably reinsert c_i on route h_b, h_b ≠ h_l and h_b ∈ D_i,
                       such as its cost is minimum.
6              if the cost of S reduces;
7          end foreach
8      end foreach
9  end foreach
```

the `Drop one point intra-depot` pseudocode.

Algorithm 5: Drop one point next depot

```
   Input: S, N, M, K, Q
   Output: S
1  max_demands ← Σ_{k=1}^{K} Q_k
2  for m ← 1 to M do
3      cluster_demands[m] ← sum of demands of all customers clustered to D_m
4  end for
5  for i ← 1 to N do
6      D_{current,c_i} ← get the depot c_i is currently associated.
7      m ← index associated to depot D_{current,c_i}.
8      if cluster_demands[m] + q_i <= max_demands then
9          D_{next,c_i} ← get D_{next,c_i} the closest depot of c_i, where D_{next,c_i} ≠ D_{current,c_i}.
10         do
11             S ← remove c_i from S and viably reinsert c_i on D_{next,c_i} on the route
                   and position where the highest enhancement is obtained.
12             cluster_demands[m] = cluster_demands[m] − q_i
13             m ← index associated to depot D_{next,c_i}.
14             cluster_demands[m] = cluster_demands[m] + q_i
15         if the cost of S reduces;
16     end if
17 end for
```

The `Drop one point next depot`, on line 1, calculates the maximum amount of demand that a depot can support. The line 2 iterates over all depots of the instance and the line 3 calculates the amount of demand the cluster has, which is the sum of demands of all customers on the cluster. The line 5 iterates over all customers c_i of the instance. The line 6 get the depot ($D_{current,ci}$) which the customer c_i is associated. The line 7 get the index m of the depot $D_{current,ci}$. The line 8 checks if the depot with index m can support the customer c_i. The line 9 get the closest depot of c_i, different than the depot c_i is already

allocated, called $D_{next,ci}$. The line 11 removes c_i and reinserts it on the depot $D_{next,ci}$ on the position that the highest enhancement is obtained, this movement is performed only if the cost of S is improved, which is checked on line 15. The lines 12-14 are used to update the amount of demand the clusters are holding.

5.1 Variable Neighborhood Descent

Variable Neighborhood Descent (VND) [36] algorithm performs a change of neighborhoods in a deterministic way. The neighborhoods are denoted as N_k, $k = 1, ..., k_{max}$ [35].

Algorithm 6 presents the VND pseudocode. The solution s is given as input. VND starts with the first neighborhood N_1, from the k neighborhoods given as input. For each neighborhood $N_l, 1 \leq l \leq k$, the algorithm will obtain the best neighbor. If the obtained solution achieves a better function objective cost, the VND will return the search on the neighbor N_1 from s', otherwise, it will search on the next neighborhood N_{l+1} from s. The algorithm concludes when it searches the last neighborhood N_k and does not find an improvement.

Algorithm 6: Variable Neighborhood Descent pseudocode

```
   Input: s, N_k
 1 k ← 1
 2 while k < k_max do
 3 |   s' ← Find the best neighbor in N_k(s)
 4 |   if Cost(s') < Cost(s) then
 5 |   |   s ← s'
 6 |   |   k ← 1
 7 |   else
 8 |   |   k ← k + 1
 9 |   end if
10 end while
```

5.2 Local search analysis

To analyze the local search results, described in Section 5, the first nine instances (pr01-pr09) of the Cordeau et al. dataset for the MDVRPTW [22] were used.

First, the problem instance is clustered with the K-means algorithm. Then the initial solution is constructed with Solomon Insertion Heuristic I1 and the local search heuristics are performed. For this analysis, we have selected the standard parameters for Solomon (α_1=0.5, $\alpha_2 = 0.5, \mu = 1, \lambda = 1$), presented in [21], which considers the same importance for distance and time windows on the instances. The farthest customer was

used for the Insertion Heuristic initialization criteria. The four local search neighborhoods were organized in the Variable Neighborhood Descent.

Table 1 shows the results of the four local search neighborhoods, where T_1 represents the 2-swap with the best improvement, T_2 represents 2-opt intra routes, T_3 represents drop one point next depot and T_4 represents drop one point intra depots. Each local search method was executed until the local minimum was reached. To select which order the local search methods would be performed, we have tested all the combinations with all the four methods, totaling 24 combinations. The combination with the lowest average was: $(T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4)$ and was the sequence used on VND.

	Solomon I1	T_1	T_2	T_3	T_4	$T_1 \rightarrow T_2$	$T_1 \rightarrow T_2$ $\rightarrow T_3$	$T_1 \rightarrow T_2$ $\rightarrow T_3 \rightarrow T_4$
pr01	1426.89	1254.08	1286.29	1271.05	1321.07	1254.08	1240.60	1205.60
pr02	2631.18	2481.22	2502.13	2466.73	2489.42	2415.56	2185.90	2090.07
pr03	3595.78	3023.13	3164.30	3499.42	3383.68	2971.68	2905.09	2754.09
pr04	4399.37	3897.92	4061.01	4310.50	3969.54	3881.40	3855.76	3591.23
pr05	4934.83	4341.49	4687.10	4637.86	4541.22	4333.64	4119.61	4056.61
pr06	5789.44	5004.82	5359.73	5654.54	5271.03	4949.85	4936.53	4613.89
pr07	1931.01	1667.39	1723.35	1841.45	1780.32	1667.39	1653.23	1633.11
pr08	3449.00	2737.46	3041.74	3332.76	3174.38	2737.46	2682.91	2532.88
pr09	4262.61	3876.95	3943.56	4120.14	3907.61	3847.04	3817.22	3401.67
Average	3602.23	3142.72	3307.69	3459.38	3315.36	3117.57	3044.09	2875.46

Table 1 – Local search methods performance on Cordeau et al. instances

Comparing the singular results of the local search neighborhoods, 2-swap achieved the best solutions and also the best average solution in the nine instances. When combining one more local search method to the comparison, except for adding the 2-opt, (columns 8,9 of Table 1), the solutions have presented an improvement for all instances. The 2-opt addition has presented the weakest improvement when performing after the 2-swap method (column 7 of Table 1), where the solution was not improved on instances pr01, pr07, and pr08.

We can conclude, for this analysis, that the neighborhoods T_1, T_3, and T_4 had the biggest impact on the solution improvement, where each addition has presented improvement for every instance of the analysis. The strategy T_2 had the smallest impact and its addition after the method T_1 did not present any improvement on instances pr01, pr07, and pr08, however, the addition has contributed to an improvement on 6 instances (pr02, pr03, pr04, pr05, pr06, and pr09). For this reason, the strategy T_2 was kept on the

algorithm.

Another analysis was conducted to compare the Variable Neighborhood Descent (VND) results against the sequential local search (LS), which changes the neighborhoods sequentially, instead of returning to the first one after an improvement. LS performs similarly to the VND, performing the change between neighborhoods in a deterministic way. However, instead of returning to the first neighborhood, it keeps exploring the next neighborhood and if an improvement is found, the procedure is repeated, and all neighborhoods are re-explored. Again, the farthest unrouted customer was used as the initialization criteria for the Solomon Insertion Heuristic I1 and the K-means was the clustering method adopted. Table 2 presents the comparison of the combined four local search methods executed by VND and LS. For each local search structure, there is a column with the solution cost, the time in seconds, and the number of exhausted neighborhoods that were necessary to reach the local optimum (described by iter. column).

