Department of Management Science & Technology

MSc in Business Analytics

«Optimisation Methodologies for Clustered Vehicle Routing Problems»

By

Michail Kalligas

Student ID Number: f2822103

Name of Supervisor: Emmanouil Zachariadis

January 2023

Athens, Greece

Table of Contents

Abstract 3
1. Introduction 3
2. Literature Review 4
  2.1. Real world applications 4
  2.2. Optimization Methodologies 5
3. Implementation 7
  3.1. Problem formulation 7
  3.2. Instances 8
  3.3. Clustering 8
  3.4. Methodologies 9
    3.4.1. Solving the high-level routing problem 9
    3.4.2. Solving the low-level routing problem 10
    3.4.3. Optimization of the solution 11
4. Results 14
  4.1. Analysis 14
  4.2. Hard vs soft solutions visualized 19
5. Conclusions 21
  5.1. The problem 21
  5.2. The implementation 21
  5.3. The results 21
  5.4. Method shortcomings 22
6. References 22
7. Appendix 24
  7.1. Code files (.py) and instances (.xml) 24
  7.2. Results analysis (Jupyter) 24
  7.3. Results table 24
  7.4. Plots and routes of presented solutions 24

Abstract

This report addresses the Clustered Vehicle Routing Problem (CluVRP), an extension of the renowned Vehicle Routing Problem (VRP). After a brief literature review of the optimization methodologies applied to the problem and some real-world applications, an implementation of some well-known methodologies combined to solve the problem is presented. In particular, the problem is divided into two problems, the high-level routing problem of visiting customer clusters and the low-level problem of intra-cluster routing. Two greedy approaches are employed to get a fast initial solution, while three metaheuristic methods are utilized to optimize it. Finally, the implementation is tested on various VRP benchmark instances adapted for the CluVRP, and the results are analyzed to determine which method is more time- and cost-effective.

1. Introduction

The Vehicle Routing Problem (VRP) is a classic combinatorial optimization problem in Operations Research and Logistics. It was first introduced by Dantzig and Ramser in 1959 and has since become one of the community's most widely studied and well-known problems. As a logistic distribution problem, it aims to find the most efficient routes for a fleet of vehicles to serve a set of geographically dispersed customers while also satisfying various constraints. Hence, finding optimal solutions requires a large amount of computational time. Even in its simplest form, it is considered NP-complete, meaning that no polynomial-bounded algorithm that solves the problem is likely to exist. Despite this, the problem has been the subject of many research papers and books due to its numerous real-world applications in transportation, network optimization, and distribution. As a result, several approximation algorithms and heuristics have been developed to provide near-optimal solutions in a reasonable amount of time. A comprehensive overview of the main achievements in the field was provided in 2008 (Golden, Raghavan, & Wasil, 2008), while a taxonomic review of the various existing metaheuristic algorithms was made in 2020 (Elshaer & Awad, 2020). The VRP is a problem of ongoing interest for researchers and practitioners, with new developments and advancements being made in the field. This thesis aims to investigate and develop well-known methods for addressing one expansion of the VRP, the Clustered Vehicle Routing Problem (CluVRP) in the Python programming language, then compare and analyze their performance. The CluVRP is an extension of the Capacitated Vehicle Routing Problem (CVRP), the most common variant of VRP, which aims to minimize the total cost of routes for the distribution of goods to geographically dispersed customers.

CluVRP extends the CVRP by incorporating a restriction on customer clustering. There are two versions of the problem: the hard-clustered and the soft-clustered. The hard-clustered variation organizes customers into clusters and imposes the limitation that if a vehicle visits one customer in a cluster, it must visit all other customers in that cluster before visiting any other customers or returning to the depot. On the other hand, the soft-clustered

version is less strict than the hard-clustered version because visits to customers in the same cluster can interrupted by visits to customers in different clusters if it's profitable. In this study, the hard-clustered problem will be the primary emphasis; however, a pseudo-soft clustering approach will also be discussed.

CluVRP was introduced in 2008 by Sevaux and Sorensen and, like the CVRP, is an NP-hard issue. Numerous methods for solving the problem have been presented in the literature. These include heuristics, metaheuristics, and exact methods among them. Heuristics are simple, quick methods that approximate the ideal solution to a problem but do not guarantee it. Metaheuristics are approaches that increase the quality of solutions and typically use a combination of heuristics and randomization to explore the solution space. They often find decent solutions quickly, even when the problem size is large. Finally, exact methods are used to optimize an objective function subject to constraints and find optimal solutions to problems that can be formulated as mathematical models but are computationally costly and limited to small-scale applications. In this implementation, a combination of heuristics and metaheuristics will be employed.

2. Literature Review

2.1. Real world applications

The CluVRP becomes useful when clients to be visited are clustered. The main reasons for customer clustering are the number of customers in delivery operations and the amount of available resources. Service providers are often only able to meet some of the requests within a given territory, especially in situations when there are limited resources or a high volume of service needs (Exposito et al., 2016). According to Charon and Hudry (1993), dividing clients into distinct subareas enables a business to organize its services so that some subareas are served on specified days of the week or by specific drivers. To this end, Bowerman, Calamiand, and Hall (1994) developed a cluster-first/route-second category of heuristics for the VRP. It organizes clients into clusters, allocates a vehicle to each cluster, and sets the sequence in which customers are visited along each route. Thus, vehicle routing can be performed on a cluster basis instead of a customer basis, which decreases the complexity of the VRP by a factor of ten.

As Exposito et al. (2016) remark, the definition of clusters may be implicitly set by the characteristics of the practical application or by the service provider to satisfy specific criteria. In general, clusters could exist for various reasons:

1) Customers' locations. Some examples are large territories with widely separated towns, areas with mountainous geography where the population lives in small villages located in bottom valleys, and island groups. In

these instances, a vehicle must serve all customers in one area before moving on to the next to reduce travel expenses.

2) Customers can cluster into groups who desire the same service or goods. This is illustrated by the VRP with Compartments (Derigs et al., 2011), in which a fleet of trucks with multiple compartments transports non-inter-mixable items. In this case, clusters are defined as groups of customers requiring the same product.
3) There are different priority levels among customers. Examples include:
    a) Injured people whose injuries are classified based on their severity and individuals with the most severe injuries must receive care first, followed by the rest of the severity levels.
    b) Electrical outages, where all customers with a high priority are gathered according to their locations (clusters) and must be visited before those with a lower priority (many clusters per level of priority) by a business vehicle. Weintraub, Aboud, Fernández, Laporte, and Ramrez (1999) investigated a similar optimization problem using point precedence in the context of emergency operations in electric distribution systems.
    c) Order-picking (de Koster, Le-Duc, & Roodbergen, 2007). It refers to a group of products distributed in a warehouse that must be retrieved simultaneously due to their shared destination.

A service provider can also divide its customers into clusters based on specific criteria, especially when there are a lot of them. This is referred to as the "districting problem" (Haugland, Ho, & Laporte, 2007). It includes optimization criteria such as proximity, balance (differences in the number of customers, service demands, areas, service revenues, and service times in the clusters.), cost-effectiveness, and time windows for the customers.

The methodologies presented in this work follow the cluster-first/route-second category of heuristics and tackle the case where customers are geographically clustered based on the distance between them.

2.2. Optimization Methodologies

As previously mentioned, CluVRP was first defined as a heuristic for decreasing the dimensions of large VRPs by Sevaux and Sorensen (2008). They grouped consumers into clusters and executed routing over a significantly reduced problem consisting of the cluster centers. In a subsequent step, intra-cluster routing is performed. For the last step, a Mixed-Integer Programming formulation is proposed. Following up, Battarra et al. created exact algorithms for CluVRP and presented findings for a collection of benchmark instances in 2014. Their best performing technique depended on a preprocessing scheme that calculated the shortest Hamiltonian path between each pair of vertices in each cluster. Their observation was that if the first and last customers visited in a cluster are known, the shortest Hamiltonian path corresponds to the optimal sequence of visits for the remaining customers in the cluster. To solve the shortest Hamiltonian path problem, they used the TSP code Concorde (Applegate, Bixby, Chvàtal, & Cook, 2001). In 2015 Vidal et al. proposed three metaheuristics for the CluVRP algorithm.

Two were based on iterated local search, while the other was a hybrid genetic algorithm that produced higher-quality results by utilizing efficient large neighborhood search procedures. Nonetheless, its preprocessing phase necessitates the computation of all feasible intra-cluster Hamiltonian routes, which increases the total processing time in instances with large clusters. They suggested that future work should focus on enhancing CPU time through heuristic preprocessing techniques.

