Investigation of the Performance of Different Mapping Orders for GE on the Max Problem

David Fagan, Miguel Nicolau, Erik Hemberg,
Michael O'Neill, Anthony Brabazon

Natural Computing Research & Applications Group
Complex & Adaptive Systems Lab
School of Computer Science & Informatics
University College Dublin
Ireland

david.fagan@ucd.ie, miguel.nicolau@ucd.ie, erik.hemberg@ucd.ie,
m.oneill@ucd.ie, anthony.brabazon@ucd.ie

February 2, 2012

Abstract

We present an analysis of how the genotype-phenotype map in Grammatical Evolution (GE) can effect performance on the Max Problem. Earlier studies have demonstrated a performance decrease for Position Independent Grammatical Evolution (πGE) in this problem domain. In πGE the genotype-phenotype map is changed so that the evolutionary algorithm controls not only what the next expansion will be but also the choice of what position in the derivation tree is expanded next. In this study we extend previous work and investigate whether the ability to change the order of expansion is responsible for the performance decrease or if the problem is simply that a certain order of expansion in the genotype-phenotype map is responsible. We conclude that the reduction of performance in the Max problem domain by πGE is rooted in the way the genotype-phenotype map and the genetic operators used with this mapping interact.

1 Introduction

The use of a genotype-phenotype map in Genetic Programming (GP) [13, 22] has been used by many within the field [1, 5, 7, 10, 11, 12, 15, 23] and a number of variants to the standard tree-based form of GP exist, amongst which some of the most popular are Linear GP [2], Cartesian GP [16] and Grammatical Evolution (GE) [3, 20]. GE is a grammar-based form of GP which takes inspiration from DNA Transcription in adopting a mapping from a linear genotype to phenotypic GP trees. The reason for adopting a genotype-phenotype map for GP where presented by O'Neill [17] as a series of arguments while also noting it can provide a number of advantages. These include a generalised encoding that can represent a variety of structures allowing GP to generate structures in an arbitrary language, efficiency gains for evolutionary search (e.g. through neutral evolution), maintenance of genetic diversity through many-to-one maps, preservation of functionality while allowing continuation of search at a genotypic level, reuse of genetic material potentially allowing information compression, and positional independence of gene functionality.

Previous work [4] showed a significant reduction in the performance of GE on the Max problem once the πGE genotype-phenotype map was applied. This was a very surprising result as previous studies by O'Neill et al. [18] had not shown a similar type of performance reduction in any of the benchmark problems in that study. This poor outcome on the Max problem sparked a lot of questions as to what was causing this result, which went against previous findings that πGE would match if not out perform standard GE on the majority of problem domains tested. The result of this examination is hoped to lead to a deeper understanding of what is going on during a πGE run and try to use this knowledge to further guide the development of different genotype-phenotype maps in GE, leading to the ultimate goal of designing the most effective, efficient version of GE.

The remainder of the paper is structured as follows. A brief overview of the essentials of GE are provided in Section 2 before an explanation of the different genotype-phenotype maps used in the study in Section 3. The next part of the paper describes the experimental setup (Section 4), the results found (Section 5) and a discussion (Section 6) before drawing conclusions and pointing to future work.

2 Grammatical Evolution Essentials

GE marries principles from molecular biology to the representational power of formal grammars. GE's rich modularity gives it a unique flexibility, making it possible to use alternative search strategies, whether evolutionary, or some other heuristic (be it stochastic or deterministic) and to radically change its behaviour by merely changing the grammar supplied. As a grammar is used to describe the structures that are generated by GE, it is trivial to modify the output structures by simply editing this grammar. The explicit grammar allows GE to easily gen-

