# A Reactive GRASP Algorithm for the Multi-depot Vehicle Routing Problem*

Israel Pereira de Souza[1][0000−0002−3745−5530], Maria Claudia Silva Boeres[1][0000−0001−9801−2410], Renato Elias Nunes de Moraes[1][0000−0001−5538−0504], and João Vinicius Corrêa Thompson[2][0000−0002−5781−3278]

[1] Federal University of Espírito Santo, Vitória ES 29075-910, Brazil
[2] Federal Center for Technological Education of Rio de Janeiro, Rio de Janeiro RJ 20271-204, Brazil

**Abstract.** The vehicle routing problem (VRP) is a well know hard to solve problem in literature. In this paper, 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). This algorithm, called RGRASP+VND, combines four distinct local search procedures and a clustering technique. The Cordeau 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 for all instances on the experimental tests when compared to the earlier GRASP approaches in the MDVRP literature.

**Keywords:** Reactive greedy randomized adaptive search procedures · Multi-depot vehicle routing problem · City logistics.

## 1 Introduction

The 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 [19] problem with several real-world applications. It was first introduced by Dantzig and Ramser [10] 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 [6] 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), VRP with pickup and delivery

* Supported by CAPES.

(VRPPD), VRP with backhauls (VRPB), split delivery VRP (SDVRP), periodic VRP (PVRP), among others. We focus on MDVRP.

MDVRP 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 [4]; 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 [7]. Examples of heuristics that follow the cluster-first route-second approach for the MDVRP are: the multi-phase modified shuffled frog leaping algorithm [20] and a tabu search heuristic with variable cluster grouping [15]. The route-first cluster-second approach is adopted by heuristics as an evolutionary algorithm for the VRP [25], a memetic algorithm for the VRPTW [18], among others. This work concerns the proposal of a heuristic for the MDVRP adopting the cluster-first route-second approach.

Recent literature on MDVRP heuristics involves: a novel two-phase method approach [1]; a biased-randomized variable neighborhood search [26]; a 2-opt guided discrete antlion optimization [3]; a harmony search [22] and an enhanced intelligent water drops algorithm [11]. A GRASP algorithm was used to solve the single truck and trailer routing problem with satellite depots (STTRPSD) [28], of which MDVRP is a special case.

In this work, we describe a Reactive GRASP algorithm for the MDVRP (RGRASP+VND) combining four distinct local search procedures, two of them presented here, and a clustering technique. 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 dataset for the MDVRP [8]. State-of-the-art algorithms are identified and compared to RGRASP+VND and the results are discussed in Section 5.2.

This paper is structured as follows: Section 2 describes the problem with its mathematical formulation. Section 3 presents MDVRP solving approaches, which are the route-first cluster-second and cluster-first route-second, and discusses two clustering techniques. Section 4 describes the RGRASP+VND algorithm for the MDVRP. The analysis of the results obtained from the computational experiments is presented in Section 5, which is organized with the algorithm parameters decisions (Section 5.1), followed by the comparison of the state-of-the-art algorithms (Section 5.2). Section 6 concludes this work.

## 2 Problem Description

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 $(v_i, v_j) \in E$ has a Euclidean distance $d_{ij}$· A fleet of up to $K$ vehicles of capacity $Q_k$ is located in each of the $M$ depots. The MDVRP goal is to minimize a set of vehicle routes such that: i) each vehicle starts and ends at the same depot; ii) each customer is served once by a single vehicle; iii) the total load of a vehicle $k$, for $k = 1, \ldots, R$, does not exceed its maximum capacity $Q_k$ and iv) the number of vehicles used by a depot does not exceed the maximum number of vehicles $K$.

