# 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 algorithm behavior on 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.

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.

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. A brief summary of other research in this area is described in Section 3. In Section 4, the implementation of the HeuristicLab tracking plugin is detailed. Section 5 discusses some preliminary results concerning the distribution of tree and fragment lengths sampled by crossover, and Section 6 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. One of the key challenges in GP is to eliminate introns and bloat while at the same time maintaining just enough diversity in the population so that the search can succeed.

## 3. RELATED WORK

This section provides a summary of existing research on the topic of evolutionary dynamics, or, more generally, on how information from successive generations (genealogical information or otherwise significant indicators like size, fitness) can guide the search process or explain, predict or improve various aspects of artificial evolution.

Burke et al. (2003) improve population diversity by taking into account lineage information in the selection phase. A genetic lineage is defined as the connection from the root parent to those individuals which were created, via crossover, from that individual. Lineage selection is implemented as an additional step to bias selection towards different lineages from the initial population. In effect, the selected parents are the results of tournament selection across lineages, so that the selection pressure is reduced from the *fittest* individuals to the *fit and diverse* (Burke et al. 2003). In terms of evolutionary dynamics, the authors conclude that adding diversity can worsen fitness on some problems that clearly benefit from elitism in a hill-climbing environment, but may avoid local optima, when deception is embedded into the problem.

In a subsequent paper, Gustafson et al. (2005) focus on the analysis of survival rates, mating success and dissimilarity between offspring and parents. They reach the conclusion that similar parents (with the same fitness) are unlikely to produce better offspring and they introduce a simple rule to prevent mating between solutions with the same fitness. Their work supports the idea that GP search may be improved by producing more differently fit solutions.

Another work by Smart et al. (2007) suggests a methodology for investigating the building blocks hypothesis in GP, by aggregating statistical information of fragments and fragment schemas. A fragment is defined as a connected set of nodes from the program tree. A fragment schema is defined as the set of all programs containing a specified fragment at some point in the program tree. In their paper, Smart et al. use the concept of *maximal fragment*, defined as the largest fragment contained in all subtrees from some subset of the population. The set of maximal fragments can be analyzed in various ways to identify properties of the set of all fragments (Smart et al. 2007). The analysis relies on a subtree-mining algorithm called TRIPS (TRee mIning algorithm using Prüfer Sequences) capable of finding frequent subtrees (fragments) from a forest of tree structures. The TRIPS algorithm is explained in detail in Tatikonda et al. (2006).

Finally, in a series of papers, Poli et al. detail the correlation between the mean program size at generation 0, the average arity of the primitive set and the internal node distributions of mixed arity trees. They develop a size evolution equation which is an exact formalization of average program size dynamics. It is shown that under standard subtree-swapping crossover (uniform selection of crossover points), the population converges to a limiting tree length distribution that will exponentially sample smaller programs (Poli and McPhee 2008, Dignum and Poli 2008). For this reason, crossover depth and size limits are found to actually have a positive effect towards bloat, as small trees are more likely to be sampled, but are less likely to generate new programs. In Poli and McPhee (2008), an effective method for dynamically setting the parsimony coefficient, and thus controlling the average program length in the population, is derived from the size evolution equation.

The work of Poli et al. is especially important as it lays the foundations for a mathematical model of evolutionary search. It also emphasizes the important role of the interplay between selection and crossover, which determines GP behavior.

## 4. TRACKING OF EVOLUTION DATA

This section outlines our proposed methodology for the study of evolutionary dynamics within GP, with a focus on genetic operator behavior and fragment statistics. In the context of this paper, a fragment denotes a subtree (usually swapped by crossover) that is part of a bigger rooted tree (the individual).

### 4.1. 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.

### 4.2. Tree and Fragment Similarity

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.

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.

## 5. 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:

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

| | |
|---|---|
| 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 from a standard GP run 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).

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 crossover, which drives the size distribution of children towards smaller lengths, and selection, which tends to reduce diversity, thus promoting uniformity in the population at the expense of smaller, less fit programs.

