# HYBRID ALGORITHM OF GUIDED LOCAL SEARCH FOR THE CAPACITATED VEHICLE ROUTING PROBLEM

**Sidnei Gouveia Junior**
Programa de Pós-Graduação em Modelagem Computacional e Sistemas,
Universidade Estadual de Montes Claros (Unimontes)
Campus Universitário Prof. Darcy Ribeiro, Av. Prof. Rui Braga, s/n
Vila Mauriceia, Montes Claros - MG, Brasil

Technical Report, PPGMCS/Unimontes – September, 2021

**Abstract.** This work proposes a hybridization of the Guided Local Search (GLS) proposed by Davenport et al. [1994], for the classic variation of the vehicle routing problem, one of the problems but more studied in the area of combinatorial optimization, the Capacitated Vehicle Routing Problem (CVRP). The proposed GLS is a hybrid version, as it incorporates, adapts and combines strategies borrowed from the Hybrid Iterated Local Search (HILS) of Subramanian et al. [2013] and the Knowledge Guided Local Search (KGLS) of Arnold and Sorensen [2019]. The efficiency of the proposed version is confirmed, using a paradigm presented in the work by Vidal [2020], allowing a comparative analysis of the version of the GLS proposed in comparison of the metaheuristics that reached the state of the art in solving the CVRP. At end, only when generating the results of computational experiments and conclusions.

**Keywords.** Vehicle Routing Problem, Metaheuristics, Combinatorial Optimization.

# 1 Introduction

The Vehicle Routing Problem (VRP) was first introduced by Dantzig and Ramser [1959] in the late 1950's to solve a gasoline distribution problem. According to Uchoa et al. [2017], its relevance stems from its direct application in real-world transportation systems that distribute goods and provide services, vital to modern economies. Reflecting the wide variety of conditions present in these transport systems, the literature stemming from the original VRP is spread out in dozens of variants. There are, for example, variants that consider time windows, multiple depots, homogeneous and heterogeneous vehicle fleets, complex loading restrictions, and so on.

Among these variants of the VRP, the Capcitated Vehicle Routing Problem (CVRP), deals with the most basic variant of the problem, and plays a crucial role in algorithmic research, both for exact and heuristic approach methods. Naturally, this variation presents itself as the most favorable test environment for experimenting with new ideas, due to the simplicity of CVRP in relation to VRP variants with more complex attributes. Basically the problem consists of a set of points $n + 1$, these being a deposit and $n$ customers; an array of distances $d$ equivalent to $(n + 1) \times (n + 1)$ points, with $d = [d_{ij}]$ being the distances between the points $i$ and $j$; an n-dimensional demand vector $q = [q_i]$ representing the quantity to be delivered to the customer $i$; and a maximum load capacity $Q$. Its resolution consists of finding a set of routes, starting and ending in the depot, each customer must be visited only once for a single route, so that the sum of customer demand for each route does not exceed the $Q$ capacity of the vehicles. The objective of the problem is to find a solution with the shortest possible total distance to travel the routes, respecting the restrictions of the problem.

Basically, in the literature there are two main classes of methods proposed for solving the CVRP, which were classified by Cordeau et al. [2007], being the classic heuristics, developed mainly between the 1960's and 1990's, and the metaheuristics developed from the decade of 2000 onwards. In the case of metaheuristics, there are those that were proposed through the hybridization and unification of the methods listed by Laporte et al. [2014], and that combine and consolidate different concepts borrowed from various methods, such as local search algorithms, exploration of large neighborhoods, intelligence of swarm, evolutionary algorithms, natural computing, integer programming, constraint programming, search trees, data mining, and parallel computing. Among the two main classes of methods listed above, metaheuristics have become widespread in recent algorithmic research for the resolution of CVRP. From the 2010s onwards, metaheuristic implementations (especially hybrids) emerged, which established themselves as the state of the art for CVRP resolution. That when tested and compared to each other by Vidal [2020] following the same computational test paradigm, they obtained very similar performance results. Also in Vidal [2020], it was observed that the exploration of new metaheuristics that differ from those presented in that work would not present themselves as a promising solution

for improving the best results found so far in the literature on CVRP, the author still argues that improvements in the quality of solutions are due to more focused and efficient local search methods.

In this work, we chose to explore a hybridization of a local search algorithm, initially presented in Davenport et al. [1994], and known as Guided Local Search (GLS), analyzing its efficiency in comparison with the main metaheuristics for CVRP in the literature. GLS basically consists of penalizing the objective function of the problem, through the inclusion of penalties in its resources, with the purpose of redirecting the search to promising regions of the candidate solution space, and escaping from being confined to local optimal solutions.

## 2 Theoretical Review

According to Toth and Vigo [2002], the VRP emerged as a generalization of the Traveling Salesman Problem (TSP), where the objective of the TSP is to minimize the distance of a single route by a given number of points (customers). By presenting itself with greater direct applicability in real-world transport systems, the VRP ended up standing out over the TSP, spreading more widely than its predecessor, without failing to bring with it the evolution of many of the techniques used for its resolution of the TSP. Due to its greater applicability in the real world, many of the authors who worked on methods for solving the VRP applied their algorithms in variations of the VRP applied to real world problems, which were described by these authors. As an example, the work of Kramer et al. [2016], which uses an Iterated Local Search (ILS) algorithm to solve a real distribution problem in a beverage industry.

Despite its indisputable importance, very specific generalizations of the problem tend to be little explored by theoretical production on VRP. Due to their nuances, complexities and resolutions already obtained, these generalizations are not very attractive to the improvement of new techniques and algorithmic research for the VRP. A solution found by the authors who dedicated themselves to the theme was the development of instances of comparison that contemplated a wide range of situations observed in previous works and offered a real difficulty to the methods that proposed to solve the VRP.

The first works that proposed the use of a set of instances to compare methods date from the 1960s and 1970s, such as the first set proposed in Christofides and Eilon [1969]. Since then, many other sets have emerged (such as those proposed in Augerat et al. [1995], Augerat et al. [1998] and Fisher [1994]) and these ended up becoming very easy for the methods developed until the 2010s. The most relevant and widespread set of instances in recent literature was presented by a group of researchers in Uchoa et al. [2017]. The authors proposed one hundred instances for the CVRP, known as X instances. With very varied characteristics, such as; different warehouse locations, different demand variations, different load capacities and a large number of customers compared to

the previous ones, ranging from 100 to 1000 customers. When dealing with an NP-Hard problem, that is, the difficulty to find the optimal solution grows exponentially as the number of clients increases. Finding optimal solutions can be a computationally impractical task depending on the size of the problem. So, as a metric for evaluating the quality of the solutions found, commonly in the literature these results obtained are expressed as Gap $= (S - S_{BKS})/S_{BKS}$, where $S$ is the value of the solution found and $S_{BKS}$ is the value of the Best Known Solution (BKS) for the different instances, the value of these solutions being listed on the CVRPLIB website at `http://vrp.atd-lab.inf.puc-rio.br/index.php/en/`. This repository being maintained by the researchers who developed the set X.

By convention, set X researchers have adopted some standards. As an example, the value (Euclidean distance) of the solutions found must be presented with a rounding to the nearest whole number. This pattern, according to its developers, even allows the algorithms to be faster, but it has other disadvantages, especially in relation to neighborhood structures. Another pattern of the set is the non-adoption of a pre-specified number of routes, as happens in set A, proposed in Augerat et al. [1995]. Also, by convention, the name given to each instance follows a pattern commonly adopted in the literature, where; X is a reference to the set, the $n$ followed by a number refers to the amount of customers plus the deposit $(n + 1)$ and the $k$ followed by a number represents the minimum amount of possible routes for that instance, (which does not necessarily correspond to the number of routes of the optimal solution). In Figure **??**, the smallest instance of set X is observed.

