An analysis of Genotype-Phenotype Maps in Grammatical Evolution

David Fagan, Michael O'Neill, Edgar Galvan-Lopez, Anthony Brabazon, Sean McGarraghy

Natural Computing Research & Applications Group
University College Dublin, Ireland
david.fagan@ucd.ie,m.oneill@ucd.ie,
edgar.galvan@ucd.ie,anthony.brabazon@ucd.ie, sean.mcgarraghy@ucd.ie

Abstract. We present an analysis of the genotype-phenotype map in Grammatical Evolution (GE). The standard map adopted in GE is a depth-first expansion of the non-terminal symbols during the derivation sequence. Earlier studies have indicated that allowing the path of the expansion to be under the guidance of evolution as opposed to a deterministic process produced significant performance gains on all of the benchmark problems analysed. In this study we extend this analysis to include a breadth-first and random map, investigate additional benchmark problems, and take into consideration the implications of recent results on alternative grammar representations with this new evidence. We conclude that it is possible to improve the performance of grammar-based Genetic Programming by the manner in which a genotype-phenotype map is performed.

1 Introduction

Within the field of Genetic Programming (GP) [11, 19] the use of a genotype-phenotype map is not new [9, 1, 10, 20, 13, 5, 4, 8] 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 [14] and Grammatical Evolution (GE) [3, 18]. GE is a grammar-based form of GP which adopts a mapping from a linear genotype to phenotypic GP trees. O'Neill [15] presented a series of arguments for the adoption of a genotype-phenotype map for GP as 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.

For the first time this study presents an examination of the genotype-phenotype map of GE. A number of alternative mappers are proposed and performance is compared against the standard genotype-phenotype map. The remainder of the

paper is structured as follows. A brief overview of the essentials of GE are provided in Section 2 before an example of the standard genotype-phenotype map of GE 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 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 the plain text grammar. The explicit grammar allows GE to easily generate 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].

3 GE's Genotype-Phenotype Map

The genotype-phenotype map of GE operates as follows. The process begins from the embryonic start symbol of the grammar. Taking the simple grammar adopted for the Max problem provided in Fig. 6 this is `<prog>`, which by default is transformed into the non-terminal `<expr>`. There are two possible transformations which can be applied to `<expr>`. Either it will be replaced with `<op><expr><expr>` or with `<var>`. To decide what happens the next unused codon (an integer in this study) is read from the genome and we mod it's value by the number of choices available (i.e., $choice = integer \ \% \ 2$). Lets assume `<expr>` is transformed into `<op><expr><expr>`. In this situation there

is more than one non-terminal symbol in the current structure which needs to be transformed. The standard mapper in GE always selects the left-most non-terminal, which means in this case `<op>`. In the Max grammar `<op>` can be transformed into one of `+` or `*`, again by reading the next codon value and applying the mapping function with modulus 2. Assuming `*` is selected we end up with the structure `*<expr><expr>`. The mapping continues by taking the left-most non-terminal until we end up with a structure that is comprised exclusively of terminal symbols (i.e., in the case of the Max grammar these are `+`, `*`, and `0.5`).

A sample grammar is outlined below including an example chromosome. Fig. 1 outlines the depth-first order of expansion of the non-terminal symbols of the standard mapping process in GE. Potentially this introduces a structure bias to the search process as the focus of search is directed towards the left-hand branches and sub-trees of an individual structure. Alternatively if a breadth-first expansion was adopted, Fig. 2 illustrates how the order changes and thus the focus of evolutionary search takes a different direction towards broader tree structures. With the πGE approach [16] the order of expansion is itself evolvable with the genome being consulted as to which non-terminal to expand at each point of the derivation sequence.

```
<e> ::= <e> <o> <e> | <v>
<o> ::= + | -
<v> ::= X | Y
Chromosome ::= 12,2,8,3,5,2,9,14,6,3,8,10,7,12
```

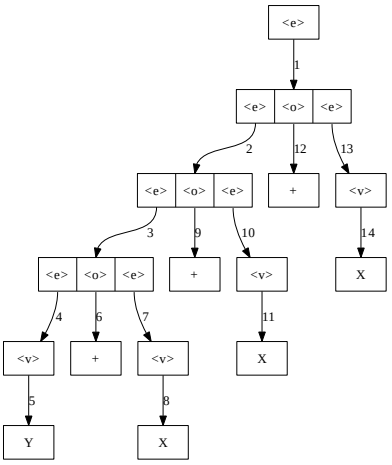

Fig. 1. An illustration of the order of a depth first expansion of the non-terminals in a derivation tree, leading to a solution of Y+X+X+X.

