EVOLUTION TRACKING IN GENETIC PROGRAMMING

Bogdan Burlacu[a], Michael Affenzeller[b], Michael Kommenda[c], Stephan M. Winkler[d], Gabriel Kronberger[e]

[a-e]University of Applied Sciences Upper Austria
Heuristic and Evolutionary Algorithms Laboratory
Softwarepark 11, 4232 Hagenberg, Austria

[a]bogdan.burlacu@fh-hagenberg.at, [b]michael.affenzeller@fh-hagenberg.at, [c]michael.kommenda@fh-hagenberg.at,
[d]stephan.winkler@fh-hagenberg.at, [e]gabriel.kronberger@fh-hagenberg.at

ABSTRACT
Much effort has been put into understanding the artificial evolutionary dynamics within genetic programming (GP). However, the details are yet unclear so far, as to which elements make GP so powerful. This paper presents an attempt to study the evolution of a population of computer programs using HeuristicLab. A newly developed methodology for recording heredity information, based on a general conceptual framework of evolution, is employed for the analysis of a symbolic regression benchmark problem. In our example, we find the complex interplay between selection and crossover to be the cause for size increase in the population, as the average amount of genetic information transmitted from parents to offspring remains constant and independent of run constraints (i.e., tree size and depth limits). Empirical results reveal many interesting details and confirm the validity and generality of our approach, as a tool for understanding the complex aspects of GP such as introns and bloat.

Keywords: genetic programming, tree fragments, evolutionary dynamics, schema theory, population diversity, bloat, introns

1. INTRODUCTION

A difficult task in genetic programming is to explain the evolutionary behavior of highly polymorphic, dynamic populations of computer programs.

Evolution within GP is characterized by complex genotype-phenotype relations that make it difficult for researchers to identify the influential factors of emergent behavior and the underlying mechanisms behind phenomena such as loss of diversity, over-fitting, bloat and introns.

Although these phenomena have been correlated by scientists with different algorithmic components (run constraints, genetic operators, function and terminal set), the main reason a causal relationship could not be derived from the work is two-fold.

On the one hand, on the theoretical level, problems arise from the inherent complexity of the Genotype-Phenotype map, which is a mathematical function to describe the relationships between genotype and phenotype. Phenotypic innovation is the result of genetic modification mediated by the GP-map (Stadler and Stephens 2003, Stadler 2006). The notion of phenotypic neighborhood induced by the GP-map may differ fundamentally from any notion of "nearness" among phenotypes based solely on the comparison of their morphological features (Fontana and Schuster 1998). Therefore, stronger metrics and measurements are required for describing fitness landscape topologies in the presence of many-to-one relationships, i.e., sets of genotypes folding into essentially the same (at least on the semantic level) phenotypic structure. Difficulties in fulfilling this particular requirement make it unclear how to determine the role of selection, crossover and mutation in genetic programming.

On the other hand, the presence of representational bias (how genotypes are represented) or procedural bias (determined by genetic operators and fitness function) in the algorithm, varying across different problem domains, and the delicate balance between performance (training accuracy) and robustness (test accuracy) makes it unclear which innovations or algorithmic improvements are related to the underlying dynamics and which ones exploit particularities of a specific class of problems. Understanding the relationship between bias and generalization ability of genetic programming is crucial in designing algorithms with good generalization capabilities (Kushchu 2002).

In this paper, we focus on the study of genetic algorithms under the framework of neo-Darwinian evolution. The suggested approach, based on the tracking of all genetic information that flows through the evolutionary graph, constitutes the first step in an attempt to analyze and explain the influence and interplay of genetic operators in producing solutions.

The paper is organized as follows: Section 2 provides a brief overview of the essential biological concepts of evolution at the base of this approach. In Section 3, the implementation of the HeuristicLab tracking plugin is detailed. Section 4 discusses some preliminary results concerning the distribution of tree and fragment lengths sampled by crossover, and Section 5 is devoted to conclusions.

2. CONCEPTS OF ARTIFICIAL EVOLUTION

Evolution is the result of selection acting upon the genetic variation within a population. In other words, it requires variation in phenotype, differential reproduction on the basis of phenotype, and heredity of the traits associated with differential reproduction. Some traits associated with better fitness survive and propagate further while others – and their corresponding genes – become extinct in the population.

In genetic programming, operators such as crossover and mutation act on genotypes, while fitness-based selection acts on phenotypes. In this context, we take genes to be minimal fragments consisting of symbols (primitives) and terminals.

