An Evolutionary Hyper-Heuristic Approach to the Large Scale Vehicle Routing Problem

Joao Guilherme Cavalcanti Costa, Yi Mei, Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600,
Wellington, 6140, New Zealand
{Joao.Costa, Yi.Mei, Mengjie.Zhang}@ecs.vuw.ac.nz

***Abstract*—The Large Scale Vehicle Routing Problem (LSVRP) is a classical combinatorial optimisation problem that serves several customers on a graph using a set of vehicles. Due to the NP-hardness and large problem size, LSVRP cannot be efficiently solved by exact approaches. Heuristic methods such as the Iterative Local Search or the Hybrid Genetic Algorithm still struggle for finding effective solutions for large scale instances. For these methods to deal with the large search space, pruning techniques are applied in order to limit the number of explored solutions. However, effective pruning is a hard task, requiring domain knowledge to craft good ways of limiting the search space without losing the ability to find better solutions. Hyper-heuristics are types of methods that aim to reduce domain knowledge on the creation of heuristics, and in this work, we also apply them for effective heuristic pruning. Our Evolutionary Hyper-Heuristic (EHH) automatically evolves limits to the solution search space together with the heuristic utilised to build and improve solutions for the LSVRP. We utilise a Guided Local Search (GLS) as the base algorithm in which our EHH searches for the best heuristic configuration. Our results show that the EHH can find better solutions for most LSVRP test instances when compared to the manually designed pruning of the GLS.**

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

I. Introduction

The Vehicle Routing Problem (VRP) [1] is being studied for years as it is highly applicable to real-world scenarios, such as the delivery of packages, mail, collection of garbage or other materials [2]–[4]. One important feature of these real-world applications, however, was not focused until more recently, which is the large number of customers (at least 200 [16]–[19]), or the Large-Scale Vehicle Routing Problem (LSVRP).

Recent developments for the VRP have been driven towards complex and domain-knowledge-powered approaches [5]–[7]. Although these approaches have been more and more effective and efficient, current state-of-the-art meta-heuristics need to apply several layers of domain knowledge/expertise to achieve good quality. This makes these algorithms hard to be replicable, time-consuming to design and very expensive due to this domain expertise. A well-known example can be traced back to the Travelling Salesman Problem's (TSP) Lin–Kernighan heuristic (LKH), also used for the VRP. The LKH is known for its non-trivial implementation [8], and is now used as one of the improvement steps for many algorithms, as in [5], [9].

On the other hand, Hyper-Heuristics (HHs) aim to make the development process easier by using heuristics to search heuristics rather than solutions [10]. The HHs have been applied for solving different problems successfully as shown in [10]. The goal of a HH is to find the optimal heuristic configuration, given a set of building blocks that solve a specific search problem, i.e. the HH will search for the best possible use of these blocks which will solve the problem. Therefore, a HH is theoretically able to find a configuration in the heuristic space which is better than (or at worst case similar to) a human-designed one, which is a more modern definition of Hyper-Heuristics as formalised by [11]. One of the most popular methods for HHs are based on Evolutionary Computation (EC) techniques. Probably the biggest advantage of EC techniques is their flexible representation scheme, which can be used to simplify the search space. A Genetic Algorithm (GA) HH's chromosome provide a way to improve the order in which operators are applied through its evolutionary process.

HHs face some challenges, however, when solving the LSVRP. For example, in order to deal with the larger search space domain knowledge is required for effective heuristic pruning and this is a contradiction to the HH idea. This layer of the method design has been relatively untouched. In the works [12] and [13] we started exploring the idea of letting the Hyper-Heuristics control the pruning of methods, but they lack a strong algorithm framework to properly show their efficacy.

To address the above issue, in this work we present a novel Evolutionary Hyper-heuristic (EHH) which is built on an effective Guided Local Search (GLS) framework. Our proposed approach applies an EHH to optimise the order in which simple operators are used to solve the LSVRP, reducing the pre-configuration required which is usually done by an expert. The evolutionary process tries to find the best neighbourhoods to be used and their sequence, as well as the solution construction method and local optima escape mechanism. Additionally, the EHH also evolves the heuristic pruning process through a new chromosome, letting the evolutionary process determine the best heuristic configuration.

II. Background

A. Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a combinatorial optimisation problem defined within a Graph G, where the Vertex set V represents the customers and depot ($V = \{0, 1, 2, ..., n\}$), with 0 typically being the depot. While the

Edge set E represent the paths between these customers ($E = \{(i, j) : i, j \in V, i \neq j\}$). The classical variant of the VRP has limited capacity on their vehicles. The goal of the VRP is to find the optimal routes which visit all customers once, respecting the capacity of each vehicle, with routes starting and ending at the depot. The classical optimisation task is to minimize the overall distance. For the full model, consider the work of [14]

The exact methods, usually based on mathematical formulations, will thoroughly search for the values of the variables to find the mathematical proven optimal. But the problem was proven to be NP-hard [15], and finding the best routes is a hard task. This is especially true for large problems. When it comes to instances with more than 200 customers, the problem is usually referred to as Large-Scale VRP (LSVRP) [16]–[19]. This larger size requires extra care regarding the amount of search effort given. Therefore, to efficiently solve the problem, there is a need for heuristic methods that do not find the optimal value, but find good solutions in a much faster time.

