Genetic programming for electric vehicle routing problem with soft time windows

Francisco Javier Gil Gala
University of Oviedo. Department of Computing
Gijón, Spain
giljavier@uniovi.es

Marko Đurasević
University of Zagreb, Faculty of Electrical Engineering and Computing
Zagreb, Croatia
marko.durasevic@fer.hr

Domagoj Jakobović
University of Zagreb, Faculty of Electrical Engineering and Computing
Zagreb, Croatia
domagoj.jakobovic@fer.hr

ABSTRACT

Vehicle routing problems (VRPs) that model transport processes have been intensively studied. Due to environmental concerns, the electric VRP (EVRP), which uses only electric vehicles, has recently attracted more attention. In many cases, such problems need to be solved in a short time, either due to their complexity or because of their dynamic nature. Routing policies (RPs), simple heuristics that build the solution incrementally, are a suitable choice to solve these problems. However, it is difficult to design efficient RPs manually. Therefore, in this paper, we consider the application of genetic programming (GP) to automatically generate new RPs. For this purpose, three RP variants and several domain-specific terminal nodes are defined to optimise three criteria. The results show that GP is able to automatically designed RPs perform, and it finds RPs with good generalisation properties that can effectively solve unseen problems.

CCS CONCEPTS

• **Computing methodologies** → **Planning and scheduling**; **Search methodologies**; • **Applied computing** → **Operations research**;

KEYWORDS

Genetic Programming, Electric Vehicle Routing Problem, Hyper-heuristics, Routing Policies

ACM Reference Format:
Francisco Javier Gil Gala, Marko Đurasević, and Domagoj Jakobović. 2022. Genetic programming for electric vehicle routing problem with soft time windows. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528994

1 INTRODUCTION

The vehicle routing problem (VRP) is one of the most important and well-studied combinatorial optimisation problems in which a set of routes must be determined for a fleet of vehicles that have to serve a given number of customers [20]. Due to the various real-world applications of VRP, different variants of the problem have been proposed over the years, some of which include additional features such as time windows (VRPTW) [18], time-dependent travel duration (TDVRP) [11], and many others [1]. The fact that current distribution logistics strategies are not sustainable has led much of the research to shift to VRP variants usually referred to as green VRP (GVRP) [16], which take into account the problems of sustainable transportation [14]. One variant of the GVRP that has received much attention in recent years is the electric VRP (EVRP) [7].

Constructive heuristics [13] to solve VRPs are the best suited when solutions need to be found more quickly, e.g., for larger problem sizes or in dynamic environments. However, developing efficient constructive heuristics is difficult because it depends on several factors such as the optimised criterion and additional problem properties. This motivates the search for approaches that can design such heuristics automatically as genetic programming (GP), which has been widely used to develop scheduling rules for various scheduling problems, including: job shop scheduling [19], unrelated machines scheduling [6], one machine scheduling [10] or constrained project scheduling [3], among others. In addition, efficient heuristics have been automatically generated using GP for the capacitated arc routing problem, which has certain similarities to VRP [15]. However, only one study has investigated the automatic development of heuristics for the VRPTW [12]. In it, the authors showed that automatically developed heuristics significantly outperform several manually developed heuristics.

In this paper, we study the application of GP to develop novel constructive heuristics, called routing policies, for the EVRP with time windows (EVRPTW). The remainder of the paper is organised as follows. Section 2 provides the definition of the EVRPTW problem under consideration. Section 3 describes the routing policies used to construct the solution to the problem. The GP hyper heuristic method is described in Section 4. Section 5 outlines the experimental setup and the results obtained. Finally, Section 6 presents the conclusion and future research directions.

