Controlled Genetic Programming Search for Solving Deceptive Problems

Emin Erkan Korkmaz
Department of Computer Engineering
Middle East Technical University
Ankara-Turkey
korkmaz@ceng.metu.edu.tr
+(90) − 312 − 210 − 5536

Göktürk Üçoluk
Department of Computer Engineering
Middle East Technical University
Ankara-Turkey
ucoluk@ceng.metu.edu.tr
+(90) − 312 − 210 − 5584

Abstract

Traditional GP randomly combines subtrees by applying crossover and mutation. There is a growing interest in methods that can control such recombination operations. In this study a new approach is presented for guiding the recombination process for GP. Our method is based on extracting the global information of the promising solutions that appear during the genetic search. The aim is to use this information to control the crossover operation afterwards.

1 Introduction

It is clear that the random recombination used in traditional GP can easily disturb the building blocks. An attempt based on determining the beneficial building blocks and preventing them to be disturbed during the recombination operations can be helpful. However for deceptive class of problems such an approach is questionable. The interaction between the partial solutions is high for these problems. The global meaning of finding a possible solution goes beyond determining isolated, non-interacting building blocks and bringing them together. In this study a different approach which focuses on the global information of promising solutions is presented. The aim is to extract the knowledge of what it is to be good globally and hence perform the right crossover operations which would keep the search among the localization of well-fit elements afterwards.

The proposed method has been applied to two different domains which are *Context-Free Grammar Induction* and *N-Parity Problem*. Both of the domains can be considered as highly deceptive. Traditional GP has exhibited quite a low performance for both of the problems. In the following section an overview of various approaches in the area are given. In section 3 our approach is presented in detail. In section 4 the application of our approach on CFG induction is given. Then in section 5 the N-Parity problem is analyzed in the light of our approach and in the last section conclusions and discussions are presented.

2 Related Work

Researchers have been interested in controlling recombination in GP. For instance [2] proposes a method called *Recombinative Guidance for GP*. The method is based on calculating the performance values for subtrees of a GP tree during evolution and then applying recombination operators so that the subtrees with high performance are not disturbed. On the other hand [11] uses a knowledge repository which is expected to guide the search towards better solutions. The knowledge repository collects code segments from the genetic population together with some associated information like fitness, number of occurrences, depth and so on. [11] proposes a method to calculate a single score for each segment that would reflect its overall contribution for the current task. The evolution proceeds by adding new code segments with high performance to the knowledge repository and excluding the ones which are subject to performance loss.

Similar approaches trying to control the recombination operators could be found in the area of Genetic Algorithms too, [1, 6, 7].

The attempts presented are usually based on determining the important building blocks and preventing them to be disturbed by the recombination operations. However [8] states that for some functions even if it is possible to decompose the function into some components, the subfunctions could interact. In such a case it becomes impossible to consider each subfunction independently, optimize it and then obtain the optimum by combining the partial solutions.

For this set of problems it is clear that an attempt based on determining building blocks is not expected to increase the performance a lot. Therefore our research has focused on analyzing the global information of well-fit elements which are expected to represent dependencies of subparts in a GP-tree and which could provide clues to increase the performance of GP for deceptive problems too.

3 Extracting the Global Information

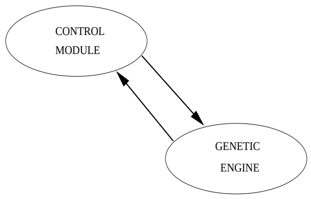

Figure 1: The dual structure proposed.

In order to process the global information, we have designed a new module called *Control Module*. Figure 1 displays the dual structure of this system. The genetic engine which can be considered as the base structure, performs the standard genetic search. The control module as a super structure, keeps an eye on the search carried out by the genetic engine. It focuses on the global information of the chromosomes and performs a meta-level learning at certain periods to determine what it is to be good globally. Once the first learning process takes place, the control module starts sending feedback to the genetic engine about the consequences of possible crossover operations. Then, the genetic engine chooses the most appropriate crossover points by using the feedback it receives.

