Grammar Bias and Initialisation in Grammar Based Genetic Programming

Eoin Murphy, Erik Hemberg, Miguel Nicolau,
Michael O'Neill, and Anthony Brabazon

Natural Computing Research and Applications Group,
University College Dublin, Ireland
{eoin.murphy,erik.hemberg,miguel.nicolau,m.oneill,anthony.brabazon}@ucd.ie

Abstract. Preferential language biases which are introduced when using Tree-Adjoining Grammars in Grammatical Evolution affect the distribution of generated derivation structures, and as such, present difficulties when designing initialisation methods. Similar initial populations allow for a fairer comparison between different GP methods. This work proposes methods for dealing with these biases and examines their effect on performance over four well known benchmark problems. In addition, a comparison is performed with a previous study that did not employ similar phenotype distributions in their initial populations. It is found that the use of this form of initialisation has a positive effect on performance.

Keywords: Grammatical evolution, Grammar bias, Initialization.

1 Introduction

It has been shown that the form of a grammar and indeed the language biases inherent to that grammar can have a large impact on the performance of grammar-based Genetic Programming (GP) systems [14, 2], such as Grammatical Evolution (GE) [13]. Modification of the grammar, an integral part of the GE algorithm, can cause the algorithm to behave very differently [2]. This is due to the ease with which the language biases in the system can change by just modifying the grammar, effecting how genotypes are mapped into phenotypes.

In a previous study by Murphy et al. [12], GE, which traditionally uses a Context-Free Grammar (CFG), was extended to make use of Tree-Adjoining Grammars (TAG) [6], in the form of Tree-Adjunct Grammatical Evolution (TAGE). A preliminary comparison of the two methods was performed, testing each method on a number of different problems, with TAGE showing improvements in performance, such as finding more correct solutions, as well as finding better solutions in fewer generations, than standard GE on those problems [12]. The transformation from CFG to TAGE modifies the existing language bias as well as introducing new language biases into the GE algorithm. These biases affect the algorithm's ability to generate certain derivation structures and phenotypes, and hence altering the search space [11]. This can be detrimental to the

A. Moraglio et al. (Eds.): EuroGP 2012, LNCS 7244, pp. 85–96, 2012.

generation of common initial populations, an important factor in performing a fair comparison between any two GP systems.

The aim of this study, therefore, is to examine and address the different types of language bias which are introduced by using the TAG representation (see Section 4.1 in order to better align the initial populations of both setups. Biases such as the structural biases imposed by the adjunction operation, as well as those introduced as a result of the grammar transformation, i.e., the loss of explicit biases imposed by the CFG. The study proposes a novel algorithm for the elimination of the grammar transformation biases (see Section 4.3) and goes on to perform a comparison between GE and TAGE. The study observes the effect of these biases on performance and comparing with the results observed by Murphy et al. [12].

In the following section a brief description of GE is given. Section 3 gives an overview of TAGE, the transformation from CFG to TAG and concludes with a sample TAGE derivation. Following this, Section 4 examines the problems involved with creating similar initialisation methods for the two different grammar types. The experimental setup is described in Section 5, with Section 6 outlining the results obtained, as well as providing some discussion on these results. Section 7 concludes the study.

2 Grammatical Evolution

GE is a grammar-based approach to GP, combining aspects of Darwinian natural selection, genetics and molecular biology with the representational power of grammar formalisms [13]. The use of a grammar enables GE to define the legal expressions and structures of an arbitrary language, which in turn allows the possible generated structures and syntax of solutions to be easily modified, something that is not trivial for other forms of GP. In addition to this, the separation of the genotype from phenotype in GE allows genetic operations to be applied to both, extending the search capabilities of GP. GE is considered to be one of the most widely applied GP systems today [10].

Representation in GE consists of a grammar and a chromosome (see Fig. 1). The genotype-phenotype mapping in GE uses codon values from the chromosome to select production rules from the grammar. By performing the modulus operation on these values with the number of possible production choices, productions are selected from the grammar to expand each non-terminal (NT) symbol. Starting from the start symbol, this process, which continues in a left-right manner until there are no more NT leaf nodes to expand, or until the end of the chromosome has been reached, constructs a derivation tree. The phenotype can then be extracted from the leaf nodes of this tree. A sample derivation tree is shown in Fig. 1 along with the grammar and chromosome used to construct it.

