Investigating Mapping Order in πGE

David Fagan, Miguel Nicolau, Michael O'Neill, Edgar Galván-López, Anthony Brabazon, Sean McGarraghy

***Abstract*— We present an investigation into the genotype-phenotype map in Position Independent Grammatical Evolution (πGE). Previous studies have shown πGE to exhibit a performance increase over standard Grammatical Evolution (GE). The only difference between the two approaches is in how the genotype-phenotype mapping process is performed. GE uses a leftmost non terminal expansion, while πGE evolves the order of mapping as well as the content. In this study, we use the idea of focused search to examine which aspect of the πGE mapping process provides the lift in performance over standard GE by applying our approaches to three benchmark problems taken from specialised literature. We examined the traditional πGE approach and compared it to two setups which examined the extremes of *mapping order search* and *content search*, and against setups with varying ratios of content and order search. In all of these tests a purely content focused πGE was shown to exhibit a performance gain over the other setups.**

I. Introduction

The adoption of a genotype-phenotype map for Genetic Programming (GP) has demonstrated performance advantages over traditional tree-based GP, with many examples in the literature, e.g., see [8], [2], [9], [6], [1]. One of the most popular grammar-based forms of GP, Grammatical Evolution (GE), adopts a genotype-phenotype map which has been argued to provide a number of advantages over standard GP [10]. The genotype-phenotype map adopted in GE results in the construction of a solution structure (a derivation tree) in the language represented by an input grammar. This map is deterministic, following a left-to-right, depth-first development of the structure. An alternative, evolvable map was proposed and significant performance gains were observed in the problem domains tested [11]. The resulting algorithm, πGE or Position Independent Grammatical Evolution, was recently compared to a number of alternative genotype-phenotype maps for GE [4]. The alternative mappers effectively modified the order in which the developing structure is expanded, and it was observed that the evolvable map of πGE was the most successful. In light of this, we now wish to investigate more deeply the impact of mapping order on πGE in an attempt to understand why performance gains are achieved.

The remainder of the paper is structured as follows. Sections II and III contain a brief overview of GE and how the πGE approach to GE differs. The specific research questions this paper addresses and the experimental setup are presented in Sections IV and V. Results are reported and analysed in Section VI. Finally the paper reaches a close with conclusions and future work in Section VII.

David Fagan is a Ph.D. student in the Computer Science and Informatics Department of University College Dublin, Dublin Ireland. He is working in the Natural Computing Research and Applications Group in UCD's Complex and Adaptive Systems Laboratory. UCD CASL, Belfield, Dublin 4, Ireland (email david.fagan@ucdconnect.ie).

Miguel Nicolau, Michael O'Neill, Edgar Galván-López, Anthony Brabazon and Sean McGarraghy are all members of the Natural Computing Research and Applications Group in UCD's Complex and Adaptive Systems Laboratory. UCD CASL, Belfield, Dublin 4, Ireland (email: {miguel.nicolau, m.oneill, edgar.galvan, anthony.brabazon, sean.mcgarraghy}@ucd.ie).

II. Grammatical Evolution

GE is a form of GP that takes principles from molecular biology and combines them with the representational power of formal grammars. GE's rich modularity provides 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 applied to 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].

III. GE to πGE

The only difference between standard GE and πGE is in the mapping process from genotype to phenotype. In GE we start of with a non terminal node (NT) or start symbol. In the case of the example grammar shown in Fig.1, that would be `<e>.<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

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

the tree, where <e> expands to <e><o><e> (12%2) . Once this is done a leftmost NT first derivation (where the leftmost unexpanded NT is chosen) is done to expand the derivation tree until no NTs remain to be expanded. 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, as well as the application of Eq.1 that results in that expansion. In the form of 1(12%2) where 1 is the expansion order and 12%2 is the application of Eq.1.

$$New\ Node\ =\ codon\ val.\ \%\ num.\ of\ rules\ for\ NT \quad (1)$$

```
<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
```

Fig. 1. Example Grammar and Chromosome.