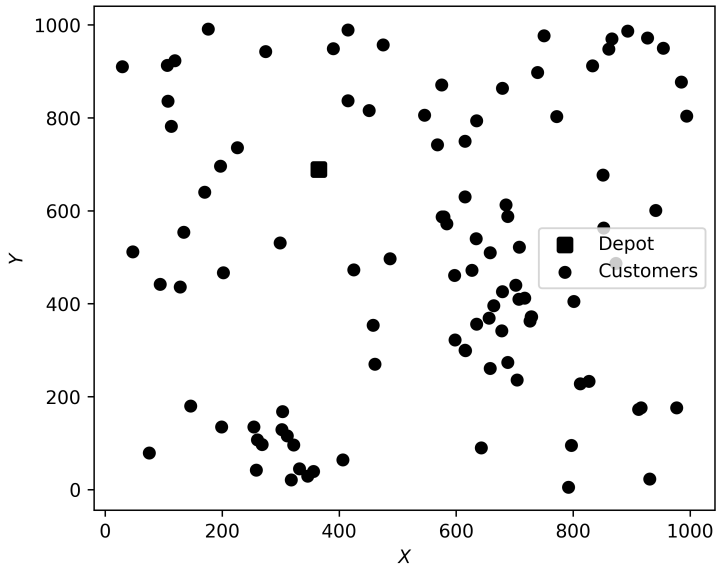


Figure 1: X-n101-k25

## 2.1 Route-First Cluster-Second Heuristic

Among the heuristics, a very common strategy, although not the only one, which can be observed among the main metaheuristics for CVRP, was initially presented in Beasley [1983], and named "Route-First Cluster-Second". This strategy is often cited in the literature (see, Laporte and Semet [2002], Cordeau et al. [2007], Prins et al. [2009], Prins et al. [2014] and Vidal [2016]) and is commonly used in the generation of initial solutions and also during the search, as the name suggests, it divides the problem solving into two phases, they are:

**Route-First;** at this stage the vehicle capacity is temporarily relaxed so that a route (as in TSP) covering all customers is computed, often called a giant tour. A common strategy for its creation (especially in the initial solution) is the connection to the nearest unserved point.

**Cluster-Second;** at this stage the giant tour is broken down into viable routes. A simple strategy is to use successful heuristics for packaging algorithms like those used in Arnold and Sorensen [2019], or more sophisticated heuristics based on dynamic programming as used by Vidal et al. [2012].

Figure 2.1 below demonstrates these phases.

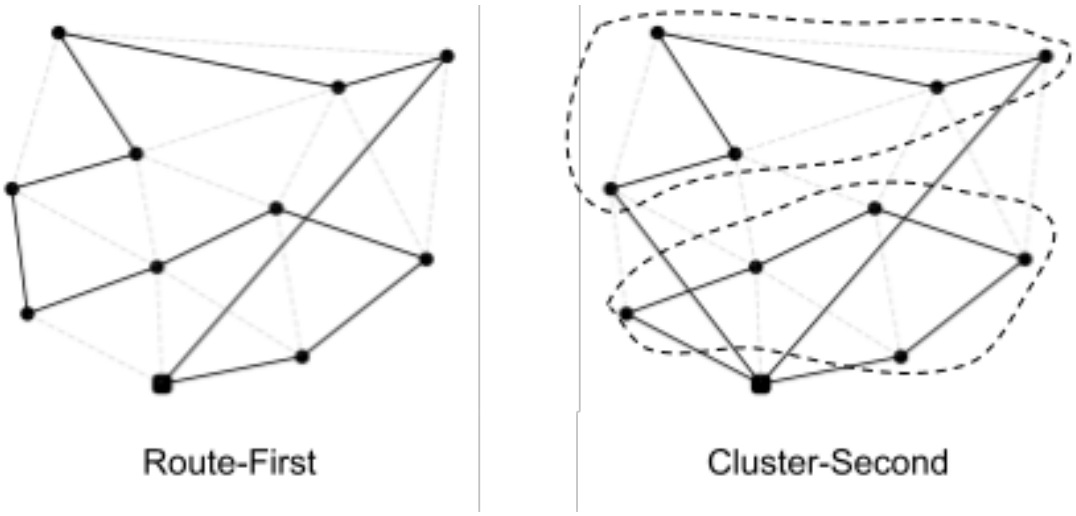


Figure 2: Route-First Cluster-Second Heuristic

# 3 Mathematical Modeling

Usually two different types of formulations are used for modeling the problem, the vehicle flow and the set partitioning, differing from each other by their variables and constraints. And they allow some instances to be solved exactly, by programs that use exact methods (known as solvers). A metaheuristic that uses a solver to improve the solutions found is the Hybrid Iterated Local Search (HILS) of Subramanian et al. [2013], using a set partitioning formulation, which inspired this version of GLS.

Considering a set of customers represented by $\mathcal{C} = \{1, \ldots, n\}$, so that a positive demand is associated with each customer $i \in \mathcal{C}$. Each route must start at the depot, visit a subset of customers, and then return to the depot so that the sum of the demands of the subset of customers visited does not exceed the maximum capacity of the vehicle. Furthermore, all customers must be visited exactly once.

## 3.1 Set Partitioning Formulation

The problem representation is done using a graph $G(\mathcal{N}, \varepsilon)$, where $\mathcal{N} = C \cup \{0, n+1\}$ is the set of nodes associated with customers in $C$ and nodes 0 and $n+1$ are the deposit, imposing that all routes must start at 0 and return to $n+1$. The set $\varepsilon$ contains the arcs $(i, j)$ for each pair of nodes $i, j \in \mathcal{N}$. The cost of the arc $(i, j)$ is denoted by $c_{ij}$. Each customer has a demand $q_i$ so that $q_i > 0$ for each $i \in C$ and $q_O = q_{n+1} = 0$. The objective of the problem is to determine a set of least cost routes that meet all the requirements defined above.

In the classic formulation of set partitioning presented in Munari et al. [2017], the variables correspond to viable routes of the problem. Let $\mathcal{R}$ be the set of routes that satisfy the problem's requirements. With $\lambda_r$ being the binary decision variable which is equal to 1 if and only if the route $r \in \mathcal{R}$ is selected. The classic formulation of set partitioning is as follows:

$$\min \quad \sum_{r \in \mathcal{R}} c_r \lambda_r \tag{1}$$

$$\text{sujeito a:} \quad \sum_{r \in \mathcal{R}} a_{ri} \lambda_r = 1, \qquad i \in \mathcal{C} \tag{2}$$

$$\lambda_r \in \{0, 1\}, \qquad r \in \mathcal{R} \tag{3}$$

The Objective Function 1 imposes that the total travel cost of the routes is minimized. The cost of the route $r \in \mathcal{R}$, denoted by $c_r$, is calculated using the cost arcs $c_{ij}$ defined above. Given a $r$ route that sequentially visits customers $i_0, i_1, \ldots, i_p, p > 0$, its total cost is given by:

$$c_r = \sum_{j=0}^{p-1} c_{i_j, i_{j+1}} \tag{4}$$

Constraint 2 ensures that all clients are visited exactly once. Each column $a_r = (a_{r1}, \ldots, a_{rn})^T$ is a binary vector where $a_{ri} = 1$ if and only if the corresponding route $r$ visits the customer $i$. Munari et al. [2017] also clarifies that generating all $\mathcal{R}$ routes is impractical in general, as the number of routes is exponential in relation to the number of clients in the problem. Therefore, set partitioning formulations require the use of the column generation technique to solve the linear relaxation of the model (1) and (3). In the algorithm proposed in this work, we used a generalization of the set partitioning formulation presented in this section, provided by the LocalSolver program, at the electronic address `https://www.localsolver.com/docs/last/exampletour/vrp.html`.

# 4 Guided Local Search

The Guided Local Search (GLS) proposed by Davenport et al. [1994], is an established algorithm used in the resolution of CVRP, see Vidal [2020], which uses two distinct variations, a hybrid proposed in Arnold and Sorensen [2019], and a pure one implemented in Google's OR-Tools open source program developed by Perron and Furnon [2019]. The GLS consists of including a set of penalty terms in the objective function of any iterated search, so the terms confine the search to promising regions, and whenever a local minimum is reached, the terms are modified and the search restarts. Given the objective function $g$ that maps the suitability of the candidate solution $S_k$, GLS defines a new objective function 5, given by:

$$h(S_k) = g(S_k) + \lambda \times \sum_i (p_i \times I_i(S_k)) \tag{5}$$

Where $S_k$ is a candidate solution, and a GLS parameter $i$ varies on the resources of the problem, $p_i$ is a resource penalty $i$ (penalties are initialized to zero) and $I_i$ is an indication of whether $S_k$ displays the resource $i$.

$$I_i(S_k) = \begin{cases} 1, \text{ if } S_k \text{ displays the resource} \\ 0, \text{ otherwise} \end{cases} \tag{6}$$

The intention is to penalize the features that matter most when the search gets stuck. Therefore, the arcs are the resources that need to be penalized in CVRP. Another factor that must be considered is the current penalty in the value of that resource. Where the resource utility penalty $i$, under an optimal local solution $S_k$ is defined by Equation 7:

$$util_i = I_i(S_k) \times \frac{c_i}{1 + p_i} \tag{7}$$

where $c_i$ is the cost and $p_i$ is the current value of the resource penalty $i$. When a resource is penalized, the penalty amount is increased by 1 and the penalty scale is adjusted by $\lambda$.