For this scheme a new representation which would enable the control module to process the 'global' information of the chromosomes, is needed. The solution we propose is to map chromosomes to single points in $\mathcal{R}^n$.

It is this mapping which reflects what is considered as global to a tree. It is thought that the frequency information of the elements in a chromosome and how they are distributed on the tree might form a beneficial global picture for the structure at hand.

This is a simple formalization about the global organization of the tree which does not have a heavy computational load. The method we have used is mapping the terminal and function elements to base vectors and then using a bottom up construction to obtain a single vector for the whole GP-tree. A leaf node is only mapped to its base vector while the vector for an internal node is obtained by adding the vectors of its children plus the base vector corresponding to it.

On the other side, the depth information is reflected in the vector as a fractional value to make a distinction with the frequency information.

In general terms, if $P(C_1, C_2, ..., C_n)$ is any subpart of a chromosome, then the vector that would correspond to P can be obtained using the following formula.

$$V_P = V_{C_1} + V_{C_2} + ... + V_{C_n} + V_{P_{base}} + V_{P_{base}} * 0.01 * depth(P) \quad (1)$$

For instance, consider the function and terminal sets; $F = \{+, -, *, /\}$ and $T = \{x\}$. The base vectors would be:

- $V_+ = [0, 0, 0, 0, 1]$
- $V_- = [0, 0, 0, 1, 0]$
- $V_/ = [0, 0, 1, 0, 0]$
- $V_* = [0, 1, 0, 0, 0]$.
- $V_x = [1, 0, 0, 0, 0]$.

Note that the dimension of the vectors is determined as the total number of function and terminal elements.

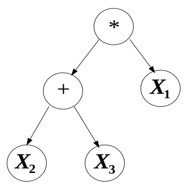

Figure 2: A sample chromosome.

For the tree in figure 2, the vector construction mechanism will be as follows. The base vectors are as specified above. The three different usages of the terminal element X are labeled as X_1, X_2, and X_3. Since X_1 and X_2 have the same depth value, their vectors will be the same. This vector would be $[1.02, 0, 0, 0, 0]$. On the other side, the vector corresponding to X_3 will be $[1.01, 0, 0, 0, 0]$ due to the depth value of 1. According to equation 1, the vector of $'+'$ will be $[2.04, 0, 0, 0, 1.01]$. Lastly, the vector corresponding to $'*'$ which would be the vector of the whole tree, can be obtained by using the vectors of $'+'$ and X_3 this time. Hence, $V_* = [3.05, 1, 0, 0, 1.01]$. Note that, each dimension of this vector provides information about the usage of a terminal or a function element. For instance the first dimension is reserved for the terminal element X. The value in this dimension denotes that the terminal element is used three times and the sum of the depths of these three different usages is five. On the other side the second dimension denotes that $'*'$ operation is only used for once as the root node of the tree.

Note that the constant value that is used to transform the depth value into a fractional one in equation 1 is 0.01. However if the sum of the depths exceeds 100, depth and frequency information will interfere with each other. If this is possible, a smaller constant has to be used. At least, it should be guaranteed that the number of such ill formed vectors are kept small enough that the learning process does not get affected.

Also note that different chromosomes can be mapped to the same vector. However this is not contradictory with our assumption since different elements in the base structure could be similar in terms of the super structure.

3.1 Using the Global Information

The interaction between the genetic engine and the global module is as follows. For each chromosome in the population, the corresponding vector is formed and sent to the control module together with the fitness value. The control module collects the vectors and fitnesses for a certain period of generations, which we call the *learning period*. Then the average and the standard deviation of the fitness values are calculated.

