Evolving Graphs with Cartesian Genetic Programming with Lexicase Selection

Yuri Lavinas
lavinas.yuri.xp@alumni.tsukuba.ac.jp
IRIT - CNRS UMR5505
Toulouse, France

Kevin Cotacero
kevin.cortacero@inserm.fr
IRIT - CNRS UMR5505, University of Toulouse
Toulouse, France

Sylvain Cussat-Blanc
sylvain.cussat-blanc@irit.fr
IRIT - CNRS UMR5505, University of Toulouse
Toulouse, France

ABSTRACT

The automatic construction of an image filter is a difficult task for which many recent machine-learning methods have been proposed. Cartesian Genetic Programming (CGP) has been effectively used in image-processing tasks by evolving programs with a function set specialized for computer vision. Although standard CGP is able to construct understandable image filter programs, we hypothesize that explicitly using a mechanism to control the size of the generated filter programs would help reduce the size of the final solution while keeping comparable efficacy on a given task. It is indeed central to keep the graph size as contained as possible as it improves our ability to understand them and explain their inner functioning. In this work, we use the Lexicase selection as the mechanism to control the size of the programs during the evolutionary process, by allowing CGP to evolve solutions based on performance and on the size of such solutions. We extend Kartezio, a Cartesian Genetic Programming for computer vision tasks, to generate our programs. We found in our preliminary experiment that CGP with Lexicase selection is able to achieve similar performance to the standard CGP while keeping the size of the solutions smaller.

CCS CONCEPTS

• **Computing methodologies → Bio-inspired approaches**.

KEYWORDS

Evolutionary computation, Cartesian Genetic Programming, Lexicase Selection, graph-based methods

ACM Reference Format:
Yuri Lavinas, Kevin Cotacero, and Sylvain Cussat-Blanc. 2023. Evolving Graphs with Cartesian Genetic Programming with Lexicase Selection. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583133.3596402

1 INTRODUCTION

Genetic Programming (GP) evolves programs based on a set of mathematical functions that, when executed, process given inputs to produce an expected output. Among variants of GP, Cartesian genetic programming (CGP) uses graph representations to encode computer programs. One of the main contribution of CGP is the use of a fixed-length integer-based genome to encode the functional graphs. Therefore, small programs can be evolved and later read for understanding [17]. Moreover, mostly because of the use of a fixed-size genome, it has been shown that bloat is not a problem for CGP [16]. Bloat is an undesirable effect that occurs in many Genetic programming variants that causes the size of the programs to grow larger without a clear impact on the overall performance of the evolved programs or even destroying the search ability of the algorithm [14].

However, there isn't an explicit mechanism to control the size of the graphs generated during the search process of CGP. This could make our ability to interpret the programs evolved and our ability to extrapolate from them harder. In addition to that, bigger programs are prone to overfit to the training set, causing generalisation problems [14]. As we understand that generating well-performing programs that are also easy to interpret and extrapolate from is imperative, we conduct a preliminary study on the effects of introducing an explicit mechanism to control the size of graphs in CGP.

The main goal of this article is to verify if CGP with an explicit size control mechanism is able to keep competitive performance in comparison with the standard CGP while reducing the size of the programs generated. The choice of the method we focus on in this work to reduce the size of the final programs generated is the ϵ-Lexicase selection, which has been shown to be able to keep the size of the programs generated with GP to a smaller size while keeping their high performance in many scenarios or metrics [6, 10, 14].

We choose to analyse the different CGP variants on a segmentation task. For that, we use a benchmark dataset, that contains images of coins in a neutral background. We use the IOU as the performance metric and the number of active nodes, nodes that are connected with the program output, as our size metric[1].

2 RELATED WORKS

Most of the works on computer vision tasks use black-box approaches, such as artificial Deep Neural Networks. However, it is generally considered to be difficult for humans to analyze and interpret their outputs [3, 13]. One way to complement these black-box methods is to use methods that are inherently explainable, that do not require additional mechanisms to be explainable.

[1]For reproducibility purposes, all the code and experimental scripts will be made available upon acceptance.

Among such methods, we highlight Cartesian Genetic Programming, an evolutionary computation algorithm that evolves easier-to-interpret programs, However, most of the works related to CGP evolve programs without a clear, explicit size control mechanism [1–3, 5, 15, 17]. This is probably because it is understood that controlling the size of the programs evolved with Cartesian Genetic Programming (CGP) isn't necessary, since CGP isn't affected by bloat. One work that caught our attention, by Kalkreuth et al. [9], where NSGA-II is use to evolve small program, following a multi-objective approach.