3 Tree-Adjunct Grammatical Evolution

TAGE, like GE, uses a representation consisting of a grammar and a chromosome. However, the type of grammar used in this case is a TAG rather than a

CFG. A TAG is defined by a quintuple (T, N, S, I, A) where T is a finite set of terminal symbols; N is a finite set of NT symbols: $T \cap N = \emptyset$; S is the start symbol: $S \in N$; I is a finite set of finite trees called *initial trees* (or α trees); and A is a finite set of finite trees called *auxiliary trees* (or β trees).

The root node of an initial tree is labelled with S and the interior nodes are labelled with NT symbols. Initial tree's leaf nodes are labelled with terminal symbols. Similarly, the interior nodes of auxiliary trees are also labelled with NT symbols, with their leaf nodes being labelled with terminal symbols. However, one special leaf node called the foot node is labelled with the same NT symbol as the root. Foot nodes are marked with * [6].

Initial trees represent the minimal non-recursive structures produced by the grammar, i.e., they contain no repeated NT symbols. Inversely, auxiliary trees of type X represent the minimal recursive structures, which allow recursion upon the NT X [8]. The union of initial trees and auxiliary trees forms the set of *elementary trees*, E; where $I \cap A = \emptyset$ and $I \cup A = E$.

During derivation, the adjunction composition operation joins elementary trees together. Adjunction takes an initial or derived tree a, creating a new derived tree d, by combining a with an auxiliary tree, b. A sub-tree, c is selected from a. The type of the sub-tree (the symbol at its root) is used to select an auxiliary tree, b, of the same type. c is removed from a. b is then attached to a as a sub-tree in place of c and c is attached to b at the position of b's foot node. An example of TAG derivation is provided in Section 3.1.

3.1 TAGE Derivation Example

TAGE generates TAGs from the CFGs used by GE. Joshi and Schabes [6] state that for a *"finitely ambiguous CFG which does not generate the empty string, there is a lexicalised tree-adjunct grammar generating the same language and tree set as that CFG"*. An algorithm was provided by Joshi and Schabes [6] for generating such a TAG. The TAG produced from Fig. 1 is shown in Fig. 2.

Derivation in TAGE is different to GE. Unlike GE derivation trees whose nodes are labeled by symbols, the nodes of a TAGE derivation tree are labelled by elementary trees. The edges between those nodes are labelled with the address of a node in the tree labelling the parent node. It is at this address that the auxiliary tree labelling the child is to be adjuncted. A derived tree in TAGE is a tree of symbols, similar to GE's derivation tree, resulting from the application of the adjunction operations defined in the TAGE derivation tree.

Given the TAG G, where $T = \{$`X, Y, +, -`$\}$, $N = \{$`<e>,<o>,<v>`$\}$, $S =$`<e>` and I and A are shown in Fig. 2, derivation using the chromosome from Fig. 1 operates as follows. The first codon value, `12`, is read and is used to choose an initial tree based on the number of trees in I. Using the same mapping function as GE, `12 mod 2 = 0`, the zero-th tree, α_0, is chosen from I. This tree is set as the root node of `t`, the derivation tree (as seen in Fig. 3(a)).

Next, a location to perform adjunction must be chosen. The vector `N` is created of the adjunctable addresses available within all nodes (trees) contained within `t`. An adjunctable address in a tree is the breadth first traversal index of a node

labelled with a NT symbol, of which there is an auxiliary tree of that type and there is currently no auxiliary tree already adjoined at that index. In this case `N` = $\{\alpha_0[0]\}$ (the zeroth node of α_0), so a codon is read and an address is selected from `N`, `3 mod 1 = 0` indicating which address to choose, `N[0]`. Adjunction will be performed at $\alpha_0[0]$, or index `0` of tree α_0, `<e>`. An auxiliary tree is now chosen from A that is of the type `T`, i.e., the label of its root node is `T`, where `T` is the label of the node where adjunction is being performed. In this case `T = <e>`. There are 8 such trees in A, Reading the next codon, `7`, `7 mod 8 = 7`, therefore β_7 is chosen. This is added to `t` as a child of the tree being adjoining to, labelling the edge with the address `0`, see Fig. 3(b). The adjunctable addresses in β_7 will be added to `N` on the next pass of the algorithm. This process is repeated until all remaining codons have been read. The resulting derivation and derived trees at each stage of this process can be seen in Fig. 3.