erate solutions in any language (or a useful subset of a language). For example, GE has been used to generate solutions in multiple languages including Lisp, Scheme, C/C++, Java, Prolog, Postscript, and English. The ease with which a user can manipulate the output structures by simply writing or modifying a grammar in a text file provides an attractive flexibility and ease of application not as readily enjoyed with the standard approach to GP. The grammar also implicitly provides a mechanism by which type information can be encoded thus overcoming the property of closure, which limits the traditional representation adopted by GP to a single type. The genotype-phenotype mapping also means that instead of operating exclusively on solution trees, as in standard GP, GE allows search operators to be performed on the genotype (e.g., integer or binary chromosomes), in addition to partially derived phenotypes, and the fully formed phenotypic derivation trees themselves. As such, standard GP tree-based operators of subtree-crossover and subtree-mutation can be easily adopted with GE. By adopting the GE approach one can therefore have the expressive power and convenience of grammars, while operating search in a standard GP or Strongly-Typed GP manner. For the latest description of GE please refer to Dempsey et al. [3].

Previous work by Hemberg et al. [8] where three grammar variants were examined in the context of Symbolic Regression is also directly relevant to this study. The language represented by each of the grammars were all semantically equivalent in terms of the phenotypic behaviour of the solutions that could be generated. The only difference was syntactical. That is, postfix, prefix and infix notations were adopted for the same set of terminal symbols of the language. Performance advantages were observed on the problems examined for the postfix notation over both alternatives. If one examines the behaviour of postfix notation it amounts to a postorder expansion of the tree. In terms of a generative grammar this means that the contents of subtrees are determined before the operator at the root of the subtree.

3 Genotype-Phenotype Maps - GE, πGE

In order for a full examination of the performance on the Max Problem to be carried out four separate mappings for GE were required, details of which are explained in this section. In GE we begin the mapping process by finding the start symbol in the grammar. This non terminal (NT) in the case of the example grammar shown in Fig. 1, `<e>` is then evaluated using Eq. 1. By taking the first codon value of the GE chromosome (12) and the number of expansions possible for the state `<e>` (2), we get the first expansion of the tree, where `<e>` expands to `<e><o><e>` (12%2) . From this point on the leftmost NT is always expanded first in the derivation process. This action will continue to be performed until no NTs remain to be expanded in the derivation tree. An example of this mapping is shown in Fig. 2 based on the example grammar shown in Fig. 1 where the order of expansion is indicated by a set of numbers on the arrows between the blocks on the diagram, in the form of 1(12%2) where 1 is the expansion order

and 12%2 is the application of Eq. 1.

$$New\ Node\ =\ Codon\ value\ \%\ Number\ of\ rules\ for\ NT \tag{1}$$

```
<e> ::= <e> <o> <e> | <v>

<o> ::= + | *

<v> ::= 0.5 | 5

Chromosome ::= 12,8,3,11,7,6,11,8,4,3,
               3,11,15,7,9,8,10,3,7,4
```

Figure 1: Example Grammar and Chromosome.

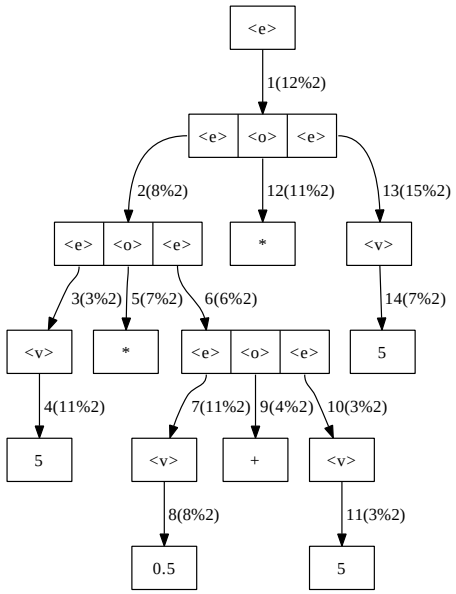

Figure 2: Standard GE Genotype to Phenotype Mapping.