Next, Expósito-Izquierdo et al. (2016) suggested a two-phase VNS (Variable Neighborhood Search) method for addressing CluVRP by separating the problem into two hierarchically structured routing problems. The problem at the highest level regards the routing of clusters, while the problem at the lowest level concerns the order of customer visits within each cluster. The former is solved using a VNS algorithm, whereas the latter employs exact or heuristic methods. The exact approaches at the lowest level are suitable for instances with few clients per cluster, but in other instances, they are replaced by a different VNS heuristic. Overall, the technique identified the best-known solutions for numerous small instances while also performing admirably on instances with up to 7500 clients. Defryn and Sorensen (2017) introduced a two-level VNS heuristic for efficiently solving CluVRP. Combining the local search at the customer level with the local search at the cluster level made it possible to acquire higher-quality solutions in a time-effective manner. The algorithm swiftly obtained the best known solutions in 71 out of 79 examples tested with small and medium-sized problems from the literature. The average gap was 0.04% across 20 runs. In large examples modified from the Golden benchmark, as proposed by Battarra et al. (2014) and Expósito-Izquierdo et al. (2016), a gap of around 1% was achieved between the best-known solutions and the average solution. Hintsch and Irnich (2018) developed an ILS-based preprocessing phase first computes all intra-cluster routes for every possible entry-exit combination. They also implemented cluster neighborhoods that exchange clusters between routes, similar to edge-exchange methods for CVRP. This complementarity allows the detection of CluVRP solutions with better quality in relatively shorter time. Out of 230 instances, LMNS improved the solutions in seven cases compared to the exact algorithm of Battarra et al. The LMNS is also competitive with the UHGS (Vidal, Battarra, Subramanian, & Erdogan, 2015) metaheuristic technique.

Later, Pop et al. (2018) devised a unique two-level optimization algorithm for CluVRP by decomposing the problem into two subproblems: an upper-level (global) subproblem and a lower-level (local) subproblem. These subproblems were solved independently by employing an efficient genetic algorithm whose purpose was to deliver:

1) collections of global routes visiting the clusters, a constructive method to generate the initial population of the CluVRP
2) an efficient method to determine the visiting order within the clusters (based on a transformation of each global route into a classical TSP which Concorde TSP then solves.

The results of computer simulations on a set of 168 benchmark cases from the literature showed that their two-level solution strategy is better than other ways of solving the CluVRP. It was able to improve 47 of the best-known solutions out of 168. Furthermore, Md. Anisul Islam, Yuvraj Gajpal, and Tarek Y. ElMekkawy presented a hybrid PSO algorithm for solving the CluVRP in 2021. The method integrated the local optimal improvement powers of VNS with the swarm-based diversification capabilities of PSO. It demonstrated encouraging results when evaluated on benchmark examples, discovering new best-known solutions for 138 out of 293 instances with an average CPU time of 6.99 seconds. It also included new features such as the use of two different types of particles and a strategy for improving personal best solutions. Their method offers application potential in scenarios such as distribution logistics with a CO2 emission cap, transportation of perishable items, and military operations. They emphasized that it has its limits and suggested that future studies investigate the possibility of combining PSO with other metaheuristics and adding real-world variables like time windows and multi-depots. Finally, Matheus Freitas, Joo Marcos Pereira Silva, and Eduardo Uchoa (2023) proposed three models for the CluVRP, among other models, for solving alternative expansions of the VRP. One of these, F3-CluVRP, demonstrated superior performance, surpassing the best exact algorithm in literature, BC from Battarra et al. (2014). In addition, F3-CluVRP demonstrated a remarkable level of performance by resolving all extremely large instances of the Li set involving up to 1200 customers. The F3-CluVRP model unifies two significant VRP variations by reducing the CluVRP to a Generalized Vehicle Routing Problem (GVRP) (Ghiani & Improta, 2000) over a directed graph. The utilization of GVRP graph reductions was one reason that contributed to F3-CluVRP's excellent performance.

3. Implementation

3.1. Problem formulation

The CluVRP can be illustrated as a complete undirected graph with nodes V and edges E: $G = (V, E)$. In addition, the set $V = \{1, \ldots, n, n+1\}$ can be used to represent the vertices of the graph, with $V_1, V_2, \ldots, V_{n-1}, V_n$ being the customers, and V_{n+1} the depot, which hosts a fleet of vehicles with identical characteristics. Each vehicle has Q maximum capacity. The vertices are organized into H vertice clusters $VC_1, VC_2, \ldots, VC_{H-1}, VC_H$. The depot cluster is also defined as $VC_{H+1} = \{H+1\}$. All customer clusters $VC_h,\ h \in \{1, 2, \ldots, H\}$, have a demand $d_h > 0$ and cardinality $|VC_h| > 0$. All edges $\{i, j\} \in E$ have associated routing costs c_{ij}. The implementation's task is:

1) To determine a set of feasible cluster routes $CR_b,\ b \in \{1,2, \ldots, B\}, B \leq H$ where each cluster route contains one or more clusters VC and can be completed by a single vehicle.
2) To arrange the nodes of the clusters of each cluster route into a corresponding node route.

3) Complete the above steps with minimum routing costs while serving each customer only once.

A cluster route is feasible if:

1) It starts and ends at the depot $n + 1$
2) it respects the clustering, meaning that:
   a) In the case of hard clustering, if customer $i \in VC_h$ is visited then all other customers in $VC_h \{ i \}$ are visited directly before or after customer i without any other intermediate customers
   b) In the case of pseudo-soft clustering (explained later in the section "Solving the low-level routing problem") if a cluster route is assigned to a vehicle, the vehicle may visit the cluster route's nodes in the most profitable way. That means that interruptions in the service of a cluster are allowed if more profitable moves exist in the same cluster route.
3) Capacity constraints per cluster route are not violated.

3.2. Instances

The solved instances are based on the CVRP instance sets of Golden et al. 1998, Li et al. 2005, and VeRoLog Members VRP 2016.

These were retrieved from www.vrp-rep.org, and their format is VRP-REP compliant. After extracting the data from the xml files and arranging it to a dictionary, the CVRP instances were converted to CluVRP instances.

3.3. Clustering

To convert a CVRP instance to a CluVRP one, KMeans clustering is used with an initial number of clusters, that rises if the created clusters violate the vehicle capacity constraint (**Algorithm 1**).

Algorithm 1: Pseudo-code for Instance generator — **SolverPackage\Clustering.py\Line 54**

```
number of clusters ← N
capacity is violated ← True
while capacity is violated:
    cluster the nodes with KMeans (n_clusters = number of clusters)
    calculate each cluster's demand
    if cluster demand_i < Q for i ∈ {1, 2, ..., N}:
        capacity is violated ← False
    else:
        number of clusters ← number of clusters + 1
    end
end
```

For example, below (**Figure 2**) are the results of converting the 1st instance of the Golden set (**Figure 1**) from CVRP to CluVRP.

Figure 1 - CVRP instance

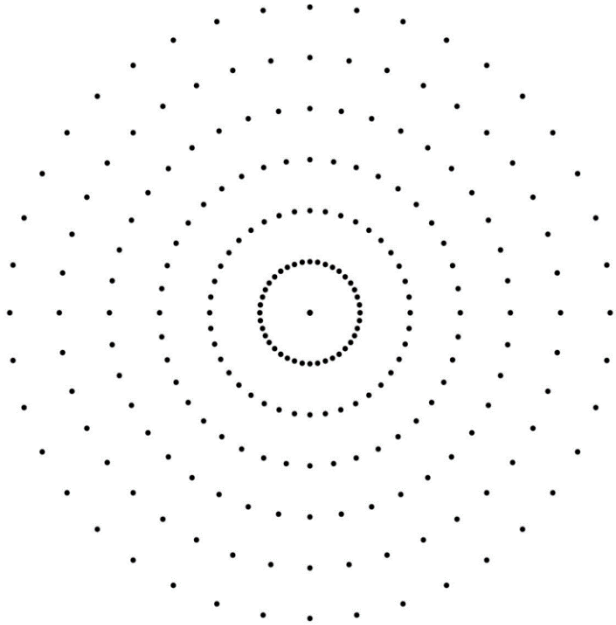
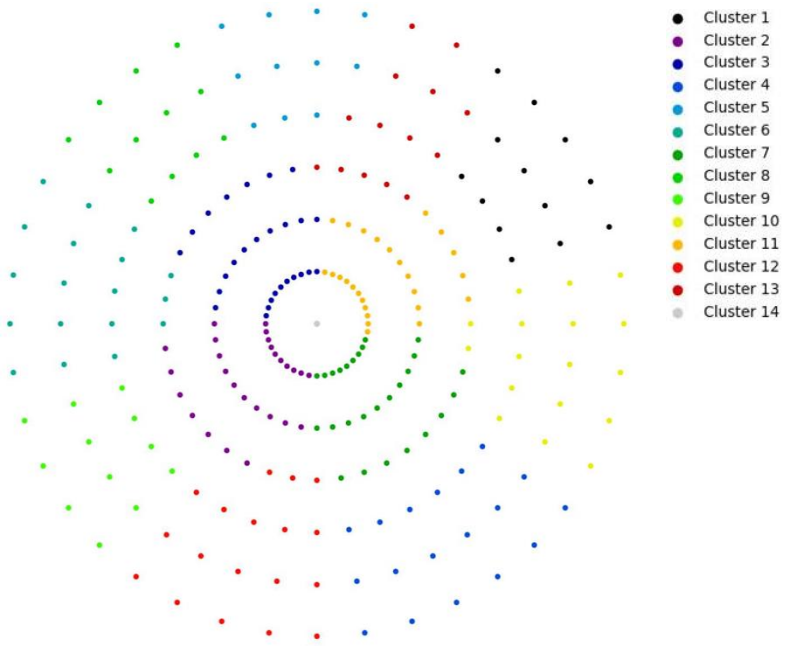


Figure 2 - CluVRP instance



3.4. Methodologies

As mentioned earlier, the methodologies employed in this work are based on the cluster-first/route-second category of heuristics (Bowerman, Calamiand, & Hall, 1994), which means that the CluVRP divides into two sub-problems:

1) The high-level problem of routing the clusters.
2) The low-level problem of routing the customers.

