On the Non-uniform Redundancy in Grammatical Evolution

Ann Thorhauer(✉)

University of Mainz, Mainz, Germany
thorhauer@uni-mainz.de
http://www.uni-mainz.de

Abstract. This paper investigates the redundancy of representation in grammatical evolution (GE) for binary trees. We analyze the entire GE solution space by creating all binary genotypes of predefined length and map them to phenotype trees, which are then characterized by their size, depth and shape. We find that the GE representation is strongly non-uniformly redundant. There are huge differences in the number of genotypes that encode one particular phenotype. Thus, it is difficult for GE to solve problems where the optimal tree solutions are underrepresented. In general, the GE mapping process is biased towards short tree structures, which implies high GE performance if the optimal solution requires small programs.

Keywords: Grammatical evolution · Redundant representation · Binary trees · Bias

1 Introduction

One core component of any evolutionary algorithm (EA) is the representation used [17]. Indeed, the choice of representation determines the success of a heuristic search method [17]. In general, there are two types of representations: a direct and an indirect representation [16]. When using a direct representation, no distinction between geno- and phenotype is made, just like in standard genetic programming (GP) [9], which uses tree structures to represent the individuals. Here, the search operators (e.g., crossover and mutation) are applied directly to these tree structures. An indirect representation distinguishes between geno- and phenotype. In this case, a representation describes how the genotypes (e.g., binary strings) are mapped to the phenotypes (e.g., expressions, trees) [16]. When using this type of representation, the search operators are applied to genotypes, but the actual effect of these operators is observed at the corresponding phenotypes.

Indirect representations may be biased due to redundant encodings [18]. If this is true, on average more than one genotype represents the same phenotype. A representation is uniformly redundant if every phenotype is represented by the same number of genotypes; it is non-uniformly redundant if one or more phenotypes are represented by a larger number of genotypes than others.

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 292–302, 2016.
DOI: 10.1007/978-3-319-45823-6_27

Consequently, the use of redundant representations may be biased and could therefore influence the search process if the optimal solution or parts of it are underrepresented [18].

Grammatical evolution (GE) uses a redundant representation [15]. In contrast to standard GP, GE [19] uses variable-length binary strings to encode the programs/expressions and a grammar in Backus-Naur form (BNF) to map the binary genotypes to the tree phenotypes. Consequently, GE applies standard genetic search operators such as one-point crossover and mutation to linear bit strings.

The distribution of trees in the GP solution space is well studied for various problems [10] and can "[...] give an indication of problem difficulty for GP" [10]. For GE, there are no similar studies. In this paper, we study the non-uniform redundancy of the GE representation, especially the GE genotype to phenotype mapping. In our analysis, we focused on binary trees. We explored the entire solution space of GE by applying different grammars and different genotype lengths. We used two approaches to characterize trees: First, we used the tree size and tree depth; second, we took the shape of a tree into account since this property cannot be neglected when dealing with realistic programs. We showed that GE representation is strongly non-uniformly redundant. The number of different genotypes strongly exceeds the number of different phenotypes, and there are phenotypes that are encoded with higher probabilities than others. In general, short tree structures are represented most frequently.

In Sect. 2 we review bias and redundant representations. Section 3 reviews former studies of representation bias in GE. Our analysis and results are presented in Sect. 4. The paper ends with some concluding remarks.

2 Bias and Redundant Representations

A bias exists if some solutions or solution structures are visited more frequently than others during the run of a search procedure, or if certain actions are performed more frequently than others during the search [22]. The existence of a bias may be advantageous or disadvantageous for the search [17]. For example, a bias of search operators may be used to guide the search in a certain direction where promising solutions are presumed; this would be a desired bias. Unwanted bias occurs if there is an interaction between the search operators used and the chosen representation such that the problem becomes deceptive [1].

In heuristic search methods like GP, a desired bias results from the selection process. By selecting highly-fit individuals for the next generation, selection pushes a population in the direction of fitter individuals. In addition, by using problem-specific recombination or mutation operators, as well as suitable terminal and function sets, GP performance can be improved [21,22]. Indeed, Dignum and Poli showed that "[...] simple length biases can significantly improve the best fitness found during a GP run" [5].

