BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2026
http://excel.fit.vutbr.cz

Transformer-Guided Mutation in Cartesian Genetic Programming for Approximate Multiplier Design

Ondřej Galeta*

Abstract
Cartesian Genetic Programming (CGP) for approximate circuit design suffers from poor scalability due to expensive evaluations. This work proposes a transformer-guided mutation operator to accelerate the design of approximate multipliers. A BERT-based model predicts where and how to mutate circuit representations, supported by dataset filtering, augmentation, and a fallback to standard mutation. Results on EvoApprox8b show faster convergence and improved solutions over standard CGP for some error thresholds. The approach improves CGP speed of convergence, creates new potentially patentable designs, and demonstrates the potential of combining evolutionary design with machine learning.

*xgalet06@vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Approximate computing is a design approach that intentionally allows inaccuracies in computations to get significant improvements in performance, energy efficiency, or hardware cost. Approximate circuits representing operations can be used, for example, in image filters or neural networks. One of the possible ways to design approximate circuits is Cartesian Genetic Programming (CGP), whose capabilities were proven in many papers, including [1, 2].

The problem of CGP lies in scalability because approximating larger functions leads to a more complex evaluation that grows in exponential time. This work focuses on an approach that uses a BERT-based transformer as a function that provides a probability distribution, followed by a mutation operator. In other words, the proposed transformer model provides advice on where and by what a given place in the chromosome should be replaced.

Apart of the model, the given work proposes a preparation technique for a training dataset and control mechanism of CGP transformer based mutation that provides more robust way of solution space exploration.

The whole method is evaluated on the EvoApprox8b [1] dataset, consisting of approximate unsigned 8-bit multipliers, and compared to the standard CGP approach.

2. Standard CGP for Circuit approximation

This work builds on Cartesian Genetic Programming (CGP) [3], an evolutionary algorithm used to automatically evolve combinational circuits. In CGP, each possible solution is called a chromosome and represents one combinational circuit as shown in Fig. 1. In every iteration, called a generation, a new set of chromosomes (also known as a population) is created using the mutation operator. The mutation operator performs small randomized changes in the chromosome that lead to the creation of a new one. Each chromosome is evaluated by a fitness function that represents the quality of the represented combinational circuit.

Many specific design choices are not defined by CGP itself. For example, our base solution follows Mrazek et. al. [2] that uses as an initial population accurate multiplier, fitness function presented as Eq. 1, where $\widetilde{M}$ is given circuit, ϵ is threshold value for WCE, WCE worst case error defined as maximal deviation from accurate multiplier, $\mathrm{WCE}_{\mathrm{zr}} = 0$ states that all zero multiplications must be accurate and $\mathrm{cost}(\widetilde{M})$ is approximation of area needed to create such multiplier in silicon, and creating a 4 offsprings each generation from the best available solution in population.

3. Proposed Method

The key idea of this work is to use CGP with a mutation operator probabilistically led by a transformer to design approximate multipliers as shown in Fig. 3. Such an approach gives three main questions: how to encode a chromosome into a transformer, how to prepare the dataset, select the most relevant circuits, and augment to obtain the most informative dataset, and how to adjust the control of CGP itself to take into account the fact that our transformer-based mutation can fail and be less efficient than a standard one.

3.1 Transformer-based Mutation

The transformer scheme can be seen in Fig. 2. The architecture is BERT-based. The deviation from the standard BERT architecture [4] lies in the fact that, prior to computing attention between chromosome nodes, each node embedding is first influenced by the embeddings of the nodes it directly depends on. This embedding influence layer is calculated as shown in Eq. 2.

The chromosome is fed to the model as a series of node triplets representing an operation and two inputs. Transformer encodes all elements of a triplet separately and concatenates them to a single embedding. The output is then a separate probability distribution for each element in node triplets that refers to the most probable changes that would lead to better fitness according to the model, and a probability distribution over the whole vector that indicates which nodes the mutation should most probably happen.

3.2 Training Data and Transformer Training

The initial multiplier EvoApprox8b [1] dataset needed to be filtered and preprocessed. Based on preliminary results, it is beneficial to filter out all multipliers from the dataset that do not fulfill constraints for fitness given in Eq. 1 (cases where fitness is ∞) as shown in Fig. 4. This figure also illustrates the method used to assign multipliers a weight based on their distance to the Pareto front, reflecting the optimal tradeoff between error and hardware area.

To increase the strength of the learning signal during training, augmentation was implemented. In this case, augmentation means creating different chromosomes that represent the same combinational circuit by shuffling the positions of nodes in a grid in a way that the behavior of the represented multiplier remains unchanged. It helps the transformer to "understand" the syntax of CGP chromosomes.

To compute data used for training a mutation probability vector for active nodes in a chromosome, random mutation is performed, and a new value of WCE is calculated for the multiplier created by that mutation and compared to the original WCE. This process identifies the most error-critical nodes that are present in the chromosome and tries to approximate which nodes are most sensitive to mutations that negatively affect WCE.

The model was then trained for 20 epochs for different maximal WCE thresholds, always using about 10^4 circuits after filtration.

3.3 CGP with a Transformer-Based Mutation

Since a transformer model provides only an approximation of the true probability distribution, a new control mechanism was developed. The algorithm switches to standard mutation if no fitness improvement occurs for a set number of generations, using a stagnation counter that resets with each new best chromosome. This serves as a backup when the model gives misleading distribution information that could stall the design process.

4. Results

The results can be seen in Fig. 5, where we can see in each timestep a boxplot representing the best 10% of 576 runs of the given algorithm under the given parameters. Proposed transformer-based mutation operator in graph referred as CGP-TR converges faster to better fitness than standard CGP for $\epsilon = 3\%$ and $\epsilon = 5\%$ (for the meaning of ϵ see Section 2) with statistical significance. For $\epsilon = 4\%$, the statistically significant improvement was not observed.

As shown in Fig. 6, results from CGP-TR were compared with the original EvoApprox8b [1] library and showed better results for most of the ϵ tried.

5. Conclusion

It was proven that the proposed method of using a transformer-based mutation operator for approximating multipliers can be better than standard mutation, as they improve convergence speed and creates new potentially patentable designs. This opens a new path to overcoming scalability issues for CGP, leading to designing more complex solutions in a shorter time.

Acknowledgements

I would like to sincerely thank my supervisor, Prof. Ing. Lukáš Sekanina, Ph.D., for his guidance, support, and willingness to help shape this thesis into a publishable article.

References

[1] V. Mrazek, R. Hrbacek, et al. Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In *Proc. of DATE'17*, pages 258–261, 2017.

[2] Vojtech Mrazek, Lukas Sekanina, and Zdenek Vasicek. Libraries of approximate circuits: Automated design and application in cnn accelerators. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(4):406–418, 2020.

[3] Julian F Miller. Cartesian genetic programming. *European Conference on Genetic Programming*, pages 121–132, 1999.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, volume 1, pages 4171–4186. Association for Computational Linguistics, June 2019.