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

This report considers the Clustered Vehicle Routing Problem (CluVRP), an extension of the renowned Vehicle Routing Problem (VRP). After a brief literature review of the optimization methodologies that have been applied to the problem and some real-world applications, an implementation of some well-known methodologies combined to solve the problem is presented. Particularly, the problem is divided into two routing problems, the high-level routing problem of visiting the customers' 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 obtain a better solution. Finally, 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.

1. Introduction

The Vehicle Routing Problem (VRP) is a classic combinatorial optimization problem in the field of Operations Research and logistics. It was first introduced by Dantzig and Ramser in 1959 and has since become one of the most widely studied and well-known problems in the community. 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 supply and demand constraints. Hence it requires a large amount of computational time to find optimal solutions. Even in its simplest form, it is considered to be NP-complete, meaning that no polynomial-bounded algorithm is likely to exist. Despite this, the problem has been widely studied due to its numerous real-world applications in transportation, network optimization, and distribution. It has been the subject of a large number of research papers and books, and 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 by Golden, Raghavan, and Wasil in 2008 (Golden, Raghavan, & Wasil, The vehicle routing problem: Latest advances and new challenges. Operations research/Computer science interfaces series, 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 all the time.

The purpose of this thesis is 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 Clustered Vehicle Routing Problem (CluVRP) is a variant of the classic Capacitated Vehicle Routing Problem (CVRP) which aims to minimize the total cost of routes for the distribution of goods to geographically dispersed customers. It is recognized that the problem is NP-hard, which means that an exact solution may be computationally infeasible for large-scale examples. CluVRP extends the CVRP by incorporating a restriction on customer clustering. It is divisible into the hard-clustered variety and the soft-clustered variation. The hard-clustered form organizes customers into clusters and imposes the limitation that if a route 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. The soft-clustered variation, on the other hand, is a relaxation of the hard-clustered variant in that visits to customers of the same cluster may be interrupted by visits to consumers of different clusters, if profitable. In this study, the hard-clustered problem will be the primary emphasis; however, a pseudo-soft clustering strategy will also be investigated. CluVRP was introduced in 2008 by Sevaux and Sorensen and is motivated by situations where physical constraints require all customers in a cluster to be served sequentially, or when clusters are used to force routes to have desirable characteristics, such as being more compact or easier for drivers to learn.

The CluVRP, like the CVRP, is an NP-hard issue, and exact solutions may be computationally impossible for large-scale cases. To solve the CluVRP, however, numerous approximate solution approaches have been presented. There are heuristics, metaheuristics, and accurate procedures among them. Heuristics are simple, quick methods that provide an approximation of the ideal solution to a problem, but do not guarantee it. Metaheuristics are approaches that increase the quality of solutions by combining heuristics with other techniques, such as local search, tabu search, and genetic algorithms. Exact methods, on the other hand, ensure an optimal solution, but are computationally costly and limited to small-scale applications.

2. Literature Review

2.1. Optimization Methodologies

As previously indicated, CluVRP was initially defined as a heuristic for decreasing the dimensions of large VRPs by Sevaux and Sorensen (2008). Their plan was to group consumers into clusters and execute routing over a significantly reduced problem consisting of the cluster centers. In a subsequent step, the 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. The suggested technique with the greatest performance depends on a preprocessing scheme that calculates 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. Hamiltonian path problems are solved in the best way possible with the TSP code Concorde (Applegate, Bixby, Chvàtal, & Cook, 2001). In 2015 Vidal et al. proposed three metaheuristics for the CluVRP algorithm. Two of them are based on local search iterations, while the other is a hybrid genetic algorithm. The evolutionary algorithm describes a route as a sequence consisting just of the first and last vertices in each cluster and employing expansive neighborhoods. It produces higher-quality results. 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.

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 dictates the routes between clusters, while the problem at the lowest level affects 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 on 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. By combining the local search at the customer level with the local search at the cluster level, it was possible to acquire higher-quality solutions in a shorter amount of time. In 71 out of 79 examples tested with small and medium-sized problems from the literature, the algorithm was able to swiftly obtain the known optimal values. 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 a Large Neighborhood Search (LNS) metaheuristic that employs multiple destruction and repair operators as well as a local improvement procedure based on Variable Neighborhood Descent (VND) with an ILS-based preprocessing phase (Iterated Local Search) that computes all routes within each cluster, taking into account all possible entry and exit points. An important concept is a new neighborhood particular to CluVRP, a generalization of the Balas–Simonetti neighborhood for the Asymmetric Traveling Salesman Problem (ATSP), that can simultaneously determine the permutation of clusters on each route and the entry and exit combinations. In a comparison of over 230 instances with the precise algorithm of Battarra et al. (2014), 217 identical solutions were achieved, while 7 were improved.

Later, Pop et al. (2018) devised a unique two-level optimization strategy 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 to deliver collections of global routes visiting the clusters, a constructive method to generate the initial population of the CluVRP, and an efficient method to determine the visiting order within the clusters based on a transformation of each global route into a classical TSP, which was then optimally computed using the Concorde TSP solver. 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 integrates 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 includes 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, though, that it has its limits, and they 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, even surpassing the best exact algorithm in the 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. In essence, the F3-CluVRP model reduces the CluVRP to a GVRP over a directed graph. This is intriguing, as it unifies two significant VRP variations. The utilization of GVRP graph reductions was one reason that contributed to F3-CluVRP's excellent performance.

2.2. Real world applications

The CluVRP becomes useful ҫhen clients to be visited are clustered. The availability of resources and the number of customers participating in delivery operations are the primary motivations for customer clustering. Service providers are often unable to meet all 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 as opposed to 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 certain criteria. In general, clusters could exist because:

1) They are geographically intrinsic in a customer base. 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 type of 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-intermixable 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 where their injuries are classified based on their severity, and individuals with the most severe injuries must receive care first, followed by the rest of severity levels.
   b) Electrical outages, where all customers with a high priority are gathered according to their
      their locations (clusters) and must be visited prior to 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.

In addition, a service provider can classify its consumers according to certain criteria when there are a significant number of them. This is referred to as the "districting problem" (Haugland, Ho, & Laporte, 2007), and it includes optimization criteria such as proximity, balance (differences in terms of number of customers, service demands, areas, service revenues, service times, etc. in the clusters.), cost-effectiveness, and time windows for the customers.