The heuristics used are usually based on simple moves which can be searched fast and repeatedly. These are known as neighbourhoods, perturbation heuristics, and local search, and they can be classified as intra-route (moves that explore one route at a time) and inter-route (moves that consider two or more routes simultaneously). Examples of such moves are the classics 2-Opt [20], Cross-Exchange [21], among others.

B. Related Work

The recent most effective and efficient methods for solving the VRP and variants use a Local Search (LS) based approach [22]. The quality of an LS heuristic is defined by the neighbourhoods utilised, ranging from the classic and simple moves to more elaborated ones. Starting from a single initial solution, the LS explores new solutions by making small moves, being robust across different problems and instances, as well able to find high-level solutions [23].

Works from [6] and [7] applied LS in combination with other techniques, such as Genetic Algorithm and Set Partitioning, respectively, to lead the solution or set of solutions towards different points from where the LS can find better solutions. However, as good as these methods are, they face some weakness when it comes to large scale. As shown in [19], they lose their ability to efficiently solve the problem within a few minutes, scaling to several hours. This can be explained by the fact that when applying LS to these scales, the large number of neighbouring solutions makes it too costly for the full search in each neighbourhood. To overcome this challenge, some LS-based methods apply some kind of heuristic pruning, reducing the number of solutions searched.

A recent case of success [5] can find solutions for up to 30000 customers within minutes of execution time, by considering move-specific pruning techniques. This method will be described in Section II-C. More methods apply limits to the search space, usually by grouping the customers or by some sort of threshold as reviewed in [17]. However, limiting the search space is not a trivial task, since if poorly done can avoid good solutions from being found.

Hyper-heuristics (HHs) aim to reduce the level of domain knowledge necessary to create a good heuristic. The HHs have been applied to automated heuristic sequencing, planning systems, parameter control (mostly in EC methods) and heuristic learning methods [10], with several cases of success. When learning heuristics, several factors need to be taken into account, such as the types of components to be considered, the techniques used and which parameters should this algorithm use. One popular approach for building HHs are the EC techniques, such as Genetic Algorithm (GA) and Genetic Programming (GP). GA has been applied for several search problems including searching for optimal heuristic sequencing, such as [24] for the bin-packing problem. GP is more used for creating a heuristic rule which builds a solution [25], rather than improving it, such as in [26] for the Dynamic Job-Shop Scheduling. We use GA in this work to evolve the order of the improvement operators used, which is described in Section III. More on Hyper-heuristics can be found on the review of [10] and the book of [11].

For large-scale problems (focusing on the VRP and variants), however, HHs have not been well explored. One example of HH being applied to an LSVRP with Time Windows can be seen in [18], where the large problem size is handled before the solution is fed to the HH. The approach reduces the search space by decomposing the problem and solving them with a column generation technique. This example shows how HHs rely on other methods for reducing the search space.

More recently, we presented a number of works considering the pruning search space as part of the HHs. In [13] a clustering technique which is guided by the HH is presented. We show that the clusters evolved by the HH are more efficient when looking for solutions when compared to the traditional fixed clusters. In [12], we present another form of controlling the pruning space with a two-layer chromosome which automatically evolves the limits in the solution search space together with the local search neighbourhood sequence. This work also shows that the evolved heuristic is more efficient than a fixed limit. Both works, however, do not generate high-quality solutions. Mostly likely due to the type of baseline heuristic used: a simple Local Search or Variable Neighbourhood Search.

In this work, we validate that automatic heuristic pruning can also lead to competitive quality. To do so, we build our framework around the Knowledge-Guided Local Search (KGLS), the state-of-the-art for solving large and very-large scale VRPs. The KGLS is described next.

C. The Knowledge-Guided Local Search [19]

The classic Guided Local Search (GLS) is a deterministic algorithm that attempts to escape the local optimum in which the LS algorithms inevitably fall into [27]. The GLS has a set of features that can be selected to penalise the current solution, moving it away from the pitfall. This is done by using different

objective functions to guide the solution, rather than changing the solution itself [5], [27].

The Knowledge-Guided Local Search (KGLS) is presented in [5] where the authors apply the classic Guided Local Search (GLS) with a newly introduced operator and penalisation functions. This was later adapted to (very) large-scale in [19]. The KGLS operates by sequential applications of a Local Search algorithm, and whenever a local optimum is reached a penalisation phase starts. These phases aim to remove the undesired edges in order to find new solutions which can potentially lead to a better overall solution. These penalisation functions were based on a study by the same authors [28] in which they investigate similarities across different VRP solutions, according to several metrics. One of the most effective metrics was the width of the routes.

The functions $b(.)$ measure the badness of an edge. The authors utilize three different functions $b(i,j)$, as described in Equations 1, 2 and 3. The first function is the standard penalization function for the GLS, penalizing long edges. While the other two penalise based on the width. The width of a given edge (i,j) is the distance between the customers i and j and a line (E) which crosses both the depot (D) and the centre of gravity (G) of the route which the edge belongs to, shown in Equation 4. These equations compute the penalization cost based on the number of penalisation already made to the edge $(p(i,j))$. Whenever an edge (i,j) has the worst value of $b(.)$, the value of $p(i,j)$ is incremented. More details can be found in the study of [5], [28].

$$b^c(i,j) = \frac{c(i,j)}{1+p(i,j)} \tag{1}$$

$$b^w(i,j) = \frac{w(i,j)}{1+p(i,j)} \tag{2}$$

$$b^{w,c}(i,j) = \frac{w(i,j)+c(i,j)}{1+p(i,j)} \tag{3}$$

$$w(i,j) = max(d_{iE}, d_{jE}) - min(d_{iE}, d_{jE}), E = Line(D,G) \tag{4}$$

The authors of [5] found that these penalisation functions are sufficient to find very good solutions across different datasets. Their work follows a similar algorithm structure as the one presented in Algorithm 2.