The ability of the genetic operators to produce useful variation (i.e., new compositions of symbols, variables and constants) plays a crucial role in algorithm performance; this requirement, however, is not sufficient to guarantee algorithm convergence, as there are cases when a relevant gene becomes extinct before getting the chance to become useful. Moreover, the closure property of the GP function set makes it possible, in theory, for genetic operators to produce variation *ad infinitum*. In practice, this possibility is limited by size and depth constraints, but this does not prevent the occurrence of bloat (i.e., gradual increase in tree size), or the appearance of introns ("intragenic region" – segments of non-viable, inefficient or neutral code), as an effect of selection pressure.

3. TRACKING OF EVOLUTION DATA

3.1. Analysis of Inheritance Information

The tracking functionality was implemented in HeuristicLab, a framework for heuristic and evolutionary algorithms developed at the Heuristic and Evolutionary Algorithms Laboratory (HEAL) of the Upper Austria University of Applied Science (http://dev.heuristiclab.com). Inheritance information is recorded in the form of an evolutionary graph, in which nodes represent individuals and arcs represent heredity relationships between them. It is clear that the graph itself consists of the union set of all lineages in the population. This representation was chosen for its potential to provide additional insight into the process of evolution by investigating the changes in the topological and algebraic properties of the graph.

Figure 1 shows an example of an individual (marked with a rectangle), its genealogy and its tree structure, in which the genetic information it received from the parent is highlighted. The interface facilitates the investigation of lineages, heredity and how the genetic material is assembled from lower building blocks during evolution.

In addition to tracking fragments transmitted via genetic recombination, it is also possible for arbitrary tree fragments to be matched against the population of individuals. This powerful tool can be used to investigate the distribution of certain fragments or schemas within the population.

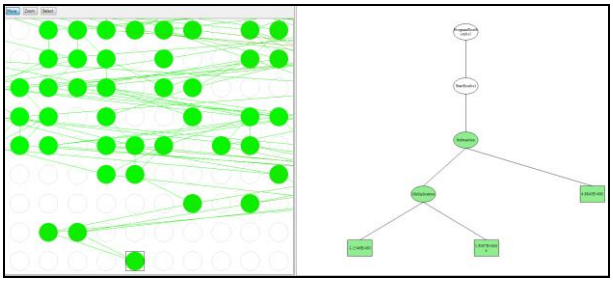

Figure 1: On the left, an individual (marked with a rectangle) and its genealogy. On the right, the tree structure of the individual. The highlighted nodes belong to the fragment that was received via crossover.

3.2. Tree and Fragment Similarity

Fragment matching is done according to three sets of rules. The terminology used below refers to *Symbols* which represent functions or operators, *Variables* which represent elements from the data set, and *Constants* which are random numbers supplied as inputs to the functions alongside the variables.

- S_1 „Exact": the entire fragment must be matched one-for-one: symbol names and arities must be the same, as well as variable names and weights, or constant values
- S_2 "High": exact matching of symbols, partial matching of leaf nodes – they are required to be of the same type (Variables or Constants)
- S_3 "Relaxed": exact matching for symbols, leaf nodes are considered wildcards

It is clear that for sets S_1, S_2, S_3 of matched fragments given by the three similarity rules defined above, $S_1 \subseteq S_2 \subseteq S_3$. Moreover, fragments contained in S_2 and S_3 are isologous (i.e., having a similar structure but containing different leafs). Isologous tree fragments can be considered elements of a set S of phenotypic instances of a gene G, in which case G can also be viewed as a schema. This provides a way to identify useful genes or schemas in the population.

Figure 2 shows an example of fragment matching. The tree fragment highlighted on the right was matched against multiple individuals in the population. The black nodes represent "exact" matches, the dark gray ones represent "high" matches, while the normal gray ones represent "relaxed" matches.

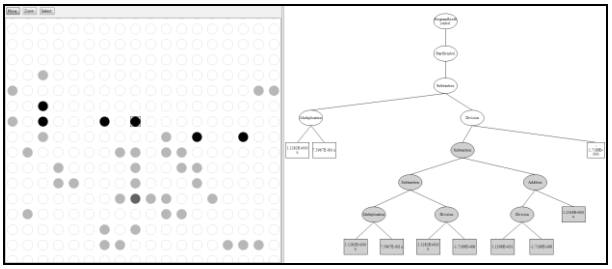

Figure 2: Example of fragment matching according to the three similarity criteria: black – "exact", dark gray – "high" and gray – "relaxed".