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

3. Implementation

3.1. Problem formulation

CluVRP can be described by a complete undirected graph $G = (V, E)$ with nodes V and edges E. The nodes comprise the set $V \setminus \{ n+1 \} = \{ 1, \ldots, n \}$ that represents the n customers and the node $n+1$ for the depot, where a homogeneous fleet of m vehicles is housed. The capacity of a vehicle is denoted by Q . The customers are partitioned into N clusters $V_1, V_2 , \ldots, V_{N-1} , V_N$. For the sake of convenience, we also define the depot cluster as $V_{N+1} = \{ N+1 \}$. All customer clusters V_h , $h \in \{ 1, 2, \ldots, N\}$, have a positive demand d_h and cardinality $\lambda_h = | V_h |$. All edges $\{ i, j \} \in E$ have associated routing costs c_{ij} . The task is to determine a set of m feasible routes with minimum total routing costs serving each customer exactly once. A route is feasible if:

1) It starts and ends at the depot $n+1$

2) It respects the clustering, meaning that if customer $i \in V_h$ is visited then all other customers in $V_h \; \{ i \}$ are visited directly before or after i without any other intermediate customers (a pseudo-soft clustering will also be examined, that will allow a vehicle to interrupt the service of a cluster in favor of another cluster under limitations)
3) The demand of the visited clusters does not exceed the vehicle capacity.

If all clusters are singletons $V_h = \{ h \}$, the CluVRP reduces to the CVRP. This also shows that the CluVRP is NP-hard. For $m = 1$, the resulting problem is the clustered traveling salesman problem (Chisman, 1975).

3.2. Instances

The instances that will be solved, are based on the CVRP instance sets of:

1) Golden et al. 1998
2) Li et al. 2005
3) VeRoLog Members VRP 2016

These were retrieved from http://www.vrp-rep.org/variants/item/cvrp.html, and their format is VRP-REP compliant. After extracting the data from the xml files and arranging it to a dictionary, the CVRP instance will be converted to a CluVRP instance.

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\

```
1  $number\ of\ clusters \leftarrow N$
2  $capacity\ is\ violated \leftarrow True$
3  while $capacity\ is\ violated$:
4      $cluster\ the\ nodes\ with\ KMeans\ (n\_clusters = number\ of\ clusters)$
5      $calculate\ each\ cluster's\ demand$
6      if $cluster\ demand\_i < Q\ for\ i \in \{1, 2, \ldots, N\}$:
7          $capacity\ is\ violated \leftarrow False$
8      else:
9          $number\ of\ clusters \leftarrow number\ of\ clusters + 1$
10     end
11 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

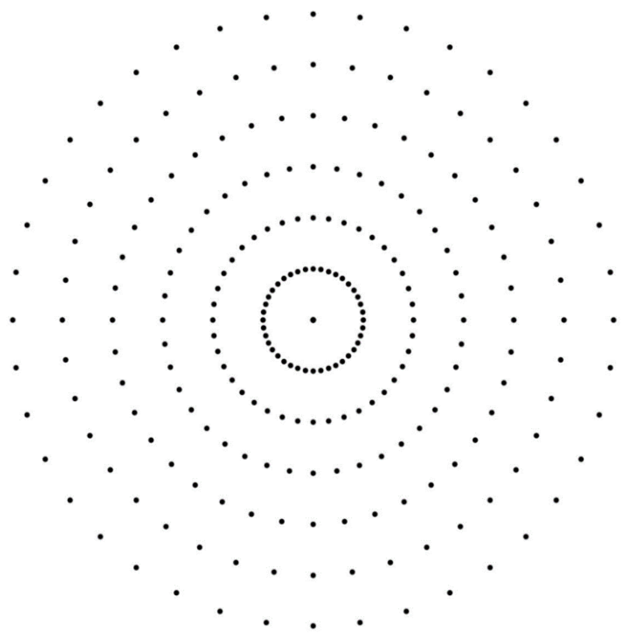
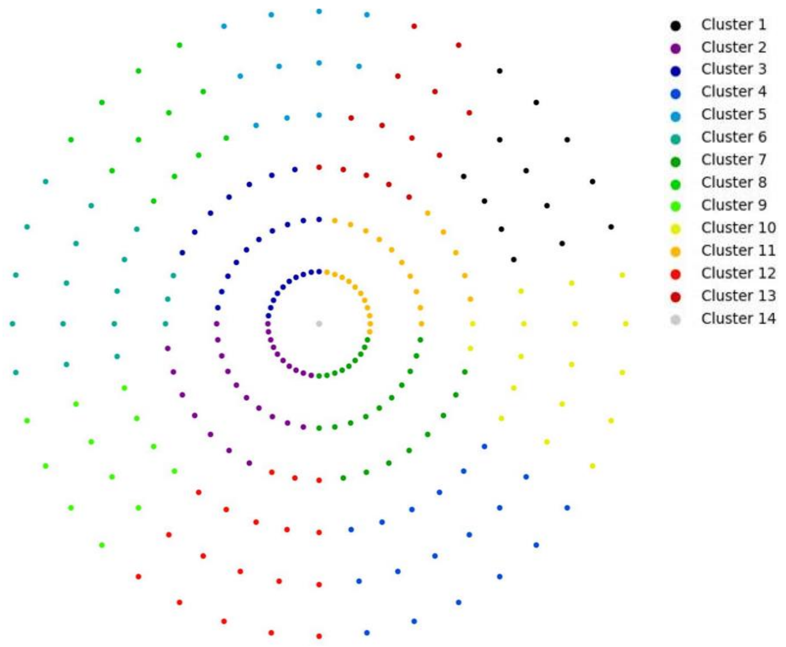


Figure 2 - CluVRP instance



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.

In 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 that result from merging two clusters into a cluster-route will be considered. The savings are calculated by subtracting the distance between the two customers from the sum of the distances of the two individual cluster-routes. The algorithm then repeatedly selects the highest savings routes and combines them until all clusters have been visited. It is a greedy approach, which means 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- or closed-type TSP) is solved by using the Nearest Neighbor (NN) algorithm. Here is where the previously mentioned "pseudo-soft" clustering (hereon referred to as simply "soft") comes into place.

In the case of a "hard clustered" problem, 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 to the depot, then serves all the customers of the closest node's cluster according to the NN algorithm, and then serves the rest of the clusters in a similar fashion (Algorithm 2). 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).

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. 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**

```
1   $\boldsymbol{for}$ $cluster\ route\ in\ cluster\ routes$:
2       $node\ route \leftarrow [\,]$
3       $\boldsymbol{if}$ $hard\ clustered$:
4           $first\ cluster \leftarrow cluster\ of\ nearest\ node$ to the depot
5           $first\ node\ list \leftarrow nodes\ of\ the\ first\ cluster$
6           $node\ route \leftarrow nearest\ neighbor\ (first\ node\ list, node\ route)$
7           $\boldsymbol{for}$ $cluster\ in\ clusters$:
8               $\boldsymbol{if}$ $cluster \neq first\ cluster$
9                   $node\ list \leftarrow nodes\ of\ cluster$
10                  $node\ route \leftarrow nearest\ neighbor\ (node\ list, node\ route)$
11              $\boldsymbol{end}$
12          $\boldsymbol{end}$
13      $\boldsymbol{else\ if}$ $soft\ clustered$:
14          $node\ list \leftarrow all\ nodes\ in\ cluster\ route$
15          $node\ route \leftarrow nearest\ neighbor\ (node\ list)$
16      $\boldsymbol{end}$
17  $\boldsymbol{end}$
```

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