4 Difficulties with Comparing GP Systems

Performing a fair comparison between different GP systems is difficult to achieve. As suggested by Hoai et al. [4], it is easy to assume that the benefits observed when testing a new modification to an algorithm are a direct consequence of the modification in question [1], whereas in reality this can be a flawed assumption. Unless the modification is very localised, there can be far reaching indirect effects, and if these effects influence the starting conditions of the algorithm, performing a comparison can be difficult. This problem is even more evident when comparing completely different algorithms.

These difficulties can be seen when comparing standard GP algorithms with grammar-based versions, as was shown in [4] when comparing GP and TAG3P. The change in representation makes it difficult to create common initial conditions for both algorithms in order to achieve a good comparison.

Comparing similar algorithms with different representations raises an interesting question: having a common initial population (or at least initial populations drawn from similar distributions) is good practice and helps ensure a fair comparison, but should these similar populations be in the genotypic or phenotypic spaces? Search is performed in the genotypic space, whereas the fitness landscape lies in phenotypic space, and depending on the mapping between the two there could be a many to one relationship between genotypes and phenotypes. That is to say, creating initial populations of genotypes for two different representations, e.g., GE and TAGE, would likely result in very different populations of phenotypes. The same can be said for similar populations of phenotypes, the populations of genotypes could be very different (see Fig. 4).

In Murphy et al. [12], a set genotype length was used; as a counter point, this study uses an initialisation method which produces similar sets of derivation (TAG derived) trees, and hence similar sets of phenotypes, for both GE and TAGE. The following subsections outline and address some of the problems faced while attempting to achieve this.

```
Grammar:
<e>:= <e><o><e> | <v>
<o>:= + | -
<v>:= X | Y

Chromosome:
12, 3, 7, 15, 9, 36, 14
```

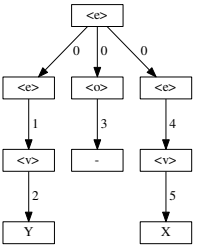
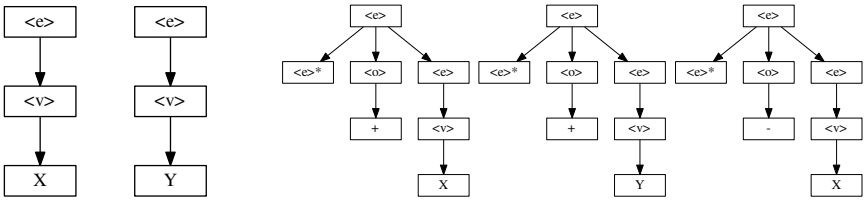
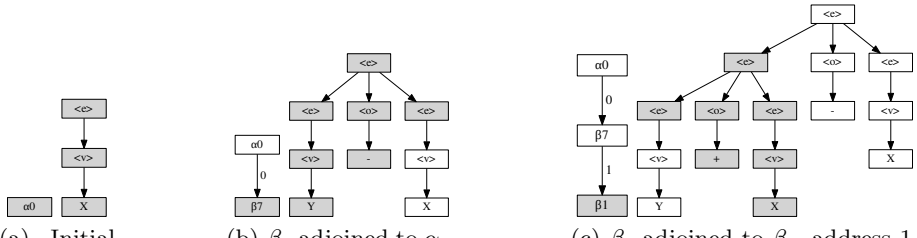


Fig. 1. Sample GE grammar, chromosome and resulting derivation tree (edge labels indicating the order of expansion). <> denotes a non-terminal symbol.

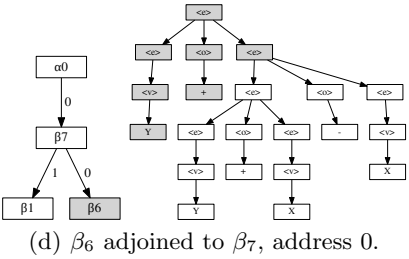


(a) I=$\{\alpha_0, \alpha_1\}$ (b) A=$\{\beta_0, \beta_1, \beta_2 ...\}$

Fig. 2. The initial tree set (I) and a subset of the auxiliary tree set (A) of the TAG produced from the CFG in Fig. 1



(a) Initial tree α_0. (b) β_7 adjoined to α_0, address 0. (c) β_1 adjoined to β_7, address 1.



(d) β_6 adjoined to β_7, address 0.

