Adaptive Search Space through Evolutionary Hyper-Heuristics for the 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*—Amid continuing increases in purchasing power and Internet shopping, demands for more efficient algorithms to solve the Large-Scale Vehicle Routing Problems are to be expected. The problem, which consists of finding routes for visiting customers, however, is very hard to solve due to its combinatorial nature. This becomes more notable when looking at large scale instances, i.e at least 200 customers, requiring sophisticated solution methods to solve, mostly heuristic-based. The existing meta-heuristics apply some limits to the solution space for handling the large scale nature. However, this is typically done manually, based on extensive domain knowledge. Hyper-Heuristics aim to reduce domain knowledge dependence, but current works disregard search space size, presenting poor scalability. In this paper, we propose a new Hyper-Heuristic to automatically evolve effective heuristics, based on a Genetic Algorithm framework. The new Hyper-Heuristic adds limits to the neighbourhoods of Low-Level Heuristics. In the method, the large search space is dealt with an adaptive chromosome which limits the neighbourhood size for each operator. The goal is to show whether more effective limits can be found if done automatically. Results show that the automatically evolved neighbourhood size limits outperform fixed ones for most instances tested and manually designed limits.**

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

I. Introduction

The Vehicle Routing Problem (VRP) is a well-known optimisation problem which has many applications in the real world. Although the VRP have been largely studied, its NP-hard nature [1] makes it difficult to find good solutions for many cases, meaning that new studies are still required and considered relevant. One of the main difficulties that recent literature is facing relates to the scale of the problem instances. Also known as the Large-Scale VRP (LSVRP), instances with more than 200 customers are considered a special case which requires a different treatment when looking for solutions, since the exponential growth in search space becomes much harder to deal with [2].

Current methods for the LSVRP are based on either exact or heuristics approaches. The exact methods scales badly, failing to reliably solve instances with more than 200 customers [3]. The heuristics and meta-heuristics have achieved very good results recently, such as [4], but these methods are mostly manually designed with extensive domain knowledge, focusing on the specific datasets they solve. For example, in [4], the authors limit the search space on one of the inter-route heuristics used as to the 3 closest routes, which although produced good results, it is unclear whether is the best option, or even if the chosen neighbourhood is the most efficient for those instances. Moreover, it is unknown whether the existing manually designed heuristics can be generalised to different datasets. When facing an unseen instance where the current best heuristic does not perform well, it is hard to manually design a specific effective heuristic again, as it highly relies on human expertise.

Hyper-Heuristics (HHs) address this issue by automatically designing heuristics rather than manually designing them for each case. The HHs are used for solving hard search problems by using a pool of Low-Level Heuristics (LLH) [5]. Then, a high-level strategy combines the LLHs to achieve generalisable results. In other words, the HH will select from a pool of heuristics or generate a new heuristic based on the instance being solved. As they search the heuristic space rather than the solution space, a HH can achieve better (or at least the same) results than if an expert were to make such choice, by exploring possibilities that were not considered by an expert. The problem of selecting the best heuristics for the given instance becomes a search problem in itself. Evolutionary HHs utilize Evolutionary Computation techniques to find the best combination of LLH for solving each problem.

The HHs have not been extensively used for solving the LSVRP, therefore literature lacks more studies regarding this. One noticeable recurrence in those HH works that aim VRP, however, is that they disregard the search space size from the LLH. Although they are mostly polynomial with lower magnitude, the extensive use of LLH for larger scales will likely lead to a high computational cost. It would be interesting to utilise HHs to automatically decide these limits, rather than have manually determined.

This study proposes a Hyper-Heuristic approach which considers automatically limiting the neighbourhood size generated by the LLH, based on an evolutionary process. The idea is to have each LLH have its own neighbourhood size (i.e. subset

of neighbouring customers to be examined) adaptively evolved according to its effectiveness in the evolution, which leads to an adaptive search space for each LLH. The goal of this paper is to investigate whether HHs could automatically find an adaptive neighbourhood size strategy that can successfully find quality results, especially when compared to fixed ones.

The rest of this paper is organised as follows: we give a more formal definition of the problem and the related work in Section II. Then our hyper-heuristic approach is detailed in Section III. The experiment design is then presented in Section IV, followed by a discussion on the results in Section V. Finally, we finish the paper with some conclusions and future research in Section VI.

II. Background

A. Capacitated Vehicle Routing Problem

The Vehicle Routing Problem was first formally defined by Dantzig and Ramser in [6], where a limited fleet of trucks needed to serve a set of customers, without exceeding the trucks' capacity. This became known as the Capacitated VRP (CVRP) and is one of the most studied combinatorial optimisation problems.

