Toward an Alternative Comparison between Different Genetic Programming Systems

Nguyen Xuan Hoai, R. I. (Bob) McKay, D. Essam, and H.A. Abbass

School of Information Technology and Electrical Engineering,
Australian Defence Force Academy, University College,
University of New South Wales, ACT 2600, Australia

x.nguyen, b.mckay, d.essam, h.abbass@adfa.edu.au

Abstract. In this paper, we use multi-objective techniques to compare different genetic programming systems, permitting our comparison to concentrate on the effect of representation and separate out the effects of different search space sizes and search algorithms. Experimental results are given, comparing the performance and search behavior of Tree Adjoining Grammar Guided Genetic Programming (TAG3P) and Standard Genetic Programming (GP) on some standard problems.

1 Introduction

Since Koza's initial book on genetic programming (GP) [19], a wide range of new systems have been proposed. Typically, when each new system is introduced, it is compared with existing GP systems. The comparisons usually report on the new system's better performance over standard GP when solving particular problems. The reports contain descriptive statistics, such as cumulative frequencies, number of independent runs and the number of individuals that must be processed to yield a success with 99% probability. However, it is generally the case that the new system differs from previous systems over a number of dimensions (search space size, structure and representation, evolutionary operators, feasibility constraints, search algorithm, genotype-to-phenotype map, decoding, evaluation etc). While reporting the above statistics is important, we agree with [7, 8, 15] that it is also necessary to understand the causes of the differences. It is all too easy to assign the improvement from a new system to differences in representation or operators when simple changes in search space size may be more important. In particular, later in the paper we will show how different types of bounds for chromosome complexity in bounded search spaces can be an important contributor to differences between GP systems, potentially masking the effects of the underlying representation changes.

In this paper, we argue that the multi-objective framework can help to solve some of these difficulties. As a test case, we compare a fairly new genetic programming system, Tree Adjoining Grammar Guided Genetic Programming Systems (TAG3P) [12], with standard GP on two standard problems from the literature. The paper proceeds as follows. In Section 2, a brief introduction to TAG3P is given. Section 3 contains our discussion of the use of multi-objective techniques to compare TAG3P and standard GP in search spaces of equivalent size. Our experiments and discussion are presented in Section 4. Finally, Section 5 concludes this paper.

2 Tree Adjoining Grammar Guided Genetic Programming

In this section, we briefly review tree adjoining grammars and the new GP system, TAG3P (see [11,12] for more details). We also raise concerns about the difficulties of making meaningful comparisons between TAG3P and GP.

2. 1 Tree Adjoining Grammars

Tree Adjoining Grammars (TAGs) are tree-rewriting systems, originally proposed for natural language processing [17]. [18] gives a standard algorithm for converting any Context Free Grammar (CFG) G into a (lexicalised) TAG G_{lex}. Briefly, a TAG system consists of a set of elementary trees $E = I \cup A$ (initial and auxiliary respectively, denoted by α and β). All nodes are labeled grammar symbols, interior labels being restricted to non-terminals. The frontier of an auxiliary tree must contain a 'foot node', with the same label as the root (signified by *). All other non-terminal symbols on the frontier of an elementary tree are available for substitution.

An X-type tree has a root labeled X. TAG systems generate conventional CFG trees (the 'derived' tree) from a derivation tree, encoding the history of substitutions and adjunctions generating the derived tree. Starting from an α tree at the root, each branch records the address for adjunction or substitution, and the elementary tree used

Adjunction takes a tree γ with an interior label A, and an A-type auxiliary tree β, producing a new tree γ' by disconnecting the sub-tree γ_1 rooted at A, attaching β to replace it, and finally re-attaching γ_1 to the foot node of β.

TAGs have a number of advantages for GP:

- Derivation trees in TAGs are more fine-grained structures than those of CFG.
- They are compact (each elementary node encodes a number of CFG derivations).
- They are closer to a semantic representation [6].
- Derivation and derived trees provide a natural genotype-to-phenotype map.
- In growing a derivation tree f, one can stop anytime and have a valid derived tree.

