Evolutionary Tree Genetic Programming

Ján Antolík
Department of Software Engineering
Charles University
Malostranské nám. 25
118 00 Praha 1, Czech Republic
jan.antolik@matfyz.cz

William H. Hsu
Department of Computing and Information
Kansas State University
234 Nichols Hall
Manhattan, Kansas 66506-2302
bhsu@cis.ksu.edu

ABSTRACT

We introduce a clustering-based method of subpopulation management in genetic programming (GP) called Evolutionary Tree Genetic Programming (ETGP). The biological motivation behind this work is the observation that the natural evolution follows a tree-like phylogenetic pattern. Our goal is to simulate similar behavior in artificial evolutionary systems such as GP. To test our model we use three common GP benchmarks: the Ant Algorithm, 11-Multiplexer, and Parity problems. The performance of the ETGP system is empirically compared to those of the GP system. Code size and variance are consistently reduced by a small but statistically significant percentage, resulting in a slight speedup in the Ant and 11-Multiplexer problems, while the same comparisons on the Parity problem are inconclusive.

Categories and Subject Descriptors: I.2.3 [Computing Methodologies]: Artificial Intelligence *Problem Solving, Control Methods, and Search*

General Terms: Algorithms

Keywords: genetic programming

1. INTRODUCTION

In the field of evolutionary computation, the problem of managing subpopulations in order to improve the convergence efficiency of a genetic algorithm (GA) or genetic programming (GP) system has proven challenging. Work on this problem has led to some research on biologically plausible models of speciation. This paper describes an approach using a clustering method inspired by evolutionary biology to reorganize subpopulations in genetic programming, with the goal of producing more highly fit individuals within a set number of fitness evaluations.

2. MOTIVATION

An interesting observation of natural evolution is that it proceeds in a tree-like pattern. Scientists believe that at the beginning of the life on the planet Earth only single extremely simple life form existed. The members of this life form gradually evolved until a point when two distinct species have been segregated. During the millennia of terrestrial evolution the same process was repeated in all the once-existing species, leading to creation of a structure that is known as the *Tree of Life*. This work represents an attempt to simulate this process in very simplified manner. It is motivated by the hypothesis that modeling an phylogenetically well-structured evolutionary process can increase efficiency of speciation and adaptation in a GP system. Several other existing EA systems were already motivated by some biological foundations of species formation [2] such as adaptive landscape and shifting balance theory or a more recent theory of punctuated equilibria.

GECCO'05, June 25–29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

3. EVOLUTIONARY TREE GENETIC PROGRAMMING

3.1 Basic algorithm

Our system consists of a population P that is separated into number of subpopulations S_i. The ETGP algorithm starts with a single large subpopulation. After each cf generations, where cf is parameter that we will refer to as the *clustering frequency*, we divide each subpopulation into a new set of subpopulations. The sizes of the subpopulations are constrained to be proportional to the average fitness of the individuals they contain.

The division of subpopulation S proceeds in two steps. At first we determine whether the given subpopulation S is large enough to undertake division. If the result of this decision is positive, we cluster the subpopulation S according to some metric. We then insert each of these clusters into the new population as new subpopulation instead of the original subpopulation S. Reader can find a detailed description of this process in [1].

As the clustering method for the ETGP algorithm, we have decided to use the Hierarchical Agglomerative Clustering (HAC). We use the minimum variance as the agglomerative clustering criterion. There are some specific modifications that we have introduced into the original HAC algorithm. In summary, we present the final modified clustering algorithm:

`find clusters A and B such`

`that the following property holds:`

$$|\mathtt{A}| \geq |\mathtt{B}| \wedge \forall_{\mathtt{x} \in \mathtt{S}} : (|\mathtt{x}| \leq |\mathtt{B}| \vee \mathtt{x} \equiv \mathtt{A})$$

`if` $(|A| + |B| \geq \alpha \sum_{x \in S} |x|) \wedge (\frac{|A|}{|B|} > \beta)$

```
then
    let U := A and V := B
    For each cluster C ∈ S except cluster A and B do
        if dist(A,C) ≤ dist(B,C) then U := U ∪ C else
        V := V ∪ C
    return (U,V)
else continue with the next level of clustering
```

where S refers to the current set of clusters, parameter α was set to 0.7, and the parameter β was set to 0.4 in all the experiments.