This way, CluVRP reduces to a combination of two well-known problems, the CVRP and the Travelling Salesman Problem (TSP).

3.4.1. Solving the high-level routing problem

The high-level problem (CVRP) is tackled by implementing the Clarke and Wright algorithm in Python (Clarke & Wright, 1964). The Clarke and Wright algorithm, also known as the "Savings Algorithm", starts by calculating the "savings" that would be achieved by combining any two customer routes. In this case, the savings from merging two clusters into a cluster route will be considered. The cluster distance matrix is calculated from the coordinates of each cluster's centroid. The savings from merging two clusters i, j are calculated by subtracting the distance between the two clusters $d(i,j)$ from the sum of the distances of the two individual clusters from the depot $d(D,i) + d(D,j)$. The algorithm then repeatedly selects the highest savings cluster pairs and merges their

routes, if possible, until all clusters have been visited. It is a greedy approach, meaning that it makes locally optimal choices at each step in the hope that these choices will lead to a globally decent solution. The output of this algorithm will be a list of cluster routes. Each of these cluster routes is now a low-level routing problem.

3.4.2. Solving the low-level routing problem

The low-level problem (open-type TSPs) is solved using the Nearest Neighbor (NN) algorithm. Here is where the previously mentioned pseudo-soft clustering (hereon referred to as soft clustering for simplicity) comes into place.

In the case of solving with hard clustering, the NN does not allow interruptions while serving a cluster in a cluster route. Specifically, for a given cluster route, the vehicle travels to the closest node of the first cluster-in-route to the depot, serves all the customers of cluster according to the NN algorithm, and then serves the rest of the clusters similarly (**Algorithm 2, Lines 3-9**). Moreover, during the optimization phase of the initial solution, each cluster route must be segmented into sub-routes depending on the number of clusters that reside in it. After that, the optimization technique must be applied to each sub-route separately. As a result, each cluster route is solved with as many open-type TSPs as the number of clusters in the cluster route (**Algorithm 3, Lines 2-7**).

However, since the cluster routes have already been created by the Clarke & Wright algorithm and they already meet the capacity constraints, allowing the vehicle to interrupt the service of a cluster to make a more profitable visit (to a node in another cluster) on the same cluster route makes the problem even simpler. That is because, now each cluster route can be tackled as a single closed-type TSP (**Algorithm 2, Lines 10-12 & Algorithm 3, Lines 8-9**). This idea led me to test how the solutions are affected when facing the problem as soft clustered.

Algorithm 2: Pseudo-code for applying the Nearest Neighbor algorithm **SolverPackage\Tsp.py\ Line 82**

```
1  for cluster route in cluster routes:
2      node route ← [depot, depot ]
3      if hard clustered:
4          for cluster in clusters:
5              if cluster ≠ depot cluster
6                  node list ← nodes of cluster
7                  node route ← nearest neighbor (node list, node route)
8              end
9          end
10     else if soft clustered:
11         node list ← all nodes in cluster route
12         node route ← nearest neighbor (node list)
13     end
14 end
```

Algorithm 3: Pseudo-code for applying VND to a solution **SolverPackage\Optimisation.py\ Line 343**

```
For route in initial solution:
    if hard clustered:
        subroutes ← create subroutes (route)
        for subroute in subroutes
            optimised subroute ← VND(subroute)
        end
        optimised route ← combined optimised subroutes
    else if soft clustered:
        optimised route ← VND(route)
    end
end
```

3.4.3. Optimization of the solution

To further improve the initial solution, three similar optimization approaches are utilized:

1) A Variable Neighborhood Descent (VND) algorithm.
2) A Variable Neighborhood Descent algorithm using multiple restarts (VND MR).
3) A General Variable Neighborhood Search (GVNS) algorithm

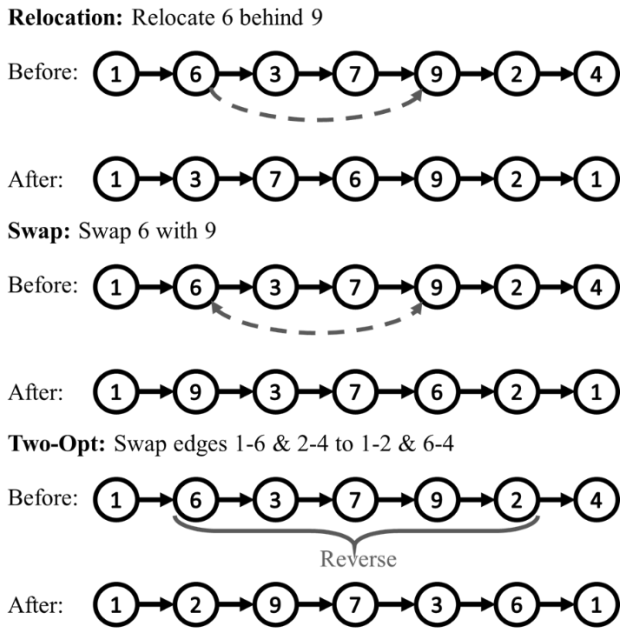
The VND algorithm is a deterministic variation of the Variable Neighborhood Search (VNS) algorithm (Mladenović & Hansen, 1997) where the solution space is explored in a non-stochastic way by changing the way of moving in the solution space when a local optimum is met. The algorithm terminates when any move types cannot improve the solution (**Algorithm 4**). The move types used in this implementation are the relocation move (which relocates a node somewhere else in the route), the swap move (which swaps two nodes in the same route), and the two-opt move (which swaps the edges of two pairs of nodes in the route). A visual representation of the move types can also be found in **Figure 3** below.

Algorithm 4: Pseudo-code of VND method **SolverPackage\Optimization.py\Line 262**

```
best route ← route
termination condition ← False
move type ← 0
failed to improve ← 0
while termination condition is False:
    if move type = 0
        find best relocation move in best route
        if move cost ← 0.00001:
            apply relocation move to best route
            failed to improve ← 0
        else:
            move type ← move type + 1
            failed to improve ← failed to improve + 1
```

```
            end
        else if move type = 1
            find best swap move in best route
            if move cost ← 0.00001:
                apply swap move to best route
                failed to improve ← 0
            else:
                move type ← move type + 1
                failed to improve ← failed to improve + 1
            end
        else if move type = 2
            find best two opt move in best route
            if move cost ← 0.00001:
                apply two opt move to best route
                failed to improve ← 0
            else:
                move type ← move type + 1
                failed to improve ← failed to improve + 1
            end
        end
        if failed to improve = 3:
            termination condition ← True
        end
    end
```

Figure 3 - Move types



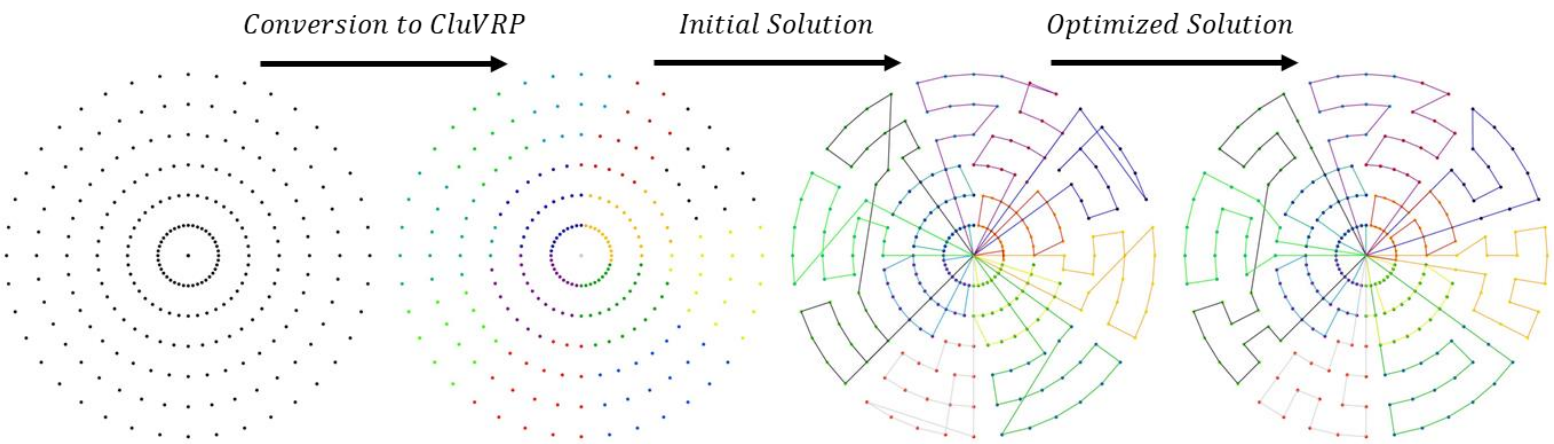
The second technique implements a stochastic element in the procedure to further search the solution space. This is achieved by applying the VND method to multiple initial solutions constructed by using the NN algorithm with a Restricted Candidate List (RCL) (Feo & Resende, 1995). The RCL is a list of the closest neighbors of the current node that is calculated in each iteration of NN. One of the candidates is selected as the next visited node; hence, the resulting route is comprised of edges that connect nodes that are relatively close to each other. This practice leads to randomized initial solutions that are not completely inefficient. In this implementation, an RCL with a length of 3 is used (**Algorithm 5**).