Parameters:

$N$ 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 of vehicle $k = 1, \ldots, R$.
$q_i$ demand of a customer $v_i$, $i = 1, \cdots, N$.
$d_{ij}$ 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 problem is formulated in [17] as:

Minimize:

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

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

$$\sum_{j=1}^{N+M} \sum_{k=1}^{R} x_{ijk} = 1, \quad i = 1, 2, \ldots, N \tag{3}$$

$$\sum_{i=1}^{N+M} x_{ihk} - \sum_{j=1}^{N+M} x_{hjk} = 0, \quad k = 1, 2, \ldots, R, \;\; h = 1, 2, \ldots, N+M \tag{4}$$

$$\sum_{i=1}^{N+M} q_i \sum_{j=1}^{N+M} x_{ijk} \leq Q_k, \quad k = 1, 2, \ldots, R \tag{5}$$

$$\sum_{i=1}^{N+M} \sum_{j=1}^{N+M} d_{ij} x_{ijk} \leq T_k, \quad k = 1, 2, \ldots, R \tag{6}$$

$$\sum_{i=N+1}^{N+M} \sum_{j=1}^{N} x_{ijk} \leq 1, \quad k = 1, 2, \ldots, R \tag{7}$$

$$\sum_{j=N+1}^{N+M} \sum_{i=1}^{N} x_{ijk} \leq 1, \quad k = 1, 2, \ldots, R \tag{8}$$

$$y_i - y_j + (M+N)x_{ijk} \leq N + M - 1 \text{ for } 1 \leq i \neq j \leq N \text{ and } 1 \leq k \leq R \tag{9}$$

The objective function (1) minimizes the total distance travelled by the vehicles. Constraints (2) and (3) guarantee that each customer is served one vehicle. The route continuity is represented by constraint (4). Constraint (5) ensures that the vehicles capacity are respected and the constraint (6) ensures that the vehicle maximum distance is respected. Constraints (7) and (8) verify vehicle availability and subtour elimination is checked by constraints (9).

## 3 The MDVRP Solution Approach

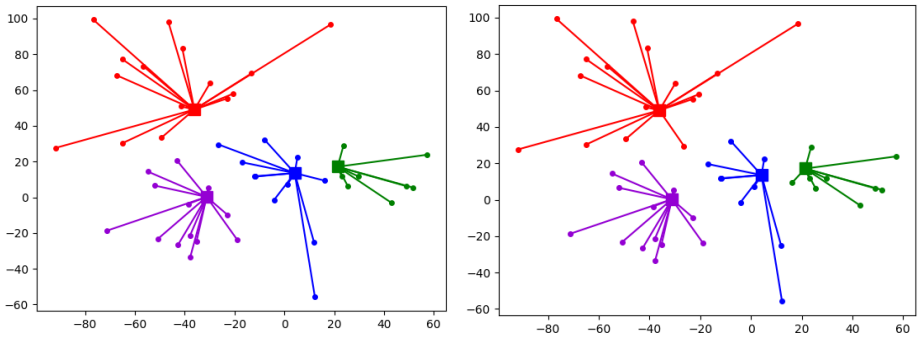
Heuristics for solving the MDVRP can be classified into cluster-first route-second or route-first cluster-second approaches. In the first approach, the vertices are clustered considering a depot as the center of each cluster. The routes are then determined by properly sequencing the customers in each cluster [7]. In the second approach, a single giant route is built considering all vertices, and then it is split into a set of feasible routes [4].

In this paper, we consider the cluster-first route-second approach. The vertices of the input test problem being solved are clustered into $M$ depots generating $M$ capacitated VRP (CVRP) subproblems.

Several clustering methods can be found in the literature. A classical clustering method is K-means [21]. 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. A group of different clustering methods has been described by Giosa et al. [14] 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. [14] also present a parallel and simplified approach for the Urgencies clustering method. The parallel approach considers all depots on the customer urgency evaluation and the simplified approach only considers two depots for the evaluation. The customer with the highest urgency is assigned to its closest depot. We have chosen to compare the Urgencies clustering method with the parallel approach and the K-means clustering method. The Equation (10) calculates the urgency $u_c$ of a customer $v_c$, $c \in 1, 2, \ldots, N$, considering the Urgencies clustering method with parallel approach. The function `distance` calculates the Euclidean distance between two vertices and the depot $D'$ is the closest depot of the customer $v_c$. The customer with the highest value of $u$ is assigned to its closest depot.