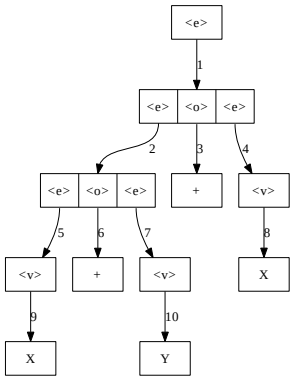

Fig. 2. An illustration of the order of a breadth first expansion of the non-terminals in a derivation tree, leading to a solution of X+Y+X.

4 Experimental Setup

We wish to test the null hypothesis that there is no difference in performance when alternative mapping strategies are adopted with GE. We will measure performance both in terms of the number of successful solutions found to each problem instance, and by examining the average best fitness.

We adopted GEVA v1.1 [17] for the experiments conducted in this study. The evolutionary parameters adopted on all problems are presented in Table 1. Note that we deliberately use a relatively small population size of 100 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 these toy benchmark problems.

Table 1. Parameter settings adopted on all problems examined.

Parameter	**Value**
generations	100
population size	100
replacement strategy	generational with elitism (10%)
selection	tournament (tsize=3)
mutation probability	0.01 (integer mutation)
crossover probability	0.9 (ripple)
initial chromosome length	200 codons (random init)
max wrap events	3

4.1 Benchmark Problems

Four standard GP benchmark problems were examined, and 50 independent runs performed for each setup on each problem. The grammar adopted in each case appear in Figs. 3, 4, 5, and 6.

Even-5-parity This is the classic benchmark problem in which evolution attempts to find the five input even-parity boolean function. The optimal fitness is obtained when the correct output is generated for each of the 32 test cases.

Symbolic Regression The classic quartic function is used here $x + x^2 + x^3 + x^4$ with 20 input-output test cases drawn from the range -1 to 1. Fitness is simply the sum of the errors. We measure success on this problem using the notion of hits, where a hit is achieved when the error is less than 0.01.

Santa Fe ant trail The objective is to evolve a program to control the movement of an artificial ant on a toroidal grid of size 32 by 32 units. 89 pieces of food are located along a broken trail, and the ant has 600 units of energy to find all

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

<expr> ::= <expr> <op> <expr>
         | ( <expr> <op> <expr> )
         | <var>
         | <pre-op> ( <var> )

<pre-op> ::= not

<op> ::= "|"
       | &
       | ^

<var> ::= d0 | d1 | d2 | d3 | d4
```

Fig. 3. The grammar adopted for the Even-5-parity problem.

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

<expr> ::= <expr> <op> <expr>
         | ( <expr> <op> <expr> )
         | <pre-op> ( <expr> )
         | <protected-op>
         | <var>

<op> ::= + | * | -

<protected-op> ::= div( <expr>, <expr>)

<pre-op> ::= sin | cos | exp | inv | log

<var> ::= X | 1.0
```

Fig. 4. The grammar adopted for the Symbolic Regression problem instance.

```
<prog> ::= <code>

<code> ::= <line> | <code> <line>

<line> ::= <condition>\n
         | <op>\n

<condition> ::= if(food_ahead()==1){
                   <opcode>
                }
                else { <opcode> }

<op> ::=  left(); | right(); | move();

<opcode> ::= <op> | <opcode> <op>
```

Fig. 5. The grammar adopted for the Santa Fe ant trail problem.