The only difference between standard GE and πGE in its purest form is in the mapping process from genotype to phenotype. πGE's mapping process differs from that of GE in that each expansion of a NT requires two codons. The standard GE chromosome is essentially split into pairs of values where the first codon of the pair is used to choose which NT to expand and the second is used to choose what to expand the NT to, based on the rules available for a NT of that type. The chromosome shown in Fig. 1 can be viewed as a list of paired values such as `((12,8),(3,11).....)`, where the first value of the pair (The Order Codon) is used to determine the next NT to expand by using Eq. 2 and this will return which NT to choose from a list of unexpanded NTs. Once the NT to be expanded has been chosen, the second codon (Content Codon) is used in conjunction with Eq. 1 (the standard GE expansion rule) to determine what the NT expands to; and if this node happens to be an NT, it is added to the list of unexpanded NTs. Figs. 3 and 4 show the expansion of the example grammar in Fig. 1 using the πGE mapping process. The number associated with each branch of the tree is a reference to the numbered steps shown in Fig. 3 which show how each choice of NT to expand comes about. It is interesting to note the different shape and size of the examples based on just a change in mapping.

```
1.  [(e)]         <-  (12%1=0)
2.  [(e),o,e]     <-  (3%3=0)
3.  [o,(e),v]     <-  (7%3=1)
4.  [o,(v),e,o,e]<-  (11%5=1)
5.  [(o),e,o,e]   <-  (4%4=0)
6.  [(e),o,e]     <-  (3%3=0)
7.  [(o),e,v]     <-  (15%3=0)
8.  [e,(v)]       <-  (9%2=1)
9.  [(e)]         <-  (10%1=0)
10. [(v)]         <-  (7%1=0)
```

Figure 3: NT selection process in πGE.

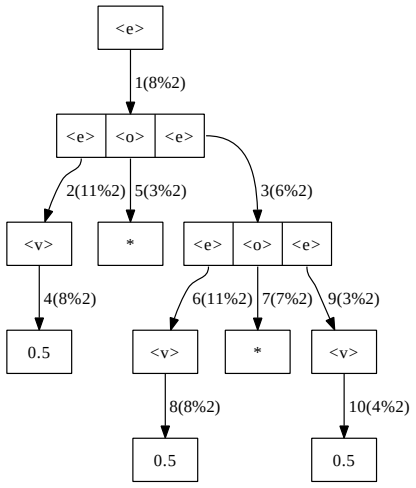

Figure 4: Standard πGE Genotype to Phenotype Mapping.

$$NT\ to\ expand\ =\ Codon\ value\ \%\ Number\ of\ NT's \tag{2}$$

In addition to the πGE mapper above a slightly tweaked version of this mapping was also required. The experiments that were planned required a version of πGE in which the order codons of the mapping were fixed across the whole population and also to be not affected by crossover or mutation during evolution. This requires the addition of an order chromosome to standard GE and then an edit to the πGE mapper so it will work with the new setup. This setup is referred to as Fixed-Order mapping and is necessary to see if the constantly changing order of expansion in the πGE mapping process is the cause of the performance decrease. The mapping also serves as an interesting experiment by itself as it will show if any randomised order might do as well as πGE . This way of mapping can be seen as fixed order πGE mapping or as a standard GE mapping but with a randomised order of mapping.

Finally Right-Most mapping is needed, this mapping is just a variant of the standard GE mapping process. Rather than taking the left-most non terminal and expanding as in standard GE, in Right-Most we simple always select the right-most non terminal for expansion. This way of mapping is required for this study as it should provide a similar performance to the standard GE mapping, whilst also allowing further insight into what is happening within the evolution of a solution to the Max problem in GE. An overview of the different Mappings can be seen in Table 1.

Table 1: Overview of the different Mappings

	GE	**Right-Most**	**πGE**	**Fixed-Order**
Expansion Order	Left-Most NT	Right-Most NT	Equation 2	Equation 2
Expansion Choice	Equation 1	Equation 1	Equation 1	Equation 1
Fixed Order Choice	Yes	Yes	No	Yes

4 Experimental Setup

We wish to test the hypothesis that the evolvability of the order codons in πGE is responsible for the performance decrease observed, as evolution has to try and cope with a changing tree structure across each member of the population. By removing this degree of freedom from the evolutionary process it is hoped that the performance decrease will be accounted for. We will measure performance in terms of examining the average best fitness.