$$u_c = \left( \sum_{i=1}^{M} \texttt{distance}(v_c, D_i) \right) - \texttt{distance}(v_c, D') \tag{10}$$

Clustering examples for the cluster-first route-second approach using the instance `pr01`, available in the Cordeau dataset [8], generated with K-means and Urgencies clustering methods are plotted in Fig. 1. The clusters are highlighted in different colors. We can see that, for this instance, 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 CVRP subproblem and the union of all CVRP subproblems composes an MDVRP problem with the cluster-first route-second approach.



(a) (b)

**Fig. 1.** (a) Plotted K-means clustering method on pr01 (b) Plotted Urgencies clustering method on pr01 (points are customers and square are depots).

In this paper, we compare the K-means clustering method and the Urgencies clustering method with the parallel approach [14] applied to the cluster-first route-second approach, which is discussed in Section 5.1. Both clustering methods were implemented considering the depot's maximum capacity, which is the sum of the vehicle capacities in the depot. If a customer that would be assigned to a cluster has a demand that surpasses the depot's maximum amount of demand, it is assigned instead to the next closest cluster which has a free load to support it.

## 4 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.

The Greedy Randomized Adaptive Search Procedures (GRASP) [12] is a multi-start metaheuristic with construction and local search iterations. 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 $\alpha$ percentage parameter. For instance, when $\alpha = 0.1$, the list will contain the 10% best candidates.

Thus, the next element selected to integrate the solution being constructed is randomly selected from this list. 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, in order to obtain a feasible solution. The GRASP algorithm requires only two input parameters: the RCL size constraint $\alpha$ and the number of GRASP iterations ($max\_iterations$) and can be easily implemented for a parallel scheme [23].

The Reactive GRASP [24] considers not a fixed value for the RCL $\alpha$ parameter, but a random $\alpha$ with a probability selected for each GRASP iteration. The probabilities are computed and self adjusted along the iterations. More formally, an initial value for $\alpha$ is selected from the discrete set $A = \{\alpha_1, \alpha_2, \cdots, \alpha_m\}$, with $m$ predetermined $\alpha$ values. Let $p_i$ be the probability associated with the choice of $\alpha_i$, for $i = 1, 2, \cdots, m$. The initial values $p_i = 1/m$, for $i = 1, 2, \cdots, m$ correspond to a uniform distribution. The original authors, Prais and Ribeiro, 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 [24]. Let *block_ iterations* be the number of iterations without probabilities update, $F(S^*)$ be the cost of the best solution found so far and $A_i$ the average of the solutions obtained in the iterations using $\alpha = \alpha_i$. After each *block_ iterations* of GRASP, 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 $q_i$ normalized values given by $q_i = \frac{F(S^*)}{A_i}^{\delta}$.

Beyond the additional parameter *block_ iterations*, the Reactive GRASP has also the parameter $\delta$, used to attenuate the $p_i$ updated probabilities, which is commonly adopted as $\delta = 10$ in a diversity of problems in the literature [5,16,24].

### 4.1 A Reactive GRASP for the MDVRP

We have utilized the Reactive GRASP using the absolute qualification rule [24] to update the probabilities. However, this rule fails when not all predetermined values for $\alpha_i$ are selected along with the algorithm execution. In this case, no solutions average ($A_i$) and their respective $q_i$ would be generated. To solve this problem we generate one solution for each $\alpha_i$, for $i = 1, 2, \ldots, m$, at the beginning of the algorithm. Considering that the probability for each $\alpha_i$ to be selected starts

uniformly, generating one solution for each value will not prejudice the rule. We also constructed a greedy randomized algorithm by adding a Restrict Candidate List (RCL) to the widely known Clarke-Wright savings algorithm [6].