Search bias can also be a result of redundant encodings [18]. Encodings are redundant if there are phenotypes that are encoded by more than one genotype.

A redundant encoding is biased (i.e., non-uniformly redundant) if not all phenotypes are encoded by the same number of genotypes; but rather some phenotypes are represented by a larger number of genotypes than others. When using GE, there is a redundant representation since there is more than one genotype that represents the same phenotype [15]. Non-uniform redundancy, where the phenotypes are non-uniformly represented by the genotypes, leads to a bias and causes structural difficulty since some solution structures are underrepresented. Rothlauf and Goldberg [18] examined the impact of redundant representations on the performance of evolutionary algorithms (EA). In general, redundant representations are less efficient because they use more alleles to store the same amount of information compared to non-redundant representations. In the case where a uniformly redundant representation is used, the general performance of the EA is neither decreased nor increased. In opposition to this, the performance of the algorithm can be increased or decreased if a non-uniformly redundant representation is used. In their experiments, Rothlauf and Goldberg [18] showed that performance can be increased when the optimal solution is overrepresented, and it can be decreased when the optimal solution is underrepresented.

3 Bias of Representation in Grammatical Evolution

GE [19] is a variant of GP that uses a complex genotype-phenotype mapping to create the phenotype programs/expressions from variable-length binary genotypes. A genotype consists of groups of eight bits (called codons) which encode an integer value that selects production rules from a grammar in BNF. These rules are used in the deterministic mapping process to create a phenotype.

Several studies have examined the bias of representation and grammar in GE. Most of them have focused on the impact of different representations or different search operators on performance rather than on the impact of the redundant representation. O'Neill and Ryan [12,14] examined the effect of genetic code degeneracy on genotypic diversity and the performance of GE. When a degenerate genetic code is used, each codon consists of more bits than actually necessary to encode a sufficient amount of integer values to select the rules from the grammar. They found that genetic diversity is higher when a degenerate genetic code is used. In addition, the amount of invalid individuals is lower. The impact on performance depends on the grammar used. Montes de Oca [11] focused on creating numerical values by concatenating digits and found that the most-commonly-used GE grammar induces a bias towards short-length numbers. Hemberg et al. [8] considered three different GE grammars (postfix, prefix, infix) and their influence on performance for various symbolic regression problems. They observed no differences between the grammars for small problem instances. However, for large problems, a postfix grammar was found to be advantageous. Fagan et al. [6] compared the performance of GE when using four different genotype-phenotype mappings (depth-first, breadth-first, random and πGE [13]) for four benchmark problems and measured the average best fitness, the average size of genotypes, and the average number of derivation tree nodes over the number of generations. The πGE mapper outperformed the other mapping strategies in three out

of four problems. The breadth-first mapper produced larger trees in three out of four problems compared to the other mapping strategies. Harper [7] showed that standard GE random bit initialization produces trees that are non-uniformly distributed since "80 % of the trees have 90 % of their nodes on one side of the tree or the other" [7]. Trees are biased to be "tall and skinny" [7]. He distinguished between two types of grammars: explosive and balanced. He defined a grammar to "be explosive if the number of non-terminals (functions) exceeds the number of terminals" [7]. A grammar is balanced if the probability for expanding a non-terminal into a multiple of the same non-terminal is equal to the probability of expanding into a terminal. Therefore, when using an explosive grammar, the probability is high that the mapping process will run out of genes [7]. Daida and co-workers [2–4] studied the influence of the standard GP tree representation on the search process, and showed that not only crossover and selection does influence GP search behavior, but so does the representation. Inspired by their work, Thorhauer and Rothlauf [20] found that a random walk with GE using standard operators (one-point crossover, mutation and duplication) has a strong bias towards sparse tree structures.

Overall, GE literature is dominated by experimental studies on the impact of different search operators or representations on performance, but a fundamental mathematical analysis of the entire solution space has been omitted.