Algorithm 5: Pseudo-code for VND with multiple restarts **SolverPackage\Solver.py\ Line 185**

```
1  max iterations ← 1000
2  limit ← 500
3  RCL length ← 3
4  best solution ← solve the instance with VND (no RCL)
5  failed to improve ← 0
6  for iteration in range(0, max iterations):
7      solution ← solve with RCL and VND
8      if cost_solution < cost_best solution:
9          best solution ← solution
10         failed to improve ← 0
11     else:
12         failed to improve ← failed to improve + 1
13     end
14     if failed to improve = limit
15         break loop
16     end
17 end
```

Finally, the third method is a more powerful version of the VNS, the GVNS (Hansen, Mladenović, & Moreno Pérez, 2010). In this expansion of VNS, the simple local search algorithm that is used to descend in the solution space is replaced by the previously described VND, which uses many move-types instead of one to explore the neighborhood. The algorithm terminates when a user-defined amount of time has passed. Alternatively, one could stop the algorithm even earlier, when it fails to improve the solution after a user-defined number of iterations. In this work, the algorithm is executed for 60 seconds, without a limit on the iterations without improvement (**Algorithm 6**). A visual representation of the whole pipeline can be seen in **Figure 4** below:

Figure 4- Implementation pipeline



Algorithm 6: Pseudo-code for GVNS **SolverPackage\Solver.py\ Line 229**

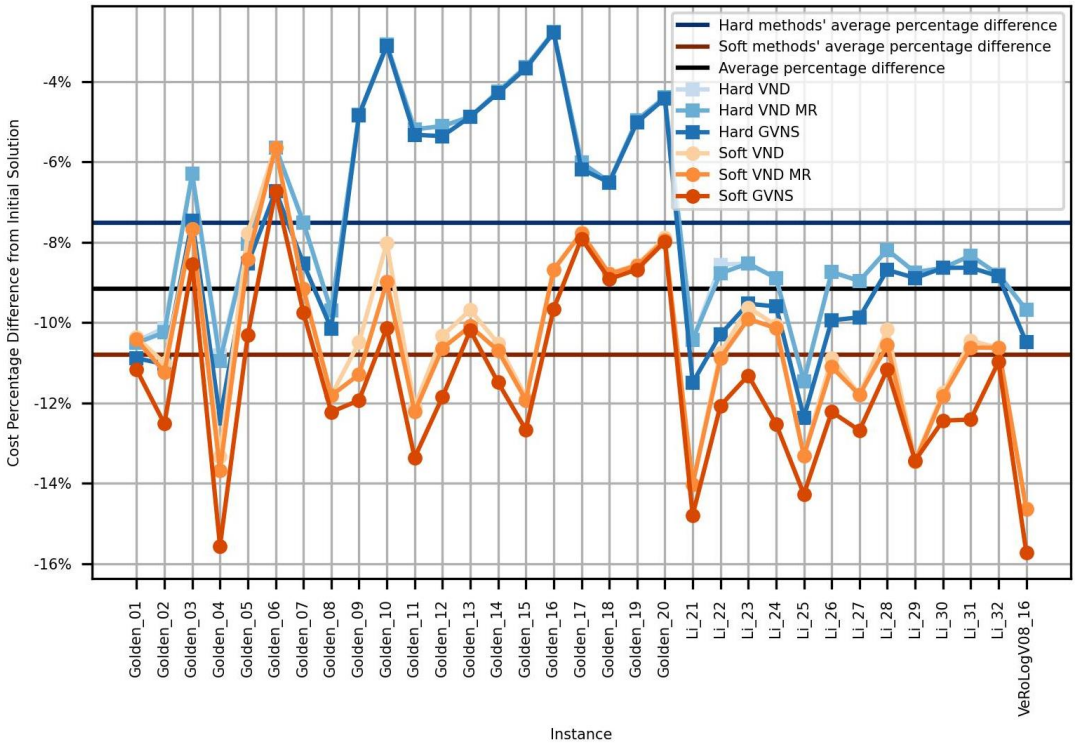
```
CPU time ← 60
elapsed time ← 0
start ← current time
best solution ← solve the instance with nearest neighbor and VND (no RCL)
while elapsed time < CPU time
    current solution ← randomly select neighboring solution
    optimised current solution ← optimise current solution with VND
    if cost_optimised current solution < cost_best solution:
        best solution ← optimised current solution
    end
    end ← current time
    elapsed time ← end − start
end
```

4. Results

4.1. Analysis

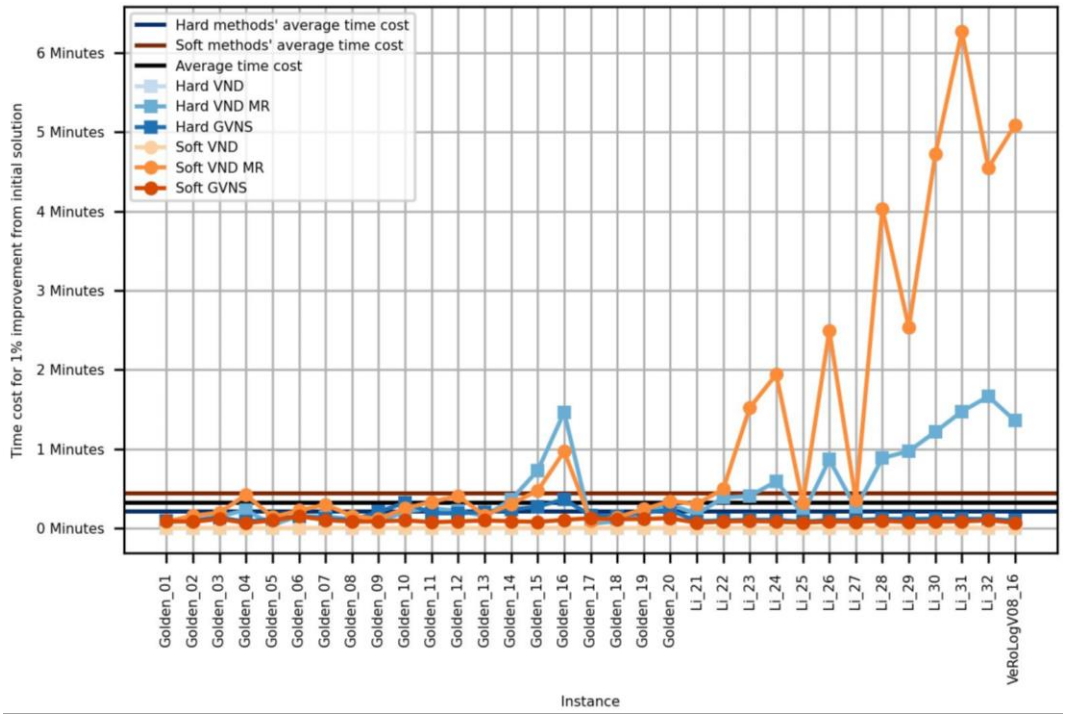
Moving on to the results for each instance, a detailed table can be found in the appendix for every instance set. An initial idea about the performance of the methods can be found in Figure 5 below, where the cost percentage difference from the initial solution (Clarke & Wright & Nearest Neighbor – Hard Clustered) is presented for every technique. On average, the techniques employed improve the initial solution by 9%. It is evident that soft methods outperform hard ones by an average of ~ 3%. In nearly all instances, soft techniques produce better results than hard ones, especially in the Golden 09-16 instances. The soft GVNS is the best-performing method overall.