The RGRASP+VND pseudocode is given by Algorithm 1. The algorithm receives as input the maximum number of iterations (*max_ iterations*), the number of $\alpha$ values $m$, the number of iterations which the set probabilities $p_i \in P, i = 1, 2, \ldots, m$, will be updated (*block_ iterations*), the $\delta$ parameter, which is used to attenuate the $p_i$ updated probabilities, the graph $G = (V, E)$, the number of vertices $N$, the number of depots $M$, the maximum number of vehicle of each depot $K$ and the set of vehicle capacities $Q$. It returns the best solution found for an MDVRP instance problem.

**Algorithm 1:** Reactive GRASP pseudocode

**Input:** *max_ iterations*, $m$, *block_ iterations*, $\delta$, $G = (V, E)$, $N$, $M$, $K$, $Q$
**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     α ← select a random α_i from A considering the probabilities set P
11     Solution ← GreedyRandomizedSolution(Clusters, K, Q, α)
12     Solution ← LocalSearch(Solution)
13     BestSolution ← UpdateSolution(Solution, BestSolution)
14     BlockSolutions ← BlockSolutions ∪ Solution
```

The RGRASP+VND algorithm starts clustering the instance problem by the function `ClusterizeInstance` on line 1. The set of $\alpha$ values $A$ is constructed by function `ConstructAlphasSet`, that generates $m$ $\alpha$ values uniformly distributed in [0,0.9], on line 2. Lines 3-4 fix the Reactive GRASP absolute qualification rule problem, by creating one solution for each $\alpha_i \in A, i = 1, \ldots, m$, on line 3 and updating the set of probabilities $P$ on line 4. This history of solutions (*Block-Solutions*) is emptied on line 5, which will later store all solutions of a block of iterations, used to update the each probability $p_i \in P$ for $\alpha_i \in A$, $i = 1, \ldots, m$. Line 6 controls the amount of solutions which will be generated by the algorithm, defined by *max_ iterations* given as input. 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 an $\alpha_i \in A$ are updated, for $i = 1, \ldots, m$ by line 8 and the history of solutions (*BlockSolu-*

*tions*) is emptied on line 9. Line 10 selects a random $\alpha_i \in A$, considering the probability of selection of each $\alpha_i$, defined by $P$. Line 11 constructs a *Solution* by the `GreedyRandomizedSolution` described by the Algorithm 2. The local search procedures described on Section 4.2 are applied by line 12 on the solution. Line 13 updates the best solution found. Line 14 stores the solution created on the current block of iterations history.

The `GreedyRandomizedSolution` pseudocode is described on Algorithm 2. 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 $\alpha$ and returns a feasible *Solution*.

**Algorithm 2:** `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 the RCL
 7 |   Solution ← Solution ∪ {s}
 8 |   Update the set of candidate elements
 9 |   Reevaluate the incremental costs
10 if Solution is not feasible then
11 |   Solution ← Repair(Solution)
```

It starts with an empty solution on line 1. Then the set of candidate elements are initialized on line 2 and their incremental costs are evaluated on line 3. While there is a candidate element, the RCL is constructed on line 5 with its size limited by the parameter $\alpha$. One random element of RCL is chosen on line 6 and incorporated to the solution on line 7. Then the set of candidate elements are updated on line 8 and their incremental cost are reevaluated on line 9. If the generated solution is infeasible, a new solution is generated by line 11.

### 4.2 Local Search

The local search used by this paper considers four methods: `2-swap`, `2-opt`, `Drop one point intra-depot`, `Drop one point next depot`. The `2-swap` and `2-opt` are classical heuristics widely used on optimization problems, the next two algorithms are new local search methods presented in this paper.

The `2-swap` and `2-opt` [9,13] heuristics were considered in the local search. Their movements can be used by a neighborhood search and perform, respectively, two vertices (customers) swaps and two edges swaps while preserving the tour. The neighborhood of a solution $s$ can be 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 heuristics to perform a neighborhood search are the first improvement and 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 of the neighborhood [23]. For this paper we chose the best improvement heuristic for the `2-opt` and `2-swap` movements.

