Cluster-based Hyper-Heuristic for Large-Scale Vehicle Routing Problem

Joao Guilherme Cavalcanti Costa
Victoria University of Wellington
Wellington, New Zealand
Joao.Costa@ecs.vuw.ac.nz

Yi Mei
Victoria University of Wellington
Wellington, New Zealand
Yi.Mei@ecs.vuw.ac.nz

Mengjie Zhang
Victoria University of Wellington
Wellington, New Zealand
Mengjie.Zhang@ecs.vuw.ac.nz

***Abstract*—One of the most known transportation problems, the Large-Scale Vehicle Routing Problem (LSVRP) requires more sophisticated methods to be solved due to the sheer amount of customers. Most current methods include manually designed heuristics and parameters, such as restrictions in the search space. Hyper-heuristics(HHs) appear as a counterpoint to the manually designed complex methods. This paper presents a preliminary study on adaptive search space based on clustering, utilizing a HH Selection framework with Genetic Algorithm (GA). The initial results show promise in having an adaptive search scope when compared to a fixed clustering approach. A comparison of the effects of having a route-first vs cluster-first initial solution is also presented, favouring the latter one, as well as a comparison between two types of chromosome decoding. Finally, the proposed method is compared to a manually designed algorithm, producing results with better quality. The method is shown to be significantly better for most scenarios, achieving solutions just as good as when no limits are applied, but in a much shorter time.**

***Index Terms*—Hyper-Heuristics, Vehicle Routing Problem, Clustering, Large-Scale**

I. Introduction

One of the main problems in transportation is the Vehicle Routing Problem (VRP). The VRP has several applications such as the delivery of goods and services such as waste collection or postal service. The core of the classical VRP is to find routes that attend every customer without visiting any of them more than once. The problem was introduced by [1], which solved the distribution of fuel for several stations. Several variants for the original problem were later proposed by several authors, [2] and [3] review those variants and solution methods. One of these variations is concerned about a large quantity of customers, also known as Large-Scale VRP (LSVRP). This variant is concerned in solving problems by a reasonable time-frame and without having to invest too much in computational power, since a larger number of customer implies in more memory and time-consuming computation. However, this size of problem is much more realistic for companies, and any reduction in distribution cost will have a significant impact due to the large scale nature. For the methods that do tackle such scale, most are very dependent on high-level knowledge on the subject. The most successful algorithms are manually designed meta-heuristics [3], making them not generalisable across datasets. Additionally, although most successful algorithms utilize some form of divide-and-conquer or limits to deal with the larger search space, they still do that manually, such as [4] [5].

Hyper-heuristics (HHs) try to tackle the design level complexity by building a more general framework using low-level and easy-to-implement heuristics, usually having a trade-off on quality over time [6] [7]. The lack of HHs work for the LSVRP, however, makes it difficult to compare both types of heuristics to each other. For example, when looking for low-level heuristics (LLH) to apply in problems of such scale, even though HHs might efficiently look for the most suitable heuristic and the standard is that the search space size is not even considered. When the search space is considered, it will usually be manually designed, where it can be either too small, avoiding finding better solutions, or too large and making it very slow, and this decision is made based on the expertise of the developer, with little to no test or proof. The general rule for limiting the search space is based on how many customers/edges to consider, which is based on a manually set parameter or the result of a fixed clustering algorithm.

The work of [8] shows that it is possible to have a non-manually designed neighbourhood search space, they do it based on distance and angle between routes. However, the considered threshold is based on the routes rather than global distribution. As shown by [9], some characteristics such as longest edges and their gravity centres are not reliable as indicators of good solutions. Therefore we propose an approach that aims to consider a more general overview of the search space, based on clusters of customers which adaptively change throughout the search.

The current HHs studies are limited in the fact that they do not employ pruning techniques at the LLH level, and therefore have limited scalability. To deal with those limitations, a HH framework for the LSVRP is proposed in this study. The proposed Cluster-based Hyper-Heuristic (CbHH) searches the heuristic space and automatically trims the search space based on the evolution of the solution. The CbHH automatically evolves the state of the exploration space with the help of a Genetic Algorithm (GA), which also contributes to the search with its parallel concept. The CbHH does not require sophisticated design and explores the solution space based on adaptive clusters. Additionally, two different types of chromosome decoding schemes are explored. The goals of this work, in the context of HHs for solving the large-scale

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

VRP, are as follows:

- **Objective 1**: to explore whether an adaptive search space will improve the computational time without losing too much quality, compared to both having no clusters or fixed ones.
- **Objective 2**: to investigate how the solutions are affected by the starting clusters.
- **Objective 3**: to study how the solutions are affected by different types of chromosome decoding schemes.
- **Objective 4**: to question whether the Cluster-based HH can find competitive results when compared to traditional manually designed heuristics.