4. EMPIRICAL CASE STUDY

The study of tree fragments was done on a symbolic regression problem (*Poly-10*) where the target function is the 10-variable cubic polynomial:

$$f(x) = x_1 x_2 + x_3 x_4 + x_5 x_6 + x_1 x_7 x_9 + x_3 x_6 x_{10}$$

Table 1 describes the algorithm settings:

Population size	500
Generations	200
Selector	Tournament (size=3)
Crossover	Size-fair crossover (p=0.9)
Mutation	Multi-mutation operator (p=0.15)

Table 1: Genetic Algorithm Settings

In a first phase, our run analysis focused on the distribution of parent, children and fragment lengths across generations. For the Poly-10 problem, a maximum tree size of 100 nodes was used.

Empirical data shows that, with the size-fair crossover, which only performs a swap if the size and depth limits are respected, the children length is not very different from the length of the parent, and is, on average, smaller (Figure 3).

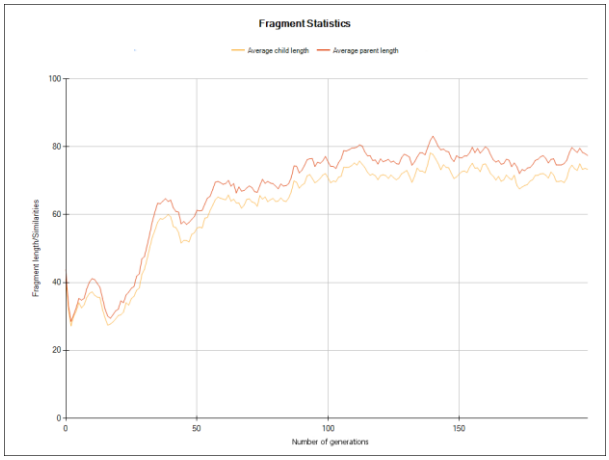

Figure 3: Average parent and child lengths for a 200-generations run

This result is somewhat surprising as it contrasts to the overall behavior of all the trees in the population (including those that did not reproduce) which is characterized by a gradual increase in length throughout evolution. The apparent contradiction can be explained by the interplay between the size-fair crossover, which on average produces smaller children, and selection, which in time tends to reduce diversity, thus promoting uniformity in the population at the expense of smaller and less fit programs.

To verify this, we propose a similarity measure based not on whole tree comparison, but on comparison between tree fragments that get transmitted via crossover. The similarity value is obtained by dividing the total number of identical fragments in the population (matched by the "exact" similarity measure) by the number of similarity groups they form.

The evolution of fragment similarity is shown in Figure 4, and is measured in average number of identical fragments per generation. The probability of a fragment occurring multiple times in the population is a product between the probability its containing individual gets selected multiple times and the probability that the fragment itself is sampled by crossover more than once. Therefore, it is reasonable to assume that even a small increase in the average number of identical fragments (i.e., similarity in the genetic material which gets passed to the offspring) can in fact mean a big decrease in population diversity.

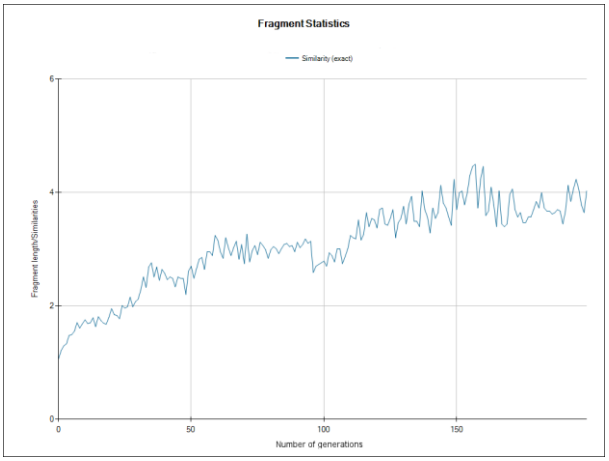

Figure 4: Increase in fragment similarity as the search converges

Another interesting result is that the average size of the crossover fragments tends to remain constant throughout the run, with a distribution clearly favoring smaller fragments (Figure 5). This distribution occurs repeatedly with very little variation across different runs with different settings. The only factor influencing average fragment size seems to be the average arity of the available functions. This means that the crossover operator is *size-invariant*, i.e., it contains symmetry; the average size of the swapped fragments tends to be the same, with a small bias towards producing children that are slightly smaller than their parents.