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 pair 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. Fig's.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.

$$NT\ to\ expand\ =\ codon\ value\ \%\ number\ of\ NT's \quad (2)$$

IV. Adjusting πGE Mapping

In order to understand why the πGE map operates so successfully we are faced with a number of questions. We can see that the order of the mapping is subject to evolutionary search in πGE whereas it is fixed in standard GE. Perhaps there is an unfavourable bias introduced into the search by the strict depth-first, left-to-right mapping order of GE, and simply by randomising this mapping order the bias is overcome with the resulting performance gains. It may be that this is not

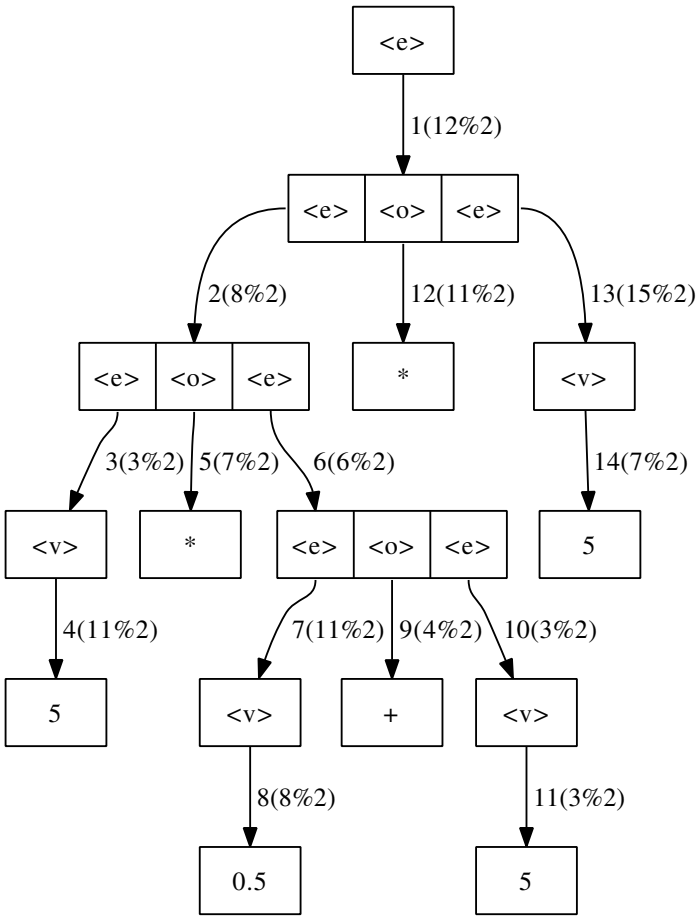

Fig. 2. Standard GE Genotype to Phenotype Mapping.

```
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)
```

Fig. 3. NT selection process in πGE.

sufficient to explain the observed gains and that the evolvable nature of the order itself confers the advantage on πGE.

In the standard πGE setup the rate of search performed on the order and content codons relative to each is on a basis of 1:1. If we allowed this search to be rebalanced, will this confer an advantage. For example, it may be that the rate of search directed towards the order of the map should be undertaken at a lower rate than that of the content in order to give each evolved mapping order a chance to be sampled for a number of alternative content sets (c.f., genetic code [7] and grammar evolution [13], [5] studies). In standard πGE, an unknown amount of order and content codons are mutated, thus giving a fluctuating mix of mutation between codon types across the population. This can have a drastic effect on the