The `2-opt`, `2-swap` and `Drop one point intra-depots` methods are intra-depots, which are performed considering only one cluster. The method `Drop one point next depot` is a inter-depot method, which consider all clusters of the solution and can adjust it's clusters.

To reference the depot that attends the customer $c_i$, we will use the notation $D_{m,c_i}$, which represents the depot $D_m$ that attends the customer $c_i$.

**Drop One Point Intra-depot** For each cluster associated with depot $D_i$ of the solution $S$, every city $c_i$ which is clustered to depot $D_i$ is removed and reinserted in the best viable position, that is the one that not surpass the vehicle maximum capacity, in a different route of the depot $D_i$ which the highest enhancement is obtained. The Algorithm 3 shows the `Drop one point intra-depot` pseudocode.

**Algorithm 3:** 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;
```

The `Drop one point intra-depot`, on line 1, iterates for each depot $D_i$ of $S$. The line 2 iterates for each route $h_l$ of $D_i$. The line 3 iterates over all cities $c_i$ on route $h_l$. The line 5 removes $c_i$ from its current route and reinsert it in a different route that gives the highest enhancement to $S$ without making an infeasible move. This movement is only performed if it improves the solution cost which is checked on 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 4 shows the `Drop one point intra-depot` pseudocode.

**Algorithm 4:** Drop one point next depot

**Input:** $S, N, M, K, Q$
**Output:** $S$
1 $max\_demands \leftarrow \sum_{k=1}^{K} Q_k$
2 **for** $m \leftarrow$ *1 to* $M$ **do**
3 | $cluster\_demands[m] \leftarrow$ sum of demands of all customers clustered to $D_m$
4 **for** $i \leftarrow$ *1 to* $N$ **do**
5 | $D_{current,c_i} \leftarrow$ get the depot $c_i$ is currently associated.
6 | $m \leftarrow$ index associated to depot $D_{current,c_i}$.
7 | **if** $cluster\_demands[m] + q_i <= max\_demands$ **then**
8 | | $D_{next,c_i} \leftarrow$ get $D_{next,c_i}$ the closest depot of $c_i$, where $D_{next,c_i} \neq D_{current,c_i}$.
9 | | **do**
10 | | | $S \leftarrow$ 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.
11 | | | $cluster\_demands[m] = cluster\_demands[m] - q_i$
12 | | | $m \leftarrow$ index associated to depot $D_{next,c_i}$.
13 | | | $cluster\_demands[m] = cluster\_demands[m] + q_i$
14 | | **if** *the cost of* $S$ *reduces*;

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 4 iterates over all customers $c_i$ of the instance. The line 5 get the depot ($D_{current,ci}$) which the customer $c_i$ is associated. The line 6 get the index $m$ of the depot $D_{current,ci}$. The line 7 checks if the depot with index $m$ can support the customer $c_i$. The line 8 get the closest depot of $c_i$, different than the depot $c_i$ is already allocated, called $D_{next,ci}$. The line 10 removes $c_i$ and reinsert 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 14. The lines 11-13 are used to update the amount of demand the clusters are holding.

## 5 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 [8], available at the Vehicle Routing Problem Repository VRP-REP [3].

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

Section 5.1 presents the strategies and parameters adopted for the algorithm and Section 5.2 discusses the literature review and the RGRASP+VND comparison.

### 5.1 Conceiving the RGRASP+VND Algorithm Structure

In order to assess the RGRASP+VND algorithm, results from preliminary tests are registered, considering the combination of two different clustering strategies together with four local search procedures performed individually and combined by the VND algorithm. The goal is to identify which combination of strategies achieved better results.

