Function Approximation by means of Multi-Branches Genetic Programming

Katya Rodríguez-Vázquez and Carlos Oliver-Morales

IIMAS-UNAM, Circuito Escolar, Ciudad Universitaria, 04510 Coyoacán
Mexico City, Mexico

{katya@uxdea4.iimas.unam.mx, oliver_carlos_99@yahoo.com}

Abstract. This work presents a performance analysis of a Multi-Branches Genetic Programming (MBGP) approach applied in symbolic regression (e.g. function approximation) problems. Genetic Programming (GP) has been previously applied to this kind of regression. However, one of the main drawbacks of GP is the fact that individuals tend to grow in size through the evolution process without a significant improvement in individual performance. In Multi-Branches Genetic Programming (MBGP), an individual is composed of several branches, each branch can evolve a part of individual solution, and final solution is composed of the integration of these partial solutions. Accurate solutions emerge by using MBGP consisting of a less complex structure in comparison with solutions generated by means of traditional GP encoding without considering any additional mechanisms such as a multi-objective fitness functions evaluation for tree size controlling.

1 Introduction

In function approximation problems based on GP approaches, two relevant aspects are considered. On the one hand, evolved functions must be accurate approximations; on the other hand, solutions complexity must be also kept as simple as possible without incurring in deterioration in accuracy.
Accuracy for function approximation is general defined as an error between real and estimated values. The must common way of calculating this error is by means of the normalised mean squared error (NRMS). In the case of complexity, this factor has been measured in a different ways. The most usual metric counts the number of functions (or nodes). However, Streeter and Becker (2003) have assigned a cost to each type of function. They have assigned a cost of 0.1 for functions such as + and -, a cost of 1 for * and /, and a cost equal to 10 for more expensive operations such as sqrt, log, exp and trigonometric operations. Then, complexity and accuracy measures must be combined in a cost function in order to produce accurate but simple solutions.
The most common way of simultaneously evaluating these objectives is to measure the performance of each individual and penalise solutions with a large number of functions (or nodes) as shown by Koza (1992). However, this approach tends to produce short solution with a poor performance. Approaches which use multi-objective fitness function have also proposed. The first work reported that used a

multi-objective fitness function evaluation based on a Pareto front concept with GP was by Langdon (1995). He did not apply it for tree size controlling. Rodríguez *et al.* (1997) and Rodríguez and Fleming (1999) proposed the use of a multiobjective fitness function based on Pareto dominance for controlling complexity. More recently, de Jong and Pollack (2003) and Streeter and Becker (2003) have also used a multiobjective fitness evaluation for tree size controlling.
Then, this paper presented an alternative GP encoding based on multiple branches. Previous work have shown to work well in simple Boolean domain problems and also in symbolic regression [Rodríguez-Vázquez and Oliver-Morales; 2003], showing the evolution of accurate and simple solution without incurring in a multi-objective fitness function formulation or any mechanism for tree size controlling. However, behaviour of branches has not been studied. Then, this paper presents an analysis of branches behaviour and a detailed study of the effect of MBGP in function approximation problems. Fitness function is defined based only on minimisation of error metric (e.g. NMRS), analysing structure of evolved MB solutions.
Then, paper is structured as follows. Section 2 gives a description of MBGP encoding. In section 3, description of function approximation problems is provided. Section 4 presents experimental results and section 5 is dedicated for discussion. Finally, section 6 draws conclusions and future work.

2 Representation of Executable Structure

An important aspect of evolutionary algorithms performance is population representation. Angeline (1996) mentioned that representation in Evolutionary Algorithms plays an important role in order to get a successful search.
The work by Cramer in 1985 introduced the representation of computer programs. Cramer represented individuals by means of constant size strings. However, representations of fixed size reduce the flexibility and applicability of these implementations. In 1992, Koza introduced the Genetic Programming paradigm. This uses representations of variable size, hierarchical tree structures. In Koza's proposal these are S-expression (Koza described GP based on LISP programming language). This GP tree representation requires a set of primitives. Selecting the adequate primitives determines the efficiency of using this encoding.
New representation proposals followed Koza's initial work. One of these was the use of modular structures (sub-routines). In this area, Angeline and Pollack (1994) introduced the GLIB. A similar version of GLIB approach is the ADF's (Automatically Defined Functions). The aim of ADF's is to protect branches with an important genetic value from the destruction of the genetic operators: mutation and recombination (Koza, 1994). Rosca and Ballard (1996) have also proposed an Adaptative Representation which defines heuristics for detecting useful branches.
In some cases, alternative representations have been proposed for solving specific problems. In symbolic regression, an alternative is to use a representation by means of different types of polynomials. An example of this sort of representations is the GMDH (Group Method of Data Handling) proposed by Ivakhnenko (1971), which uses a network of transfer polynomials for pruning a layer and the outputs of a previous layer are the inputs of subsequent ones. Nikolaev and Iba (2001) introduced

