Perspective into Model-based Genetic Programming

†P. He[1, 2] and A. C. Hu[1]

[1]School of Computer Science and Educational Software, Guangzhou University, P. R. China
[2]School of Computer and Communication Engineering, Changsha University of Science and Technology, P. R. China

*Presenting author:bk_he@126.com
†Corresponding author: bk_he@126.com

Abstract

Representation is an open issue in GP (Genetic Programming) research area, having close relationships with its performance improvements. This paper introduces a novel GP framework called model-based grammatical evolution (MGE) as well as the principle it obeys. In MGE, individuals take the form of sequences of productions, therefore providing means for structural analysis and semantic reuses. To certify the effectiveness of MGE, comparisons with some other GP variants like classical grammatical evolution (CGE), integer representation GE are also conducted.

Keywords: Genetic Programming, Grammatical Evolution, Model, Finite Sate Automaton.

Introduction

Genetic Programming (GP) [1] as one of the most important automatically programming approaches constructs programs by means of evolution principle. It generates populations of chromosomes in terms of genetic algorithm (GA) [2], chooses at last the fittest individual from the final population for the desired solution. So, GP could be recognized as a GA variant, but it is much simpler than GA in delineating complex structures, therefore having been applied in a wide range of fields like mathematical modeling, circuit design, pattern recognition, and financial prediction, etc. [1][3]-[8].

Up to now, GP grows up into a big family comprising of a large number of variants such as classical GP, gene expression programming (GEP), multi-expression programming (MEP), grammatical evolution (GE), and so on [4]-[8]. However, while using them extensively, we should take notice of the following deficiencies.

- Many GPs like tree based GP and grammar based GP are difficult to use for the sake of their complex representations.
- Most of existing GPs are devised from the principle of software testing, providing few means dealing with semantics.
- Some GP variants like GEP are easy to use, but their expressiveness is very limited. For instance, GEP, as far as the expressiveness is concerned, can essentially be described by GE.

In view of these, we will provide a novel GP framework, which was called model based GE, for coping with the abovementioned problems. It borrows some ideas of model checking. We will introduce the principle abided by and a sample model-based GE in the following parts. Finally, to demonstrate the effectiveness of the present approach, comparisons with some other GP variants like IGE (Integer representation GE) [7], PIGE and CGE (Classical GE) [4][5][8] are conducted.

Modeling Principle

Model approach has long been regarded as a powerful solution to system representation, system analysis, and software development. GP as program generation tool can naturally benefit from using of model approach. By model approach, we mean [9]:

1. Delineating both the problem and property of concern, say M and φ, in the context of some description model;
2. Establishing that $M \models \varphi$ holds. When M is a transition system, our goal is to prove $M, S \models \varphi$ holds for some special state S in M.

Consequently, GP can be modeled as follows based on the above model checking strategy.

1. Constructing a finite state transition system M for the concerned GP;
2. Delineating what we are interested;
3. Designing an algorithm suitable to check the satisfaction of $M, S \models \varphi$.

Model-based GP

So far, we have obtained two model-based GP variants called HGP (Hoare Logic-based Genetic Programming) [10][11] and MGE (Model-based Grammatical Evolution) [12]-[14] in terms of the principle of part II. The unified method is summarized as the following steps. If having further interest, one can refer to [12] [13] for the details.

1. Constructing a transition diagram G= <V, E> with some vertex $v_0 \in V$ as the start symbol by steps 2 through 5. Here V and E are sets of vertices and edges, respectively.
2. Regarding the states of V either as sets of logic formulas or as sets of sentential forms;
3. Regarding e in E either as programs or as productions of some context-free grammar. In this case, both states and edges could be used to define Hoare triples or grammatical deviations.
4. Defining relations among states to be connected.
5. The formal framework obtained from steps 2 through 4 is suitable for either verifying and generating the desired programs or deriving programs grammatically;
6. Constructing genetic operations over the formal framework of step 5, we obtain either HGP or MGE [10]-[13].

For instance, the transition matrix given in table 1 is the model of languages of the grammar in Figure 1. This model covers all the leftmost derivations of the concerned grammar. According to the matrix, we can solve certain regression problems (see the following part) as shown in Figure 2.

(1) <expr>::= <expr><op><expr>	(11)	(3) <pre_op>::= sin	(31)
| (<expr><op><expr>)	(12)	|cos	(32)
| <pre_op> (<expr>)	(13)	| exp	(33)
| <var>	(14)	| log	(34)
(2) <op>::= +	(21)	(4) <var>::= y	(41)
|-	(22)	|1.0	(42)
| *	(23)		
| /	(24)		

Figure 1 grammar of expression

Table 1 Transition Matrix

S	1	2	3	4	5	6	7	8	9	10
ε	1	1,2	3,4	4	5,6	6	1,7	8,9	9	1,10
11	2									
12	2									
13	8									
14	3									
21						7				
22						7				
23						7				
24						7				
31									10	
32									10	
33									10	
34									10	
41				5						
42				5						