	$LS(T_1, T_2, T_3, T_4)$			VND(T_1, T_2, T_3, T_4)		
	Solution	Time (s)	iter.	Solution	Time (s)	iter.
pr01	**1205.60**	0.0510	8	**1205.60**	0.0597	12
pr02	2224.47	0.3306	16	**2090.07**	0.7491	40
pr03	2760.97	1.3766	28	**2754.08**	1.8156	41
pr04	3602.42	2.2765	20	**3591.23**	2.8061	27
pr05	**3973.98**	4.6637	20	4056.60	6.1296	37
pr06	**4461.15**	6.8192	20	4613.89	10.3344	45
pr07	**1633.11**	0.0747	12	**1633.11**	0.1034	17
pr08	**2532.88**	0.5533	16	**2532.88**	0.7090	22
pr09	3464.47	1.8987	24	**3401.67**	3.2440	47
Average	**2873.23**	2.00	18.22	2875.46	2.88	32.00

Table 2 – Comparison of the four local search methods executed consecutively and by VND.

The comparison in Table 2 does not show a clear winner. The sequential local search structure was faster for every instance, where it has exhausted a lower number of created neighborhoods to reach the local optimum (a lower iter.), which is also justified by the fact that VND needs to go to the first neighborhood after improving a solution. The sequential local search also has reached a lower local minimum against the VND on 2 out of 9 instances (pr05 and pr06). However, the VND has reached a lower local minimum against the LS on 4 out of 9 instances (pr02, pr03, pr04, and pr09) and there were 3 ties (pr01, pr07, and pr08). In general, the VND achieved a higher amount of better local

minimums against the LS, but the LS was faster and also achieved a lower average solution on the analysis.

6 The RGRASP+VND algorithm

In this section, we describe the Reactive GRASP with variable neighborhood descent (VND) algorithm (RGRASP+VND), implemented to solve the MDVRP and MDVRPTW.

The Greedy Randomized Adaptive Search Procedures (GRASP) [37] is a multi-start metaheuristic with construction and local search procedures. In each GRASP iteration, a solution is built using a greedy randomized adaptive algorithm, which is then submitted to a local search procedure. The best overall solution is kept as the algorithm result. The GRASP construction procedure iteratively evaluates a set of candidate elements to be integrated into the solution, also checking feasibility. Those with the smallest incremental cost are inserted in an ordered list called the restricted candidate list (RCL), which is limited by the percentage parameter α. For instance, when $\alpha = 0.1$, the list contains only the 10% best candidates.

The next element selected to integrate the solution being constructed is randomly selected from RCL. In this way, one of the best RCL components is chosen to belong to the solution, but not necessarily the top candidate. The RCL is updated and its elements reevaluated until the solution is complete. If feasibility is not achieved, repairing procedures and/or new trials are activated, to obtain a feasible solution. The GRASP algorithm requires only two input parameters: the RCL size constraint α and the maximum number of GRASP iterations ($max\_iterations$) and can be easily implemented for a parallel scheme [35].

Reactive GRASP [38] considers not a fixed value for the RCL α parameter, but a random α with a probability selected from each GRASP iteration. The probabilities are computed and self-adjusted along the iterations. More formally, an initial value for α is selected from the discrete set $A = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$, with m predetermined α values. Let p_i be the probability associated with the choice of α_i, for $i = 1, 2, \cdots, m$. The initial values $p_i = 1/m$, for $i = 1, 2, \cdots, m$ correspond to a uniform distribution. Prais and Ribeiro [38], suggest to periodically update the p_i values for $i = 1, 2, \cdots, m$, taking into account information based on the average cost of the solutions obtained over the iterations. Let *block_iterations* be the number of iterations without probabilities update, $F(S^*)$ be the cost of the best solution found so far S^* and A_i the average of the solutions obtained in the iterations with $\alpha = \alpha_i$. After each GRASP *block_iterations*, the p_i probabilities are updated following the absolute qualification rule, defined as $p_i = \frac{q_i}{\sum_{j=1}^{m} q_j}$, with normalized values $q_i = \frac{F(S^*)}{A_i}^{\delta}$.

Reactive GRASP has as parameters, in addition to *block_iterations*, the parameter

δ, used to attenuate the p_i updated probabilities. The parameter δ is commonly adopted as $\delta = 10$, in a diversity of problems in the literature [38, 39, 40].

Reactive GRASP was implemented with the absolute qualification rule [38] to update the probabilities. However, this rule fails when not all predetermined values for α_i are selected along with the algorithm execution. Hence, no solutions average A_i and its respective normalized value q_i can not be generated. To settle this problem, one solution are generated for each α_i, for $i = 1, 2, \ldots, m$, as a initialization step. Thus, generating one solution for each value will not prejudice the rule, since the probability for selecting each α_i starts uniformly, The greedy randomized algorithm was obtained by adding a Restrict Candidate List (RCL) to the known Solomon Insertion Heuristic I1 [21]. The local search neighborhoods are described in Section 5, which are organized in the Variable Neighborhood Descent structure.

RGRASP+VND pseudocode is given by Algorithm 7. The algorithm receives as input the maximum number of iterations (*max_iterations*), the number of α values m, the number of iterations after which the set probabilities $p_i \in P, i = 1, 2, \ldots, m$, will be updated (*block_iterations*), the δ parameter, the graph $G = (V, E)$, the number of vertices N, the number of depots M, the maximum number of vehicles K for each depot and the set of vehicle capacities Q. The algorithms return the best solution (*BestSolution*) found for an MDVRP instance problem.

Algorithm 7: Reactive GRASP pseudocode

Input: *max_iterations*, m, *block_iterations*, δ, $G = (V, E)$, N, M, K, Q
Output: *BestSolution*

```
1  Clusters ← ClusterizeInstance(G, N, M, K, Q)
2  A ← ConstuctAlphaSet(m)
3  BlockSolutions ← generate one solution for each α_i from A
4  P ← UpdateAlphaProbabilities(BlockSolutions)
5  BlockSolutions ← ∅
6  for it ← 1 to max_iterations do
7      if it % block_iterations == 0 then
8          P ← UpdateAlphaProbabilities(BlockSolutions)
9          BlockSolutions ← ∅
10     end if
11     α ← select a random α_i from A considering the probabilities set P
12     Solution ← GreedyRandomizedSolution(Clusters, K, Q, α)
13     Solution ← VND(Solution)
14     BestSolution ← UpdateSolution(Solution, BestSolution)
15     BlockSolutions ← BlockSolutions ∪ Solution
16 end for
```

RGRASP+VND starts by clustering the customers set using `ClusterizeInstance` (line 1). The set A of α values is constructed in `ConstructAlphasSet`, that generates m

α values uniformly distributed in (0,0.9] (line 2). The absolute qualification rule problem is applied in lines3-4, by creating one solution for each $\alpha_i \in A, i = 1, \ldots, m$, (line 3) and updating the set of probabilities P (line 4). The history of solutions (*BlockSolutions*) is emptied in line 5 and all solutions of a block of iterations, used to update each probability $p_i \in P$ for $\alpha_i \in A$, $i = 1, \ldots, m$. Line 6 controls the algorithm *max_iterations*. Line 7 checks if the number of generated solutions is equal the *block_iterations*, by performing the rest of integer division (%) with the current iteration number (*it*) and the *block_iterations* number, if it succeeds, the probabilities $p_i \in P$ of selecting $\alpha_i \in A$ are updated, for $i = 1, \ldots, m$ by line 8 and the history of solutions (*BlockSolutions*) is emptied (line 9). Line 11 selects a random $\alpha_i \in A$, considering the probability of selection of each α_i, defined by P. Line 12 constructs a *Solution* with function `GreedyRandomizedSolution`, described in Algorithm 8. The local search neighborhoods described in Section 5 are applied in the solution using the VND structure (line 13). The best solution found is updated in line 14 and the solution is stored in the current block of iterations history (line 15).

The `GreedyRandomizedSolution` pseudocode is described in Algorithm 8. The algorithm receives as input the clustered instance (*Clusters*), the maximum number of vehicles K of each depot, the vehicle capacity set Q and the RCL size constraint α and returns a feasible *Solution*.