the Accelerated Genetic Programming of Polynomials which also considers the use of transfer polynomials combined with a recursive least squares algorithm. Others problems have been also solved by using genetic programming and polynomials as the case of modelling chemical process (1996). Using polynomials has the advantage of estimating associated coefficients by means of diverse forms. In Koza's proposal (1992), coefficients and constants values are obtaining by means of evolution process. Other GP versions compute coefficients by means of Least Squares algorithms (Nikolaev and Iba, 2001; Rodríguez *et al.*, 1997).
The multi-branches representation presented in this paper also states the use of polynomials. This approach searches a solution by dividing the problem into sub-problems (branches) and solving each of these sub-problems individually. Once sub-problems are solved, partial solutions of branches are integrated in order to obtain a global solution of the problem.

2.1 Multi-Branches GP Representation

The multi-branches (MB) representation used in this work consists of four parts: a root node, *N* branches, *N+1* coefficients and an output. The number of coefficients is *N+1*, a coefficient for each branch plus the constant term. Following expression provides details of multi-branches representation.

```
(ADD
       V1
       (divd (.* V4 (.* V4 V4)) V5)
       (divd (divd V1 (divd V1 V4)) V5)
       (.* V1 V4)
       (divd (divd (.* V2 V2) (.* V5 V2)) V5)
       (divd V3 (divd V5 V1))
       (divd V3 V5)
       (.* (+ V1 (divd V4 V5)) V4)
       (divd V3 (divd V5 (- V3 V1)))
       3.3067
       3.5036
       0.87467
       0.82177
       0.0060857
       0.70499
       -0.2266
       -2.6548
       0.59913
       -1.6498
)
```

In this example of MBGP encoding, rooted-node has been defined as the addition operation (ADD). First branch only consists of variable V1, while second branch is defined as $V4^3/V5$. Coefficients for each branch are also expressed. Last coefficient corresponds to constant term into polynomial expressin.
The two main genetic operators are crossover and mutation. Crossing over consists of selecting a pair of parent structures. Then, a branch is randomly selected in each parent; finally, selected branches are exchanged between them. Mutation operator

randomly selects an individual. A branch is then selected, deleted and substituted by a new branch randomly generated.

3 Problem Statement

Fitting data points (curves) can be performed by means of a combination of known functions. The function approximation *g(x)* is obtained by estimating the set of coefficients a_i, one for each of the *n* known functions *f(x)* and a_0 (the independent coefficient), as given by equation (1).

$$\begin{aligned} g(x) &= a_0 + a_1 f_1(x) + \cdots + a_n f_n(x) \\ &= a_0 + \sum_{i=1}^{n} a_i f_i(x) \end{aligned} \quad (1)$$

Determining the set of more adequate functions a priori is not an easy task. The set of known functions can be larger and the value of some coefficients can be so small that the associated function is insignificant and estimating coefficients for all possible functions is computationally expensive. A better approximation method should consider only the most significant functions and computational costs must be also low. Thus, the main point is to find such functions *p(x)* and associated coefficients which are the most significant in order to better approximate a curve.
Using genetic programming, a given finite set of primitives (functions and terminals or argument) bounds the problem of function approximation. The set of primitives can then build a wide range of functions. The only restriction presented in GP is the maximum size of the hierarchical tree structure.

4 Experiments

A set of problems was defined in order to have a wide range of function approximation cases, which are described as follows.

Case a: In this case, the harmonic number defined in equation (2) is used in order to approximate the well-known function expansion given in equation (3), an asymptotic expansion where γ is the Euler's constant ($\gamma \approx 0.57722$).

$$H_n \equiv \sum_{i=1}^{n} \frac{1}{i} \quad (2)$$

$$H_n = \gamma + \ln(n) + \frac{1}{2n} - \frac{1}{12\,n^2} + \frac{1}{120\,n^4} + O(\frac{1}{n^6}) \quad (3)$$

where n = 1, 2, …, 50 (the first 50 harmonic numbers defined as fitness cases). The function set used in this problem is F = {+, -, *, %, -x, plog, psqrt, cos}, where %, plog and psqrt are the protected division, the protected natural logarithm and the protected square root, respectively. Then, the aim of this experiment is to rediscover terms of an already known approximation.