The control module forms the training set using the elements with fitness values deviating from the average more than the standard deviation. The ones with positive deviation are marked as positive examples and the others as negative. The *"C4.5, Decision Tree Generator"* is used to generate the abstraction over the test set. Then for each crossover operation to be performed, the genetic engine sends to the control module three different alternative crossover points. The control module predicts if the alternative offsprings will be in the positive class or in the negative class by using the abstraction made so far. The best alternative is chosen by the genetic engine and the learning process is repeated periodically.

Using a certain percentage of the best and the worst elements could be another method to form the test set. However it is observed that using standard deviation provides a flexibility for the control module. Sometimes it is possible for the control module to guide the genetic search to a local minima. In such a case the standard deviation decreases a lot and no positive examples could be found for the training set. In such a situation since nothing could be learnt, crossover becomes random again. This makes it more probable to escape from the local minima since recombination is not controlled. However determining a percentage of the examples as positive always, looks like insisting on the mistake that the control module has made.

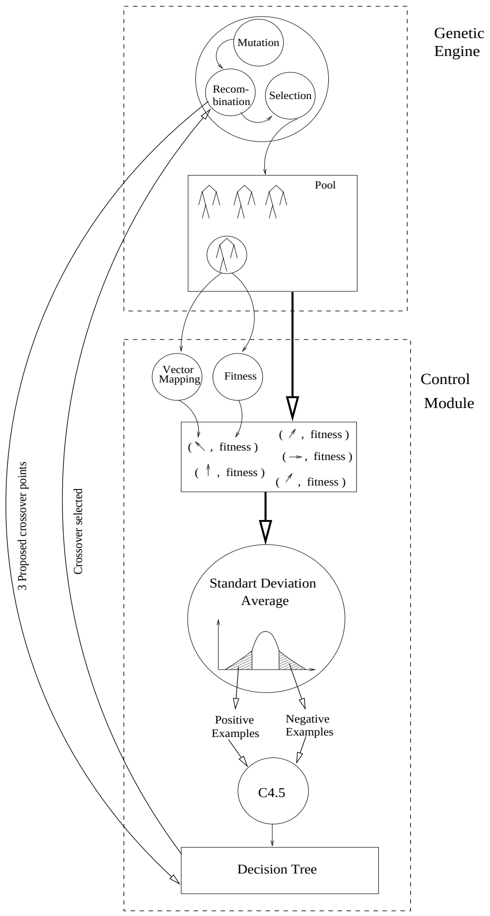

Figure 3: Interaction between the Control Module and the Genetic search.

4 $TESTBED_1$: Context Free Grammar Induction

Natural language sentences have been used in order to form the training set for the CFG-induction problem. The training set consists of 21 positive examples and 17 negative examples. The sentences formalize a subset of English including sentences consisting of structures like NP, VP and PP. The Noun phrase (NP) is quite simple and consists of a determiner (D) followed by a noun (N) or compound noun. On the other hand, the verb phrase (VP) can be intransitive, transitive or ditransitive and the prepositional phrase

(PP) could be attached to VP or NP. The aim is to induce a CFG that can parse the positive examples and reject the negative ones. Each chromosome in the population is a candidate grammar and the details of this representation can be found in [5].

The problem can be considered as a highly deceptive one. It is possible to divide a grammar into subparts like NP, VP or PP, however these subparts do not have clear borders. Overlapping exists due to the fact that NP is a part of VP and PP.

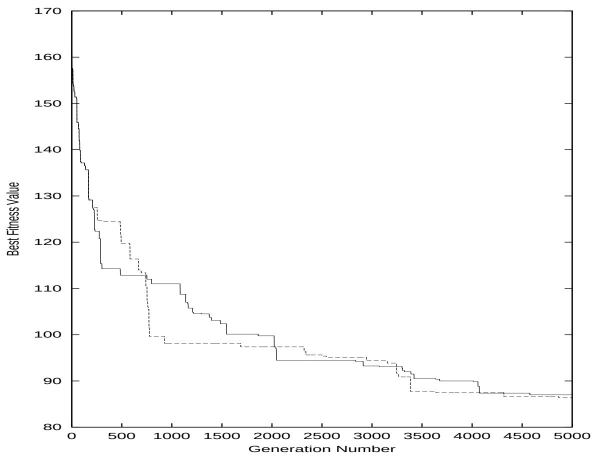

Figure 4: Comparison of controlled search and normal run for the CFG-induction problem. The dashed lines denote the performance of controlled search. Learning period is 200.

