Performance Improvement in Genetic Programming using Modified Crossover and Node Mutation

Arpit Bhardwaj
Indian Institute of Technology Indore
Indore, India
phd12110102@iiti.ac.in

Aruna Tiwari
Indian Institute of Technology Indore
Indore, India
artiwari@iiti.ac.in

ABSTRACT
During the evolution of solutions using Genetic Programming (GP) there is generally an increase in average tree size without a corresponding increase in fitness—a phenomenon commonly referred to as bloat. Bloating increases time to find the best solution. Sometimes, best solution can never be obtained. In this paper we are proposing a modified crossover and point mutation operation in GP algorithm in order to reduce the problem of bloat. To demonstrate our approach, we have designed a Multiclass Classifier using GP by taking few benchmark datasets. The results obtained show that by applying modified crossover together with modified node mutation reduces the problem of bloat substantially without compromising the performance.

Categories and Subject Descriptors
I.2 [Artificial Intelligence]: Problem Solving, Control Methods, and Search

General Terms
Performance.

Keywords
Genetic Programming, Bloat, Modified Crossover.

1. INTRODUCTION
Genetic Programming (GP) [1] is an evolutionary algorithm-based methodology inspired by biological evolution to find computer programs that perform a user-defined task. Crossover (sexual recombination) is recognized as the primary genetic operator for improving program structures in tree-based GP [3]. It plays vital role in improving GP process. Mutation affects an individual in the population. It replaces a whole node in the selected individual, or it can replace just the node's information [4]. Bloat is a well-known phenomenon in GP [2]. An individual program in GP could be in any size. Such flexibility in representation provides more freedom in searching solutions, but at the same time it causes the bloat problem.

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

2. Proposed Work
To control the problem of bloat we have proposed modified crossover and node mutation operation.

2.1 Modified Node Mutation
In our approach, to control bloat, we use modified node mutation instead of sub tree mutation. In the modified node mutation technique, we replace the randomly selected function/terminal node of the parent with some randomly generated function/terminal in order to provide some diversity among the individuals. If the fitness of the children is better than the parent then only, we transfer the children to the next generation otherwise, we reject the children and perform the mutation once again and repeat this process till we get the children which is better than the parent. If for repeating this process, for all the nodes we don't get the better children then, we replace two nodes rather than one and generate the children. Continue this process till we get the required result.

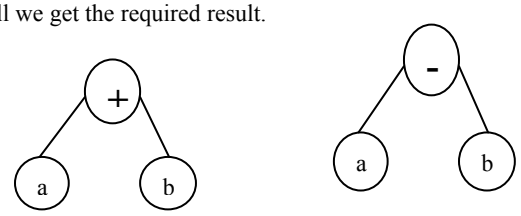

Figure 1. Node Mutation

2.2 Modified Crossover
We use the double tournament scheme for selecting chromosomes for the crossover operation. The two individual which are selected from the double tournament (say C_1 and C_2) perform the Modified Crossover. Each of C_1 and C_2 has c trees $T_1, T_2, ...T_c$. Out of these c trees one is selected on the basis of fitness, the tree with the highest fitness is chosen to perform the crossover. The two individual which are selected we call them P_1 and P_2, will generate the four children ch_1, ch_2, ch_3 and ch_4 by randomly interchanging the nodes. From these four generated children two are rejected on the basis of depth and size, i.e. children with the smaller depth and size are selected say ch_1, and ch_3 are selected because they have smaller size and depth than ch_2 and ch_4. Now we apply elitism, in this we compare the fitness of ch_1 and ch_3 with the fitness of P_1 and P_2. If the fitness of child is better than the parent we transfer the children to the next generation, otherwise repeat this process till we get the better solution. After checking all the possible combination if we do not get the better children than we transfer the P_1 and P_2 to the next generation rather ch_1 and ch_3. In

this way, we can ensure that by using modified crossover technique we reduce the size and depth limit of the generated children and also, by applying the elitism we can ensure that we always transfer the good building block to the next generation. Thus, we can get the best classifier in less time and this consequently reduces the problem of bloat.

In Figure 2, we have shown the modified crossover technique, we had generated the four children ch_1, ch_2, ch_3 and ch_4 from the two parents.