The paper [5] also adds a new improvement heuristic, namely Relocate-Chain, which moves a customer to another route, even if this breaks the feasibility of the route. In the cases this happens, it is fixed by relocating another customer to make it feasible. This can lead to a chain reaction of unfeasible moves until one route is found where the feasibility is restored. Although they show this move to improve the solution quality, it also requires an exponential number of chains and evaluations. The authors [5] avoid that by limiting the number of jumps a move can make. This is an example of how domain knowledge is applied to the problem solution.

Our work employs a similar KGLS but as a hyper-heuristic, which reduces some of the domain-specific parameters tuning and allow for a more independent heuristic pool.

III. The Proposed Evolutionary Hyper-Heuristic

In this section, we describe the Evolutionary Hyper-Heuristic (EHH) developed for this work. The overall idea is to use a Genetic Algorithm (GA) to evolve three different parts of the KGLS and also the order in which the operators are applied (as a Selection Hyper-Heuristics), together with how much of the neighbourhood should be searched for each operator, and the order of penalization criteria.

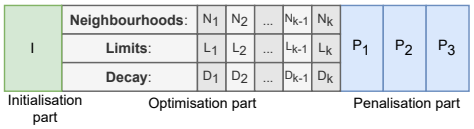

Fig. 1: Chromosome for our evolutionary hyper-heuristic.

A. Representation

The chromosome for this GA has three parts, as shown in Figure 1. The first part of the array, or **Initialisation part**, decides which initial algorithm to use. The integer value represents the algorithms that will build the initial solution. We consider a Random initialisation, the Savings heuristics from [29], and a Round-Trip (all customers are served by one vehicle each), with values $1, 2, 3$, respectively.

The second part, or **Optimisation part**, utilises a three-layer array: the Neighbourhoods layer, the Limits layer and the Decay layer. The first layer represents the sequence in which the operators are to be used. This layer searches for the most effective order in which to apply the operators to all instances through the evolutionary process. This concept has been applied to other problems before and have shown competitive results, such as [30]–[33] and was explored for the VRP in [12]. The second layer, or Limits layer, contains the limits applied to the number of customers to be searched, ranked by the closest ones for every single customer. For example, if the operator has 30% in the second layer, it means that only the 30% closest customers can be considered for the move. The Decay layer also trims the search space by reducing the number of routes that can be evaluated in inter-route moves. For example, if the third layer has a value of 50%, the correspondent operator will only consider 50% of all routes which are closest to the current one being evaluated. This closeness is calculated by the distance between the route's centre of gravity.

For the Optimisation part, the selected operators are easy-to-implement traditional VRP neighbourhoods. They are, with index (bold ones are inter-route):

1) Swap: Swaps position of two customers within the same route.
2) Two Opt: Exchanges two edges within the same route.
3) **Cross Exchange**: Exchanges two sub-routes of size k between two routes.
4) **Swap Star**: Swaps two customers on different routes.
5) Three Opt: Exchanges three edges within the same routes.

6) **Relocate**: Moves a customer from one route to another

The **Penalisation part** is the final part of the chromosome and represents the order of the three badness functions in which the KGLS penalises the solution. The values can be the functions 1, 2 and 3 (mapped as integers). However, these badness functions are not limited to inter-route moves only, nor does the heuristic follow the same pattern of inter to intra, as the original work does. This flexibility allows for fewer concerns regarding how to apply the penalisation.

B. GA Hyper-Heuristics

The framework utilises the traditional GA approach where the individuals go through the evolutionary process to optimise the fitness of the individuals. The process is divided into the training and the test phases and is summarised in Figure 2.

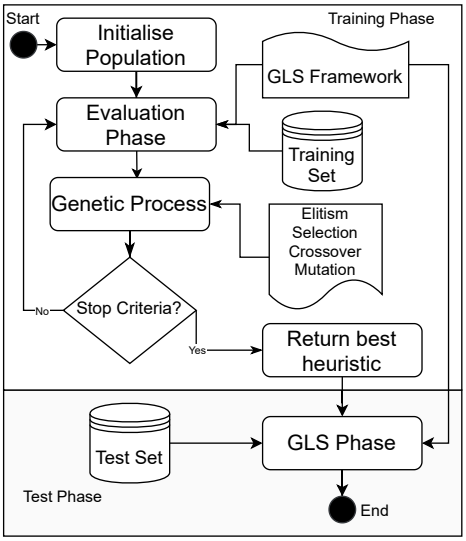

Fig. 2: Flowchart for the HH framework.