The fitness function used is the standard one. For grammar G, if S is the set of sentences consisting of the positive examples that G cannot parse and the negative examples that G parses, then the fitness of G is defined as:

$$F(G) = \sum_{S_i \in S} SENTENCELENGTH(S_i) \quad (2)$$

So the aim is to minimize the fitness function. For the test data the worst fitness for a grammar could be 243 which is the sum of the length of all sentences both in the positive and the negative set. And the best fitness is certainly zero which can be achieved when a grammar parses all of the examples in the positive set and parses none of the ones in the negative set.

The grammar evolved is subject to only one restriction. The number of right hand side elements in a grammar could be at most two. The terminal and function sets are $T = \{D, N, V, P\}$ and $F = \{X_1, X_2, ..., X_{10}\}$. Considering the restriction specified above each element of the function set could have one or two arguments.

The mapping process described in section 3 is used to form the vectors for the control module. Since the total number of elements in the function and the terminal set is 14, the vectors will be formed in $\mathcal{R}^{14}$.

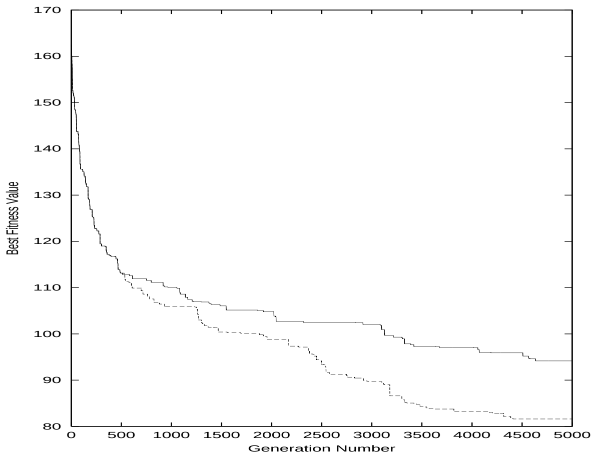

Figure 5: Comparison of controlled search and normal run for the CFG-induction problem. The dashed lines denote the performance of controlled search. Learning period is 500.

For the first trial the learning period has been set as 30. The genetic parameters used for the trials are as follows:

- Population size = 100
- Crossover at function point fraction = 0.1
- Crossover at any point fraction = 0.7
- Reproduction fraction = 0.1
- Mutation fraction = 0.1
- Number of Generations = 5000

Both the controlled search and the straightforward application of GP have been run using eight different random seeds. Surprisingly it has been observed that the controlled search performed worse than the straightforward application. It seems that the information sent by the control module to the genetic engine was misleading and directed the search to a local minima resulting a performance worse than random crossover. An increase in the performance had been obtained with simpler data and with smaller number of function elements. The details of this initial attempt can be found in [5]. The main difference with this initial attempt is the total number of function and terminal elements used. This total number is 14 for this new setup. Therefore it is thought that the data collected with the learning period of 30 might be quite low for making a reasonable abstraction over vectors with this dimension. On the other side, we have observed that the decision trees induced for this case are simple and contain less information. Therefore it has been decided to increase the learning period.

Figure 4 presents the comparison with the straightforward application of GP when the learning period is increased to 200. Again the results denote the average of eight runs with different random seeds. The performance of the controlled search clearly increased, compared to the trial with a learning period of 30. However still it is not the case that controlled search can outperform the straightforward application.