Figure 5 - Cost Percentage Difference from Initial Solution per Instance



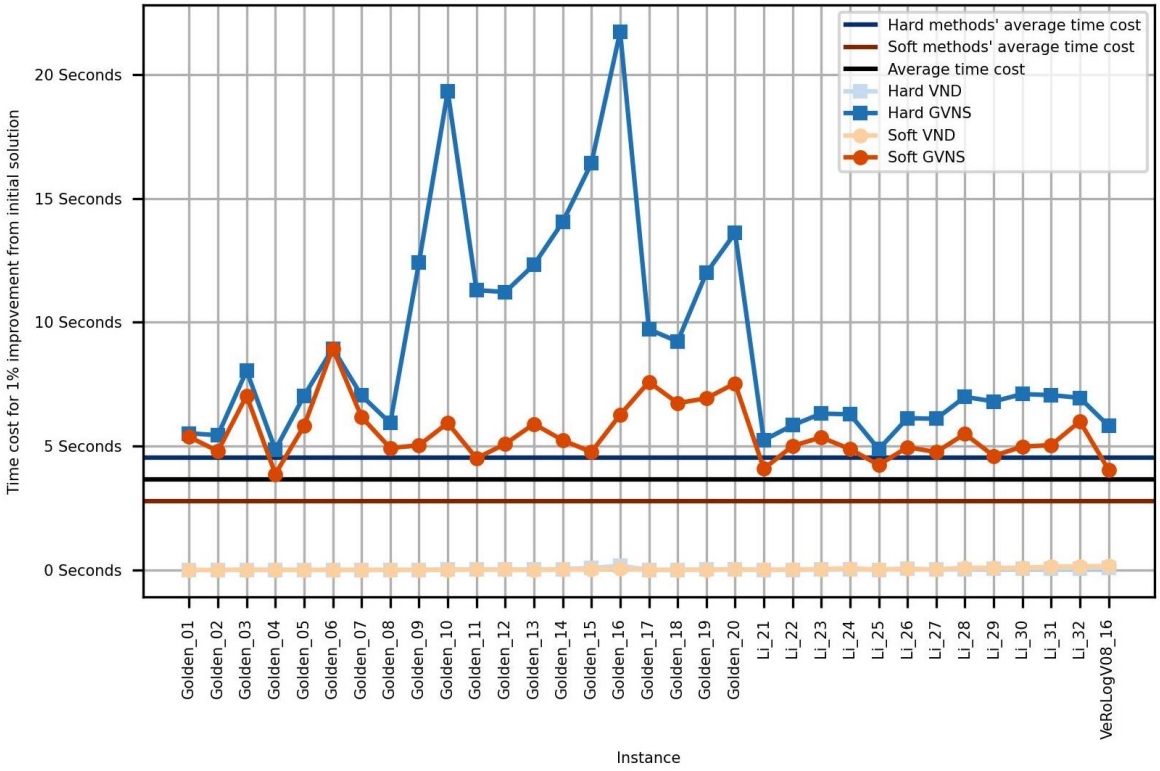
To get a clearer image of the performance of each optimization technique, the dimension of computational time is incorporated into the results by calculating the time cost for every 1% improvement from the cost of the initial solution (**Figure 6**).

Figure 6 - Time cost for 1% improvement from initial solution



After a first look, it is noticeable that the multiple restart methods skew the results, as an excessive amount of computational time is required for the last instances. Therefore, to better interpret the findings, it is necessary to isolate the rest of the methods (**Figure 7**). On average, four seconds are required for a 1% improvement. Moreover, soft clustered instances need, on average, two less seconds than hard ones to improve the initial solution by 1%. Interestingly, the simple VND methods for hard and soft clustered instances are very efficient, achieving a 1% improvement almost instantaneously, but that is offset by the limited solution neighborhood they can explore. Finally, the hard GVNS loses some of its efficiency in Golden instances 09-16, while soft GVNS maintains the same efficiency level in all the cases.

Figure 7 - Time cost for 1% improvement from initial solution (without VND MR)



Moving on, an examination of the actual computational time that each method needed to complete is made. In **Figure 8** below, the multiple restart methods skew the results again. This happens because the construction of a randomized initial solution and its optimization happen for a predetermined number of steps, in this case, at least 500 times. Furthermore, for the final instances that are computationally demanding even for the simple VND optimization technique, the slight extra difference in time needed to complete one iteration accumulates over the minimum 500 iterations, making the multiple restarts method time-consuming and inefficient.

To make a more meaningful interpretation of execution time, in **Figure 9** below, the results of multiple restart methods are removed. Moreover, the results of the GVNS methods are also withdrawn, as their execution time is fixed at 60 seconds.

Figure 8 – Computational time for all methods

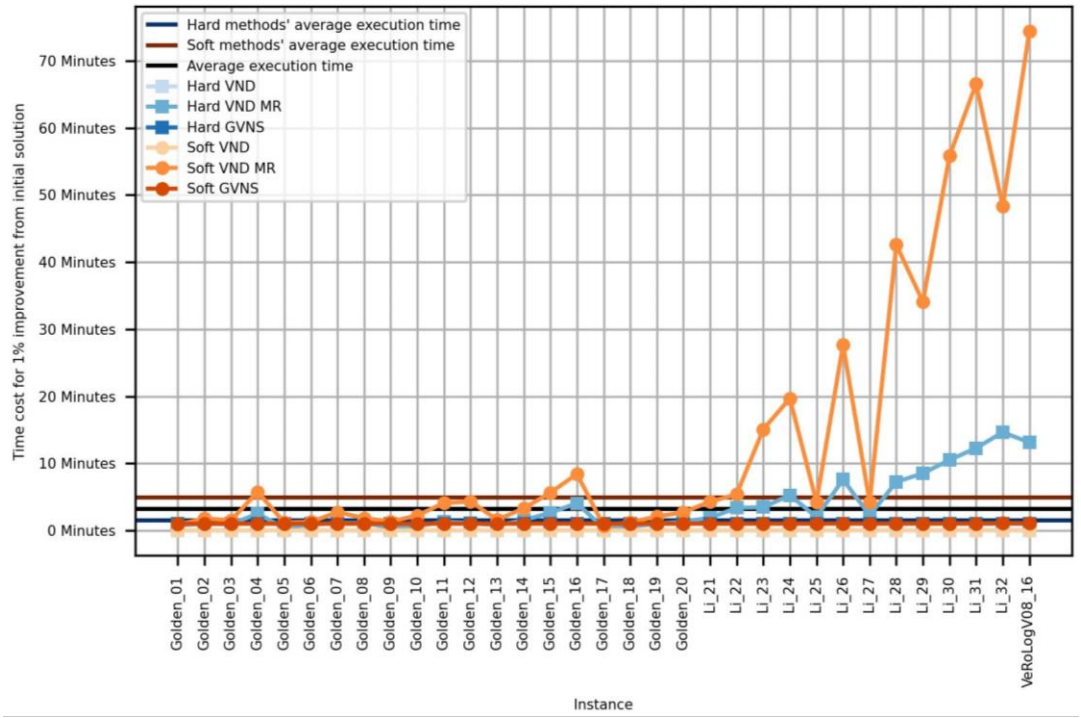
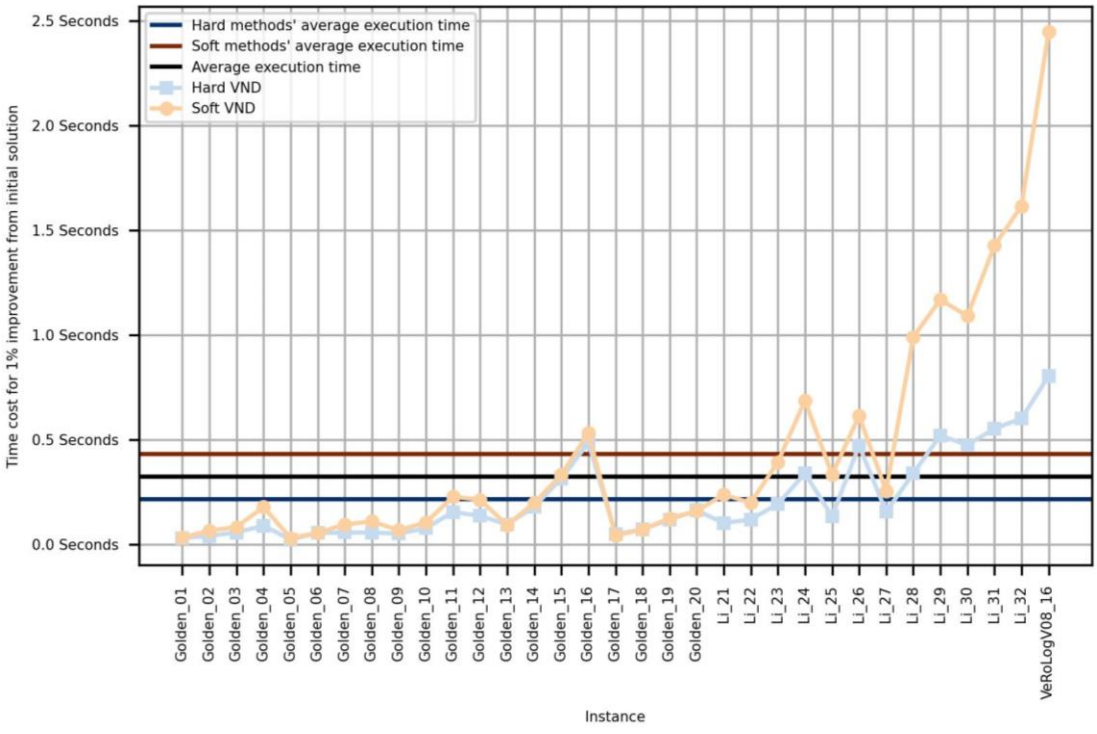


