ISSN: 1803-3814

FUNCTION SET STRUCTURE INFLUENCE ONTO GPA EFFICIENCY

Tomas Brandejsky

University of Pardubice
Department of Software Technologies, Faculty of Electrical Engineering and Informatics
CS Legies Square 565
Czech Republic
Tomas.Brandejsky@upce.cz

Abstract: *The paper discusses the influence of function set structure onto efficiency of GPA (Genetic Programming Algorithms), and hierarchical algorithms like GPA-ES (GPA with Evolutionary Strategy to separate parameter optimization) algorithm efficiency. On the foreword, the discussed GPA algorithm is described. Then there is depicted function set and common requirements to its structure. On the end of this contribution, the test examples and environment as well as results of measurement of influence of superfluous functions presence in the used function set is discussed.*

Keywords: *Genetic Programming Algorithm, Efficiency, Function set, Function set structure, Inapplicable functions, mutually replaceable functions.*

1 Introduction

Investigating the influences impacts onto GPA efficiency, many practical experiments points to function set as significant but non well explored one. Function set means set of building blocks used by GPA to build solutions in terminology used in [1]. It is interesting that many other influences like initial population structures or various crossover and mutation operations are investigated in much deeper details. The paper starts with description of GP algorithm used in experiments presented in this study. Then there are described compared function sets and test example. The paper ends with discussion of measured data. Obtained results conclude that minimal function set satisfying sufficiency condition gives the best results. Many other approaches of studying information dynamics of GPAs are discussed in work [2].

2 Used GP algorithm

To presented test, standard structure GP algorithm was used. It is tree-based one to simplify the structure of evolutionary operation implementation. Structure of the GP algorithm is outlined on the Alg. 1. Initialization of the population is controlled by two parameters – minimal and maximal depth of the structures. After evaluation of individuals, there the sorting of population on the base of fitness function is applied. Sorting of the population causes some mixing of individuals to prevent repeated evolutionary operations of the same parents. Then the fulfilment of ending condition is tested. If it is not satisfied, new separate population is produced by application of evolutionary operations. Used GPA utilize the following evolutionary operations:

- mutation
- one-point-crossower
- regenerating (generating of totally new individual without respect to structure of existing ones).

Binary evolutionary operations like crossover are applied to immediate adjacent individuals in sorted population vector. Then each individual of the new population is evaluated to determine its fitness. Offspring replaces its parent only if it is better than the best of them. In the opposite case it is throw away.

Mutation operation is applied only to change function of given arity to another of the equal number of attributes. Mutation does not allow e.g. to replace binary addition by unary sin function because such changes are subject of crossover. In practice, mutation works as node replacement mutation described in [1], page 43. From this viewpoint, the used GPA is similar to grammar based approaches, but it is not Grammatical Evolution, because the tree structure of genes is implemented. If such constrained mutation is applied, it forms simple rewriting grammar on the base of given function set been used in GP in a variety of ways [3, 4,5, 6,7].

Probability of both crossover operators is 0.5. The main advance of one-point-crossover is that it tends to producing of semantically comparable structures. On the opposite side, it does not allow e.g. to replicate useful structure discovering in the right part of the tree also on the left part. Thus two different crossover operators are applied.

The book [1] introduces as optimal proportion of crossover and mutation operation 9:1, but it admits that proportion 1:1 might work too. Observation of presented GPA rather concludes idea of [8] to prefer mutation in the first GPA cycles and later 1:1. The number of pure mutation cycles is in used GPA equal to maximal initial depth of trees representing individuals.

Bowing of arising structures might be controlled in the case of GPA-ES version [9] by constrained number of parameter optimizing cycles of nested ES (Evolutionary Strategy) algorithm. The principle of this bowing is essential – when there is overcomplicated structure, it has bigger number of parameters and it is not probable that these parameters will be determined well in the constrained number of ES evolutionary cycles and thus this overcomplicated structure is indirectly penalised.