The rest of the paper is organised as follows. In the next section (II) a brief problem description for the VRP is presented, followed by a succinct review of the literature. Next, Section III expands on the Cluster-based Hyper-heuristics. Section IV presents the experimental design, followed by the results and discussions on Section V. Finally, Section VI presents final remarks and research perspectives.

II. Background

A. Problem Description

The Vehicle Routing Problem in its classical form (also known as Capacitated VRP, or CVRP) is defined as a graph $G = (V, E)$, where V is the vertex set in the graph, and E the edges set. In other words, the vertex represents the customers and the edges are the paths between them. The set V is given as $V = \{0, 1, \ldots, n\}$, where 0 represents the depot, and the set E is given as $E = \{(i, j) : i, j \in V, i \neq j\}$. Additionally, there is a fleet of identical vehicles with a maximum capacity of Q each. Each customer has a demand $q_i > 0$, representing the amount or the weight of the parcel it requires. The goal of the classical VRP is to find the optimal routes, i.e. routes with the least possible length. These routes must visit all customers exactly once; they must not exceed each vehicle capacity, and all routes must start and end at the depot.

B. Related Work

Among the different variants for the Vehicle Routing Problem, the Large-Scale VRP (LSVRP) is particularly hard to solve due to its size. Although it is not a clear rule, to be considered a large scale, an instance needs to have at least 200 customers [10] [11] [12] [4]. The most advanced exact methods can optimally solve up to 600 customers, but only for specific instances [13], and require hours of computational time, which is expected due to the problem's $\mathcal{NP}$-hardness [2]. Heuristics that aim to find the best possible solution within a reasonable time has become a promising alternative to these exact methods.

While some heuristics methods can efficiently solve instances of large scale, the emphasis of most studies so far has been tested only to the smaller ones. This focus leads to very good and fast results to these small instances, but which usually do not scale well for larger ones. For example, [14] have solved several variations of the VRP with their Hybrid Genetic Algorithm achieving state-of-the-art results, but it takes several minutes for large instances, taking more than 100 minutes for 600 customers and more. The hybrid Iterated Local Search (ILS) from [15] follows the same pattern after instances of that size or larger. Both pointed by [4].

As a counter-point to these manually designed heuristics, the Hyper-Heuristics (HHs) operates at a higher level of abstraction than meta-heuristics [16] and are usually proposed as frameworks which are designed to provide a more general solution for any type of instance domain [17]. Some of these HHs frameworks are applied to different types of problems, some including VRP, such as [18] [19] [20] [12]. Generally speaking, the idea of a HH is to let some mechanism choose which kind of search or construction step will be applied in each iteration. For that, HHs utilize low-level heuristics (LLH), which are heuristics of easy implementation and generic for one (or more) problem(s), without the need of extensive and carefully elaborated search steps. One of the goals is to have basic and easy-to-understand heuristics at the application level and to leave the more carefully designed techniques to the high-level layer, which minimizes the need for expertise on the given problem. The focus of HHs work has been on this aspect, but not acknowledging scalability issues at the LLH-level. For more on this subject and HHs applications, the reader can refer to [6].

As for HHs solving the VRP, the work of [21] revise the framework of [19] with an adaptive ILS, tackling the VRP with Time Windows (VRPTW), a variation which customers have a service window. Genetic Programming (GP), an evolutionary approach that evolves programs instead of solutions, are regularly used with the Dynamic VRP (DVRP), as in [22] and [23]. The work of [24] focus on a bi-objective variation, the Mixed-Shift VRPTW (MSVRPTW). In [25] the authors propose a Grammatical-based HH to solve the CVRP. The work of [7] combines both the selective and generative types of HHs in a GP framework for the VRP. But, to the best of our knowledge, so far only one work uses HH for a Large-Scale variant of the VRP. In [12] the search space is divided into m sub-problems which are solved by column generation techniques to solve the LSVRP with Time-Windows. The combination is improved with a HH framework, where a multi-armed bandit mechanism search for which low-level heuristic to be applied next. No emphasis is given to the scalability of the LLH.

As shown in the review of [11], most VRP approaches apply a form of reducing the search space when dealing with such size to achieve scalable methods. A good portion of them is based on a manually set threshold on the neighbourhood search space or by using clusters. To cite some examples, in [11] the authors explore a database with geographical and experts experience data to partition the customers into clusters, and in combination with some AI techniques return a sequence of clusters to be visited. By applying this partitioning process they are providing a smart way to reduce the search space, which resulted in reducing the number of used vehicles. More recently, [4] have designed a local search algorithm for very large scale VRPs (with up to 30000 customers), starting from a simple and quick solution, and improving it by adding

several pruning techniques to the search space, for example, considering just the closest 100 customers in a neighbourhood move. This produces an effective limit to the enormous amount of neighbours, achieving state-of-the-art solutions.

The use of clustering or pruning techniques are not limited to large scale applications. For example, in [26] the authors utilize clusters to reduce the search space to within neighbouring routes. Although the clusters are fixed, they still provided a balance between effectiveness and efficiency.