1) Training Phase: This phase contains the core part in which the hyper-heuristics *learn* how to solve the problem more efficiently. This phase consists of a population that goes through the evolutionary process and is initialised at random. At each generation, each individual is first evaluated. The rest of the evolutionary process follows with an elitism step which saves the best individuals and a crossover and mutation steps that try to improve the population quality for the next generations. These steps are summarized at Algorithm 1.

For the *evaluation step*, each individual is evaluated by applying the GLS framework (summarised in Figure 3 and Algorithm 2). Then, a normalized function based on how much the solution was improved compared to the Savings heuristic as the initial solution (which is pre-calculated before execution). This is done to avoid biases towards a larger improvement having better fitness if compared to the percentage improved, while in reality, the solution quality is the determinative factor. This fitness is calculated by the following equations:

$$N_i^F = \frac{f(x_i)}{CW_i} \tag{5}$$

Algorithm 1: Hyper-Heuristic Genetic Algorithm's Training Phase.

Input : Training Set T, Number of Penalised moves P, Elitist rate L, Crossover rate C, Mutation rate M
Output : Best Individual F_b

Calculate CW_i and L_i, i.e. the CW Savings for all Instances $i \in T$ with their average edge cost ;
Create a random initial *Population* ;
while *Stopping criteria not met* **do**
  foreach *Individual* $F \in$ *Population* **do**
    foreach *Instance* $i \in T$ **do**
      $N_i^F \leftarrow$ GLS(Individual,Instance, P)$/CW_i$
    $fitness^F \leftarrow \sum_{i \in T} N_i^F$
  Rank population by fitness ;
  $H_b \leftarrow$ Rank #1 of population ;
  Save best individuals according to rate E ;
  Apply one-point crossover to the population with rate C, following Tournament Selection as the Selection process ;
  Apply mutation to the population with rate M ;
Return H_b ;

$$fitness^F = \sum_{i=0}^{T} N_i^F \tag{6}$$

In Equation 5, we calculate the normalized improvement (N) for one instance i, for individual F, given the VRP solution x_i and the evaluation function f, which is the standard CVRP minimization function, minimizing total distance. The solution after the improvement given by the individual will likely be better than the initial solution constructed by the savings heuristic (CW_i). Therefore, the final value of N_i^F will be between 0 and 1, where the smaller means better. In Equation 6, we calculate the fitness of individual F given all values of N for the instances in the training set T.

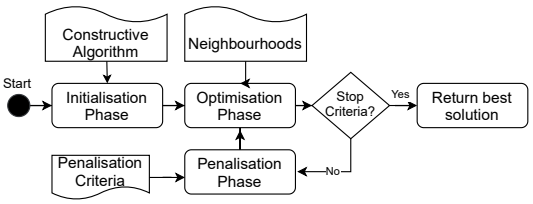

Fig. 3: Flowchart for the GLS framework.

For the *elitism step*, a percentage of the population is saved for the next generations according to the parameter E. The best individuals are added back to the population after the crossover and mutation steps finish, but are still included in the population for the selection process since they can generate better offspring.

As part of the *crossover step*, a Selection process applies a Tournament Selection (TS) to select which pair of individuals to crossover. TS will select random individuals based on the tournament size parameter, and return the fittest individual among them. We apply TS twice to get both individuals to match. Given the pair selected, there is still a probability of mating considering the parameter Crossover rate C. The two parents are replaced by the new offspring in the next generation, keeping the population with the same number of

individuals. The parents pair selected is passed to a one-point crossover, which is only applied for the **Optimisation part**, the other parts of the individuals are passed to each child the same as the parents. The one-point crossover selects a random position from the array and each child will get one of each parents part.

During the *mutation step* all the components of the individuals can be changed. If a given individual passes the mutation rate M, its **Initialisation part** and **Penalisation part** will get a new random initial heuristic or random order (by shuffling the penalisation part), respectively. The Optimisation part will get one of its heuristics modified to a random one.

This model allows the heuristic definition to be controlled by the GA, rather than being manually designed. The elitism guarantees good individuals to be kept in the population. The exchange of information of the crossover operator as well as the randomness of the mutation operator allows for an increase in diversity. The resulting best fit heuristic, i.e. the individual which got a better result across all instances in the training set (on average), will be the output of the algorithm and will be applied to the test instances [1]. Since larger neighbourhoods might contain better possibilities, it would be expected for the larger limits to obtain better fitness and dominate the population. However, as the running time for each individual is the same, the ones that find better solutions will be more fit, regardless of limits. This way the selection pressure acts in the algorithm and selects the best limits, regardless of the developer's idea of what should be better or faster. If the outcome results in smaller limits, it means that those are enough to find good solutions compared to larger ones[2].

2) Test Phase: Given the best individual from the previous phase, the unseen instances can be used to measure how efficient the training process was. The test phase simply applies the KGLS* (Algorithm 2), with this best individual. The test instances are then evaluated individually by their fitness, just as a regular meta-heuristic.

IV. Experiment Design

As the KGLS requires a set of complicated move sets for achieving its maximum capabilities, we re-implemented the algorithm with simpler moves, the same used in our hyper-heuristic, as in Section III-A. This adaptation (which will be referred to as KGLS*) was also required to do a more fairly comparison to our hyper-heuristics since both use the same components. Additionally, in order to avoid unnecessary computation by running the training set for a long amount of time, we performed preliminary experiments with this KGLS* adaptation to determine fair parameters for our

[1]Regarding time spent searching each neighbourhood, which although is not directly accounted for, there is an underlying factor which will try to optimise it.

