Understanding Expansion Order and Phenotypic Connectivity in πGE

David Fagan[1], Erik Hemberg[2], Michael O'Neill[1], and Sean McGarraghy[1]

[1] Natural Computing Research & Applications Group
University College Dublin, Ireland
[2] MIT CSAIL
{david.fagan,m.oneill,sean.mcgarraghy}@ucd.ie, hembergerik@csail.mit.edu

Abstract. Since its inception, πGE has used evolution to guide the order of how to construct derivation trees. It was hypothesised that this would allow evolution to adjust the order of expansion during the run and thus help with search. This research aims to identify if a specific order is reachable, how reachable it may be, and goes on to investigate what happens to the expansion order during a πGE run. It is concluded that within πGE we do not evolve towards a specific order but a rather distribution of orders. The added complexity that an evolvable order gives πGE can make it difficult to understand how it can effectively search, by examining the connectivity of the phenotypic landscape it is hoped to understand this. It is concluded that the addition of an evolvable derivation tree expansion order makes the phenotypic landscape associated with πGE very densely connected, with solutions now linked via a single mutation event that were not previously connected.

1 Introduction

Position Independent Grammatical Evolution[11], or πGE, has been shown to exhibit performance on a par with and in many cases exceeds the performance of Grammatical Evolution (GE)[13] on a wide range of problem domains[2,3,5,11]. πGE is an extension of GE where the order of expansion of the derivation tree is controlled by evolution. It was proposed that this added dimension to the standard GE genotype-phenotype map would allow for search to be performed in the derivation order space of solutions, overcoming the left-most expansion bias exhibited by GE[6].

While there have been many papers dealing with πGE[2,3,5,11], this paper presents the first in depth look into the expansion order in πGE. What orders are actually explored? How does the order of πGE change over a run? Does the algorithm evolve towards a certain order? To answer these questions a metric must be used to determine the distance from a known order. The Order Bias Distance Metric is proposed for this task and then used to examine how πGE behaves. As well as investigating the order behaviour, the first steps are taken to quantify the cost of this order overhead to the search compared with GE.

K. Krawiec et al. (Eds.): EuroGP 2013, LNCS 7831, pp. 37–48, 2013.

Another important aspect of πGE is to explore how does a more complex representation effect the connectivity of the algorithm. Does the addition of position independent expansion order provide πGE with a more connected solution space, thus is it easier or harder for πGE to move about the solution space than it is for GE. Previous work by Murphy et al.[9] investigated the connectivity of TAGE compared to that of GE. It was found that the richer representation of TAGE provided it with a much more connected phenotypic space. This is of interest as both TAGE and πGE share a very similar evolution controlled position independent mapping. It has been shown that visualising the program space can be useful in understanding how an algorithm works[7], and through the processes outlined in Murphy's work it is hoped that further understanding of πGE may be gained.

The remainder of the paper is structured as follows. An overview of GE is provided in Section 2, before examining the differences between the GE and πGE genotype-phenotype mappings in Section 3. The new distance metric used in this paper is outlined in Section 4. The results are outlined and explained in Section 5, firstly the order experiments in Section 5.1 followed by the connectivity experiments in Section 6. This is followed by a discussion in Section 6 and finally some conclusions and future work are outlined in Section 7.

2 Grammatical Evolution

Grammatical Evolution(GE)[2,13], is a grammar based form of Genetic Programming(GP)[8]. Whilst GP[14] relies upon the constructing of expression trees, and performing operations on the expression trees, GE takes inspiration from DNA-Protein mapping in its approach to the generation of solutions. GE relies upon the use of a list of integers referred to as a chromosome, or genotype. This chromosome is then mapped to a phenotype, or solution, through the application of a grammar to the chromosome as described in detail in Section 3.

O'Neill[10] 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.