Algorithm 8: `GreedyRandomizedSolution` pseudocode

```
   Input: Clusters, K, Q, α
   Output: Solution
 1 Solution ← ∅
 2 Initialize the set of candidate elements
 3 Evaluate the incremental costs of the candidate elements
 4 while exists at least one candidate element do
 5 |   RCL ← ConstructRestrictedCandidateList(Clusters, K, Q, α)
 6 |   s ← Select a random element from RCL
 7 |   Solution ← Solution ∪ {s}
 8 |   Update the set of candidate elements
 9 |   Reevaluate the incremental costs
10 end while
11 if Solution is not feasible then
12 |   Solution ← Repair(Solution)
13 end if
```

The algorithm starts with an empty solution on line 1. The set of candidate elements is initialized in line 2 and their incremental costs are evaluated (3). While exists a candidate element, the RCL is constructed on line 5 with its size limited by the parameter α. One random element of RCL is chosen on line 6 and incorporated to the solution on line 7. The set of candidate elements is updated in line 8 and their incremental cost are

reevaluated (9). If the constructed solution is infeasible, a new solution is generated with the procedure `Repair`, where a new random solution is created (12).

7 Computational experiments and discussion

This section discusses the experimental results of the described algorithms. We tested the RGRASP+VND algorithm, considering as input, the Cordeau et al. (1997) benchmark dataset [4], available at the Vehicle Routing Problem Repository VRP-REP [1].

The RGRASP+VND was coded in Python 3.8 and the experiments were carried out on an Intel i5-2310 processor with 4GB RAM, running under a Linux system Ubuntu 18.04. The tests were carried out in 2 stages: the first, for parameter tuning, and the third, for comparison of RGRASP+VND with state-of-the-art algorithms.

Section 7.1 presents the RGRASP+VND parameters tuning. Section 7.2 presents the discussion from the comparison of RGRASP+VND with literature and state-of-the-art algorithms.

7.1 Algorithm strategies of procedures application

Preliminary tests for RGRASP+VND parameters tuning, for MDVRP and MDVRPTW problems were carried out. The results are presented in Section 7.1.1 and 7.1.2 respectively.

The RGRASP+VND basic procedures are: i) a clustering scheme (K-means or Urgencies); ii) a constructive algorithm (Solomon Insertion heuristic I1); iii) and the local search (Variable Neighborhood Descent with four neighborhoods). The four neighborhoods 2-swap, 2-opt intra routes, drop one point next depot and drop one point intra-depot are identified in this section respectively as T_1, T_2, T_3 and T_4 and performed each with the best improvement. Each neighborhoods is explored following the order of identification T_i, $i = 1, \ldots, 4$, into two ways: sequentially, denoted as LS(T_1, T_2, T_3, T_4) or combined by the VND algorithm, denoted as VND(T_1, T_2, T_3, T_4).

Eleven sets of values for the constructive algorithm Solomon Insertion heuristic I1 parameters were tested using as initialization criteria the farthest customer from the depot (FC) and the customer with the closest ending service time (CTW). Each set is presented and identified in Table 3 as strategy. The set of strategies modifies the α_1 and α_2 parameters, which changes the importance of distance and time windows, respectively, in the route construction.

[1] *Available at: http://www.vrp-rep.org/ (access date: 03/22/2022).*

Strategy	Solomon $(\alpha_1, \alpha_2, \mu, \lambda)$
r_0	(0.0, 1.0, 1, 1)
r_1	(0.1, 0.9, 1, 1)
r_2	(0.2, 0.8, 1, 1)
r_3	(0.3, 0.7, 1, 1)
r_4	(0.4, 0.6, 1, 1)
r_5	(0.5, 0.5, 1, 1)
r_6	(0.6, 0.4, 1, 1)
r_7	(0.7, 0.3, 1, 1)
r_8	(0.8, 0.2, 1, 1)
r_9	(0.9, 0.1, 1, 1)
r_{10}	(1.0, 0.0, 1, 1)

Table 3 – Eleven strategies used in the parametrization study.

7.1.1 The MDVRP scenario

We performed four different sequences of the procedures implemented. Since the MDVRP has no time window constraints, we have used the Solomon r_{10} strategy (which does not consider time windows in evaluation) as the constructive algorithm for the restricted candidate list application and the farthest customer as the initialization criteria. For the MDVRP application, the sequence of procedures considered are: PS_1) K-means clustering method $\rightarrow$ Solomon Insertion heuristic I1 $\rightarrow$ LS(T_1, T_2, T_3, T_4); PS_2) K-means clustering method $\rightarrow$ Solomon Insertion heuristic I1 $\rightarrow$ VND(T_1, T_2, T_3, T_4); PS_3) Urgencies clustering method $\rightarrow$ Solomon Insertion heuristic I1 $\rightarrow$ LS(T_1, T_2, T_3, T_4) and PS_4) Urgencies clustering method $\rightarrow$ Solomon Insertion heuristic I1 $\rightarrow$ VND(T_1, T_2, T_3, T_4). All the procedures sequences PS_1, PS_2, PS_3 and PS_4 were performed over ten instances (p01 to p10) of Cordeau et al. dataset [4]. For MDVRP, we have considered the size of the set A of α values in the Reactive GRASP $m = 100$, and the farthest unrouted customer (FC) as the Solomon Insertion heuristic I1 initialization criteria.

The results are presented in Table 4. The first column specifies the procedure sequence applied. The second column represents the clustering method utilized. The third column specifies the strategy adopted for Solomon Insertion heuristic I1. The fourth column specifies the local search strategy adopted. The fifth column depicts the average solution over 10 runs for eight instances of the Cordeau et al. dataset, for each procedure sequence.

Using the K-means clustering method, the Reactive GRASP did not generate any feasible solution for the instances p04 and p07. However, using the Urgencies clustering method the constructive method generates a feasible solution for all ten instances. To compare the average results only the eight instances in which both methods generate

	Cluster method	Local search	Average sol.
PS_1	K-means	LS	2245.17
PS_2	K-means	VND	2238.34
PS_3	Urgencies	LS	2093.78
PS_4	Urgencies	VND	**2092.80**

Table 4 – Average solution values on eight instances of the Cordeau et al. dataset [4] for MDVRP on the implemented procedures sequences.

feasible solutions are considered. For this case, the instances p04 and p07 are not considered in Table 4 for the average solution evaluation.

The algorithm has reached better average solutions with the Urgencies clustering method against the K-means clustering method for the procedure sequences. The VND achieved better results than the sequential local search. Thus, we have selected the Urgencies clustering method and the VND local search strategy. Ultimately, the parameters used on this algorithm for the experimental tests on Cordeau et al. dataset [4] for the MDVRP are presented in Table 5. The first column specifies the clustering method adopted. The second column specifies the maximum number of iterations of the algorithm. The third column specifies the δ parameter, used to attenuate the Reactive GRASP probabilities set P. The fourth column specifies the size of the set A of α values in the Reactive GRASP. The fifth column specifies the number of iterations necessary to update the Reactive GRASP probabilities set P.

The *max_iterations* and m input parameters was set to 10,000 and 100, respectively, for presenting good results in preliminary tests. The *block_iterations* input parameter was set to 100 as used in the original proposal of Reactive GRASP algorithm [38] and the parameter δ is commonly adopted as $\delta = 10$, in a diversity of problems in the literature [38, 39, 40].

Clustering method	Solomon initialization criteria	Solomon strategy	*max_iterations*	δ	m	*block_iterations*
Urgencies	FC	r_{10}	10,000	10	100	100

Table 5 – RGRASP+VND decisions for the experimental tests on MDVRP instances.

7.1.2 The MDVRPTW scenario

The preliminary test strategy adopted for the RGRASP+VND algorithm applied on MDVRPTW considers comparing the clustering methods K-means and Urgencies clustering methods with the two initialization criteria (FC and CTW) for the Solomon Insertion heuristic I1. The $r_i, i = 0, ..., 10$ strategies presented in Table 3, also were tested

on the first ten instances (pr01-pr10) of the Cordeau et al. dataset [22]. For each strategy, 10 RGRASP+VND runs were performed, considering the first 10 integers as seeds. Table 6 presents the RGRASP+VND parameters used for the RGRASP+VND for the experimental tests combining the clustering methods and the Solomon insertion I1 heuristic. In total, 1100×4 RGRASP+VND runs were performed.