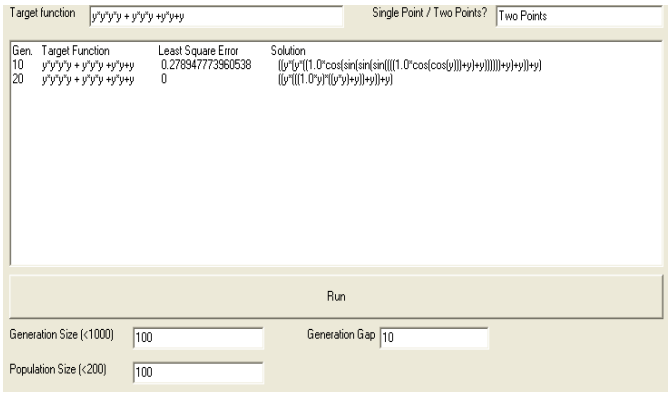

Figure 2 Screenshot of the `method` with population size=100, generation size=100.

Experiments

In this part, we will demonstrate the performance improvement of the present approach through comparisons of it with CGE[5], IGE [7] and PIGE in solving regression problems. The grammar used here is given in figure 1, and the objective is to find Eq. 1 based on 20 sample input values {-1, -0.9, -0.8, -0.76, -0.72, -0.68, -0.64, -0.4, -0.2, 0, 0.2, 0.4, 0.63, 0.72, 0.81, 0.90, 0.93, 0.96, 0.99, 1} in the range [-1…1].

$$f(y) = y^4 + y^3 + y^2 + y \qquad \text{(Eq. 1)}$$

The method is as follows: constructing the grammar model as shown in table 1; analyzing the structure of the model, and constructing building-block based GE; running the obtained GE over sample input values will result in figures 3 to 4 [12]. It follows from these figures that the present approach has advantages over the other GE variants in efficiency.

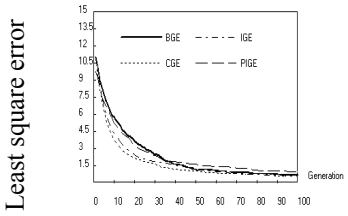

Figure 3 Average fitness of 100 runs of the four GEs in Eq. 1

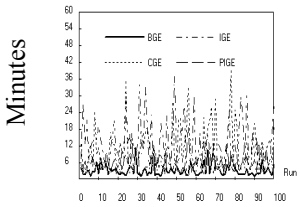

Figure 4 Time used of 100 individual runs of the four GEs in Eq. 1

Conclusions

This paper introduces the principle MGE abides by, and application method in solving real-world problems. Experiment demonstrates that MGE has advantage over classical GE, integer representation GE, and PIGE in performance improvement. Our future work will focus deeply on its semantic computing, and unifications with other GP variants.

Acknowledgements

The research work was supported in part by National Natural Science Foundation of China (Grant No. 61170199), the Scientific Research Fund of Education Department of Hunan Province, China (Grant No.11A004), and the Natural Science Foundation of Guangdong Province, China (Grant No. 2015A030313501)

References

[1] Koza J.R (1992). Genetic Programming. MIT Press, Cambridge M. A.
[2] Holland J.(1975) Adaptation in Natural and Artificial Systems, The University of Michigan Press: Michigan.
[3] Langdon W.B, Harman M. (2015). Optimizing Existing Software with Genetic Programming, IEEE Transactions on Evolutionary Computation, 19(1): 118-135
[4] Oltean M, Grosan C, Diosan L, Mihaila C (2009). Genetic Programming with Linear Representation: A Survey, International Journal on Artificial Intelligence Tools, 19(2): 197-239
[5] O'Neill M., Ryan C. (2001) Grammatical Evolution. IEEE Transactions on Evolutionary Computation., 5(4): 349-358.
[6] Ferreira C. (2001) Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. Complex Systems, 13(2): 87-129
[7] Hugosson J, Hemberg E, Brabazon A, O'Neill M (2010). Genotype Representation in Grammatical Evolution. Applied Soft Computing. 10: 36-43
[8] Alfonseca M., Gil F.J.S. (2013). Evolving an ecology of Mathematical expressions with Grammatical Evolution, BioSystems, 111: 111-119
[9] Michael Huth, Mark Ryan.(2004) Logic in Computer Science: Modelling and Reasoning about System. Cambridge University Press: England.
[10] He P. Kang L.S., Fu M. (2008). Formality Based Genetic Programming, IEEE CEC.
[11] He P, Kang L.S, Johnson C. G. and Ying S (2011), Hoare logic-based genetic programming, Science China Information Sciences, 52, 623-637
[12] He P., Johnson C.G. Wang H.F (2011), Modeling Grammatical Evolution by Automaton, Science China Information Sciences, 52, 2544-2553
[13] He P. Deng Z.L. Wang H.F, Liu Z.S (2015), Model Approach to grammatical evolution: theory and case study, Soft Comput, 2015. DOI 10.1007/s00500-015-1710-9
[14] He P. Deng Z. L., Gao C.Z., Wang X. N., Li J. (2016). Model Approach to grammatical evolution: Deep-Structured Analyzing of Model and Representation, Soft Comput, 2016. DOI 10.1007/s00500-016-2130-1