We call the last property "feasibility". Feasibility helps TAG3P (described in next subsection) control the exact size of its chromosomes, and also to implement a wide range of genetic operators. A number of them are bio-inspired [12].

2.2 Tree Adjoining Grammar Guided Genetic Programming (TAG3P)

To relate TAG3P to previous systems using CFGs [26], we frame the discussion in terms of a CFG G and corresponding LTAG, G_{lex}. TAG3P evolves the derivation trees in G_{lex} (genotype) instead of the derived trees (the derivation trees in G – phenotype), creating a genotype-phenotype map, usually many-to-one. TAG3P may be specified as follows [19]:

Program representation: the derivation trees in LTAG G_{lex}.

Parameters: minimum size of genomes (MIN_SIZE), maximum size of genomes (MAX_SIZE), size of population (POP_SIZE), maximum number of generations (MAX_GEN) and probabilities for genetic operators.

Initialization procedure: Each individual is generated by randomly growing a derivation tree in G_{lex} to a size randomly chosen between MIN_SIZE and MAX_SIZE (unlike most GP systems, which use depth bounds). Because of TAG feasibility, this always generates valid individuals of exact size. An alternative ramped half-and-half initialization generates half of the derivation trees randomly but of full shape.

Fitness Evaluation: an individual derivation is first mapped to the derived CFG tree. The expression defined by the derived tree is semantically evaluated as in grammar guided genetic programming (GGGP) [26].

Main Genetic operators: sub-tree crossover and sub-tree mutation.

In sub-tree crossover, two individuals are selected based on their fitness. Points with the same adjunction label are randomly selected within each tree and the two sub-trees are exchanged. If no such points are found, the two individuals are discarded and the process is repeated until a bound is exceeded.

In sub-tree mutation, a point in the derivation tree is chosen at random and the sub-tree rooted at that point is replaced by a newly generated sub-derivation tree.

2.3 Difficulties in Making Meaningful Comparison Between TAG3P and GP

TAG3P differs in many ways from standard GP. It uses grammars (a LTAG and a CFG), can solve typed problem domains; can handle context-sensitive information; has a genotype-to-phenotype map (therefore different search space); has different genetic operators and a different type of bound on chromosomal complexity (length rather than depth). If TAG3P and GP performance differ, it could be a result of any or all of these differences. Consequently, understanding the relationship between TAG3P and GP performance is very challenging.

Firstly, to ensure the grammars give no favorable bias for TAG3P they are chosen as follows. From a description of a GP set of functions and terminals, a context-free grammar, G, is created according to [26] (page 130) thus ensuring G is bias-free and the correspondence, between derivation trees of G and parse trees of GP, is one-to-one. G_{lex} is then derived from G using the algorithm in [18]. Secondly, in order to evaluate fitness of an individual (derivation tree in G_{lex}), it is decoded first into a derivation tree in G, then to the equivalent parse tree in GP. On this final parse tree the evaluation is systematically processed in the same way as in GP. Next, tunable parameters in the two systems are set as uniformly as possible.

However GP systems usually use a bounded search to limit chromosome complexity. In TAG3P, the bound is the maximum number of nodes, whereas in standard GP it is the maximum allowed depth (although some recent GP systems use a maximum number of nodes [20]). It is virtually impossible to adjust these bounds to give the same phenotype space in each, because there is no systematic mapping between nodes in TAG3P – elementary trees – and nodes in GP (see examples in the next section). This problem is not restricted to TAG3P; we believe it applies equally to a range of other GP systems, especially those that use grammars and/or genotype-to-phenotype maps (eg Linear GP [1], GE [24], and GEP [14]). Therefore, while the work in this paper relates only to the problem of making comparison between TAG3P and GP, it has clear implications for other GP systems.