[2]Comparing the same operator, if one searches its full neighbourhood for 5 seconds, and it takes the full time to do so, versus another which searches only half its neighbourhood, but does it in half the time, the second version can move from one local optimum from the first search and then to another one for the second run, given the same 5 seconds in total. This can lead to a better overall solution, although it is not guaranteed.

Algorithm 2: Knowledge-Guided Local Search used to evaluate the individuals.

```
Input  : Individual F, Instance i, Number of penalised moves P
Output : Instance value
Find initial solution to i according to F's Initialisation part heuristic ;
while Stopping criteria not met do
    while Local Optimum not reached do
        foreach LLH ∈ F's Hyper-Heuristic part do
            Apply LLH with its limits to the current solution ;
    Change penalisation function b to the next one in F's Penalisation part j ;
    while Number of penalised moves ≤ P do
        Penalise current solution with current penalise function b_j ;
        Select penalised edge e ;
        foreach LLH ∈ F's Hyper-Heuristic part do
            Apply LLH with its limits to edge e considering penalised edges;
        if A move was made then
            Increment number of penalised moves ;
Return value of best solution ;
```

EHH. Those experiments consisted of running the KGLS* for all instances for the same amount of time (10 minutes) and analysing their convergence speed and curves. Figure 4 shows the quality of the solution over time for the test set. The graph shows how much the solution has improved from the starting solution, relatively. As can be seen from the figure, most of the improvement happens within the first seconds of execution. Therefore, we determined that the training time limit for each individual in our hyper-heuristic would be of 30 seconds, which should be enough for finding the best improvement application of the heuristics. A similar trend was observed for the training set.

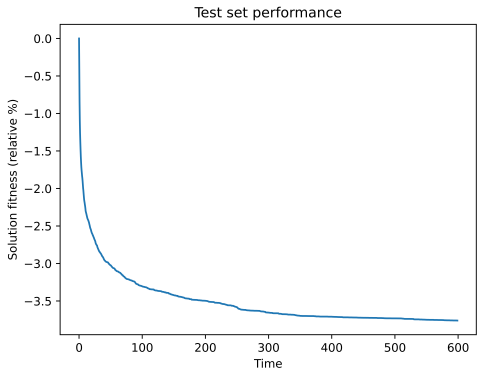

Fig. 4: Improvement over time for the KGLS* implementation considering the test instance set.

The experiment was designed to verify whether our EHH can find a better heuristic configuration than the manually designed KGLS. In this experiment, the EHH is compared to how good the final solution is, given the same amount of time. This aims to show whether our approach can find a better final solution compared to the KGLS*. This experiment also compares the final result to the original KGLS [19] and to

the Best-Known-Solution (BKS)[3]. The results for the KGLS were collected by running the available online tool the authors provided[4].

Finally, in order to have the experiments running faster, and to take advantage of modern computers with multi-threaded CPUs, we added a parallel evaluation of the individuals in a given generation. Since each individual evaluation is independent of one another, we can safely do this and save a significant amount of time in the training phase. Since we are running the training set with the instances which have a smaller number of customers, memory is also easier to be handled, as it will not require as much as the larger instances.

A. Training and Testing

The experiments were realized with the VRPLib dataset [34], which contains 100 instances varying in size (from 100 to 1000 customers), in the distribution of customers (randomly distributed, clustered, mixed), in customers demand (from 1 to 100, with different combinations of sizes), vehicle capacity (varies according to the demand for the instance) and depot location (central, eccentric and random). These different instances try to simulate several real-world scenarios in which the VRP can be applicable. A good solution method would perform well in all scenarios, even though that is hard to achieve since in general different heuristics work better under different situations.

The training set used was composed of all instances smaller than 200 customers, containing 21 instances (from 100 to 199). While the rest (79 instances) were used as the test set. We assume the sample of the 21 instances have enough variety for the algorithm to learn properly and apply the knowledge efficiently for the larger instances. The instances with fewer customers are also easier to compute due to the reduced search space and are able to find better solutions within a small amount of time. And although this does not necessarily translate to better quality for larger instances, it is a good indication according to our preliminary experiments (as shown in Figure 4).

B. Parameters

One of the main parameters of the KGLS is the number of penalised moves before moving to the optimisation phase. The authors of the original work have tried with 10, 100 and 1000, for different configurations of their KGLS (by comparing with different components). Although their final performance was better with 100, the 10 was actually better for the simple version of the KGLS with less advanced components. We compared this value and confirmed that 10 is better for our version too, and the value was used for our EHH.

For our EHH, the parameters are as shown in Table I. The population size and number of generations were defined by a few experiments, finding that larger numbers would take too long to evaluate all 21 training instances. The bounds for the hyper-heuristic part of the chromosome are wide enough to allow for the evolution to determine the better values.

TABLE I: The GA Hyper-Heuristics parameters.

Parameter	Value
Population size	20
No. of Generations	50
Cross. Rate	0.8
Mut. Rate	0.1
Elite Rate	0.1
Lower bound (for limit and decay)	0.1
Upper bound (for limit and decay)	0.9
Number of runs	30

All tests were realized in a Intel®Core™i7-8700 @ 3.2GHz and 15GB available memory, using half of the 12 threads for the parallel evaluation, and the implementation was done with C++ version 17.