Solomon strategy	*max_iterations*	δ	m	*block_iterations*
$[r_0, r_1, \ldots, r_{10}]$	10	10	10	5

Table 6 – Algorithm parameter setting for the clustering and constructive study on Cordeau et al. dataset.

The Urgencies with the CTW did not generate a feasible solution for instances pr04 and pr05 for a group of Solomon strategies and were not considered for the average solution cost evaluation. The K-means clustering method did not generate any feasible solution for the instance pr10. To compare the average solution among the four procedure sequences analyzed, the instance pr10 was not considered for the evaluation.

Figure 4 and 5 show, respectively, the graphics of the average solutions and the best solution found of ten runs, respectively, for the nine instances pr01-pr09. Each figure has three lines, that describe the clustering method and the Solomon Insertion heuristic I1 initialization criteria chosen. The horizontal axis represents the Solomon Insertion heuristic I1 strategy according to Table 3 and the vertical axis represents the average solution cost. Figure 4 presents the average solution cost of all instances, over the average solution cost in 10 runs for each. Figure 5 presents the average solution cost of all instances, over the best solution cost in 10 runs for each.

Table 7 presents the number of feasible solutions achieved by each procedure. The best result was achieved using the Urgencies clustering method with FC. The Urgencies was the only clustering method that presented a feasible solution for the instance pr10. The Urgencies with FC have found a feasible solution on all executions on instance pr10 and was the procedure with the highest number of feasible solutions. The K-means clustering method presented stable results for most instances and the FC initialization criteria got a feasible solution on all executions and instances, except for the instance pr10. The worst result obtained was for the Urgencies with CTW. It found only 665 feasible solutions out of 1100 executions, followed by K-means with CTW with 915 feasible solutions, K-means with FC, with 990 feasible solutions, and Urgencies with FC, with 1068 feasible solutions. Overall, the FC presented the highest number of feasible solutions against the CTW and K-means presented more stable results, but the best and worst results were achieved by Urgencies.

The MDVRPTW associates two constraints that can diverge. Choosing the closest

Instance	K-means		Urgencies	
	FC	CTW	FC	CTW
p01	110	110	110	110
p02	110	110	100	104
p03	110	110	110	110
p04	110	94	106	4
p05	110	51	109	2
p06	110	110	93	3
p07	110	110	110	110
p08	110	110	110	110
p09	110	110	110	110
p10	0	0	110	2
Sum	990	915	1068	665

Table 7 – Number of feasible solutions achieved.

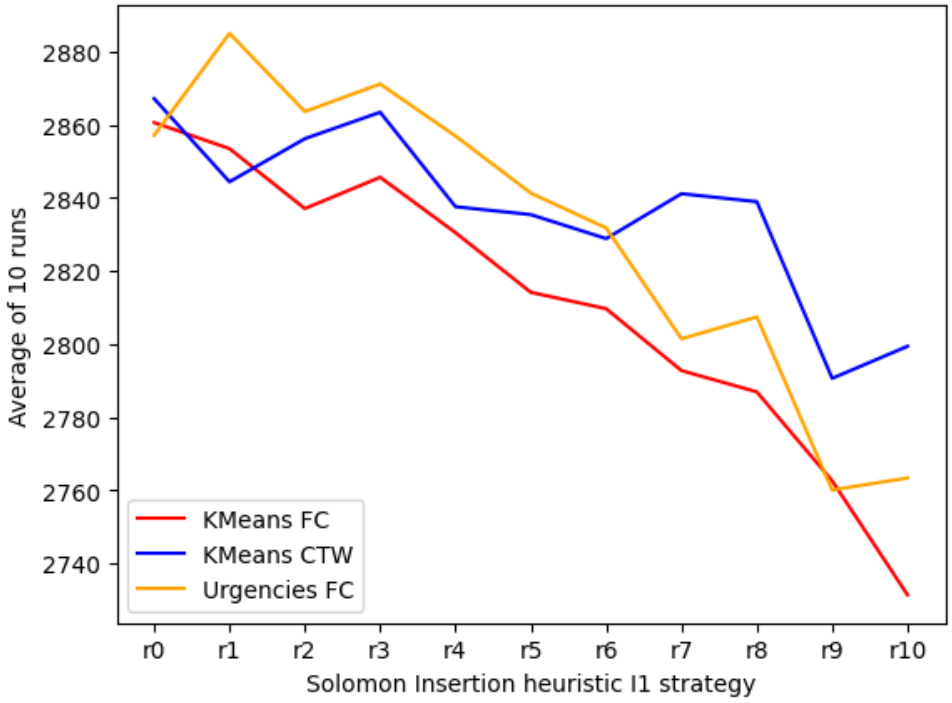


Figure 4 – Average solutions of the RGRASP+VND parametrization analysis.

customer on the clustering method can lead to a lower final cost, as evidenced in Figure 4, where, in general, more importance is given to distance criteria, the lowest the average cost obtained. However, closest customers can have very different time windows, and optimizing distance can turn the time window constraint infeasible. This can be seen in Figure 4, where the strategy with the lowest average cost (K-means FC with Solomon strategy=r_{10}) did not make any feasible solution for the instance pr10 (see Table 7).

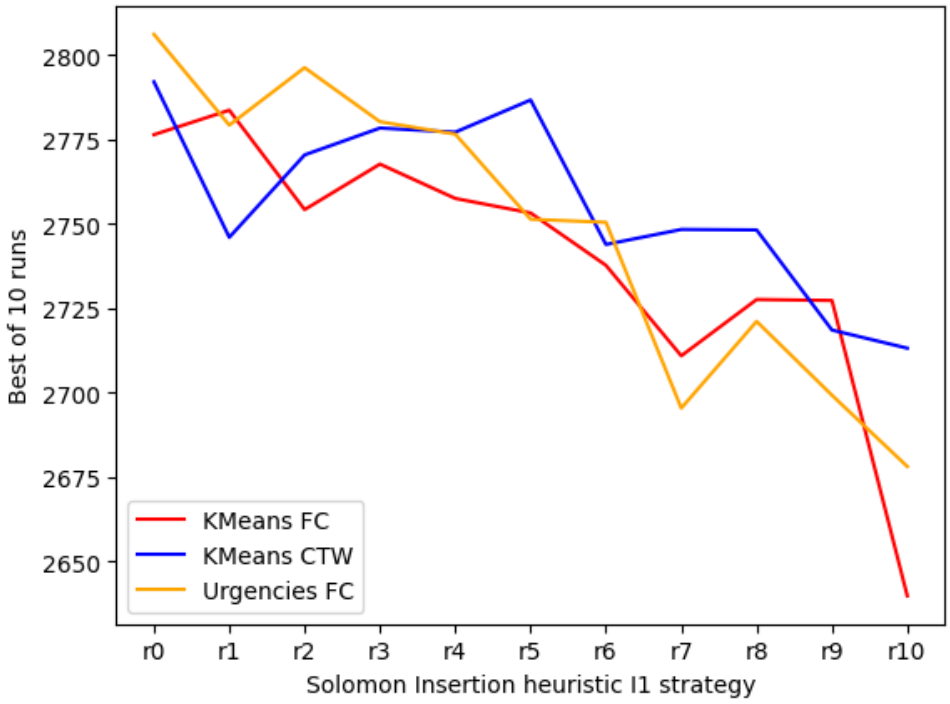


Figure 5 – Best solutions of RGRASP+VND parametrization analysis.

Using the Urgencies clustering method with Solomon strategy=r_9 and farthest customer as initialization criteria, it has constructed a feasible solution in all executions for the instance pr10 and a higher amount of feasible solutions overall (see Table 7). Seeking a lower average solution while still solving hard instances, we chose to divide the algorithm number of iterations in half with these two approaches.