3 The Use of Multi-objective Techniques for Comparisons

To solve the problem of adjusting chromosome complexity bounds, one option might be to remove the bounds. However, unbounded GP systems usually bloat, which is why bounds on the chromosome complexity are usually set. Bloat is a well-documented phenomenon in GP [3, 20, 25]; [20] suggests that code bloat is inevitable for all evolutionary systems that use length-variant chromosomes. But code bloat has serious effects on search performance [3,20,25]; and there is no reason to expect that these effects are invariant between different GP systems. Hence removing bounds simply adds one more confounding variable. Equally, when solving inductive learning problems, one is not indifferent to solution size - short solutions are preferred [23].

One alternative is to use a combined objective. Although there are many ways to combat bloat with single objective selection by integrating chromosome complexity into the fitness function [4, 16, 27], this introduces significant problems for our purpose. When using chromosome complexity as part of a single objective fitness function, some tunable parameters must be defined to determine how much the chromosome complexity of an individual will affect its fitness. It is at least difficult and time-consuming to find an optimal setting for these parameters; it is virtually impossible to find one, which is fair to both systems. Hence we argue that multi-objective selection is more appropriate for this purpose, especially since in [2] it is shown that multi-objective selection can outperform single objective selection in combating bloat. To understand the effect of the difference in search space representation and operators between TAG3P and standard GP, we use an unbounded search space, but apply multi-objective selection, with chromosome complexity as the second objective, to combat bloat. This has two main advantages. Searching with this multi-objective selection pressure can solve the problem of code bloat [2, 5, 9, 10, 21], therefore permit a more meaningful comparison. Moreover, multi-objective selection can unify very different GP-search spaces into a single objective space, providing a common ground for looking into the search performance and behavior of GP systems that use different representations and different genetic operators.

In this paper, we use the strength Pareto evolutionary algorithm (SPEA2) [28, 29], a state-of-the-art evolutionary multi-objective optimization (EMO) algorithm, to implement the comparison between GP and TAG3P. We chose SPEA2 because of its superior performance over other EMO algorithms [28, 29] and its efficiencies in reducing bloat in GP [2]. Moreover, Pareto fitness calculation in SPEA2 uses density estimation techniques, helping to promote diversity in the objective space. In turn, that reduces the common effect whereby the whole population may converge to the individual with minimal chromosome complexity [9]. This was sufficiently effective that in our experiments using SPEA2, we did not observe the effect. It is also possible to use other EMO algorithms, and we intend to investigate alternatives in the future.

4 Experiments

Using the SPEA2 multi-objective selection, we compared TAG3P with standard GP on two standard test problems: the 6-multiplexer, and symbolic regression. In the 6-multiplexer problem [19], a 6-multiplexer uses two address lines to output one of four data lines and the task is to learn this function from its 64 possible fitness cases using function set F={IF, AND, OR, NOT} and terminal set {a0, a1, d0, d1, d2, d3}. In the symbolic regression problem [19], the task is to learn the quadratic function: X4+X3+X2+X from 20 sample points in [-1..1]; the function and terminal set are F={+,-,*,/,sin, cos, exp, rlog} and T={X}. They were chosen as frequently-used GP test-beds, the (shortest) solutions being known. The fitness values of the first are discrete and finite; of the second, continuous and infinite.

4.1 Experimental Design

As in [2], we use weak Pareto non-domination selection. The second objective is the size (in number of nodes) of the parse trees, on which fitness is evaluated. To study the effect of population initialization, for each problem, we ran three systems. The first was TAG3P, the second was GP with the initial population translated from the initial population of TAG3P (GP-I) such that they had the same initial population in the phenotype space, and the third was GP with Ramped-Half-and-Half initialization (GP-RHH). The range of size in the initial population of TAG3P and the maximum depth of the initial population in GP-RHH was calibrated so that on average the size of the final parse trees (phenotype) in the TAG3P initial population was the same as the size of parse trees (genotype) in GP. To investigate the effect of variance in population size, and number of generations, for each problem, we experimented with three settings of population and number of generations. All other parameters (such as genetic operator rates) are the same for all three systems. We can summarize the parameter settings in two categories:

Fixed common settings: Genetic operators – subtree crossover and subtree mutation; crossover rate=0.9; mutation rate=0.1; Calibrated initial size for TAG3P: 10-60 (for 6-multiplexer problem) and 2-60 (for symbolic regression); Calibrated maximal depth for initial population of GP-RHH: 6 (for 6-multiplexer problem – GP-RHH6) and 8 (for symbolic regression – GP-RHH8). In SPEA2 settings, weak dominance was used and the size of the archive was set equal to the size of the main population. No maximal limits were set on the depth or size of any tree. All runs only finished when the maximum number of generations was reached.

Varied common settings: On each problem, three varied settings on population size and number of generations were given to all three systems (TAG3P, GP-I and GP-RHH). The experimented population sizes were 250, 500, and 1000, while the maximum numbers of generations were 101, 51, and 26 respectively.

There were 100 runs allocated for each system for each varied setting, making the total number of runs 1800. The *i*th run of each setting used the same initial random key as every other *i*th run. The keys themselves were generated at random. All the runs used the same random generator. The grammars for TAG3P were generated using algorithms in [26] (page 130) and in [18].

The grammars for the 6-multiplexer problem were: G={V={B}, T={a0, a1, d0, d1, d2, d3, d4, and, or, not, if}, P, {B}}, where the rule set P is defined as follows:

B □ a0| a1| d0| d1| d2| d3.

B □ B and B| B or B | not B | if B B B

G_{lex}={V={B, TL}, T={a0, a1, d0, d1, d2, d3, and, or, not, if}, I, A}, where I □ A is depicted in Figure 2. TL is a lexicon that can be substituted by one lexeme in {a0, a1, d0, d1, d2, d3}.

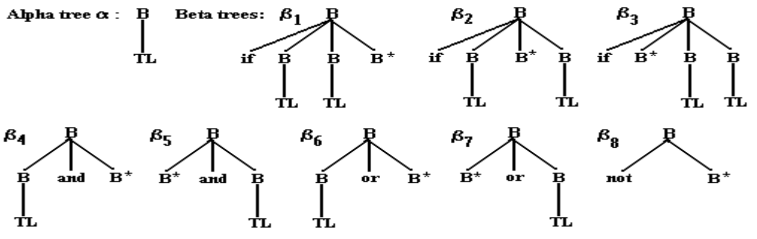

Fig. 1. Elementary trees for G_{lex}

The grammars for the symbolic regression problem were: G = (V={EXP, PRE, OP, VAR,},T= {X, sin, cos, log, ep, +, -, *, /, (,)},P,{EXP}} where ep is the exponential function, and the rule set P is as follows

EXP□ EXP OP EXP | PRE (EXP) | VAR

OP□ + | - | * | /

PRE□ cos | sin | rlog | ep

VAR□ X

G_{lex}= {V={EXP, PRE, OP,VAR},T={X, sin, cos, log, ep,+, -, *, /, (,)}, I, A) where I□ A is as in Figure 3.

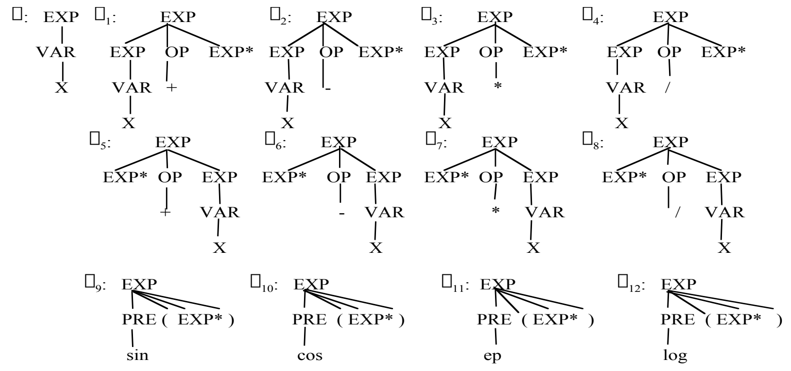

Fig. 2. Elementary trees for G_{lex}.