Fig. 3. The derivation tree (left) and derived tree (right) throughout TAGE derivation. The shaded areas indicate new content added at each step.

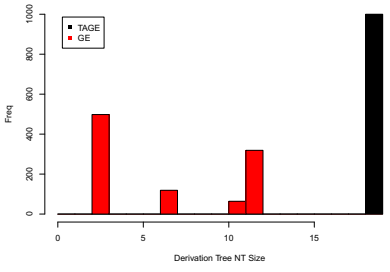


Fig. 4. The distributions of tree size (NT nodes) when initialising to a common genotype length

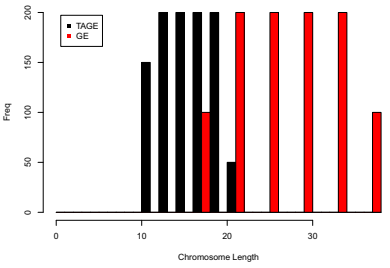


Fig. 5. The distributions of genotype lengths when initialising to common distribution of derivation tree sizes

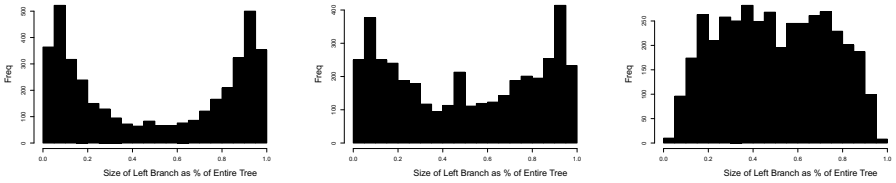
4.1 Initialisation and Transformation Bias

While the typical method of initialisation in GP is the Ramped Half and Half method [7], dividing the population between a minimum and maximum depth interval with half the trees being grown randomly and the other half being grown to be full trees, depth is not as important in GE as in GP. In GE, to ensure that there is a good distribution of phenotypes, the distribution of the number of NT nodes, or tree size is more important. With that in mind, a ramped tree size with a max depth initialisation method is employed by this study (similar to the method used by Harper [2], and PTC2 by Luke [9] without probability tables).

As mentioned in Section 3, it is possible to generate a lexicalised TAG from a finitely ambiguous CFG. However, there are biases inherent to TAGs which affect the probabilities of certain shapes being generated, as well as biases inherent to CFGs that are not preserved by this grammar transformation. Specifically these are an adjunction bias, biases imposed upon the language by the choice of adjunction points, and a grammar transformation bias, introduced when transforming from one grammar type to another. More detail on these is given below.

4.2 Adjunction Bias

While TAGs are said to be both weakly and strongly equivalent to the CFG used to generate them [5], depending on the constraints imposed upon the adjunction operation, biases appear in the shapes of randomly generated derived trees. For example, in the initial implementation of TAGE [12] adjunction is not allowed to be performed on foot nodes of auxiliary trees already in the derivation tree. The result of which is that once an adjunction is performed, the tree can only be expanded at its other adjunctable addresses, preventing the branch contained by the foot node from being expanded. Fig. 6(a) shows the distribution of tree shapes when using the adjunction constraints from Murphy et al. [12]. Tree shape is measured as the percentage of NT nodes used to build the left branch of the tree. This figure shows that the distribution of tree shapes are heavily skewed



(a) Original adjunction (b) Addition of foot node adjunction (c) Foot node with no root node adjunction

Fig. 6. Adjunction biases affecting the shape of generated trees. The histograms plot the frequency of the percentage of nodes used by the left side of the tree. The initialisation method described in Section 4.1 was used to generate 4000 trees for each approach, with a maximum depth of 20, a minimum/maximum size interval of 21/220.

towards trees with either very large left or right branches, with very few full trees. Ideally this distribution would be even across the entire spectrum of tree shapes favouring no particular shape. Fig. 6(b) shows that the distribution of tree shapes begins to level out once foot node adjunction is allowed.

In addition, allowing adjunction at the root nodes of auxiliary trees can have a similar but more pronounced effect on the form of the tree shape distribution. If at any point during derivation, an adjunction is performed on the top-most adjunctable address, usually the root of an auxiliary tree, the derived tree below this point of adjunction becomes the a sub-tree of the new tree's foot node, with the remainder of the new auxiliary tree off to one side. This causes the shape of the tree to be heavily skewed. By eliminating the adjunction at the root nodes of auxiliary trees, the tree shape distribution become much more level (see Fig. 6(c)). The probability of the tree reverting to a less skewed state depends on the ratio of adjunctable addresses available in the new auxiliary tree (usually quite small) to the adjunctable addresses in the displaced sub tree.