The procedures implemented for the RGRASP+VND algorithm are: the clustering methods K-means and Urgencies; the Clarke-Wright savings constructive algorithm; the local search strategies 2-swap, 2-opt intra routes, drop one point next depot and drop one point intra-depot, identified in this section respectively as $T_1$, $T_2$, $T_3$ and $T_4$, and performed each with best improvement; and VND. The four local search strategies are performed in two ways: sequentially, denoted as LS($T_1$, $T_2$, $T_3$, $T_4$) or combined by the VND, denoted as VND($T_1$, $T_2$, $T_3$, $T_4$), both considering the sequence of execution given by the identification order $T_i$, $i = 1, \ldots, 4$.

**Table 1.** Average solution values on eight instances of the Cordeau dataset [8] for MDVRP on the implemented procedures sequences.

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

Four different sequences of all procedures were implemented: $PS_1$, $PS_2$, $PS_3$ and $PS_4$ (see Table 1), each performed over ten instances (p01 to p10) from the Cordeau dataset [8]. Using the K-means clustering method, the Clarke-Wright savings method could not generate a feasible solution for the instances p04 and p07. However, using the Urgencies clustering method the constructive method could generate a feasible solution for all ten instances. To compare the average results, only the eight instances in which both methods could generate feasible solutions are considered. Thus, Table 1 presents the average solutions for eight instances of the Cordeau dataset considering the four procedures sequences.

The algorithm has reached the 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 dataset are: i) Cluster method: Urgencies; ii) *max_ iterations* = 10,000; iii) $\delta = 10$; iv) $m = 100$; v) *block_ iterations* = 100.

### 5.2 Computational Results

In this section, the experiments with the RGRASP+VND are discussed. A literature review was conducted to identify the GRASP state in the MDVRP literature. In this study, the search string: *"grasp" AND ("mdvrp" OR "multi-depot vehicle routing problem")* was used in March of 2022 at the Scopus, IEEE Xplore, and Web of Science datasets without any time restriction. Only two distinct papers were obtained and only one (GRASP/VND) [28] has presented results considering instances from the literature.

**Table 2.** Algorithms solution comparison for the MDVRP instances.

| | | GRASP/VND [28] | | | 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 [2].

The RGRASP+VND and the GRASP/VND both consider the VND local search structure and share the GRASP metaheuristic. However, there are several differences between them. The GRASP/VND uses the route-first cluster-second approach (see Section 3), it also adopts the classical GRASP version, where the $\alpha$ parameter calibration is necessary (see Section 4) and a different set of local search heuristics for the VND structure were applied. The RGRASP+ VND considers the cluster-first route-second approach and the Reactive GRASP, which is a GRASP adaptation to enhance the $\alpha$ parameter automatically. Moreover, the set of local search heuristics employed for the VND structure is different.

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. Table 2 compares the RGRASP+VND and the GRASP/VND algorithms. The Instance column denotes the instance, the BKS column presents the best known results for the instance, and values followed by * present the optimum solution, proved by R. Baldacci and A. Mingozzi [2]. The

following columns consider the algorithm results: the column *Avg.* presents the average solution of 10 runs; the column *Best* presents the best solution in 10 runs and the column *%Dev* presents the algorithm percentage deviation, which is evaluated for each instance as $\frac{Best-BKS}{BKR} \times 100$.

The RGRASP+VND obtained a better solution than GRASP/VND for the instances p01, p02, p04, and p06. On the other hand, GRASP/VND outperforms the RGRASP+VND algorithm for the 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 which the best solution was only 6.82 units far from the BKS (%Dev=1.44) against the GRASP/VND, which got the best result with 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 on the biggest instance p21, in which the best solution value was 660.84 units far from the BKS (%Dev=12.07) against the GRASP/VND with 428.79 units far from the BKS (%Dev=7.83), with a difference of 232.05 units between the algorithms.

