Analysing the Genotype-Phenotype Map in Grammatical Evolution

David Fagan

B.Sc. University College Dublin

Thesis submitted to University College Dublin

for the degree of Ph.D.

at the School of Computer Science and Informatics

College of Science

Research Supervisors:

Prof. Michael O'Neill

Dr. Seán McGarraghy

External Examiner:

Dr. Anikó Ekárt

October 30, 2013

Abstract

The Genotype-Phenotype Map (GPM) is an important aspect of the representation in Evolutionary Computing (EC). The GPM decouples the search space of the EC algorithm into a many-to-one mapping, allowing an abstraction of the search and solution spaces, which can bring a number of benefits to search. Grammatical Evolution (GE) is a grammar based form of Genetic Programming (GP) that incorporates a GPM at its core, which is loosely inspired by nature.

This thesis investigates whether different approaches to the GPM can have a positive effect on GE's performance. By examining a range of GPMs that use differing expansion order principles it was found the one approach, Position Independent Grammatical Evolution (πGE) presented a viable alternative to the canonical GE GPM.

πGE, while showing good performance, uses a variable expansion order controlled by evolution. This variable ordering increases the size of the search space that must be navigated by πGE during evolution. It is found that πGE gains a significant increase in connectivity by using an evolvable order, while also providing πGE with additional neutrality.

Knowing what orders πGE uses during evolution may provide insight into new GPM approaches. With this in mind a set of measures are devised, that allow for the monitoring of πGE's population during an evolutionary run. What is found is that πGE doesn't converge to a single order but rather a distribution of GPM orders.

The addition of the evolvable order in πGE provides an added degree of freedom in the mapping that is not exploited by standard genetic operations. A mutation operation is presented that will allow the algorithm to focus mutation on certain aspects of the πGE chromosome. It is found that with this ability the performance of πGE is increased.

For Mam and Dad

In Memory

Vincent Fagan

1954 - 2013

Acknowledgements

Firstly I would like to thank my supervisors, Prof. Michael O'Neil and Dr. Seán McGarraghy. Michael saw something in me, and decided I was worth investing time and money in. I am forever indebted to you for all the support, encouragement, and supervision you have given me. I couldn't have asked for a better supervisor, you knew when to push me and when to just let me figure things out by myself. Seán you were always there to bounce ideas off, and get hands on as I tried to muddle my way through certain mathematical challenges. I must also thank Prof. Anthony Brabazon, while not my supervisor you always showed interest in my work, providing invaluable advice and encouragement.

I want to thank all the members, past and present, of the Natural Computing Research and Applications group for the support, guidance, and welcoming environment they provided. Dr. James McDermott, Dr. Miguel Nicolau, Dr. Erik Hemberg, and Dr. Alexandros Agapitos, as post-docs you all provided me with a sounding board for my ideas and I am grateful for the feedback and friendship that has come of it. Special thanks must also be given to Dr. Jonathan Byrne and Dr. John Mark Swafford. We were in the trenches together doing our PhDs. You welcomed me into the group with open arms and I am very grateful for the friends you have become.

I honestly don't think I would have enjoyed my time in research as much if it was not for Dr. Eoin Murphy. We have been through a lot together, BSc and PhD, and remained good friends throughout. I could not be happier knowing that we will graduate together.

I do not like for my work life to interfere with my private life, mainly so I don't have to explain to friends what I do, but also so that I can escape and unwind when not in work. So I wish to thanks all my friends (you know who you are) for doing nothing. I know your help and support was always offered but I like to not burden you guys with having to fain interest in my research. The time we spent just sitting around having a laugh was exactly what I needed.

My family plays an important part in my life. Kevin, Laura, Fiona, Rachel, Sarah, Keith, and Emma; thank you, I could not have done this without your love and support. I would not change a thing about the cloud of madness we exist in as a family.

Most importantly I must thank my parents. Everything I do and achieve in life is to try an make you guys proud. The love and support you have shown me throughout my life has made me into the man I am today. Without your presence I am certain I would not be here today with this PhD thesis. Mam you pestered me about not wasting my intelligence after my first unsuccessful attempt at this college lark. My returning to education to do my undergraduate degree was a direct result of this. Without that I would never have achieved what has followed that decision. I can't express in words how much I love you Mam, and how grateful I am to have you as a mother.