We adopted GEVA v1.1 [19] for the experiments conducted in this study. The evolutionary parameters adopted on all problems are presented in Table 2. Note that we deliberately use a relatively small population size of 200 compared to the standard 500 that would typically be adopted for these problem instances. This was to make it harder for the mappers to find a perfect solution, and therefore allow us to discriminate more clearly performance differences on the various setups. For each setup we performed 100 runs using the same range of random seeds, in the case of the Fixed-Order setup we examined 100 different random expansion orders and for each order we then did 100 runs.

Table 2: Parameter settings adopted on all problems examined.

Parameter	**Value**
Generations	100
Population size	200
Replacement strategy	Generational with elitism (10%)
Selection	Tournament size=5
Mutation probability	0.01 (integer mutation)
Crossover probability	0.9 (variable single point)
Initial chromosome length	200 codons (random init)
Initial πGE chromosome length	400 codons (random init)

4.1 The Max Problem

The problem used in this paper is the Max problem first implemented for GE in [9] which is based upon the problem as described in [14] and [6]. The grammar adopted used for the problem appears in Fig. 5.

The aim of the problem is to evolve a tree that returns the largest value within a set depth limit (8 in this study). A minimal function set of addition and multiplication is provided alongside a single constant (0.5). The optimal

```
<prog> ::= <expr>

<expr> ::= <op> <expr> <expr>
         | <var>

<op> ::= +
       | *

<var> ::= 0.5
```

Figure 5: The grammar adopted for the Max problem instance.

solution to this problem will have addition operators towards the leaves of the tree to create as large a variable as possible greater than 1.0 in order to exploit multiplication operators towards the root of the tree. This problem is considered difficult for GP as solutions tend to converge on suboptimal solutions which can be difficult to escape from [14].

4.2 Mappers

For the purposes of this study we examined four different mapping strategies. The standard mapper adopted in GE we refer to as **Depth-first**. The name reflects the path this mapper takes through the non-terminal symbols in the derivation tree always selecting the left-most NT. The next mapping to consider was the opposite of the standard GE mapping, in which a depth first expansion of the derivation tree is done, but working on the right-most non terminal nodes first; we refer to this as **Right-most**. The π**GE** mapper as first described by O'Neill et al. [18] is the third mapper used. πGE lets the evolving genome decide which non-terminal to expand at each step in the derivation sequence. Finally a mapping was required that fixed the order codons of πGE to a random order known as **Fixed-Order**. This mapping was used to fix the πGE order codons for the whole population; unlike πGE we will have the same random order of expansion for the whole population. This mapping strategy will examine the influence of different fixed mapping orders on the Max problem, instead of only Depth-first and Right-most. Fixed-Order will also be used to examine if the evolvability of the order in πGE , or just the fact that it is a different order gives the performance decrease observed in the Max problem.

5 Results

The first set of experiments tested the hypothesis that the evolvability of the order codons in πGE were responsible for the observed performance decrease. For this the mapping order of πGE was fixed for the whole population and was not subject to crossover or mutation. One hundred different orders were generated and each order was then run one hundred times. The five best and five worst performing orders where then extracted and compared to a run of standard πGE, the results of which are shown in Table 3 in which the results

shown are minimised so the optimal solution is zero. It is worth noting that all of the runs are within a standard deviation and thus show that the evolvable order is not where the performance decrease is caused. Further examination of the results show a slight performance difference between πGE and the fixed order mappers, which shows that the extra search performed by πGE for good representations is not negatively affecting performance.

Table 3: Average Best Fitness Values after 100 generations over 100 independent runs. For the five best and worst performing mapping orders

Mapping	**Avg.Best (std. dev.)**
πGE	29.78 (24.03)
Best 1	32.59 (24.67)
Best 2	33.21 (23.92)
Best 3	33.42 (28.97)
Best 4	35.26 (29.09)
Best 5	35.56 (25.84)
Worst 1	52.20 (39.93)
Worst 2	49.82 (35.01)
Worst 3	49.59 (40.03)
Worst 4	49.35 (34.08)
Worst 5	49.15 (36.45)

Table 4: Average Best Fitness Values after 100 generations over 100 independent runs.