Figure 9 – Computational time for VND methods



For **Figure 9** above, interesting is the fact that in some instances, the Soft VND method takes significantly more time to complete, sometimes more than double the time that the Hard VND needs. This should happen because, in some cases, the possible moves for the soft optimization technique are way more numerous. That is because hard constraints on intra-cluster routing severely limit the move types that optimize the route in some instances. Hence, the soft method requires, on average, almost twice the computational time of the hard method to complete. This, however, does not change the fact that the soft GVNS (which relies on soft VND) achieves both better results and efficiency than the hard GVNS) in the 60-second window, as evidenced by **Figure 5** and **Figure 7**.

Following is an analysis of how the different aspects of the instances affect execution time for the simple VND methods (**Figure 10**) (the isolation of these methods happens for the same reasons mentioned in the previous paragraph). For variables “Number of nodes”, “Average distance between nodes” and “Vehicle capacity”, there seems to be a positive relationship between them and computational time. That is, as their value rises, the computational time also rises. Initially, it does not seem that a relation between average node demand and computational time exists. To further examine the said relation, a new variable is created that considers both average node demand and vehicle capacity. This variable is the average number of nodes that could be serviced by a vehicle based on the average node demand of the instance (Vehicle capacity / Average node demand). The results can be seen in **Figure 11**.

Figure 10 - Computational time relation with various instance metrics

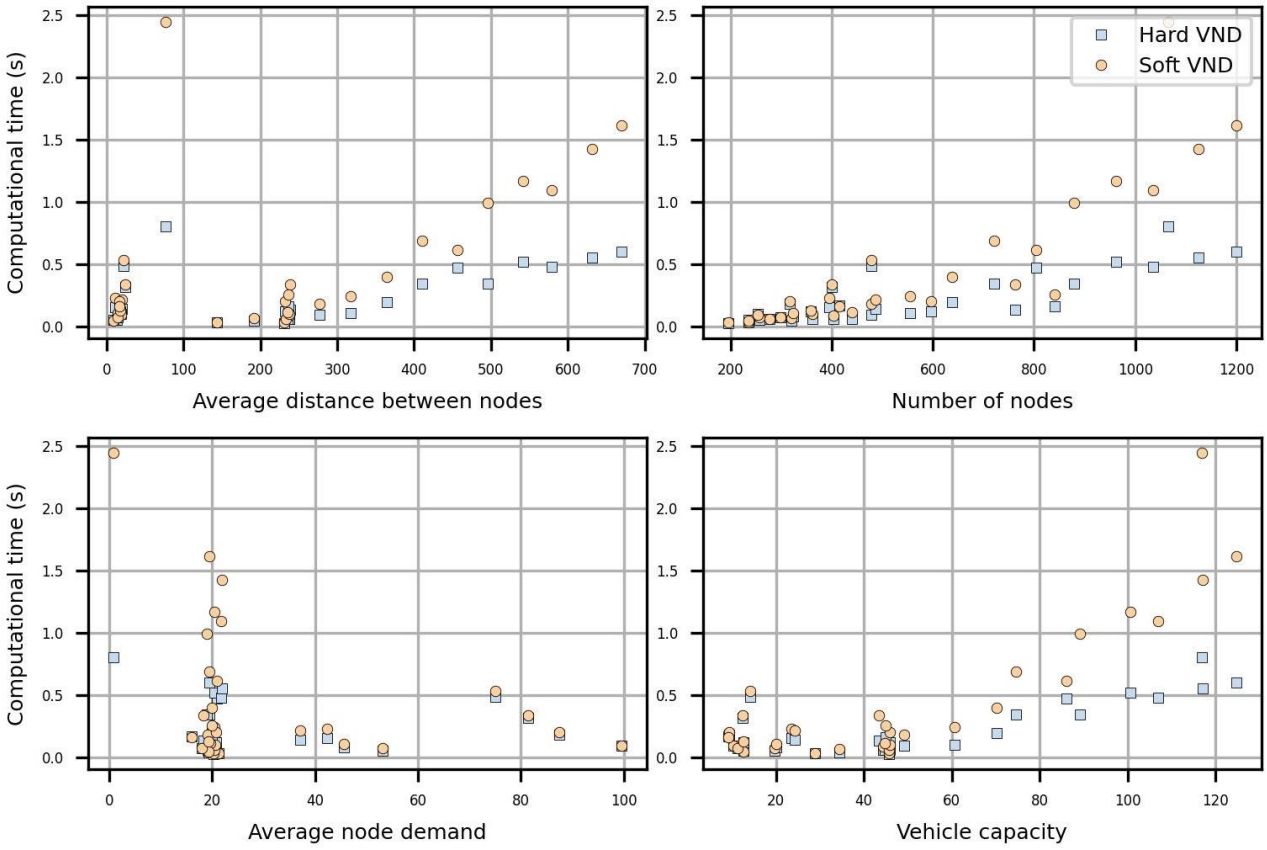
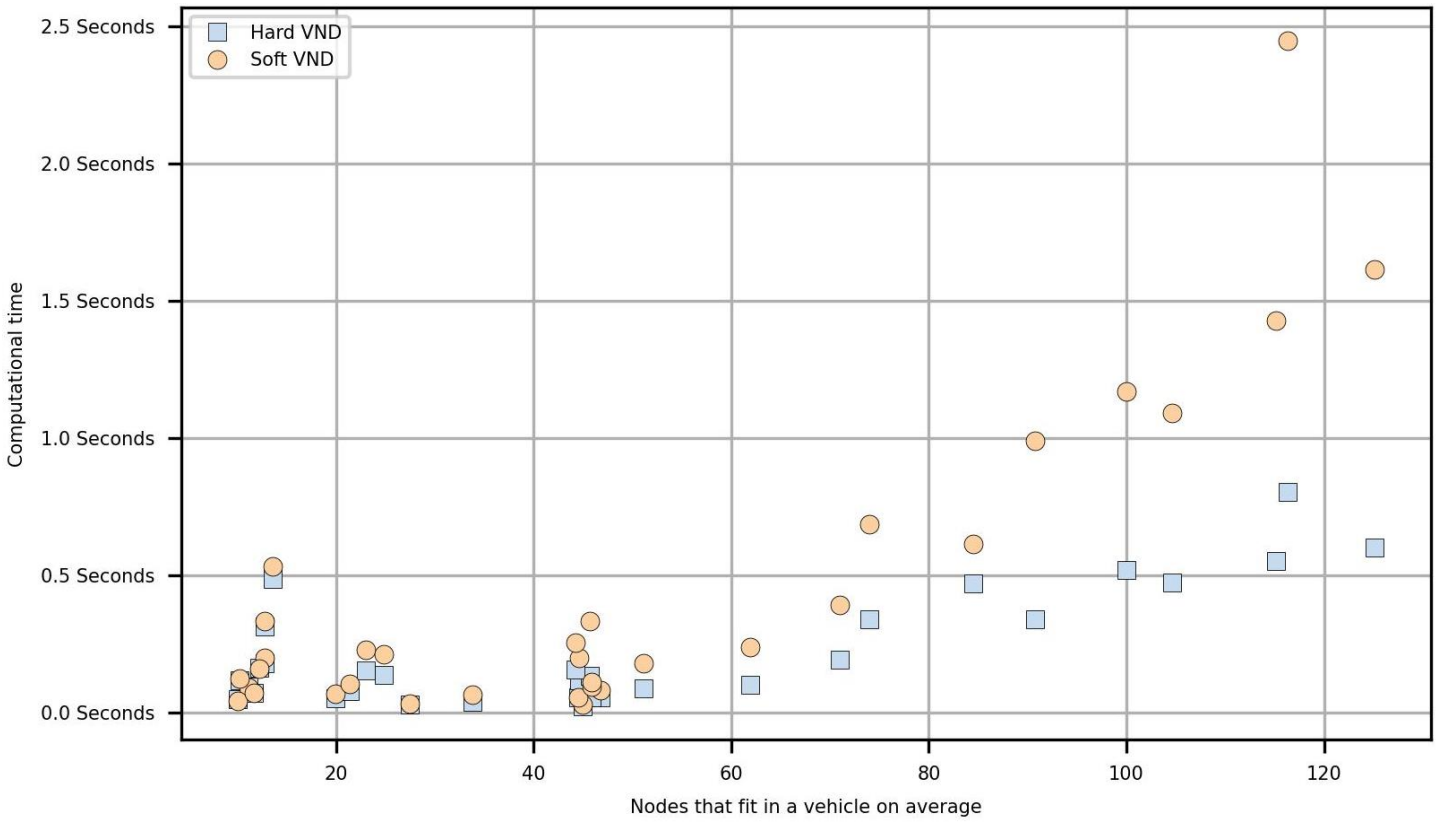


Figure 11 – Relation of computational time with average number of nodes that fit in vehicle



It is apparent that when this metric's value rises, the computational time also rises.