Case b: A set of three problems consisting of one independent variable is tested in this case. Function approximations are performed over three different ranges as shown in Table 1. A maximum branch depth was set to 6, except for case b.1 where this parameter was 4. The function set consists of the four basic arithmetic operations F = {+, -, *, %}.

Table 1. Range for each approximation problem.

function	*Range 1*		*Range 2*		*Range 3*	
ln(x)	[1,2]	(b.1)	[1,11]	(b.2)	[1,101]	(b.3)
sqrt(x)	[0,1]	(b.4)	[0,10]	(b.5)	[0,100]	(b.6)
arcsinh(x)	[0,1]	(b.7)	[0,10]	(b.8)	[0,100]	(b.9)

Case c: In this case, a function of two (equation (4)) independent variables was used in order to analysis the ability of MBGP for approximating functions consisting of more than two variables. Here, the function set is defined as case b.

$$z = x^y \tag{4}$$

Results of this experiment will compare with Streeter and Becker's work (2003) which is based on traditional GP representation (Koza's style) but using a multi-objective fitness function. Parameter setting for these three experiments is shown in Table 2.

Table 2. Parameter settting for cases a), b) and c)

Population Size	50
Maximum Number of Generation	100
% Selection	95
% Crossover	100
% Mutation	5
Maximum Number of Branches	10
Maximum Branch Depth	6

5 Result Analysis and Discussion

Results of case b) in terms of accuracy (error metric), number of generations and number of nodes (complexity) are summarised in Table 3. The minimum, average and maximum values of 25 runs are presented.

In following Figures (1 to 11), the Padé approximations and GP Pareto front (see Streeter and Becker, 2003) as well as the solution with smallest error generated by means of MBGP are presented. Figure 1 shows experiment (a) results. Figures 2 to 10 present solutions for experiment (b); finally, Figure 11 displays solutions of problem (c). It is important to mention that cost was calculated based on Streeter and Becker's criterion: assigning a cost of 0.1 for + and -, 1.0 for % and *, and 10 for other functions.

Table 3. Performance of experiment b)

	Error			**Generations minimun**			**Nodes**			
case	**min**	**mean**	**max**	**min**	**mean**	**Max**	**min**	**mean**	**max**	**cost**
a.1	3.19E-5	2.56E-4	7.09E-4	19	43.28	70	15	17.88	22	18.5
b.1	2.24E-8	1.38E-7	4.91E-7	14	58.48	97	42	64.72	104	6.7
b.2	1.59E-5	0.00055	0.00265	43	86.2	100	48	126.9	190	15.1
b.3	4.15E-8	1.92E-6	1.68E-5	21	81.04	100	44	106.2	204	10.2
b.4	7.60E-7	3.51E-6	1.77E-5	49	83.04	100	86	136.6	200	14.4
b.5	1.01E-5	6.05E-4	2.60E-3	39	85.92	100	58	133.7	246	13.7
b.6	1.95E-5	4.90E-4	2.53E-3	54	87.52	100	78	130.3	250	8.0
b.7	1.08E-4	7.19E-4	2.97E-3	36	82.12	100	80	135.0	190	17.9
b.8	5.32E-4	6.37E-3	5.02E-2	23	83.96	100	38	114.1	196	15.9
b.9	2.09E-4	2.34E-3	6.39E-3	53	90.32	100	90	124.6	188	15.7
c.1	3.50E-4	1.75E-3	6.18E-3	28	85.8	100	34	68.64	98	13.2

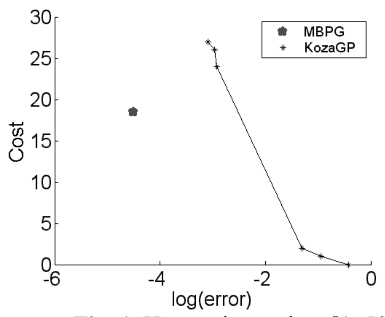


Fig. 1. Harmonic number, [1, 50].