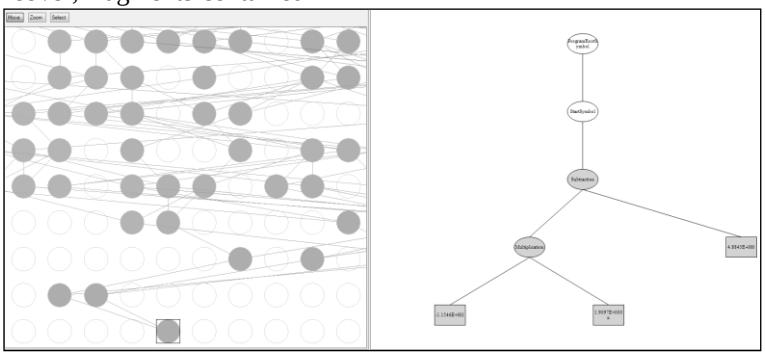

Figure 1: On the left, an individual (marked with a rectangle) and its genealogy. On the right, the tree structure of the individual. The highlighted nodes belong to the fragment that was received via crossover.

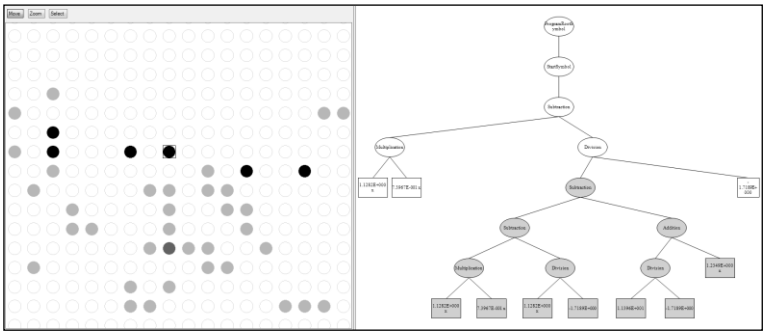

Figure 2: Example of fragment matching according to the three similarity criteria: black – "exact", dark gray – "high" and gray – "relaxed". The matched fragment is highlighted in the right-hand side.

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.

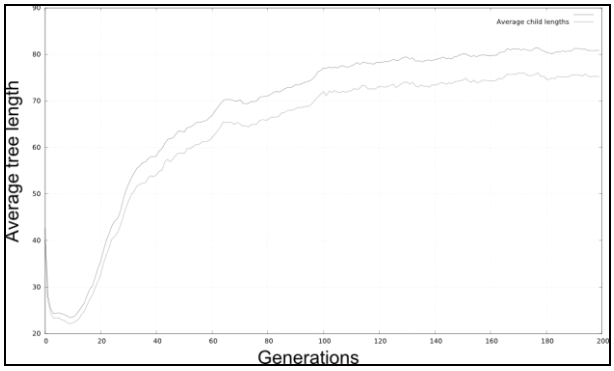


Figure 3: Average parent and child lengths over 200 generations (10-run average)

The evolution of fragment similarity is shown in Figure 4, and is measured in average number of similar 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.

Another interesting result is that the average size of the crossover fragments tends to remain constant throughout the run. The only factor influencing average fragment size seems to be the average arity of the available functions. Figure 4 shows an increase in average fragment length in the beginning of the run (first 20 generations), followed by a decrease and stabilization at a fragment size value of approximately 4.6 nodes. This can be explained by the fact that, after the initial exploratory phase of the algorithm, when many different points in the solution space are sampled, the search becomes gradually more local and the accepted changes become those that have a small positive effect and do not affect the overall structure of the tree.