A formal definition for the CVRP can be expressed, as shown in [7], by a graph $G = (V, A)$. Where $V = \{0, 1, ..., n\}$ represents the set of vertex(or nodes) in the graph, while $A = \{(i, j), i, j \in V, i \neq j\}$ represent the set of arcs (or edges if symmetrical) which connect the nodes. In other words, the vertices represent the customers, with 0 being the depot, and the edges are the paths between them. Each customer has a demand $q_i \geq 0$ which must be served by a single vehicle, available from a fleet, each with capacity Q. The goal is to minimize the total distance traversed by the vehicles, and the following constraints need to be satisfied:

- Each customers are served by exactly one vehicle without interruption;
- Each vehicle departs from the depot, and return to the depot at the end;
- The total demand served by a vehicle cannot exceed the capacity of the vehicle.

For this paper, we are focusing on the Large-Scale version of the CVRP, where the number of customer are at least 200 [2], i.e $|V| \geq 200$. The challenges of a large scale VRP include not only the exponential number of solutions, but also the additional resources computational-wise. This extra load have led to methods which try to find ways to couple with the scale by adding manually decided or weakly proven limits to the search space.

B. Related Work

Considering the most recent approaches for the VRP and its variants, several have found success in finding good solutions for their respective datasets. For the Large-Scale VRP (LSVRP), the methods have shown some mixed results, since it is way harder to balance the exploration and exploitation concepts on larger search spaces. For an instance to be considered large-scale, it needs to have at least 200 customers [2] [8] [9] [4]. For other methods and an overall view of the VRP, we suggest the following reviews [7] [2] [10].

The high complexity of the VRP requires sophisticated methods to find good solutions. The Hybrid Genetic Unified Framework from Vidal et al [11] utilises a well elaborated Genetic Algorithm which carefully selects how to keep a good population of solution and applies a Local Search to improve each generation. The Iterated Local Search from Subramanian et al [12], applies an exact method in combination with Local Search to continuously improve a solution. Both have set several best known solutions across different variants, being known good and reliable methods. On the other hand, as shown in [4], both these suffer with scalability, taking hours of execution time to solve instances of larger size, of 600 customers and more. These and the most efficient VRP methods follow a simple rule of applying Local Search to improve their solutions, and has been shown to be the cornerstone of effective VRP approaches [13], [14]. Local Search is a type of heuristic approach which explores solutions in neighbourhoods starting from an initial solution, and is a robust approach that allows the achievement of high-level solutions [15]. These neighbourhoods for the VRP can be divided into the intra-route (moves that explore each route individually) and inter-route (moves that consider different routes simultaneously).

The methods that tackle the LSVRP, however, face an issue when applying Local Search for such scales due to the large number of neighbouring solutions. Therefore, they need to reduce the huge number of possible solution in someway. Most methods deal with this size by setting some kind of limit to the search space, avoiding exploring all possible moves for a given neighbourhood. For example, [4] propose a Guided Local Search that can find very good solutions for up to 30000 customers, having good scalability and execution time. They selected a few but powerful neighbourhoods and set limits to how much each of them explore, as in 30 closest customers. Other methods also apply some sort of limit, whether by clustering or some threshold as reviewed in [8], and are essential for achieving a good balance between time and quality. These limits are not trivial to be designed effectively, and are often also instance dependent. Usually they are set manually or experimentally requiring much human expertise, and are fixed throughout the whole algorithm run. This means that the chosen limits might be subpar if the same value is considered for different scenarios.

To investigate how to automatically design these limits, a Hyper-Heuristic (HH) seemed like an ideal approach. The idea of a Hyper-Heuristics is to automatically design heuristics that solve hard computational search problems [5]. They aim to reduce problem dependency as a more generic approach that can find solutions of acceptable quality, achieving that by using easy-to-implement components (low-level heuristics, LLH). This concept could be used in the search for the mentioned limits. There are mainly two types of HH when looking at the nature of the heuristic search space, according to [5]: Selection and Generation approaches, where each can be divided based on the nature of the components they use, either construction

or perturbation blocks. Evolutionary Hyper-Heuristics are the ones that utilize Evolutionary Computation algorithms as the framework. Although the most known are based on Genetic Programming, as in [16], there are also some work based on Genetic Algorithm, such as [17]. For more on Hyper-Heuristics we point to the works of [5] and [18].