4 Analysis and Results

We studied the bias of representation in GE for binary trees. Using the definition from Harper [7], we used a balanced grammar (Fig. 1(a)) and an explosive grammar (Fig. 1(b)) to map the genotypes to the phenotype trees. We set the number of codons to 10 and 20, and created all possible binary genotypes by using 10 and 20 codons respectively, and studied their phenotypic properties. For the balanced grammar A (Fig. 1(a)), only the least significant bit in each codon (usually eight bits) determines which rule to choose during the mapping process. Therefore, we reduced the total number of possible binary genotypes from 2^{80} to $2^{10} = 1024$ when using 10 codons (each consists of one bit), and from 2^{160} to 2^{20} when using 20 codons. For the explosive grammar B (Fig. 1(b)), we needed the last two bits of each codon to define the rule to choose since we wanted to ensure that all rules were chosen with equal probability. Again, we got 2^{20} different genotypes. So the value of any bit directly affects the phenotype. We used standard depth-first mapping to create the phenotypes. No wrapping operator was used since the mapping process would never terminate and thus the corresponding individuals would be invalid anyway. This is a result of the structure of the chosen grammars. Before we analyzed the characteristics of all phenotype trees, the derivation trees were transformed into syntax trees to get binary trees that only consisted of functions (internal nodes) and terminals (leaf nodes). For all the tree structures, we measured the following properties: tree depth, tree size (internal nodes plus leaf nodes) and tree shape.

We started our analysis by distinguishing individuals according to their phenotype tree properties: size and depth. Figures 2(a)–(c) show the phenotype

```
<start> ::= <expr>
<expr>  ::= (<expr> + <expr>)
          | <var>
<var>   ::= X
```

(a) grammar A

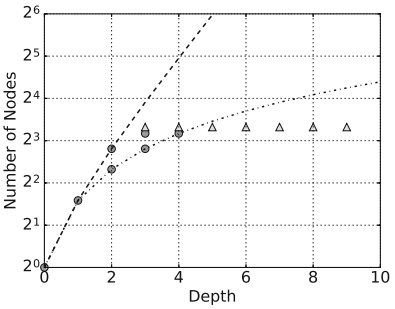
```
<start> ::= <expr>
<expr>  ::= (<expr> + <expr>)
          | (<expr> + <expr>)
          | (<expr> + <expr>)
          | <var>
<var>   ::= X
```

(b) grammar B

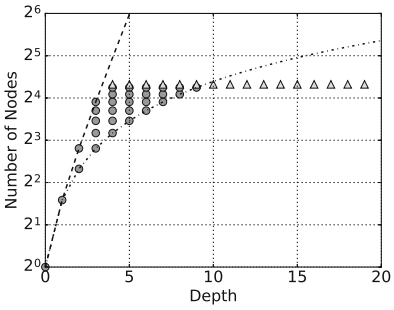
Fig. 1. Production rules in BNF. Balanced (left) and explosive variant (right).

(tree) solution spaces that were created with 10 codons using grammar A, 20 codons using grammar A, and 10 codons using grammar B. We plotted the number of nodes (tree size) over the tree depth for all phenotype trees. The outer dashed lines show the boundaries for valid binary trees with a minimum and a maximum number of nodes. The circles between these lines represent valid trees, whereas the triangles represent invalid trees. In these cases, the mapping process could not be finished because the genotype ran out of genes. As a result, there are invalid binary trees, where not all internal nodes (+) have two child nodes or not all external nodes are terminals (X). When 10 codons and grammar A were used, we observe 7 different valid trees and 7 different invalid trees that can be created with 2^{10} different genotypes (Fig. 2(a)). Under these conditions, valid trees with a maximum size of 9 nodes can be created. Consequently, when using 20 codons and the same grammar, the phenotype solution space increases, and the limit of the maximum tree size rises to 19 nodes (Fig. 2(b)). Thus 2^{20} different genotypes encode 30 different valid and 16 invalid trees. Figure 2(c) shows the same result as seen in Fig. 2(a). Here, we used 10 codons and grammar B. In summary, the solution space of binary trees that can be covered with GE depends on the length of the genotype. The change from the balanced grammar A to the explosive grammar B does not modify the solution space of possible binary trees in GE.