2 PROBLEM DEFINITION

In this paper we use the formulation proposed in [17]. The problem is represented as a fully connected graph, where the set of vertices N represents the union of the set of customers P, the set of charging stations S, and the depot D. In the graph, the arc value $d_{ni,nj}$ represents the distance between the vertices n_i and n_j. The distance can be used to calculate the time $t_{ni,nj}$ and the energy $e_{ni,nj}$ required

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07…$15.00
https://doi.org/10.1145/3520304.3528994

for a vehicle to travel this distance. The objective of the problem is to serve all customers with a homogeneous fleet of m vehicles, denoted as V. Each vehicle has two properties, the remaining cargo capacity c_{vk} and the current energy level of the vehicle e_{vk}. All vehicles start at the depot with full capacity and energy. A route R_vk is created for each vehicle by assigning it customers to visit. When a vehicle adds a customer to its route, it must be ensured that this route remains feasible in terms of capacity and energy. In terms of capacity, this means that the vehicle v_k must have sufficient remaining capacity when serving the customers on its route. In terms of energy, it is necessary to ensure that the vehicle has sufficient remaining energy at any point along the route to visit the next customer. To this end, vehicles may also visit charging stations that fully recharge the vehicle. It is assumed that the charging stations have infinite charging capacity and can be visited by multiple vehicles simultaneously. At the end of the route, the vehicle returns to the depot.

In this paper, we consider the EVRPTW variant, which extends the basic problem by introducing time windows. The beginning of the time window will be denoted as ready time, while the end of the time window is called the due date. Unlike the energy and capacity constraints above, which are treated as hard constraints, the constraints imposed by time windows are considered as soft constraints and are optimised. We assume that the customer cannot be served before the start of its time window, and if the vehicle arrives earlier, it must wait until the beginning of the time window. However, the vehicle may arrive after the time window and serve the customer. In this case, however, the vehicle invokes a certain tardiness which must be minimised.

In this work, three objectives are optimised separately: 1) the number of vehicles used, 2) the total energy consumed, and 3) the total tardiness of the vehicles. The tardiness is calculated as the difference between the arrival time of the vehicle t_{vk} to vertex n_i and its due date dd_{ni}, i.e. $max(0.0, t_{vk} - dd_{ni})$.

3 ROUTING POLICIES

RPs are simple constructive heuristics that incrementally create routes for vehicles in VRPs. Each RP consists of two parts, a route generation scheme and a priority function.

3.1 Route generation scheme

The route generation scheme (RGS) is used to determine which location to go to next once a vehicle is available. The serial RGS creates routes vehicle by vehicle. In this version, a complete route is first created for one vehicle and then another route is created for the next vehicle. Another option is that the routes for all vehicles are created in parallel. In this case, the RGS starts with a certain number of vehicles and constructs the routes for all of them simultaneously. This is done so that the vehicle that is available earliest is selected and the next destination is determined for it. Once the routes for all vehicles are constructed, there is a possibility that there are still unattended customers. In this case, this RGS reverts to the serial approach and adds vehicle by vehicle to the solution and constructs the routes for them until all customers are served. This version is called the semiparallel RGS. Finally, another strategy can be used to add vehicles to the solution when a customer cannot be served by any of the vehicles, which is called parallel RGS. In this version, instead of adding new vehicles at the end by switching to serial RGS, a new vehicle is added immediately when another vehicle returns to the depot. The route for the newly added vehicle is created in parallel with all other currently available vehicles, as if it were available from the beginning.

In each iteration, the RGS first selects the active vehicle v_k for which the next destination is determined. The selection of the vehicle depends on which RGS variant is used. If the serial variant is used, the active vehicle is the first vehicle for which the route has not yet been completed. For the other two variants, the active vehicle is the vehicle that becomes available earliest. After the vehicle is selected, all unserved customers are ranked using a priority function (see next section) and the customer with the highest value is selected. Then the RGS checks if the vehicle has enough capacity to serve the selected customer n_i. If so, customer n_i is set as the destination, otherwise the vehicle returns to the depot. At this point, it is necessary to ensure that the vehicle has enough energy to reach the next selected customer, and that after serving a customer, the vehicle has enough energy to go to the nearest charging station. If this were not the case, there would be a possibility that a vehicle would not have enough energy to leave the current customer and travel to another location. If both conditions are met, then the vehicle v_k can drive directly to the customer n_i. Otherwise, the vehicle must visit one or more charging stations to reach the customer. In this case, the charging station that can be reached without violating the energy constraint and that is closest to customer n_i is selected. The same strategy is applied when the vehicle returns to the depot, but in this case the second constraint is not checked since the visit to the depot represents the end of the route.