Although HHs have been applied to the VRP, such as in [19] [20] [21] [22] [23], very few have experienced with Large-Scale. For example, in [9] the authors propose a two-phase algorithm for the VRP with Time-Windows (VRPTW), where the set of customers are subdivided for several column generation problems, which are then combined and fed as a start solution for a selection HH. Since exact approaches are known to be non-scalable, this method will always depend on how this division and column generation are applied. Consequently, the method would need to be manually adjusted for other instances.

In [24] we explore the clustering of customers which adaptively change based on the heuristics and routes evolution, finding solutions with similar quality when compared to the traditional fixed clusters, but with a much better computational time. This present work is similar to that in the fact that it tries to adaptively change the search space. However, here in this paper we focus more on how to limit each neighbourhood used rather than the customer search space. Similarly, our proposed method uses a Genetic Algorithm-based Hyper-Heuristic to have the automatic design of the heuristic, and also to automatically set limits to the search space for each neighbourhood. In the next section we specify how the framework is organized.

III. The Proposed Method

Our method utilizes an Evolutionary Hyper-Heuristic framework based on a Genetic Algorithm to evolve the selection of LLH available to the best order, up to a fixed number of generations. But before going into more details, we present the solution representation.

A. Solution Representation

The solution chromosome utilizes a two-dimensional array with variable length, where the first level is an integer ID which represents a given LLH, and the second level is the percentage of the closest neighbours that the LLH will search for. Even though all alleles have the two levels, the search space limit is only applied for inter-route neighbourhoods. This is because a intra-route operator will only perform the search within the same route, which is usually a very short number of customers due to the capacity constraints. In inter-route operators, however, all other customers can potentially be searched for a move. In a large scale problem the number of customers outside the route can be quite large, making it a very expensive evaluation, which is why we want to limit only these operators. The limits are still kept for all alleles because crossover operations can alter the structure of the alleles in such a way that an intra-route allele might affect an inter-route one. These operators are further explained later.

For the utilized LLH, we have selected a set of intra and inter perturbative operators. Some are very frequently used and historically show good improvements, such as Two-Opt and Cross Exchange. But more importantly, they are all relatively easy and straightforward to implement and use, which is one of the arguments of using Hyper-Heuristics [5]. The full list of LLH and how they work are:

- Two-Opt: remove two edges from a route and relink them in a different way. It is given the ID 0 in the individual representation.
- Or-Opt: transfers a sub-string of $2, 3$ or 4 consecutive customers to another position in the same route. It is given the ID 1 in the individual representation.
- Cross Exchange: a sub-string is swapped between two routes. It is given the ID 2 in the individual representation.
- Two-Opt*: remove two edges from distinct routes and relink them in a different way. It is given the ID 3 in the individual representation.
- Or-Opt*: transfers a sub-string of $2, 3$ or 4 consecutive customers to another position in a distinct route. It is given the ID 4 in the individual representation.
- Merge: combines two different routes into one, by concatenating one into any position of the other. It is given the ID 5 in the individual representation.

Figure 1 is an example of the representation. As the ID 3 in the first position of the first dimension represents a two-opt* move, then it will consider the closest 50% of all customers to perform the move; the next ID 3 represents the same operator, but now only considers 30% of these customers.

3	1	5	2	0	4	1	5	0	3
50%	30%	20%	20%	50%	20%	10%	40%	10%	30%

Fig. 1. An example of chromosome for the Genetic Algorithm.

B. Evolutionary Hyper-Heuristic

The hyper-heuristic framework has a evolution process where each individual, represented by the above mentioned two-dimensional array, will apply its sequence of LLH to a single starting solution. Algorithm 1 summarizes the process. First an initial solution is created heuristically (line 2), using either the Savings heuristics [25] or a simple round-way trip for each costumer, both cases are tested to see which performs better, and the results will be shown in Section V. Next, the initial population of heuristics, each represented as a LLH sequence, is created at random (line 3). In an individual, each LLH is associated with the percentage of closest neighbours it considers. The main loop starts (line 5) and each individual will be evaluated by how much they improve the initial solution. Each allele and its correspondent LLH will modify the current solution if that improves it (lines 8-9), i.e, the acceptance criteria is Only Improvement, as shown in Algorithm 2. If the solution of the current individual is the best found, we save or update that individual (lines

10-12). Finally, if the stopping criteria are not met (line 13), the crossover and mutation operators are called to evolve the population (line 14). The algorithm returns the result given when solving the instance with the best individual (line 16).