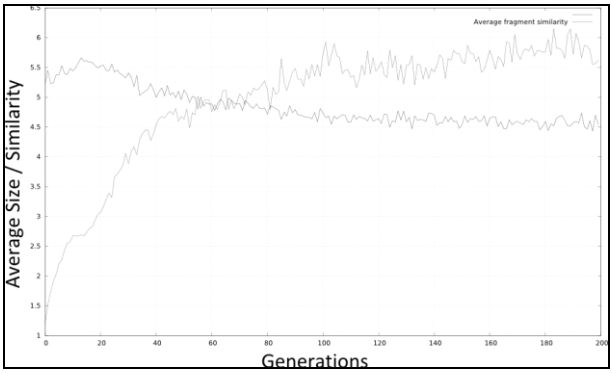


Figure 4: Fragment similarity increasing from 1 to 5.5, fragment length stabilizing at ~4.6 nodes

The correlation between fragment size and search locality becomes more obvious if we look at the Offspring Selection Genetic Algorithm (Affenzeller et al. 2009). We outline the main idea of the algorithm here: in OSGA the selection mechanism is extended to include an "offspring selection" step after the parents are selected via the usual means (i.e., proportional, roulette or tournament selection). In this extra step, the fitness value of the offspring is compared with the fitness values of its parents (either the worst or best parent, or an average between the two). Only the offspring that are able to outperform their parents (denoted as "successful" offspring) are accepted as candidates for the further evolutionary process, according to a predefined *success ratio* which gives the percentage of the population that is expected to surpass their parents' fitness. The rest of the population is filled with random individuals chosen from the pool of individuals that were also created by crossover but did not reach the success criterion. This strategy guarantees that evolution is resumed mainly with crossover results that were able to mix the properties of their parents in an advantageous way. The value of selection pressure is given by the quotient of new individuals that had to be created until the success ratio was reached, and the number of individuals in the population (Affenzeller et al. 2009).

Consequently, in OSGA, because of the dynamic selection pressure, the shift from global to local search is more pronounced and the correlation with average fragment size is easier to observe. Similar to successful/unsuccessful individuals, we define a successful fragment as a fragment contained in an individual that was selected for reproduction. The OSGA algorithm run settings are shown in Table 2.

| | |
|---|---|
| Population size | 500 |
| Generations | 1000 |
| Maximum tree depth | 10 |
| Maximum tree length | 100 |
| Selector | Proportional |
| Crossover | Size-fair crossover |
| Mutation rate | 20 % |
| Comparison factor | 1 |
| Success ratio | 1 |

Table 2: OSGA Algorithm Settings

Note that even though the algorithm is allowed to run for 1000 generations, the termination criteria will be reached before that, when the selection pressure exceeds a predefined threshold (when no more successful individuals can be produced).

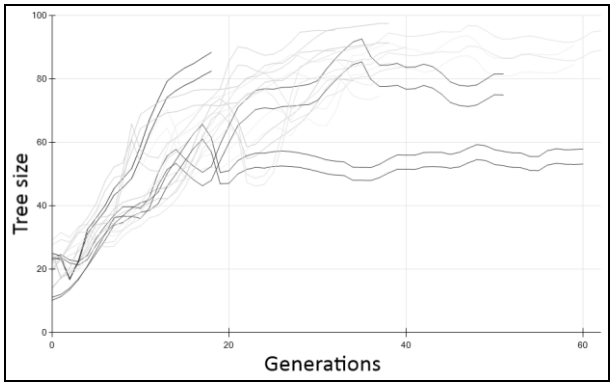


Figure 3: OSGA – average parent/child lengths, 10 runs

Figure 3 shows the evolution of parent/child average length for 10 OSGA runs. The average child length is always lower than the average parent length. Different runs have different lengths depending on when the stop criterion is reached

.