In the 1 Algorithm, the pseudocode of the classical GLS implementation is represented.

**Algorithm 1:** Guided Local Search

**Input** : $g, \lambda, M, c$

$k \leftarrow 0$

$S_k \leftarrow$ InitialSolution ▷ `random or heuristic method`

**for** $i = 1 \rightarrow |M|$ **do**

  $p_i \leftarrow 0$

**while** the stop condition is not reached, **do**

  $h(S_k) \leftarrow g(S_k) + \lambda \times \sum_i (p_i \times I_i(S_k))$

  $S_{k+1} \leftarrow \text{LocalSearch}(S_k, h(S_k))$

  **for** $i = 1 \rightarrow |M|$ **do**

    $\text{util}_i \leftarrow I_i(S_{k+1}) \times c_i/(1 + p_i)$

  **for** $i = 1 \rightarrow |M|$ **do**

    $p_i \leftarrow p_i + 1$

  $k \leftarrow k + 1$

$S^* \leftarrow$ best solution found in relation to $g$ function

**Output :** $S^*$

## 5 Hybrid Guided Local Search

In this section, the hybrid version of the GLS developed in this work for the resolution of CVRP is detailed. The idea of this version is to incorporate and adapt very successful techniques found in the main metaheuristics described in the literature. For example:

- **The initial solution**, was adapted from the KGLS algorithm of Arnold and Sorensen [2019], using the strategy of route before and group after: routing through the connection to the nearest unserved client, and grouping routes based on in customer demand with a bundling algorithm known as *First-fit* from Blazewicz et al. [1989].
- **The local search**, used classic operators presented in Laporte and Semet [2002] and implemented in the GLS of Perron and Furnon [2019]. Using the strategy used in Arnold and Sorensen [2019] to change neighborhoods, combined with the strategy used in Subramanian et al. [2013] of using a solver based on a problem formulation.

For a better understanding of its operators and strategies, an entire subsection has been dedicated further on in order to explain how the local search of this version turned out. The Algorithm 2 presents the pseudocode of the implementation of this version of the GLS, allowing to understand its differences from the original version presented in the Algorithm 1.

**Algorithm 2:** Hybrid Guided Local Search

**Input :** $g, \lambda, M, c$

1 Model $\leftarrow$ create the model for CVRP
2 $k \leftarrow 0$
3 $S_k \leftarrow$ InitialSolution
4 $S_{k+1} \leftarrow$ LocalSearch$(S_k, g)$ ▷ `Best improvement`
5 **while** the stop condition is not reached, **do**
6 iter $\leftarrow 0$
7 **for** $i = 1 \rightarrow |M|$ **do**
8 $p_i \leftarrow 0$
9 **while** iter $\leq \text{Max}_{\text{iter}}$ **do**
10 $h(S_k) \leftarrow g(S_k) + \lambda \times \sum_i (p_i \times I_i(S_k))$
11 $S_k \leftarrow$ BuscaLocal$(S_k, h)$ ▷ `First improvement`
12 **for** $i = 1 \rightarrow |M|$ **do**
13 $\text{util}_i \leftarrow I_i(S_{k+1}) \times c_i/(1 + p_i)$
14 **for** $i = 1 \rightarrow |M|$ **do**
15 $p_i \leftarrow p_i + 1$
16 iter $\leftarrow$ iter $+ 1$
17 $k \leftarrow k + 1$
18 $S_k \leftarrow$Solver$(S_k, g, \text{Max}_{\text{tempo}}$, Model$)$
19 $S^* \leftarrow$ best solution found in relation to $g$ function

**Output :** $S^*$

Where the Model variable is a generalization of the set partitioning formulation presented in Section 3 and provided by LocalSolver in , and the other parameters are:

- $\text{Max}_{\text{iter}}$, set to 30. This parameter represents the movements in the current solution $S_k$ (similar to KGLS of Arnold and Sorensen [2019]), for variation of the search's neighborhood.
- Solver, is the optimization program used to improve the current solution $S_k$. The solver used for this work is a program that offers an open license for academic use, LocalSolver *Software* [2020].
- $\text{Max}_{\text{time}}$, is the maximum seek time of the solver to improve the current solution $S_k$. And the stop condition in line 5 of Algorithm 2 must be reconciled, in order to avoid conflicts.

For the computational experiments (Section 6), which follow a paradigm proposed in the literature for method comparison and which stipulates a stop condition by search time of the algorithm based on the size of the instance to be solved, 5% of algorithm seek time.

### 5.1 Local Search

Local search can be understood as any heuristic or non-heuristic method used to find solutions to optimization problems, which by nature tend to be computationally difficult to solve, such as a CVRP, for example. Operators are resources used by the local search to move between different neighborhood structures in the candidate solution space, applying changes to the solution resources. There are two strategies presented in Hansen and Mladenović [2006] and used in this work following the implementation used in Perron and Furnon [2019], to define the criteria that interrupt the local search, they are:

- **First improvement**, where the local search will look for neighboring solutions until a single solution better than the current solution is found.
- **Best improvement**, where the local search will look for neighboring solutions while the ones found are better than the current best solution.

It may seem obvious, but it is worth noting that the local search needs a cost function to assess the quality of the solutions found, as in the 1 and 2 Algorithms. The pseudocode of the Algorithm 3 demonstrates a logical, basic and generic structure to represent the local search used in HGLS.

**Algorithm 3:** Generic Local Search Structure

```
Input  : f, S_k
while the stop condition is not reached do
    for each route r ∈ S_k do                          ▷ Intra-route Phase
        S_k ← 2-Opt(r, S_k, f)
    for each route r ∈ S_k do                          ▷ Inter-route Phase
        for each route r' ∈ S_k do
            if r ≠ r' then
                S_k ← Relocate(r, r', S_k, f)
                S_k ← Exchange(r, r', S_k, f)
                S_k ← Cross(r, r', S_k, f)
Output : S_k
```

In the Algorithm 3, $f$ represents the cost function of the candidate solution $S_k$. 2-Opt, Relocate, Exchange and Cross are the local search operators, which will be detailed below.

### 5.1.1 Local Search Operators

Local search operators are heuristics that move the current solution through different neighboring solutions in the search space of candidate solutions. These operators can be divided into two distinct categories, inter-route operators, which carry out movements by combining clients from two or more routes, and intra-route operators who carry out movements by changing the order of visit of clients on a route.

- **Inter-Route Operators**
  - **Relocate**, moves a client from one route to another route. Trying all possible combinations to move a customer to any valid position on the other route that produces an improvement in the solution.

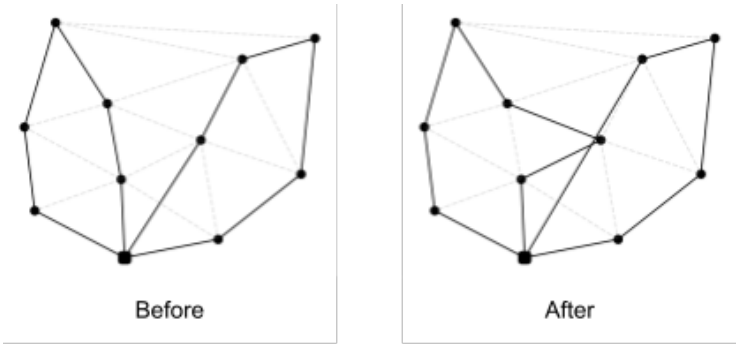


Figure 3: Relocate

  - **Exchange**, exchanges a client of one route with another client of another route. Trying all possible combinations of customer exchanges, only if that exchange produces an improvement.

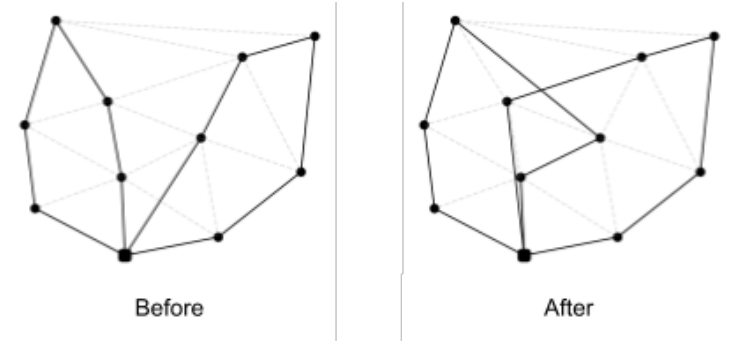


Figure 4: Exchange

– **Cross**, two different routes are combined by exchanging clients with each other.