4.2 Results

For the 6-multiplexer problem, Table 1 summarizes the results of the three systems (TAG3P, GP-I, GP-RHH6) based on 900 runs for three settings. Column 4 contains the percentage of runs that found a solution (i.e first objective value is 0). Column 5 gives the number of runs that found an optimal-size solution. In this problem the optimal size of solutions is 10. An example of such solutions is IF(a0, IF(a1, d3, d2), IF(a1, d1, d0)). Column 6 depicts the average size of solutions, and column 7 is the standard deviation of the data used to calculate column 6.

For the symbolic regression problem, Table 2 summarizes the results of the three systems (TAG3P, GP-I, GP-RHH6) based on 900 runs for three settings. The columns are as above. For this problem, the optimal solution size is 13, (X+X*X)*(X/X+X*X) being an example.

Table1. 6-Multiplexer Results

GEN	POP	System	Succ	Nº perfect	Avg sol. S ize	Std
101	250	**TAG3P**	58	35	11.45	2.51
101	250	**GP-I**	45	35	11.05	3.30
101	250	**GP-RHH6**	43	15	16.28	9.22
51	500	**TAG3P**	82	45	11.97	3.27
51	500	**GP-I**	74	52	11.80	4.05
51	500	**GP-RHH6**	63	27	18.62	12.62
26	1000	**TAG3P**	87	59	11.56	4.28
26	1000	**GP-I**	75	38	13.13	5.85
26	1000	**GP-RHH6**	62	25	32.37	46.00

Table2. Symbolic Regression Results

Gen	Pop	Systems	succ	Nº perfect	Avg sol.	Std
101	250	**TAG3P**	46	35	13.85	1.74
101	250	**GP-I**	34	23	13.74	1.24
101	250	**GP-**	18	16	13.17	0.50
51	500	**TAG3P**	57	39	14.37	3.12
51	500	**GP-I**	39	30	14.00	2.34
51	500	**GP-**	15	11	13.53	0.96
26	1000	**TAG3P**	53	23	15.87	4.33
26	1000	**GP-I**	43	21	15.19	2.71
26	1000	**GP-**	17	8	17.24	9.40

Figures 3, 4, 5 depict the cumulative frequencies and time series of average fitness of the population in six systems. Due to limited space, only those of the first setting of population and number of generations (population = 250, generations = 101) are given. The corresponding figures for other settings are quite similar.

4.3 Discussion of Results

On the two standard problems, TAG3P was better than GP-I and GP-RHH in finding solutions with (first objective) fitness 0. Moreover, it was also better than the others at finding solutions with optimal size (except one case – table 1, row 4). One reason is TAG3P's faster convergence in the size objective. For GP–I and GP-RHH, the average size also converges to the size of optimal solution in the 6-multiplexer problem but not in the symbolic regression problem. This is explained by the discreteness (only 64 values) of the first objective in the 6-multiplexer problem, so that any large individual has little chance to survive under the multi-objective selection pressure. Therefore, the convergence in average size was common for all three systems. On the other hand, the first objective in symbolic regression problems is continuous, so that a large individual has more chance to survive under the multi-objective selection pressure. Since TAG3P converges faster than the others, it finds solutions more frequently in the subspace of smaller trees, whereas GP-I and GP-H search more frequently in the subspace of bigger trees. It can also be seen from the different results of GP-I and GP-RHH, that the method for seeding the initial population is important for the search performance and behavior of GP. This result is contrary to the results found in [22], where the seeding method used to generate the initial population had no significant impact on the search performance and behavior of GP, in a bounded search space and under single objective selection pressure. We see better performance from GP-I than GP-RHH. Since in TAG3P the initial population could be uniformly distributed in size (due to TAG feasibility), the initial population in GP-I (and in TAG3P) was more spread in the objective space than in GP-RHH. The discussions above are also supported by the distribution of tree size over time for all 1800 runs. However, due to limited space, those detailed results are omitted here. In [13], the comparative results on symbolic regression, using single objective and bounded search space, we reported that TAG3P was far better than GP. At that time, we believed that it was because of the new representation and/or operators. The results above show that while those factors did contribute to the better performance of TAG3P, the bound on search spaces in [13] was a greater contributor to TAG3P's better performance compared with GP.