Ultimately, the parameters used on this algorithm for the experimental tests on Cordeau et al. dataset [22] for the MDVRPTW are presented in Table 8. The first column specifies the clustering method adopted. The second column specifies the Solomon Insertion heuristic I1 initialization criteria. The third column specifies the Solomon strategy. The fourth column specifies the δ parameter, used to attenuate the Reactive GRASP probabilities set P. The fifth column specifies the size of the set A of α values in the Reactive GRASP. The sixty columns specify the number of iterations necessary to update the Reactive GRASP probabilities set P.

The *max_iterations* and m input parameters was set to 10,000 and 100, respectively, for presenting good results in preliminary tests. The *block_iterations* input parameter was set to 100 as used in the original proposal of Reactive GRASP algorithm [38] and the parameter δ is commonly adopted as $\delta = 10$, in a diversity of problems in the literature [38, 39, 40].

Clustering method	Solomon initialization criteria	Solomon strategy	*max_iterations*	δ	m	*block_iterations*
K-means	FC	r_{10}	10,000	10	100	100
Urgencies	FC	r_9	10,000	10	100	100

Table 8 – RGRASP+VND decisions for the experimental tests on MDVRPTW instances.

7.2 Computational results

In this section, the experiment results of the RGRASP+VND algorithm, over the Cordeau et al. instances, are discussed.

7.2.1 MDVRP results discussion

A literature review was conducted to identify the application of the metaheuristic GRASP for the solution of the MDVRP literature. In this study, the search string: *"grasp" AND ("mdvrp" OR "multi-depot vehicle routing problem")* was used in databases Scopus, IEEE Xplore, and Web of Science up to March 2022. Only two distinct papers were returned and only one (GRASP/VND) [2] presented results considering instances from the literature.

RGRASP+VND and GRASP/VND consider the VND local search procedure in the GRASP metaheuristic. However, there are several differences between them. The GRASP/VND uses the route-first cluster-second approach (see Section 4), and the classical GRASP version, where α parameter calibration is necessary (see Section 6). The RGRASP+VND considers the cluster-first route-second approach and the Reactive GRASP, with a dynamic update of the α parameter, is performed automatically. Moreover, the set of local search neighborhoods employed for the VND structure is different for both algorithms.

Table 9 compares the RGRASP+VND and the GRASP/VND algorithms. The Instance column identifies the instance, the BKS column presents the best-known results for the instance, and those with * are optimum, as proved by R. Baldacci and A. Mingozzi [41]. The following columns show the algorithm results: the column *Avg.* presents the average solution over 10 runs; *Best* presents the best solution over 10 runs and *%Dev* indicates the algorithm percentage deviation from BKS, which is evaluated for each instance as $\frac{Best-BKS}{BKS} \times 100$.

Better results were obtained with RGRASP+VND for the instances p01, p02, p04, and p06. On the other hand, GRASP/VND outperforms the RGRASP+VND for instances p03, p07, p12, p15, and p21. We noticed that for smaller instances the RGRASP+VND outperforms GRASP/VND, while the opposite occurs for larger instances. The most remarkable result achieved by RGRASP+VND was for the instance p02. The best so-

		GRASP/VND [2]			RGRASP+VND		
Instance	BKS	*Avg.*	*Best*	%Dev	*Avg.*	*Best*	%Dev
p01	576.87*	610.79	592.21	2.66	588.53	**581.10**	**0.73**
p02	473.53*	556.35	529.64	11.85	480.35	**480.35**	**1.44**
p03	640.65*	694.08	**648.68**	**1.25**	664.52	660.32	3.07
p04	999.21*	1071.49	1055.26	5.61	1056.64	**1051.10**	**5.19**
p05	751.26	803.93	**769.37**	**2.41**	794.68	789.89	5.14
p06	876.5*	963.10	924.68	5.50	904.38	**901.67**	**2.87**
p07	881.97*	955.76	**925.80**	**4.97**	940.96	931.09	5.57
p12	1318.95*	1409.02	**1326.85**	**0.60**	1363.32	1332.71	1.04
p15	2505.42	2809.46	**2553.80**	**1.93**	2699.66	2649.15	5.74
p21	5474.84	6187.00	**5903.63**	**7.83**	6174.64	6135.68	12.07

* proved optimality [41].

Table 9 – Algorithms solution comparison for the MDVRP instances.

lution achieved for this instance was only 6.82 units far from the BKS (%Dev=1.44) against GRASP/VND, whose best result is 56.11 units far from the BKS (%Dev=11.85), with a difference of 49.29 units between the algorithms. The worst result obtained by RGRASP+VND was for the biggest instance p21, with the best solution value of 660.84 units far from the BKS (%Dev=12.07), against GRASP/VND, with 428.79 units far from the BKS (%Dev=7.83), with a difference of 232.05 units between the algorithms.

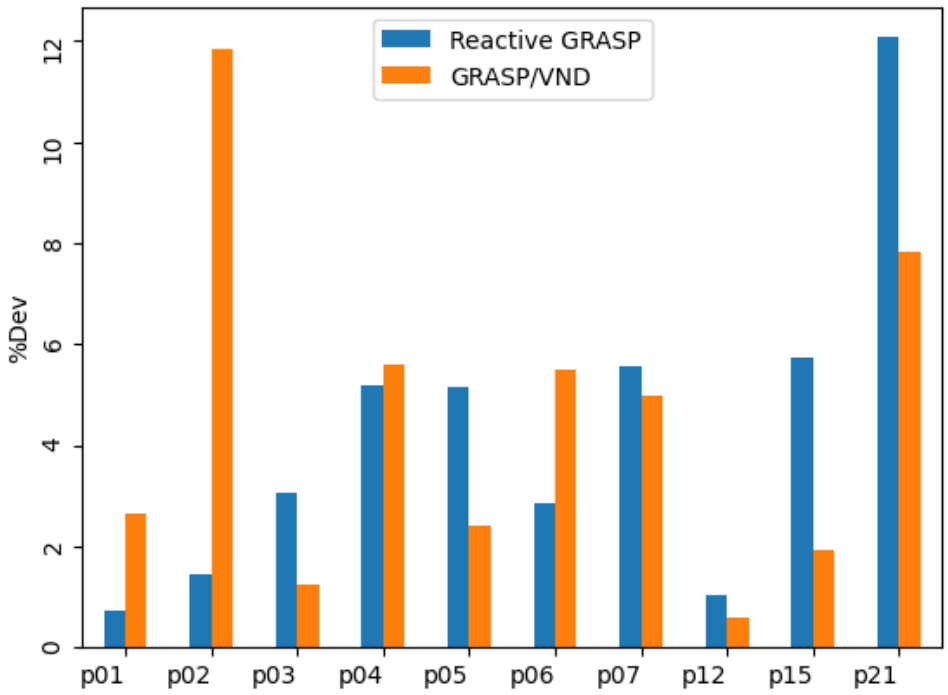


Figure 6 – Comparison of deviation percentage for the RGRASP+VND and the GRASP/VND [2].

Figure 6 presents for each instance, the RGRASP+VND and GRASP/VND percentage deviations from BKS. The highest difference achieved was for the instance p02,

where RGRASP+VND outperforms GRASP/VND, with a 10.41% deviation difference. For instance p01 and p04, RGRASP+VND also outperforms GRASP/VND, but with a small deviation percentage difference (1.93% and 0.42% respectively). For the instance p06, RGRASP+VND performed better than GRASP/VND, with a 2.63% deviation difference. For the remainder instances p03, p05, p07, p12, p15, p21, GRASP/VND outperforms RGRASP+VND, with 1.82%, 2.73%, 0.60%, 0.44%, 3.81%, 4.24% deviation differences respectively.

RGRASP+VND algorithm brings contributions to GRASP literature for the MDVRP. There are improvements over the GRASP/VND algorithm for most small instances of the Cordeau et al. dataset. We also highlight the robustness of RGRASP+VND, where it presents a small difference between the average and best solution costs (the highest difference was obtained on instance p15 of 50.51 units). It also presents better average results for every instance on the experimental tests when compared to GRASP/VND (see Table 9).