In the context of bloat in Genetic Programming, a popular size and bloat control method is the Lexicase selection. Another argument in favor of this method is its simplicity implementation [4, 8, 10] and therefore applicability to not yet explored domains. Interestingly, Lexicase selection is able to focus on multiple goals at the same time [14], since it selects programs according to performance on different metrics considered [7].

3 PRELIMINARIES

3.1 Cartesian Genetic Programming

Cartesian Genetic Programming is a Genetic Programming variant [11] specialized in evolving graphs. Such graphs are often direct and acyclic and are indexed by Cartesian coordinates. The evolutionary process defines how to connect the nodes of the graphs and the instructions or functions of each node.

CGP has been successfully used in multiple domains [1, 2, 15]. In especial, CGP has been applied in computer vision tasks as can be seen for example for controlling agents to play ATARI games [17], and in computer vision tasks that uses image processing functions from openCV and which applies for programs directly to images [3, 5]. Only a few nodes, called "active" nodes, are used as they actually are connected to the output of the program graph. The other nodes with no connections to the output are called "inactive" nodes. The outputs of the program are taken from any internal node or input, also defined during the evolutionary process.

The solutions in CGP are generally optimized by using the 1+λ algorithm, although any other evolutionary algorithm could be used, in theory. A population of λ individuals are randomly generated and evaluated on the problem in question. The evaluation process is conducted by first generating the program from the graphs evolved and then measuring the performance of such programs on the task considered. The solution with the highest performance is maintained to the next iteration step, influencing the next λ individuals. This process is repeated until the stop criteria are met. For more information see [2, 11, 17].

3.2 Lexicase Selection

Lexicase selection is a selection method designed to manage multiple metrics at the same time originally proposed for program synthesis tasks. At every step, this method orders the test cases and gives priority to solutions with high metric values on the randomly ordered metric. Therefore, the method can search for solutions that have good metrics values at the same time with the benefit of increasing diversity during the search process [6]. Overall, Lexicase selection considers multiple metrics during the whole process, but without which selecting solutions based on aggregations of metrics.

One disadvantage of this selection method is that it can lead to poor performance on problems with a continuous fitness space, and to deal with La Clava et al. [10] introduced the ϵ-Lexicase selection, that uses an adaptive threshold parameter σ. In contexts with a limited number of metrics, this selection method can lead to low diversity values [6]. In this work, we only consider two evaluation metrics, one based on the detection performance and one on the graph size. Therefore, we used a ϵ-Lexicase selection method as our base mechanism to reduce the size of the programs during the evolution process.

3.3 Evaluation metrics

Intersection over Union (IoU). evaluates the performance of an algorithm in detecting objects within an image by comparing the ratio among areas occupied by different objects [12] by measuring the quality of a predicted mask A mask covering exactly the annotation will produce an Intersection equals to Union, higher values suggest a higher overlap among such objects (the maximum is 1.0).

Number of active nodes. calculates how many nodes are connected to the program output. We use it to indicate the size of the programs generated.

4 CGP-LEX

ϵ-Lexicase selection based Cartesian Genetic Programming (CGP-LEX) is a CGP that selects parents according to different metrics considered at the same time. The main idea behind implementing such a variant of CGP is to evolve programs that perform well on all metrics considered at the same time. The CGP implementation we use is based on Kartezio, a modular-based Cartesian Genetic Programming to generate our programs for computer vision tasks, that introduces the notion of non-evolvable nodes which are functions not subjected to optimization of the syntactic graph. The Image Processing stack through the algorithm optimizes the arrangement of functions and the parameters in order to produce an image processing algorithm similar to an human-designed one. Our implementation adapts Kartezio to incorporate Lexicase selection.

5 DESIGN OF EXPERIMENTS

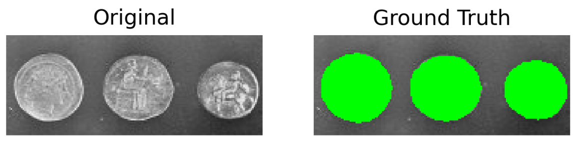

Figure 1: Example of the coins dataset used.

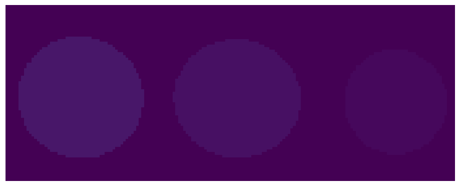

Figure 2: Output of a program generated by CGP-LEX.