All in all, GVNS is the most cost-effective method. In addition, the simple VND method performs unexpectedly well considering the computational time it requires, while the VND with multiple restarts method was prohibitively computationally demanding in some instances. It slightly outperformed GVNS only in 1 of the 33 cases, while it took considerably longer to complete in most cases. Hence, VND MR is deemed an inefficient method for optimizing the solution. Furthermore, it is shown that tackling the problem as soft clustered rather than hard clustered led to cost reductions of 3% on average (**Figure 5)**, and with GVNS, that is done in the same amount of time (60s). Finally, it is indicated that as the number of nodes, the average distance between nodes, and the average number of nodes that fit into the vehicle rise, more computational time is needed to optimize the solution.

4.2. Hard vs soft solutions visualized

As mentioned in the previous section, soft methods outperform hard ones. In this part, some solutions produced by the GVNS optimization technique, the best-performing method for hard and soft clustered problems, are visualized. In this way, we grasp better why the soft methods outperform the hard methods.

Figure 11 - Golden 03: Hard GVNS with cost = 12,410 | Soft GVNS with cost = 12,264

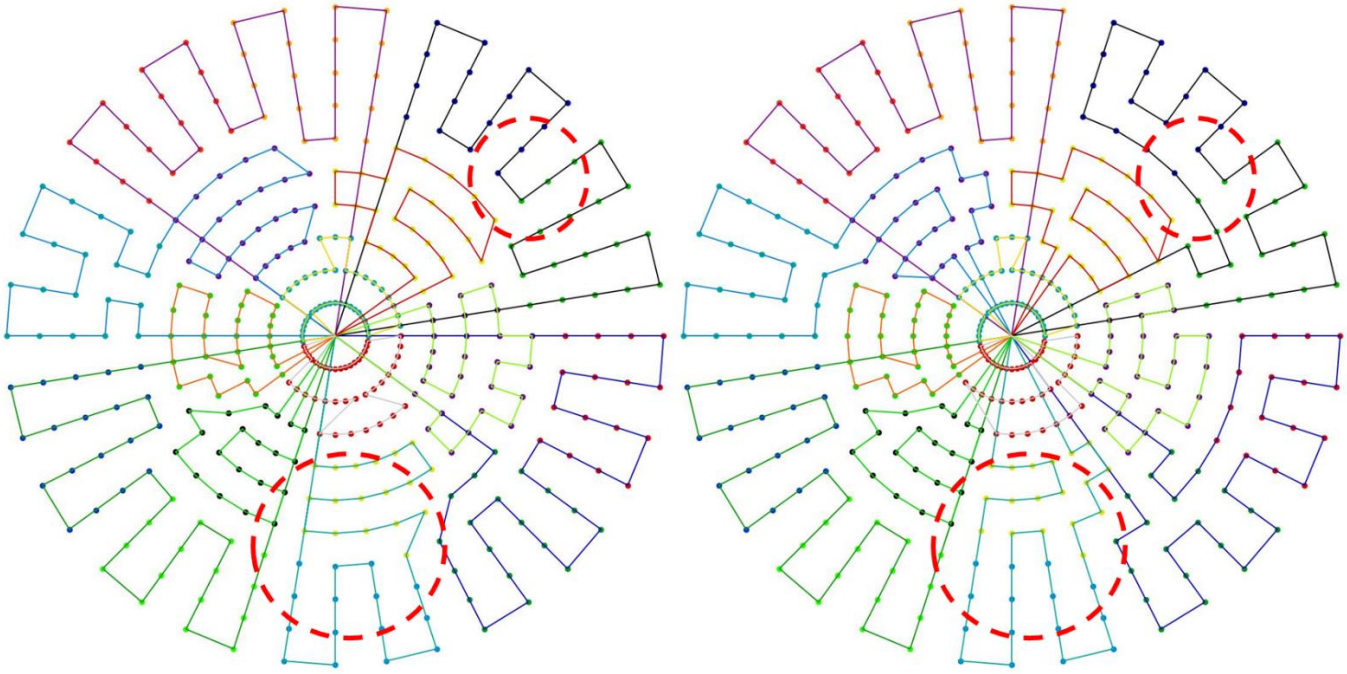


Figure 12 - Li 22: Hard GVNS with cost = 15,791 | Soft GVNS with cost = 15,479

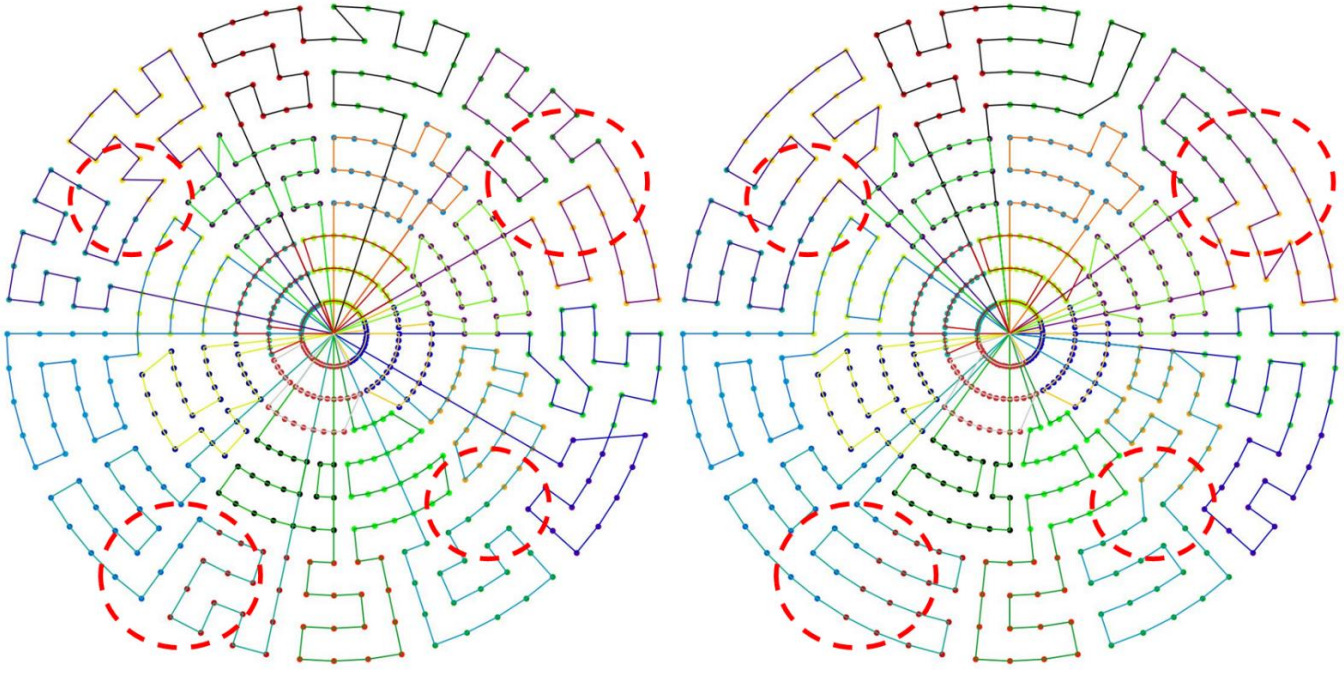
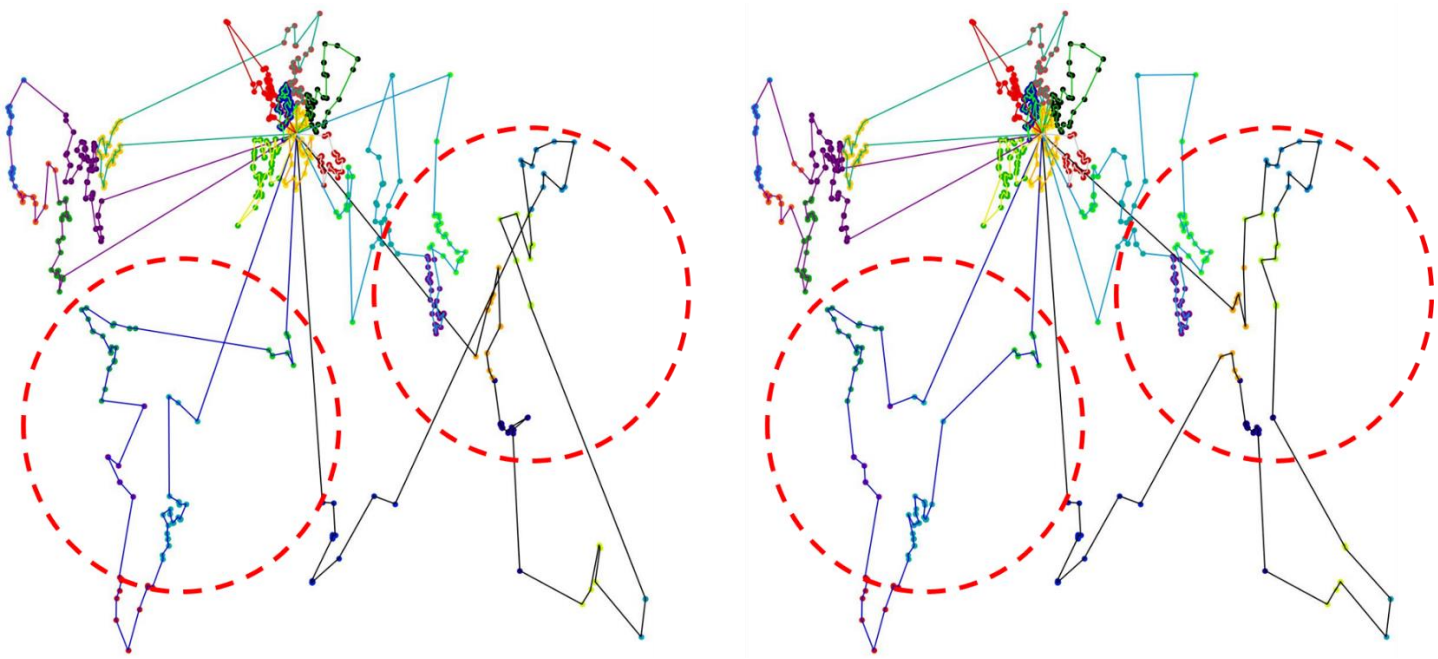