Tragically my father passed away the week after my viva. Dad I love you and miss you immensely. I know it brought you so much happiness the day I passed my viva. Dad, I am forever grateful for everything you did for me. You provided me with whatever you possibly could, and tried to nurture any interest I had. I am deeply saddened that you won't be around for my graduation, but I know you will be there with me in spirit.

Finally, I would like to thank Science Foundation Ireland (SFI). This research is based upon works supported by the SFI under Grant No. 08/IN.1/I1868.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Publications Arising

1. David Fagan, Michael O'Neill, Edgar Galván-López, Anthony Brabazon, and Séan McGarraghy. An Analysis of Genotype-Phenotype Maps in Grammatical Evolution. In Anna Isabel Esparcia-Alcazar, Aniko Ekárt, Sara Silva, Stephen Dignum, and A. Sima Uyar, editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 62–73, Istanbul, 7-9 April 2010. Springer

2. David Fagan, Miguel Nicolau, Michael O'Neill, Edgar Galván-López, Anthony Brabazon, and Séan McGarraghy. Investigating Mapping Order in πGE. In *2010 IEEE World Congress on Computational Intelligence*, pages 3058–3064, Barcelona, Spain, 18-23 July 2010. IEEE Computational Intelligence Society, IEEE Press

3. Edgar Galván-López, David Fagan, Eoin Murphy, John Mark Swafford, Alexandros Agapitos, Michael O'Neill, and Anthony Brabazon. Comparing the Performance of the Evolvable PiGrammatical Evolution Genotype-Phenotype Map to Grammatical Evolution in the Dynamic Ms. Pac-Man Environment. In *2010 IEEE World Congress on Computational Intelligence*, pages 1587–1594, Barcelona, Spain, 18-23 July 2010. IEEE Computational Intelligence Society, IEEE Press

4. David Fagan, Miguel Nicolau, Erik Hemberg, Michael O'Neill, Anthony Brabazon, and Séan McGarraghy. Investigation of the Performance of Different Mapping Orders for GE on the Max Problem. In Sara Silva, James A. Foster, Miguel Nicolau, Mario Giacobini, and Penousal Machado, editors, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 286–297, Turin, Italy, 27-29 April 2011. Springer Verlag

5. David Fagan. Genotype-Phenotype Mapping in Dynamic Environments with Grammatical Evolution. In Miguel Nicolau, editor, *GECCO 2011 Graduate students workshop*, pages 783–786, Dublin, Ireland, 12-16 July 2011. ACM

6. David Fagan, Miguel Nicolau, Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Dynamic Ant: Introducing a New Benchmark for Genetic Programming in Dynamic Environments. In Natalio Krasnogor, Pier Luca Lanzi, Andries Engelbrecht, David Pelta, Carlos Gershenson, Giovanni Squillero, Alex Freitas, Marylyn Ritchie, Mike Preuss, Christian Gagne, Yew Soon Ong, Guenther Raidl, Marcus Gallager, Jose Lozano, Carlos Coello-Coello, Dario Landa Silva, Nikolaus Hansen, Silja Meyer-Nieberg, Jim Smith, Gus Eiben, Ester Bernado-Mansilla, Will Browne, Lee Spector, Tina Yu, Jeff Clune, Greg Hornby, Man-Leung Wong, Pierre Collet, Steve Gustafson, Jean-Paul Watson, Moshe Sipper, Simon Poulding, Gabriela Ochoa, Marc Schoenauer, Carsten Witt, and Anne Auger, editors, *GECCO '11: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 183–184, Dublin, Ireland, 12-16 July 2011. ACM

7. David Fagan, Miguel Nicolau, Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Dynamic ant: Introducing a new benchmark for genetic programming in dynamic environments. Technical Report UCD-CSI-2011-04, School of Computer Science and Informatics, University College Dublin, `http://www.csi.ucd.ie/biblio`, 2011

8. David Fagan, Erik Hemberg, Miguel Nicolau, Michael O'Neill, and Séan McGarraghy. Towards Adaptive Mutation in Grammatical Evolution. In Terry Soule, Anne Auger, Jason Moore, David Pelta, Christine Solnon, Mike Preuss, Alan Dorin, Yew-Soon Ong, Christian Blum, Dario Landa Silva, Frank Neumann, Tina Yu, Aniko Ekart, Wil Browne, Tim Kovacs, Man-Leung Wong, Clara Pizzuti, Jon Rowe, Tobias Friedrich, Giovanni Squillero, Nicolas Bredeche, Stephen Smith, Alison Motsinger-