```
1   For route in initial solution:
2       if hard clustered:
3           subroutes <- create subroutes (route)
4           for subroute in subroutes
5               optimised subroute <- VND(subroute)
6           end
7           optimised route <- combined optimised subroutes
8       else if soft clustered:
9           optimised route <- VND(route)
10      end
11  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 algorithm.
2) A Variable Neighborhood Descent algorithm using multiple restarts.
3) A General Variable Neighborhood Search algorithm

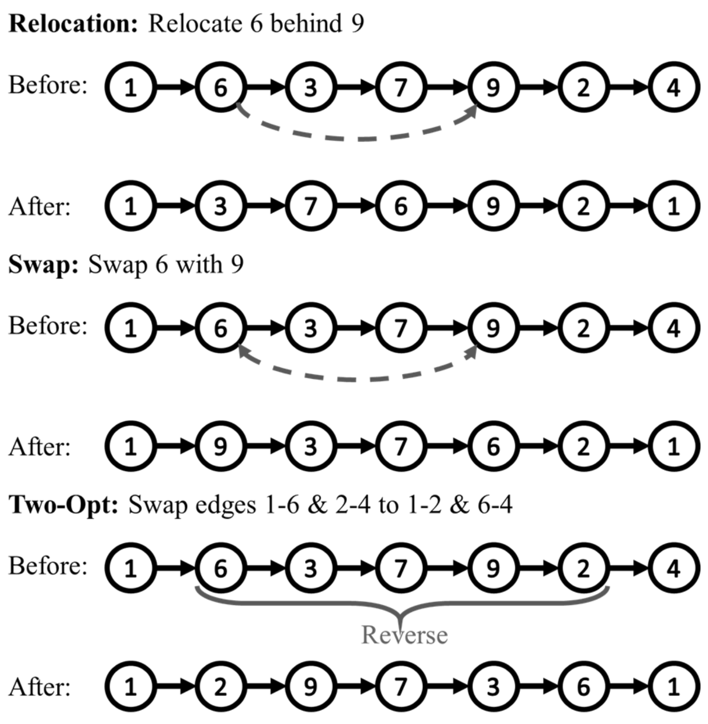
The Variable Neighborhood Descent (VND) algorithm is a deterministic variation of the Variable Neighborhood Search (VNS) algorithm (Mladenović & Hansen, 1997) where the solution space is explored in a non-stochastic way by changing the way of moving in the solution space when a local optimum is met. The algorithm terminates when the solution cannot be improved by any of the move-types (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**

```
1   best route <- route
2   termination condition <- False
3   move type <- 0
4   failed to improve <- 0
5   while termination condition is False:
6       if move type = 0
7           find best relocation move in best route
8           if move cost < −0.00001:
9               apply relocation move to best route
10              failed to improve <- 0
11          else:
12              move type <- move type + 1
13              failed to improve <- failed to improve + 1
```

```
            $\textbf{end}$
        $\textbf{else if}$ $move\ type\ =\ 1$
            $find\ best\ swap\ move\ in\ best\ route$
            $\textbf{if}$ $move\ cost\ <\ -0.00001$:
                $apply\ swap\ move\ to\ best\ route$
                $failed\ to\ improve\ <-0$
            $\textbf{else}$:
                $move\ type\ <-move\ type\ +\ 1$
                $failed\ to\ improve\ <-failed\ to\ improve\ +1$
            $\textbf{end}$
        $\textbf{else if}$ $move\ type\ =\ 2$
            $\textbf{find}$ $best\ two\ opt\ move\ in\ best\ route$
            $\textbf{if}$ $move\ cost\ <\ -0.00001$:
                $apply\ two\ opt\ move\ to\ best\ route$
                $failed\ to\ improve\ <-0$
            $\textbf{else}$:
                $move\ type\ <-move\ type\ +\ 1$
                $failed\ to\ improve\ <-failed\ to\ improve\ +1$
            $\textbf{end}$
        $\textbf{end}$
        $\textbf{if}$ $failed\ to\ improve\ =\ 3$:
            $termination\ condition\ <-True$
        $\textbf{end}$
    $\textbf{end}$
```

Figure 3 - Move types



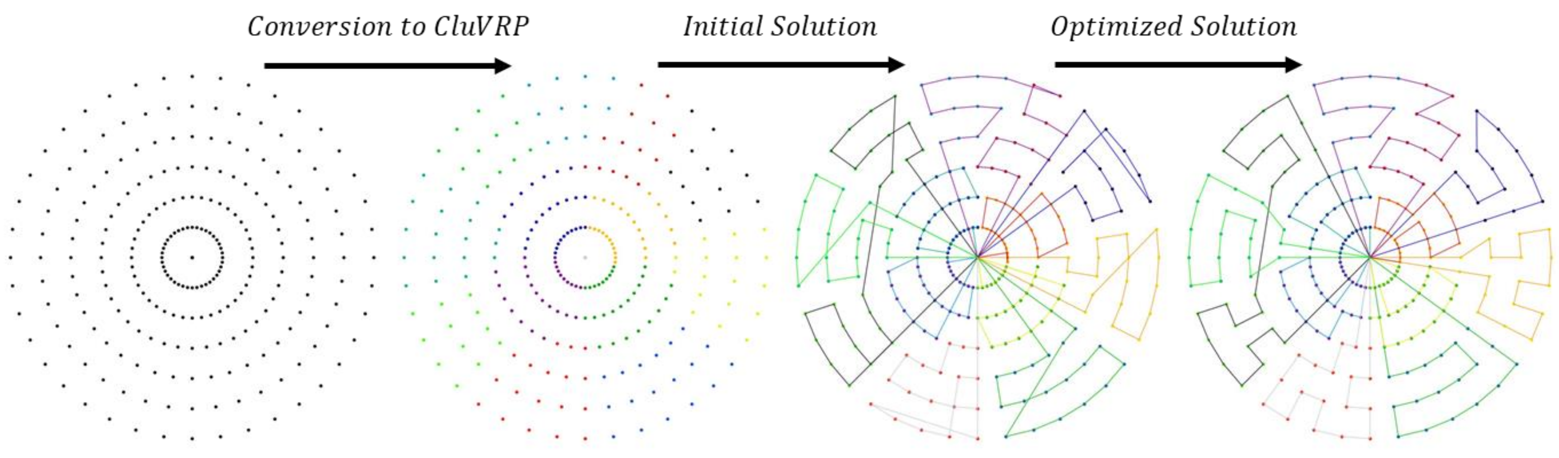
The second technique implements a stochastic element in the procedure to further search the solution space. This is achieved by applying the VND method to multiple initial solutions constructed by using the NN algorithm with a Restricted Candidate List (RCL) (Feo & Resende, 1995). The RCL is a list of 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, a RCL with a length of 3 is used (Algorithm 5).