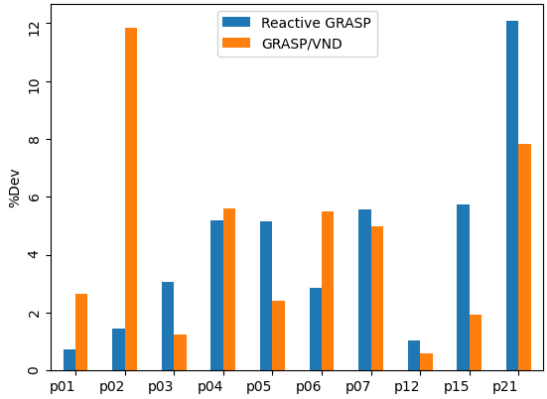



**Fig. 2.** Comparison of deviation percentage for the RGRASP+VND and the GRASP/VND [28].

Figure 2 presents, for each instance, the RGRASP+VND and GRASP/VND percentage deviations. The highest difference in the methods solutions was for the instance p02, where the RGRASP+VND outperforms the GRASP/VND, with a 10.41% deviation difference. For instances p01 and p04, RGRASP+VND was better than GRASP/VND, but with a small deviation percentage difference (1.93% and 0.42% respectively). For instance p06, the RGRASP+VND was better than GRASP/VND, with a 2.63% deviation difference. GRASP/VND was better than RGRASP+VND for the instances p03, p05, p07, p12, p15, p21, with 1.82%, 2.73%, 0.60%, 0.44%, 3.81%, 4.24% deviation differences respectively.

A literature review was 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")* was used on Scopus database limited to years 2018 to March of 2022. The algorithms found and also those used for comparison were ranked considering their best average results on the Cordeau set of instances of 1997, which is the most used MDVRP benchmark. The two methods with the best average results were: A Hybrid Genetic Algorithm [27] and A biased-randomized variable neighborhood search [26]. Both methods outperform the reported GRASP results for the Cordeau instances package.

Table 3 compares the RGRASP+VND and the state-of-the-art algorithms: HGSADC [27], BR-VNS [26]. The HGSADC achieves the best results, where it obtained the BKS on all instances of the experimental tests, excluding the p03 and p04, and their average deviation was lower than 0.20% on all instances. The BR-VNS had the second best results and their average deviation was lower than 2%. The RGRASP+VND was the worse algorithm in this comparison and its average deviation has reached 12.07% on the biggest instance (p21).

**Table 3.** RGRASP+VND comparison with state-of-art methods for the MDVRP instances.

| | | HGSADC [27] | | BR-VNS [26] | | 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 | 12.07 |

* proved optimality [2].

GRASP metaheuristic is not known as performing better than state-of-the-art MDVRP algorithms. However, the RGRASP+VND algorithm brings contributions to the GRASP literature for the MDVRP which currently is minimal. It shows improvement over the GRASP/VND for most small instances of the experimental tests, where the Cordeau dataset is considered. We also highlight that the RGRASP+VND presents more stable results, where it achieves better average results against the GRASP/VND for every instance on the experimental tests (see Table 2).

## 6 Conclusions

This work introduced the Reactive GRASP metaheuristic with VND local search strategy for the MDVRP, combining four distinct local search procedures, two of them presented in this paper, and a clustering technique. Through the literature review, we observed that the GRASP metaheuristic is not known as performing better than the highest performing algorithms in the literature. 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 dataset [8] and also achieves better average solutions for all instances on the experimental tests against the GRASP/VND [28] (see Table 2).

In future work, the application of other constructive algorithms can be investigated for the RGRASP+VND algorithm. Furthermore, the RGRASP+VND parametrization can be improved as well with more local search procedures. The application of RGRASP+VND could also be investigated on other VRP versions, for instance, those with time window constraints.

## References

1. Baghbadorani, R.R., Ghanavati, A., Zajkani, M., Haeri, M.: A novel two-phase approach to solve multi-depot vehicle routing problem. In: 2021 25th International Conference on System Theory, Control and Computing (ICSTCC). pp. 390–394. IEEE (2021)