Algorithm 1: Studied GPA algorithm.

```
1) FOR ALL individuals DO Initialize() END FOR;
2) FOR ALL individuals DO Evaluate()=>fitness END FOR;
3) Sort(individuals);
4) IF Terminal_condition() THEN STOP END IF;
5) FOR ALL individuals DO
      SELECT Rand() OF
        CASE a DO Mutate()=> new_individuals;
        CASE b DO Symetric_crossover() => new_individuals;
        CASE c DO One_point_crossover() => new_individuals;
        CASE d DO Re-gerating() => new_individuals;
      END SELECT;
   END FOR;
6) FOR ALL individuals DO Evaluate => new_fitness END FOR;
7) FOR ALL individuals DO
     IF new_fitness<fitness THEN
        individual = new_individual;
        fitness = new_fitness;
     END IF;
8) GOTO 3);
```

Measuring efficiency of GP algorithm there is problem of Operating System Scheduler, especially in multi-thread implementations. Due to task switching, computation time is influenced by other task present in computer during execution non-looking to the fact that solution structure and needed number of iterations is constant. It is better to use measurement based on number of iteration cycles or on the number of fitness function evaluation respectively.

3 Function set

Used function set strongly influence efficiency of GPA. There are known common recommendations as to use “rich” set to prevent the need to interpolate missing function e.g. by spontaneously discovered Taylor series which tends to need on many GPA iteration cycles and thus to loose of efficiency. In the work [1] there it is used term sufficiency to label fact that function set contains all needed functions. Problem is that in the case of unknown and complicated function symbolic regression it is extremely difficult or even impossible to determine which function set is sufficient. There are also possible other sources of inefficiency like mutually replaceable functions as e.g. twin sin() and cos(). The presence of such twins should decrease efficiency of GPA.

In presented study used terminal set was formed by two terminals – constant and variable. Typically, there are more possible variables than only one, e.g. x, y, z. In GPA there are two ways of implementation. One is to reason variable as unary function with argument representing number of variable. This construct opens way to build such structures as arrays of variables if argument of unary variable function is not constant. In used GP algorithm the another approach is used. Variable is terminal with one parameter. This parameter representing variable ordinal number might be set during initialization and changed only by mutation operator.

In this study, three function sets were applied. The basic one consists of the following functions:

$$+,-,* \tag{1}$$

$$\sin(), +, -, * \tag{2}$$

$$\text{abs}(), +, -, *, \text{mod}() \tag{3}$$

$$\text{abs}(), \sin(), \cos(), +, -, *, \text{mod}() \tag{4}$$

Solving symbolic regression of differential equations from data representing Lorenz attractor system (5) with parameters (6), which serves as test bed in this study, some functions are superfluous. Function set (1) is minimal set covering really used functions only. In the basic function set (2) it is the only unary function sin(). In the set (3), there was added second superfluous binary function mod() and superfluous unary function sin() was replaced by abs(), which is also superfluous, but it is better applicable in some equations representing Lorenz attractor system. Function set (4) adds two superfluous, mutually replaceable unary functions sin() and cos () to function set (2). These functions can replace themselves mutually because of there it is need to change their argument offset only.

Lorenz attractor system was used for comparability with other experiments. There exists standard GP benchmarks, some of them are oriented to symbolic regression problem. As it was published in [10, 11], if we have sufficient number

of experiments and if we solve symbolical regression problems with small residual error, average results depends only on statistical properties of initial population generator and applied evolutionary operators and it is possible to estimate them by application of Markov chain.

$$\begin{aligned} x'(t) &= \sigma\,(y(t) - x(t)) \\ y'(t) &= x(t)(\rho - z(t)) - y(t) \\ z'(t) &= x(t)\,y(t) - \beta\,z(t) \end{aligned} \tag{5}$$

$$\begin{aligned} \sigma &= 16 \\ \beta &= 4 \\ \rho &= 45.91 \end{aligned} \tag{6}$$