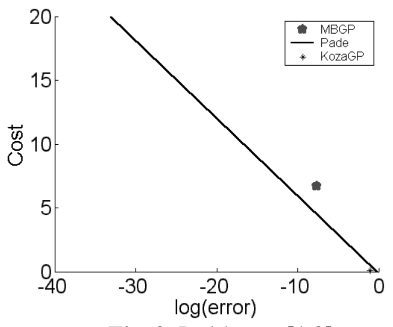


Fig. 2. Ln(x), x ∈ [1,2].

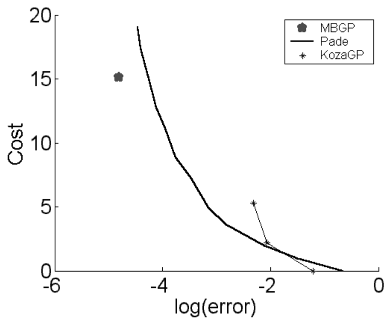


Fig. 3. sqrt (x), x ∈ [0,1].

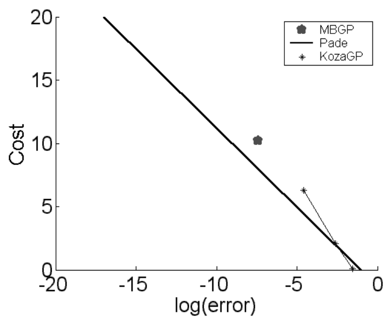


Fig. 4. arcsinh(x), x ∈ [0,1].

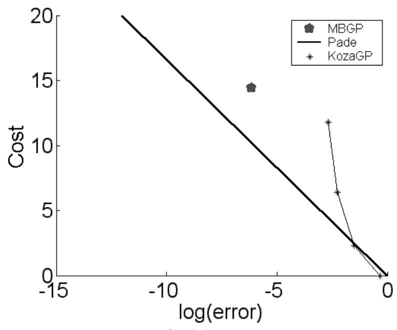


Fig. 5. ln(x), x ∈ [1,11].

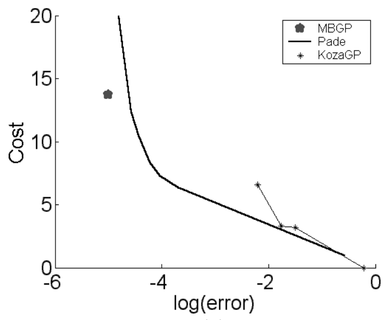


Fig. 6. sqrt (x), x ∈ [0,10].

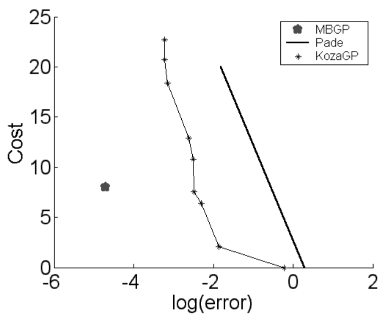


Fig. 7. arcsinh(x), x ∈ [0,10].

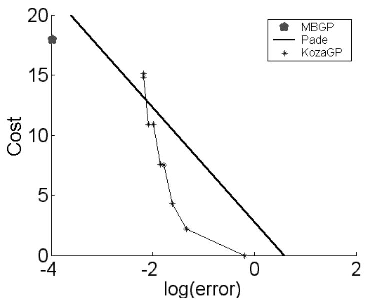


Fig. 8. ln(x), x ∈ [1,101].

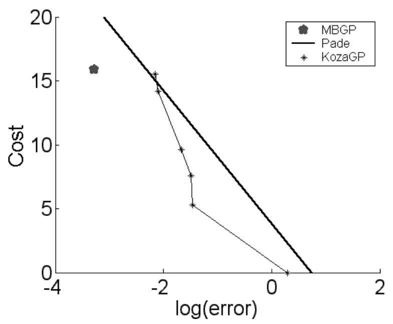


Fig. 9. sqrt(x), x ∈ [0,100].

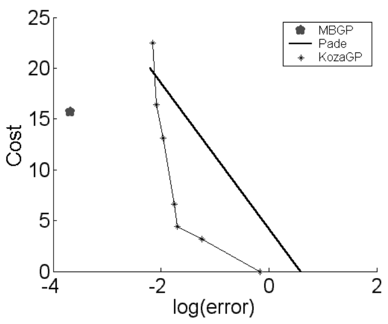


Fig. 10. arcsinh(x), x ∈ [0,100].