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

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

Finally, the third method is a more powerful version of the VNS, the General Variable Neighborhood Search (GVNS) (Hansen, Mladenović, & Moreno Pérez, 2010). In this expansion of VNS, the iterative improvement 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**

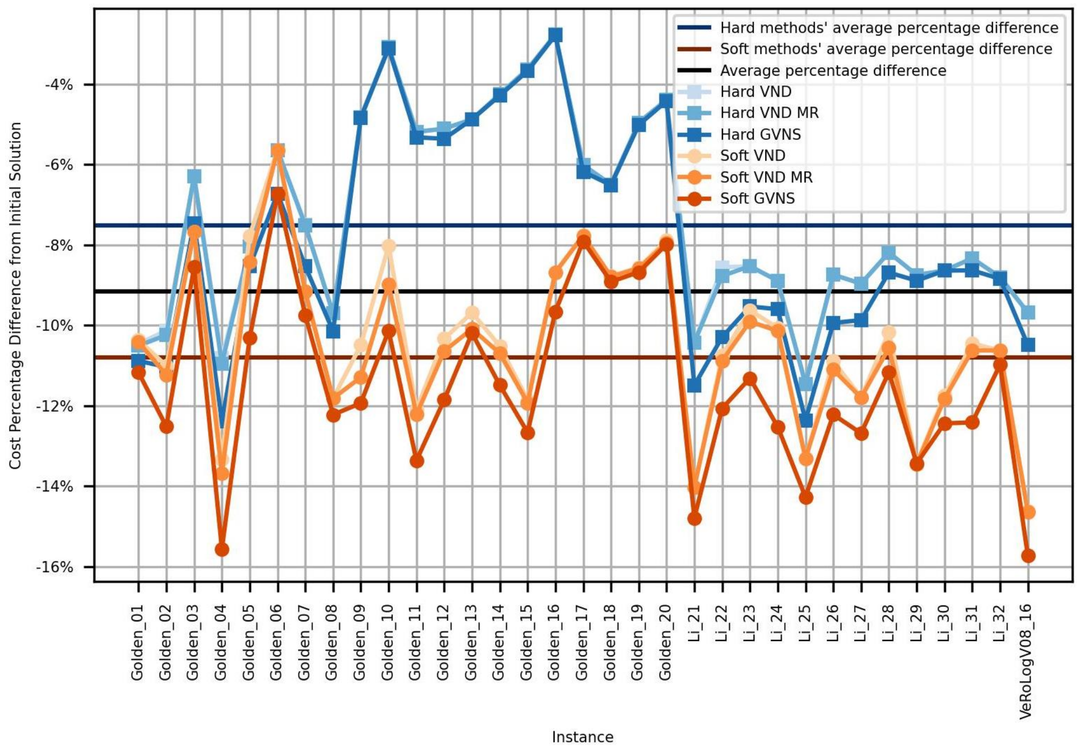
```
1   $CPU\ time\ <-60$
2   $elapsed\ time\ <-0$
3   $start\ <-current\ time$
4   $best\ solution\ <-solve\ the\ instance\ with\ nearest\ neighbor\ and\ VND\ (no\ RCL)$
5   $\boldsymbol{while}\ elapsed\ time\ <\ CPU\ time$
6       $current\ solution\ <-randomly\ select\ neighboring\ solution$
7       $optimised\ current\ solution\ <-optimise\ current\ solution\ with\ VND$
8       $\boldsymbol{if}\ cost\_(optimised\ current\ solution)\ <\ cost\_(best\ solution):$
9           $best\ solution\ <-optimised\ current\ solution$
10      $\boldsymbol{end}$
11      $end\ <-current\ time$
12      $elapsed\ time\ <-end-start$
13  $\boldsymbol{end}$
```

4. Results

4.1. Analysis

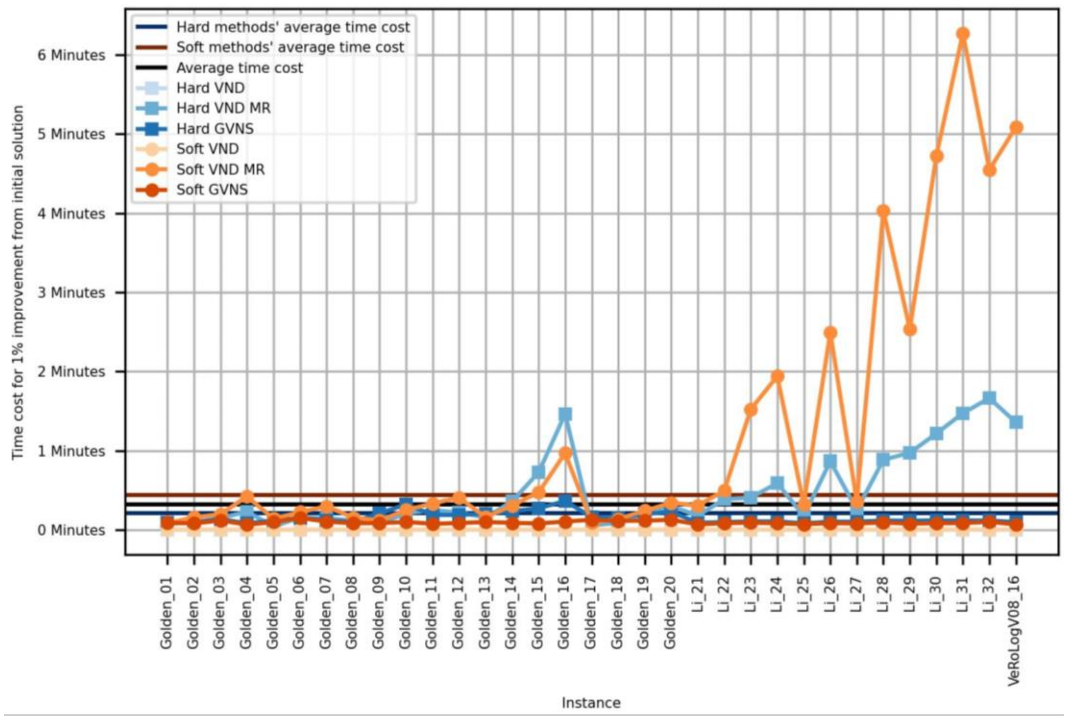
Moving on to the results for each instance, a detailed table can be found in the appendix for every instance set. An initial idea about the performance of the methods can be found in **Figure 5** below, where the cost percentage difference from the initial solution (1. Clarke & Wright, 2. Nearest Neighbor) 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%, and it is noticeable that in nearly all instances, all soft techniques produce better results than the hard ones, especially in the Golden 09-16 instances. The best overall performing method seems to be that of the GVNS applied on soft clustered problems.