5 Conclusion

In this paper, we have introduced and discussed an alternative type of comparison between different GP systems. In particular, we pointed out that different types of bounds on chromosome complexity, which might derive from different representations, makes it hard to determine the causes of different performance between GP systems. We have argued that the use of multi-objective selection, coupled with an unbounded search space, can help to understand the effects of search space size. Thus, by using an EMO algorithm (SPEA2), we were able to make this alternative type of comparison between our GP system (TAG3P) and standard GP on two standard problems. The results show the differences in search performance and behavior between TAG3P and standard GP. We have also found that setting different types of bounds on search spaces was the prime factor in the better performance of TAG3P compared with GP in [13]. Moreover, the method for seeding the initial population in GP is very important, contrary to previous results.

We now plan to use the same technique to make meaningful comparison between TAG3P and other grammar guided genetic programming systems on typed problems. We also plan to compare other EMO algorithms with SPEA2.

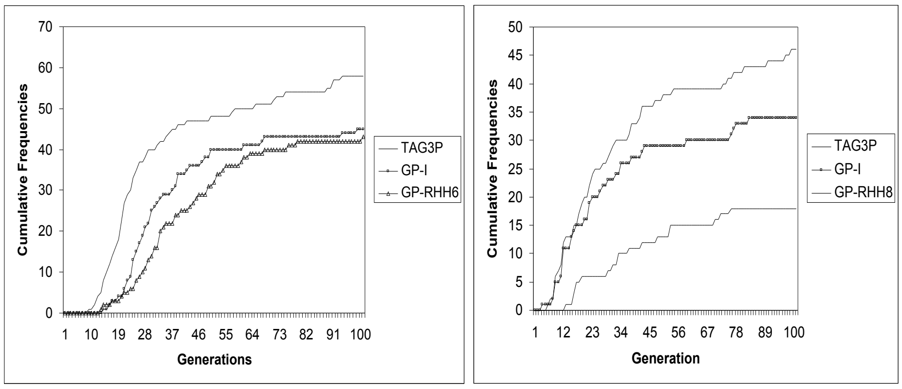

Fig. 3. Cumulative frequencies of success. POP_SIZE=250, MAX_GEN=101. 6-multiplexer results on left, symbolic regression on right.

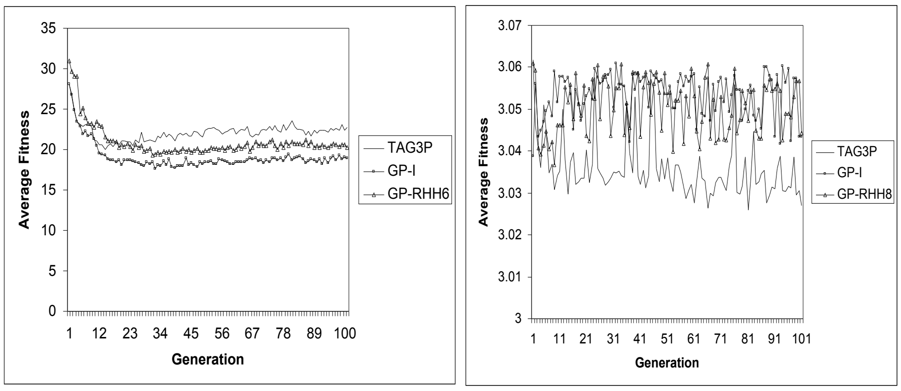

Fig. 4. Time series of average (first) fitness. POP_SIZE=250, MAX_GEN=101; 6-multiplexer results on left, symbolic regression on right.