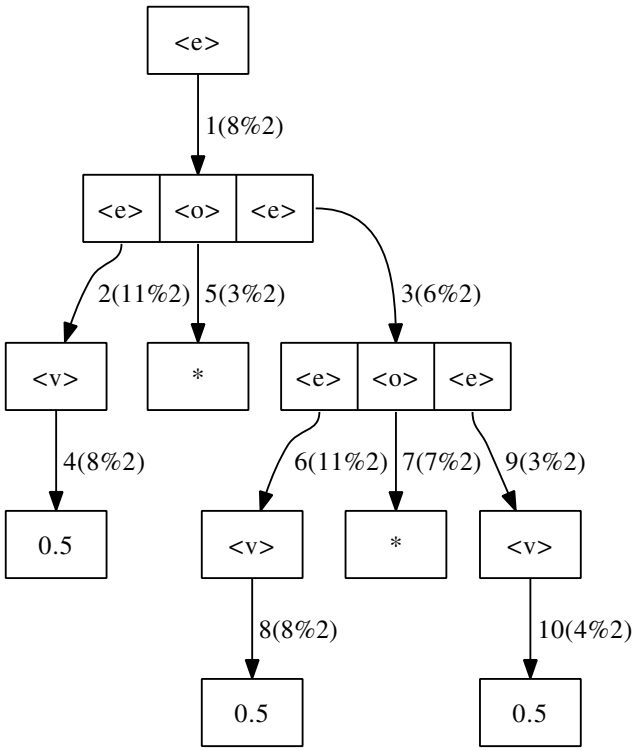

Fig. 4. Standard πGE Genotype to Phenotype Mapping.

mapping process. Consider the effect changing one of the order codons has on the mapping, compared to changing one content codon. The act of changing a content codon will change what the current non-terminal becomes. While this will effect any subsequent mapping in the sub-tree emanating from this non-terminal (the ripple effect), it will not effect any other sub-trees. Changing one order codon on the other hand, will in most cases move the position of expansion on the tree to a new position. This will affect both expansion on the sub-tree emanating from the original position in the tree and the new position that is to be expanded based on the changing of one codon.

Four experimental setups are examined where we control the order of the search being undertaken.

1) **Order**: Mutation events are restricted to codons responsible for determining the mapping order. The content codons are fixed in this setup with the exception of the operation of the crossover operator which may shuffle the content codons between individuals. The results observed on this setup relative to the others will allow us to determine the contribution of the search focused on the order codons towards the success of πGE.
2) **Content**: Mutation events are restricted to codons responsible for production rule selection. The order codons are fixed here (with the exception of the shuffling of order codons between individuals by crossover). When compared to a standard GE mapping, in effect the mapping order is largely randomised here upon initialisation of the order codons in the first generation.
3) **πGE**: Mutation events are allowed on both order and content codons.
4) **Content:Order**: Two variations on πGE are examined where the ratio of order to content mutation events are varied to examine the situation where the search is allowed to continue on both the order and content codons, but at different relative rates (namely, 2:1 and 1:2, in contrast to the 1:1 of πGE). This will allow us to determine if there may be an advantage in rebalancing the relative rate of codon and order search.

TABLE I
PARAMETER SETTINGS ADOPTED ON ALL PROBLEMS.

Parameter	Value
generations	100
population size	100
replacement strategy	generational with elitism (10%)
selection	tournament (tsize=3)
mutation probability	0.01(int. mut.)
crossover probability	0.9 (ripple)
initial chromosome length	200 codons (rand. init)
max wrap events	0

By undertaking this investigation we hope to gain an understanding into which aspect of the πGE mapping process is responsible for the performance gain over GE and also to see if the current πGE setup can be refined for greater performance at finding solutions to problems through the application of focused mutation to the chromosome.

V. EXPERIMENTAL SETUP

We wish to test the null hypothesis that there is no difference in performance when we focus mutation on different parts of the chromosome in πGE. Performance in these cases will be assessed in terms of the number of successful solutions found for each problem instance, and by examining the average best fitness.

We modified GEVA v1.1[12] for the experiments conducted in this study, in order to perform the πGE mapping. The evolutionary parameters adopted on all problems are presented in Table I. 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 to find a perfect solution, and therefore allow us to discriminate more clearly performance differences on these toy benchmark problems.

A. Benchmark Problems

Three standard GP benchmark problems were examined, and 50 independent runs performed for each setup on each problem. The grammar adopted in each case appears in Figs. 5, 6 and 7.

a) 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.