What all previously mentioned work have in common, is that they have a parameter-based and fixed limit on the search space, whether by experimentation, manually set, or set by as a result of an auxiliary algorithm. This limit can become a liability when searching for better solutions since the improving step can be outside this search range. To give a practical example, as pointed by [27], the optimal solution in a given 532-city problem utilizes the 22nd nearest neighbour, and any fixed size limit that explores less than 22 neighbours would likely not find the optimal. Although this example is for a Travelling Salesperson Problem, the principle is the same for the VRP: any fixed limit can easily avoid finding better solutions, especially with larger instances since they have more possibilities, and it may not be the most suitable throughout the whole search.

To tackle this issue of finding a suitable limit to the search space, and the complexity of design previously mentioned, the Cluster-based Hyper-Heuristic (CbHH) is proposed. This framework combines a selection of perturbation heuristics and an adaptive cluster model, aiming to find a search space which is compact enough to be easily explored, and also not hard-limited, allowing changes in such a way that it can find better solutions. Besides, the method also has a parallel evolution using a Genetic Algorithm (GA) as the selection heuristic, which would search for several outcomes.

III. Cluster-based Hyper-Heuristics Approach

The proposed method aims to prune the search space automatically. The idea is to initially create clusters based on the customers' location, and let the clusters evolve according to where the better solutions are heading to, as seen in Fig. 2. In the example, there is a randomly created customer-distribution (2a), then the initial clusters are created (2b), and as the evolution process progresses, the clusters will adaptively change (2c). The solution chromosome is detailed and the utilized Low-Level Heuristics are specified next. Followed by the framework specifications, concluded by the Genetic Algorithm (GA) details.

A. Solution Representation

A solution represents a sequence of low-level heuristics that will be applied to improve an initial solution. The chromosome is a string of integers (as shown in Figure 1) where each allele represents the index of one of the LLH, which is essentially one neighbourhood type. Each value is unique for intra and inter clusters heuristics. In the Figure, for example, the *3* can be a Two-Opt* applied only inside each cluster, while *4* could be Two-Opt* applied inter-cluster. Additionally, the chromosome has variable size, it can grow or shrink depending on the evolution process, which will be detailed in Section III-C.

Fig. 1: Example of a chromosome.

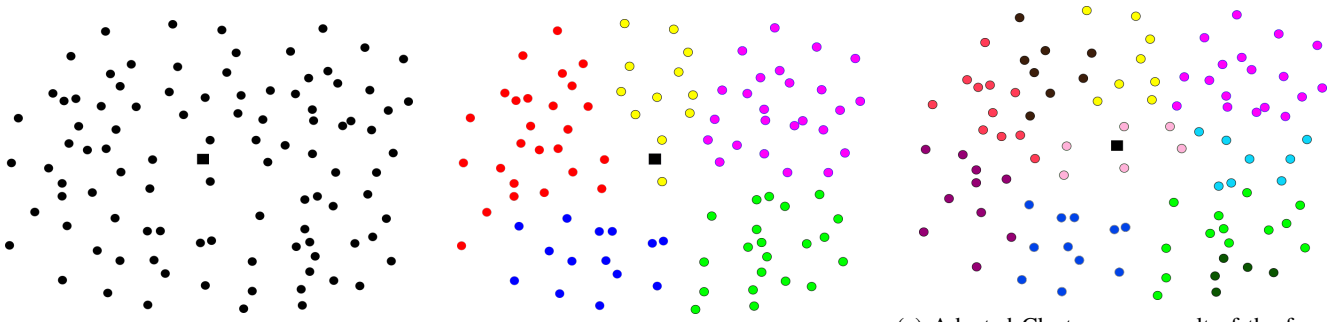
The LLHs are classical local-search operators which are here divided into two groups based on how their neighbourhoods are explored: the first group considers the whole cluster of a given route, named as "regular"; and the second group will consider M closest clusters, here named "cluster-based"; both are detailed in the next sub-section. Some of the LLH are utilized in both groups but are characterized by a different integer. Next a list of these LLH, totalling 11 possible heuristics, since the ones in bold are utilized in both groups, otherwise, it is specified where they are applied.

- **Low-Level Heuristics 1: Cross-Exchange**, swaps a sub-string of customers between two routes. Indexes 0 and 1, respectively for each type.
- Low-Level Heuristics 2: 2-OPT, modifies two edges in a given route. Used only in the local cluster. Index 2.
- **Low-Level Heuristics 3: 2-OPT***, swap two edges in two given routes. Indexes 3 and 4, respectively for each type.
- Low-Level Heuristics 4: OR-OPT, transfers a sub-string with a length of 2, 3 or 4 to another position. Used only in the local cluster. Index 5.
- Low-Level Heuristics 5: OR-OPT*, transfers a sub-string with a length of 2, 3 or 4 to another route. Used only between clusters. Index 6.
- **Low-Level Heuristics 6: MERGE**, combines two different routes into one, by concatenating one into any position of the other. Indexes 7 and 8, respectively for each type.
- **Low-Level Heuristics 7: Reverse 2-OPT***, swap two edges in two given routes, but reversing the order of the remaining route. Indexes 9 and 10, respectively for each type.