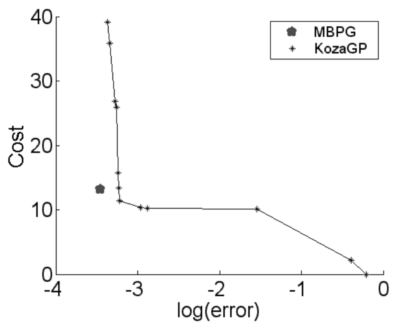


Fig. 11. x^y, x, y ∈ [1,2].

5.1 Discussion

First experiment had the aim of finding terms and coefficients already known based on function expansion used for approximating the harmonic number (equation (3)). In this case, the following approximation emerged,

```
    (ADD
                  (divd (divd V1 V1) (.* V1 V1))
                  (log_p (+ V1 V1))
                  (divd (ngt V1) (.* V1 V1))
                  -0.072444
                  2.302
                  -0.49414
                  -0.1147)
       )
```

Translating this MBGP model and substituting *V1* by *n*, the function expansion is expressed as,

$$H_n = -0.072444\frac{1}{n^2} + 2.302\,\mathbf{ln}(2n) - 0.49414\frac{-n}{n^2} - 0.1147$$

ordering this equation,

$$H_n = [\mathbf{ln}(2) - 0.1147] + \mathbf{ln}(n) + \frac{1}{2.0237n} - \frac{1}{13.8n^2} + 1.302\,\mathbf{ln}(2n)$$
$$= 0.5784 + \mathbf{ln}(n) + \frac{1}{2.0237n} - \frac{1}{13.8n^2} + 1.302\,\mathbf{ln}(2n)$$

It is observed that MBGP approximated the first four terms given in equation (3), showing an approximation error of 3.18742x10^{-5}.
Experiments (b) and (c) had the aim of finding a good approximation but also to analyses the computational cost. As can be seen form Figures 1-11, MBGP was superior to traditional GP in terms of accuracy. These figures show two curves corresponding to the Pareto front produced by GP using a multi-objective fitness function and Padé approximations (Streeter and Becker, 2003). It is also shown an isolated dot corresponding to the solution generated by means of MBGP. Because of MBGP does not use a multiobjective fitness function, only one solution emerges for each run. From these graphs, it is observed that MBGP solutions dominate some solutions from the Pareto-GP approach. Thus, the Pareto front is modified by including MBGP individual solution. Solutions that are not dominated by MBGP, are better in terms of complexity (cost). Comparing Padé, Pareto-GP (see Streeter and Becker, 2003) and MBGP, it was observed that both GP approaches showed a better performance than Padé for larger ranges (see cases b.7, b.8 and b.9 from Table 3). In contrast, GP showed a worse performance when the range was small (see cases b.1, b.2 and b.3 from Table 3).