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

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

<pre-op> ::= not

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

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

Fig. 5. 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. 6. The grammar adopted for the Symbolic Regression problem instance.

b) 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.

c) 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 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.

```
<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. 7. The grammar adopted for the Santa Fe ant trail problem.

TABLE II
MUTATION RATES FOR EXPERIMENTAL SETUPS

Setup	Order Rate	Content Rate
1:1 ratio (πGE)	0.01	0.01
1:0 ratio (all order)	0.02	0.00
0:1 ratio (all content)	0.00	0.02
2:1 ratio (double order)	0.0133	0.0067
1:2 ratio (double content)	0.0067	0.0133

B. Mutation Rates

In the four experimental setups proposed, we are modifying the effective rate of mutational search, by varying the rate of mutation between content and order codons. In standard πGE, each codon is mutated at the specified rate, but when mutation is restricted to order or content codons (or any order:content ratio), specific mutation rates for the order and content codons have to be applied. These have been calculated using the following equations:

$$p_o = \frac{o}{o+c} * p_{mut} * 2 \qquad p_c = \frac{c}{o+c} * p_{mut} * 2$$

where o and c are the ratios for order and content mutations (as in o:c), and p_o and p_c are the required mutation rates for content and order codons, respectively.

Note that this equation is applicable to any sort of ratios, including 1:1 (i.e. standard πGE), 1:0 (i.e. order only), and 0:1 (i.e. content only). The calculated probabilities of mutation used are shown in Table II.

VI. RESULTS

The results for the four experimental setups described in Section IV are now presented. We divide their exposition into two parts by focusing in the first instance on the scenarios where search is restricted to either the content or order codons, before examining the results for the alternative relative rates of order:content search.

A. Results for standard πGE, order-only and content-only

Tables III (with crossover) and IV (without crossover) show the number of solutions found to the three problems as well as the mean best fitness and the standard deviation over the 50 runs performed for each setup. Figs. 8 to 10 also plot the mean best fitness for the results with crossover, and Figs. 11 to 13 the results without crossover.