A literature review was also conducted to study the MDVRP state-of-the-art. In this study, the search string: *("mdvrp" OR "multi-depot vehicle routing problem') AND ("metaheuristic" OR "meta-heuristic" OR "exact algorithm" OR "matheuristic" OR "column generation" OR "hybrid metaheuristic")* was used on Scopus database limited to years 2018 to March 2022. The algorithms returned as well as those chosen for comparison in their work, were ranked considering their average solution cost on the Cordeau et al. The two methods returned by the literature review, with the best average results were: A Hybrid Genetic Algorithm (HGSADC) [42] and A biased-randomized variable neighborhood search (BR-VNS) [15]. Both methods outperform the reported GRASP results (GRASP/VND and RGRASP+VND) for the Cordeau et al. dataset. HGSADC algorithm was later extended in literature in a unified solution framework [43] to solve many VRP variants.

Table 10 brings the comparison of RGRASP+VND with the state-of-the-art algorithms. HGSADC [42] and BR-VNS [15]. HGSADC achieves the best results, obtaining the BKS for all instances of the experimental tests, excluding the p03 and p04, and their average deviation was lower than 0.20% for all instances. The BR-VNS had the second-best results and their average deviation was lower than 2%. The RGRASP+VND average deviation reached 5.27% from BKS and 5.24 and 4.90, respectively from HGSADC and BR-VNS.

7.2.2 MDVRPTW results comparison

A literature review was conducted to identify the MDVRP state-of-the-art. In this study, the search string: *("mdvrptw" OR "multi-depot vehicle routing problem with*

		HGSADC [42]		BR-VNS [15]		RGRASP+VND	
Instance	BKS	*Avg.*	%Dev	*Avg.*	%Dev	*Avg.*	%Dev
p01	576.87*	**576.87**	0.00	**576.87**	0.00	588.53	2.02
p02	473.53*	**473.53**	0.00	**473.87**	0.07	480.35	1.44
p03	640.65*	**641.19**	0.08	**641.19**	0.08	664.52	3.73
p04	999.21*	**1001.04**	0.18	1003.49	0.43	1056.64	5.75
p05	751.26	**750.03**	0.00	751.94	0.25	794.68	5.95
p06	876.5*	**876.5**	0.00	**876.5**	0.00	904.38	3.18
p07	881.97*	**881.97**	0.00	885.74	0.43	940.96	6.69
p12	1318.95*	**1318.95**	0.00	**1318.95**	0.00	1363.32	3.36
p15	2505.42	**2505.42**	0.00	2511.43	0.24	2699.66	7.75
p21	5474.84	**5474.84**	0.00	5576.25	1.85	6174.64	12.78

* proved optimality [41].

Table 10 – RGRASP+VND comparison with state-of-art methods for the MDVRP instances.

time windows") AND ("metaheuristic" OR "meta-heuristic" OR "exact algorithm" OR "matheuristic" OR "hybrid metaheuristic" OR "column generation") was used on Scopus, IEEE Xplore, and Web of Science databases up to March 2022. The algorithms returned as well as those chosen for comparison in their works, were ranked considering their average results considering the Cordeau et al. dataset of 2001, which is a widely used MDVRPTW benchmark.

PACOT [44] was published in 2009 and was the reported algorithm with the best results. However, it does not present details for any instance solution. It presents a high difference in cost solutions against the literature algorithms. For example, excluding PACOT, the literature best know solution has 151.34 units of the higher cost compared to PACOT (1074.12 against 922.78 from PACOT) on the first instance pr01 and a similar behavior occurs in the other instances. For the lack of details as not a single solution is plotted or described, and the lack of citations (PACOT has only one citation from a different journal, which was made from a survey)[2] we will exclude PACOT from the comparison. Thus, VNS [3] was the selected algorithm with the overall best results. For the comparison with RGRASP+VND, the algorithms: Improved firefly [45] and High-Level Relay Hybrid Metaheuristic [46] were also considered. The Improved firefly [45] is a very recent work, published in 2022, and the High-Level Relay Hybrid Metaheuristic [46] has presented a better result for two instances against the VNS. The GRASP metaheuristic was not presented by any algorithm on the search and could not be compared with RGRASP+VND.

Table 11 presents the algorithms comparison with the Cordeau et al. dataset [22]. The algorithms are compared with the VNS, which is the method with the overall best

[2] checked on 06/04/2022.

results found in the literature. The percentage deviation (%Dev) considers the algorithm best solution and VNS [3] best solution. However, the algorithm HLRH [46] only presents the average result, and for this reason, it was used instead. The algorithm IF [45] does not report if the results are the average or the best, and the only reported results are also used for the percentage deviation evaluation. Figure 7 presents the algorithms deviation percentage with VNS [3] best solution.

	VNS [3]		HLRH [46]		IF [45]		RGRASP+VND		
	Avg.	*Best.*	*Avg.*	%Dev		%Dev	*Avg.*	*Best*	%Dev
pr01	1074.12	1074.12	1101.80	2.58	1083.98	0.92	1117.74	1102.80	2.67
pr02	1763.66	1762.21	1762.21	0.00	1763.02	0.05	1936.55	1909.62	8.36
pr03	2388.73	2373.65	2408.42	1.46	2408.43	1.47	2623.43	2606.15	9.80
pr04	2847.56	2815.48	2858.20	1.52	2958.21	5.07	3163.38	3095.40	9.94
pr05	3015.27	2993.94	3029.65	1.19	3134.05	4.68	3367.78	3322.15	10.96
pr06	3674.60	3629.72	3758.36	3.54	3904.16	7.56	3997.58	3953.22	8.91
pr07	1418.22	1418.22	1418.22	0.00	1423.33	0.36	1474.60	1465.23	3.31
pr08	2103.21	2096.73	2103.89	0.34	2150.22	2.55	2276.20	2263.93	7.97
pr09	2753.61	2730.54	2737.82	0.27	2831.95	3.71	3106.41	3084.13	12.95
pr10	3541.01	3499.56	3577.28	2.22	3714.77	6.15	4152.99	4067.91	16.24
pr13	2012.37	2001.83	2014.02	0.61	2020.66	0.94	2207.04	2138.08	6.81
pr14	2239.02	2215.51	2202.08	-0.61	2247.7	1.45	2794.95	2638.29	19.08
pr16	2909.45	2896.03	2896.03	0.00	2943.72	1.65	3435.47	3381.41	16.76
pr18	1809.25	1796.21	1792.61	-0.20	1809.35	0.73	2438.25	2438.25	35.74
pr19	2294.19	2292.45	2310.92	0.81	2310.92	0.81	2800.00	2753.36	20.11
pr20	3093.51	3076.37	3079.73	0.11	3131.89	1.80	4173.60	4102.84	33.37
Average	2433.61	2417.04	2440.70	0.87	2489.77	2.49	2816.62	2770.17	13.94

Table 11 – Algorithms comparison for the Cordeau et al. MDVRPTW instances.

RGRASP+VND did not generate any feasible solution on instances pr11, pr12, pr15, and pr17 and these instances were not presented in Table 11. VNS achieved better results in 14 out of 16 instances of Table 11 and is the algorithm with the best results in the literature. HLRH outperformed VNS on instances pr14 and pr18 and was the second-best algorithm in the comparison, followed by IF. HLRH achieved 0.87 and 2.49 percentage deviation, respectively, from VNS, and RGRASP+VND achieved 13.94 percentage deviation from VNS. Overall, RGRASP+VND achieved the highest percentage deviation results on instances pr11-pr20, excluding pr13, where the percentage deviations were higher than 12%. These results can be related to the difficulty RGRASP+VND presented for generating a feasible solution for the instances, as shown in Table 12. Instance pr13 was the only instance that had a feasible solution on all executions and also the lowest %Dev in Table 11.

