ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Influence of function nodes on automated generation of routing policies with genetic programming

Marko Đurasević[1] and Francisco Javier Gil Gala[2] *

1- Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb - Croatia

2- Department of Computer Science, University of Oviedo
Gijón - Spain

Abstract. Routing policies (RPs) are simple heuristics used to solve the electric vehicle routing problem, suitable for large or dynamic problems. Designing efficient RPs is difficult, because of which researchers started applying genetic programming (GP) for their automated development. For GP to be able to generate efficient RPs, it must be supplied with appropriate building blocks, i.e., functions and problem features, to construct the solution. This study investigates the selection of appropriate function nodes to construct RPs. The experiments demonstrate that the best results are obtained when using the most simple arithmetic operators enhanced with some additional operators.

1 Introduction

Routing policies (RPs) are simple heuristic methods that are used to solve numerous variants of the vehicle routing problem [1]. These RPs construct the solution incrementally, by determining the next decision that should be selected and executed. In that way they do not construct the entire solution, but rather only determine the next state of the system. This makes them suitable in generating solutions for large, dynamic, or uncertain problems, since they can always take the latest information into account when performing the decisions.

However, manually designing RPs is a quite tedious and time consuming task which requires expert knowledge about the problem domain. For that purpose, genetic programming (GP) has often been used to automatically designing heuristics for various combinatorial optimisation problems [2, 3]. Until now, this methodology has proven to be quite efficient in designing new and efficient heuristics for various vehicle routing problems [4, 5].

One of the most important decisions when using GP to design heuristics is which building blocks, i.e., function and terminal nodes, should be used to construct the solution. These two sets define the expressiveness of the potential solution. Although the selection of terminal nodes is commonly investigated, function nodes are usually selected based on previous experience or rule

*This research has been supported by the European Union - NextGenerationEU under the grant NPOO.C3.2.R2-I1.06.0110., by the Croatian Science Foundation under the project IP-2022-10-5398, and the Spanish Government under projects MCINN-23-PID2022 and TED2021-131938B-I00.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

of thumb. Therefore, in this study we investigate the effect of various function node sets on the quality of the RPs that are generated by GP for the Electric Vehicle Routing Problem with Time Windows (EVRPTW) [6].

2 Electric Vehicle Routing Problem

EVRPTW is modelled as a symmetrical fully connected graph, where nodes represent customers, depots, or charging station, and edge weights represent the distances between the different nodes. Each customer has a certain demand that needs to be satisfied, a service time that defines the amount of time required to serve the customer, and a time window that defines during which time the service can start. In order to serve the customers, a certain number of electric vehicles is available, each with a cargo capacity, battery level, and speed. As the vehicle traverses an edge the energy level of the battery is reduced, and as the vehicle visits a customer the cargo capacity is also reduced. At the start all the vehicles are located at the depot, to which they also need to return when they complete their routes. As the battery levels of the vehicles diminish, the vehicles can visit charging stations to recharge their battery. In this problem variant it is presumed that the battery is recharged until 100%, with the time required to recharge it linearly depending on the amount that was recharged. In the considered problem, the fleet consists of homogeneous vehicles, meaning that all of them have the same characteristics. The optimisation criterion that is minimised in this problems is first the number of vehicles, and then the lateness. The lateness is defined as the amount of time that the vehicle arrived at the customers after the end of the time window. To optimise both objectives the fitness function is defined as:$f(x) = c * V(x) + L(x)$, where $f(x)$ denotes the total fitness of solution x, V denotes the number of used vehicles, c denotes a user defined scaling parameter, and L denotes the total lateness of vehicles in solution x. In this study $c = 10^9$ to primarily focus on optimising the number of vehicles, and then the lateness.

3 Automated Design of RPs

A RP consists of two parts, the routing scheme (RS) and the priority function (PF). The RS defines how the solution is being constructed. This means that at each decision moment, which happens when a vehicle becomes available, it determines the next customer that should be served. In order to determine which customer should be visited, the RS ranks all the customers and selects the best one as the one that should be visited next. In order to obtain the ranking of all customer, the RS uses a PF which assigns a numeric value to all the decisions, based on which the best one can be selected.

The PF represents a certain mathematical function that based on certain problem characteristics provides a numerical value. This makes the PF appropriate for being designed by GP. However, in order to be able to do this, it is required to define the set of terminal and function nodes that will be used as

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

building blocks to construct this expression. The set of terminal nodes that were used for this purpose are represented in Table 1. As one can see, terminal nodes model different characteristics and aspects of the problem, such as the distance between the nodes, the end of the time windows of the customer, but also some more sophisticated characteristics like the slack of the vehicle or features of other vehicles in the fleet.

Table 1: Description of the terminal nodes used by GP.