B. Framework

The overall idea of the framework is to evolve a sequence of operators that will improve an initial solution. Each operator in the sequence will consider the limits set by the clusters based on its group (detailed in the previous sub-section), modifying the clusters according to the result of each move and repeating the process for each chromosome, always starting from the same initial solution. The evolutionary process evolves the population of sequences. The final output of the framework will be the sequence of operators (and implicitly the sequence of cluster moves) which resulted in the best improvement. Fig. 3 show the Hyper-Heuristic workflow.

Following Algorithm 1, the initial clusters are created using the K-means algorithm (line 2), a partitioning algorithm which



(a) Original instance (b) Initial Clusters (c) Adapted Clusters as a result of the framework

Fig. 2: In this figure, there is a fictional example of how the cluster-based search adaptively changes the cluster space.

is easy to implement, fast and efficient [28]. Then an initial solution is created respecting the clusters' limits (line 3). For this work, the savings algorithm of Clark and Wright (CW) [29] was selected for the initial solution, since it is a fast deterministic algorithm with fairly good results. Next, the evolutionary process (lines 4-21) starts by creating the initial population (line 4). The main loop starts by selecting each individual in the population (lines 6-7), and copies of the initial solution and clusters are created (lines 8-9), only the copies are updated by the LLH. Then, for each allele in the individual (line 10) the correspondent LLH is applied to the current solution (line 11). The decoding process, shown in lines 13-14, determines if the next LLH will be the first one or the next one, based on the type of decoding chosen. If the current solution is better then the current best, the current solution becomes the new best (lines 15-16). If the stopping criteria are not met (maximum number of generations), the crossover and mutation steps occur (line 18), otherwise the algorithm return the best individual (line 20).

The clusters are updated according to the type of LLH applied, as shown in Algorithm 2. The cluster-based type is the one which explores more than the cluster for improvements and is composed of only inter-route LLHs. Their neighbourhoods consider the M closest clusters considering their centroids C_i (defined by Equations 1, 2 and 3, with Cl_i being the cluster i), where M is a percentage of the total number of clusters. For example, in an instance with 10 clusters, if the next selected LLH is Two-Opt* and $M = 20\%$, then for each cluster, the two (20% of 10) closest clusters will be considered. This is the main diversification step (other than the genetic process), since it allows for a larger, yet limited, neighbourhood to be explored. The clusters are updated based on the resulting changes on the routes and whether they are majority inside a cluster or not. The procedure, summarized in Algorithm 3, receives the clusters and the routes from a VRP solution. Then for each route, it verifies whether a new cluster will be created or just update the current ones (as shown in Figures 2b and 2c). It does that with the function *ClusterWithMajority*, which returns the cluster in which most of the customers belongs to, if it has at least 50% of the route, otherwise returns 0. It can be noted that a new cluster will only contain the customer from one route, however on the next iteration of the selection process, new customers can be added (or even removed), dynamically changing the clusters and search space. As an example, suppose in a given solution state, one of the routes starts at a given cluster **A**, passes through cluster **B**, and ends in cluster **C**, as a result of applying the Two-Opt* algorithm on cluster **A**, with **B** and **C** being considered neighbour clusters. And suppose this route contains 15, 14 and 17 customers in each cluster, respectively. Then this route will flourish as a new cluster **D** since the majority of the route is not within one cluster (the majority would be at least 50% of $15+14+17$). Another example, if some route had 10, 4 and 6 in each cluster, respectively, then the customers from **B** and **C** would become part of cluster **A**. If these customers were the only customers in either **B** or **C**, these clusters would be removed from the cluster list.

Algorithm 1 Clustering-based Selection Perturbative Hyper-Heuristic

```
1:  procedure CBSPHH(an instance, a list of LLHs)
2:      clusters ← Kmeans()
3:      initial_solution ← CW_Savings(clusters)
4:      population ← InitializePopulation()
5:      best ← ∅
6:      loop
7:          for each individual ∈ population do
8:              new_s ← initial_solution
9:              new_c ← clusters
10:             for each allele ∈ individual do
11:                 new_s ← LLH(allele, new_s, new_c)
12:                 Eval(new_s)
13:                 if Improvement and VNS mode then
14:                     Restart from first Allele
15:             if Best Improvement then
16:                 best ← individual
17:         if Stopping Criteria not met then
18:             Evolve the population by Crossover and Mutation
19:         else
20:             return best
```

$$\bar{x_i} = \frac{\sum_{j \in Cl_i} x_j}{|Cl_i|} \tag{1}$$

$$\bar{y_i} = \frac{\sum_{j \in Cl_i} y_j}{|Cl_i|} \tag{2}$$

$$C_i = (\bar{x_i}, \bar{y_i}) \tag{3}$$

A regular LLH will only consider its cluster, therefore the idea is to improve each cluster individually. The type of heuristic determines if they are inter-route or intra-route, but whenever possible, both are explored. This step is considered as an intensification one, which will end up in a local minimum for all clusters, according to the selected LLH.