Figure 5 - Cost Percentage Difference from Initial Solution per Instance



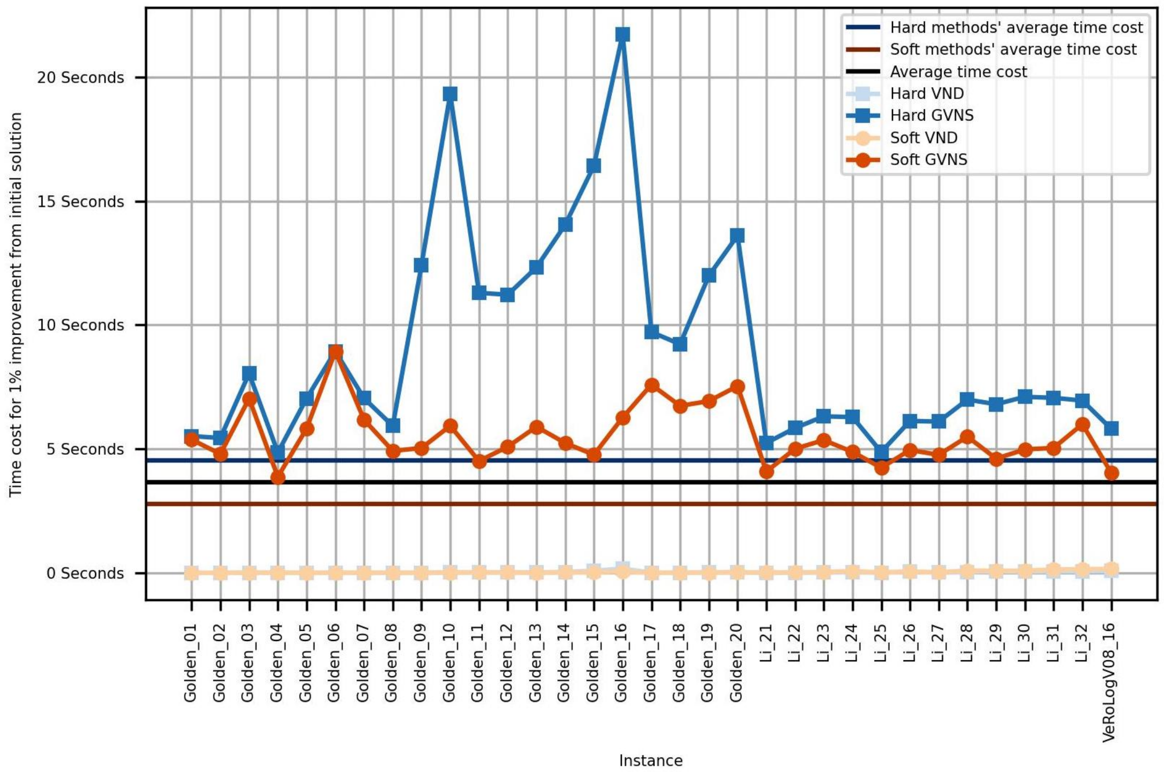
To get a more complete image about the performance of each optimization technique, the dimension of computational time is incorporated into the results, by calculating the time cost for every 1% improvement from the cost of the initial solution (**Figure 6**).

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



After a first look, it seems 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**). It seems that, on average, 4 seconds are required for a 1% improvement. Moreover, soft clustered instances require on average 2 less seconds than hard ones to improve the initial solution by 1%. Interestingly, the simple VND methods both 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. Continuing, the GVNS, performed on hard clustered problems, is not so efficient, especially in Golden instances 09-16. Finally, GVNS, performed on soft clustered problems, maintains the same level of efficiency in all the instances.

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 once again skew the results. This happens because both 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 small extra difference in time needed to complete one iteration accumulates over the minimum 500 iterations, making the multiple restarts method terribly time consuming.

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

Figure 8 – Computational time for all methods

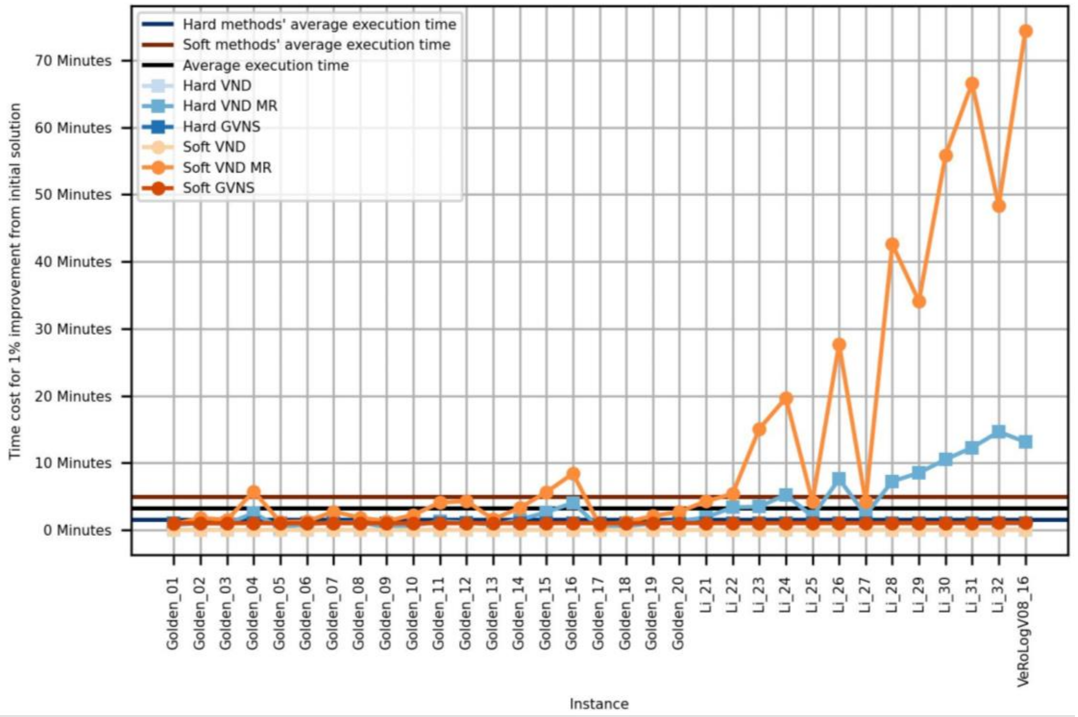
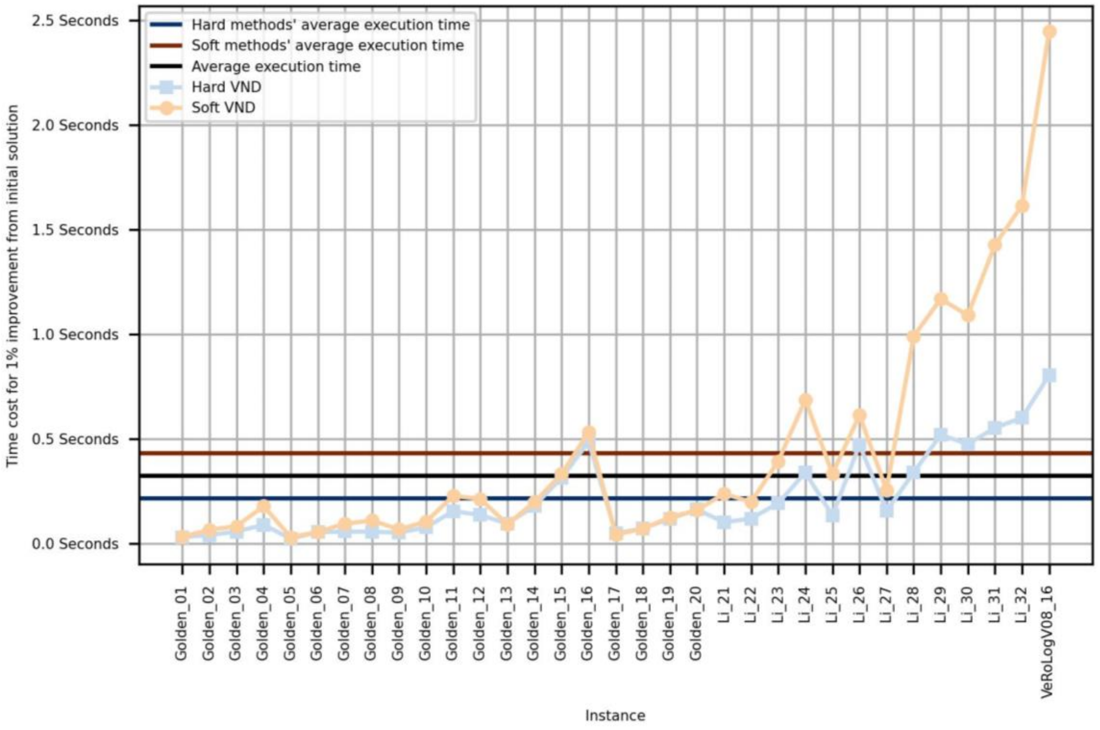