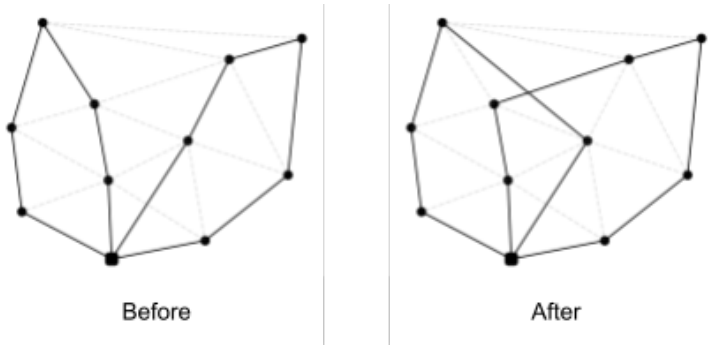


Figure 5: Cross

- **Intra-route operators**
  - **2-Opt**, removes two edges of the route and checks if the inversion of the end customers would result in an improvement in the current solution.

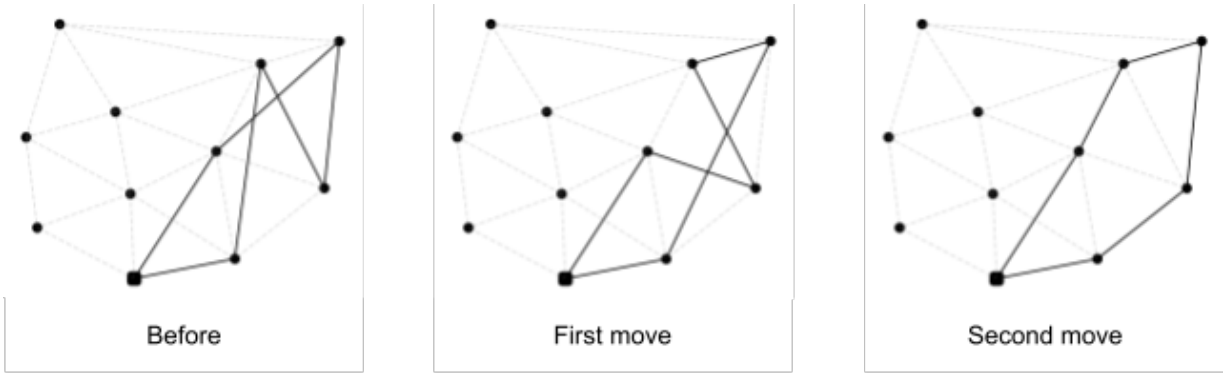


Figure 6: 2-Opt

## 6 Computer Experiments

A conduction of computational experiments that allowed the evaluation without bias of the HGLS in relation to the state of the art methods for CVRP, was to follow the paradigm used in Vidal [2020], based on good practices for testing optimization problems, with the experiments run in single thread and the same stopping condition for each instance, on a machine (server) with an Intel Gold 6148 Skylake 2.4 GHz processor with 40 GB of RAM, and a CentOS 7.8.2003 operating system. The metaheuristics evaluated were:

- Guided Local Search[1] (GLS), implemented by Perron and Furnon [2019].
- Hybrid Iterated Local Search[1] (HILS) by Subramanian et al. [2013].
- Knowledge Guided Local Search[2] (KGLS) by Arnold and Sorensen [2019].
- Slack Induction by String Removal[2] (SISR) by Christiaens and Vanden Berghe [2020].
- Hybrid Genetic Search (HGS)[1] by Vidal et al. [2012] and enhanced in Vidal [2020].

For the attempt to equate the computational environment, in the HGLS[3] language developed in this work, a machine (notebook) with an Intel Core i7-7500U 2.7 GHz processor with 8 GB was used of RAM, and a MS Windows 10.0.19041 operating system, with a single thread performance capacity of 96% (based on PassMark Software [2021]) compared to the machine above, using the same conditions as stop. The stop condition defined was the search time limit equal to $T_{max} = n \times 240/100$ seconds. Therefore, the smallest instance with 100 clients must resolve in 4 minutes, while the largest instance containing 1000 clients must resolve in 40 minutes.

To increase statistical significance, ten independent runs were performed for each metaheuristic. A curiosity is that both GLS, KGLS and HGLS are deterministic methods that depend on customer order. To get around this peculiarity, different random permutations of the clients were used in the ten executions of each instance. The value of the Gap[4] of each of the ten executions of each instance. All these parameters are exactly the same followed in Vidal [2020].

In Table 1 below are some compiled statistics of the Gap values of the ten runs of all one hundred instances of the set.

Table 1: Gap statistics of ten runs of all instances

| Statistics | **GLS** | **HGLS** | **HILS** | **KGLS** | **SISR** | **HGS** |
|---|---|---|---|---|---|---|
| Minimum | 0.38 | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 |
| Average | 4.01 | 1.57 | 0.66 | 0.53 | 0.19 | 0.11 |
| Maximum | 8.62 | 4.40 | 2.49 | 1.24 | 1.63 | 0.55 |
| Standard deviation | 1.87 | 0.91 | 0.58 | 0.30 | 0.19 | 0.13 |

In the next Tables (2, 3, 4 and 5) show the Comparison of the solution quality of each metaheuristic in each instance at $T_{max}$.

[1]Implemented in C++.

[2]Implemented in the Java language.

[3]Implemented in the Python 3.8.

[4]The value of the best known solution of each instance of the set was used as a general evaluation metric, as listed on the CVRPLIB website on November 1, 2020, being the same used as a reference in Vidal [2020] to calculate the Gap.

Table 2: Comparison of solution quality of each meta-heuristic in each instance at $T_{max}$