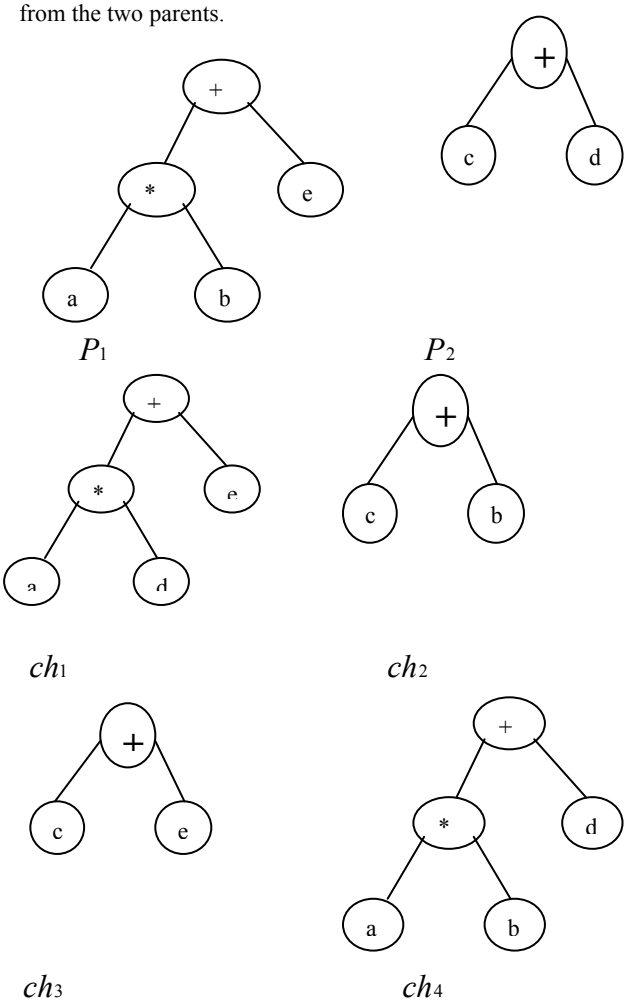

Figure 2. Modified Crossover

2.3 Experiments and Results

For the proposed classifier, datasets are tested using 10-fold-cross-validation method. Training time and Classification accuracy is measured for every datasets as given in Table 1 and Table 2.

We have compared the outcome of our results with the conventional crossover and mutation method and with FEDS crossover and point mutation technique [2] shown in Table 2.

TABLE 1. GP RUN TIME

Data Sets	IRIS	WBC	BUPA
TIME (hour:min:sec)	0:0:13	0:1:52	0:2:23

TABLE 2. COMPARISON OF CONVENTIONAL CROSSOVER, MUTAION, FEDS CROSSOVER, NODE MUTATION METHOD WITH MODIFIED CROSSOVER AND MODIFIED POINT MUTATION METHOD

NAME OF DATASETS	Conventional Crossover and Mutation Method	FEDS crossover and Node Mutation Method	Modified Crossover and Modified Point Mutation Method
	Generalization Accuracy	Generalization Accuracy	Generalization Accuracy
IRIS	81.56%	97.42%	98.12%
WBC	83.64%	85.56%	87.14%
BUPA	64.93%	66.89%	69.32%

We reduced the problem of bloat and improved the results using modified crossover. In conventional crossover, there is no limit to the average size of the tree and no restriction according to the fitness. In our crossover we are applying the limit to the size and depth of the individuals thus saving the space. We are also applying the local elitism replacement on the individuals so that only the best individual can go to the next generation. In our crossover, we get a smaller tree size because we apply the size limit on each generated children. The tree with the smaller size and depth with higher fitness can only proceed to the next generation. Due to this, the size of the trees in our crossover has reduced tremendously which helped in controlling bloat and improved the performance of the classifier designed.

3. CONCLUSION

A GP based classifier is designed based on [2], [4] for further improvement in the performance. In the proposed work, problem of code bloat is controlled using modified crossover and node mutation operations. The same is being tested on various benchmark datasets using 10-fold-cross-validation technique. It is observed that our method outperforms the previous bloat control methods [2]. Our proposed method reduces the size and depth of the chromosomes. It also improves the training and generalization accuracy of the classifier as represented in Table 3. Our proposed method also reduces the training time of the individual. The same is verified by our results with the experiments conducted.

4. REFERENCES

[1] J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA: MIT Press, 1992.

[2] A. Purohit, A.Bhardwaj, A.Tiwari, N.S.Choudhari, "Removing Code Bloating In Crossover Operation In Genetic Programming", IEEE ICRTIT, June 2011.

[3] P. Nordin, F. Francone, and W. Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In J. P. Rosca, editor, Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, pages 6–22, 1995.