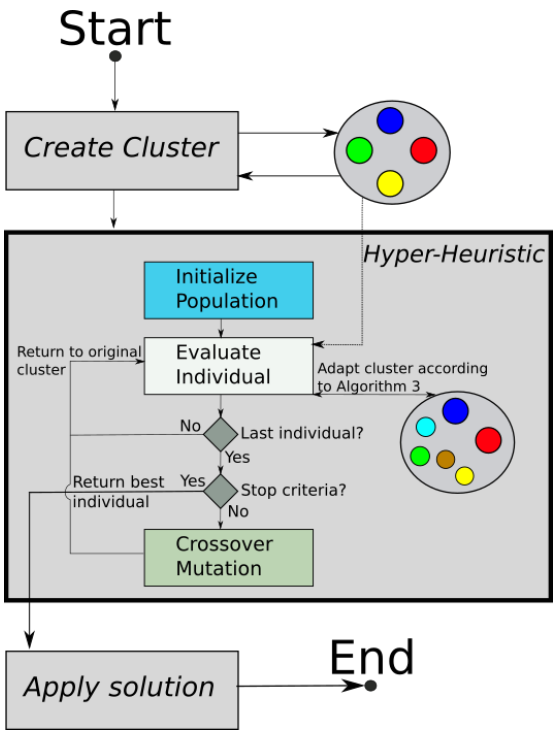


Fig. 3: Overall flow chart of the proposed framework.

Algorithm 2 Low-Level Heuristic Phase

1: **procedure** LLH(*Integer allele*, *Routes solution*, *List clusters*)
2: **repeat**
3: Limit search-space based on *allele* type
4: Call *allele* function from LLH database ▷ For example, if *allele* is Cross-Exchange cluster-based, invoke it.
5: **if** *allele is* cluster-based **then**
6: $UpdateClusters(solution, clusters)$
7: **until** No improvement

C. Genetic Component

The Genetic Algorithm (GA) utilized is specified in this section. The algorithm follows the common GA framework where there is a string type chromosome, a crossover is applied, as well as a mutation step according to some rate. Each initial individual is created as a random permutation that allows repetition of the LLH, with a fixed initial size but that can grow up to a δ **(delta)**, for example, if the initial size is 10 and δ is 5, the individuals can have anywhere from 10 to 15 alleles.

Algorithm 3 Cluster Update Phase

1: **procedure** UPDATECLUSTERS(*Routes solution*, *List clusters*)
2: **for each** *Route* $r \in$ *solution* **do**
3: $c \leftarrow ClusterWithMajority(r)$
4: **if** $c = 0$ **then**
5: $RemoveFromClusters(r)$
6: $AddNewCluster(clusters, r)$
7: **else**
8: $RemoveFromClusters(r)$
9: $AddInExistingCluster(clusters, c, r)$

Two crossover operators are selected here. The first one is the same utilized in [30], called best-best, where for each parent the best-improving sequence of the chromosome is passed on to the children as in Figure 4. The second one works very similarly, but instead of the best-improving sequence, the cut points are random, in a way to increase diversity. Each type has a 50% probability of being chosen. The top 10% individuals are saved for the next generation, while the remainder is composed by the resulting crossover of the current population. The parents' selection is done by random pairs, but one given parent can breed at most two times, being removed from the selection pool after the second time, to avoid oversampling of a given scheme. For the mutation step, similarly to the crossover, two methods are applied with a 65-35 chance, if the mutation occurs. The first one removes the worst improving sequence of a chromosome, called remove-worst, also from [30]. While the second randomly adds new LLH, up to one fifth ($\frac{1}{5}$) of the selected chromosome length, at random positions. Every individual in the population (after the crossover step) can suffer the mutation with a 10% chance. Additionally, and to avoid falling in a local optimum schema, half the population is reset (a new random permutation) every 20 continuous generations without improvement, the other half are the 50% best individuals.

Since each individual in the population will have a different order in which the LLH are applied, there is a second implicit diversification step (other than the cluster-based LLH mentioned above). To explain a bit further, with a different order in which the LLH are applied, then the changes in clusters and in routes will ripple into the next selected LLH. Because applying a cluster-based Cross-Exchange first into the initial solution and then applying a Two-Opt in each cluster will

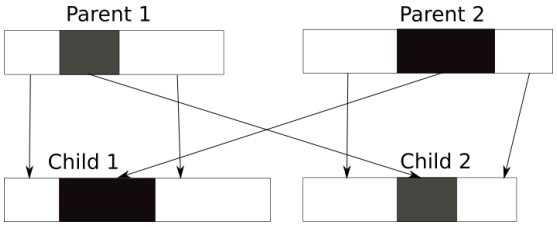


Fig. 4: Example of a best-best crossover, based on [30].

most likely result in a different set of routes and clusters than if doing the other way around (Two-Opt first and CE second). Therefore, the method explores the heuristic space while also exploring the solution space.