3 Genotype-Phenotype Maps - GE, πGE

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. 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
```

Fig. 1. Example Grammar and Chromosome

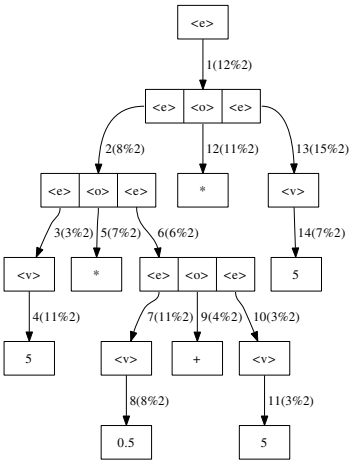

Fig. 2. Standard GE Genotype to Phenotype Mapping

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

```
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

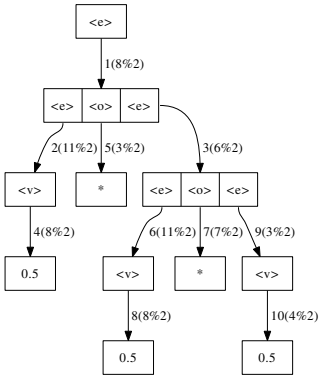

Fig. 4. Standard πGE Genotype to Phenotype Mapping

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.

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

4 Order Bias Distance Metric

Order Bias Distance Metric (OBDM) is a measure that shows how far away from a desired derivation order a πGE order is. The metric is measured in terms of the average percentage distance away from the desired order. The metric is very dependent on the πGE algorithms implementation. In πGE, all non terminals are added to a list of possible expansion sites and selection from this list is controlled by the chromosome. When a non terminal is expanded any non terminals generated from the expansion are then placed in the list in the position the parent NT was taken from.

Considering this, it needs to be determined what, if any, orders can an explicit distance from be calculated. Due to the variable length of the list of NTs, selecting a codon value that can always select the correct position in the list means that the only orders that are allowed for comparison to πGE are orders that are constructed by selecting the first NT in the list. As the πGE expansion rule, Eq. 2, can only be set to consistently select the first item in the list and no other position, only orders that rely on using zero codon values for the order codons and select the first codon in the list can be measured.

With this knowledge it can be determined that from the default implementation of the algorithm the only order that can be initialised to and a distance measured from using OBDM is Left-Most Depth First, also known as the standard GE mapping. To ascertain how far from the desired order the current order is, the position selected by Eq. 2, NT Choice, at each step of the πGE derivation

is converted into a percentage by using, $(100/|NT\ list|) * NT\ Choice$. The idea is that the desired order, or 0% distance, is always position zero in the list and then 100% distance would be selecting the last item in the list. The distance at each expansion is noted and at the end averaged to provide the percentage distance from the desired order for each individual.

4.1 Alternative Orders

To initialise the initial πGE population to any other desired order requires fundamental changes to the mapping algorithm. *Right Most First Order* can be achieved if the NT list order is reversed so that when the first element is selected in the list it is always the rightmost non terminal. *Breadth First Mapping* can be achieved by appending the new non terminals to the end of the list and then always selecting the first item in the list, allowing the algorithm to process all NT's at the first depth of the tree before moving onto the lower levels. *Breadth First Right Most* can be achieved by reversing the Breath First list above.

The monitoring of any other order becomes far too computationally intensive if a method other than an OBDM style of measurement is used. A metric for any type of non fixed expansion order would have to store all possible outcomes of all possible trees and then see how far away from the original tree the resulting tree was. The branching factor that πGE's order brings makes this task exponentially increasing in difficulty. On a test run trying to monitor all possible valid πGE trees, to a chromosome length of twelve with a simple symbolic regression grammar, the algorithm was using in excess of 20GB of RAM and substantially increased runtime in the order of several hours. OBDM provides a metric that requires zero extra online monitoring and doesn't slow down the algorithm.

5 Results

In this section the result for this study are reported regarding order and connectivity in πGE. Firstly the results for the experiment relating to the understanding of how order works and behaves in πGE is examined. This is then followed by the reporting of the other facet of this study, how does this added order change the connectivity of the πGE representation when compared to GE. For all experiments reported here GEVA v2.0[12] was used and modified as needed to produce the required output.