Mapping	**Avg.Best (std. dev.)**
GE	2.50 (5.42)
Right-Most	9.14 (15.54)
πGE	29.78 (24.03)

After examination of these results it was hypothesised that perhaps the answer lay in how operations such as crossover and mutation would effect performance. Previous work on what is known as the ripple effect of crossover [21] and the ripple effect of mutation [9] have shown how the GE derivation is trimmed by these operators. In these papers it was shown how a single change in the genotype can lead to a change in the expansion of every codon following on from that position in the genotype. This led to the introduction of the right-most mapping which is essentially a transpose of the traditional GE mapper. The results for this new comparison of mappings is shown below in Table 4 These results will be explained more fully in the discussion section, Section 6.

6 Discussion

The main focus of this study started out as an investigation to see if the evolvability of the order codons in πGE is responsible for the performance decrease observed in the Max problem. By examining the results in Table. 3 it can be seen that the evolvability of πGE's mapping order is not responsible for the performance decrease. This is clearly visible from comparing the fixed orders and πGE , that the fixed orders more closely resemble the results achieved by πGE and as the only difference between the two is that πGE 's order is subject to change that this change cannot be blamed for the observed performance decrease.

The real key to the performance by examining the results in Table 4 is in the operators and how they interact with the mapper, by interact with the mapper we mean how changes in the genotype (upon which all GE operations such as crossover and mutation act) lead to the mapping of a different phenotype. The Right-Most mapping was chosen so that we could examine if the performance on the max problem was down to burrowing down in a depth first manner. From the results in Table 4 we can see that it is the standard GE that achieves the best performance, this is due to the fact the the Max problem used is in prefix notation. Thus in the standard GE mapping we would be developing the arithmetic component of the equation first as we always expand the left-most NT; unlike the other mappings presented. Now consider when using GE how both mutation and crossover get this ripple effect as outlined in detail in [9], [21], which means that from the chosen point on the chromosome for the genetic operation all codons following that point are subject to a context change and in the majority of cases this causes a trimming of the tree as can be seen in Fig. 6. In Fig. 7 we can see that the same ripple effect is in the right-most setup of GE. Note how in the right-most tree the only remaining part of the derivation tree contains structure and terminals that describe the end of a possible max solution. As we are using a prefix notation in the grammar we can say this will not contain the important additions or multiplications, yet it is imperative that we start our expression with a multiplication to maximize the possible value. GE can thus preserve this most important part of the possible solution throughout the evolutionary process, where as Right-Most mapping will have to discover this after nearly every mutation or crossover.

The big clue as to where the performance loss in πGE and Max comes from lies in Fig. 8. It can be seen that the ripple effect of πGE crossover will in general leave a saw tooth cut in the tree that will leave few if any of the leaf nodes of the trees. This means that in general a πGE mapping will have to try and evolve the whole solution at once rather than having a starting point like the other mappings have. If we compare the results in the previous section to these figures, we can say that there exists a preferential bias towards evolving a solution in a left to right manner as is performed by the standard GE mapping. There is a performance drop seen in the Right-Most mapping too, this is due to the reason stated in the previous paragraph where the ability to retain knowledge of the starting part of the expression is deemed most important as the difference

between adding 20+20 and multiplying 20*20 is huge and so can be deemed to be the most important symbol in the equation.

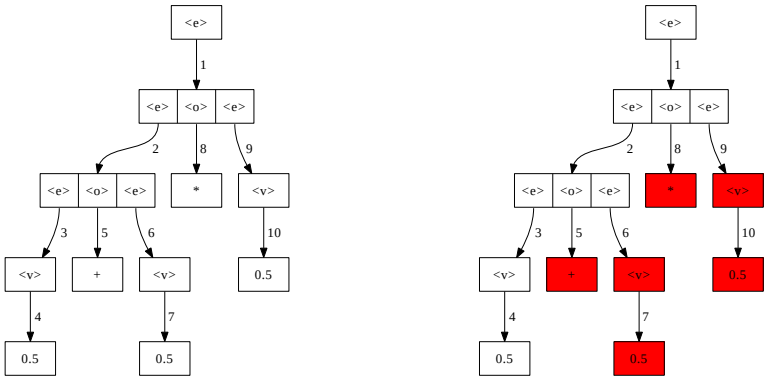

Figure 6: An illustration of the ripple effect of crossover and mutation on a GE derivation tree, Note the tree before crossover on the left and then on the right once crossover is applied after the fourth expansion how we are left with structure and solution on the left only as the shaded nodes will be replaced.