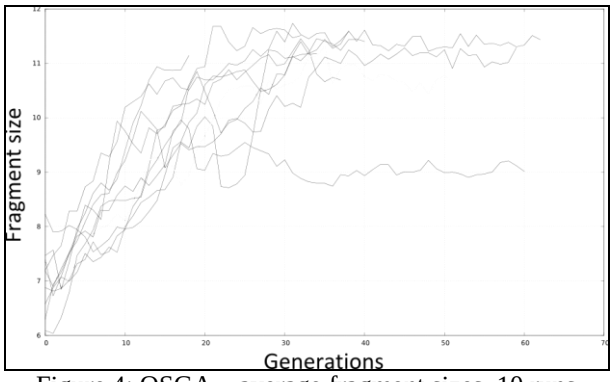


Figure 4: OSGA – average fragment sizes, 10 runs

In OSGA, the selection pressure increases as it becomes harder to produce successful offspring. This translates into an overall increase in fragment size as the algorithm spends more effort trying to find new and better programs. However, while the average fragment size increases (Figure 4), the average size of successful fragments gradually becomes smaller, as the search converges (Figure 5). This proves that a selection pressure steering mechanism is an effective guide for the evolutionary search. In contrast with the standard GA, the OSGA behavior illustrates more clearly what happens when the search converges. The accepted modifications gradually become smaller until no better children can be obtained from the available genetic material.

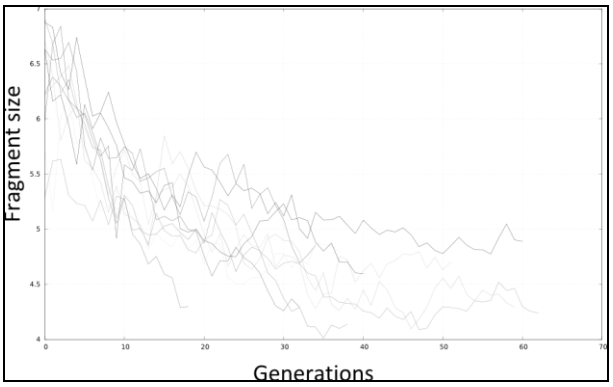


Figure 5: OSGA – average size of successful fragments, 10 runs

## 6. DISCUSSION AND CONCLUSIONS

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.

The fragment statistics acquired during the run provide an additional insight into the behavior and dynamics of the evolutionary run. The increase in fragment similarity shows how genetic diversity is lost as the search progresses. The results show that average fragment length stabilizes and remains constant, and that although the overall tendency of the GA population is to increase in size, the children produced by crossover are on average smaller than their parents. Using the notion of "successful" fragments, we provided an image and explanation of GP convergent behavior.

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


Affenzeller, M., Winkler, S., Wagner, S., Beham, A., 2009. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Chapman & Hall/CRC ©2009.

Burke, E., Gustafson, S., Kendall, G. and Krasnogor, N., 2003. Is Increased Diversity in Genetic Programming Beneficial? An Analysis of Lineage Selection. *Evolutionary Computation*, 8-12 Dec 2003, Vol. 2, pp. 1398-1405.

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*, pp 158-169. March 26-28, 2008. Naples, Italy.

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.

Gustafson, S., Burke, E.K. and Krasnogor, N.., 2005. On Improving Genetic Programming for Symbolic Regression. *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, CEC 2005, 2-5 Sep 2005, Vol. 1, pp. 912-919.

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*, pp 135-145. March 26-28, 2008. Naples, Italy.

Poli, R., McPhee, N.F., 2008. Covariant Parsimony Pressure for Genetic Programming. Technical Report, CES-479, Department of Computing and Electronic Systems, University of Essex, UK, 2008.

Smart, W., Andreae, P. and Zhang, M., 2007. Empirical Analysis of GP Tree-Fragments. *EuroGP'07 Proceedings of the 10th European conference on Genetic programming*, pp. 55-67. April 11-13, 2007, Valencia, Spain.

Tatikonda, S., Parthasarathy, S. and Kurc, T., 2006. TRIPS and TIDES: new algorithms for tree mining. *CIKM '06 Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 55-67. 2006. New York, NY, USA.


## 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 professor at the UAS Research Center Hagenberg. His research interests include genetic programming, machine learning, and data mining and knowledge discovery.