Node	Description
d_j	Distance from current vehicle location to customer j
dd_j	End of the time window for serving customer j
va_j	Time when the current vehicle becomes available
vc	Remaining vehicle cargo capacity
vb	Remaining vehicle battery capacity
sl_j	Slack (time until late) of vehicle for customer j
ttr_j	Time until customer j is ready
wt_j	Waiting time for vehicle until customer j becomes ready
ddc_j	Distance of customer j to the depot
sad	Distance of second next available vehicle to customer j
sac	The capacity of the second next available vehicle
sal	Energy level of second next available vehicle
nva	Time when the nearest vehicle to customer j is available
nvd	Distance of the nearest vehicle to customer
$nvel$	The energy level of the nearest vehicle to customer

Function nodes examined in this study are grouped into five groups:

- Group 1 — consists of the basic arithmetic operators: +, -, *, /, where / is defined as the protected division operator, meaning that if division by 0 occurs that a default value of 1 is returned.
- Group 2 — consists of the binary minimum and maximum operators that return the smaller or larger out of the two given values, respectively.
- Group 3 — consists out of conditional branches, $ifpos$ and $ifgt$. The $ifpos$ function accepts three arguments and returns the second argument if the first one is positive, or the third one if the first one is negative. $ifgt$ accepts four arguments and if the first one is larger than the second one, the function returns the third argument, otherwise it returns the fourth.
- Group 4 — consists of various nonlinear mathematical functions, such as sin, cos, ln, exp, and $sqrt$. The $sqrt$ and ln operators are implemented as protected operators, meaning they return 1 if the argument is negative.
- Group 5 — consists of two unary functions neg and pos. The neg operator simply negates its argument, whereas the pos operator returns the maximum between the argument and 0, meaning it return the argument if it is positive, otherwise it returns 0.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

In the experimental study, all the combinations that include group 1 will be investigated. The reason why this group is always included is since it includes the basic arithmetic operators that are always used.

4 Experimental study

4.1 Setup

In order to investigate the performance of different function sets on the quality of the obtained RPs, the Schneider dataset is used [7]. This dataset consists out of 56 instances that contain 100 customers, 1 depot, and 21 charging stations. The number of vehicles required to serve the customers depends on the instance. based on the distribution of the customers, the instances can be divided into random (customers are distributed randomly across the space), denoted with R, cluster (customers are distributed around predefined clusters), denoted with C, and random cluster (represents a combination of both previous), denoted with RC. Furthermore, the instances can also be divided into those with tight time windows (difficult to timely serve the customers), denoted with T, and those with loose time windows (easier to serve the customers on time), denoted as L. The original dataset is used to evaluate the RPs on an unseen set of problems, whereas for training GP another dataset generated in the same way was used.

To evolve RPs, a standard steady state 3-tournament GP algorithm was used. In each iteration this algorithm selects three individuals randomly that participate in a tournament, and selects the better two and performs crossover on them to obtain a child individual. The child individual is mutated with a certain probability and inserted into the population by replacing the worst individual in the tournament. GP uses 100 individuals, a mutation probability of 0.3, and 10 000 function evaluations. Regarding the genetic operators, for crossover the uniform, size fair, context preserving, subtree, and one point operators are used, whereas for mutation the hoist, subtree, permutation, node complement, node replacement, and shrink operators are used [8]. The implementation was done in C++ using the ECF framework [9], whereas the experiments were executed on a Windows 11 PC with an AMD Ryzen Threadripper 7980X 64-Core processor and 256 GB of RAM. Each experiment was executed 30 times.

4.2 Results

Table 2 outlines the results obtained in the experiments. The table also includes the results of the manual nearest neighbour (NN) rule that serves as the baseline. This rule selects as the next destination the customer that is closest to the current location of the vehicle. GP was executed for each function set combination and for each problem instance type separately. The table outlines the median values of the 30 executions that were performed. Furthermore, the table also outlines the average rank (column "A. R.") of each function set calculated across all problem types, and also the final rank (column "T. R.") of the sets based on their average ranks.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Table 2: Results for different function node combinations.

Function set	C-T	C-L	RC-1	RC-2	R1	R2	A. R.	T. R.
NN	366410	1031580	176810	1150980	123412	879007	-	
1	768	96773	58236	784952	52504	856396	9.7	12
1+2	754	**84337**	44224	796274	45755	866618	5.7	2
1+3	833	93746	51185	796984	47500	855590	8.7	10
1+4	734	97326	42549	794872	48223	852292	6.5	4
1+5	635	90565	**42282**	**783910**	47547	864059	**5.5**	**1**
1+2+3	865	89779	50773	815077	47235	859323	9.3	11
1+2+4	689	96526	43340	853724	48568	862527	9.7	12
1+2+5	803	88576	46319	793766	47259	853043	5.8	3
1+3+4	886	94810	42360	827247	48017	**849333**	8.0	6
1+3+5	932	90487	44857	810439	46130	858957	8.2	7
1+4+5	760	98432	44604	819881	51349	855743	9.7	12
1+2+3+4	802	94223	45407	831958	46087	858614	8.3	8
1+2+3+5	**632**	95342	48863	819132	**45525**	856860	6.8	5
1+3+4+5	809	97771	52433	794105	46382	858304	9.7	12
1+2+3+4+5	781	94860	46576	867947	46233	854972	8.5	9

