Self-Improvement for the ADATE Automatic Programming System

Roland Olsson
Computer Science Dept.
Østfold College
1750 HALDEN, Norway
Roland.Olsson@hiof.no
+47 69215369
http://www-ia.hiof.no/~rolando

Brock Wilcox
Northern Arizona University
NAU Box 8087
Flagstaff, AZ 86011, USA
rbw3@cet.nau.edu
928 - 523 - 4903

Automatic Design of Algorithms Through Evolution (ADATE) [2] is a system for automatic programming based on the neutral theory of evolution [1] . This theory states that the majority of molecular changes in evolution are due to neutral or almost neutral mutations. A consequence is that most of the variability and polymorphism within a species comes from mutation-driven drift of alleles that are selectively neutral or nearly neutral. In ADATE, as well as in natural evolution, neutral walks in genotype space are essential for avoiding combinatorial explosions due to complex mutations.

The compound transformations in ADATE were designed according to this model of transition from one species to the next. In ADATE, the first part of a compound program transformation is the neutral walk and the typically small second part is the brutal mutation.

We have taken aim at using ADATE to improve itself by evolving better transition (neutral transforms and mutations) operators. There are many different paths to self-improvement of these transition operators. We explore two such methods which we've called SIG-reshaping and SIG-rewriting.

SIG-rewriting is the automatic synthesis of semantics-preserving rewriting rules that are employed for neutral random walks. Such rules increase the number of connections in a neutral network and thereby also the number of genotypes reachable through a neutral walk without fitness computation.

Since ADATE is optimized to exploit neutrality, it is more difficult to improve ADATE using SIG-rewriting. An advantage of automatically synthesized rewrite rules, however, is that they can be optimized to the specific problem in a way that could not be anticipated when we designed ADATE. Employing several synthesized rewriting rules in ADATE showed no significant performance increase.

The primary goal of SIG-reshaping is to alter the distribution of synthesized (mutated) expressions so that they cover as many equivalence classes as possible. For example synthesizing and using both `E` and `not(not(E))` or both `or(`E_1`,`E_2`)` and `or(`E_2`,`E_1`)` for arbitrary boolean expressions `E`, E_1 and E_2 may be a waste of time. SIG-reshaping synthesizes an acceptance predicate that determines if a given synthesized expression is used.

The fitness function used in our experiments requires that a synthesized acceptance predicate accepts at least one minimum size expression in each equivalence class and rejects as many other expressions as possible. Given this fitness function ADATE generated an acceptance predicate that rejects 999 out of the 1055 expressions of size five or less consisting of `not`, `and`, `or`, `false`, `true` and three variables `X1`, `X2`, `X3` – while still accepting at least one minimum size member of each class.

SIG-reshaping has demonstrated some self-improvement. After generating an acceptance predicate for boolean expressions, limiting the use of equivalent expressions, the predicate was tested on four, five, and six bit xor problems. Comparing the results with and without the predicate indicate that self-improvement has occurred.

Our web site contains an ADATE specification file for SIG-reshaping as well as the source code of ADATE itself, which should make it easy to reproduce our results.

References

[1] J.L. King and T. H. Jukes (1969). Non-Darwinian evolution. *Science* **164** : 788–798.

[2] J. R. Olsson (1995). Inductive functional programming using incremental program transformation. *Artificial Intelligence.* Vol. 74, No. 1, 55–83.