3.2 Priority function

The priority function (PF) applied in the RGS is used to assign a numeric value to each customer based on the current state of the system. Based on the assigned numerical value, the RGS selects the most appropriate customer to visit. Therefore, the PF must use various system attributes in a meaningful way to assign a priority to each customer. For example, a simple PF could be defined as $\frac{1}{d_{n_i,n_j}}$, which would mean that the customer j closest to the current customer i should be selected. However, it is immediately apparent that such a PF will not perform well in general because it does not take into account other system attributes. To produce high-performance PFs, the PF should base its decision on several system parameters. However, it is quite difficult to design such sophisticated PFs manually.

4 HYPER-HEURISTIC METHOD FOR GENERATING ROUTING POLICIES

In this work, we have opted for a GP based Hyper-Heuristic approach similar to the one proposed in [9]. To adapt GP to a particular problem, the most important part is to define a set of primitives to be used to construct the expressions. The set of primitives consists of terminal symbols, which are used to encode the relevant information about the domain, and a set of functions, which can be either unary or binary. The set of terminal nodes we use is listed in Table 1. In developing the terminals, we chose simple terminals

that provide basic information about the system and avoided more complex terminals that could be defined as a combination of several simple ones. The terminal nodes E_{ni}, D_{ni}, DD_{ni}, ST_{ni}, and RT_{ni} represent the basic information about the customer ni that needs to be visited. Terminals C_{vk}, T_{vk} and E_{vk} represent the information about the active vehicle v_k that is updated each time a vehicle visits a customer. The remaining terminals EC_{ni}, ERP_{ni}, $EDep_{ni}$, ERP_{pvk}, and $EDep_{pvk}$ provide information about the customer and the vehicle relative to other customers, the depot, and the charging stations. All terminals are more or less self-explanatory, except for the energy required to visit the centroid EC_{ni} terminal. As suggested in [2], the distance to the centroid can be helpful to avoid visiting isolated customers. In this way, EC_{ni} is defined as the energy required to travel from the centroid to the customer EC_{ni}, which is the arithmetic mean of positions of all unvisited customers except n_i. In addition to the above terminals, constant values of $0, 0.1, 0.2, \ldots, 1$ are also used. Regarding the function used, we have used the symbols $-$, $+$, $/$, $\times$, max and min as binary functions, whereas $-$, pow_2, $sqrt$, exp, ln, max_0 and min_0 were used as unary functions.

Table 1: Terminal set used to build expression trees. v_k is the active vehicle, p_{vk} is the actual position of v_k and n_i is the destination of v_k.

Symbol	Description
E_{ni}	Energy required to visit n_i
D_{ni}	Demand of n_i
DD_{ni}	Due date of n_i
ST_{ni}	Service Time of n_i
RT_{ni}	Ready time of n_i
C_{vk}	Remaining capacity of v_k
T_{vk}	Current time of v_k
E_{vk}	Remaining energy of v_k
EC_{ni}	Energy required to visit the centroid from n_i
ERP_{ni}	Energy required to visit the nearest charging station from n_i
$EDep_{ni}$	Energy required to visit the depot from n_i
ERP_{pvk}	Energy required to visit the nearest charging station from p_{vk}
$EDep_{pvk}$	Energy required to visit the depot from p_{vk}

To compute the fitness function, each individual is interpreted as a PF embedded in the RGS and used to solve a set of problem instances of the EVRP. In this case, the evolved tree is used to assign a priority to each customer based on which the best customer is selected. If the priority value calculated by the tree is undefined (e.g., if a division by 0 has occurred) or infinite, it is interpreted as 0. Since the objectives that are optimised for the EVRP problem are all minimised, the fitness is defined as the inverse of the optimised criterion value. If two rules give exactly the same fitness, the number of symbols (size) is used for tie breaking in favour of the smaller trees.