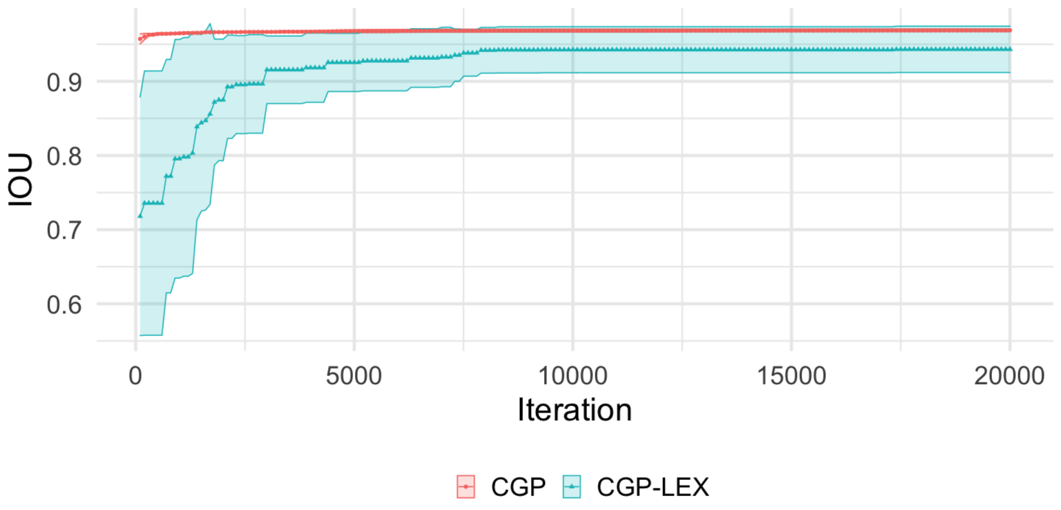

Figure 3: Mean IOU values (shaded areas show the standard deviation) of the elite solution over the iterations.

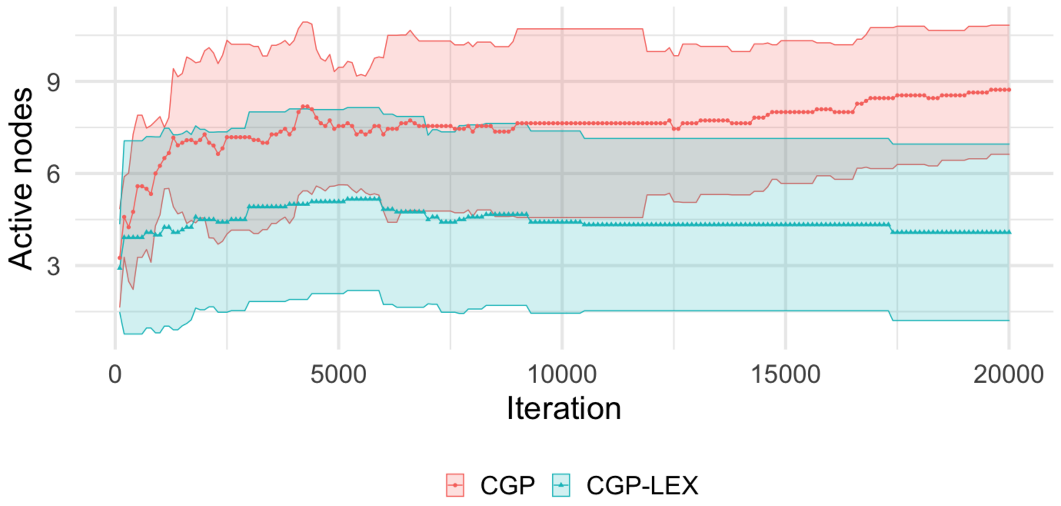

Figure 4: Mean number of nodes (shaded areas show the standard deviation) of the elite solutions over the iterations.

To evaluate CGP and CGP-LEX, we compare their performance and the size of the programs generated on a simple image segmentation task. We use a dataset of coin images, where the goal is to identify these objects in the background. Figures 1 and 2 show an example of the dataset and one possible output of the segmentation made by a program generated by one of the CGP variants.

The operators in the nodes, for any of the CGP methods studied here, are function calls to the OpenCV Python package [2]. All functions are image-size independent, which implies that the input and output for a given image are equivalent. We use the standard 1+λ EA is used to evolve the programs. We run all CGP variants of the models with 30 nodes (30 columns and 1 row), λ = 5 offspring over 20,000 iterations. The mutation probability was 0.15 for the functional nodes and 0.2 for the outputs, as in [3]. More work in tuning the parameters for the CGP variants. Finally, 10 independent runs were made for each experiment for statistical purposes, due to time constraints.