5.1 Order and πGE

To ascertain what is happening to the πGE expansion order during evolution, a method of recording the expansion process is needed. For this it was decided to store the NT list choice that was taken to first select the parent NT for expansion and the list length when this was taken as well as the tree depth of the parent in every child node. Once this was done the parsing of the data was done and represented using the OBDM above. At each expansion the distance

from standard GE order was calculated and then compressed and represented on a population level per generation.

The general setting for the experiments are displayed in Table 1. There were four setups examined of varying population size and generation length. These setups were then applied to three problem domains, *Santa Fe Ant Trail*, *Even 5 Parity* and *Symbolic Regression*. The experiments were then repeated using a fixed order initialisation, setting every expansion codon to zero so as to guarantee a standard GE order, and then examined to see how the order would change starting from a fixed order. Would it maintain the order or something close to it, or would it follow the behaviour of standard πGE?

Table 1. Parameter settings adopted for the order experiments

Parameter	**Value**
Setup A	100 Generations 100 Population
Setup B	400 Generations 100 Population
Setup C	100 Generations 400 Population
Setup D	400 Generations 400 Population
Replacement strategy	Generational with elitism (10%)
Selection	Tournament size=2
Mutation probability	0.01 (integer mutation)
Crossover probability	0.0 & 0.9 (variable single point)
Initial chromosome length	200 codons (random init)
Runs	100 per setup & problem

In Fig. 7 the results for Setup D are displayed on the *Even 5 Parity Problem*. Results for other setups and problems where omitted due to space constraints. By examining the figure it can be seen that πGE starts off with a large amount of individuals that have a very GE like mapping order. This anomaly comes from the fact that πGE and GE generate a lot of small individuals at the start of a randomly initialised run and it can be seen that these individuals are greatly reduced after 100 generations. This trend was seen across all setups and problems. Examining Fig. 7, focusing on the left hand side of the figure it shows how the order of the πGE population changes during the run. In fact by the end of the 400 generations it looks like a slightly offset normal distribution of orders is seen. Examining the right hand side of the figure the order of the fixed initialiser is shown. The population starts off with a GE order of expansion but over time this order moves to be more like the order seen in πGE. These findings were seen across all setups and problems. One thing of note was that with a reduced amount of generations the populations drift away from the GE order was reduced but there was no way to stop the drift. The stopping of this drift towards a πGE order is discussed further in Section 6.

5.2 Connectivity and πGE

To fully understand an algorithm it is helpful to visualise the connectivity of the phenotypic space associated with the algorithm. In this experiment the aim is to try and represent a single mutation event in the πGE genotypic space and map the resulting move in the phenotypic space. Through this it is hoped to gain an understanding of how the added search that the evolvable order in πGE causes can lead to results on a par with and in many cases better than GE.

In this experiment GEVA was extended to incorporate the Mutate and Store operation as described in detail by Murphy[9]. Mutate and Store basically starts off with an all zero chromosome and then iterates along the chromosome finding all valid chromosomes and storing them in a neighbourhood. The operator then calls these neighbours and does the same process finding all valid genotypes. This continues until all valid genotypes have been mapped and explored. Once this process is done all the individual neighbourhoods are compressed into a single neighbourhood. The operator removes all degeneracy in the genotypes by only allowing the codon values at each point of the chromosome to represent the choices available thus removing the neutral mutations that GE can take advantage of. For example a GE codon valued 62 is mutated to 64 and this codon is applied to a binary grammar rule, the mutation results in no change to the expansion of the tree. The grammar used for this experiment is similar to the one shown in Fig. 3 except now $< o >::= + \mid -$ and $< v >::= x0 \mid x1 \mid 1.0$. Mutate and Store was run on GE and πGE and was setup in such a way as to limit both algorithms to the same phenotype space.