IV. Experiment Design

In this section, the experiment details are described. The algorithms compared in this experiment were coded in C++[1] with the library VRPH (created by [31]), from which the VRP basics and some of the implemented heuristics were directly reused. The computer in which the experiments were run on an Intel®Core™i7-8700 @ 3.2GHz and 15GB memory.

A. Experiment Goals

The objective of these experiments is to validate the idea of the Cluster-based Hyper-heuristics, specifically to answer the four objectives presented in Section I. For this purpose, four types of experiments were conducted, all considering the Large-Scale VRP:

- **Experiment 1**: the goal is to show the effects of changing the search space with the parameter M (the percentage of neighbour clusters which are considered in a search), by considering a larger or smaller size, and to compare with fixed clusters, or no clusters at all. This experiment was conducted by having three different instances being run varying the parameter M to 25% and 50% (0% would not consider inter-routes between clusters and therefore was not tested, and 100% is the same as no clusters since all are considered). Expectations are that with a larger number of neighbouring clusters, the solution will improve at the cost of higher computation time, and the exact opposite for a lower value. However, by how much, and what point has the best trade-off between quality and execution time?
- **Experiment 2**: this experiment investigates how much the starting clusters affect the solution. To achieve that, two approaches are chosen: the first one is a route-first cluster-second approach, where the K-means is used after the initial routes are created, this way the starting clusters will have multiple routes passing and the cluster evolution will happen accordingly; the second is a cluster-first route-second, where the K-means is run 30 times (since the clusters depend on the initial random points, they can have different outcomes) and the one having the best initial solution is chosen (creating the clusters first, will make the initial routes obey the cluster delimitation).
- **Experiment 3**: its shows the effects of different chromosome decoding approaches. The objective is to show by how much solution quality changes from a VNS-based type decoding to a single pass, and how is the execution time affected.
- **Experiment 4**: it compares the Cluster-based HH solutions to the best-known solution, as well as to a deterministic algorithm (CW) and to a manually designed combination of the same LLH. The objective of this experiment is to show whether the proposed method is competitive to existing methods, in both solution quality and execution time. This experiment utilizes the best configuration found by experiments 1 and 2, aiming to be the most competitive.

The results and discussions regarding these experiments are given in Section V. But first, the dataset and parameters are shown.

B. Dataset and Parameters

The instances utilized are a sub-set from the CVRPLIB dataset of [13], which ranges from 100 to 1000 customers, varying both geographical distribution and vehicle capacity. The size of these instances meet the large-scale definition and have known optima or a good solution is known.

Unless otherwise mentioned in a given experiment, the parameters use the default values described in Table I.

Parameter Name	Value
M (for closest clusters)	25% of the number of clusters
K (for K-means)	10% of instance size
VNS Type	Explore Sequence
Generations	100
Population size	30
Minimum individual size + δ	$22 + 11$
Number of runs	10

TABLE I: Default parameters.

V. Results and Discussion

This section discusses the results of the experiments described in Section IV-A.

A. Experiment 1

This experiment considers the effect of changing the search space, as well as comparing it to its counterparts: not using the cluster limit on the search, and not updating the cluster as the proposed method. Table II presents the results, showing for each case the average solution with standard deviation, the best result and the average execution time in minutes. As expected, the cases in which the search space includes more clusters (such as No Clusters and 50%) take longer, since the number of evaluations needed is significantly higher, while the quality increase is not much different. The fixed clusters also have a worse performance overall, since the search space is likely larger than the proposed approach, and it avoids different search areas, searching the same clusters throughout execution.

[1] Version 11 compiled with g++ and optimisation flag -O2.

	No Cluster			Fixed Cluster (25%)			CbHH (25%)			CbHH (50%)		
Instance	Avg(s.d.)	Best	Time	Avg(s.d.)	Best	Time	Avg(s.d.)	Best	Time	Avg(s.d.)	Best	Time
X-n200-k36	60898(429)	60537	216.6	61024(460.22)	60537	126.84	60996.89(194.96)	60600	6.53	60826.50(368.22)	60543	40.84
X-n251-k28	40401.5(96.62)	40266	250.68	40298.15(173.65)	39847	47.51	40339(129.52)	39974	8.48	40428.41(146.38)	40195	73.83
X-n308-k13	27580(155.32)	27361	1032.72	27736.65(187.08)	27343	170.40	27973.73(386.11)	27205	33.66	27353.5(160.12)	27016	297.8 3

TABLE II: Results for Experiment 1: the effectiveness of the proposed method when changing the search space size. The average solution, as well as the standard deviation and best solution, are shown, followed by execution time in minutes.