Algorithm 1 Adaptive-based Selection Perturbative Hyper-Heuristic

```
1: procedure ABSPHH(Dataset instance, List LLH)
2:     initial_solution ← FindInitialSolution()
3:     population ← InitializePopulation()
4:     best ← ∅
5:     loop
6:         for each individual ∈ population do
7:             new_s ← initial_solution
8:             for each allele ∈ individual do
9:                 new_s ← LLH(allele, new_s)
10:            Eval(new_s)
11:            if Improvement then
12:                best ← individual
13:        if Stopping Criteria not met then
14:            new_population ←top 10% individuals
15:            while new_population ≠ Full do
16:                Select 2 parents at random from population
17:                new_individuals ← Crossover selected parents with one of the two operators
18:                Mutation of new_individuals according to probability P_M with one of the two operators
19:                new_population ← new_individuals
20:            population ← new_population
21:        else
22:            return best(instance)
```

The evolutionary operators are responsible for improving the individuals over the generations. They will diversify the population and try to exploit the best individuals characteristics. Before this process, however, elitism is applied to the top 10% best individuals, which are kept for the next generation. The remainder of the population is replaced with the children. The parents selection occurs by random chance, but no individual can be chosen more than twice.

For crossover, two operators were chosen aiming to create different diversification aspects, each with the same probability of happening. The crossover operators were chosen in order to maintain promising schemes in the population, but also allows for some diversification to find new ones. The first operator is the classical two-point crossover, where two points are chosen at random, and the middle portion of each parent is swapped between both to generate the offspring. For this operator, the search space limit at the second dimension of the resulting array will be updated according to the average between the parents' limits, considering the same positions, as shown in Figure 2. If a child has a larger size, the original values are kept. This operator is mainly used to produce some diversity in the population by considering new neighbourhood limits and by not considering the solution quality.

The second operator is the Best-sequence crossover, based on [26]. The best improving sequence from both parents, i.e. the largest subset of consecutive operators that contribute the most for the solution, are inherited by the children. For example, given an individual with 5 operators that has been evaluated, if the sequence of improvements for the solution was $[3\%, 4\%, 5\%, 3\%, 2\%]$, then the largest improving sequence would be $[3\%, 4\%, 5\%]$. It works similar to the first operator, but the average is not applied. Instead, the sequences are copied exactly the same, since its goal is to keep the good parts of the solution.

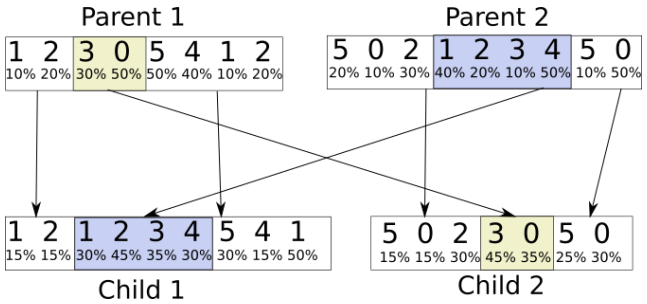

Fig. 2. Example of the two-point crossover applied to the two-dimensional chromosome.

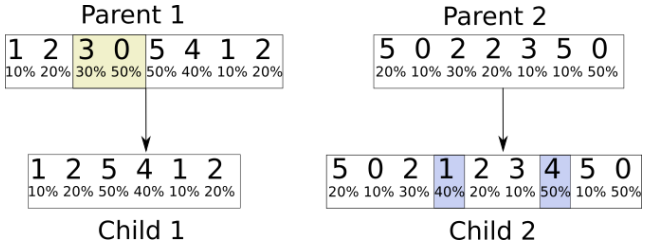

Fig. 3. Example of both mutation operators used. On the left, the worst sequence is removed. On the right, new random LLH are added to the chromosome.

Algorithm 2 Low-Level Heuristic Phase

```
1: procedure LLH(Array allele, Routes solution)
2:     repeat
3:         Limit search-space based on allele[1]
4:         Call allele[0] function from LLH database, update solution accordingly  ▷ For example, if allele[0] is Cross-Exchange invoke it.
5:         solution ← correspondentLLH()
6:     until No improvement
7: return solution
```

Mutation has a probablity $P_M = 10\%$ chance of happening for each individual of the new population, and also has two operators with equal chance of being chosen. The first one, also presented in [26], removes the worst performing consecutive sequence of operators from the individual which do not contribute to the solution. For example, considering the sequence $[3\%, 0\%, 5\%, 0\%, 0\%, 6\%]$, by removing the worst

improving sequence the result would be $[3\%, 0\%, 5\%, 6\%]$. The second one adds new random operators from the LLH pool as if newly created, which can vary between 1 and up to 20% of the length of the chromosome, and can be added in any place at random. These mutation operators aim to remove the bad performing schemes from the population and to add new elements for diversification. The two operators are illustrated in Figure 3.