Table 12 presents the number of executions that found a feasible solution on the experimental tests of RGRASP+VND, reported in Table 11. A higher amount of feasible

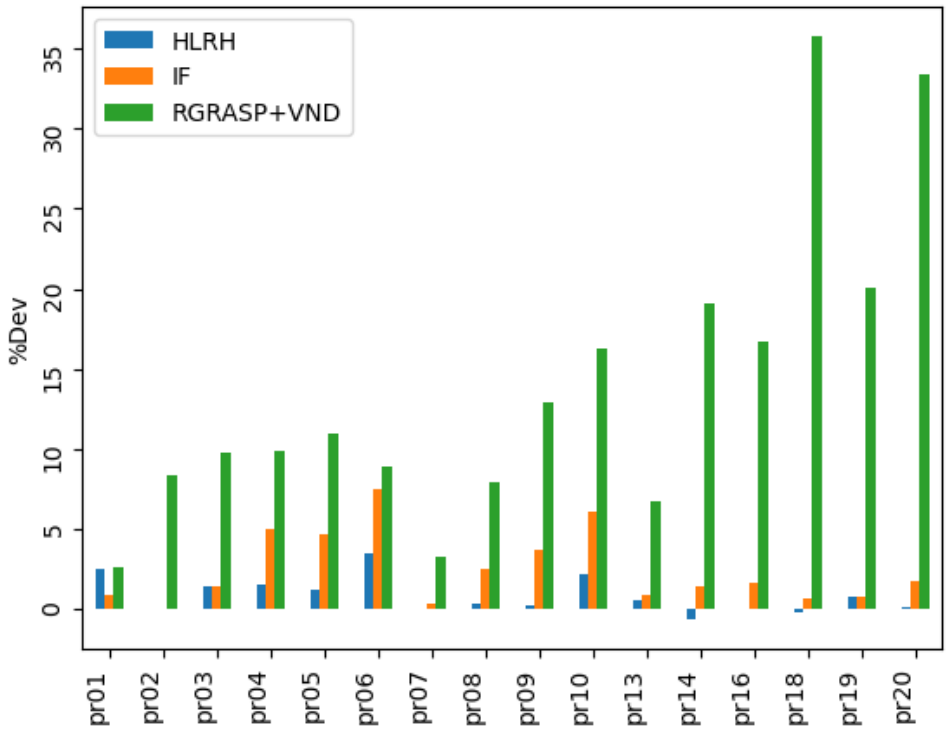


Figure 7 – Comparison of algorithms deviation percentage with VNS [3].

Instance	K-means	Urgencies	Instance	K-means	Urgencies
pr01	10	10	pr11	0	0
pr02	10	10	pr12	0	0
pr03	10	10	pr13	10	10
pr04	10	10	pr14	0	6
pr05	10	10	pr15	0	0
pr06	10	10	pr16	0	10
pr07	10	10	pr17	0	0
pr08	10	10	pr18	0	1
pr09	10	10	pr19	0	10
pr10	0	10	pr20	0	2

Table 12 – Number of executions that RGRASP+VND found a feasible solution Cordeau et al. dataset for MDVRPTW considering both clustering methods.

solutions were obtained for the instances pr01-pr10 compared to pr11-pr20 instances. For the instances pr11, pr12, pr15, and pr17 the algorithm did not find any feasible solution. For the instances pr14, pr16, pr18, pr19, and pr20 the algorithm only constructed a feasible solution using the Urgencies clustering method. However, a few feasible solutions were constructed for the instances pr14, pr19, and pr20 (6, 1, 2, respectively).

The Urgencies clustering method with Solomon strategy=r_9 was not sufficient to solve the most challenging instances of the Cordeau et al. dataset. We noticed that the parametrization analysis that was conducted using the instances pr01-pr10 was not sufficient to solve the pr11-pr20 instances of the dataset. Thus, the selected subset was

not representative of the complete set. For the instances pr11-pr20, a higher percentage deviation was obtained compared to pr01-pr10 instances. Another study of feasibility construction needs to be conducted, considering a different set of instances, since using pr01-pr10 instances was not presented as sufficient to represent the dataset complexity and calibrate the RGRASP+VND parameters for the MDVRPTW.

8 Conclusions

This work introduced the RGRASP+VND algorithm, a Reactive GRASP metaheuristic with VND local search strategy for the MDVRP and MDVRPTW, combining four distinct local search neighborhoods, two of them presented in this work, and a clustering technique. GRASP literature for those VRP variants is minimal (only two papers were obtained in the review) and according to the literature review to identify state-of-the-art algorithms, we observed that GRASP metaheuristic is not top-ranked for their solution for those VRP variants. However, the RGRASP+VND algorithm brings contributions to the GRASP literature of the MDVRP. It achieves better results on most small instances of the Cordeau et al. dataset [4] and also achieves better average solutions for all instances of this set, when compared with results of the GRASP/VND [2] algorithm (see Table 9). In the MDVRP variant, RGRASP+VND achieves an average of 5.24 percentage deviation of HGSADC [42] and an average of 4.90 percentage deviation of BR-VNS [15] solutions (state-of-the-art algorithms). In the MDVRPTW variant, RGRASP+VND achieves an average of 13.94 percentage deviation of VNS [3], the state-of-the-art algorithm for MDVRPTW.

The RGRASP+VND was also introduced in the MDVRPTW literature. The major limitation of RGRASP+VND was in the generation of feasible solutions for the MDVRPTW variant (it did not generate a feasible solution for 4 out of 20 instances of the Cordeau et al. dataset for the MDVRPTW). It also did not generate a feasible solution for all executions in 3 out of 20 instances. We noticed that the parameter analysis that was conducted using the instances pr01-pr10 for the MDVRPTW was not sufficient to achieve more competitive solutions to the hardest instances of the dataset, and this subset was not representative of the complete set. Thus, for the instances pr11-pr20, a higher percentage deviation was obtained compared to pr01-pr10 instances. A more in-depth study of the feasibility of solutions should be carried out considering a more representative set of instances. Also, other clustering techniques can be investigated.

For future work, we point out the application of other constructive procedures for the RGRASP+VND algorithm, a broader parametrization study as well as the investigation of more local search procedures. Other clustering methods to generate a higher amount of feasible solutions should also be investigated. The application of RGRASP+VND could also be investigated on other VRP variants, for instance, those with pickup and delivery constraints.

Bibliography

1 ELSHAER, R.; AWAD, H. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering*, Elsevier, v. 140, p. 106242, 2020. Cited 5 times on pages 8, 15, 16, 17, and 18.

2 VILLEGAS, J. G. et al. Grasp/vnd and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 23, n. 5, p. 780–794, 2010. Cited 5 times on pages 8, 13, 45, 46, and 52.

3 POLACEK, M. et al. A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. *Business Research*, Springer, v. 1, n. 2, p. 207–218, 2008. Cited 5 times on pages 8, 48, 49, 50, and 52.

4 CORDEAU, J.-F.; GENDREAU, M.; LAPORTE, G. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks: An International Journal*, Wiley Online Library, v. 30, n. 2, p. 105–119, 1997. Cited 7 times on pages 9, 14, 24, 39, 40, 41, and 52.

5 LENSTRA, J. K.; KAN, A. R. Complexity of vehicle routing and scheduling problems. *Networks*, Wiley Online Library, v. 11, n. 2, p. 221–227, 1981. Cited on page 13.

6 DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. *Management science*, Informs, v. 6, n. 1, p. 80–91, 1959. Cited 2 times on pages 13 and 15.

7 CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, Informs, v. 12, n. 4, p. 568–581, 1964. Cited 4 times on pages 13, 15, 26, and 27.

8 BEASLEY, J. E. Route first—cluster second methods for vehicle routing. *Omega*, Elsevier, v. 11, n. 4, p. 403–408, 1983. Cited 2 times on pages 13 and 23.