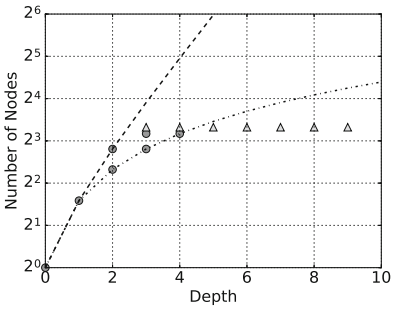
To study the redundancy in GE representation, we had to examine the frequency of all trees. Figures 3(a)–(c) present the proportion of trees of a given size and depth in a 3D view. This clearly shows that the GE mapping process creates specific trees with higher probabilities than it does others. Therefore, representation in GE is non-uniformly redundant. Indeed, trees with a size of one and a depth of zero were created most frequently, independent from codon lenght or the grammar used. All trees of size 10 (Figs. 3(a) and (c)) and 20 (Fig. 3(b)) are the result of an unfinished mapping. As Harper [7] described, the use of the explosive grammar B strongly overrepresents these invalid individuals (Fig. 3(c)) compared to the use of the balanced grammar A (Figs. 3(a) and (b)). In summary, we observe an overrepresentation of short valid trees in all three figures. The GE mapping process has a strong bias towards short trees when grammar A is used (Fig. 3(a) and (b)); whereas when grammar B is used, the mapping process more frequently creates longer, but invalid, trees (Fig. 3(c)).



(a) 10 codons grammar A

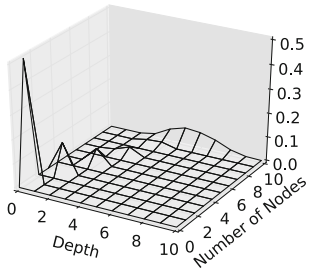


(b) 20 codons grammar A

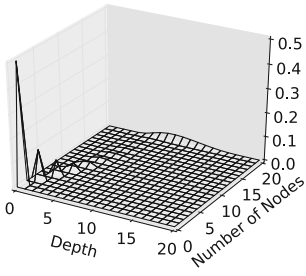


(c) 10 codons grammar B

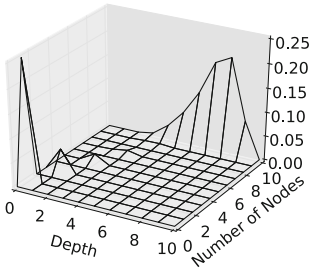
Fig. 2. Phenotype (binary tree) solution spaces for different codon lengths and grammars.



(a) 10 codons grammar A



(b) 20 codons grammar A



(c) 10 codons grammar B

Fig. 3. Proportion of different binary trees for different codon lengths and grammars.

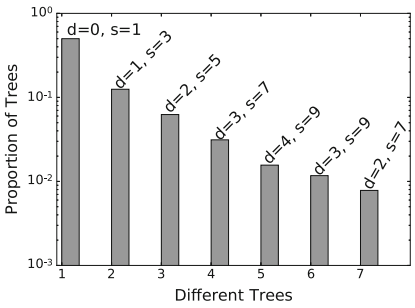
Figures 4(a)–(c) represent the proportion of trees that have the same size and depth (the proportion of invalid trees are not shown). Table 1 (rows A and B) describes how the probabilities to create particular trees can be calculated. The probabilities to represent specific trees in Fig. 4(a) when using 10 codons and grammar A cover a range between 0.5 and 0.0078125 ($\frac{1}{2^7}$). This implies 50 % of

genotypes represent a tree that consists of one terminal X, whereas only 0.78125 % of genotypes represent a full tree of depth two (Fig. 4(a) d = 2, s = 7). The probability that a sparse tree with the same size of seven and a depth of three (Fig. 4(a) d = 3, s = 7), which has only one expanded node at any depth level, is represented by a binary genotype is four times higher (3.125 %). For 20 codons and grammar A, the probabilties actually range between 0.5 and approximately 0.000031. The use of grammar B and 10 codons hugely changes the probabilities to create each tree since it is three times more likely to choose the rule $< expr > + < expr >$ than to choose $< var >$ (see Table 1 column B). In this case, the probabilities range between 0.25 and about 0.00165. In summary, these results reveal a strong overrepresentation of short and rather sparse trees.