Note that the order of the operators will result in different final solutions. But also the limits play a role in this, since even having the same operators sequence, the search limits may be different and will result in a distinct solution.

IV. Experiment Design

The experiments were designed aiming to test the method's ability to limit the search space. However the evolution can be sensitive to the initial solution and a test was designed to measure this. We also want to show whether there is a trade-off between the size of the limits and the quality of the final solution. We designed three main experiments for this preliminary work, which are detailed next.

Experiment 1 measures the impact of different initial solutions on the performance of the HH approach. As traditional perturbation operators might not escape local minimums too easily, the initial solution can significantly change the outcome. We test an obvious feasible approach which is a round trip between the depot and all customers, and compare it with the traditional Savings from Clarke and Wright [25]. The Savings algorithm is a deterministic constructive heuristic which starts building routes by connecting different round trips, whose connection will bring the most savings. This test will also identify wheter the chosen simplistic LLHs are enough to produce sufficient improvement without any local minima escape mechanism given the two different starting points.

Experiment 2 aims to evaluate both the efficiency and performance of the adaptive limits to the search space when compared to a fixed limit. To achieve that, we apply the same Hyper-Heuristics framework but set a fixed limit to the neighbourhood search. The fixed limit used as rule of thumb has all alleles set to 25%. On the other hand, the adaptive limits are randomly chosen for each LLH, ranging from 10% to 50%. For this experiment we assume that the full neighbourhood (i.e, 100% size), would not find results a lot better given the extra amount of time necessary to do so. Hence this experiment aim to show how having a fixed neighbourhood size (like most works do) compares to having different limits throughout execution. Since we have only a few set of LLHs and are comparing this fixed size based on our own framework, all claims made here would be true at least considering the tested circumstances. But at least if finding positive results we can show that this idea is worth investigating. For example, with this experiment we can analyse whether our Hyper-Heuristics would always give priority to the LLHs which have a larger neighbourhood, which could be expected since they will cover more possibilities.

Finally, **Experiment 3** compares the proposed method with two manually designed heuristics, as well as the best known solutions. Two manually designed heuristics, which use the limits to the search space as proposed in this paper, were created based on intuition from the authors. Both share the same work process, listing all operators used in our LLH pool twice. For example, with the six operators used numbered from 1 to 6, the manually designed heuristic would be [1,2,3,4,5,6,1,2,3,4,5,6]. Then, we apply it to the initial solution generated by the Savings heuristic [25] in this fixed order. The difference between the two manually designed heuristic are obtained by changing the neighbourhood limits between the two groups of operators. The first heuristic simulates a top-down approach, where we apply broader (50%) search spaces to the first group of operators and narrower (10%) to the second. The second manually designed heuristic is the bottom-up approach, which is the exact opposite of the previous one, applying the narrower size for the first group of operators, and broader size for the second. We will use **MHTD** and **MHBU** when referring to each heuristic, respectively, in the next sections.

The program was coded in C++[1] with the VRPH library [27], from which the VRP structure and some of the implemented operators were used. The experiments were run on a Intel®Core™i7-8700 @ 3.2GHz and 15GB memory. Table I shows the parameters of the Genetic Algorithm. Since the individual size has a variable length, the δ value indicates the upper bound for the initial population chromosomes. Additionally, in order to speed-up the process and escape local minima, if the GA finds no improvement for 10 consecutive generations, the lowest performing half of the population is replaced by randomly generated individuals, and if the lack of improvement persists for another 10 generations, the main loop is finished. Next we present and discuss the results.

TABLE I
Genetic Algorithm parameters.

Parameter Name	Value
Generations	100
Population size	30
Initial chromosome size + δ	$18 + 9$
Crossover rate	Every generation
Mutation rate	10%
Elitism	10% top rated
Number of runs	30

V. Results and Discussions

All the experiments were conducted by using a sub-set of instances from the CVRPLIB proposed by Uchoa et al [28], which vary between distribution, size, demand and capacity.

A. Effect of initial solution

The first experiment aims to show the impacts of initial solution to the algorithm convergence. As shown in Table II,

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

both the Round-Trip and the Savings heuristic [25] were used as initial solution. To verify if the results are significant, a Wilcoxon rank-sum test was also performed comparing the two approaches. If the *p-value* is less or equal to 0.05 we say that the algorithm is statistically better when solving that instance. The instances in which there are significant advantage are marked in the table with (+). For this case, all results are significantly better, when considering Savings as the starting algorithm.