[2]A complete list of functions can be founda the github repository.

6 RESULTS

Figure 3 shows the mean IOU performance (shaded areas show the standard deviation) of elite solutions over the iterations of the 10 runs of CGP and CGP-LEX. We can see that CGP is able to achieve higher performance at the very beginning of the search, while GCP-LEX takes longer to get to about the same IOU values. Moreover, it seems that the performance of CGP-LEX converges to a lower

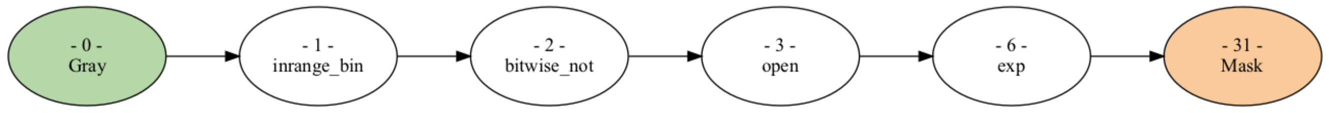

Figure 5: Final program (randomly selected) evolved by CGP.

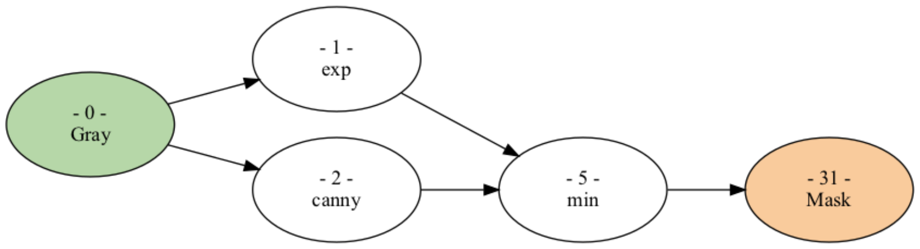

Figure 6: Final program (randomly selected) evolved by CGP-LEX.

value than the traditional method, although they both seem to reach good IOU metrics. This overall high performance of both methods suggests that the dataset considered in our experiments might not challenge the algorithms and might hide potential differences among the algorithms.

Figure 4 illustrates the mean number of active nodes (shaded areas show the standard deviation) over the iterations of the 10 runs of CGP and CGP-LEX. We can see that CGP keeps finding bigger problems during the search progress. These results, together with the IOU values discussed earlier indicate that even if larger programs can perform better, in this scenario, there is no big advantage in having them. On the contrary, CGP-LEX keeps exploring programs with smaller sizes, to about half of the size of the final programs evolved by CGP.

Finally, we briefly comment on the final programs evolved by the CGP variants considered in this study, shown in Figures 5 and 6. Interestingly, the size of the program returned by CGP is not much different than the program returned by CGP-LEX, which indicates that there is a high potential on exploring small-size programs that are present in both variants, although reinforced by CGP-LEX. Again, we remember the reader that the simplicity of the dataset could be hiding a more diverse behavior between CGP and CGP-LEX.

7 CONCLUSION

The aim of this work was to introduce the ϵ-Lexicase selection in CGP and to verify if this selection method was able to keep the performance of CGP competitive in comparison with the standard CGP while reducing the size of the programs generated. Our results showed that there exists a clear influence of ϵ-Lexicase selection in CGP.

Overall, this study strengthens the idea that controlling the size of the programs generated with CGP with a control mechanism might help the generalisation of simpler, easier-to-understand programs with no strong drawback. That suggests that using ϵ-Lexicase selection in CGP is a promising direction to further be explored, especially in the context of interpretability, since CGP-LEX can generate programs that can be understood, evaluated and used as a base of analysis.

A natural progression of this work is to investigate the introduction of other explicit size control mechanisms and the evaluation of the CGP variants in more challenging datasets. Furthermore, we aim to investigate the real impact of the ability of CGP-LEX to help us better interpret evolved programs.

ACKNOWLEDGEMENTS

Anonymized acknowledgements. This work was is funded by the Laboratoire d'Excellence Toulouse Cancer TOUCAN (contract ANR11-LABX). This work was also supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under Grant agreement no. ANR-19-PI3A-0004.