Figure 9 – Computational time for VND methods



For **Figure 8** above, interesting is the fact that in some cases the Soft VND method takes significantly more time to complete, in some cases 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 the existence of hard constraints on intra-cluster routing severely limits 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 time window of 60s, as evident by **Figure 4** and **Figure 6**.

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 note 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 can be serviced by a vehicle (Vehicle capacity / Average node demand). The results can be seen in **Figure 11**.

Figure 10 - Computational time relation with various instance metrics

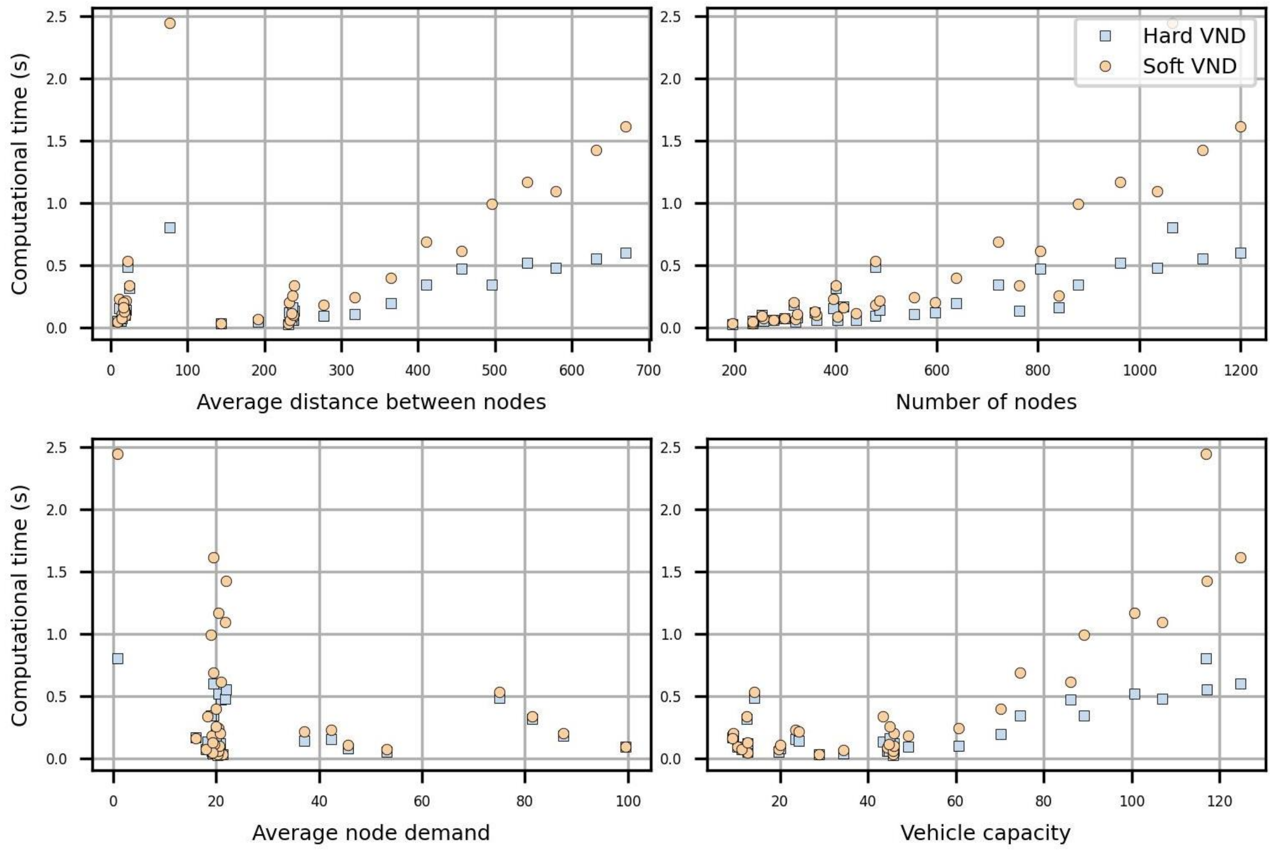
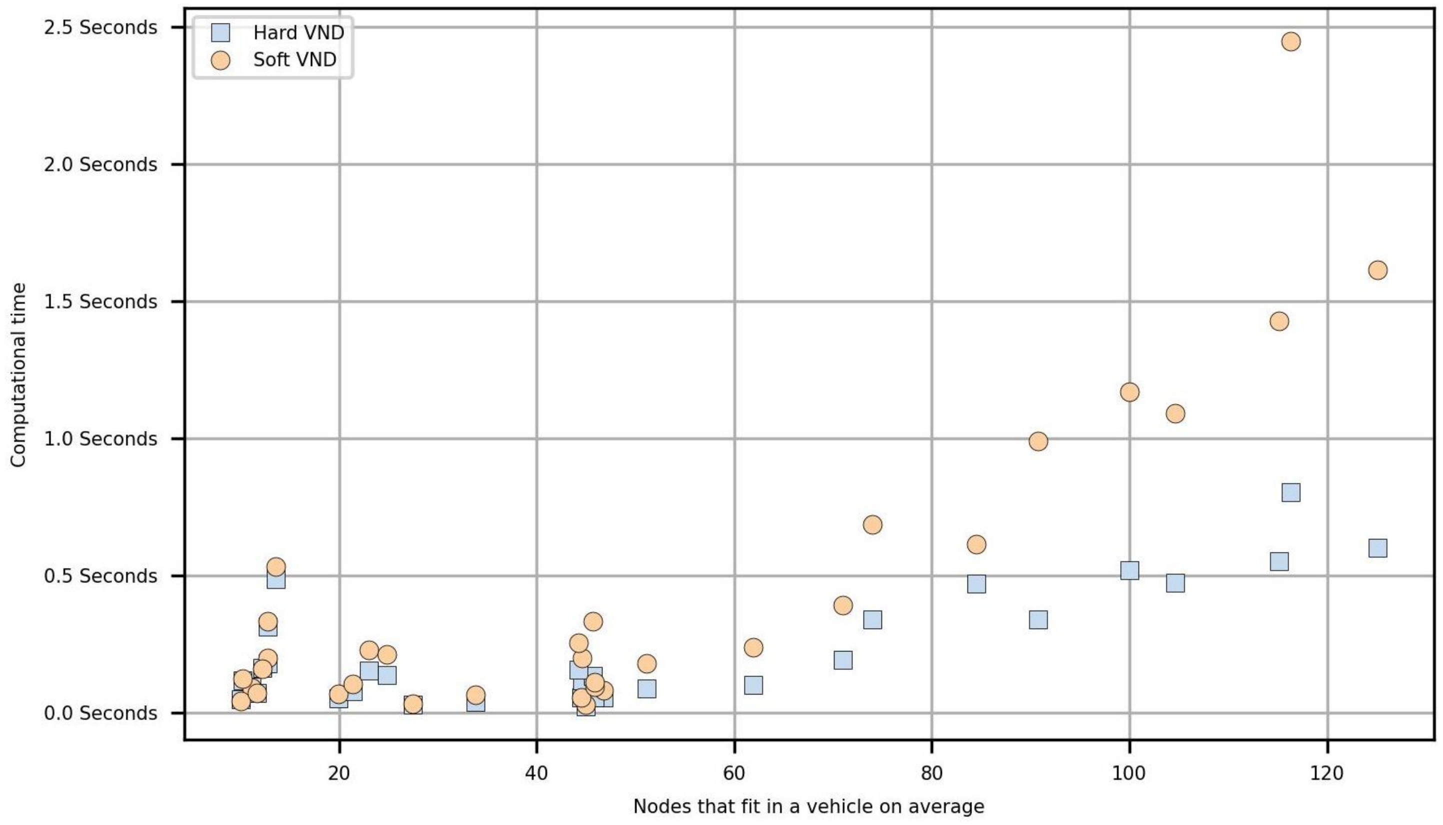