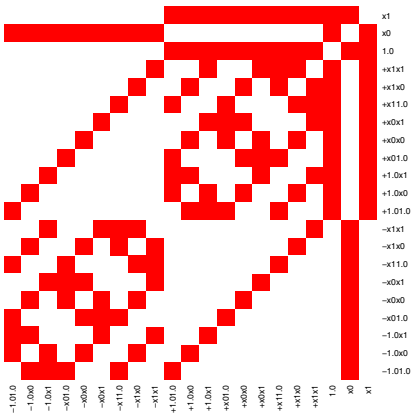

Fig. 5. GE Adjacency Matrix. The x-axis and y-axis are the same and display the phenotypes attainable from the available chromosome length

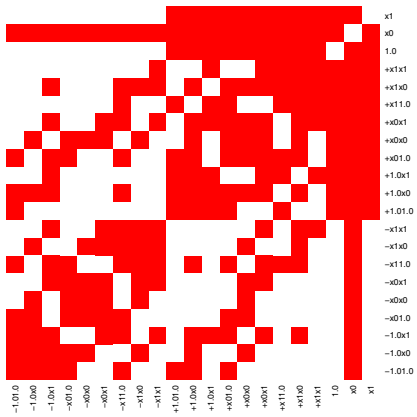

Fig. 6. πGE Adjacency Matrix. The x-axis and y-axis are the same and display the phenotypes attainable from the available chromosome length.

The first examination of the connectivity of πGE versus GE was performed by converting the connections to a graph and representing the graph as an adjacency matrix or connectivity map as in the Murphy study. Adjacency matrices are good for showing how connected the valid phenotypes are. An algorithm whose phenotype space has a densely populated adjacency matrix will have an easier time moving from phenotype to phenotype and thus it can more easily search the space. Fig. 5 shows the adjacency matrix for GE and Fig. 6 the adjacency matrix for πGE. It is obvious when the two figures are compared that πGE's phenotype space is more densely connected than GE's phenotype space, also worth noting is how GE has no neutral mutation but with the addition of order πGE exhibits neutral mutation. A phenotype of $x1$ cannot exhibit neutral mutation as the NT list for such a tree never exceeds a size of one thus the left-most nonterminal is always picked.

The adjacency matrix representation is good for quickly showing connectivity but it lacks the ability to show multiple connections between the same phenotypes. By examining Fig. 8 and 9 and the actual graph of the neighbourhood for both algorithms it becomes very clear how connected both are. From these figures a couple of interesting things can be seen. Firstly we can see the density of the connections is far more in πGE, also the neutral mutation are clearly displayed. Finally it is also clear that there are multiple edges between the same vertices. These edges are distinct in that they represent clearly different ways to make the same transition, this feature is not seen in GE's graph. A more detailed comparison is done in Table 2 where it is shown that πGE has a much larger total graph degree, the amount of connections to the vertices in the graph, as well as more edges and that every vertex in the graph has a degree, the number of connections coming from a vertex, on average double that of GE.

Table 2. Table outlining features of the connectivity graphs shown in both Fig. 5 and Fig. 6. Of note is the more than double increase in connections for πGE.

Graph Features	GE Graph - Fig. 5	πGE Graph - Fig. 6
$\sum_{i=0}^{n} Vertex\ Degree$	98	198
$\sum_{i=0}^{n} Edges$	49	99
$\sum_{i=0}^{n} Vertices$	21	21
$\overline{VertexDegree}$	4.67	9.43

6 Discussion - Restricting Order Drift in πGE

It has been shown in the experimental section of this paper that with πGE, evolution does not evolve towards a specific mapping order. πGE instead evolves to a population of individuals with a distribution of mappings orders. However is there a way to limit this drift and force πGE to maintain a mapping order?