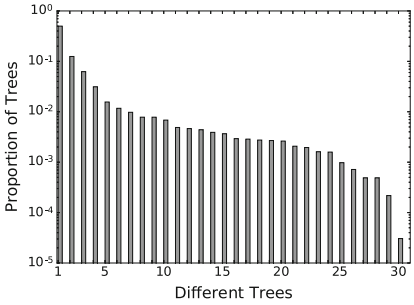
Table 1. The probabilities of representing particular trees for both grammars. Row A and B: Trees are characterized by their size and depth. Row A: The probabilities to create a full tree. Row B: The probabilities to create a sparse tree (i.e., same number of nodes as a full tree but only one expanded node at any depth level). Row C: Trees are characterized by their size, depth and shape; the probabilities to create any tree of a given size, depth and shape.

	grammar A	grammar B
A	$(\frac{1}{2})^{size}$	$(\frac{3}{4})^{(number\ internal\ nodes)} \times (\frac{1}{4})^{(number\ leaf\ nodes)}$
B	$(\frac{1}{2})^{size} \times 2^{(d-1)}$	$(\frac{3}{4})^{(number\ internal\ nodes)} \times (\frac{1}{4})^{(number\ leaf\ nodes)} * 2^{(d-1)}$
C	$(\frac{1}{2})^{size}$	$(\frac{3}{4})^{(number\ internal\ nodes)} \times (\frac{1}{4})^{(number\ leaf\ nodes)}$

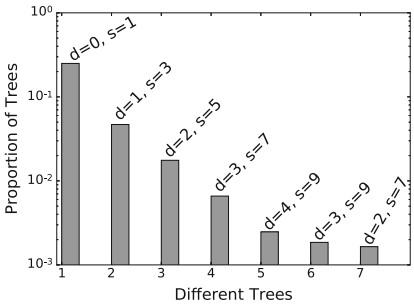
To study the distribution of different trees in greater detail, we also took the shape of the trees into account. This implies that the exact ordering of nodes is relevant (i.e., important in realistic programs). Consequently, the number of different valid phenotype trees that can be encoded by 2^{10} different binary genotypes increases from 7 (Figs. 4(a) and (c)) to 23 (Figs. 5(a) and (c)). With 2^{20} genotypes, the number of trees increases from 30 (Fig. 4(b)) to 6918 (Fig. 5(b)). As an example, let's take the case of two trees of different shapes that have a size of five and a depth of two (Fig. 5(a) d = 2, s = 5). Table 1, row C shows how to calculate the probability that a specific tree will be created. The probabilities to encode the different phenotypes range between 0.5 and approximately 0.00195 ($\frac{1}{2^9}$) when grammar A and binary genotypes of 10 codons are used, and between 0.5 and about 0.000002 when grammar A and 20 codons are used. Again, the use of grammar B and 10 codons hugely changes the probabilty that a genotype creates a particular tree. In this case, the distribution of probabilites ranges between 0.25 and about 0.00031. In summary, the larger the size of a particular tree, the lower the probability that this tree is represented by a binary genotype (independently from the grammar used). The probability of creating a specific binary tree depends only on its size. The additional consideration of the tree shape increases the number of different trees that can be encoded by 2^{10} and 2^{20} different genotypes, but does not prevent the overrepresentation of particular binary trees.