Analysing Table II, the impact of the two different starting solutions is more clearly noticeable when looking at the execution time. This is because Savings can provide a much better initial solution than Round Trip, making the fitness evaluation (improving the solution until convergence) much less time consuming. In addition, as can be noticed by the average and standard deviation, the Savings solution tends to converge to a single point or a very close one. This means most solutions reach the same local minima when considering the used LLH.

On the other hand, the round trip version does not converge to a single point and has a greater relative improvement. This behaviour allows us to investigate the use of the neighbourhoods limits more clearly, as it provides a longer range of possible solutions. Therefore, next tests shown will utilise this method for generating the initial solution.

B. Effectiveness of the adaptive size limit

The second experiment is focused on showing how the proposed chromosome and evolutionary strategy performs for different search space limits, given the same Hyper-Heuristic framework. Considering a fixed neighbourhood search limit of 25% and the adaptive one, which range from 10% to 50%. This experiment aim to test if they have significant difference regarding both quality and execution time. Since starting with the Savings heuristic lead to prematurely convergence (as seen in the previous experiment), this test will use Round Trip starting solutions.

Table III summarizes these results. Although the adaptive limit does not outperform the fixed one in every scenario, the larger instances show better results. We also compare the p-value given the Wilcoxon rank-sum statistical test. Like in the previous experiment, the (+) sign means that the results are significantly better. The (=) sign means that there is no clear advantage when using the approach and the (-) sign means it performs worse. As the table shows, the adaptive method outperforms the fixed one for most instances, especially in larger scenarios. Although there is a somewhat significant increase in time, we deem this result as encouraging to keep exploring adaptive-based limits. Additionally, the ones that the adaptive method do not outperform, can also be an indication that our evolutionary mechanisms could not find solutions with similar (or better) schemes of limits.

C. Comparison with other methods

The third experiment compares the proposed approach with the Best Known Solution[2] (BKS) and with the two manually designed heuristics specified in previous section, MHTD and MHBU. The Savings algorithm [25] solution is shown as a comparison point. The proposed approach, as does the manually designed heuristics, uses the Round-trip as initial solution, where each customer is served by a different vehicle. The adaptive search approach is described in Section III, and range of neighbourhood limits is between 10%-50%.

Table IV shows these results. When compared to BKS the proposed solutions are significantly close, specially considering our approach do not have local optima escape mechanisms. This indicates that the neighbourhood limits can be effective regarding solution quality. The better or similar performance by the Savings algorithm, however, proves that the starting point is far too bad to lead to better solutions given the chosen parameters and LLH. The results also show that the proposed method does have an overall better solution quality than the manually designed heuristics which have fixed search limits, outperforming them for all cases, even in the average case. Therefore, this suggests that having the different order and search sizes improves the search space exploration.

D. Further Analysis of the evolved solutions

In this paper, we want to show whether the automatic evolution of the search space size will give benefit to the overall search. Experiments 2 and 3 measure this more directly, while experiment 1 tries to show whether the initial solution is the only factor making the results seem like they are. As shown in Experiment 2, some of the fixed size search performed just as good as the proposed approach. However, this can be explained if we analyse the evolved heuristics. As shown in Table V, the evolved best individual for each case has an average of their search space limits around the 29%, meaning the fixed approach was actually a good approximation of this. Figure 4 shows one run's best individual for instance *n308-k13*.

This also gives insight on how larger neighbourhoods might not always translate to better solutions. Other work might use smaller neighbourhoods assuming they get a time benefit in exchange for some reduction of the overall quality, but these initial results actually tell otherwise. As these evolved limits barely do not converge to using the largest available limit, since we are only measuring quality. In fact, as already mentioned, on average all best individuals limits' are around 29%, when considering all non-fixed solutions. Additionally, on average only 29% of the chromosome have a limit of 40% or higher (up to 50%), considering the same best individuals. This means that the evolution process does not prefer larger neighbourhoods, and uses them only for around $1/3$ of the alleles. And around 35% of the chromosomes have neighbourhood sizes of equal to or less than 25%. Meaning the evolution