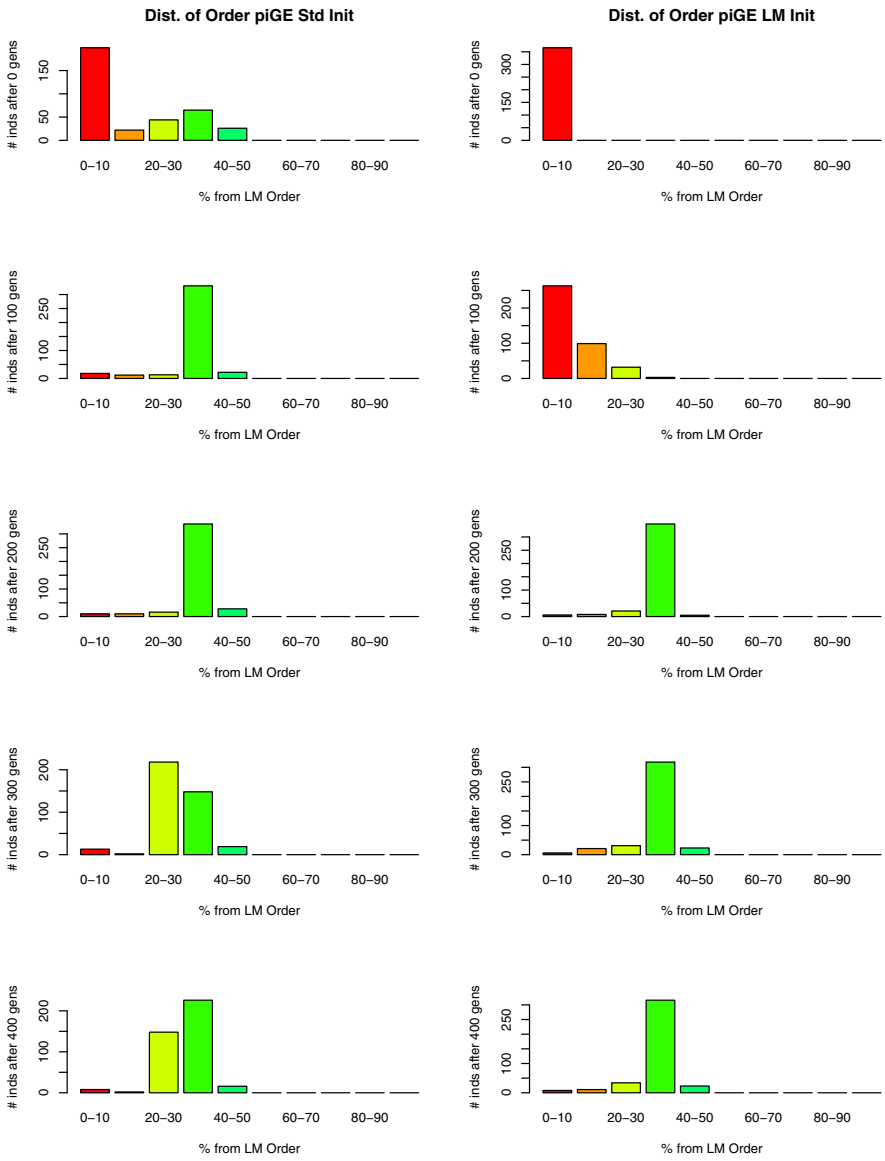

Fig. 7. Figure displays how the order of πGE varies during evolution using the OBDM. For each sub graph the x-axis shows the distance from the fixed order, while the y-axis shows the number of individuals

In previous work[4], a mutation operation was proposed that could focus on order codons or content codons of a πGE chromosome and the algorithm could be setup in such a way as to turn off mutation of the order codons completely. Upon further inspection of how the mapping process works for πGE, even if the