Rei, Jose Lozano, Martin Pelikan, Silja Meyer-Nienber, Christian Igel, Greg Hornby, Rene Doursat, Steve Gustafson, Gustavo Olague, Shin Yoo, John Clark, Gabriela Ochoa, Gisele Pappa, Fernando Lobo, Daniel Tauritz, Jurgen Branke, and Kalyanmoy Deb, editors, *GECCO Companion '12: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, pages 1481–1482, Philadelphia, Pennsylvania, USA, 7-11 July 2012. ACM

9. David Fagan, Erik Hemberg, Michael O'Neill, and Séan McGarraghy. Fitness Reactive Mutation in Grammatical Evolution. In Radomil Matousek, editor, *18th International Conference on Soft Computing, MENDEL 2012*, pages 144–149, Brno, Czech Republic, 27-29 June 2012. Brno University of Technology

10. David Fagan, Erik Hemberg, Michael O'Neill, and Séan McGarraghy. Understanding Expansion Order and Phenotypic Connectivity in πGE. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu, editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 37–48, Vienna, Austria, 3-5 April 2013. Springer Verlag

Part I

Introduction and Literature Review

Chapter 1

Introduction

This thesis is set in the research field of Evolutionary Computing (EC), a field that takes inspiration from evolutionary principles and turns them into computational problem solving tools. Genetic Programming is one of the most prominent methods of EC. GP breeds populations of parse trees that are subjected to neo-Darwinian principles of evolution, leading to the evolution of fitter solutions to the desired problem over time.

The adoption of a genotype-phenotype map for Genetic Programming (GP) [91, 140] has demonstrated performance advantages over traditional tree-based GP [2, 9, 83, 90, 102]. One of the most popular grammar-based forms of GP [101], Grammatical Evolution (GE), adopts a genotype-phenotype map which has been argued to provide a number of advantages over standard GP [124]. The Genotype-Phenotype Map (GPM) provides GE with the ability to search both genotypic space and solution space in a many-to-one relationship, unlike traditional GP which has a one-to-one mapping. The many-to-one mapping allows for multiple solutions to have the same performance, but be structured differently. This feature allows for neutral search, which allows the Evolutionary Algorithm to search with zero impact on performance amongst the different variants of the same solution, and has been shown to allow GPM-based variants of GP to resist getting stuck at locally optimal

solutions [2]. Whilst these results are encouraging, it has yet to be established what effect the Genotype-Phenotype mapping has on an Evolutionary Algorithm (EA) such as GE. How does changing the order in which the mapping is done affect the EA? Can further inspiration be taken from biology, which first inspired the Genotype-Phenotype Map in GP, to improve the EA? Recently these topics have started to be tackled by myself and others [1, 26, 38, 39, 40, 42, 126], details of which are outlined in Chapter 2, but many more avenues of exploration remain, as the interpretation of mapping used by GE is simplistic and lacking in some of the desired advanced features of the GPM that exist in nature [3].

1.1 Genotype-Phenotype Map and EC

Evolutionary Computing (EC) is computing using evolutionary principles. In EC a population of possible solutions is generated. Each of these possible solutions is then evaluated and assigned a fitness value. At this point a host of so-called genetic operations are carried out on the population, based on neo-Darwinian principles of evolution, such as mutation, selection, crossover and elitism. These processes help move the population of solutions towards the current best solution found, whilst exploring other possible solutions generated through the application of variation operations to the population. The population is then evaluated and the operators applied again. This process continues until the optimal solution is found or a generation limit is reached.

There exists many forms of EC, amongst the most popular is Genetic Programming (GP) [91, 140]. Traditional GP, like many types of EC algorithms, uses a one-to-one mapping from search space to the solution space. This rigid mapping to the solution search space allows for fast location of optimal solutions, however shows a tendency to get stuck at local optima. The mapping also has another shortfall, in nature it is possible to arrive at the same solution two a problem but using different approaches, while with

traditional GP and its direct mapping this is not possible. To overcome this shortfall, inspiration was taken from the DNA to Protein Mapping model put forward by Crick [21], and the idea of a Genotype-Phenotype Mapping (GPM) was applied.