B. Experiment 2

In this experiment, the starting solution is explored. As shown in Table III, the route-first solution is worse than its counterpart. One possible explanation is that the initial Savings algorithm will make the clusters adapt to its routes and reach a local optimum much faster. In fact, all converge to the same solution. While for the cluster-first the initial routes are freer to be updated.

	Route-First	Cluster-First
Instance	Avg(s.d)	Avg(s.d)
X-n200-k36	60993(0)	60996.89(194.96)
X-n251-k28	40406(0)	40339(129.52)
X-n308-k13	28242(0)	27973.73(386.11)

TABLE III: Results for Experiment 2: a comparison between both methods of initialization. The solution average and standard deviation are shown.

C. Experiment 3

This experiment investigates the difference in time and quality based on how the chromosome is decoded. Table IV shows the effects of different decoding of the chromosome on quality and performance for the given instances'. The VNS is expected to have a better result since it explores the search space more heavily, at the cost of more time. However, the difference is much less than expected, with less than 2% between average solutions, finding the best solution in all test cases. The extra computational cost is especially noticeable in smaller instances.

	Explore Sequence		VNS	
Instance	Avg(s.d)	Time	Avg(s.d)	Time
X-n200-k36	60996.89(194.96)	6.53	61034(358.25)	33.91
X-n251-k28	40339(129.52)	8.48	40277.88(166.18)	15.48
X-n308-k13	27973.73(386.11)	33.66	27435.82(117.88)	35.49

TABLE IV: Results for Experiment 3: a comparison between the two types of decoding for the chromosome. Showing the average, standard deviation and execution time in minutes.

D. Experiment 4

This final experiment compares the proposed method with the best-known solution [2], a manually designed combination of the LLH and to the Savings algorithm. The manually designed version is based on creating chromosomes of fixed length which uses all the selected LLH (therefore of length 11), chosen in random order, and are not put through the GA process (all other parameters are the same as in I). This aim to simulate the process of selecting the LLH and using them in some specific order, which is commonly done in heuristics design. Table V summarizes the comparison. The proposed method performs better than the other two and is within 5% of the best-known solution for all instances.

Overall the results seem promising in showing that having the adaptive clusters improve the search speed, without much loss of quality. It performs better than some of the compared cases. We performed a Wicolxon Ran-Sum test to verify the statistical significance of this performance comparing the base case with the tested variants: the fixed clusters, regarding parameter M, cluster-first and decoding type VNS. The results, as seen in Table VI, show that the method is significantly better for most cases.

[2]Taken from the CVRPLIB website (http://vrp.galgos.inf.puc-rio.br/index.php/en/)

VI. Conclusions and Future Work

In this paper, an adaptive clustering technique based on solution evolution in combination with a HH framework is proposed, aiming Large-Scale Vehicle Routing Problems. The **4** experiments realized aimed in showing the effectiveness of both the clustering technique and its combination with a Genetic Algorithm for selecting low-level heuristics, as well to compare it to other methods. The results show that the clustering technique significantly improves the performance when compared to scenarios with no search limits, and with fixed clusters. Therefore, the proposed technique finds its purpose of limiting the search space more efficiently. Additionally, even though the evolution process can still be considered slow when compared to other meta-heuristics, this can be done offline, and the application of the resulting chromosome takes very little time (less than a minute for all instances).

Future work will investigate whether evolving the algorithm for multiple instances could lead to a chromosome applicable to more cases. Expanding the experiments to compare to manually designed limits on the search space is also considered as a next step. Another possible study concerns the automatic evolution of the parameter M based on the current search space, adaptively changing the search size according to recent progress.

References

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, Oct. 1959.

[2] G. Laporte, "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, Nov. 2009.

[3] K. Braekers, K. Ramaekers, and I. V. Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300–313, Sep. 2016.

		Manually Designed			Savings-CW			Cluster-based HH (25%)		
Instance	Best Known Solution	Best	Avg(s.d)	GAP	Best	Avg(s.d)	GAP	Best	Avg(s.d)	GAP
X-n200-k36	58578*	61220	61892.17(437.72)	4.5%	61167	61167(0)	4.4%	60600	60996.89(194.96)	3.4%
X-n251-k28	38684*	40846	41096.2 (250.2)	5.5%	40576	40576(0)	4.9%	39974	40339(129.52)	3.3%
X-n308-k13	25859	27970	28106.4 (136)	8%	28555	28555(0)	10.4%	27205	27973.73(386.11)	5%

TABLE V: Results for Experiment 4: comparison between different BKS and different algorithms with the proposed method. The best, the average with standard deviation, as well as the GAP from the best solution to the BKS are shown. BKS with (*) indicate that they are the optimal.