The results demonstrate that there is not a single function set that results in the best performance of the RPs for each problem type. The best rank was achieved by the function sets including groups 1, 2, and 5, in any of the tested combinations. Since it is always better to use a smaller function set, it is better to use either the set 1+2 or 1+5, since the results are comparable between them. On the other hand, the worst results are obtained either when only arithmetic operators are used (group 1), or when the set of nonlinear operators is included (group 4). Therefore, from the results we see that as the size of the function set increases the results do not improve, rather they even become worse. We see that in all cases the manually designed RRs achieve a better performance than the manually designed NN rule.

Figure 1 represents the box plot representation of the results aggregated for each problem type. We see that most of the function sets achieve comparable results, and that the dispersion of the results depends on the function node set that is used. Furthermore, the sets that lead to the best results also achieved the least dispersed results. The inclusion of conditional functions or nonlinear mathematical operators increases the dispersion of the results, thus denoting that it is more difficult to obtain better results.

5 Conclusion

The goal of this study was to investigate the influence of different function nodes on the quality of the RPs designed by GP for solving the EVRPTW. For that purpose, different function sets were investigated, which included different mathematical or conditional operators. The obtained results show that the best results are obtained by using only arithmetic operators extended with either the minimum and maximum operators or the *pos* and *neg* functions. On the other hand, including various nonlinear or conditional functions does not help improve the results. Furthermore, it is better to keep the size of the function set as small as possible, due to the increase in the search space that the larger sets cause.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

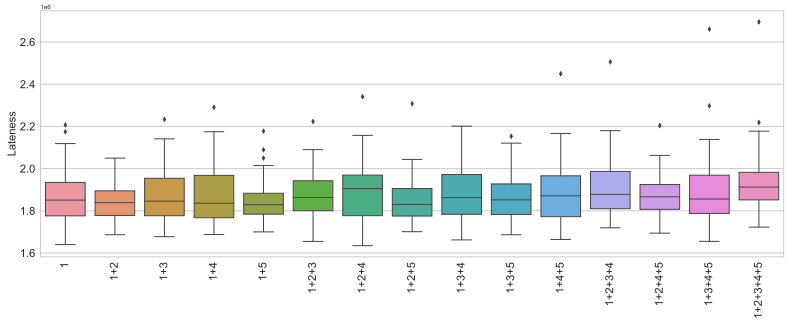

Fig. 1: Box plot of the total obtained results.

In future studies our goal is to continue with topics that are connected to the representation of the PF. Therefore, alternative representations will be investigated, such as linear GP. Furthermore, feature construction methods will also be investigated to determine whether it is possible to construct new features from the PFs that were previously evolved. Finally, the plan is also to deal with the topic of interpretability and explainability of the evolved PFs, with the goal of obtaining PFs that are easier to interpret by humans.

References

[1] Francisco J. Gil-Gala, Marko Đurasević, and Domagoj Jakobović. Evolving routing policies for electric vehicles by means of genetic programming. *Applied Intelligence*, 54(23):12391–12419, September 2024.

[2] Jürgen Branke, Su Nguyen, Christoph W. Pickardt, and Mengjie Zhang. Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation*, 20(1):110–124, 2016.

[3] Su Nguyen, Yi Mei, and Mengjie Zhang. Genetic programming for production scheduling: a survey with a unified framework. *Complex amp; Intelligent Systems*, 3(1):41–66, February 2017.

[4] Josiah Jacobsen-Grocott, Yi Mei, Gang Chen, and Mengjie Zhang. Evolving heuristics for dynamic vehicle routing with time windows using genetic programming. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1948–1955, 2017.

[5] Francisco Javier Gil-Gala, Sezin Afsar, Marko Durasevic, Juan José Palacios, and Murat Afsar. Genetic programming for the vehicle routing problem with zone-based pricing. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 1118–1126, New York, NY, USA, 2023. Association for Computing Machinery.

[6] Hu Qin, Xinxin Su, Teng Ren, and Zhixing Luo. A review on the electric vehicle routing problems: Variants and algorithms. *Frontiers of Engineering Management*, 8(3):370–389, May 2021.

[7] Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48:500–520, 03 2014.

[8] Riccardo Poli, William Langdon, and Nicholas Mcphee. *A Field Guide to Genetic Programming*. 01 2008.

[9] Domagoj Jakobovic, Marko Đurasević, Stjepan Picek, and Bruno Gašperov. Ecf: A c++ framework for evolutionary computation. *SoftwareX*, 27:101640, September 2024.