However the increase in the performance parallel to the increase in the learning period is encouraging. Therefore another trial has been been carried out with a learning period of 500 generations this time . Figure 5 presents this new trial. This time the average of twenty different runs are used in order to increase the liability of the performance increase obtained. As it can be seen in the figure, the desired performance increase has been obtained.

5 $TESTBED_2$: N-Parity Problem

The N-parity problem has been selected also in order to analyze our approach. The aim is to induce a function consisting of internal operators $AND, OR, NAND$ and NOR which takes a binary sequence of of length n and returns true if the number of ones in the sequence is odd and false otherwise.

The problem is to our interest as it is highly deceptive. [3] states that the problem quickly becomes more difficult with increasing order. He also denotes that flipping any bit in the sequence inverts the outcome of the parity function and notes this as a fact to denote the hardness of the problem.

The 5-parity problem has been chosen for the test cases since [3] denotes that no solutions is found by basic GP for the 5-parity.

The function and the terminal sets are $F = \{AND, OR, NAND, NOR\}$ and $T = \{X_1, X_2, X_3, X_4, X_5\}$. T represents the binary input sequence of length five. The number of possible input binary sequences is 32 for the 5-parity problem. The fitness function simply adds a penalty of one if the induced function returns the wrong answer for an input sequence. Hence, the fitness value may range between 0 and 32.

The genetic parameters used for the trials are as follows:

- Population size = 100
- Crossover at function point fraction = 0.1
- Crossover at any point fraction = 0.7
- Reproduction fraction = 0.1
- Mutation fraction = 0.1
- Number of Generations = 20000

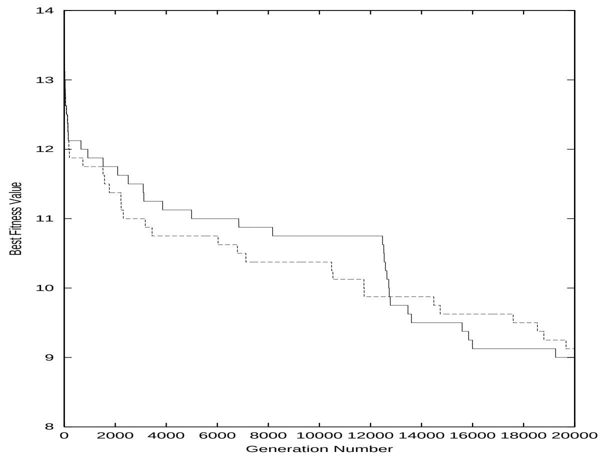

Figure 6: Comparison of controlled search and normal run for the N-Parity problem. The dashed lines denote the performance of controlled search. Learning period is 200.

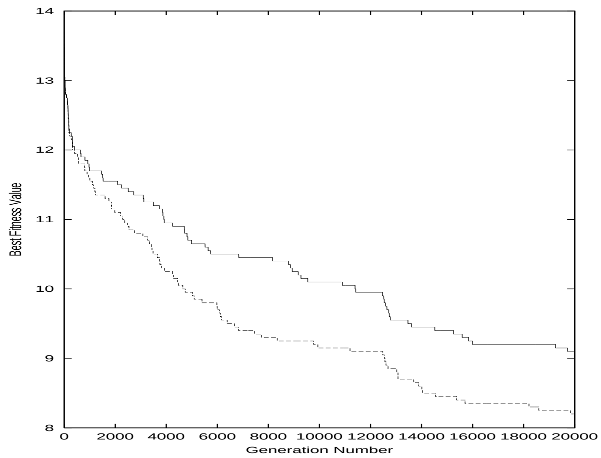

Figure 7: Comparison of controlled search and normal run for the N-Parity problem. The dashed lines denote the performance of controlled search. Learning period is 500.