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

TABLE II
RESULTS COMPARING THE INITIAL SOLUTION IMPACT. EXECUTION TIME IN MINUTES

	Round Trip			CW Savings [25]		
Instance	Initial Solution	Avg(s.d)	Time	Initial Solution	Avg(s.d)	Time
n200-k36	295558	61205.2(124.66)	31.39	61167	60993(0)(+)	15.11
n251-k28	290890	41045.73(200.84)	55.82	40576	40385(0)(+)	22.29
n308-k13	410930	27453.33(129.41)	191.77	28555	28040.3(7.97)(+)	49.80
n351-k40	28240	28049.7(184.59)	200.54	27123	26978(0)(+)	52.20
n376-k94	558384	151778.77(430.00)	487.78	149659	148957.2(8.26)(+)	279.02
n401-k29	755820	69031.8(311.91)	250.22	68975	68666.1(84.26)(+)	90.81
n420-k130	318530	113619.77(332.58)	1002.74	112604	112286.63(29.26)(+)	517.06
n449-k29	704352	59424.9(285.00)	322.63	58614	58395(0)(+)	75.60

TABLE III
RESULTS COMPARING A FIXED SEARCH SPACE SIZE AND AN ADAPTIVE ONE. EXECUTION TIME GIVEN IS IN MINUTES

	Fixed size			Adaptive size		
Instance	Best	Avg(s.d)	Time	Best	Avg(s.d)	Time
n200-k36	**60659**	61270(304.61)	39.19	60959	61205.2(124.66)(=)	31.39
n251-k28	40560	41022.76(204.00)	62.34	**40512**	41045.73(200.84)(=)	55.82
n308-k13	**27041**	27331.26(102.96)	210.52	27150	27506.76(151.49)(-)	191.77
n351-k40	27748	28240.63(360.46)	187.08	**27641**	28049.7(184.59)(=)	200.54
n376-k94	**150591**	152397.93(1009.14)	502.00	150715	151778.76(430.00)(+)	487.78
n401-k29	68484	69303.18(403.75)	274.30	**68466**	69031.8(311.91)(+)	250.22
n420-k130	112976	114116.67(420.37)	731.18	**112963**	113619.77(332.58)(+)	1002.74
n449-k29	59075	59773.9(351.28)	270.56	**58783**	59424.9(285.00)(+)	322.63

TABLE IV
A COMPARISON BETWEEN BKS AND THE MANUALLY DESIGNED HEURISTICS WITH THE PROPOSED METHOD. THE GAP SHOWS HOW CLOSE TO THE BKS THE AVERAGE SOLUTION IS. BKS WITH (*) INDICATE THAT THEY ARE THE OPTIMAL. IN BOLD THE BEST SOLUTION FROM THE TEST ALGORITHMS. THE SAVINGS ALGORITHM FROM [25] (CW) IS SHOWN AS A POINT OF COMPARISON.

		MHTD		MHBU		CW [25]		Proposed Method		
Instance	BKS	Best	GAP	Best	GAP	Best	GAP	Best	Average(sd)	GAP
n200-k36	58578*	62306	6.3%	62067	5.9%	61167	4.4%	**60959**	61205.2(124.66)	4.4%
n251-k28	38684*	41787	8.0%	41960	8.4%	40576	4.8%	**40512**	41045.73(200.84)	6.1%
n308-k13	25859	29012	12.2%	28374	9.7%	28555	10.4%	**27150**	27506.76(151.49)	6.3%
n351-k40	25928	29253	12.8%	28263	9.0%	**27123**	4.4%	27641	28049.7(184.59)	8.1%
n376-k94	147713*	159458	7.9%	152638	3.3%	**149659**	1.3%	150715	151778.76(430.00)	2.7%
n401-k29	66187	73098	10.4%	70183	6.0%	68975	4.2%	**68466**	69031.8(311.91)	4.2%
n420-k130	107798*	115910	7.5%	115640	7.3%	**112604**	4.4%	112963	113619.77(332.58)	5.4%
n449-k29	55269	61982	12.1%	60376	9.2%	**58614**	6.0%	58783	59424.9(285.00)	7.5%

process is bringin a balance between larger and smaller sized neighbourhoods.

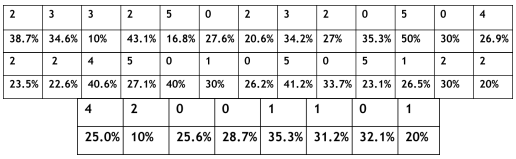

Fig. 4. Example of an evolved chromosome.