Figure 13 - VeRoLogV08_16: Hard GVNS cost = 2593 | Soft GVNS cost = 2442



In **Figures 11**,**12**, and **13** above, it is noticeable that Hard GVNS, in its effort to optimize the solution while respecting the cluster constraints, makes some inefficient routing choices. This is even clearer in the Verolog instance (**Figure 13**).

5. Conclusions

5.1. The problem

In this thesis, the CluVRP is addressed. It is an extension of the classical VRP, with several optimization methodologies proposed in the literature and many practical applications such as delivery operations, warehousing, and emergency contexts. The constraints imposed by this problem concern the assignment of customers into clusters and how they are serviced afterward.

5.2. The implementation

In this implementation, the problem is divided into two problems: the high-level routing problem of visiting the customers' clusters and the low-level problem of intra-cluster routing. Two greedy approaches (Clarke & Wright, Nearest Neighbor) are employed to get a fast initial solution, while three similar metaheuristic methods (VND, VND multiple restart, GVNS) are utilized to optimize said initial solution.

5.3. The results

Moving on, the implementation is tested on various VRP benchmark instances that have been adapted for the CluVRP, and the results are analyzed to determine which method is more time- and cost-effective. After

examining the findings, it was clear that the GVNS approach was the most cost-effective method. Additionally, the simple VND approach performed admirably well for the computational time it needed. Moreover, the VND with multiple restarts method was prohibitively computationally demanding in some cases, while it delivered better results than GVNS only in 1 of the 33 instances. To further examine the results, an analysis was made regarding the relation of each method's performance to the different characteristics (number of nodes, average demand, average distance, vehicle capacity) of the adapted instances. Finally, it is shown that allowing interruptions in the service of a cluster to serve a customer of another cluster in a limited way (pseudo-soft clustering) led to better solutions in the same amount of time (for the GVNS method).

5.4. Method shortcomings

Three known problems hinder this implementation:

1) The way the clusters are made could be more efficient. A more sophisticated way of grouping the customers into clusters without violating capacity constraints can be explored. In this way, the number of vehicles used could also be optimized.
2) The initial order of visiting the clusters, produced by the Clarke & Wright algorithm, is not optimized in this work. An intermediate step of using a metaheuristic algorithm to optimize the order of visiting the clusters before solving the low-level routing problem could further improve the solutions.
3) The use of a distance matrix possibly prohibits the utilization of the algorithm on large instances.

6. References

Applegate, D., Bixby, R., Chvàtal, V., & Cook, W. (2001). *Concorde TSP solver.* Ανάκτηση από http://www.tsp.gatech.edu/concorde/index.html.

Battara, M., Erdoğan, G., & Vigo, D. (2014). Exact algorithms for the clustered vehicle routing problem. *Oper. Res. 62 (1),* 58-71.

Bowerman, R., Calamiand, P., & Hall, G. B. (1994). The spacefilling curve with optimal partitioning heuristic for the vehicle routing problem. *European Journal of Operational Research, 76,* 128–142.

Charon, I., & Hudry, O. (1993). The noising method: A new method for combinatorial optimization. *Operations Research Letters, 14(3),* 133–137.

Chisman, J. A. (1975). The clustered traveling salesman problem. *Computers & Operations Research, 2 (2),* 115–119.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research 12 (4),* 568-581.

Dantzig, G. B., & Ramser, R. H. (1959). The truck dispatching problem. . *Management Science 6*, 80-91.

de Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research, 182 (2)*, 481-501.

Defryn, C., & Sörensen, K. (2017). A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Comput. Oper. Res. 83*, 78-94.

Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., & Vogel, U. (2011). Vehicle routing with compartments: Applications, modelling and heuristics. *OR Spectrum 33(4)*, 885–914.

Dondo, R., & Cerdá, J. (2007). A cluster-based optimization approach for the multidepot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research, 176(3)*, 1478–1507.

Expósito-Izquierdo, C., Rossi, A., & Sevaux, M. (2016). A two-level solution approach to solve the clustered capacitated vehicle routing problem. *Comput. Ind. Eng. 91*, 274–289.

Feo, T., & Resende, M. (1995). Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization. 6.*, 109-133.

Freitas, M., Silva, J. M., & Uchoa, E. (2023). A unified exact approach for Clustered and Generalized Vehicle Routing Problems. *Computers & Operations Research, 149*, 106040.

Golden, B. L., Raghavan, S., & Wasil, E. A. (Επιμ.). (2008). *The vehicle routing problem: Latest advances and new challenges. Operations research/Computer science interfaces series* (Vol. 43). Springer.

Golden, B. L., Wasil, E. A., Kelly, J. P., & Chao, I.-M. (1998). The impact of metaheuristics on solving the vehicle routing problem: **Algorithm**s, problem sets, and computational results. *T. G. Crainic & G. Laporte (Eds.), Fleet management and logistics*, 33-56.

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

Haugland, D. H. (2007). Designing delivery districts for the vehicle routing problem with stochastic demands. *European Journal of Operational Research, 180(3)*, 997–1010.

Hintsch, T., & Irnich, S. (2018). Large multiple neighborhood search for the clustered vehicle-routing problem. *European J. Oper. Res. 270 (1)*, 118–131.

Islam, M. A., Gajpal, Y., & Y., E. T. (2021). Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem,. *Applied Soft Computing, 110*, 107655.

Li, F., Golden, B. L., & Wasil, E. A. (2005). Very large-scale vehicle routing: New test problems, algorithms, and results. *Computers & Operations Research 32 (5)*, 1165–1179.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research 24*, 1097–1100.

Pop, P. C., Fuksz, L., Marc, A. H., & Sabo, C. (2018). A novel two-level optimization approach for clustered vehicle routing problem. *Computers & Industrial Engineering, 115*, 304-318.

Schmid, V., Doerner, K. F., & Laporte, G. (2013). Rich routing problems arising in supply chain management. *European Journal of Operational Research, 224(3)*, 435-448.

Vidal, T., Battarra, M., Subramanian, A., & Erdogan, G. (2015). Hybrid metaheuristics for the clustered vehicle routing problem. *Computers & Operations Research 48 (0)*, 87-99.

Weintraub, A., Aboud, J., Fernández, C., Laporte, G., & Ramírez, E. (1999). An emergency vehicle dispatching system for an electric utility in Chile. *Journal of the Operational Research Society 7*, 690-696.

7. Appendix

7.1. Code files (.py) and instances (.xml)

All the code files and instances needed to test the methodologies presented in this work, as well as their documentation, can be found here.

7.2. Results analysis (Jupyter)

The results' analysis can be found in this Jupiter notebook.

7.3. Results table

The results table used to create the plots can be found here.

7.4. Plots and routes of presented solutions

The plots and the routes that correspond to the results' analysis in section 4, can be made available upon request.