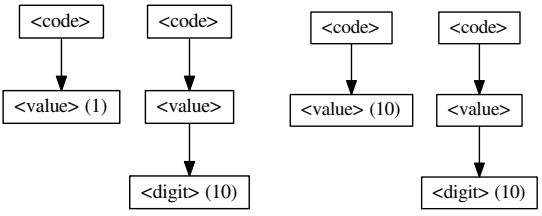
4.3 Grammar Transformation Bias

The probability of a specific terminal production being selected when generating a word using a CFG not only depends on the probability of that terminal production being selected within its own rule, but also on the probability of selecting each preceding production in order to reach the current rule from the start symbol. When transforming a CFG into a TAG these biases are lost and while it can be argued that this is a feature of the TAG representation, it can have unexpected effects for certain types of grammars.

For example, when generating a word from the balanced CFG presented in Fig. 7, whose derivation does not contain any recursive productions, there is a 0.5 chance of selecting `x` or `<digit>`. If `<digit>` is chosen there is a 0.1 chance of selecting any of the digits. From this it can be seen that even though there are 11 different words which could be generated, there is an equal probability of ending up with either an `x` or any one of the digits. When the CFG is transformed into a

```
<code>  := <value>
<value> := <value> <value> + |
           <value> <value> - |
           <value> <value> * |
           <value> <value> / |
           <digit> | <digit>
           x | x
<digit> := 0 | 1 | 2 | ... | 9
```

Fig. 7. A sample balanced grammar (equal probability of recursion and termination)



(a) TAGE stubs (b) PTAGE stubs

Fig. 8. TAGE tree stubs with a $0.\overline{09}$ chance of selecting `x`. Equivalent PTAGE stubs with a 0.5 chance.

TAG there are 11 trees to chose from, one with an `x` on the frontier and ten with a digit. Consequently, there is now a $0.\overline{09}$ chance of generating an `x`, as opposed to a 0.5 chance when using the CFG. This can make it difficult for certain words to be generated, both during initialisation and throughout a run.

In order to correct for the problems mentioned above, a novel method was designed which examines the probabilities contained within CFGs and applies them to the TAG. This method, named Probabilistic Tree-Adjoining Grammatical Evolution (PTAGE), is similar in theory to the structural and lexical biases imposed using TAGs by Hoai et al. [3] with TAG3P+. Whereas TAG3P+ uses properties of TAGs to impose language bias on the search, PTAGE's main function is to recreate the biases imposed by the CFGs used to generate each TAG. While this aids in generating similarly distributed initial populations, these biases affect the mapping process throughout the entire run.

In TAGE, the sets of initial and auxiliary trees are not generated at the beginning of the algorithm, but rather sets of elementary tree *stubs* are generated. This can greatly reduce the amount of memory needed to store the grammar. An elementary tree stub is an almost fully expanded elementary tree, with the terminal symbol leaf nodes excluded. In their place is a number representing the total number of different terminal nodes (variations) that can be attached at that point to complete the tree. For example, continuing with the sample grammar from Fig. 7 above, there would be a stub with the number `1` rather than an `x` and another stub with the number `10` rather than 10 different trees each with one of the digits, `0` through `9` (see Fig. 8(a)). The process of expanding a stub into a complete elementary tree is explained in the proceeding paragraph.

When choosing a tree in TAGE, the modulus operation is performed on the codon value and the total number of trees to select from, resulting in a number, c, between zero and the total number of trees minus one. If c has been used before, the correct tree is retrieved directly from a map. Alternatively, if c has not been seen before, each stub's variations are summed in order until the sum is greater than c in order to find the correct stub to expand. Then, proceeding in a depth first manner, the stub is completed by visiting each NT leaf node and dividing c by the product of the variations of all the NT leaves visited so far

while expanding that stub, performing the modulus operation on this product and the number of possible variations at the current NT node. This results in a number between zero and the total variations possible at that node, allowing the selection of the correct terminal production to expand the node by. This process continues until there are no more NT nodes to expand, storing the complete tree in a map for later use before being returned.