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

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

<op> ::= +
       | *

<var> ::= 0.5
```

Fig. 6. The grammar adopted for the Max problem instance.

the food. A unit of energy is consumed when the ant uses one of the following operations: `move()`, `right()` or `left()`. The ant also has the capability to look ahead into the square directly facing it to determine if there is food present.

Max 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 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 as is shown by Langdon et al [12].

4.2 Mappers

Four alternative mapping strategies are examined in this study. 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. The opposite **Breadth-first** strategy was implemented, which maps all of the non-terminal symbols at each successive level of the derivation tree before moving on to the next deepest level. The **πGE** mapper as first described by O'Neill et al. [16] is the third mapper analysed. πGE lets the evolving genome decide which non-terminal to expand at each step in the derivation sequence. Finally we adopt a **Random** control strategy, which randomly selects a non-terminal to expand amongst all of the non-terminals that currently exist in an expanding derivation sequences. This is equivalent to a randomised πGE approach where the order of expansion is not evolved, rather it is chosen at random each time it is performed.

5 Results

The number of runs (out of 50) that successfully found a perfect solution to each problem is presented in Table 2. On three out of the four problems the πGE mapper is the most successful. None of the mappers found a perfect solution to the Max problem with the parameter settings adopted.

Average best fitness plots (over 50 runs) for each problem can be seen in Figs. 7, 8, 9 and 10. Table 3 records the average best fitness and standard deviation at the final generation. The results presented in these figures and table support the success rate data, with the πGE mapper variant outperforming the alternatives on Even-5-Parity, Symbolic Regression and the Santa Fe Ant. However, on the Max problem instance the standard depth-first mapper has a performance edge.

It is worth noting that the random "control" mapper performs the worst on all of the problems examined in terms of success rates and in terms of the average best fitness attained. A slight exception is on the Santa Fe ant trail where the random mappers performs as well as both depth and breadth-first alternatives, in terms of the average best fitness at the final generation.