V. Results

This section introduces the results obtained from the experiments. The results compared are for the test set only. We comment on the results based on the average case, best case and the evolve individuals over the 30 runs. The rank-sum Wilcoxon statistical test was performed to verify the significance of the results (p-value $= 0.05$).

Average Case: For the average case, we can observe that the EHH proposed performs relatively poor, as shown in Table II. On average, the EHH only beats (with significance, on the table in boldface) the KGLS* in 11 out of the 79 instances and drawing in 12 of them (no significant advantage or disadvantage). The results are even more significant if compared to the original KGLS, not beating it in any case. This, however, is to be expected since they apply some complex neighbourhood moves (such as the LK heuristic and the Relocate Chain).

Best Case: The best case, however, shows encouraging results. Also in Table II, the best EHH gap to the BKS is shown. The best EHH beats the KGLS* in 46 of the 79 instances, and even beating the original KGLS for one instance. However, even considering this case, the overall average is still shy of the KGLS*.

By analysing the results, we can see that most of the EHH best results are on the smaller side (less than 600 customers), losing in most of the larger instances. This could indicate that the heuristic configuration that performs well for the training set is not generalisable for larger instances.

When looking at the best-evolved heuristics, shown in Figure 6, and how they perform over time, we see that they present very similar behaviour. Showing that our EHH is robust to the randomness of the initial population, being all individuals within the $2 \sim 2.5\%$ relative improvement range, with only one outlier.

Without surprise, all best heuristics had the Savings algorithm as the initialisation method, since it provides a much better initial solution. The order of the penalisation operators appears to be less relevant, with most methods presenting the same order as the original KGLS* (sometimes shifted). Figure 5 shows an example of one of the best heuristics evolved.

[3]Retrieved from the dataset website: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

[4]Available on the website: https://antor.uantwerpen.be/routingsolver/

TABLE II: Results compared to BKS. In bold are the best between KGLS* and EHH, while underlined are the overall best.