PTAGE examines the CFG and updates the number of variations at each stub's leaf nodes to reflect the probability of reaching those terminal symbols when expanding using the original CFG. In this example, since there should be an equal chance of generating an `x` as a digit, the variations on that stub are updated from `1` to `10` (as shown in Fig. 8(b)). The effect of this is that when selecting a tree there are now twenty trees to chose from, ten `x` trees and a single tree for each digit.

5 Experiments

The focus of this study is to improve the similarity of the initial setup of both GE and TAGE by examining the language biases which affect this, enabling a better comparison of performance and behaviour. As such, the experiments run in the initial study [12] are repeated twice here. First, using only the new method of initialisation and a second time incorporating the modified adjunction constraints (adjunction at foot nodes and no adjunction at root nodes).

Four benchmarks are used for this study, *Even Five Parity*, *Santa Fe Ant Trail*, *Symbolic Regression* and *Six Multiplexer*.The grammars used are identical to those used by Murphy et al. [12] apart from those of Symbolic Regression and Six Multiplexer, which were each given a new extra start symbol (see Fig. 9). This grammar change does not affect the behaviour of either algorithm but enables the disabling of root node adjunction. 100 independent runs were performed for each of GE, TAGE and PTAGE on both setups, the first using the new initialisation method, outlined in Sec. 4.1, with the original adjunction constraints (**NI**), and a second time using the new adjunction constraints (**NA**), outlined in Sec. 4.2. See Table 1 for the GE parameters.

6 Results and Discussion

6.1 Initialisation

It is clear from Table 2 that the improved initialisation of the population (using the original adjunction addresses) has a dramatic effect on the performance of GE. Improvements range from an increase of ∼10% (Even Parity) to almost an increase of an order of magnitude (Santa Fe) in the success rate. The new initialisation method did not have an effect of the same magnitude on TAGE, with improvements in success rate between ∼10% and ∼80%. As a result of this, GE's performance has surpassed that of TAGE on the Santa Fe Ant Trail problem, with TAGE showing superior performance on the remaining three problems.

Table 1. GE parameters adopted for each of the benchmark problems

Parameter	Value
Generations	200
Population Size	100
Initialisation	Ramped NT Size with Max Depth
Min NT Size	21
Max NT Size	70
Max Depth	10
Max Chromosome Wraps	0
Replacement Strategy	Generational
Elitism	10 Individuals
Selection Operation	Tournament
Tournament Size	3
One Point Crossover Prob	0.9
Integer Mutation Prob	0.02

```
<prog> ::= <expr>
<expr> ::= ( <op> <expr> <expr> ) | <var>
<op>   ::= + | - | *
<var>  ::= x0 | 1.0
```

(a) Symbolic Regression Grammar

```
<prog> ::=  <B>
<B>    ::= (<B>) && (<B>) | (<B>) "||" (<B>)
         | !(<B>) | (<B>) ? (<B>) : (<B>)
         | a0 | a1 | d0| d1 | d2 | d3
```

(b) Six Multiplexer Grammar

Fig. 9. Updated grammars for Symbolic Regression and Six Multiplexer problems

Table 2. The number of successful runs out of the 100 runs performed for all setups. **GE** and **TAGE** are the results from the original comparison (**NI** indicates the use of the new initialisation method and **NA** indicates the use of the new adjunction addresses).

	Even 5	**Santa Fe**	**Sym. Reg.**	**Six Multi.**
GE	79	3	44	6
TAGE	88	12	76	63
GE-NI	88	28	75	18
TAGE-NI	100	22	99	72
PTAGE-NI	100	13	99	20
TAGE-NI-NA	98	23	98	78
PTAGE-NI-NA	98	14	98	34

6.2 The Effect of PTAGE

From Table. 2 it can be seen that the application of PTAGE on two of the problems examined, Santa Fe and Six Multiplexer, has had a negative effect on performance. By examining the grammars of these two problems it can be noted that as a result of the transformation from CFG to TAG a bias is introduced, causing the selection of certain structures to be favoured over others. In the case of the Six Multiplexer problem, there is a probability of ~ 0.81 of selecting a tree containing `<B>?<B>:<B>` compared to a ~ 0.18 chance of selecting a tree containing either `<B> || <B>` or `<B> && <B>` or a < 0.01 chance of selecting a tree containing `!(<B>)`. This bias causes the TAGE tree to grow much wider than when using the PTAGE approach, as PTAGE balances these probabilities to be equal since they have an equal chance of being selected when deriving using CFGs. Similar effects are observed in the Santa Fe grammar. PTAGE has no effect on the other problems as the grammar transformation does not introduce any new biases.