REFERENCES

[1] Arbab Masood Ahmad, Gul Muhammad Khan, Sahibzada Ali Mahmud, and Julian Francis Miller. 2012. Breast Cancer Detection Using Cartesian Genetic Programming Evolved Artificial Neural Networks. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation* (Philadelphia, Pennsylvania, USA) *(GECCO '12)*. Association for Computing Machinery, New York, NY, USA, 1031–1038. https://doi.org/10.1145/2330163.2330307

[2] Julien Biau, Dennis Wilson, Sylvain Cussat-Blanc, and Hervé Luga. 2021. Improving Image Filters with Cartesian Genetic Programming.. In *IJCCI*. 17–27.

[3] Kévin Cortacero, Brienne McKenzie, Sabina Müller, Roxana Khazen, Fanny Lafouresse, Gaëlle Corsaut, Nathalie Van Acker, François-Xavier Frenois, Laurence Lamant, Nicolas Meyer, Béatrice Vergier, Dennis G. Wilson, Hervé Luga, Oskar Staufer, Michael L. Dustin, Salvatore Valitutti, and Sylvain Cussat-Blanc. 2023. Kartezio: Evolutionary Design of Explainable Pipelines for Biomedical Image Analysis. arXiv:2302.14762 [cs.CV]

[4] Allan de Lima, Samuel Carvalho, Douglas Mota Dias, Enrique Naredo, Joseph P. Sullivan, and Conor Ryan. 2022. Lexi2: Lexicase Selection with Lexicographic Parsimony Pressure. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 929–937. https://doi.org/10.1145/3512290.3528803

[5] Simon Harding, Jürgen Leitner, and Jürgen Schmidhuber. 2013. *Cartesian Genetic Programming for Image Processing*. Springer New York, New York, NY, 31–44. https://doi.org/10.1007/978-1-4614-6846-2_3

[6] Yifan He, Claus Aranha, Antony Hallam, and Romain Chassagne. 2022. Optimization of subsurface models with multiple criteria using Lexicase Selection. *Operations Research Perspectives* 9 (2022), 100237. https://doi.org/10.1016/j.orp.2022.100237

[7] Thomas Helmuth and Lee Spector. 2013. Evolving a Digital Multiplier with the Pushgp Genetic Programming System. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation* (Amsterdam, The Netherlands) *(GECCO '13 Companion)*. Association for Computing Machinery, New York, NY, USA, 1627–1634. https://doi.org/10.1145/2464576.2466814

[8] Thomas Helmuth, Lee Spector, and James Matheson. 2015. Solving Uncompromising Problems With Lexicase Selection. *IEEE Transactions on Evolutionary Computation* 19, 5 (2015), 630–643. https://doi.org/10.1109/TEVC.2014.2362729

[9] Roman Kalkreuth, Günter Rudolph, and Jorg Krone. 2016. More efficient evolution of small genetic programs in Cartesian Genetic Programming by using genotypie age. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. 5052–5059. https://doi.org/10.1109/CEC.2016.7748330

[10] William La Cava, Lee Spector, and Kourosh Danai. 2016. Epsilon-Lexicase Selection for Regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016* (Denver, Colorado, USA) *(GECCO '16)*. Association for Computing Machinery, New York, NY, USA, 741–748. https://doi.org/10.1145/2908812.2908898

[11] Julian F. Miller. 1999. An Empirical Study of the Efficiency of Learning Boolean Functions Using a Cartesian Genetic Programming Approach. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2* (Orlando, Florida) *(GECCO'99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1135–1142.

[12] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. 2019. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[13] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence* 1, 5 (2019), 206–215.

[14] Lee Spector. 2012. Assessment of Problem Modality by Differential Performance of Lexicase Selection in Genetic Programming: A Preliminary Report. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation* (Philadelphia, Pennsylvania, USA) *(GECCO '12)*. Association for Computing Machinery, New York, NY, USA, 401–408. https://doi.org/10.1145/2330784.2330846

[15] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2020. *Designing Convolutional Neural Network Architectures Using Cartesian Genetic Programming*. Springer Singapore, Singapore, 185–208. https://doi.org/10.1007/978-981-15-3685-4_7

[16] Andrew James Turner and Julian Francis Miller. 2014. Cartesian Genetic Programming: Why No Bloat?. In *Genetic Programming*, Miguel Nicolau, Krzysztof Krawiec, Malcolm I. Heywood, Mauro Castelli, Pablo García-Sánchez, Juan J. Merelo, Victor M. Rivas Santos, and Kevin Sim (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 222–233.

[17] Dennis G Wilson, Sylvain Cussat-Blanc, Hervé Luga, and Julian F Miller. 2018. Evolving Simple Programs for Playing Atari Games. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Kyoto, Japan) *(GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 229–236. https://doi.org/10.1145/3205455.3205578