| Instance | GLS | | HGLS | | HILS | | KGLS | | SISR | | HGS | | BKS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Best | Avg | Best | Avg | Best | Avg | Best | Avg | Best | Avg | Best | Avg | |
| X-n101-k25 | 0.98 | 1.38 | 0.23 | 0.78 | 0.00 | 0.00 | 0.01 | 0.15 | 0.00 | 0.01 | 0.00 | 0.00 | 27591 |
| X-n106-k14 | 1.44 | 1.48 | 0.21 | 0.51 | 0.07 | 0.11 | 0.05 | 0.19 | 0.02 | 0.07 | 0.01 | 0.07 | 26362 |
| X-n110-k13 | 0.10 | 0.85 | 0.00 | 0.07 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 | 14971 |
| X-n115-k10 | 0.16 | 0.48 | 0.00 | 0.13 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 12747 |
| X-n120-k6 | 0.94 | 1.26 | 0.00 | 0.49 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 13332 |
| X-n125-k30 | 1.88 | 2.31 | 1.21 | 1.78 | 0.31 | 0.55 | 0.24 | 0.36 | 0.00 | 0.04 | 0.00 | 0.00 | 55539 |
| X-n129-k18 | 2.45 | 2.63 | 0.41 | 0.87 | 0.05 | 0.11 | 0.05 | 0.11 | 0.00 | 0.03 | 0.00 | 0.00 | 28940 |
| X-n134-k13 | 1.62 | 2.28 | 0.22 | 0.84 | 0.12 | 0.29 | 0.13 | 0.22 | 0.02 | 0.20 | 0.00 | 0.00 | 10916 |
| X-n139-k10 | 0.75 | 1.10 | 0.09 | 0.22 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 13590 |
| X-n143-k7 | 1.99 | 2.70 | 0.26 | 1.26 | 0.15 | 0.23 | 0.17 | 0.19 | 0.00 | 0.18 | 0.00 | 0.00 | 15700 |
| X-n148-k46 | 2.00 | 2.58 | 1.45 | 1.71 | 0.00 | 0.00 | 0.14 | 0.32 | 0.00 | 0.04 | 0.00 | 0.00 | 43448 |
| X-n153-k22 | 1.78 | 2.61 | 0.65 | 1.12 | 0.02 | 1.08 | 0.73 | 0.78 | 0.02 | 0.04 | 0.02 | 0.02 | 21220 |
| X-n157-k13 | 1.23 | 1.53 | 0.00 | 0.10 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.01 | 0.00 | 0.00 | 16876 |
| X-n162-k11 | 0.70 | 0.87 | 0.06 | 0.50 | 0.00 | 0.10 | 0.06 | 0.06 | 0.00 | 0.15 | 0.00 | 0.00 | 14138 |
| X-n167-k10 | 2.84 | 2.92 | 0.19 | 1.16 | 0.00 | 0.23 | 0.00 | 0.15 | 0.00 | 0.01 | 0.00 | 0.00 | 20557 |
| X-n172-k51 | 2.33 | 2.70 | 1.13 | 2.31 | 0.00 | 0.13 | 0.34 | 0.43 | 0.00 | 0.03 | 0.00 | 0.00 | 45607 |
| X-n176-k26 | 2.40 | 2.94 | 1.24 | 1.58 | 0.68 | 0.84 | 0.30 | 0.37 | 0.00 | 0.02 | 0.00 | 0.00 | 47812 |
| X-n181-k23 | 0.85 | 1.41 | 0.05 | 0.30 | 0.00 | 0.01 | 0.10 | 0.13 | 0.00 | 0.02 | 0.00 | 0.00 | 25569 |
| X-n186-k15 | 3.06 | 3.06 | 0.37 | 0.48 | 0.01 | 0.11 | 0.05 | 0.14 | 0.02 | 0.09 | 0.00 | 0.00 | 24145 |
| X-n190-k8 | 2.30 | 2.54 | 1.10 | 1.72 | 0.29 | 0.75 | 0.12 | 0.31 | 0.00 | 0.02 | 0.00 | 0.02 | 16980 |
| X-n195-k51 | 3.35 | 4.17 | 1.10 | 2.23 | 0.00 | 0.18 | 0.39 | 0.46 | 0.04 | 0.15 | 0.00 | 0.00 | 44225 |
| X-n200-k36 | 2.92 | 3.09 | 1.05 | 1.94 | 0.07 | 0.35 | 0.30 | 0.42 | 0.02 | 0.10 | 0.00 | 0.00 | 58578 |
| X-n204-k19 | 3.20 | 3.85 | 0.72 | 1.18 | 0.00 | 0.27 | 0.08 | 0.29 | 0.00 | 0.45 | 0.00 | 0.00 | 19565 |
| X-n209-k16 | 3.42 | 3.52 | 0.40 | 0.75 | 0.15 | 0.27 | 0.09 | 0.17 | 0.00 | 0.02 | 0.00 | 0.00 | 30656 |
| X-n214-k11 | 3.31 | 4.55 | 2.81 | 4.40 | 0.56 | 2.00 | 0.52 | 0.81 | 0.17 | 0.35 | 0.00 | 0.04 | 10856 |

Table 3: Comparison of solution quality of each meta-heuristic in each instance at $T_{max}$ (continued)

| Instance | GLS | | HGLS | | HILS | | KGLS | | SISR | | HGS | | BKS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Best | Avg | Best | Avg | Best | Avg | Best | Avg | Best | Avg | Best | Avg | |
| X-n219-k73 | 0.28 | 0.38 | 0.00 | 0.05 | 0.00 | 0.00 | 0.05 | 0.08 | 0.00 | 0.02 | 0.00 | 0.00 | 117595 |
| X-n223-k34 | 3.25 | 3.83 | 0.66 | 1.14 | 0.13 | 0.28 | 0.61 | 0.68 | 0.17 | 0.24 | 0.00 | 0.00 | 40437 |
| X-n228-k23 | 2.48 | 3.27 | 0.51 | 2.90 | 0.01 | 0.24 | 0.26 | 0.37 | 0.16 | 0.28 | 0.00 | 0.00 | 25742 |
| X-n233-k16 | 2.30 | 3.29 | 1.55 | 1.92 | 0.24 | 0.34 | 0.20 | 0.51 | 0.01 | 0.29 | 0.00 | 0.00 | 19230 |
| X-n237-k14 | 2.76 | 3.17 | 0.81 | 1.60 | 0.00 | 0.10 | 0.01 | 0.20 | 0.00 | 0.14 | 0.00 | 0.00 | 27042 |
| X-n242-k48 | 3.24 | 3.24 | 0.97 | 1.26 | 0.07 | 0.14 | 0.46 | 0.55 | 0.07 | 0.16 | 0.02 | 0.07 | 82751 |
| X-n247-k50 | 1.53 | 2.64 | 1.13 | 1.78 | 0.07 | 0.61 | 0.12 | 0.31 | 0.00 | 0.28 | 0.00 | 0.01 | 37274 |
| X-n251-k28 | 3.31 | 3.50 | 0.67 | 0.89 | 0.29 | 0.45 | 0.42 | 0.54 | 0.01 | 0.21 | 0.00 | 0.02 | 38684 |
| X-n256-k16 | 1.20 | 2.36 | 0.47 | 0.64 | 0.22 | 0.22 | 0.26 | 0.28 | 0.22 | 0.26 | 0.00 | 0.00 | 18839 |
| X-n261-k13 | 4.33 | 4.88 | 1.19 | 2.05 | 0.50 | 0.93 | 0.42 | 0.60 | 0.00 | 0.14 | 0.00 | 0.00 | 26558 |
| X-n266-k58 | 2.33 | 2.81 | 0.98 | 1.17 | 0.00 | 0.18 | 0.42 | 0.63 | 0.09 | 0.17 | 0.00 | 0.11 | 75478 |
| X-n270-k35 | 3.05 | 3.84 | 0.52 | 0.79 | 0.09 | 0.18 | 0.44 | 0.48 | 0.10 | 0.21 | 0.03 | 0.03 | 35291 |
| X-n275-k28 | 3.07 | 3.81 | 0.43 | 0.76 | 0.00 | 0.08 | 0.09 | 0.26 | 0.00 | 0.03 | 0.00 | 0.00 | 21245 |
| X-n280-k17 | 3.89 | 4.43 | 1.05 | 1.99 | 0.66 | 0.89 | 0.28 | 0.50 | 0.13 | 0.43 | 0.01 | 0.12 | 33503 |
| X-n284-k15 | 3.15 | 4.37 | 2.26 | 3.16 | 0.54 | 0.98 | 0.53 | 0.71 | 0.23 | 0.36 | 0.08 | 0.15 | 20215 |
| X-n289-k60 | 2.78 | 3.46 | 1.48 | 1.80 | 0.26 | 0.38 | 0.65 | 0.76 | 0.10 | 0.20 | 0.10 | 0.16 | 95151 |
| X-n294-k50 | 3.77 | 4.34 | 1.20 | 1.49 | 0.17 | 0.21 | 0.53 | 0.62 | 0.08 | 0.19 | 0.01 | 0.05 | 47161 |
| X-n298-k31 | 5.69 | 7.41 | 1.21 | 1.89 | 0.25 | 0.44 | 0.37 | 0.43 | 0.01 | 0.11 | 0.00 | 0.01 | 34231 |
| X-n303-k21 | 2.86 | 3.71 | 0.83 | 1.50 | 0.32 | 0.75 | 0.50 | 0.76 | 0.08 | 0.17 | 0.01 | 0.06 | 21736 |
| X-n308-k13 | 3.99 | 4.72 | 1.51 | 2.51 | 0.50 | 0.77 | 0.54 | 0.83 | 1.39 | 1.61 | 0.01 | 0.05 | 25859 |
| X-n313-k71 | 3.01 | 3.54 | 1.35 | 1.77 | 0.18 | 0.26 | 0.64 | 0.76 | 0.06 | 0.12 | 0.00 | 0.07 | 94043 |
| X-n317-k53 | 0.64 | 1.08 | 0.07 | 0.20 | 0.00 | 0.00 | 0.05 | 0.07 | 0.01 | 0.04 | 0.00 | 0.00 | 78355 |
| X-n322-k28 | 3.55 | 5.25 | 0.80 | 1.13 | 0.30 | 0.54 | 0.59 | 0.68 | 0.09 | 0.20 | 0.00 | 0.05 | 29834 |
| X-n327-k20 | 3.71 | 4.33 | 0.78 | 1.63 | 0.85 | 1.02 | 0.29 | 0.42 | 0.29 | 0.41 | 0.00 | 0.03 | 27532 |
| X-n331-k15 | 4.28 | 4.74 | 0.68 | 1.41 | 0.11 | 0.40 | 0.03 | 0.31 | 0.06 | 0.07 | 0.00 | 0.00 | 31102 |