The GPM would split the search space into two parts, a Genotype search space and a Phenotype search space. This enables a many-to-one mapping from genotype (individual) to phenotype (solution) as is seen in nature. This many-to-one mapping brings with it some added benefits, as shown by Kimura [86]. The many-to-one mapping adds degeneracy, or redundancy, to the individuals of a population, and allows for neutral search within the genotypic search space. Neutral search consists of moving position within the genotypic search space, without affecting the fitness of the solution. Kimura argued that this was the core idea behind evolution, that this ability to search around current optimal values, without affecting fitness would lead to the discovery of a better solution if it existed. This theory was investigated in the context of EC by Banzhaf [2] in which it was also claimed that the use of a GPM leads to better diversity within the population. The GPM has been applied in a broad range of EA's, for example in [2, 9, 25, 41, 51, 66, 82, 83, 84, 85, 97, 100, 102, 103, 133, 157].

GE is the EA that I use to examine the GPM. The GPM used in GE is a simple abstraction first established by O'Neill and Ryan [133], and provides an appropriate environment to use for my research. GE has many desirable features such as its use of grammars which provide great expressive power [66, 124] and this will add to the complexity we can achieve within the GPM. The mapping in GE has been subject to some investigation already [38, 39, 40, 42, 126], and O'Neill [124] highlighted some of the benefits of a GPM for GP.

1.2 Aim of Thesis

The aim of this thesis is to perform an investigation into the GE GPM. Specifically this research focuses on two core mapping approaches. The traditional GE GPM, that uses a fixed order or mapping, and a position independent variant, πGE [126]. πGE leaves the order of expansion under the control of evolution. To aid in the investigation some ancillary mapping approaches are also examined, this provides a spectrum of approaches to the GPM. The intention of this thesis is to try and understand how the πGE GPM works, and from this can πGE be improved upon.

1.2.1 Research Questions

The core aim of this thesis is to explore how the addition of a position independent GPM affects GE's ability to search. To this end, a number of research questions are proposed.

Question 1: Do other GPMs exist for GE that provide comparable or better performance?

Question 2: How does πGE present good performance given the added search space of having an evolvable GPM order?

Question 3: What GPM orders is πGE actually using during evolution?

Question 4: Do genetic operations exist that may take advantage of these new GPMs?

1.3 Contributions

Many of the contributions provided by this thesis have been published. These publications are enumerated on page ix. The main contributions of this work are listed by order of appearance in this thesis here, noting what research questions they address:

Literature Review

An extensive review of Grammatical Evolution is presented in Chapter 2. The topics cover: investigations into the algorithm's behaviour, extensions to GE, application of GE, before a comprehensive list of implementations is finally compiled.

Suite of Genotype-Phenotype Maps (GPM) for GE

Several variants to the GE GPM are required for comparison of mapping approaches in this thesis. To this end a suite of GPMs for GE are detailed in Chapter 3 (Research Question 1).

Performance Comparison of Different Approaches to the GPM

Chapter 3 presents a detailed comparison of mapping approaches performed on a range of benchmark problems. The comparison is the foundation on which this thesis is based (Research Question 1).

Analysis of Genotype-Phenotype Map connectivity

Connectivity of a GPM relies heavily on the representation underpinning the approach. Chapter 4 provides an in-depth analysis of the two most promising GPMs from this thesis, examined in Chapter 3. Visualisation of the comparison was aided by the usage of graphs and adjacency matrices (Research Question 2).

Methods of Monitoring Expansion Order in πGE

πGE has a variable order of expansion in the genotype-phenotype map. Understanding what orders are being used by the population, or a subset of its individuals, can provide valuable insights into the algorithm. Chapter 5 presents two metrics that can measure distance from known mapping orders (Research Question 2).

Analysis of Population Order Dynamics

Some GPMs use a variable expansion order during the mapping from genotype to phenotype. The order dynamics of one such mapping, πGE, are explored over a set of benchmark problems in Chapter 5 (Research Question 3).

Identification and Analysis of an Advanced Mutation Operation

Through investigation of other variants of the GPM, it is possible to discover added degrees of freedom in the mapping approach. These degrees of freedom may be exploited to improve the performance of the GPM. Chapter 6 sees the realisation of one such operator, Focused Mutation. The operator's behaviour is analysed, and different setups of the operator are compared across a range of problems (Research Question 4).