(a) 10 codons grammar A

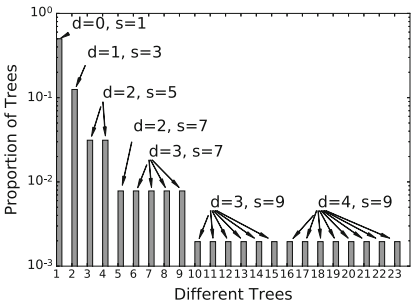


(b) 20 codons grammar A

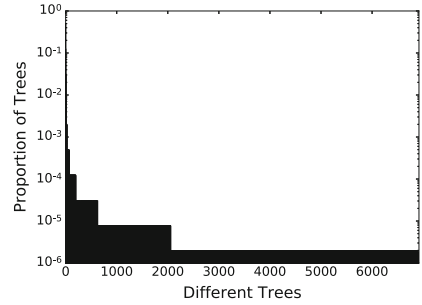


(c) 10 codons grammar B

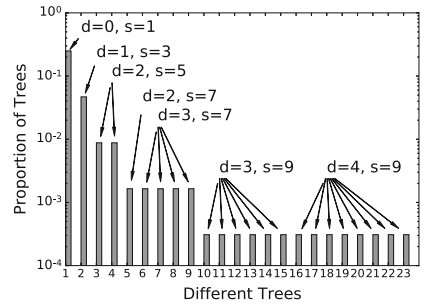
Fig. 4. Number of instances per valid tree (normalized over all tree instances) for different codon lengths and grammars. Trees are characterized by their size and depth.



(a) 10 codons grammar A



(b) 20 codons grammar A



(c) 10 codons grammar B

Fig. 5. Number of instances per valid tree (normalized over all tree instances) for different codon lengths and grammars. Trees are characterized by their size, depth and shape.

Table 2 presents the number $|\Phi_p|$ of different valid phenotypes and the number $|\Phi_{p_{invalid}}|$ of different invalid phenotypes that can be encoded by genotypes of either 10 or 20 codons using grammars A and B. These results emphasize that

Table 2. Number $|\Phi_g|$ of different genotypes (genotype search space Φ_g), number $|\Phi_p|$ of different possible valid phenotype trees (phenotype solution space Φ_p), number $|\Phi_{p_{invalid}}|$ of different possible invalid phenotype trees ($\Phi_{p_{invalid}}$). Trees are characterized by their size, depth and shape.

	$\lvert\Phi_g\rvert$	$\lvert\Phi_p\rvert$	$\frac{\lvert\Phi_p\rvert}{\lvert\Phi_g\rvert}$	$\lvert\Phi_{p_{invalid}}\rvert$	$\frac{\lvert\Phi_{p_{invalid}}\rvert}{\lvert\Phi_g\rvert}$
grammar A, 10 codons	2^{10}	23	0.0225	252	0.2461
grammar A, 20 codons	2^{20}	6 918	0.0066	184 756	0.1762
grammar B, 10 codons	2^{20}	23	0.00002	252	0.00024

GE representation is strongly redundant since the phenotype solution space (Φ_p) is obviously smaller than the genotype search space (Φ_g). The number of different valid phenotype trees over the number of different genotypes for grammar A and 10 codons is 0.0225, whereas the proportion of different invalid phenotype trees is obviously larger (0.2461). Both values are even lower when grammar B is used. In general, $|\Phi_{p_{invalid}}|$ exceeds $|\Phi_p|$ for both grammars and codon lengths. During a GE run, these invalid trees are penalized with a mimimum fitness value, and provide no additional benefit since they will be sorted out.

5 Conclusions

We studied the redundancy in GE representation for binary trees. We used two different grammars (balanced and explosive) and two different genotype lengths. We explored the entire solution space of GE by creating all possible binary genotypes and mapped them to phenotype trees. When trees are characterized by their size and depth, sparse trees are more likely to be represented than full trees of the same size. If in addition the shape of a tree is relevant, the probability of creating a particular binary tree depends only on its size. In this case, the number of different invalid trees is larger than that of valid trees. Independent from the grammars used or codon lengths, the number of different binary genotypes strongly exceeds the number of different binary phenotypes. Moreover, there are large differences in the number of genotypes that encode one particular phenotype tree. Thus, it is difficult for GE to solve problems if the optimal tree solutions are underrepresented. In general, the GE mapping process is biased towards short tree structures.