VI. CONCLUSIONS AND PERSPECTIVES

This paper presented a new preliminary Genetic Algorithm Hyper-Heuristic for solving the Large-Scale Vehicle Routing Problem. The method is based on a Genetic Algorithm with a two-level chromosome that limits the search space for each operator in an attempt to find more effective neighbourhood sizes. The experiments show that the method managed to achieve better solution quality than the compared manually designed approaches for all the test cases, and a fixed limit Hyper-Heuristic for most cases. They also successfully show that not only the order of which operator are applied impacts the solution, but also the sizes for each neighbourhood. Another finding regards the pattern among the evolved heuristics, which do not give preference to larger limits over smaller ones. This can indicate that the search limits actually are an effective way of improving performance of LSVRP algorithms, since they improve efficiency (when compared to full neighbourhood size) without loss of quality.

However, there are still some questions raised from looking at results. For example, since one of the possible permutations for the adaptive approach can be the same as the fixed one, then how did it not evolve for something similar? Perhaps the lack of more impactful Genetic operators, did not allow the adaptive search to find the same (or better) limits. New studies on the genetic components used will be considered to provide an evolution which at leasts can match the fixed approach. Some other future research can investigate if this

TABLE V
THE AVERAGE NEIGHBOURHOOD LIMIT FOR EACH LOW-LEVEL HEURISTIC AND THE AVERAGE IN TOTAL FOR EACH GIVEN INSTANCE.

Instance	Two-Opt	Or-Opt	Cross-Exchange	Two-Opt*	Or-Opt*	Merge	Total
n200-k36	30.08%	30.32%	29.96%	29.13%	30.46%	27.39%	29.52%
n251-k28	30.36%	29.60%	30.18%	30.50%	29.67%	28.93%	29.99%
n308-k13	30.54%	30.45%	29.32%	29.79%	31.07%	28.57%	29.93%
n351-k40	29.23%	30.10%	30.25%	29.51%	30.74%	26.03%	29.39%
n376-k94	28.92%	30.57%	29.89%	30.26%	30.78%	26.44%	29.46%
n401-k29	30.64%	30.34%	30.19%	29.49%	29.51%	26.85%	29.70%
n420-k130	29.78%	29.75%	30.59%	29.78%	32.41%	26.63%	29.62%
n449-k29	28.66%	29.91%	30.25%	30.51%	30.88%	27.40%	29.69%

type of pruning methodology is viable for non population-based algorithms, and whether they can have a better convergence speed. Additionally, seeing that the initial solution does have a large impact on the final solution, especially in computational cost, the Hyper-Heuristic could be adapted for also finding a better initial solution, considering different LLHs for creating routes. Finally, the current pool of LLH operators do not allow escaping of local optima and stronger algorithms, such as destroy and recreate, can be investigated on how they impact the results and how to limit the search space for such operators.

REFERENCES

[1] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems." *Networks*, vol. 11, no. 2, p. 221–227, 1981.

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

[3] D. Pecin, A. Pessoa, M. Poggi, and E. Uchoa, "Improved branch-cut-and-price for capacitated vehicle routing," *Mathematical Programming Computation*, vol. 9, no. 1, p. 61–100, jun 2016.

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

[5] 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, p. 1695–1724, Dec. 2013.

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

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

[8] 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, p. 5809–5819, 2012.

[9] 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.

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

[11] 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, p. 658–673, May 2014.

[12] 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, p. 2519–2531, Oct. 2013.

[13] F. Arnold and K. Sörensen, "Knowledge-guided local search for the vehicle routing problem," *Computers & Operations Research*, vol. 105, p. 32–46, may 2019.

[14] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "Heuristics for multi-attribute vehicle routing problems: A survey and synthesis," *European Journal of Operational Research*, vol. 231, no. 1, p. 1–21, nov 2013.

[15] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*. Princeton University Press, 2003. [Online]. Available: https://www.ebook.de/de/product/2790794/local_search_in_combinatorial_optimization.html

[16] F. Zhang, Y. Mei, and M. Zhang, "A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, p. 347–355.

[17] R. Raghavjee and N. Pillay, "A genetic algorithm selection perturbative hyper-heuristic for solving the school timetabling problem," *ORiON*, vol. 31, no. 1, p. 39–60, 2015.

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

[19] 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, p. 795–834, Feb. 2010.

[20] 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.

[21] 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, p. 1948–1955.

[22] 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.

[23] 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, p. 4937–4959, Aug. 2018.

[24] J. G. C. Costa, Y. Mei, and M. Zhang, "Cluster-based hyper-heuristic for the large-scale vehicle routing problem," in *IEEE Congress on Evolutionary Computation (CEC) 2020*. IEEE, 2020, to appear.

[25] 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, p. 568–581, 1964.

[26] 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, p. 506–525.

[27] 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, p. 79–101, Apr. 2010.

[28] 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, p. 845–858, Mar. 2017.