These figures show that focusing the search purely on the order codons (1:0 setup) tends to give the worse results. There is however still evolution in terms of fitness over time, which suggests that merely by shuffling the mapping order, the change of context of the content codons will in effect generate new content material (or rather change the function of the existing content material). Also note the better performance of this setup when the crossover operator is used; this suggests that crossover is of great help in maximising the utility of content codons, by trying out different contexts across individuals.

The results obtained with content-only mutation (0:1 setup) seem to give the best results. When mutation events are

TABLE III

RESULTS WITH CROSSOVER BASED ON 50 RUNS

Problem	Setup	Mean Best Fitness (stdev)	Successes
Even 5	πGE	2.70(3.56)	30
	1:0	4.72(3.84)	16
	0:1	1.86(3.14)	35
Santa Fe	πGE	29.58(12.66)	1
	1:0	27.86(13.98)	3
	0:1	24.4(13.77)	3
Sym Reg	πGE	0.44(0.94)	19
	1:0	0.54(1.14)	7
	0:1	0.19(0.27)	18

TABLE IV

RESULTS WITH NO CROSSOVER BASED ON 50 RUNS

Problem	Setup	Mean Best Fitness (stdev)	Successes
Even 5	πGE	1.92(3.35)	36
	1:0	5.18(3.46)	11
	0:1	1.94(3.13)	34
Santa Fe	πGE	33.54(14.73)	0
	1:0	31.78(14.41)	0
	0:1	35.96(11.71)	1
Sym Reg	πGE	0.97(1.86)	12
	1:0	1.37(2.43)	8
	0:1	0.69(1.15)	13

restricted to the content codons, the amount of search on the order setup is effectively reduced, which suggests that alternative (and initially random) mapping orders are superior. Note however the difference in performance when crossover is not used. This seems to suggest that some limited search in the order codons, through the effect of the crossover operator, is useful to explore the search space.

B. Results for 2:1 *and* 1:2 *ratios*

In order to investigate the contribution of the order search to the success of πGE, different ratios of order:content mutation were explored, and are presented in this section. Tables V (with crossover) and VI (without crossover) show the results obtained.

The two ratio mixes presented were designed to see what effect refining the ratio of search has in πGE. What is interesting to note is that the 1:2 and 2:1, perform on a par with the purely content focused mutation when we eliminate crossover. However with crossover, a content based approach gets a performance gain. This backs up the claims made above that crossover performs some form of order based search in the search space when dealing with the content only setup and suggests that some form of order search would be desirable. However the variance in performance between the two ratio setups suggest that trying to find the universal good setup for a set of problems using a ratio technique of search is quite hard. It is evident in that 1:2 out performs 2:1 and vice versa in different problems. This volatility however is not present in a purely content based setup which has what can be deemed consistent good performance across the board.