4 Experiment results

The results of numerical experiment were sumarised on Fig. 1. and Tab. 1. Experiments were consisting of symbolic regression of differential equations describing data set with sum of error squares from all data samples less than 10^{-7}. There was computed 9000 of experiments for each function set. Big number of experiments is caused by multiple sources of errors. Especially, pRNG controling GPA function and generating initial populations is not thread safe. Novel standard of C++ version 11 asks for thread safe implementation of <random> library which is used in GPA-ES algorithm, but many Linux distributions do not implement. Even thread safe implementation is not presented in Intel C/C++ Compiler used on supercomputers. Thus, task switching significantly influences obtained results. To utilize multicore processors it is need to use multi-thread OpenMP implementation. Such simplementation is sensitive to task switching influencing pRNG. Also popuation sizes has influence to GPA property. Such number of hardly controllable parameters tends to extreme numbers of experiments to eliminate them and their number was limited especially by available computing sources. GPA population size was chosen as 50 and ES population as 30 individuals. This sizes are rather small and was used to increase sensitivity (repsented by number of needed iterations to reach required error). Extremely small sum of error squares eliminates occurrence of incomparable individuals approximating training data set by alternative function.

Data set described movement of Lorenz attractor system in 500 points simulated with time step 0.01[s].

The presented results concludes that number of iterations increases with number of superflous functions in average. Comparing to previously published results [9] it is possible to state that influence of operator set to GP algorithm efficiency is at least 10 times bigger than influence of applied pseudo random number generator.

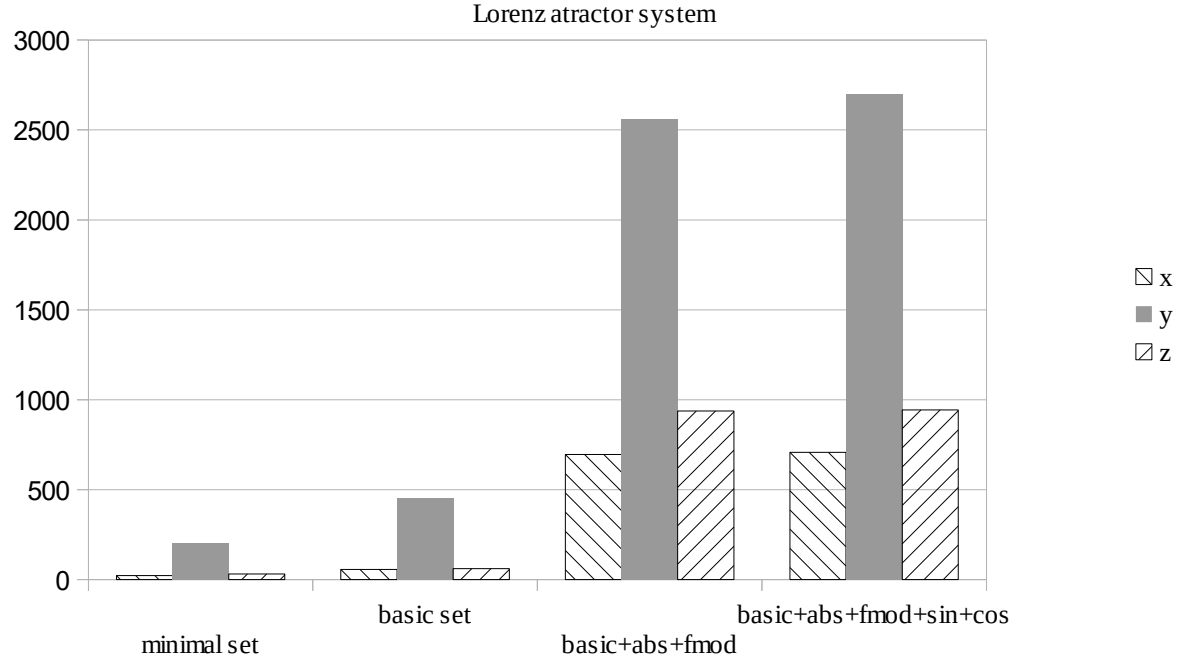

Figure 1: Average number iterations from 9000 of symbolic regression experiments

5 Conclusion

Structure of operator set significantly influences efficiency of Genetic Programming Algorithm. The most significant is satisfaction of the sufficiency condition (it means presence of all needed operators to prevent function approximation e.g. by spontaneously discovered Taylor approximation) presented in [1] but not only. The next is the non-presence of superfluous functions. Each function in function set which will not contribute to final solution decreases efficiency of crossover and mutation operations. The problematic are superfluous functions which can be replaced one by other and vice versa. Typical example of such function twins are sin() and cos() functions, which differs only by offset. Presence of such twins to possibility to replace them mutually without any significant improvement of regressed function quality, but the worst are non-linear superfluous functions without any relation to searched function. Such kind of functions was represented by abs and fmod (float-point-modulo) functions which makes search space strongly non-linear and harder to investigate.