6.3 New Adjunction Addresses

The inclusion of adjunction at root nodes and the exclusion of adjunction at the foot nodes appear to have only a marginal difference on performance, whereas they had a significant difference on the distribution of tree shapes as was seen in Fig. 6. This might suggest that TAGE benefits less from having a more diverse initial population than GE. This could be a result of the greater connectivity observed in TAGE landscapes in [11].

7 Conclusions

In the process of creating an initialisation method which generates similar sets of trees for both GE and TAGE, biases in the shape of the trees being generated by TAGE were detected. These biases were introduced due to the constraints placed upon the adjunction operation by TAGE as well as by the transformation algorithm used to generate TAGs from the original CFGs. New adjunction constraints and a system to eliminate the transformation biases, PTAGE, were described. It was noted that the transformation biases can be beneficial to the algorithm but are dependant on the grammar and the problem in question.

Subsequently, by improving the initialisation method used for the comparison of GE and TAGE, ensuring that similar distributions of trees sizes and shapes were created by each setup, it was seen that while there does not appear to be a statistically significant improvement in performance of TAGE over that of GE as was suggested in [12], TAGE still manages to generate more successful solutions in three of the four problems.

Interesting future work prompted as a result of this study includes examining the trends of the distributions of derivation tree shapes and sizes over the course of a run. Investigating these trends with both commonly distributed initial genotypic populations, as well as phenotypic populations, might give better insight into why a more diverse initial population appeared to be more beneficial to GE than TAGE.

Acknowledgements. This research is based upon work supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868.

References

[1] Daida, J.M., Ampy, D.S., Ratanasavetavadhana, M., Li, H., Chaudhri, O.A.: Challenges with verification, repeatability, and meaningful comparison in genetic programming: Gibson's magic. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 1851–1858. Morgan Kaufmann, Orlando (1999)

[2] Harper, R.: GE, explosive grammars and the lasting legacy of bad initialisation. In: IEEE Congress on Evolutionary Computation (CEC 2010). IEEE Press, Barcelona (2010)

[3] Nguyen, X.H., McKay, R.I., Abbass, H.A.: Tree adjoining grammars, language bias, and genetic programming. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 335–344. Springer, Heidelberg (2003)
[4] Nguyen, X.H., McKay, R., I(B.), E.D.L., Abbass, H.A.: Toward an Alternative Comparison between Different Genetic Programming Systems. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 67–77. Springer, Heidelberg (2004)
[5] Joshi, A.: Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions, ch. 6, pp. 205–250. Cambridge University Press, New York (1985)
[6] Joshi, A., Schabes, Y.: Tree-Adjoining Grammars. In: Handbook of Formal Languages, Beyond Words, vol. 3, pp. 69–123 (1997)
[7] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
[8] Kroch, A., Joshi, A.: The Linguistic Relevance of Tree Adjoining Grammar, Technical Report, University Of Pennsylvania (1985)
[9] Luke, S.: Two fast tree-creation algorithms for genetic programming. IEEE Transactions on Evolutionary Computation 4(3), 274–283 (2000)
[10] McKay, R., Hoai, N., Whigham, P., Shan, Y., O'Neill, M.: Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines 11, 365–396 (2010)
[11] Murphy, E., O'Neill, M., Brabazon, A.: Examining Mutation Landscapes In Grammar Based Genetic Programming. In: Silva, S., Foster, J.A., Nicolau, M., Machado, P., Giacobini, M. (eds.) EuroGP 2011. LNCS, vol. 6621, pp. 130–141. Springer, Heidelberg (2011)
[12] Murphy, E., O'Neill, M., Galvan-Lopez, E., Brabazon, A.: Tree-adjunct grammatical evolution. In: 2010 IEEE World Congress on Computational Intelligence, pp. 4449–4456. IEEE Computational Intelligence Society, IEEE Press, Barcelona, Spain (2010)
[13] O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. Genetic programming, vol. 4. Kluwer Academic Publishers (2003)
[14] Whigham, P.A.: Grammatical Bias for Evolutionary Learning. Ph.D. thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, Canberra, Australia (1996)