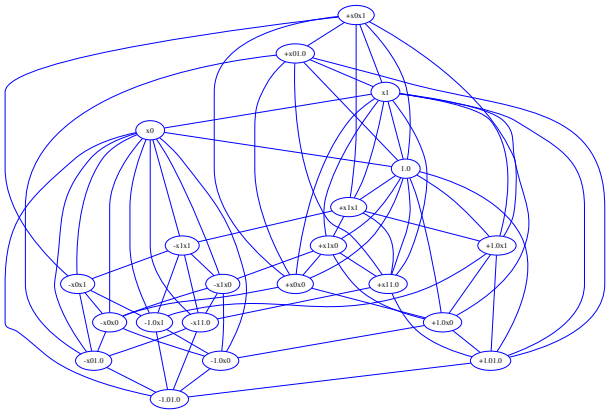

Fig. 8. GE - Single Mutation Graph

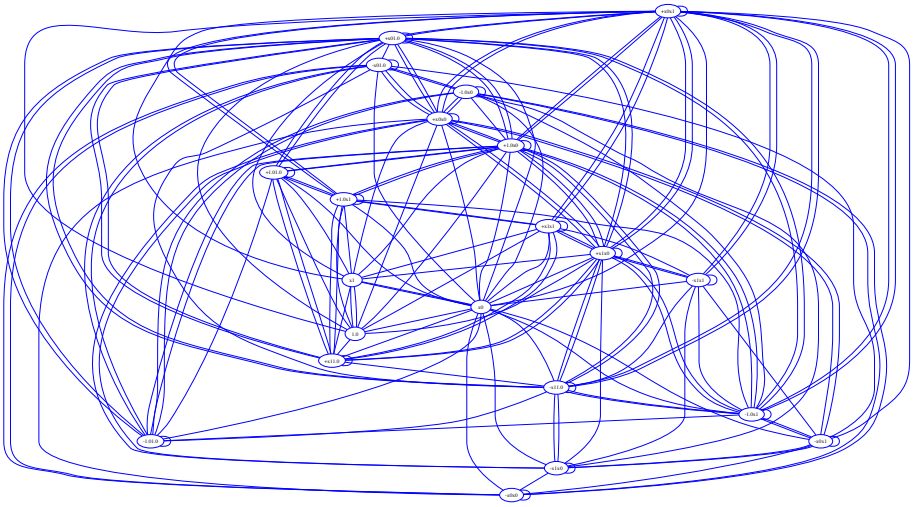

Fig. 9. πGE - Single Mutation Graph

mutation of order codons is not allowed the order of the individual will change with the mutation of the content codons. In πGE the order of the individual is linked to not only the order codon but also the number of NT's left to be expanded. So if a content mutation changes the number of NT's in the list then this may change the expansion order that follows from that point on. This is a similar ripple effect to that noted in standard GE mutation[1], but in this case the ripple is caused by the change in the number of NT's to be expanded.

Consider the following example, there is a section of chromosome and the algorithm is currently pointed at the codon with the value 5, *Chromosome* : $[3, \underline{5}, 9, 7, 8]$, and a current NT list mid run: $NT's = [e, o, e, o, v]$. Applying the πGE order rule, $5\%5 = 0$, leads to the mapper selecting NT zero in the list to

expand, $NT's = [\underline{e}, o, e, o, v]$. Applying the GE expansion rule, $9\%2 = 0$, results in this e being replaced by v and sets the NT list for the next expansion in the derivation tree, $NT's = [v, o, e, o, v]$. Next the mapper selects index 2, $7\%5 = 2$, and continues on from there. However if the codon valued 9, that controls what the first e expanded to, is mutated to 4 the list now looks drastically different, $NT's = [\underline{e, o, e}, o, e, o, v]$ and so when we apply the πGE NT selection equation to choose the next codon, $7\%7 = 0$, the NT at position zero is now selected and thus the ripple is started and all the following order choices will be effected.

In general it can be shown that if a content codon is mutated and this mutation results in the number of NT's available for expansion being changed then a resulting ripple will change the order of expansion for πGE.