A higher genotype length increases the number of different phenotypes that can be encoded. Moving from a balanced to an explosive grammar alters the probabilities for creating any tree. Furthermore, the probability that a genotype encodes an invalid tree is higher.

The focus of our study was on binary trees. Using more complex grammars and non-binary trees would create a greater variety of different trees and therefore reduce the probability of creating trees with the same characteristics. In the future, we will extend this analysis to different grammars that allow the creation of more complex non-binary trees.

References

1. Caruana, R.A., Schaffer, J.D.: Representation and hidden bias: Gray vs. Binary coding for genetic algorithms. In: Proceedings of the Fifth International Conference on Machine Learning, pp. 153–161. Morgan Kaufmann (1988)
2. Daida, J.M.: Limits to expression in genetic programming: lattice-aggregate modeling. In: CEC 2002, pp. 273–278. IEEE Press, NJ, USA (2002)
3. Daida, J.M., Hilss, A.M.: Identifying structural mechanisms in standard genetic programming. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1639–1651. Springer, Heidelberg (2003)
4. Daida, J.M., Li, H., Tang, R., Hilss, A.M.: What makes a problem GP-Hard? validating a hypothesis of structural causes. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1665–1677. Springer, Heidelberg (2003)
5. Dignum, S., Poli, R.: Operator equalisation and bloat free GP. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 110–121. Springer, Heidelberg (2008)
6. Fagan, D., O'Neill, M., Galván-López, E., Brabazon, A., McGarraghy, S.: An analysis of genotype-phenotype maps in grammatical evolution. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 62–73. Springer, Heidelberg (2010)
7. Harper, R.: GE, explosive grammars and the lasting legacy of bad initialisation. In: CEC 2010, pp. 1–8. IEEE Press (2010)
8. Hemberg, E., McPhee, N., O'Neill, M., Brabazon, A.: Pre-, In- and postfix grammars for symbolic regression in grammatical evolution. In: IEEE Workshop and Summer School on Evolutionary Computing, pp. 18–22 (2008)
9. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
10. Langdon, W., Poli, R.: Foundations of Genetic Programming. Springer, Berlin (2002)
11. Montes de Oca, M.A.: Exposing a bias toward short-length numbers in grammatical evolution. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 278–288. Springer, Heidelberg (2008)
12. O'Neill, M., Ryan, C.: Genetic code degeneracy: implications for grammatical evolution and beyond. In: Floreano, D., Nicoud, J.D., Mondada, F. (eds.) Advances in Artificial Life. LNCS, vol. 1674, pp. 149–153. Springer, Berlin (1999)
13. O'Neill, M., Brabazon, A., Nicolau, M., Garraghy, S.M., Keenan, P.: π grammatical evolution. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 617–629. Springer, Heidelberg (2004)
14. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers, Norwell (2003)
15. O'Neill, M., Ryan, C., Keijzer, M., Cattolico, M.: Crossover in grammatical evolution. Genet. Program. Evolvable Mach. **4**(1), 67–93 (2003)
16. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Springer, Berlin (2006)
17. Rothlauf, F.: Design of Modern Heuristics. Springer, Heidelberg (2011)
18. Rothlauf, F., Goldberg, D.E.: Redundant representations in evolutionary computation. Evol. Comput. **11**(4), 381–415 (2003)

19. Ryan, C., Collins, J.J., O'Neill, M.: Grammatical evolution: evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) EuroGP 1998. LNCS, vol. 1391, pp. 83–95. Springer, Heidelberg (1998)
20. Thorhauer, A., Rothlauf, F.: Structural difficulty in grammatical evolution versus genetic programming. In: GECCO 2013, pp. 997–1004. ACM (2013)
21. Whigham, P.A.: Grammatically-based genetic programming. In: Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, pp. 33–41. Morgan Kaufmann, San Mateo (1995)
22. Whigham, P.A.: Search bias, language bias and genetic programming. In: Proceedings of the First Annual Conference on Genetic Programming 1996, pp. 230–237. MIT Press, CA, USA (1996)