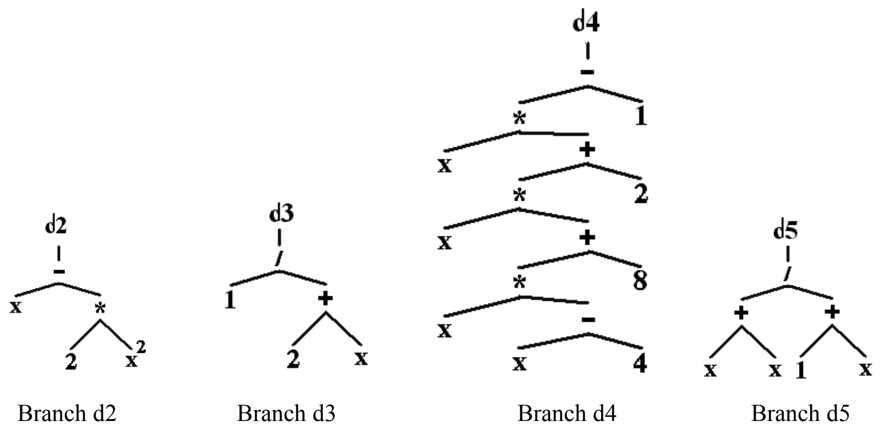
5.2 Introns

In GP, the individual growth in size without any improvement in performance is known as bloat (Langdon and Poli, 1997). Some of the identified causes that produce bloating are the presence of introns (Harries and Smith, 1998; Nordin and Banzhaf, 1995; Soule et al., 1996). Introns are parts of a chromosome that does not affect the individual evaluation (phenotype).
Introns are considered as a sort of structures that protect part of the chromosome from destructive effects produced by genetic operators (Nordin et al., 1996). However, some researchers argue that introns have more negative effects than beneficial. In the literature, diverse mechanisms for destructing introns are reported: remove of redundant code (introns) by means of an edition operator (Koza, 1992), penalisation of individual size (Nordin and Banzhaf, 1995; Zhang and Muhlenbein, 1998), constraint operators (Langdon, 1999; Sims, 1993) and alternatives selection schema (Harries and Smith, 1998).

In the case of MBGP, two types of introns are identified. Analysing solution of problem b.6., its structure is graphically represented in Figure 12. In this Figure, content of branches are detailed. Note that content of branch d4 is repeated more than one time. However, only one of these duplicated branches has a non-zero coefficient. During evolution process, copies of branch d4 are kept in a same individual. Thus, it is protected from destruction by the effects of crossover and mutation. The second type of intron is presented in branch d9. Content of such branch is independent of any variable. Equation (5) shows the structure of branch d9 prior to simplification. When this expression is simplified, it remains -1. The effect of this sort of intron is to reduce the number of branches needed for approximating a function.

Introns were not presented in problems that have the largest range. The number of branches in these problems was up to 10.

$$\frac{\frac{x-x}{x^2}-\frac{x}{x}}{\frac{x-x(x-x)}{x}}=-1 \tag{5}$$



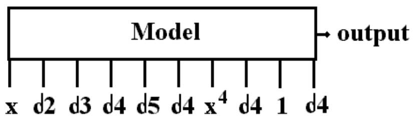


Fig. 12. Solution structure of problem b.6.

6 Conclusions

The multi-branches representation for genetic programming analysed in this paper has proved to be powerful. It has been tested on function approximation problems and results showed to be promising. It was also observed that complexity tends to be reduced by using this representation. It is also relevant to note that intros in multi-branches representation are easily detected and show beneficial effects: allow reducing the number of branches and the effect of destructing a relevant branch is reduced.

Further studies will focused on both the flexibility of this representation in diverse domains and the effects and control of introns.

Acknowledgements

Authors would like to thank the financial support of Consejo Nacional de Ciencia y Tecnología (CONACyT), México, under the project 40602-A, PAPPIT-UNAM under the project EN100201.

References

1. ANGELINE P.J. (1996). Parse trees. *Evolutionary Computation 1, basic algorithms and operators*. Edited by T. Back, D.B. Fogel and T. Michalewickz.
2. ANGELINE P.J. AND J.B. POLLACK (1994). Co–evolving high level representation *Artificial life III* (C.G. Langton, ed.), Addison Wesley, pp 55–71.
3. CRAMER N.L. (1985) A representation for the adaptative generation of simple sequencial programs. In *Proc. 1st. Int. Conference on Genetic Algorithms,* Pittsburg, PA, July 1985 (J.J. Grefenstette, ed.), pp 183 – 187.
4. DE JONG, E.D. AND J.B. POLLACK (2003) Multi-Objective Methods for Tree Size Control, *Genetic Programming and Evolvable Machines*, **4**(3), pp. 211-234.
5. HARRIES K. AND P.W.H. SMITH (1998) Code Growth, Explicitly Defined Introns and Alternative Selection Schemes, *Evolutionary Computation*, **6**(4), pp 346-364.
6. HINCHIFFE, M., H. HIDEN, B. MCKAY, M. WILLIS, M. THAM AND G. BARTON (1996) Modelling Chemical Process System using multi-gene Genetic Programming Algorithm. In *Late Breaking Papers at the Genetic Programming Conference* (J.R. Koza, ed.) Stanford University, C.A. Stanford Bookstore, pp. 56-65.
7. IVANKHNENKO, A.G. (1971) Polynomial Theory of Complex Systems, *IEEE Trans. on Systems, Man and Cybernetics*, **1**(4), pp. 364-378.
8. KEIJZER M. (2003). Improving Symbolic Regresion with Interval Arithmetic and Linear Scaling. In *Genetic Programming, 6th European Conference, EuroGP 2003.* Essex, U.K. (Ryan et al., eds.), pp. 70-82.
9. KEIJZER M. AND V. BABOVIC (2000) Genetic Programming, Ensemble Methods and the Bias/Variance Tradeoff–introductory Investigations. In *Genetic Programming, European Conference, EuroGP 2000 (Edinburgh)* (Poli et al., eds.) LNCS Vol. 1802, Springer Verlag, 15-16 April 2000, pp. 76 – 90.
10. KOZA, J.R. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. Cambridge Massachuset. MIT Press.
11. KOZA, J.R. (1994*). Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge Massachuset. MIT Press.
12. LANGDON, W.B. (1995) Evolving Data Structures with Genetic Programming. In *Proceedings of the Sixth International Conference on Genetic Algorithms* (Eshelman, editor), Morgan Kaufmann, pp. 295-302.
13. LANGDON W. B. (1999) Size Fair and Homologous Tree Genetic Programming Crossovers. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'99*, Vol 2, (Banzhaf et al., eds.), Morgan Kaufmann, pp 1092-1097.
14. LANGDON W.B. AND R. POLI (1997) Fitness Causes Bloat. *Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing*. Springer-Verlag London (Chawdhry et al., eds.), pp 13-22.
15. NIKOLAEV I.N AND H. IBA (2001). Accelerated Genetic Programming of Polynomials. In *Genetic Programming and Evolvable Machines*, **2**(3), pp 231-258.
16. NORDIN P., F. FRANCONE AND W. BANZHAF (1996) Explicitly defined introns and destructive crossover. In *Advances in Genetic programming* (Angeline and Kinnear, eds), chapter 6, MIT Press, Cambridge, MA, USA, pp. 111-134.