Table 4: Comparison of solution quality of each meta-heuristic in each instance at $T_{max}$ (continued)

| Instance | GLS | | HGLS | | HILS | | KGLS | | SISR | | HGS | | BKS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Best | Avg | Best | Avg | Best | Avg | Best | Avg | Best | Avg | Best | Avg | |
| X-n336-k84 | 2.65 | 2.92 | 2.28 | 2.89 | 0.17 | 0.32 | 1.14 | 1.22 | 0.12 | 0.23 | 0.07 | 0.12 | 139111 |
| X-n344-k43 | 3.47 | 4.51 | 0.93 | 1.29 | 0.33 | 0.61 | 0.42 | 0.71 | 0.07 | 0.17 | 0.03 | 0.06 | 42050 |
| X-n351-k40 | 4.42 | 5.60 | 3.17 | 3.94 | 0.59 | 0.91 | 0.97 | 1.13 | 0.27 | 0.31 | 0.11 | 0.18 | 25896 |
| X-n359-k29 | 3.80 | 4.37 | 1.30 | 1.67 | 0.61 | 1.12 | 0.30 | 0.76 | 0.02 | 0.09 | 0.12 | 0.22 | 51505 |
| X-n367-k17 | 3.32 | 4.44 | 1.08 | 1.67 | 0.62 | 0.75 | 0.23 | 0.57 | 0.03 | 0.10 | 0.00 | 0.00 | 22814 |
| X-n376-k94 | 0.62 | 0.71 | 0.07 | 0.11 | 0.00 | 0.00 | 0.06 | 0.10 | 0.02 | 0.03 | 0.00 | 0.00 | 147713 |
| X-n384-k52 | 3.81 | 4.47 | 0.80 | 1.18 | 0.62 | 0.70 | 0.64 | 0.76 | 0.16 | 0.26 | 0.09 | 0.17 | 65940 |
| X-n393-k38 | 5.07 | 6.19 | 1.01 | 1.49 | 0.26 | 0.66 | 0.45 | 0.54 | 0.20 | 0.32 | 0.00 | 0.00 | 38260 |
| X-n401-k29 | 2.58 | 3.06 | 0.76 | 1.03 | 0.65 | 0.85 | 0.46 | 0.51 | 0.09 | 0.12 | 0.07 | 0.14 | 66163 |
| X-n411-k19 | 4.18 | 5.28 | 1.53 | 2.97 | 0.62 | 1.30 | 0.35 | 1.07 | 0.23 | 0.33 | 0.02 | 0.04 | 19712 |
| X-n420-k130 | 2.76 | 3.40 | 1.03 | 1.77 | 0.00 | 0.04 | 0.35 | 0.46 | 0.01 | 0.05 | 0.01 | 0.04 | 107798 |
| X-n429-k61 | 3.91 | 4.95 | 0.97 | 1.32 | 0.37 | 0.51 | 0.53 | 0.62 | 0.07 | 0.14 | 0.05 | 0.08 | 65449 |
| X-n439-k37 | 2.10 | 3.36 | 0.59 | 0.80 | 0.03 | 0.16 | 0.15 | 0.25 | 0.03 | 0.18 | 0.01 | 0.01 | 36391 |
| X-n449-k29 | 4.88 | 5.47 | 2.49 | 3.01 | 1.64 | 1.85 | 0.79 | 0.96 | 0.11 | 0.28 | 0.13 | 0.24 | 55233 |
| X-n459-k26 | 5.10 | 6.56 | 1.16 | 2.08 | 0.92 | 1.39 | 0.39 | 0.46 | 0.20 | 0.37 | 0.00 | 0.10 | 24139 |
| X-n469-k138 | 3.75 | 3.96 | 1.57 | 1.92 | 0.05 | 0.16 | 0.57 | 0.74 | 0.12 | 0.19 | 0.04 | 0.16 | 221824 |
| X-n480-k70 | 3.25 | 3.74 | 0.68 | 1.01 | 0.20 | 0.45 | 0.53 | 0.60 | 0.01 | 0.07 | 0.05 | 0.08 | 89449 |
| X-n491-k59 | 4.94 | 6.11 | 1.80 | 2.15 | 0.91 | 1.12 | 0.82 | 0.98 | 0.02 | 0.18 | 0.12 | 0.23 | 66487 |
| X-n502-k39 | 1.15 | 1.34 | 0.21 | 0.29 | 0.16 | 0.22 | 0.12 | 0.16 | 0.02 | 0.07 | 0.01 | 0.02 | 69226 |
| X-n513-k21 | 4.32 | 6.36 | 1.13 | 2.23 | 0.47 | 0.84 | 0.38 | 0.66 | 0.15 | 0.38 | 0.00 | 0.00 | 24201 |
| X-n524-k153 | 1.11 | 1.47 | 0.67 | 1.09 | 0.04 | 0.38 | 0.53 | 0.71 | 0.11 | 0.19 | 0.03 | 0.10 | 154593 |
| X-n536-k96 | 3.99 | 4.73 | 1.42 | 1.74 | 0.83 | 0.88 | 0.95 | 1.04 | 0.21 | 0.29 | 0.18 | 0.24 | 94868 |
| X-n548-k50 | 2.66 | 3.00 | 0.35 | 0.54 | 0.13 | 0.32 | 0.23 | 0.27 | 0.01 | 0.10 | 0.01 | 0.09 | 86700 |
| X-n561-k42 | 5.76 | 6.65 | 1.29 | 1.84 | 0.47 | 0.88 | 0.63 | 0.73 | 0.19 | 0.37 | 0.02 | 0.06 | 42717 |
| X-n573-k30 | 2.70 | 3.36 | 0.90 | 1.45 | 0.84 | 1.04 | 0.35 | 0.56 | 0.20 | 0.33 | 0.17 | 0.28 | 50673 |

Table 5: Comparison of solution quality of each meta-heuristic in each instance at $T_{max}$ (end)