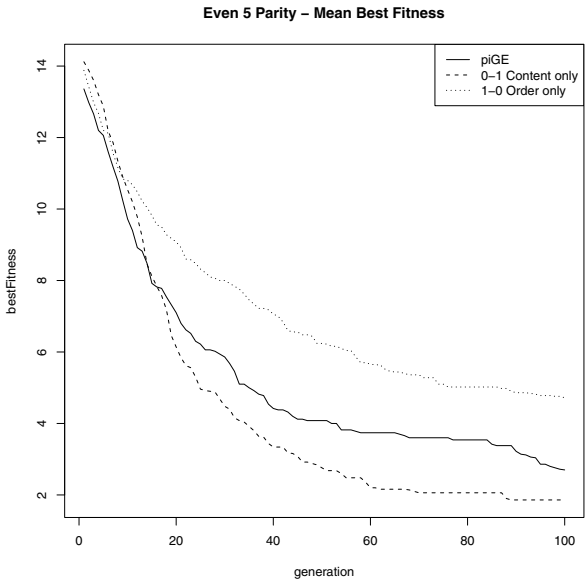

Fig. 8. Even 5 average best fitness with Crossover.

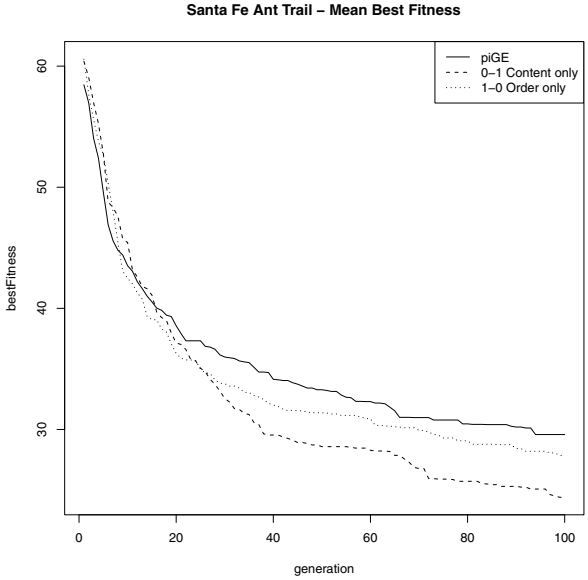

Fig. 9. Santa Fe Ant average best fitness with Crossover.

It is worth keeping in mind the mutation rates in these setups and from these results it can be said the while the 1:2 and 2:1 ratio setups were designed to see what more order search and more content search, both setups are more of less a slightly skewed standard πGE. It would be of interest to see what the addition of some order mutation may have had to the content-only setup rather than trying to balance the mutation rates, to keep an effective mutation across the board.

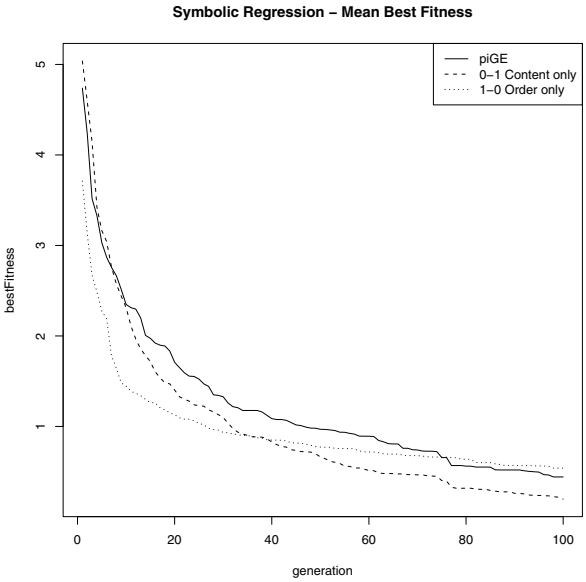

Fig. 10. Sym. Reg. average best fitness with Crossover.

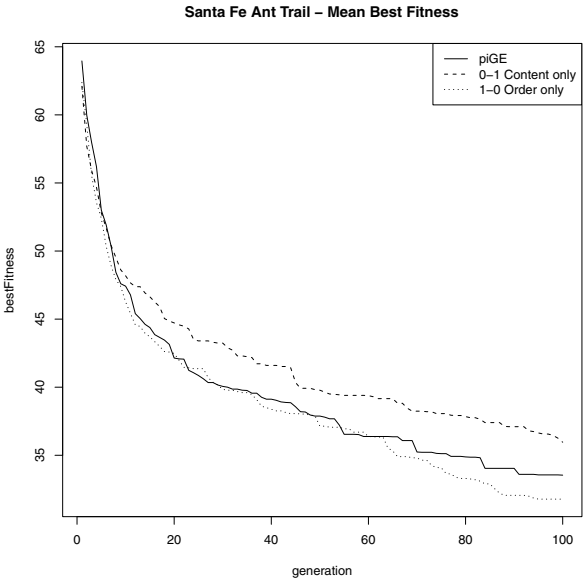

Fig. 12. Santa Fe Ant average best fitness without Crossover.

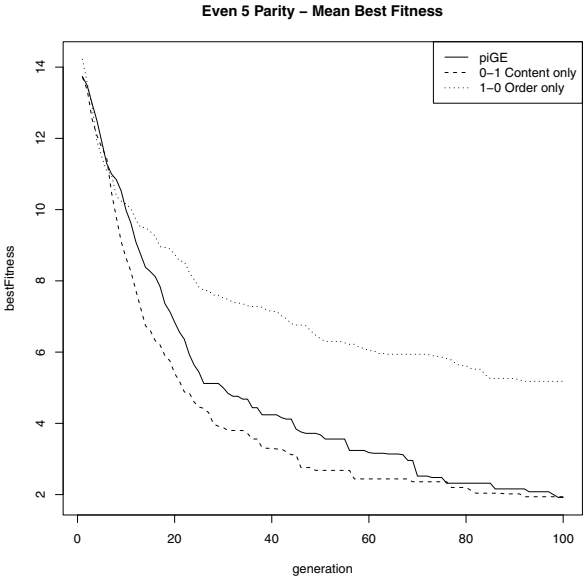

Fig. 11. Even 5 average best fitness without Crossover.

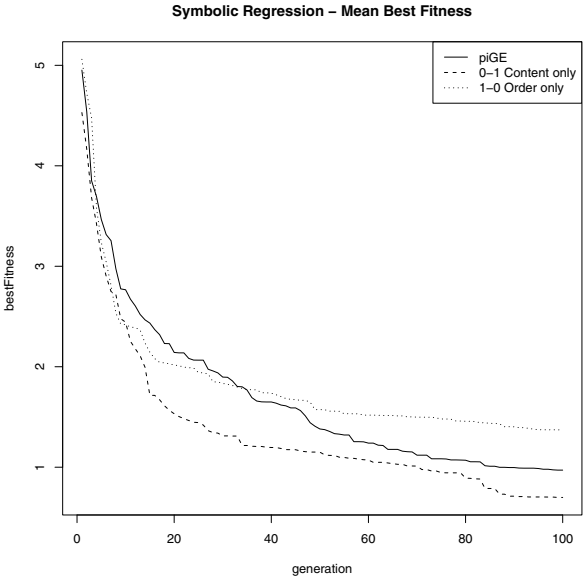

Fig. 13. Sym. Reg. average best fitness without Crossover.

TABLE V
RESULTS WITH CROSSOVER BASED ON 50 RUNS

Problem	Setup	Mean best fitness (stdev)	Successes
Even 5	0:1	1.86(3.14)	35
	1:2	1.40(2.68)	38
	2:1	1.86(3.00)	35
Santa Fe	0:1	24.4(13.77)	3
	1:2	26.84(13.33)	2
	2:1	25.84(15.56)	5
Sym Reg	0:1	0.19(0.27)	18
	1:2	0.47(0.49)	9
	2:1	0.33(0.39)	11

TABLE VI
RESULTS WITH NO CROSSOVER BASED ON 50 RUNS

Problem	Setup	Mean best fitness (stdev)	Successes
Even 5	0:1	1.94(3.13)	34
	2:1	2.98(3.81)	29
	1:2	2.22(3.51)	35
Santa Fe	0:1	35.96(11.71)	1
	2:1	25.86(16.50)	4
	1:2	31.24(12.48)	1
Sym Reg	0:1	0.69(1.15)	13
	2:1	0.39(0.59)	12
	1:2	0.95(1.81)	11

VII. Conclusions and Future Work

We set out to examine the impact of mapping order on the performance of πGrammatical Evolution, and to try to measure why the πGE map operates so successfully relative to standard GE. This is achieved through the comparison of a series of setups where the relative amount of content to order search is rebalanced. The results provide evidence to support a rebalancing of the search towards the content codons relative to the mapping order. In other words a slower rate of search on mapping order is desirable relative to the content rate of search. It is hypothesised that this presents an advantage by allowing each mapping order a fair chance to be sampled by alternative content sets.

In future work we wish to test this hypothesis and also examine the implications of these findings for search in dynamic problem environments. Through the adoption of an evolvable (dynamic) mapping process, the ability to inject additional order codon mutation during and/or after environmental change may allow a faster adaptation to the new environmental conditions.