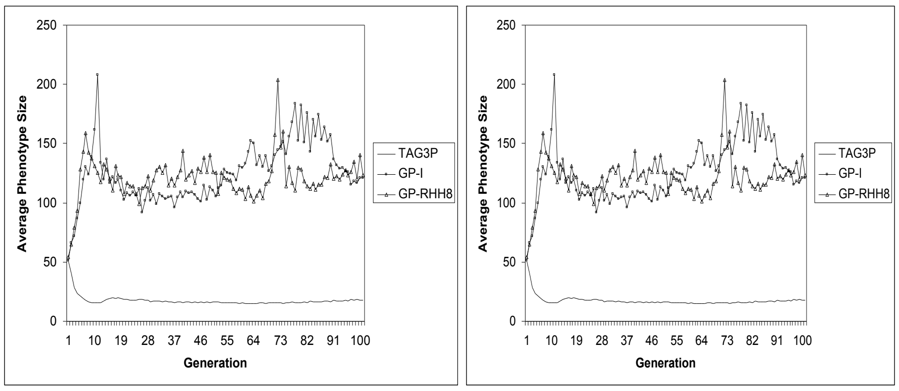

Fig. 5. Time series of average phenotype size. POP_SIZE=250, MAX_GEN=101; 6-multiplexer results on left, symbolic regression on right.

Acknowledgements

The authors would like to express their thanks to Dr. Zitzler for his prompt replies to our questions related to SPEA2.

References

1. Banzhaf W., Nordin P., Keller R.E., and Francone F.D.: *Genetic Programming: An Introduction*. Morgan Kaufmann Pub (1998).
2. Bleuler S., Brack M., Thiele L., and Zitzler E.: Multiobjective Genetic Programming: Reducing Bloat Using SPEA2. *Proc. Congress on Evolutionary Computation* (CEC'2001) (2001) 536-543.
3. Blickle T. and Thiele L.: Genetic Programming and Redundancy. In: *Genetic Algorithms within the Framework of Evolutionary Computation*, Hopf J. (Ed.), (1994) 33-38.
4. Blickle T.: Evolving Compact Solutions in Genetic Programming: A Case Study. In: *PPSN IV*, Voigt H.M., Ebeling W., Rechenberg I,, and Schwefel P. (Eds.), Springer-Verlag (1996) 564-573.
5. Bot M.C.J: Improving Induction of Linear Classification Tree with Genetic Programming. In: *Proc. The Genetic and Evolutionary Computation* (GECCO'2000), Darrell Whitley et al (Eds.). Morgan-Kaufman Publishers (2000) 403-410.
6. Candito M. H. and Kahane S.: Can the TAG Derivation Tree Represent a Semantic Graph? An Answer in the Light of Meaning-Text Theory. In: *Proc. of TAG+4*, Philadelphia, (1999) 25-28.
7. Daida J. M., Ampy D.S., Ratanasavetavadhana M., Li H., and Chaudhri O.A.: Challenges with Verification, Repeatability, and Meaningful Comparison in Genetic Programming: Gibson's Magic. Accessed at http://citeseer.nj.nec.com/257412.html. Date: 11-10-2003.
8. Daida, J.M., Ross S. J., McClain J.J., Ampy D.S., and Holczer M.: Challenges with Verification, Repeatability, and Meaningful Comparison in Genetic Programming. In: *Genetic Programming 97: Proceedings of the Second Annual Conference*, Koza J.R., Deb K., Dorigo M. et al (Eds.), Morgan Kaufman Publishers (1998) 122-127.
9. Dejong E.D. and Pollack J.B.: Multi-Objective Methods for Tree Size Control. *Genetic Programming and Evolvable Machines*, 4 (2003) 211-233.
10. Ekart A. and Nemeth S. Z.: Selection Based on the Pareto Non-domination Criterion for Controlling Code Growth in Genetic Programming *Genetic Programming & Evolvable Machines* 2(1) (2001) 61-73.
11. Nguyen Xuan Hoai and McKay R.I.: A Framework for Tree Adjunct Grammar Guided Genetic Programming. In: *Proceedings of Post Graduate ADFA Conference on Computer Science (PACCS'01)*, H.A. Abbass and M. Barlow (Eds), (2001) 93-99.