The first test case has been carried out again using a learning period of 30. Similarly eight different runs have been carried out with various random seeds both for the basic and the controlled GP. The results obtained were consistent with the CFG-induction problem. Again the controlled search exhibited a worse performance. Considering the total number of terminal and function set elements which is 9, obtaining a similar performance is not surprising. Therefore the second test case has been tried with a period of 200 generations. The results of this test case are presented in figure 6. Again the results are consistent with the results obtained for CFG-induction. The controlled search can compete with the straightforward application but still cannot outperform it. A test with a learning period of 500 generations has been carried out and the results are presented in figure 7. Again the average of twenty different runs is used for this learning period. This period is sufficient for 5-parity problem too and the performance increase is outstanding.

6 Conclusion and Future Work

Our initial question was, if it could be possible to extract information during the genetic evolution and use this information to control the recombination operations afterwards. Our focus has been on highly deceptive problems, therefore we have tried to extract information about the global structure of the chromosomes. The results obtained in two different domains provides strong evidence about the success of our approach.

The critical question about the method proposed is about the extra processing time required for the control module. The learning process takes place at certain time intervals (every 500 generations for the successful tests) and could be considered as a constant increase in the processing time. However the main overhead depends on the procedure of forming vectors for the alternative offsprings and determining the class they belong to. Obviously the algorithm presented to form the vectors is linear in terms of the total number of nodes on a GP-tree. The effect of this overhead on the total processing time is related to the fitness function used. For problems with non-linear fitness functions, this overhead could be negligible and the performance increase becomes more important. CFG-induction problem is such an example as the fitness function includes the procedure of parsing the training examples.

References

[1] Baluja,S. & Davies, S. (1997) *Using optimal dependency-trees for combinatorial optimization: Learning structure of the search space.* In Proceedings of the 14th International Conference on Machine Learning (pp. 30-38). Morgan Kaufmann.

[2] Hitoshi Iba and Hugo de Garis, *Extending Genetic Programming with Recombinative Guidance*, In P. Angeline and K. E. Kinnear, Jr., editors, Advances in Genetic Programming 2, chapter 4. MIT Press, Cambridge, MA, USA, 1996.

[3] de Jong, Edwin D., Watson, Richard A., Pollack, Jordan B. *Reducing Bloat and Promoting Diversity using Multi-Objective Methods.* In Proceedings of the Genetic and Evolutionary Computation Conference. July 7-11,2001. San Francisco, California.

[4] Keller, B. and Lutz, R. (1997).*Evolving stochastic context-free grammars from examples using a minimum description length principle.* The Workshop on Automata, Inductive Grammatical Inference and Language Acquisition. ICML-97.

[5] Emin Erkan Korkmaz, Göktürk Üçoluk. *Genetic Programming for Grammar Induction* In Proceedings of 2001 Genetic and Evolutionary Computation Conference Late Breaking Papers, July 9-11, 2001 San Francisco, California.

[6] Mühlenbein, H., & Paab, G. (1996). *From recombination of genes to the estimation of distributions I. Binary Parameters.* Parallel Problem Solving from Nature, PPSN IV,178-187.

[7] Pelikan, M., & Mühlenbein, H.(1999). *The bivariate marginal distribution algorithm.* "Advances in Soft Computing - Engineering Design and Manufacturing",London Springer-Verlag. (pp 521-535)

[8] Pelikan, M., D. E. Goldberg, and E. Cantu-Paz (2000). *Linkage problem, distribution estimation, and bayesian networks.* Evolutionary Computation–340.

[9] Simon Lucas, *Structuring Chromosomes for Context-Free Grammar Evolution*, Proceedings of first IEEE International Conference on Evolutionary Computation, pp 130-135, June, 1994.

[10] Smith, T.C. and Witten, I.H. (1995) *A genetic algorithm for the induction of natural language grammars*, Proc IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing, 17-24, Montreal, Canada.

[11] Zannoni,Elena; Reynolds, Robert G. (1997) *Learning to Control the Program Evolution Process with Cultural Algorithms* Evolutionary Computation, Summer97, Vol. 5 Issue 2, p181.