VIII. Acknowledgments

This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868. MN thanks the National Digital Research Centre for funding support.

References

[1] W. Banzhaf. Genotype–phenotype–mapping and neutral variation – A case study in genetic programming. In Y. Davidor and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature – PPSN III*, volume 866 of *Lecture Notes in Computer Science*, pages 322–332, Berlin, 1994. Springer Verlag.

[2] M. Brameier and W. Banzhaf. *Linear Genetic Programming*. Number XVI in Genetic and Evolutionary Computation. 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. O'Neill, E. Galvan-Lopez, A. Brabazon, and S. McGarraghy. An analysis of genotype-phenotype maps in grammatical evolution. In A. I. Esparcia-Alcazar, A. Ekart, S. Silva, and S. Dignum, editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 62–73, Istanbul, 7-9 Apr. 2010. Springer.

[5] E. Hemberg, M. O'Neill, and A. Brabazon. Altering search rates of the meta and solution grammars in the mGGA. In M. O'Neill, L. Vanneschi, S. Gustafson, A. I. Esparcia Alcazar, I. De Falco, A. Della Cioppa, and E. Tarantino, editors, *Proceedings of the 11th European Conference on Genetic Programming, EuroGP 2008*, volume 4971 of *Lecture Notes in Computer Science*, pages 362–373, Naples, 26-28 Mar. 2008. Springer.

[6] R. E. Keller and W. Banzhaf. Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 116–122, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[7] R. E. Keller and W. Banzhaf. The evolution of genetic code in genetic programming. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1077–1082, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[8] J. R. Koza and F. H. Bennett III. Automatic synthesis, placement, and routing of electrical circuits by means of genetic programming. In L. Spector, W. B. Langdon, U.-M. O'Reilly, and P. J. Angeline, editors, *Advances in Genetic Programming 3*, chapter 6, pages 105–134. MIT Press, Cambridge, MA, USA, June 1999.

[9] J. F. Miller and S. L. Smith. Redundancy and computational efficiency in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 10(2):167–174, Apr. 2006.

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

[11] 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.

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

[13] M. O'Neill and C. Ryan. Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code. In M. Keijzer, U.-M. O'Reilly, S. M. Lucas, E. Costa, and T. Soule, editors, *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, volume 3003 of *LNCS*, pages 138–149, Coimbra, Portugal, 5-7 Apr. 2004. Springer-Verlag.