Table 1: Deviations of obtained results of Lorenz attractor system symbolic regression

	Variables		
Operator set	*x*	*y*	*z*
Minimal set	18.70	157.65	21.85
Basic set	76.38	406.29	44.25
Basic+abs+fmod	726.02	1675.68	712.67
Basic+abs+fmod+sin+cos	613.90	1766.94	751.89

Acknowledgement: This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project „IT4Innovations National Supercomputing Center – LM2015070“.

References

[1] Poli, R., Langdon, W. B., McPhee N. F.: *A field guide to genetic programming,* Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk (2008). (With contributions by J. R. Koza).

[2] Langdon, W.B., Poli, R. *Foundations of Genetic Programming*. Springer-Verlag Berlin Heidelberg. ISBN 978-3-540-42451-2. DOI 10.1007/978-3-662-04726-2

[3] Gruau F.: On using syntactic constraints with genetic programming. In P. J. Angeline and K. E. Kinnear, Jr., editors, Advances in Genetic Programming 2, chapter 19, pp. 377–394 (1996). MIT Press, Cambridge, MA, USA.

[4] Hoai, N. X., McKay R. I., Abbass H. A.: Tree adjoining grammars, language bias, and genetic programming. In C. Ryan, et al., editors, Genetic Programming, Proceedings of EuroGP'2003, volume 2610 of LNCS, pages 335–344, Essex, pp. 14-16 April 2003. Springer-Verlag. ISBN 3-540-00971-X. http://www.cs.adfa.edu.au/~abbass/publications/hardcopies/TAG3P-EuroGp-03.pdf

[5] O'Neill, M., Ryan C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, volume 4 of Genetic programming. Kluwer Academic Publishers, 2003. ISBN 1-4020-7444-1. http://www.wkap.nl/prod/b/1-4020-7444-1.

[6] Whigham P. A. : Search bias, language bias, and genetic programming. In J. R. Koza, et al., editors, Genetic Programming 1996: Proceedings of the First Annual Conference, pages 230–237, Stanford University, CA, USA, pp. 28–31 July 1996. MIT Press (1996)

[7] Wong, M. L., Leung K. S.: Evolving recursive functions for the even-parity problem using genetic programming. In P. J. Angeline and K. E. Kinnear, Jr., (eds) Adxxxxxvances in Genetic Programming 2, chapter 11, pp. 221–240. MIT Press, Cambridge, MA, USA, 1996. ISBN 0-262-01158-1.

[8] Koza J.R., Bennett III F. H., Andre D. and Keane M. A.: (1999), Genetic Programming III – Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, San Francisco

[8] Brandejsky, T.: Multi-layered evolutionary system suitable to symbolic model regression. In: Recent Researches in Applied Informatics. Athens: WSEAS Press, 2011, vol. 1, pp. 222-225. ISBN 978-1-61804-034-

[9] Brandejsky T.: Problems of analyse of PRNGs influence onto the GPA-ES algorithm behaviours. In *Proceedings of 22nd International Conference on Soft Computing* – MENDEL 2016, pp. 57–60. BUT Press, Brno (2016)

[10] Brandejsky, T., Zelinka, I.: Specific Bahaviour of GPA-ES Evolutionary System Observed in Deterministic Chaos Regression. In: Zelinka, I., et al., eds. *Nostradamus: Modern Methods of Prediction, Modeling and Analysis of Nonlinear Systems.* Nostradamus. Ostrava, 05.09.2012 - 07.09.2012. Heidelberg: Springer. 2013, pp. 73-81. Advances in Intelligent Systems and Computing. ISSN 2194-5357. ISBN 978-3-642-33226-5.

[11] Soustek, P., Matousek, R., Dvorak, J., Bednar, J.: Canadian traveller problem: A solution using ant colony optimization. In *19th International Conference of Soft Computing, MENDEL 2013. Mendel journal series,* 2013. Brno, Czech Republic, 2013. pp. 439–444. ISSN: 1803-3814.