Figure 11 – Relation of computational time with average number of nodes that fit in vehicle



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

4.2. Hard vs Soft solutions visualized

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

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

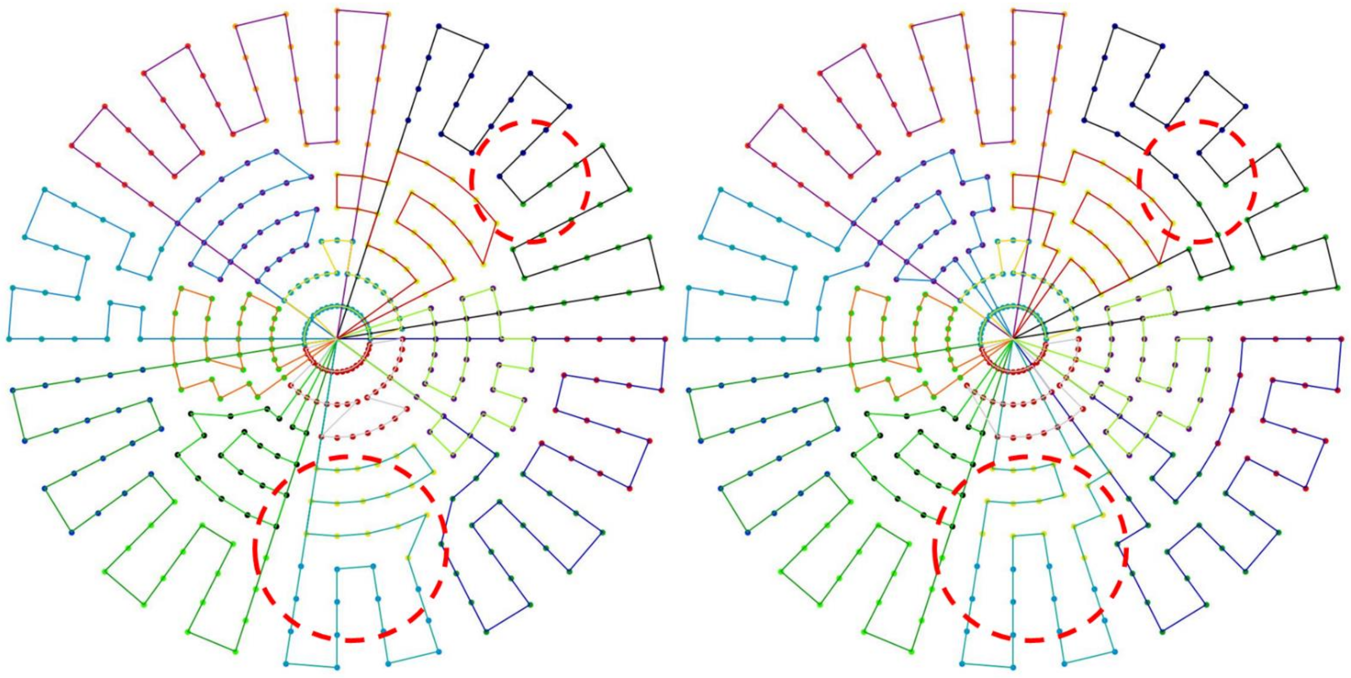


Figure 12 - Li 22: Hard GVNS with cost = 15,791 | Soft GVNS with cost = 15,479

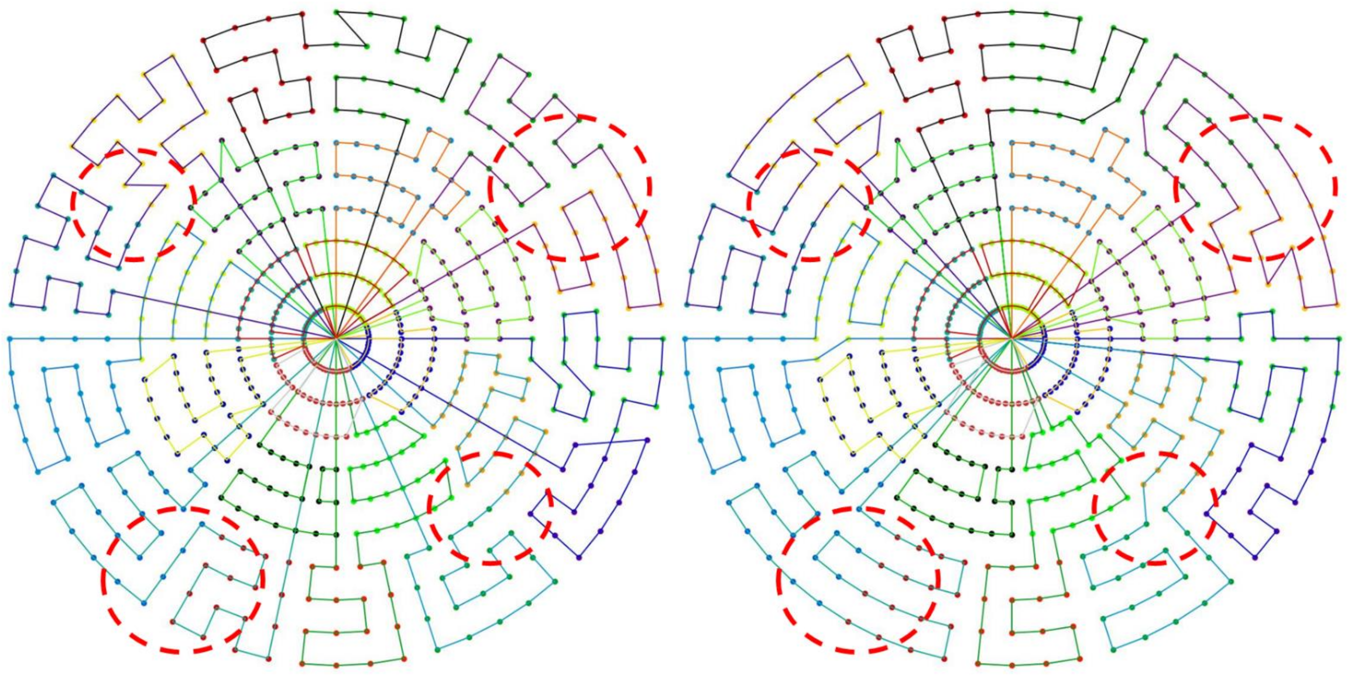
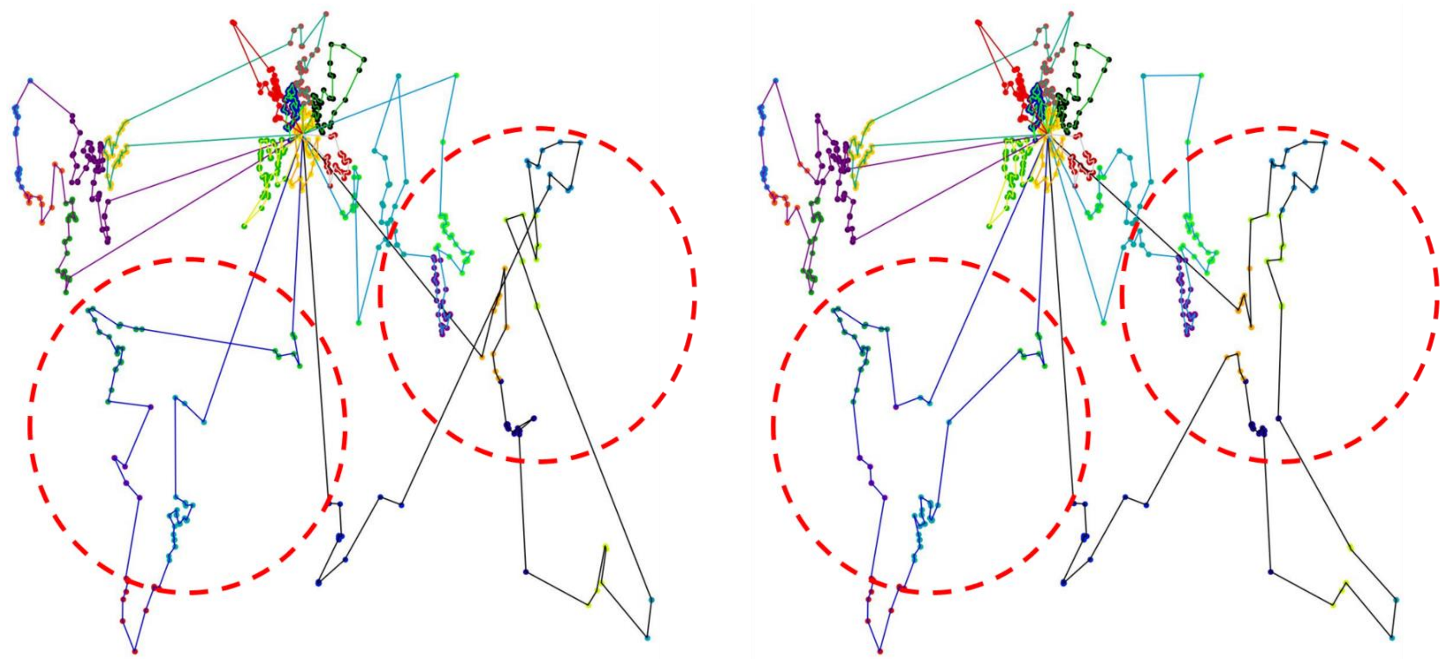


Figure 13 - VeRoLogV08_16: Hard GVNS cost = 2593 | Soft GVNS cost = 2442



In **Figure 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 as well as the way they are serviced afterwards.

5.2. The implementation

In this implementation, the problem is divided into two routing problems: the high-level routing problem of visiting the customers' 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 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 results, it was clear that the GVNS approach was the most cost-effective method, while the simple VND approach performed admirably well for the computational time it needs to execute. Moreover, in some instances, the VND with multiple restarts method was prohibitively computationally demanding. To further examine the results, an analysis was made regarding the relation of each method's performance to the different aspects 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 approximately the same amount of time.

5.4. Method shortcomings

This implementation is hindered by three known problems:

1) The way the clusters are made is not the most 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. An intermediate step of using a metaheuristic algorithm to optimize the order of visiting the clusters before moving on to improve the initial solution could further improve the solutions.
3) The use of a distance matrix possibly prohibits the utilization of the algorithm on large instances

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* (Τόμ. 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: Algorithms, 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.

Appendix

Code Files, Instances and Results Table

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

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.

here