Publication of the work presented in this thesis

Each of the experimental chapters in this thesis is based on a corresponding publication. These works, in order of publication, are listed here:

- David Fagan, Michael O'Neill, Edgar Galván-López, Anthony Brabazon, and Séan McGarraghy. An Analysis of Genotype-Phenotype Maps in Grammatical Evolution. In Anna Isabel Esparcia-Alcazar, Aniko Ekárt, Sara Silva, Stephen Dignum, and A. Sima Uyar, editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 62–73, Istanbul, 7-9 April 2010. Springer (Chapter 3),
- David Fagan, Miguel Nicolau, Michael O'Neill, Edgar Galván-López, Anthony Brabazon, and Séan McGarraghy. Investigating Mapping Order in πGE. In *2010 IEEE World Congress on Computational Intelligence*, pages 3058–3064, Barcelona, Spain, 18-23 July 2010. IEEE Computational Intelligence Society, IEEE Press (Chapter 6),

- David Fagan, Erik Hemberg, Michael O'Neill, and Séan McGarraghy. Understanding Expansion Order and Phenotypic Connectivity in πGE. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu, editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 37–48, Vienna, Austria, 3-5 April 2013. Springer Verlag (Chapter 4 and 5).

1.4 Limitations

The investigations in this thesis covered a broad range of topics. Owing to this, all avenues of exploration were not covered. Within the context of the genotype-phenotype maps chosen for exploration in Chapter 3, the GPMs chosen were selected so as to provide a spectrum of mapping orders. Similarly the thesis is restricted in scope to the canonical GE's context free grammar. Only the GPM that fit these criteria were selected from.

Throughout the thesis comparisons are made based parameters commonly used in the GE literature unless otherwise stated. No exhaustive parameter sweep was performed.

Finally, limitations based on available computational power have impacted the scope of investigation in this thesis, specifically Chapter 4, where the combinatorial explosion when dealing with πGE prevented a more complex network from being constructed.

1.5 Thesis Summary

The goal of this thesis is to investigate the genotype-phenotype map used in Grammatical Evolution. The investigation focuses on identifying different mapping orders for the GPM, and investigating interesting behaviours the differing approaches present. Through the work contained within this thesis insight into how these novel GPM behave during the

evolutionary process is gained, and methods to take advantage of new degrees of freedom in the GPM are investigated. The thesis consists of the chapters outlined below:

Chapter 1

The chapter introduces the main ideas behind the genotype-mapping process and its usage in EC. The chapter also sets out the aims, research questions, objectives, contributions, and limitations of the thesis.

Chapter 2

The chapter introduces Grammatical Evolution (GE), the main focus of the research in this thesis. The inner workings of the GE algorithm are described and a literature review of GE is provided.

Chapter 3

Chapter 3 marks the beginning of the experimental research in the thesis. Several variants of the GE genotype-phenotype mapping process are proposed, and their performance compared on a set of benchmark problems.

Chapter 4

Chapter 4 contains the first avenue of investigation into how πGE works. Having presented itself as a very competitive GPM, πGE's phenotypic connectivity is explored to understand how an algorithm with such a massive increase in search space complexity can maintain a good search performance level. This is done by visualising the relevant search spaces, and comparing with the connectivity of GE.

Chapter 5

The third experimental chapter tackles the task of exploring how the evolvable GPM ordering in πGE behaves during a run. A suite of metrics are introduced to help track the orders being evolved in relation to known fixed order mapping approaches.

Chapter 6

The final experimental chapter presents a new mutation operator for πGE. This operator is designed to exploit the added degree of freedom that πGE provides. A range of setups are compared to identify what performance gains can be discovered by using the new operator over the traditional GE mutation operator.

Chapter 7

The final chapter of this thesis presents a summary of each chapter in the thesis, and draws conclusions from the experimentation and analysis performed within. Finally it outlines a number of questions raised during the course of this research, and proposes future work to continue the research presented in the thesis.

We now go on to introduce, and review the current state of the art for Grammatical Evolution (GE) in Chapter 2.

Chapter 2

Grammatical Evolution

This chapter introduces Grammatical Evolution (GE) [25, 133], the Evolutionary Computing (EC) algorithm used to conduct the research in this thesis. GE is a very powerful and expressive form of grammar-based Genetic Programming (GP) [100]. The modular design of GE allows for easy manipulation of the algorithm, whilst the use of grammars allows for evolution of solutions to problems in any language that can be represented with a grammar [25, 133]. Section 2.1 provides the reader with an introduction to GE. The section outlines GE's unique features, and notes how GE differs from GP. Section 2.2 outlines the GE algorithm and control flow. Sections 2.3 and 2.4 provide a more in depth explanation of some of the ideas expressed in Section 2.2. Section 2.5 provides an overview of previous work carried out on GE, covering examinations and extensions of the algorithm, as well as applications of GE to real world problems, and concluding with a list of implementations of GE. Finally the chapter concludes with a brief summary section in Section 2.6.