4. RESULTS

First we would like to very briefly discuss the methodology we have used in our experiments. We have conducted a large number of different tests. In order to make the situation more comprehensible, we have divided them into three series. Because of the scope of this paper only the first two will be discussed. All the tests were repeated 100 times and the averages over these 100 runs are reported. Everything was implemented as an extension of *Evolutionary Computation in Java (ECJ)* package by Luke [3]. As benchmark problems the **11-multiplexer**, **Artificial Ant (Santa Fe Trail)** and the **Odd 11-Parity** problems were selected.

4.1 First experiment series

The goal of the first series of experiments was to find the optimal values of the branch factor and clustering frequency of the ETGP method. We have conducted two experiments. In both of them, one of these parameters was changed while the other was fixed to a predetermined value. Therefore, the interrelationship of these two parameters was not explored. Also we have used only the **11-multiplexer** and **Artificial Ant** in these tests. Following table summarizes the best values of these parameters for both benchmark problems selected in these experiments:

	Clustering frequency	Branch factor
Artificial Ant	3, 4	0.2, 0.25
11-multiplexer	7, 10	0.2, 0.3, 0.4
Selected value	7	0.2

Table 1: Best performance values

With respect to this table we have decided that a reasonable compromise will be to use a value of 0.2 as the Branch factor and 7 as a value of the Clustering Frequency in all subsequent experiments.

4.2 Second experiment series

The second series of experiments finally provides us with a comparison of the ETGP algorithm with the basic version of the GP system. In these tests we have also switched on the two additional features of the ETGP algorithm that are not present in standard GP systems - the mutation and depth controll. We did not mention the second modification of ETGP algorithm so far. The reason is, that the motivation for introducing this parameter follows from constructive argument that did not fit into our motivation section because the scope of this paper. Anyways, let us briefly describe what the depth control modification does. There are two parameters related to this extension. The first one *Starting depth* S defines what is the maximal allowed depth of any GP tree in the population in the beginning of the evolution. The second one called *End depth* E defines the maximal depth of GP trees in the population in the end of evolution. The actual maximal allowed depth m in generation g can be computed from these two parameters in following way $m = S + (E - S)\frac{g}{numGen}$ where $numGen$ stands for the maximal number of generation in the given run.

In our experiments we have explored also the relationship between these two parameters. The *mutation ratio* was explored in the range of [0.05 - 0.4] with step 0.05. For the depth growth control the *Start depth* parameter is varied in the range [3 - 17] with step of 2. Because of the scope of this paper we summarize the results of these experiments in Table 2. Let us note that the table reports only the best results with respect to the two varied parameters.

	ETGP		GP	
	Best hits		Best hits	Speedup
	MR	SD		
Artificial Ant	74.5		71.01	+4.91%
	0.35	7		
11-multiplexer	1996.06		1960.5	+1.81%
	0.25	7		
11-parity	559.11		562.18	-0.5%
	0.05	7		

Table 2: This table summarizes the performance of the ETGP and standard GP algorithm over the three benchmark problems. The results for the ETGP algorithm are those achieved with the best combination of *Start Depth (SD)* and *Mutation Ratio (MR)* parameters.

5. CONCLUSIONS

The focus of this work was to explore the possibility to simulate evolution of species as we can observe it in nature. We have extended the basic GP algorithm by incorporating a dynamic formation of species. The goal was to build a system in which the species evolution form a evolutionary tree just as in natural evolution. As the mechanism of separation of new species we have employed a clustering technique that divides the given subpopulation into two new one, according to the genetic similarity of the individuals. The comparison with the standard GP algorithm was performed. As the previous section indicates, the speedup in the convergence curve, while statistically significant and robust, is minor. A third experiment series that provided us with some insights into the dynamics of species formation in ETGP system is omitted for brevity. The overall conclusion following from these experiments is that while the system does successfully use clustering to manage subpopulations, it does not fully simulate the natural properties of 'tree-like' evolution.

6. REFERENCES

[1] J. Antolík. Evolutionary tree genetic programming. Master's thesis, Computing and Information Sciences department, Kansas State University, Apr. 2004.

[2] S. Baluja. A massively distributed parallel genetic algorithm, 1992.

[3] S. Luke. Evolutionary computation in java. http://www.cs.umd.edu/projects/plus/ec/ecj/, 2001.