Table 2. Instances of Successful Solution Found over 50 runs.

Mapper	Even 5	Santa Fe	Sym Reg	Max
BF	29	1	9	0
DF	31	2	9	0
Rand	13	0	0	0
πGE	38	4	17	0

We also recorded the size of evolving genomes (Figs. 11, 12, 13 and 14) and derivation trees. Derivation tree size was measured both in terms of the number of nodes in a tree (Figs. 15, 16, 17 and 18) and the depth of a tree (Figs. 19, 20, 21 and 22).

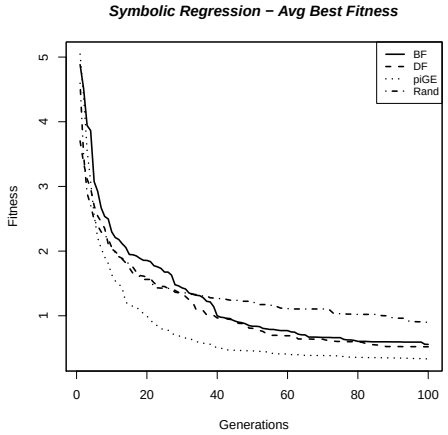

Fig. 7. Average Best Fitness on the Symbolic Regression problem instance.

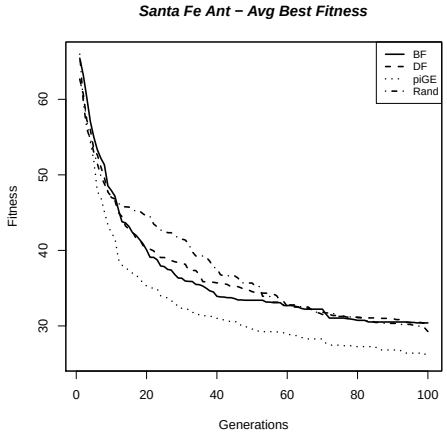

Fig. 8. Average Best Fitness on the Santa Fe ant problem.

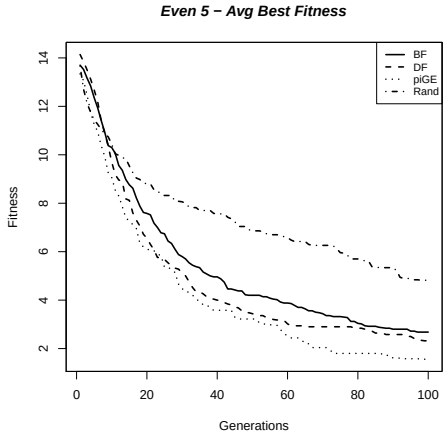

Fig. 9. Average Best Fitness on the Even-5-parity problem.

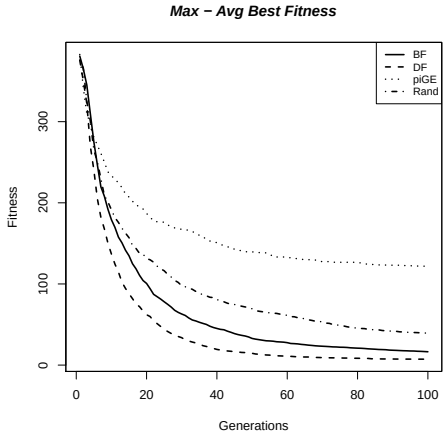

Fig. 10. Average Best Fitness on the Max problem instance.

6 Discussion

While the results show that πGE did not perform as well on the Max problem, relative to the other problems, it is worth noting that solving the Max problem is more about refining the content of the tree not the structure [7]. The Depth-first map appears to be able to generate larger tree structures more rapidly (both in terms of number of nodes and tree depth, see Figs. 18 and 22) when compared to the alternative mapping strategies. This allows search additional time to focus

Table 3. Average Best Fitness Values after 100 generations over 50 independent runs.

Mapper	Even 5	Santa Fe	Sym Reg	Max
	Avg.Best(std)	Avg.Best(std)	Avg.Best(std)	Avg.Best(std)
BF	2.68(3.41)	30.4(13.92)	0.56(0.65)	16.44(14.66)
DF	2.32(3.26)	30.34(14.39)	0.52(0.89)	7.23(10.14)
Rand	4.82(3.29)	29.26(12.07)	0.89(0.76)	121.89(27.45)
πGE	1.52(2.92)	25.64(14.52)	0.33(0.56)	39.31(24.97)

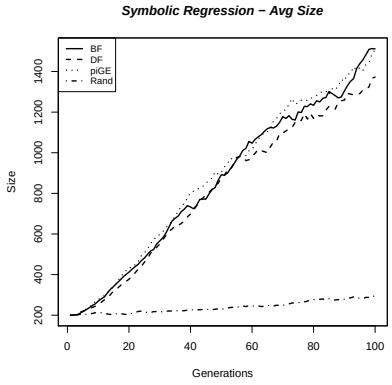

Fig. 11. Average size of individual on the Symbolic Regression problem instance.

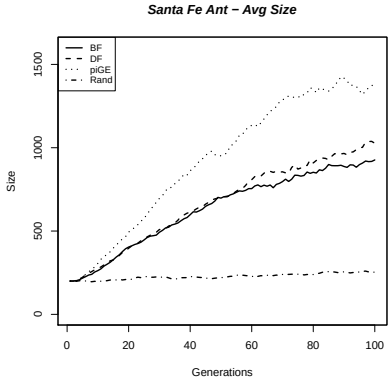

Fig. 12. Average size of individual on the Santa Fe ant problem.

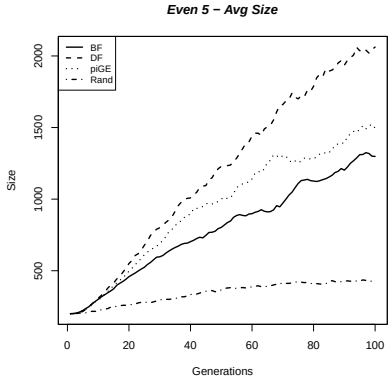

Fig. 13. Average size of individual on the Even-5-Parity problem.

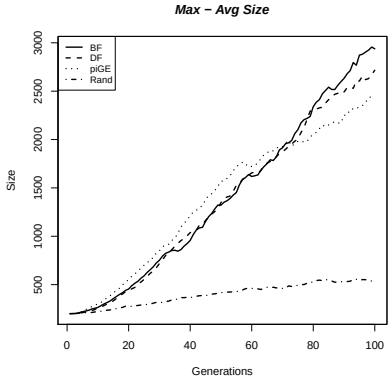

Fig. 14. Average size of individual on the Max problem instance.

on tree content towards the desired `*`'s towards the root and `+`'s towards the function nodes near the leaves. The Max problem is more suited to a systematic

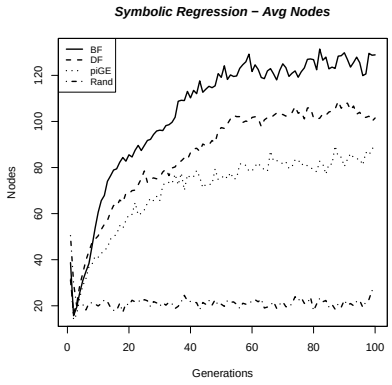

Fig. 15. Average number of Derivation Tree Nodes on the Symbolic Regression problem instance.

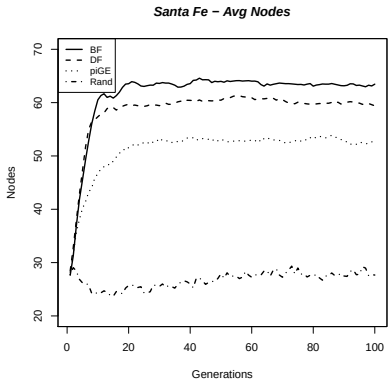

Fig. 16. Average number of Derivation Tree Nodes on the Santa Fe ant problem.