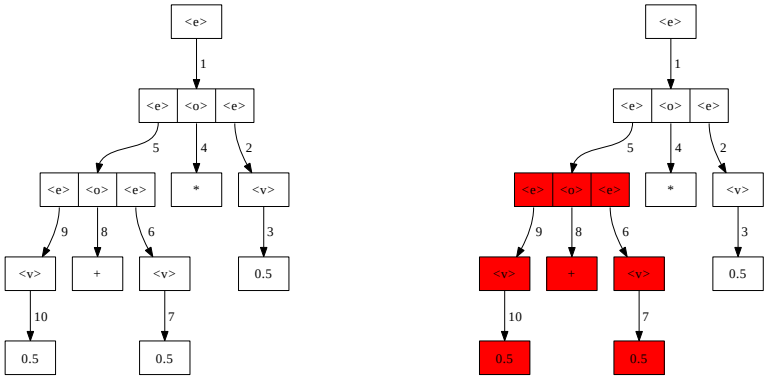

Figure 7: An illustration of the ripple effect of crossover and mutation on a Right-Most derivation tree, Note the tree before crossover on the left and then on the right once crossover is applied after the fourth expansion how we are left with structure and solution on the right only as the shaded nodes will be replaced.

Looking again at [8] and the results above we would like to further examine if a certain order of notation is better suited to a certain mapper. From what we have seen above the prefix notation used in this grammar for Max it is the reason for the performance decrease and should also be further examined to see if changing to a infix notation might level the playing field and remove all bias towards a certain type of mapping.

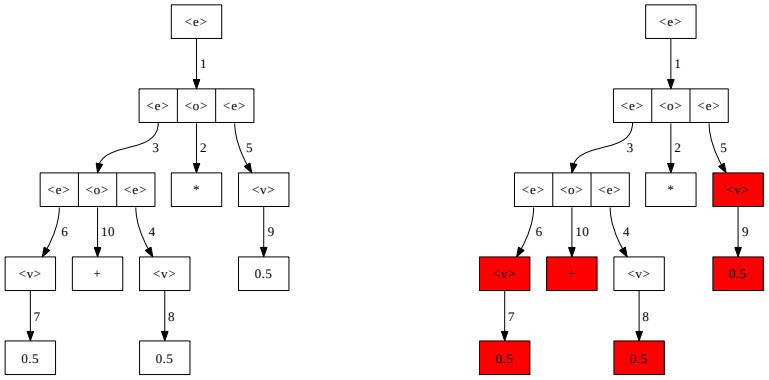

Figure 8: An illustration of the ripple effect of crossover and mutation on a πGE or Fixed-Order derivation tree, Note the tree before crossover on the left and then on the right one crossover is applied after the fourth expansion how we are left with structure and a very tiny amount of a solution as the shaded nodes will be replaced.

7 Conclusions & Future Work

We presented an investigation into the different performance levels of genotype-phenotype maps in GE on the Max problem in order to determine the respective levels of performance of each mapping. This was done to investigate whether the evolvability of πGE was responsible for the discrepancy. By fixing the order to be a single non evolvable order across the entire population the results indicated that the difference in performance was in some other aspect of the mapping.