9 COMERT, S. E. et al. A cluster first-route second approach for a capacitated vehicle routing problem: a case study. *International Journal of Procurement Management*, Inderscience Publishers (IEL), v. 11, n. 4, p. 399–419, 2018. Cited 2 times on pages 13 and 23.

10 LUO, J.; CHEN, M.-R. Multi-phase modified shuffled frog leaping algorithm with extremal optimization for the mdvrp and the mdvrptw. *Computers & Industrial Engineering*, Elsevier, v. 72, p. 84–97, 2014. Cited on page 13.

11 HE, Y. et al. A tabu search algorithm with variable cluster grouping for multi-depot vehicle routing problem. In: IEEE. *Proceedings of the 2014 IEEE 18th international conference on computer supported cooperative work in design (CSCWD)*. [S.l.], 2014. p. 12–17. Cited on page 13.

12 PRINS, C. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & operations research*, Elsevier, v. 31, n. 12, p. 1985–2002, 2004. Cited on page 13.

13 LABADI, N.; PRINS, C.; REGHIOUI, M. A memetic algorithm for the vehicle routing problem with time windows. *RAIRO-Operations research*, EDP Sciences, v. 42, n. 3, p. 415–431, 2008. Cited on page 13.

14 BAGHBADORANI, R. R. et al. A novel two-phase approach to solve multi-depot vehicle routing problem. In: IEEE. *2021 25th International Conference on System Theory, Control and Computing (ICSTCC)*. [S.l.], 2021. p. 390–394. Cited on page 13.

15 REYES-RUBIANO, L. et al. A biased-randomized variable neighborhood search for sustainable multi-depot vehicle routing problems. *Journal of Heuristics*, Springer, v. 26, n. 3, p. 401–422, 2020. Cited 4 times on pages 13, 47, 48, and 52.

16 BARMA, P. S.; DUTTA, J.; MUKHERJEE, A. A 2-opt guided discrete antlion optimization algorithm for multi-depot vehicle routing problem. *Decision Making: Applications in Management and Engineering*, v. 2, n. 2, p. 112–125, 2019. Cited on page 13.

17 MISNI, F.; LEE, L. Harmony search for multi-depot vehicle routing problem. *Malaysian Journal of Mathematical Sciences*, v. 13, n. 3, p. 311–328, 2019. Cited on page 13.

18 EZUGWU, A. E. et al. Enhanced intelligent water drops algorithm for multi-depot vehicle routing problem. *PloS one*, Public Library of Science San Francisco, CA USA, v. 13, n. 3, p. e0193751, 2018. Cited on page 13.

19 MA, Y. et al. An improved aco for the multi-depot vehicle routing problem with time windows. In: SPRINGER. *Proceedings of the Tenth International Conference on Management Science and Engineering Management*. [S.l.], 2017. p. 1181–1189. Cited 3 times on pages 13, 19, and 21.

20 OSTERTAG, A. et al. Popmusic for a real-world large-scale vehicle routing problem with time windows. *Journal of the Operational Research Society*, Taylor & Francis, v. 60, n. 7, p. 934–943, 2009. Cited on page 13.

21 SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, Informs, v. 35, n. 2, p. 254–265, 1987. Cited 4 times on pages 14, 26, 31, and 36.

22 CORDEAU, J.-F.; LAPORTE, G.; MERCIER, A. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, Springer, v. 52, n. 8, p. 928–936, 2001. Cited 5 times on pages 14, 31, 42, 44, and 48.

23 TOTH, P.; VIGO, D. *The vehicle routing problem*. [S.l.]: SIAM, 2002. Cited on page 15.

24 YU, B.; YANG, Z.; XIE, J. A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society*, Taylor & Francis, v. 62, n. 1, p. 183–188, 2011. Cited on page 16.

25 KARAKATIČ, S.; PODGORELEC, V. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, Elsevier, v. 27, p. 519–532, 2015. Cited on page 16.

26 ZHANG, Y. et al. A generalized multi-depot vehicle routing problem with replenishment based on localsolver. *International Journal of Industrial Engineering Computations*, v. 6, n. 1, p. 81–98, 2015. Cited on page 16.

27 BRAEKERS, K.; RAMAEKERS, K.; NIEUWENHUYSE, I. V. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, Elsevier, v. 99, p. 300–313, 2016. Cited on page 16.

28 BOUSSAÏD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. *Information sciences*, Elsevier, v. 237, p. 82–117, 2013. Cited on page 17.

29 KULKARNI, R.; BHAVE, P. R. Integer programming formulations of vehicle routing problems. *European journal of operational research*, Elsevier, v. 20, n. 1, p. 58–67, 1985. Cited 2 times on pages 19 and 20.

30 MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. [S.l.], 1967. v. 1, n. 14, p. 281–297. Cited on page 23.

31 SINGH, D.; REDDY, C. K. A survey on platforms for big data analytics. *Journal of big data*, Springer, v. 2, n. 1, p. 1–20, 2015. Cited on page 23.

32 GIOSA, I.; TANSINI, I.; VIERA, I. New assignment algorithms for the multi-depot vehicle routing problem. *Journal of the operational research society*, Taylor & Francis, v. 53, n. 9, p. 977–984, 2002. Cited 2 times on pages 23 and 25.

33 CROES, G. A. A method for solving traveling-salesman problems. *Operations research*, INFORMS, v. 6, n. 6, p. 791–812, 1958. Cited on page 29.

34 FLOOD, M. M. The traveling-salesman problem. *Operations research*, INFORMS, v. 4, n. 1, p. 61–75, 1956. Cited on page 29.

35 POTVIN, J.-Y.; GENDREAU, M. *Handbook of Metaheuristics*. [S.l.]: Springer, 2018. Cited 3 times on pages 29, 31, and 35.

36 MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. *Computers & operations research*, Elsevier, v. 24, n. 11, p. 1097–1100, 1997. Cited on page 31.

37 FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of global optimization*, Springer, v. 6, n. 2, p. 109–133, 1995. Cited on page 35.

38 PRAIS, M.; RIBEIRO, C. C. Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, INFORMS, v. 12, n. 3, p. 164–176, 2000. Cited 4 times on pages 35, 36, 41, and 44.

39 BOUDIA, M.; LOULY, M. A. O.; PRINS, C. A reactive grasp and path relinking for a combined production–distribution problem. *Computers & Operations Research*, Elsevier, v. 34, n. 11, p. 3402–3419, 2007. Cited 3 times on pages 36, 41, and 44.

40 IORI, M. et al. Reactive grasp-based algorithm for pallet building problem with visibility and contiguity constraints. In: SPRINGER. *International Conference on Computational Logistics*. [S.l.], 2020. p. 651–665. Cited 3 times on pages 36, 41, and 44.

41 BALDACCI, R.; MINGOZZI, A. A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, Springer, v. 120, n. 2, p. 347–380, 2009. Cited 3 times on pages 45, 46, and 48.

42 VIDAL, T. et al. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, INFORMS, v. 60, n. 3, p. 611–624, 2012. Cited 3 times on pages 47, 48, and 52.

43 VIDAL, T. et al. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research*, Elsevier, v. 234, n. 3, p. 658–673, 2014. Cited on page 47.

44 RAJMOHAN, M.; SHAHABUDEEN, P. Metaheuristic for solving routing problem in logistics management. *International Journal of Operational Research*, Inderscience Publishers, v. 6, n. 2, p. 223–246, 2009. Cited on page 48.

45 YESODHA, R.; AMUDHA, T. A bio-inspired approach: Firefly algorithm for multi-depot vehicle routing problem with time windows. *Computer Communications*, Elsevier, v. 190, p. 48–56, 2022. Cited 2 times on pages 48 and 49.

46 NOORI, S.; GHANNADPOUR, S. F. High-level relay hybrid metaheuristic method for multi-depot vehicle routing problem with time windows. *Journal of Mathematical Modelling and Algorithms*, Springer, v. 11, n. 2, p. 159–179, 2012. Cited 2 times on pages 48 and 49.