12. Nguyen Xuan Hoai, McKay R.I., and Abbass H.A.: Tree Adjoining Grammars, Language Bias, and Genetic Programming. In *Proceedings of EuroGP 2003*, Ryan C. et al (Eds), LNCS 2610, Springer Verlag, (2003) 335-344.
13. Nguyen Xuan Hoai, McKay R.I., Essam, D., and Chau R.: Solving the Symbolic Regression Problem with Tree Adjunct Grammar Guided Genetic Programming: The Comparative Result. In Proceedings of Congress on Evolutionary Computation (CEC'2002), Hawai (2002) 1326-1331.
14. Ferreira C., Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. *Complex Systems*, 13 (2), (2001) 87-129.
15. Haynes, T.: Perturbing the Representation, Decoding, and Evaluation of Chromosomes. In: *Genetic Programming 98: Proceedings of the Third Annual Conference*, Koza J.R et al (Eds.) Morgan Kaufman Publishers (1998) 122-127.
16. Iba H., Garis H., and Sato T.: Genetic Programming Using a Minimum Description Length Principle. In: *Advances in Genetic Programming*, Kinnear Jr K.E. (Ed.), MIT Press (1994), Chapter 12.
17. Joshi A. K., Levy L. S., and Takahashi M.: Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 10 (1), (1975) 136-163.
18. Joshi, A. K. and Schabes, Y.: Tree Adjoining Grammars. In: *Handbook of Formal Languages*, Rozenberg G. and Saloma A. (Eds.) Springer-Verlag, (1997) 69-123.
19. Koza, J. : *Genetic Programming*. The MIT Press (1992).
20. Langdon W.B. and Poli R.: *Foundations of Genetic Programming*. Springer-Verlag (2002).
21. Langdon W.B.: *Genetic Programming + Data Structure =Automatic Programming*. Kluwer Academic (1998).
22. Luke S. and Panait L.: A Survey and Comparison of Tree Generation Algorithms. In: *Proceedings of The Genetic and Evolutionary Computation* (GECCO'2001), Spector L. et al (Eds.), Morgan Kaufman Publishers (2001) 81-88.
23. Michell T.M.: *Machine Learning*. McGraw-Hill (1997).
24. O'Neil M. and Ryan C.: Grammatical Evolution. *IEEE Trans on Evolutionary Computation*, 4 (4), (2000) 349-357.
25. Soule T. and Foster J.: Effects of Code Growth and Parsimony Pressure on Population in Genetic Programming. *Evolutionary Computation*, 6 (4), (1999) 293-309.
26. Whigham P. A.: *Grammatical Bias for Evolutionary Learning*. Ph.D Thesis, University of New South Wales, Australia, (1996).
27. Zhang B.T. and Muhlenbein H.: Balancing Accuracy and Parsimony in Genetic Programming. *Evolutionary Computation*, 3(1), (1995) 17-38.
28. Zitzler E. and Thiele L.: Multi-objective Evolutionary Algorithms: A Comparative Case Study and The Strength Pareto Approach. *IEEE Trans on Evolutionary Computation*, 3 (1), (1999) 257-271.
29. Zitzler E., Laumanns M., and Thiele L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Technical Report 103*, Computer Engineering and Networks Laboratory (TK), ETH Zurich, Switzerland, 2001.