Upon further examination of how the mappers worked we decided to examine how the genetic operators effect the performance specifically the ripple effect of crossover and mutation in GE. The idea of a new rightmost first expansion order was tested and from this it became apparent that to achieve the best performance in the Max problem domain, with a prefix grammar, such as the one used here, a left-most genotype-phenotype map is the best mapping order for GE. This study has helped us to gain a better understanding of how πGE works and also surfaced the idea that there is a mapping order suited for each type of problem. This is of great interest and is worth further investigation to identify what mapper is best for a specific type of problems based on criteria such as grammar notation, as well as which is the best general mapper for a range of problems. Once this has been established we intend to see if this trend holds true as we progress our research into the dynamic problem domain, or if what we now consider drawbacks to certain mappings become advantages in the dynamic problem domain.

Acknowledgments

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

References

[1] W. Banzhaf. Genotype-phenotype-mapping and neutral variation – A case study in genetic programming. In *Parallel Problem Solving from Nature III*, LNCS866. Springer-Verlag, 1994.

[2] M. F. Brameier and W. Banzhaf. *Linear Genetic Programming*. Springer, 2007.

[3] I. Dempsey, M. O'Neill, and A. Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments*. Studies in Computational Intelligence. Springer, 2009.

[4] D. Fagan, M. ONeill, E. Galván-López, A. Brabazon, and S. McGarraghy. An Analysis of Genotype-Phenotype Maps in Grammatical Evolution. *Genetic Programming*, pages 62–73, 2010.

[5] J.-L. Fernandez-Villacanas Martin and M. Shackleton. Investigation of the importance of the genotype-phenotype mapping in information retrieval. *Future Generation Computer Systems*, 19(1), 2003.

[6] C. Gathercole and P. Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 291–296. MIT Press, 1996.

[7] S. Harding, J. F. Miller, and W. Banzhaf. Evolution, development and learning using self-modifying cartesian genetic programming. In *GECCO '09: Proc. of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009.

[8] E. Hemberg, N. McPhee, M. O'Neill, and A. Brabazon. Pre-, in- and postfix grammars for symbolic regression in grammatical evolution. In *IEEE Workshop and Summer School on Evolutionary Computing*, 2008.

[9] B. J., M. J., O. M., and B. A. An analysis of the behaviour of mutation in grammatical evolution. In *EuroGP 2010 the 13th European Conference on Genetic Programming*. Springer, 2010.

[10] D. B. Kell. Genotype-phenotype mapping: genes as computer programs. *Trends in Genetics*, 18(11), 2002.

[11] R. E. Keller and W. Banzhaf. Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In *Genetic Programming 1996: Proc. of the First Annual Conference*. MIT Press, 1996.

[12] R. E. Keller and W. Banzhaf. Evolution of genetic code on a hard problem. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann, 2001.

[13] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

[14] W. Langdon and R. Poli. An analysis of the MAX problem in genetic programming. *Genetic Programming*, 1997.

[15] S. Margetts and A. J. Jones. An adaptive mapping for developmental genetic programming. In *Genetic Programming, Proc. of EuroGP'2001*, LNCS2038. Springer-Verlag, 2001.

[16] J. F. Miller and P. Thomson. Cartesian genetic programming. In *Genetic Programming, Proc. of EuroGP'2000*, LNCS1802. Springer-Verlag, 2000.

[17] M. O'Neill. *Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution*. PhD thesis, University Of Limerick, 2001.

[18] M. O'Neill, A. Brabazon, M. Nicolau, S. M. Garraghy, and P. Keenan. πgrammatical evolution. In *Genetic and Evolutionary Computation – GECCO-2004, Part II*, LNCS3103. Springer-Verlag, 2004.

[19] M. O'Neill, E. Hemberg, C. Gilligan, E. Bartley, J. McDermott, and A. Brabazon. GEVA: Grammatical evolution in java. *SIGEVOlution*, 3(2), 2008.

[20] M. O'Neill and C. Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language*. Genetic programming. Kluwer Academic Publishers, 2003.

[21] M. O'neill, C. Ryan, M. Keijzer, and M. Cattolico. Crossover in grammatical evolution. *Genetic Programming and Evolvable Machines*, 4(1):67–93, 2003.

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

[23] C. R. Stephens. Effect of mutation and recombination on the genotype-phenotype map. In *Proc. of the Genetic and Evolutionary Computation Conference*, volume 2. Morgan Kaufmann, 1999.