2. Baldacci, R., Mingozzi, A.: A unified exact method for solving different classes of vehicle routing problems. Mathematical Programming **120**(2), 347–380 (2009)
3. 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 **2**(2), 112–125 (2019)
4. Beasley, J.E.: Route first—cluster second methods for vehicle routing. Omega **11**(4), 403–408 (1983)
5. Boudia, M., Louly, M.A.O., Prins, C.: A reactive grasp and path relinking for a combined production–distribution problem. Computers & Operations Research **34**(11), 3402–3419 (2007)
6. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations research **12**(4), 568–581 (1964)
7. Comert, S.E., Yazgan, H.R., Kır, S., Yener, F.: A cluster first-route second approach for a capacitated vehicle routing problem: a case study. International Journal of Procurement Management **11**(4), 399–419 (2018)
8. Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks: An International Journal **30**(2), 105–119 (1997)
9. Croes, G.A.: A method for solving traveling-salesman problems. Operations research **6**(6), 791–812 (1958)
10. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management science **6**(1), 80–91 (1959)
11. Ezugwu, A.E., Akutsah, F., Olusanya, M.O., Adewumi, A.O.: Enhanced intelligent water drops algorithm for multi-depot vehicle routing problem. PloS one **13**(3), e0193751 (2018)

12. Feo, T.A., Resende, M.G.: Greedy randomized adaptive search procedures. Journal of global optimization **6**(2), 109–133 (1995)
13. Flood, M.M.: The traveling-salesman problem. Operations research **4**(1), 61–75 (1956)
14. Giosa, I., Tansini, I., Viera, I.: New assignment algorithms for the multi-depot vehicle routing problem. Journal of the operational research society **53**(9), 977–984 (2002)
15. He, Y., Miao, W., Xie, R., Shi, Y.: A tabu search algorithm with variable cluster grouping for multi-depot vehicle routing problem. In: Proceedings of the 2014 IEEE 18th international conference on computer supported cooperative work in design (CSCWD). pp. 12–17. IEEE (2014)
16. Iori, M., Locatelli, M., Moreira, M.C., Silveira, T.: Reactive grasp-based algorithm for pallet building problem with visibility and contiguity constraints. In: International Conference on Computational Logistics. pp. 651–665. Springer (2020)
17. Kulkarni, R., Bhave, P.R.: Integer programming formulations of vehicle routing problems. European journal of operational research **20**(1), 58–67 (1985)
18. Labadi, N., Prins, C., Reghioui, M.: A memetic algorithm for the vehicle routing problem with time windows. RAIRO-Operations research **42**(3), 415–431 (2008)
19. Lenstra, J.K., Rinnooy Kan, A.H.G.: Complexity of vehicle routing and scheduling problems. Networks **11**(2), 221–227 (1981)
20. Luo, J., Chen, M.R.: Multi-phase modified shuffled frog leaping algorithm with extremal optimization for the mdvrp and the mdvrptw. Computers & Industrial Engineering **72**, 84–97 (2014)
21. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297. Oakland, CA, USA (1967)
22. Misni, F., Lee, L.: Harmony search for multi-depot vehicle routing problem. Malaysian Journal of Mathematical Sciences **13**(3), 311–328 (2019)
23. Potvin, J.Y., Gendreau, M.: Handbook of Metaheuristics. Springer (2018)
24. Prais, M., Ribeiro, C.C.: Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. INFORMS Journal on Computing **12**(3), 164–176 (2000)
25. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & operations research **31**(12), 1985–2002 (2004)
26. Reyes-Rubiano, L., Calvet, L., Juan, A.A., Faulin, J., Bové, L.: A biased-randomized variable neighborhood search for sustainable multi-depot vehicle routing problems. Journal of Heuristics **26**(3), 401–422 (2020)
27. Vidal, T., Crainic, T.G., Gendreau, M., Lahrichi, N., Rei, W.: A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research **60**(3), 611–624 (2012)
28. Villegas, J.G., Prins, C., Prodhon, C., Medaglia, A.L., Velasco, N.: Grasp/vnd and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. Engineering Applications of Artificial Intelligence **23**(5), 780–794 (2010)