| Instance | GLS | | HGLS | | HILS | | KGLS | | SISR | | HGS | | BKS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Best | Avg | Best | Avg | Best | Avg | Best | Avg | Best | Avg | Best | Avg | |
| X-n586-k159 | 3.81 | 4.05 | 1.32 | 1.66 | 0.20 | 0.27 | 0.49 | 0.57 | 0.07 | 0.17 | 0.08 | 0.14 | 190316 |
| X-n599-k92 | 3.88 | 4.35 | 1.22 | 1.40 | 0.61 | 0.92 | 0.62 | 0.83 | 0.14 | 0.22 | 0.14 | 0.19 | 108451 |
| X-n613-k62 | 6.33 | 7.08 | 1.93 | 2.65 | 1.32 | 1.53 | 0.96 | 1.11 | 0.12 | 0.29 | 0.17 | 0.27 | 59535 |
| X-n627-k43 | 3.76 | 4.21 | 1.28 | 1.59 | 1.21 | 1.41 | 0.52 | 0.65 | 0.09 | 0.21 | 0.12 | 0.33 | 62164 |
| X-n641-k35 | 4.44 | 4.74 | 1.02 | 1.78 | 1.39 | 1.57 | 0.40 | 0.62 | 0.17 | 0.25 | 0.14 | 0.28 | 63694 |
| X-n655-k131 | 0.86 | 0.96 | 0.20 | 0.28 | 0.00 | 0.01 | 0.15 | 0.17 | 0.03 | 0.06 | 0.00 | 0.03 | 106780 |
| X-n670-k130 | 3.14 | 3.65 | 1.35 | 2.22 | 1.11 | 1.31 | 0.78 | 0.90 | 0.23 | 0.43 | 0.21 | 0.30 | 146332 |
| X-n685-k75 | 6.68 | 7.94 | 2.52 | 3.23 | 0.93 | 1.14 | 0.62 | 0.94 | 0.10 | 0.26 | 0.12 | 0.20 | 68205 |
| X-n701-k44 | 5.41 | 5.90 | 1.54 | 1.79 | 1.28 | 1.49 | 0.64 | 0.72 | 0.10 | 0.16 | 0.19 | 0.38 | 81923 |
| X-n716-k35 | 5.07 | 5.71 | 1.60 | 2.19 | 1.52 | 1.98 | 0.55 | 0.79 | 0.14 | 0.24 | 0.17 | 0.27 | 43387 |
| X-n733-k159 | 4.53 | 5.31 | 1.81 | 2.39 | 0.48 | 0.60 | 0.73 | 0.81 | 0.11 | 0.19 | 0.10 | 0.17 | 136190 |
| X-n749-k98 | 5.81 | 6.64 | 2.53 | 2.84 | 1.15 | 1.29 | 1.02 | 1.15 | 0.11 | 0.28 | 0.32 | 0.44 | 77314 |
| X-n766-k71 | 5.91 | 7.03 | 2.02 | 2.62 | 0.82 | 1.21 | 0.48 | 0.64 | 0.26 | 0.33 | 0.20 | 0.27 | 114454 |
| X-n783-k48 | 5.69 | 6.61 | 1.92 | 2.48 | 1.52 | 1.86 | 0.79 | 0.89 | 0.21 | 0.33 | 0.43 | 0.54 | 72394 |
| X-n801-k40 | 3.88 | 4.09 | 0.92 | 1.24 | 0.90 | 1.13 | 0.27 | 0.39 | 0.08 | 0.15 | 0.12 | 0.27 | 73305 |
| X-n819-k171 | 3.81 | 4.21 | 1.55 | 1.82 | 0.72 | 0.78 | 0.80 | 0.91 | 0.14 | 0.19 | 0.17 | 0.25 | 158121 |
| X-n837-k142 | 3.86 | 4.01 | 1.24 | 1.44 | 0.57 | 0.68 | 0.64 | 0.72 | 0.07 | 0.11 | 0.19 | 0.25 | 193737 |
| X-n856-k95 | 2.35 | 2.89 | 0.55 | 0.71 | 0.13 | 0.34 | 0.28 | 0.41 | 0.09 | 0.16 | 0.02 | 0.08 | 88965 |
| X-n876-k59 | 3.61 | 4.13 | 1.14 | 1.44 | 1.11 | 1.18 | 0.75 | 0.82 | 0.11 | 0.19 | 0.30 | 0.38 | 99299 |
| X-n895-k37 | 6.50 | 7.44 | 2.50 | 2.80 | 1.82 | 2.11 | 0.70 | 0.82 | 0.23 | 0.39 | 0.30 | 0.39 | 53860 |
| X-n916-k207 | 3.45 | 3.78 | 1.48 | 1.66 | 0.53 | 0.60 | 0.55 | 0.58 | 0.07 | 0.12 | 0.12 | 0.20 | 329179 |
| X-n936-k151 | 4.83 | 5.52 | 3.28 | 4.32 | 1.49 | 1.72 | 0.74 | 0.83 | 0.35 | 0.58 | 0.30 | 0.48 | 132725 |
| X-n957-k87 | 3.13 | 3.54 | 0.69 | 0.96 | 0.46 | 0.60 | 0.22 | 0.33 | 0.03 | 0.11 | 0.05 | 0.10 | 85465 |
| X-n979-k58 | 3.56 | 3.95 | 1.01 | 1.39 | 0.96 | 1.31 | 0.48 | 0.51 | 0.07 | 0.10 | 0.16 | 0.22 | 118987 |
| X-n1001-k43 | 6.17 | 7.33 | 2.60 | 2.82 | 1.88 | 2.43 | 0.72 | 0.88 | 0.08 | 0.24 | 0.44 | 0.54 | 72359 |

For a better visualization of the mean Gap values of the table 1, a box diagram was drawn with the results of each metaheuristic in Figure 7 below.

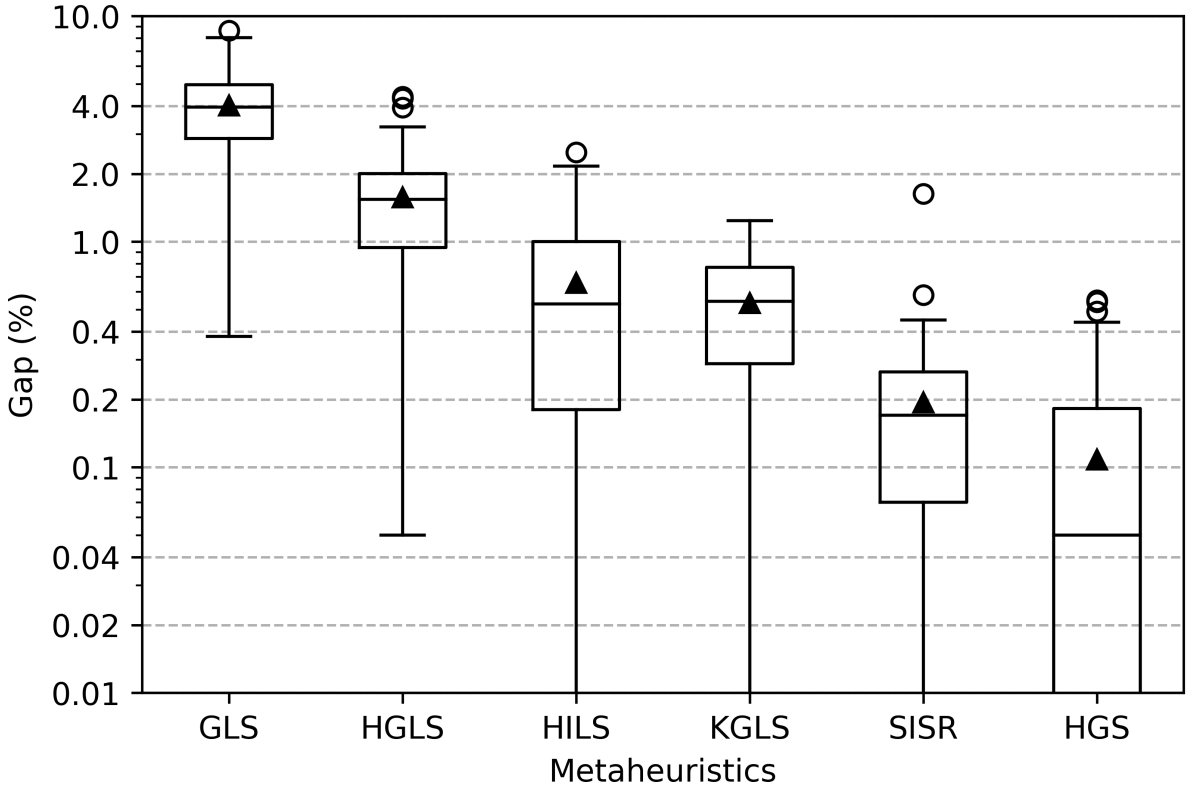


Figure 7: Box-plot with the average results of the metaheuristics at $T_{max}$.

As we can see in Table 1 and Figure 7, the latest generation metaheuristics such as SISR and HGS, compared under computationally similar conditions to others based on local search (with acronyms starting with LS) , are clearly superior in terms of the quality of the solutions found. The results obtained by HGLS, however, can be understood as satisfactory, since the hybridization proposed in this work for GLS surpassed its classical implementation with relative slack. Also as a metric for evaluating the convergence of algorithms, changes were made to the HGLS so that, like the others, it would return the value of the best solution so far found at different stages of the search time. In Figure 8, it is possible to observe the convergence profile of the search of the analyzed metaheuristics.