7 Conclusions

The main aim of this paper was to further investigate what goes on within πGE with regards to the expansion orders used in the algorithm. The orders of πGE individuals during evolution was recorded, from a random order initialisation and a fixed order initialisation, on a range of setups and problems. It was shown that πGE drifts towards a distribution of orders rather than one particular order, this exhibits behaviour similar to that of crossover in GP whereby crossover causes evolution to a distribution of tree sizes. However the drift away from a fixed order can be reduced if a shorter number of generations are used during evolution. The monitoring of other orders was also discussed and some other fixed order initialisations for πGE were discussed, but they would require fundamental changes to the algorithm due to the sensitivity of the order to the size of the NT list. Finally the idea of trying to constrain the order was discussed but again the sensitivity to the NT list size makes this a computationally prohibitive idea.

Given the search overhead the order gives to πGE it was decided to investigate if the order added anything to πGE and try to gain further understanding into how πGE works. By creating graphs of the neighbourhood of single mutation events in GE and πGE it was shown that with the addition of order a significant increase in connectivity was seen. A more densely connected algorithm has the benefit of easier movement within the search space. The order also added pure degeneracy and neutral mutation unlike GE that relies upon codons to provide this, while πGE benefits from both GE's neutral mutations and the ones it gains from the use of variable order. In conclusion it can be said that the overhead of the added search space could represent a problem for πGE to search the solution space, but the increased connectivity could be said to counteract this.

In the future, further examination of the order with πGE is desirable focusing on the behaviour of the elites in the population. It would be good to see how big an impact mutation and crossover would have on fitness and order. The connectivity of πGE presented in this paper while clearly greater than GE presents some interesting ideas. Firstly a more efficient way to explore the connectivity needs to be investigated to allow for much larger phenotype space. Finally repeating this study using crossover may provide more insight into πGE.

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

References

1. Byrne, J.: Approaches to Evolutionary Architectural Design Exploration Using Grammatical Evolution. Ph.D. thesis, University College Dublin, Ireland (2012)
2. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments. SCI, vol. 194. Springer, Heidelberg (2009)
3. 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)
4. Fagan, D., Nicolau, M., O'Neill, M., Galvan-Lopez, E., Brabazon, A., McGarraghy, S.: Investigating mapping order in piGE. In: WCCI 2010, July 18-23, pp. 3058–3064. IEEE Press, Barcelona (2010)
5. Galvan-Lopez, E., Fagan, D., Murphy, E., Swafford, J.M., Agapitos, A., O'Neill, M., Brabazon, A.: Comparing the performance of the evolvable pigrammatical evolution genotype-phenotype map to grammatical evolution in the dynamic ms. pacman environment. In: WCCI 2010, pp. 1587–1594. IEEE Press, Barcelona (2010)
6. Hemberg, E.: An Exploration of Grammars in Grammatical Evolution. Ph.D. thesis, University College Dublin, Ireland (September 17, 2010)
7. Koza, J.R., Poli, R.: Genetic programming. In: Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, ch. 5, pp. 127–164. Springer (2005)
8. McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'Neill, M.: Grammar-based genetic programming: a survey. Genetic Programming and Evolvable Machines 11(3/4), 365–396 (2010), Tenth Anniversary Issue: Progress in Genetic Programming and Evolvable Machines
9. Murphy, E., O'Neill, M., Brabazon, A.: Examining Mutation Landscapes in Grammar Based Genetic Programming. In: Silva, S., Foster, J.A., Nicolau, M., Machado, P., Giacobini, M. (eds.) EuroGP 2011. LNCS, vol. 6621, pp. 130–141. Springer, Heidelberg (2011)
10. O'Neill, M.: Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution. Ph.D. thesis, University Of Limerick (2001)
11. O'Neill, M., Brabazon, A., Nicolau, M., Garraghy, S.M., Keenan, P.: πGrammatical Evolution. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 617–629. Springer, Heidelberg (2004)
12. O'Neill, M., Hemberg, E., Gilligan, C., Bartley, E., McDermott, J., Brabazon, A.: GEVA: Grammatical Evolution in Java. SIGEVOlution 3(2) (2008)
13. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. Genetic programming. Kluwer (2003)
14. Poli, R., Langdon, W.B., McPhee, N.F.: 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)