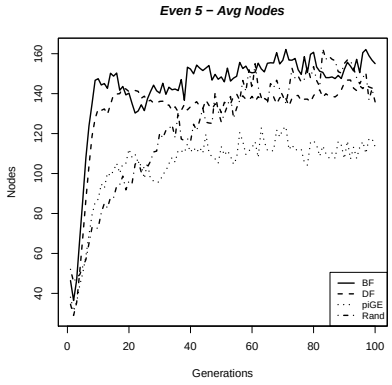

Fig. 17. Average number of Derivation Tree Nodes on the Even-5-parity problem.

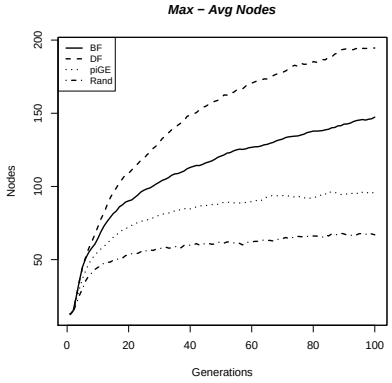

Fig. 18. Average number of Derivation Tree Nodes on the Max problem.

pre-order (Depth-first) or level-order (Breadth-first) traversal of the tree, leading to better results faster than the πGE alternative. On all the other problems the Breadth-first map produces larger tree structures both in terms of node count and tree depth (Figs. 15-22).

With respect to the length of the integer genomes it is clear from Figs. 11-14 that the control random mappers lack of order results in the overall lengths of individuals remaining relatively constant over time. The opposite behaviour is observed in the cases of Depth-first, Breadth-first and πGE with the usual GP-bloat behaviour being observed.

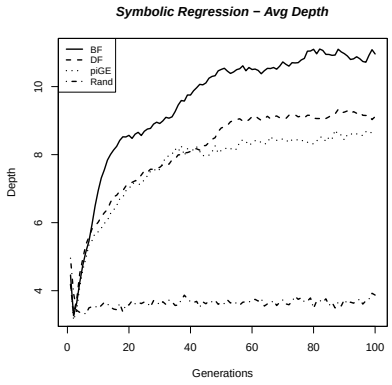

Fig. 19. Average Derivation Tree Depth on the Symbolic Regression problem.

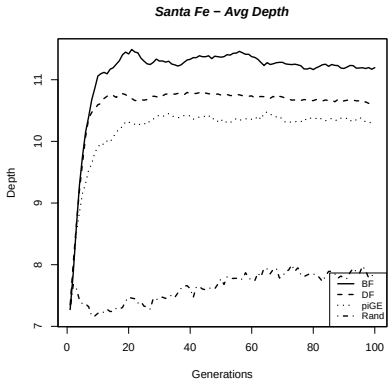

Fig. 20. Average Derivation Tree Depth on the Santa Fe ant problem.

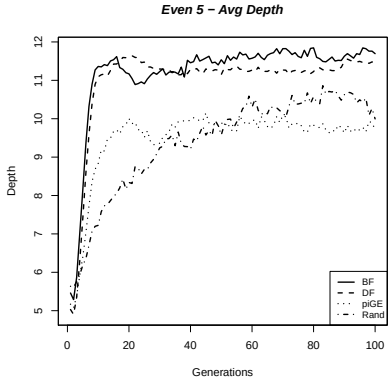

Fig. 21. Average Derivation Tree Depth on the Even-5-parity problem.

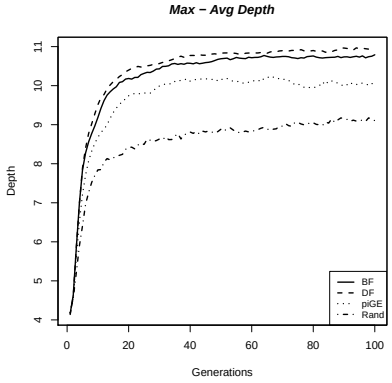

Fig. 22. Average Derivation Tree Depth on the Max problem instance.

In light of the comparison between a Depth-first and Breadth-first mapper presented here, it is interesting to recall the observations made in a study by Hemberg et al. [6]. In the earlier research three grammar variants were examined in the context of Symbolic Regression. 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.

Effectively the order of the mapping sequence was modified in the Hemberg et al. study to a *Postorder* mapper purely by modifying the syntax of the expressions being evolved. Given that the Breadth-first map adopted in this study is producing similar performance characteristics to the standard Depth-first map, there must be some advantage in conducting the mapping sequence at least partly Breadth-first, and partly in a Depth-first manner. Given the earlier findings on the *Postorder* mapping, there may also be an advantage in reversing the order of expansion between a pre-, post-order, and possibly in-order. It will require further analysis to ascertain if a similar mixture of mapping order is effectively being evolved with the πGE approach, which may go some way to explain the relative advantage πGE has over the other mappers.

7 Conclusions & Future Work