2.1 Grammatical Evolution - Overview

GE uses principles of Neo-Darwinian evolution to evolve automatically generated solutions to problems. GE is commonly referred to as a grammar based form of Genetic Programming (GP). However this description can prove inaccurate once the inner workings of GE are examined. Genetic Algorithms (GA) [49] and Genetic Programming (GP) [89, 140] are two of the most popular forms of EC. A GA uses a population of bit string individuals, and evolves this population of strings until the desired solution is found. The bit strings of a GA can represent a multitude of solutions depending on the encoding used. GP on the other hand uses a population of parse tree individuals. At their core both algorithms share similar workings. GE takes inspiration from both these classic EC algorithms, while also harnessing the added power of grammars.

GE differs substantially from GP in many areas of the algorithm. GP uses a population of parse tree individuals and performs variation operations on these trees. GE, on the other hand, uses a population of variable length integer array chromosomes, that are evolved using GA style operations on the chromosome. It is worth noting that some advanced versions of GE [129] now feature GP style operations, that work directly on the derivation tree. These chromosomes are mapped to solutions via a biologically inspired mapping process, referred to as the Genotype-Phenotype Map (GPM). The GPM is modelled on the DNA to Protein transcription process, as shown in Figure 2.1. In the figure it can be seen how DNA changes during the transcription process. A similar transcription process can be seen in GE, and is governed by a Backus-Naur Form (BNF) context-free grammar specified for the problem. This mapping process is examined in more detail in Section 2.3.

Whereas GP has a one-to-one mapping between parse tree and solution (although it could be argued that the genotype and phenotype are essentially the same thing in canonical GP), GE gains a many-to-one mapping by adopting the genotype-phenotype

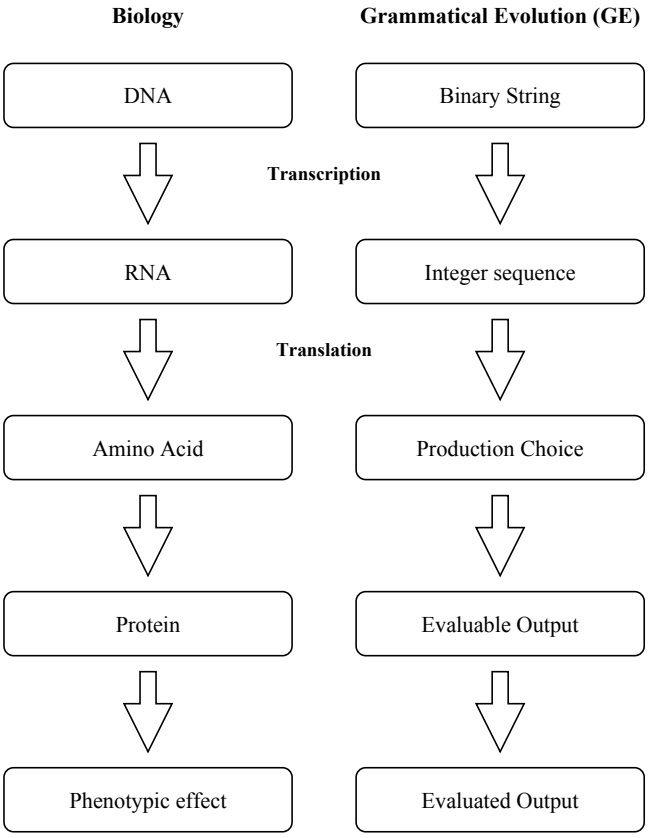


Figure 2.1: Comparison of the transcription of DNA to Proteins and the Genotype-Phenotype mapping process in GE.

map. In this many-to-one mapping many chromosomes will map to the same program. The usage of grammars in GE allows for the evolution of problems in a domain representable by a grammar, unlike canonical GP and its implicit grammar, where limitations such as the closure property have to be dealt with.

2.2 Grammatical Evolution - Algorithm

The main steps in the GE algorithm mirror those of a canonical GP or GA algorithm. The primary common ideas are to initialise a fixed sized population of individuals, and evaluate the population using some form of fitness measure. Once the population is evaluated, check if a stopping condition has been reached, if not then vary the population

and re-evaluate. The variation and re-evaluation of the population will be repeated until a stopping condition is met. This process is shown in Figure 2.2 and the terms explained below. A control flow algorithm is outlined in Algorithm 2.1.

