Keeping the Diversity with Small Populations Using Logic-Based Genetic Programming

Ken Taniguchi and Takao Terano

Graduate School of Business Sciences, University of Tsukuba,
3-29-1, Otsuka, Bunkyo-ku, Tokyo 112-0012, Japan
{taniguti, terano}@gssm.otsuka.tsukuba.ac.jp
http://www.gssm.otsuka.tsukuba.ac.jp/staff/terano/

Abstract. We present a new method of Logic-Based Genetic Programming (LBGP). Using the intrinsic mechanism of backtracking in Prolog, we utilize large individual programs with redundant clauses, and apply them to small populations. Our method is validated by the experiments about Symbolic Regression, and XOR problems.

Genetic programming is a method to automate programming tasks in which programs are evolved through Genetic Algorithm (GA) techniques. One of the difficulties of GP [2] is that I) they must have a large number of individuals in order to keep the population diversity, and that II) they cannot reuse sub-structures of program codes destroyed by improper uses of genetic operators. To cope with the issues, we will focus on the Prolog-based logic programming framework.

In this paper, we propose a new Logic-Based Genetic Programming (LBGP) and show the effectiveness via computer experiments. In our research, we follow the ideas: (1) Each program as an individual consists of a set of clauses, which are redundant like the representation of individuals in Messy GAs [1] and we allow them to execute only some parts of clauses, and (2) Instead of using large populations, flexible genetic operations are designed to exchange the roles of clause sets, which are directly executable or not in the run.

In our LBGP, we directly handle tree structures of program codes. We use Prolog language to represent an individual as a set of clauses. The order of clauses corresponds to the locus of chromosomes. Multiple clauses with same heads are executed in the ascending order similar.

We define the two concepts: exon and intron parts, which are similar to the ones in the literature [4]. The exon part is a clause directly executed and evaluated by means of a given fitness function. Exon parts are usually the most top clauses among the same heads. The introns part is the other clauses, however, by the backtracking mechanism, they will become candidates for the execution.

The exonizing operation stochastically activates the intron or inactive parts generated by sub-tree crossover or intronizing operations. By the operation, a selected Horn clause is replaced to the most upper line from the other places. The restructured sub-

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 724–725, 2004.

trees are expected to be effective in the later phases of genetic cycles, because they may contain indirectly executable and potentially usable components.

The procedure of our LBGP is summarized as follows:

Step 1: Initialize
Generate the initial population with random trees
Step 2: Selection of Reproduction
Select two parents randomly from the populations.
Step 3: Genetic Operation
Stochastically apply the one of the following genetic operations:
3-1) Sub-tree Crossover;
3-2) One Point Crossover;
3-3) Mutation Operation
3-4) Activate/Inactivate Operation:
3-4-1) If there are Intron Parts, Apply Exonizing Operation;
3-4-2) Ohterwise, Apply Intronizing Operation to the Elitist Individual.
Step 4: Selection of Survival
Apply Generation Model for the survival.
Step 5: Loop
If the terminal conditions are satisfied, then stop. Else go to Step 1.

We have carried out experiments about Symbolic Regression and XOR problems to validate the performance by means of program execution. Also, we have compared our implementation with the conventional LBGP code from CMU AI Repository [3]. The results have suggested the proposed method outperform ten times by means of the number of program evaluation.

The performance of the method are attained by (1) variable length chromosome representation of codes in Prolog programs, (2) employments of exonizing and intronizing operations, (3) integration of several generation models, and (4) introduction of the stack counter. Our future work includes the further integration of LBGP with multi-modal and multi-objective problems, the validation of applicability to large scale practical problems.

References

1. Goldberg, D. E., Korb, B., and Deb, K.: Messy Genetic Algorithms: Motivation, Analysis, and First Results, *Complex Systems*, Vol. 3, pp. 493-530 (1989)
2. Koza, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press (1992)
3. Raik, S.: Genetic Programming in Prolog, CMU-AI-Reposiroty (1993)
4. Terao, M. and Iba, H.: Controlling Effective Introns for Multi-Agent Learning by Genetic Programming, *Proc. of GECCO 2000*, pp. 419-426 (2000)