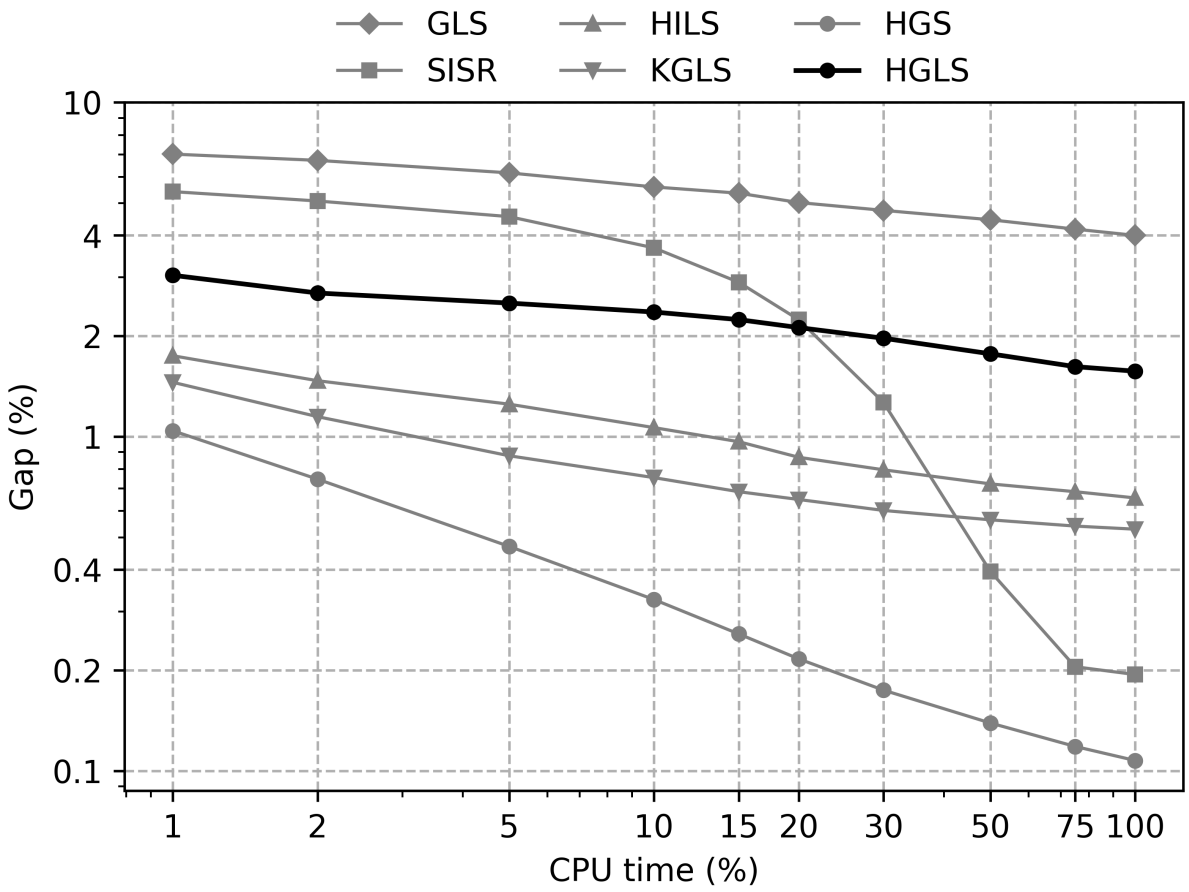


Figure 8: Convergence profiles of metaheuristics over CPU time ($T_{max}$).

As in the quality results, the HGS also obtained the best convergence profile, whereas the SISR, as it is a generalization of the simulated annealing metaheuristic, has the expected behavior, due to its criterion for accepting new solutions (with an exponential drop temperature), favors exploration during the initial stages of the search. HGLS, on the other hand, presented an expected convergence profile, very similar to GLS, HILS and KGLS.

## Conclusions and Future Works

Although a computational environment was not used, exactly similar to that of Vidal [2020], the computational experiments (presented in Section 6) were carried out in an environment with single thread processing with performance of 96% in relation to the work environment mentioned above (based on PassMark Software [2021], being very close to each other). In the computational experiments performed, it can be seen that the modifications employed in HGLS in relation to

GLS, in fact, produced a substantial improvement in the quality of the solutions found (see Table 1 and Figure 7 ), but compared to the state-of-the-art metaheuristics in the resolution of CVRP, this quality was considerably lower. Another metric to be analyzed, as Vidal [2020] clarifies, is the convergence profile of the quality of the solutions found over the search time of the algorithms, which can provide a better understanding of the specific causes and effects on the quality of the final solution of the algorithm. From this understanding, it can be seen (according to Figure 8, of Section 6), that the HGLS convergence profile was very similar to that of the local search algorithms (HILS and KGLS) listed in the state of the art, and that an improvement in the method of generating the initial solution of HGLS proposed in this work, may culminate in quality results closer to or even equivalent to those of the methods of the state of the art. For future works that are interested in the HGLS developed in this work, we suggest the use of the Initial Solution generation method, used in the KGLS of Arnold and Sorensen [2019] or the dynamic programming method for the grouping of routes, used in the HGS of Vidal [2020] and that was very detailed in Vidal [2016]. It is also advisable to use a low-level programming language such as C++, instead of Python 3.8 used in this work, which can bring better performance to HGLS.

## Acknowledgments

The author would like to thank Prof. Dr. Narciso Ferreira dos Santos Neto, my advisor, Prof. Dr. João Batista Mendes, my co-advisor, Prof. Dr. Gladyston Mattos Ribeiro from the Department of Transport Engineering at COPPE/UFRJ and Prof. Dr. José Luiz Lopes Teixeira Filho, Department of Transport and Geotechnics, Faculty of Engineering/UFJF, for his great contribution to the development of this work. And in addition, thanks to Marc-Antoine Augé, Optimization Scientist at LocalSolver for kindly helping to model the problem.

## References

Florian Arnold and Kenneth Sorensen. Knowledge-guided local search for the vehicle routing problem. *Computers & Operations Research*, 105:32–46, 2019.

P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, and D. Naddef. Separating capacity constraints in the cvrp using tabu search. *European Journal of Operational Research*, 106(2): 546–557, 1998.

Philippe Augerat, José Manuel Belenguer, Enrique Benavent, Angel Corberán, D. Naddef, and Giovanni Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. 01 1995.

J.E. Beasley. Route first-cluster second methods for vehicle routing. *Omega*, 11(4):403–408, 1983.

Jacek Blazewicz, Maciej Drozdowski, Boleslaw Soniewicki, and Rafal Walkowiak. Two-dimensional cutting problem - basic complexity results and algorithms for irregular shapes. *Foundations of Control Engineering*, 14, 01 1989.

Jan Christiaens and Greet Vanden Berghe. Slack induction by string removals for vehicle routing problems. *Transportation Science*, 54(2):417–433, 2020. doi:10.1287/trsc.2019.0914.

N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Journal of the Operational Research Society*, 20(3):309–318, Sep 1969.

Jean-François Cordeau, Gilbert Laporte, Martin Savelsbergh, and Daniele Vigo. *Vehicle Routing*, volume 14, pages 195–224. 01 2007.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1): 80–91, 1959.

Andrew Davenport, Edward Tsang, Chang Wang, and Kangmin Zhu. Genet: A connectionist architecture for solving constraint satisfaction problems by iterative improvement. *Proceedings of the National Conference on Artificial Intelligence*, 1, 08 1994.

Marshall L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42(4):626–642, 1994. ISSN 0030364X, 15265463.

Pierre Hansen and Nenad Mladenović. First vs. best improvement: An empirical study. *Discrete Applied Mathematics*, 154(5):802–817, 2006. IV ALIO/EURO Workshop on Applied Combinatorial Optimization.

Raphael Harry Frederico Ribeiro Kramer, Anand Subramanian, and Puca Huachi Vaz Penna. Problema de roteamento de veículos assimétrico com frota heterogênea limitada: um estudo de caso em uma indústria de bebidas. *Gestão & Produção*, 23:165–176, 03 2016.

G. Laporte, Stefan Røpke, and T. Vidal. *Heuristics for the Vehicle Routing Problem*, pages 87–116. Vehicle Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics, 2 edition, 2014.

Gilbert Laporte and Frédéric Semet. *5. Classical Heuristics for the Capacitated VRP*, pages 109–128. 01 2002.

LocalSolver *Software*. Localsolver, 2020. URL `https://www.localsolver.com/about.html`. Versão: 10.5.

Pedro Munari, Twan Dollevoet, and Remy Spliet. A generalized formulation for vehicle routing problems. Technical report, ArXiV, 2017.

PassMark Software. CPU Benchmark, 2021. URL `https://www.cpubenchmark.net/`. Acessedo em 27-01-2021.

Laurent Perron and Vincent Furnon. OR-Tools. Google, 2019. URL `https://developers.google.com/optimization/`. Versão: 8.1.

C. Prins, N. Labadi, and M. Reghioui. Tour splitting algorithms for vehicle routing problems. *International Journal of Production Research*, 47(2):507–535, 2009.

Christian Prins, Philippe Lacomme, and Caroline Prodhon. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies*, 40:179–200, 2014.

Anand Subramanian, Eduardo Uchoa, and Luiz Satoru Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531, 2013.

Paolo Toth and Daniele Vigo. The vehicle routing problem. *SIAM*, 01 2002. doi:10.1137/1.9780898718515.

Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

Thibaut Vidal. Technical note: Split algorithm in o(n) for the capacitated vehicle routing problem. *Computers & Operations Research*, 69:40–47, 2016.

Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. Technical report, ArXiV, 2020.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.