Instance	KGLS to BKS	KGLS* to BKS	EHH Avg. to BKS	EHH Best to BKS	Instance	KGLS to BKS	KGLS* to BKS	EHH Avg. to BKS	EHH Best to BKS
X-n200-k36	0.29%	2.28%	**1.37%**± 0.57%	**0.84%**	X-n429-k61	0.43%	**1.58%**	2.09%± 0.38%	**1.35%**
X-n204-k19	0.52%	1.05%	1.09% ± 0.26%	**0.35%**	X-n439-k37	0.39%	**0.93%**	1.42%± 0.22%	1.10%
X-n209-k16	0.25%	1.69%	**1.22%**± 0.48%	**0.62%**	X-n449-k29	0.75%	2.55%	2.75%± 0.51%	**1.84%**
X-n214-k11	0.67%	**2.46%**	3.29%± 1.01%	**1.75%**	X-n459-k26	0.20%	**1.80%**	3.48%± 0.54%	2.86%
X-n219-k73	0.07%	0.15%	0.20%± 0.07%	**0.07%**	X-n469-k138	0.66%	**1.20%**	2.10%± 0.28%	**1.44%**
X-n223-k34	0.59%	1.79%	**1.57%**± 0.25%	**1.26%**	X-n480-k70	0.38%	**1.27%**	1.78%± 0.27%	1.32%
X-n228-k23	0.37%	**1.37%**	2.34%± 0.52%	**1.13%**	X-n491-k59	0.75%	**1.78%**	2.00%± 0.25%	**1.60%**
X-n233-k16	0.54%	**1.12%**	1.89%± 0.46%	**0.93%**	X-n502-k39	0.11%	0.57%	0.56%± 0.12%	**0.44%**
X-n237-k14	0.24%	0.82%	**0.65%**± 0.39%	**0.29%**	X-n513-k21	0.44%	**2.16%**	6.51%± 1.28%	4.05%
X-n242-k48	0.47%	1.23%	1.19%± 0.23%	**0.72%**	X-n524-k137	1.77%	**2.66%**	3.55%± 0.49%	**2.50%**
X-n247-k47	1.11%	2.60%	**2.07%**± 0.62%	**1.34%**	X-n536-k96	0.81%	**1.55%**	2.19%± 0.37%	1.60%
X-n251-k28	0.60%	1.64%	**1.33%**± 0.20%	**1.01%**	X-n548-k50	0.25%	1.00%	1.03%± 0.16%	**0.71%**
X-n256-k16	0.05%	**0.80%**	1.52%± 0.54%	**0.75%**	X-n561-k42	0.63%	**1.71%**	3.16%± 0.61%	1.96%
X-n261-k13	0.43%	**2.00%**	2.57%± 0.64%	**1.41%**	X-n573-k30	0.19%	**1.39%**	2.21%± 0.23%	1.82%
X-n266-k58	0.64%	1.31%	1.33%± 0.19%	**0.91%**	X-n586-k159	0.53%	**1.19%**	2.18%± 0.28%	1.63%
X-n270-k35	0.45%	**0.95%**	1.27%± 0.21%	**0.95%**	X-n599-k92	0.54%	**1.22%**	1.67%± 0.15%	1.39%
X-n275-k28	0.16%	**0.97%**	1.19%± 0.22%	**0.81%**	X-n613-k62	0.72%	**2.27%**	2.93%± 0.27%	2.43%
X-n280-k17	0.56%	**2.28%**	3.74%± 0.43%	2.96%	X-n627-k43	0.31%	**1.78%**	2.45%± 0.26%	1.99%
X-n284-k15	0.80%	**2.31%**	2.84%± 0.42%	**2.25%**	X-n641-k35	0.39%	**1.90%**	3.37%± 0.38%	2.68%
X-n289-k60	0.86%	1.56%	1.61%± 0.26%	**1.16%**	X-n655-k131	0.19%	0.49%	0.53%± 0.08%	**0.40%**
X-n294-k50	0.41%	1.65%	**1.43%**± 0.22%	**0.97%**	X-n670-k126	3.28%	**5.33%**	6.28%± 0.32%	5.83%
X-n298-k31	0.41%	**1.34%**	2.09%± 0.32%	1.55%	X-n685-k75	0.70%	**1.72%**	2.90%± 0.22%	2.37%
X-n303-k21	0.61%	**1.72%**	2.71%± 0.46%	2.06%	X-n701-k44	0.22%	**1.55%**	3.12%± 0.17%	2.72%
X-n308-k13	1.17%	**2.39%**	3.77%± 0.95%	**1.64%**	X-n716-k35	0.69%	**2.60%**	4.42%± 0.19%	3.96%
X-n313-k71	0.93%	1.49%	1.57%± 0.22%	**1.19%**	X-n733-k159	0.61%	**1.39%**	1.67%± 0.14%	**1.33%**
X-n317-k53	0.07%	0.41%	0.53%± 0.21%	**0.28%**	X-n749-k98	0.58%	1.69%	**1.56%**± 0.12%	**1.36%**
X-n322-k28	0.58%	**1.94%**	2.26%± 0.41%	**1.53%**	X-n766-k71	1.04%	**2.84%**	3.69%± 0.12%	3.47%
X-n327-k20	0.40%	**1.33%**	2.10%± 0.55%	1.43%	X-n783-k48	0.49%	**1.60%**	3.31%± 0.25%	2.72%
X-n331-k15	0.24%	**1.28%**	1.90%± 0.48%	**1.14%**	X-n801-k40	0.07%	**0.96%**	2.47%± 0.25%	1.82%
X-n336-k84	1.03%	**1.66%**	1.82%± 0.34%	**1.42%**	X-n819-k171	0.61%	**1.16%**	3.00%± 0.29%	2.51%
X-n344-k43	0.71%	**1.57%**	1.92%± 0.30%	**1.43%**	X-n837-k142	0.46%	**1.26%**	2.21%± 0.17%	1.83%
X-n351-k40	0.82%	2.58%	**2.15%**± 0.29%	**1.52%**	X-n856-k95	0.34%	**0.83%**	1.53%± 0.21%	1.28%
X-n359-k29	0.93%	**1.52%**	2.04%± 0.37%	**1.42%**	X-n876-k59	0.42%	**1.67%**	1.86%± 0.12%	1.71%
X-n367-k17	0.51%	**2.04%**	2.65%± 0.64%	**1.43%**	X-n895-k37	0.20%	**2.54%**	6.05%± 0.43%	4.88%
X-n376-k94	0.09%	**0.32%**	0.38%± 0.07%	**0.24%**	X-n916-k207	0.43%	**0.91%**	2.56%± 0.25%	2.10%
X-n384-k52	0.44%	1.39%	**1.34%**± 0.26%	**0.89%**	X-n936-k151	3.40%	**6.22%**	8.03%± 0.41%	7.32%
X-n393-k38	0.53%	1.93%	1.96%± 0.33%	**1.42%**	X-n957-k87	0.47%	**0.84%**	1.67%± 0.26%	1.49%
X-n401-k29	0.54%	**1.05%**	1.19%± 0.28%	**0.81%**	X-n979-k58	0.55%	**1.23%**	2.53%± 0.20%	2.16%
X-n411-k19	1.89%	**2.70%**	6.11%± 0.58%	**4.81%**	X-n1001-k43	0.40%	**2.85%**	5.10%± 0.19%	4.79%
X-n420-k130	0.56%	1.35%	**1.24%**± 0.41%	**0.71%**	**Average**	**0.61%**	**1.67%**	2.37%	1.77%

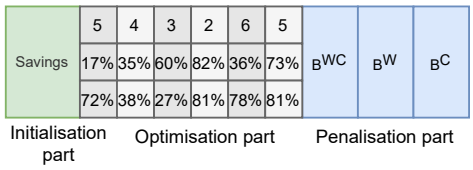

Fig. 5: Example of heuristic evolved individual.

VI. Conclusions

This paper introduces a new evolutionary hyper-heuristic to automatically configure a Knowledge-Guided Local Search algorithm, as well as its heuristics' pruning. The experiments and results show that the proposed EHH only perform well for the instances with up to 500 customers. One possible reason for the unsatisfactory performance on the larger instances is the evaluation of each training instance bound by a fixed time. Therefore, the EHH would only learn to efficiently solve the instances with a similar number of customers to those of the training set. The large number of high percentages (75% or more) limits and decays can be damaging for the larger neighbourhoods. Although this happens, we are confident that the goal of this work was reached, providing a hyper-heuristic method that can automatically and effectively prune the solution search space.