5 EXPERIMENTAL ANALYSIS

5.1 Setup

We conducted an experimental study aimed at analysing the GP. To this end, we implemented a prototype in Java 8 and ran a series of experiments on a Linux cluster (Intel Xeon 2.26 GHz. 128 GB RAM).

The GP uses the one-point crossover and subtree mutation as genetic operators with probabilities 1.00 and 0.02, respectively. The population is composed of 200 individuals that are initialised by the Ramped half-and-half. The maximum execution time of 100 minutes is used as the termination criterion to ensure a fair comparison between all runs. Each experiment was repeated 100 times to obtain statistically significant results. To test whether differences exist between different variants, the Kruskal-Wallis test with Bonferroni analysis is used. The results are considered significantly different if a p-value below 0.05 is obtained.

The benchmark set consists of 92 EVRP instances obtained from [17], which are divided into three categories: random, cluster and random-cluster, depending on how the customer locations were generated. This set of instances is divided into a training set, which is used by GP to develop RPs, and a test set, which is used to evaluate their performance. These sets are constructed so that for each instance type, the instances are sorted by the number of customers and the instances with the odd index are added to the training set while the remaining instances are added to the test set. The result is a training set that consists of 47 instances, while the test set consists of 45 instances.

5.2 Results

Figure 1 shows boxplots for the results obtained on the training and test sets for optimisation of the number of vehicles, energy and tardiness criteria. We observe that the patterns of results obtained on the training and test sets are quite similar. This suggests that the selected training set is representative enough and that the GP produces policies that behave similarly in both problem sets.

By optimising the number of vehicles, GP is able to create rules that generate solutions with the optimal number of vehicles for the serial and semiparallel scheme, since they use the number of vehicles equal to the lower bound. When minimising the energy criterion, we find that the serial RGS performs best by a large margin. This seems to indicate that the other two RGSs are not suitable for this objective, regardless of the method to create the priority function. For the tardiness criterion, the best results were obtained with the parallel RGS. The semiparallel scheme performs slightly worse, while the serial RGS performs the worst. In this case, the best results are expected to be obtained by the parallel version, since it immediately includes more vehicles in the solution. This allows more customers to be served on time, but at the cost of using 13% more vehicles. However, it can be noted that the semiparallel and serial versions achieve similar results in terms of the number of vehicles. This shows that the parallel and semiparallel schemes, which are constructing the schedule by creating the routes for multiple vehicles in parallel, work better for this criterion.

Additionally, we can observe significant differences exist between the tested RGSs, which is especially evident when optimising the energy criterion. Statistical tests were performed to determine if the differences were significant, and the resulting p-values are shown above the boxplots. Based on these values, it is clear that there are significant differences between the RGSs for energy and tardiness criteria, so a post hoc analysis was performed. The Dunn test and Bonferroni correction were used to determine that there were significant differences between all RGSs. In terms of energy,

