

# Submission Abstract

## 0.1 Algorithm

In this work, I propose a new algorithm named Symbolic Regression Multifactorial Evolutionary Algorithm (SymMFEA). This algorithm leverages the Multifactorial Evolutionary Algorithm (MFEA) to enhance Genetic Programming (GP). It has been demonstrated [1]–[3] that finding and storing building blocks (the term refers to small, compact patterns that can be combined into more complex expression) can improve Genetic Programming’s performance in Symbolic Regression problem significantly.

Therefore, I divide the GP population into multiple subpopulations in MFEA’s fashion by constraint each subpopulation by one or several aspects:

- **Search space:** Each subpopulation has different constraints about the size (maximum length and maximum depth) of the solutions. While the subpopulations with compact solutions are responsible for finding the building blocks, the subpopulations which are allowed to have complex solutions can leverage these blocks and combine them for more accurate solutions.
- **Subset of the Dataset:** Bagging [4], also known as bootstrap aggregation, is an ensemble learning method that is used to reduce the variance of the model prediction. The idea of the Bagging method is to ensemble multiple versions of the same predictor, which are trained with different subsets of the original dataset. By ensembling these predictors, they can correct individual mistakes potentially caused by overfitting in particular fragments of the dataset. Applying this method in the algorithm, each subpopulation also has access to different subsets of the dataset (data subsampling) to prevent the algorithm from overfitting. Furthermore, subsampling the dataset linearly reduces the runtime of the solution evaluation, which takes the majority of the overall runtime.
- **Terminal sets:** Using random features (terminals) were reported to improving the model accuracy [5]. Hence, I limit to each population to used only a random subset of terminals.

By regulating the population as above, the algorithm creates synthetic factorials from the original problem, which are assumed to be able to provide the algorithm with useful knowledge about the origin problem. SymMFEA’s full pipeline is illustrated in Figure 0.1. Other components of the algorithm is indicated in table 1.

**Note:** I did not use any preprocessing and postprocessing and only the Sym-MFEA algorithm for all three datasets.

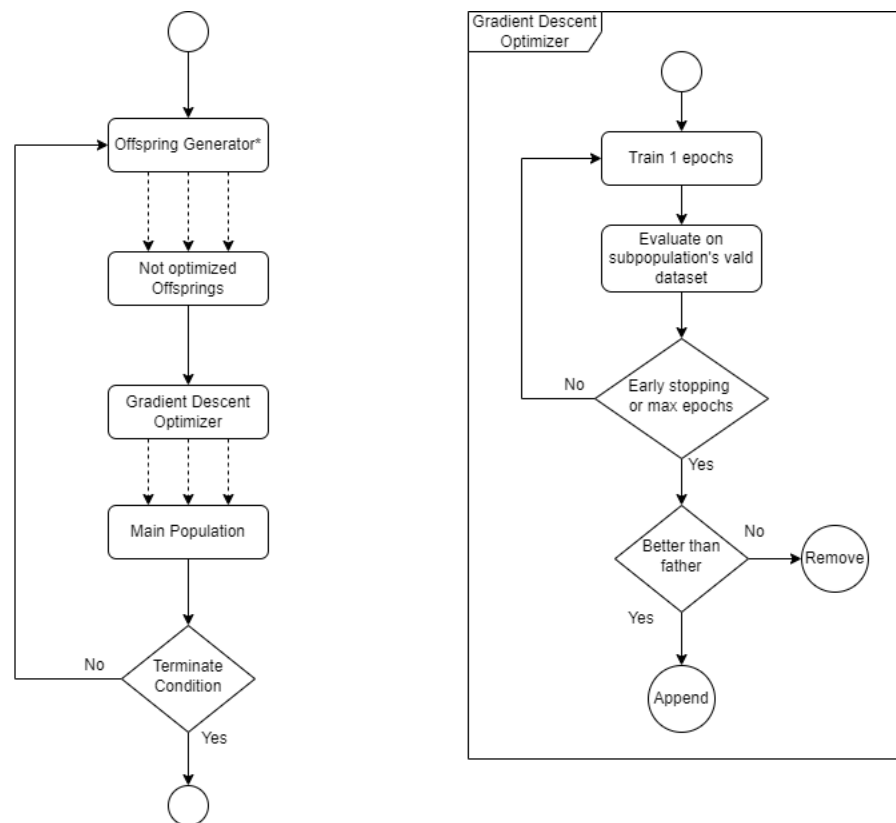
**Table 1:** Algorithm components


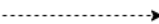
Components	
Local Search	Gradient Descent
Expression	Operon's Linear Expression [6]
Crossover	Subtree Crossover
Mutation	Variable Mutation and Grow Tree Mutation

## 0.2 Contact Information

This is my contact information:

- **Name:** Hai Minh Nguyen
- **Affiliation:** School of Information and Communications Technology, Hanoi University of Science and Technology (HUST)
- **Email:** minh.nh194120@sis.hust.edu.vn
- **Personal Email:** haiminhnguyen2001@gmail.com
- **Phone** (+84) 839865565



Symbols	
	Sequential Process
	Concurency Process

**Figure 0.1:** SymMFEA pipeline

## REFERENCE

- [1] F. O. de Franca and G. S. I. Aldeia, “Interaction–transformation evolutionary algorithm for symbolic regression,” *Evolutionary computation*, vol. 29, no. 3, pp. 367–390, 2021.
- [2] L. Trujillo, L. Muñoz, E. Galván-López and S. Silva, “Neat genetic programming: Controlling bloat naturally,” *Information Sciences*, vol. 333, pp. 21–43, 2016.
- [3] M. Virgolin, T. Alderliesten, C. Witteveen and P. A. N. Bosman, “Improving model-based genetic programming for symbolic regression of small expressions,” *Evolutionary Computation*, vol. 29, no. 2, pp. 211–237, 2021.
- [4] L. Breiman, “Bagging predictors,” *Machine learning*, vol. 24, pp. 123–140, 1996.
- [5] L. Breiman, “Random forests,” *Machine learning*, vol. 45, pp. 5–32, 2001.
- [6] B. Burlacu, G. Kronberger and M. Kommenda, “Operon c++: An efficient genetic programming framework for symbolic regression,” Jul. 2020, pp. 1562–1570. DOI: 10.1145/3377929.3398099.

# Appendices