However, the weaknesses are clear, such as the lack of competitive results with state-of-the-art techniques, and, more importantly, the lack of generalisation for larger instances. To tackle these drawbacks, future work can be done, such as considering the same heuristic components as the KGLS (LK

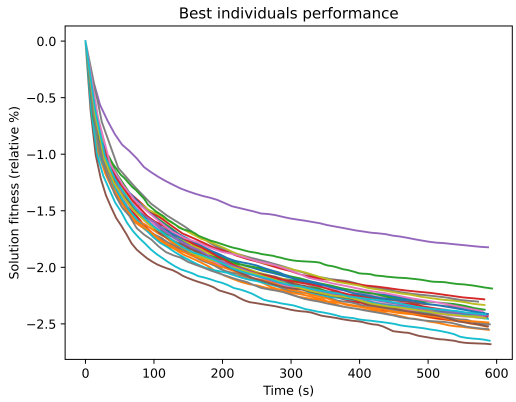

Fig. 6: Performance of the final heuristic over time, for all 30 runs.

and Relocate Chain), as well as providing a better form of evaluation, such as a surrogate technique.

REFERENCES

[1] P. Toth and D. Vigo, *The vehicle routing problem*. Philadelphia: Society for Industrial and Applied Mathematics, 2002.

[2] K. Buhrkal, A. Larsen, and S. Ropke, "The waste collection vehicle routing problem with time windows in a city logistics context," *Procedia - Social and Behavioral Sciences*, vol. 39, p. 241–254, 2012, seventh International Conference on City Logistics which was held on June 7- 9,2011, Mallorca, Spain. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877042812005721

[3] B. Hollis, M. Forbes, and B. Douglas, "Vehicle routing and crew scheduling for metropolitan mail distribution at australia post," *European Journal of Operational Research*, vol. 173, no. 1, p. 133–150, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221705000329

[4] L. Junqueira and R. Morabito, "Heuristic algorithms for a three-dimensional loading capacitated vehicle routing problem in a carrier," *Computers & Industrial Engineering*, vol. 88, p. 110–130, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835215002648

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

[6] 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.

[7] 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.

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

[9] A. Bevilaqua, D. Bevilaqua, and K. Yamanaka, "Parallel island based memetic algorithm with lin-kernighan local search for a real-life two-echelon heterogeneous vehicle routing problem based on brazilian wholesale companies." *Appl. Soft Comput.*, vol. 76, p. 697–711, 2019. [Online]. Available: http://dblp.uni-trier.de/db/journals/asc/asc76.html#BevilaquaBY19

[10] 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, 2013.

[11] N. Pillay and R. Qu, "Theoretical aspect—a formal definition," *Hyper-Heuristics: Theory and Applications*, p. 37–48, 2018.

[12] J. G. C. Costa, Y. Mei, and M. Zhang, "Adaptive search space through evolutionary hyper-heuristics for the large-scale vehicle routing problem," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Dec. 2020, p. 2415–2422.

[13] ——, "Cluster-based hyper-heuristic for large-scale vehicle routing problem," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Jul. 2020, p. 1–8.

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

[15] 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.

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

[17] 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.

[18] N. R. Sabar, X. J. Zhang, and A. Song, "A math-hyper-heuristic approach for large-scale vehicle routing problems with time windows," in *CEC*. IEEE, 2015, p. 830–837.

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

[20] M. M. Flood, "The traveling-salesman problem," *Operations Research*, vol. 4, no. 1, p. 61–75, feb 1956.

[21] Éric Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transportation Science*, vol. 31, no. 2, p. 170–186, may 1997.

[22] 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.

[23] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*. Princeton University Press, 2003.

[24] N. Pillay, "A study of evolutionary algorithm selection hyper-heuristics for the one-dimensional bin-packing problem." *South African Computer Journal*, vol. 48, p. 31–40, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/saj/saj48.html#Pillay12

[25] H. Al-Sahaf, Y. Bi, Q. Chen, A. Lensen, Y. Mei, Y. Sun, B. Tran, B. Xue, and M. Zhang, "A survey on evolutionary machine learning," *J. Roy. Soc. New Zeal.*, vol. 49, no. 2, p. 205–228, 2019.

[26] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job shop scheduling," *IEEE Transactions on Cybernetics*, 2020. Doi: 10.1109/TCYB.2020.3024849.

[27] C. Voudouris, E. P. Tsang, and A. Alsheddy, "Guided local search," in *Handbook of Metaheuristics*. Springer US, 2010, p. 321–361.

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

[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, p. 568–581, 1964.

[30] P. Cowling, G. Kendall, and L. Han, "An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem," *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, 2002.

[31] 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.

[32] G. Jiang, H. Dong, L. Yang, G. Li, and F. Xiang, "Hyperheuristic genetic algorithm for steelmaking continuous casting rescheduling based on strong disturbance of task," *International Journal of Wireless and Mobile Computing*, vol. 15, no. 3, p. 231, 2018.

[33] 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.

[34] E. Uchoa, D. Pecin, A. A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem." *Eur. J. Oper. Res.*, vol. 257, no. 3, p. 845–858, 2017.