Algorithm 2.1 The GE control flow algorithm - The algorithm mirrors that of many evolutionary algorithms. The population is defined, then initialised, and following this enters the evolution loop. In this loop the population is evaluated and varied repeatedly until a termination condition is reached.

```
Population = new population(pop_size)
Generations = num_gens
Solution_Found = false
Initialise(Population)
while Generations ≥ 0 do
  Evaluate(Population)
  if Solution_Found then
    Return Solution
    End
  else
    Perform_Variation_Operations(Population)
  Generations−−
Return Best Solution Found
```

1. **Initialisation** It is desirable to start with a population of dispersed individuals, so as to sample as many points in the search space as possible. There have been many ideas of how to do this, but the two main initialisation methods in GE are, Ramped Half and Half (RHH), and Random. These methods are explained in more detail in Section 2.4.1.

2. **Evaluation** Evaluation is where the individuals of the population are assigned a value or fitness. Individuals are candidate solutions to a problem. By measuring how well these solutions perform at a desired task, it is possible to assign them a value representative of how well they have performed the task. In GE this process is performed by a fitness function that returns a fitness value, indicative of how well the individual performs on the given test criteria. This is an integral step in all

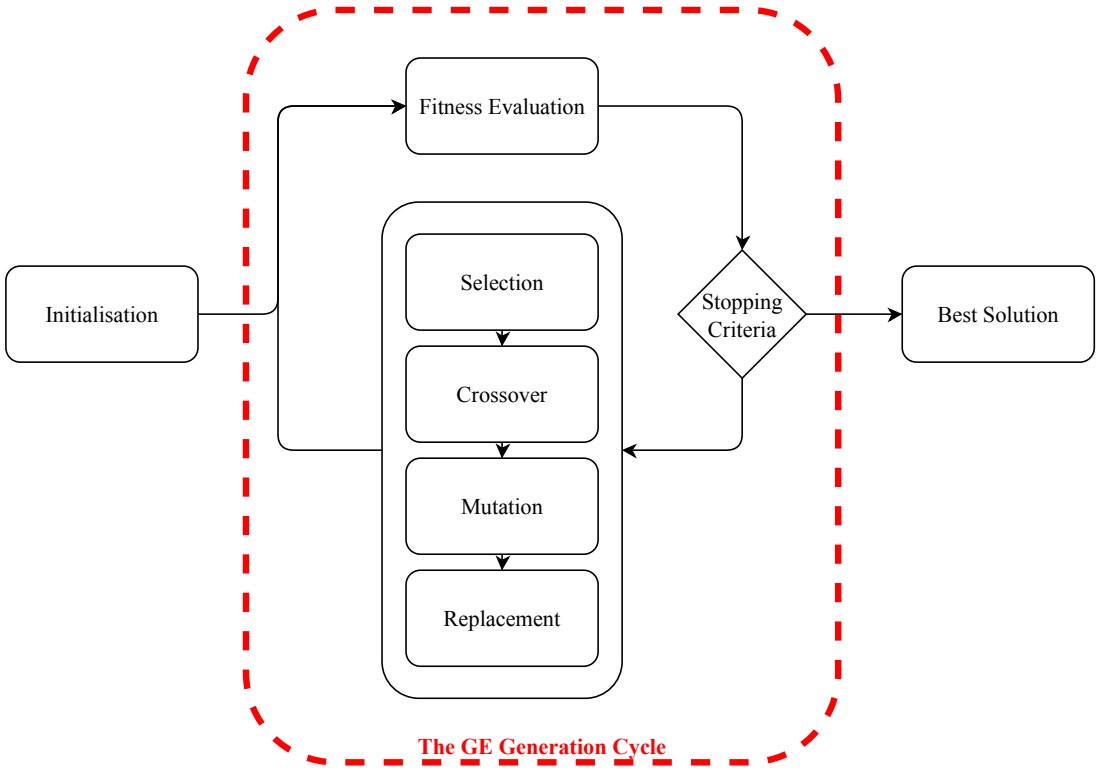


Figure 2.2: The diagram displays how GE works, starting by initialising the population, before entering the GE Generation Cycle, and exiting the algorithm once a stopping condition has been reached.

evolutionary algorithms as it provides the information used to rank the population for selection. This is of paramount importance to the performance of the other operators that rely on selection to function. Selection can be considered as the mechanism that drives evolutionary search. Selection is generally modelled upon the idea of survival of the fittest in nature.