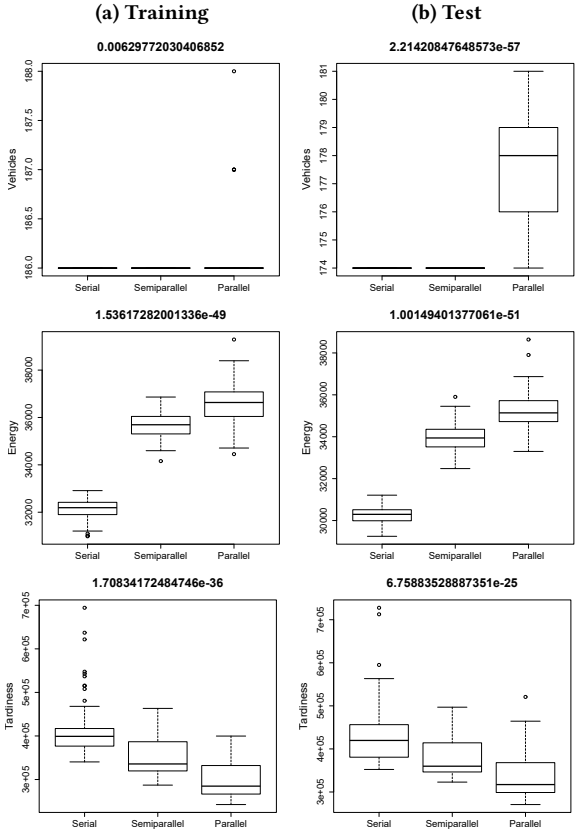

Figure 1: Boxplot from the results

the analysis showed that the serial RGS achieved significantly better results than the other two. In addition, the semiparallel RGS achieved a significantly better result than the parallel RGS. On the other hand, when considering the tardiness criterion the parallel RGS significantly outperforms the other two schemes, and the semiparallel RGS significantly outperforms the serial version.

6 CONCLUSIONS AND FUTURE WORK

This paper proposes the generation of RPs by GP to solve the Electric Vehicle Routing Problem with Time Windows (EVRPTW) and optimise three criteria: the number of vehicles, the energy consumed, and tardiness while visiting customers. Three RP variants are proposed and GP is adapted to generate PFs automatically for such RPs. The obtained results show that different RP variants perform better in solving different criteria, which highlights the importance of designing good route generation schemes in addition to PFs. In future work it is planned to extend this research in several directions. First, the problem will be extended to consider time windows as hard constraints. In addition, it is planned to analyse the proposed method in more detail and investigate how different terminal sets or expression sizes affect the performance. Another direction will also be to apply multi-objective optimisation to generate RPs that optimise multiple criteria simultaneously [4]. Finally, to improve the performance of the generated RPs, different ensemble learning methods will be tested to develop ensembles of RPs [5, 8].

REFERENCES

[1] Hasan Murat Afsar, Sezin Afsar, and Juan José Palacios. 2021. Vehicle routing problem with zone-based pricing. *Transportation Research Part E: Logistics and Transportation Review* 152 (2021), 102383. https://doi.org/10.1016/j.tre.2021.102383

[2] Gabriel Duflo, Emmanuel Kieffer, Matthias R. Brust, Grégoire Danoy, and Pascal Bouvry. 2019. A GP Hyper-Heuristic Approach for Generating TSP Heuristics. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 521–529. https://doi.org/10.1109/IPDPSW.2019.00094

[3] Mateja Dumić, Dominik Šišejković, Rebeka Čorić, and Domagoj Jakobović. 2018. Evolving priority rules for resource constrained project scheduling problem with genetic programming. *Future Generation Computer Systems* 86 (2018), 211–221. https://doi.org/10.1016/j.future.2018.04.029

[4] Marko Durasević and Domagoj Jakobović. 2017. Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. *Genetic Programming and Evolvable Machines* 19 (2017), 9–51.