We presented an analysis of the genotype-phenotype map in Grammatical Evolution by comparing performance of the standard *depth-first* approach to *breadth-first*, π*GE*, and *random* variations. Across the benchmark problems analysed we observe an advantage to the adoption of the more flexible πGE map, which is under the control of evolution. Given the additional overhead that the πGE map has, due to the extra degree of freedom which allowing the path of the derivation sequence to be evolvable and the subsequent increase of the overall search space size that this entails, the results are even more impressive. Further research is required to establish Why the more evolvable approach is providing a performance advantage, and this is the current focus of our efforts. With this deeper understanding we can then potentially improve upon the πGE approach and/or develop novel mappers with more evolvable characteristics. We are especially interested in how evolvable genotype-phenotype maps will perform in dynamic environments, and this will form an integral part of the next phase of this research.

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. 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.
5. 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.
6. 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.
7. 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.
8. D. B. Kell. Genotype-phenotype mapping: genes as computer programs. *Trends in Genetics*, 18(11), 2002.
9. 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.
10. 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.
11. 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.
12. W. Langdon and R. Poli. An analysis of the MAX problem in genetic programming. *Genetic Programming*, 1997.
13. S. Margetts and A. J. Jones. An adaptive mapping for developmental genetic programming. In *Genetic Programming, Proc. of EuroGP'2001*, LNCS2038. Springer-Verlag, 2001.
14. J. F. Miller and P. Thomson. Cartesian genetic programming. In *Genetic Programming, Proc. of EuroGP'2000*, LNCS1802. Springer-Verlag, 2000.
15. M. O'Neill. *Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution*. PhD thesis, University Of Limerick, 2001.
16. 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.
17. M. O'Neill, E. Hemberg, C. Gilligan, E. Bartley, J. McDermott, and A. Brabazon. GEVA: Grammatical evolution in java. *SIGEVOlution*, 3(2), 2008.
18. M. O'Neill and C. Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language*. Genetic programming. Kluwer Academic Publishers, 2003.
19. 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).
20. 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.