17. NORDIN, J.P. AND W. BANZHAF (1995) Complexity Compression and Evolution. In *Proceedings of Sixth International Conference of Genetic Algorithms*, (Eshelman, ed.), Morgan Kaufmann, San Mateo, CA, pp. 310-317.
18. OLIVER, C. AND K. RODRÍGUEZ (2002) Estructuta de Arbol vs Estructura Polinomial con Programación Genética en el Modelado de Variables Climatológicas. In *1er Congreso Español de Algoritmos Evolutivos y Bioinspirados AEB'02*, Merida, España, pp. 124-130.
19. RODRÍGUEZ-VÁZQUEZ, K., C.M. FONSECA AND P.J. FLEMING (1997b) Multiobjective Genetic Programming: A Non-Lineat System Identification Application. *Late Breaking Paper at the 2nd Int. Genetic Programming 97 Conference*, Stanford University, pp. 207-212.
20. RODRÍGUEZ-VÁZQUEZ, K. AND P.J. FLEMING (1999) *Controlling Tree Size Growth in Genetic Programming*. Research Report No. 746. Department of Automatic Control and Systems Engineering, University of Sheffield, United Kingdom.
21. RODRÍGUEZ, K. AND C. OLIVER (2003) Divide and Conquer: Genetic Programming Based on Multiple Branches Encoding. In *Genetic Programming, 6th European Conference, EuroGP 2003* (Ryan et al., eds.), Essez, U.K., pp. 218-228.
22. ROSCA J.P. AND D.H. BALLARD (1996) Discovery of subroutines. In *Advances in Genetic Programming*, Vol. 2 (Angeline and Kinnear, eds.) Cambridge Massachuset, MIT Press, pp 177–202.
23. STREETER M. AND L.A. BECKER (2001) Automated Discovery of Numerical Approximation Formulae via Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'01*, San Francisco CA. (Spector et al., eds.), Morgam Kaufmann, 7–11 July 2001, pp 147–154.
24. STREETER M. AND L.A. BECKER (2003) Automated Discovery of Numerical Approximation Formulae via Genetic Programming. *Genetic Programming and Evolvable Machines*, **4**(3), pp. 255-286.
25. SOULE, T., J.A. FOSTER AND J. DICKINSON (1996) Code growth in genetic programming. In Genetic Programming 1996: Proceedings of the First Annual Conference (Koza, ed.), Stanford University, CA, USA, pp. 215-223.
26. SIMS, K. (1993) Interactive Evolution of Equations for Procedural Models. *The Visual Computer*, **9**, pp. 466-476.
27. ZHANG, B.-T. AND H. MUHLENBEIN (1995) Balancing Accuracy and Parsimony in Genetic Programming, *Evolutionary Computation*, **3**(1), pp.17-38.