[5] Marko Durasević and Domagoj Jakobović. 2018. Comparison of Ensemble Learning Methods for Creating Ensembles of Dispatching Rules for the Unrelated Machines Environment. *Genetic Programming and Evolvable Machines* 19, 1–2 (jun 2018), 53–92. https://doi.org/10.1007/s10710-017-9302-3

[6] Marko Durasević, Domagoj Jakobović, and Karlo Knežević. 2016. Adaptive scheduling on unrelated machines with genetic programming. *Applied Soft Computing* 48 (2016), 419–430. https://doi.org/10.1016/j.asoc.2016.07.025

[7] Tomislav Erdelić and Tonči Carić. 2019. A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches. *Journal of Advanced Transportation* 2019 (May 2019), 1–48. https://doi.org/10.1155/2019/5075671

[8] Francisco J. Gil-Gala, Carlos Mencía, María R. Sierra, and Ramiro Varela. 2021. Learning Ensembles of Priority Rules for Online Scheduling by Hybrid Evolutionary Algorithms. *Integrated Computer Aided Engineering* 28 (2021), 65–80. https://doi.org/10.3233/ICA-200634

[9] Francisco J. Gil-Gala, Carlos Mencía, María R. Sierra, and Ramiro Varela. 2019. Evolving priority rules for on-line scheduling of jobs on a single machine with variable capacity over time. *Applied Soft Computing* 85 (2019), 105782. https://doi.org/10.1016/j.asoc.2019.105782

[10] Francisco J. Gil-Gala, María R. Sierra, Carlos Mencía, and Ramiro Varela. 2021. Genetic programming with local search to evolve priority rules for scheduling jobs on a machine with time-varying capacity. *Swarm and Evolutionary Computation* 66 (2021), 100944. https://doi.org/10.1016/j.swevo.2021.100944

[11] Ali Haghani and Soojung Jung. 2005. A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research* 32, 11 (2005), 2959–2986. https://doi.org/10.1016/j.cor.2004.04.013

[12] Josiah Jacobsen-Grocott, Yi Mei, Gang Chen, and Mengjie Zhang. 2017. Evolving heuristics for Dynamic Vehicle Routing with Time Windows using genetic programming. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. 1948–1955. https://doi.org/10.1109/CEC.2017.7969539

[13] Gilbert Laporte, Stefan Ropke, and Thibaut Vidal. 2014. *Chapter 4: Heuristics for the Vehicle Routing Problem*. 87–116. https://doi.org/10.1137/1.9781611973594.ch4

[14] Canhong Lin, K.L. Choy, G.T.S. Ho, S.H. Chung, and H.Y. Lam. 2014. Survey of Green Vehicle Routing Problem: Past and future trends. *Expert Systems with Applications* 41, 4, Part 1 (2014), 1118–1138. https://doi.org/10.1016/j.eswa.2013.07.107

[15] Jordan MacLachlan, Yi Mei, Juergen Branke, and Mengjie Zhang. 2020. Genetic Programming Hyper-Heuristics with Vehicle Collaboration for Uncertain Capacitated Arc Routing Problems. *Evolutionary Computation* 28, 4 (12 2020), 563–593. https://doi.org/10.1162/evco_a_00267 arXiv:https://direct.mit.edu/evco/article-pdf/28/4/563/1859049/evco_a_00267.pdf

[16] Reza Moghdani, Khodakaram Salimifard, Emrah Demir, and Abdelkader Benyettou. 2021. The green vehicle routing problem: A systematic literature review. *Journal of Cleaner Production* 279 (2021), 123691. https://doi.org/10.1016/j.jclepro.2020.123691

[17] Michael Schneider, Andreas Stenger, and Dominik Goeke. 2014. The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations. *Transportation Science* 48 (03 2014), 500–520. https://doi.org/10.1287/trsc.2013.0490

[18] K.C Tan, L.H Lee, Q.L Zhu, and K Ou. 2001. Heuristic methods for vehicle routing problem with time windows. *Artificial Intelligence in Engineering* 15, 3 (2001), 281–295. https://doi.org/10.1016/S0954-1810(01)00005-X

[19] Joc Cing Tay and Nhu Binh Ho. 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers and Industrial Engineering* 54, 3 (2008), 453–473. https://doi.org/10.1016/j.cie.2007.08.008

[20] Paolo Toth and Daniele Vigo. 2014. *Vehicle Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA. https://doi.org/10.1137/1.9781611973594 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611973594