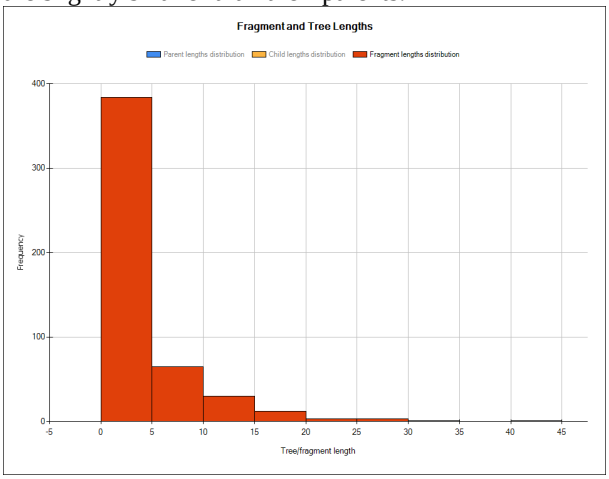

Figure 5: Sample length distribution of crossover fragments

5. CONCLUSIONS AND FUTURE PERSPECTIVES

In this paper, we described a powerful and general methodology for analyzing the evolutionary dynamics in genetic programming. The approach integrates the basic principles of evolution with a set of novel techniques for the tracking and analysis of inheritance and hereditary information.

Preliminary results are promising as they bring insights into the behavior of genetic operators and their combined effects. In our test case, we explain the increase in the average size of individuals as the result of the complex interplay between crossover and selection. We also explain bloat in terms of genetic variation and selection pressure, as the tendency towards functional neutrality of fragments following the increase in fitness to a stable value (this is linked, of course, to the properties of the function set and the genotype-phenotype map which is a many-to-one mapping).

Future research will provide more details regarding the extent to which each genetic operator influences the overall behavior of the algorithm. Detailed lineage analysis in terms of fragments and fitness similarities can reveal more about the characteristics of the underlying genotype-phenotype mappings. Concerning the evolutionary graph, we plan to see if the underlying topology can substantially affect the results of the evolutionary process.

Overall, the work described in this paper opens a breadth of new possibilities for the study and understanding of genetic programming and artificial evolution.

REFERENCES

Fontana, W. and Schuster, P., 1998. Continuity in evolution: On the nature of transitions. *Science,* 29 May 1998, Vol. 280, no. 5368, pp. 1451-1455.

Dignum, S. and Poli, R., 2008. Crossover, Sampling, Bloat and the Harmful Effects of Size Limits. *EuroGP'08 Proceedings of the 11th European conference on Genetic programming*.

Kushchu, I., 2002. Genetic Programming and Evolutionary Generalisation. *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 5, October 2002.

McPhee, N., Ohs, B. and Hutchison, T., 2008. Semantic building blocks in genetic programming. *EuroGP'08 Proceedings of the 11th European conference on Genetic programming*.

AUTHORS BIOGRAPHIES

BOGDAN BURLACU received his MsC in computer science and systems engineering in 2009 from the "Gheorghe Asachi" Technical University in Iasi, Romania. Currently he is a research associate in the Heuristic and Evolutionary Algorithms Laboratory in Hagenberg, under the supervision of Michael Affenzeller.

MICHAEL AFFENZELLER has published several papers, journal articles and books dealing with theoretical and practical aspects of evolutionary computation, genetic algorithms, and meta-heuristics in general. In 2001 he received his PhD in engineering sciences and in 2004 he received his habilitation in applied systems engineering, both from the Johannes Kepler University of Linz, Austria. Michael Affenzeller is professor at UAS, Campus Hagenberg, and head of the Josef Ressel Center *Heureka!* at Hagenberg.

MICHAEL KOMMENDA finished his studies in bioinformatics at Upper Austria University of Applied Sciences in 2007. Currently he is a research associate at the UAS Research Center Hagenberg working on data-based modeling algorithms for complex systems within *Heureka!*.

STEPHAN M. WINKLER received his PhD in engineering sciences in 2008 from Johannes Kepler University (JKU) Linz, Austria. His research interests include genetic programming, nonlinear model identification and machine learning. Since 2009, Dr. Winkler is professor at the Department for Medical and Bioinformatics at the University of Applied Sciences (UAS) Upper Austria at Hagenberg Campus; since 2010, Dr. Winkler is head of the Bioinformatics Research Group at UAS, Hagenberg.

GABRIEL KRONBERGER received his PhD in engineering sciences in 2010 from JKU Linz, Austria, and is a research associate at the UAS Research Center Hagenberg. His research interests include genetic programming, machine learning, and data mining and knowledge discovery.