	Fixed Cluster(25%)	CbHH50	Cluster-First	VNS
X-n200-k36	0.27	0.01	0.82	0.55
X-n251-k28	0.24	0.04	0.06	0.06
X-n308-k13	0.006	0.00	0.01	0.00

TABLE VI: Wicolxon Rank-Sum test: p-values of the comparison between the base method (CbHH 25%) and the other evaluated variations.

[4] F. Arnold, M. Gendreau, and K. Sörensen, "Efficiently solving very large-scale routing problems," *Computers & Operations Research*, vol. 107, pp. 32–42, Jul. 2019.

[5] Komarudin and A. Chandra, "Optimization of very large scale capacitated vehicle routing problems," in *Proceedings of the 2019 5th International Conference on Industrial and Business Engineering - ICIBE 2019*. ACM Press, 2019.

[6] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, Dec. 2013.

[7] K. Sim and E. Hart, "A combined generative and selective hyper-heuristic for the vehicle routing problem," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16*. ACM Press, 2016.

[8] H. Liu, Z. Zhang, and X. Guo, "Restricted neighborhood search for large scale vehicle routing problems," in *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, May 2019.

[9] F. Arnold and K. Sörensen, "What makes a VRP solution good? the generation of problem-specific knowledge for heuristics," *Computers & Operations Research*, vol. 106, pp. 280–288, Jun. 2019.

[10] M. Gendreau and C. D. Tarantilis, *Solving large-scale vehicle routing problems with time windows: The state-of-the-art*. Cirrelt Montreal, 2010.

[11] M. Huang and X. Hu, "Large scale vehicle routing problem: An overview of algorithms and an intelligent procedure," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 8, pp. 5809–5819, 2012.

[12] N. R. Sabar, X. J. Zhang, and A. Song, "A math-hyper-heuristic approach for large-scale vehicle routing problems with time windows," in *2015 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, May 2015.

[13] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 257, no. 3, pp. 845–858, Mar. 2017.

[14] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A unified solution framework for multi-attribute vehicle routing problems," *European Journal of Operational Research*, vol. 234, no. 3, pp. 658–673, May 2014.

[15] A. Subramanian, E. Uchoa, and L. S. Ochi, "A hybrid algorithm for a class of vehicle routing problems," *Computers & Operations Research*, vol. 40, no. 10, pp. 2519–2531, Oct. 2013.

[16] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2000, pp. 176–190.

[17] N. Pillay and R. Qu, *Hyper-Heuristics: Theory and Applications*. Springer International Publishing, 2018.

[18] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational intelligence*. Springer, 2009, pp. 177–201.

[19] G. Ochoa, M. Hyde, T. Curtois, J. A. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. J. Parkes, S. Petrovic, and E. K. Burke, "Hyflex: A benchmark framework for cross-domain heuristic search," in *Evolutionary Computation in Combinatorial Optimization*. Springer Berlin Heidelberg, 2012, pp. 136–147.

[20] M. Mısır, K. Verbeeck, P. D. Causmaecker, and G. V. Berghe, "A new hyper-heuristic as a general problem solver: an implementation in HyFlex," *Journal of Scheduling*, vol. 16, no. 3, pp. 291–311, Nov. 2012.

[21] J. D. Walker, G. Ochoa, M. Gendreau, and E. K. Burke, "Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework," in *International conference on learning and intelligent optimization*. Springer, 2012, pp. 265–276.

[22] P. Garrido and M. C. Riff, "DVRP: a hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic," *Journal of Heuristics*, vol. 16, no. 6, pp. 795–834, Feb. 2010.

[23] J. Jacobsen-Grocott, Y. Mei, G. Chen, and M. Zhang, "Evolving heuristics for dynamic vehicle routing with time windows using genetic programming," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 1948–1955.

[24] B. Chen, R. Qu, R. Bai, and W. Laesanklang, "A hyper-heuristic with two guidance indicators for bi-objective mixed-shift vehicle routing problem with time windows," *Applied Intelligence*, vol. 48, no. 12, pp. 4937–4959, Aug. 2018.

[25] R. J. Marshall, M. Johnston, and M. Zhang, "Hyper-heuristics, grammatical evolution and the capacitated vehicle routing problem," in *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp '14*. ACM Press, 2014.

[26] D. B. Coral, M. O. Santos, C. Toledo, and L. F. Niño, "Clustering-based search in a memetic algorithm for the vehicle routing problem with time windows," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[27] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, Oct. 2000.

[28] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, Jun. 2010.

[29] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations research*, vol. 12, no. 4, pp. 568–581, 1964.

[30] L. Han, G. Kendall, and P. Cowling, "An adaptive length chromosome hyper-heuristic genetic algorithm for a trainer scheduling problem," in *Recent Advances In Simulated Evolution And Learning*. World Scientific, 2004, pp. 506–525.

[31] C. Groër, B. Golden, and E. Wasil, "A library of local search heuristics for the vehicle routing problem," *Mathematical Programming Computation*, vol. 2, no. 2, pp. 79–101, Apr. 2010.