3. **Termination** When to stop is an important part of the algorithm, as some domains provide problems that may never be solved. With this in mind there are two main ways to stop GE. Firstly, before the run starts a limit to the number of generations can be set. This provides an explicit limit to how long the algorithm will run. The desired outcome is that an acceptable solution is found before this limit is reached. Secondly, in GE the norm is to minimise fitness values such that the program will

halt once a fitness of zero is reached. In certain cases the minimum value may be changed to be a limit, such as in certain symbolic regression problems where a limit of 0.05 might be viewed as the acceptable fitness for a solution.

4. **Operators** The operators in GE all concern the variation of the population through variation, selection, and replacement operations. Firstly selection of individuals for variation is important and is discussed in Section 2.4.4. Once the individuals are selected the variation operators crossover (Section 2.4.2) and mutation (Section 2.4.3) are applied. Variation of the population will result in there being more individuals than our population can hold. To resolve this issue the population is passed to the replacement operator, which trims the population down to size, and is discussed in Section 2.4.5.

2.3 Mapping

The mapping process in GE is the conversion of a chromosome (genotype), to a solution (phenotype). In canonical GE this is done by taking a chromosome and a BNF context free grammar, and using the mod rule (Equation 2.1) a derivation tree is constructed. From this derivation tree the solution can be extracted for evaluation. This mapping process is outlined in Figure 2.3. The key aspect of the mapping in canonical GE is the construction of the derivation tree. What follows is an explanation of this process in more detail.

$$New\ Node\ =\ Codon\ value\ \%\ Number\ of\ rules\ for\ Non\ Terminal \tag{2.1}$$

2.3.1 Mapping Example

In GE the mapping process starts by identifying the start symbol in the grammar and using the mod rule (Equation 2.1), the derivation tree is constructed by expanding all non

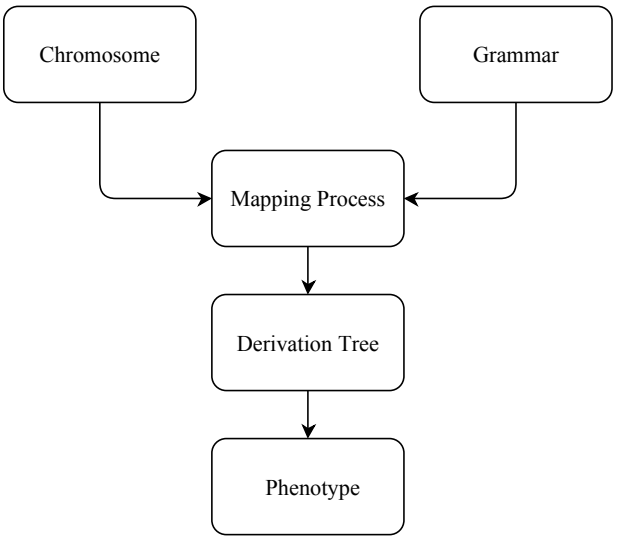


Figure 2.3: The diagram highlights the main idea behind mapping in GE. Taking a chromosome consisting of a list of integers, evolved similarly to a GA, and a grammar for input, the mapping process combines these two items using the mod rule (Equation 2.1), resulting in a phenotype or output being generated.

terminal symbols (NT) in a leftmost NT first expansion, until no more NT's remain. In the case of the grammar shown in Figure 2.4, which generates an arithmetic expression, that start symbol would be `<e>`. This NT is then evaluated using Eq. 2.1. By taking the first codon value of the GE chromosome 12, shown in Figure 2.4, and the number of expansions possible for the NT `<e>` 2, the first expansion of the tree results in `<e>` expanding to `<e><o><e>` as $12\%2 = 0$, as `<e><o><e>` is the 0^{th} choice for `<e>` (from the grammar shown in Figure 2.4). The process iterates from this point on, always expanding the leftmost NT in the derivation tree. This mapping process will continue until no NTs remain to be expanded in the derivation tree, or there are no codons remaining in the chromosome. Mapping termination is discussed in detail in Section 2.3.2. An example of this mapping is shown in Figure 2.5 based on the example grammar shown in Figure 2.4, 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 Equation 2.1.

```
<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 2.4: Example Grammar and Chromosome.

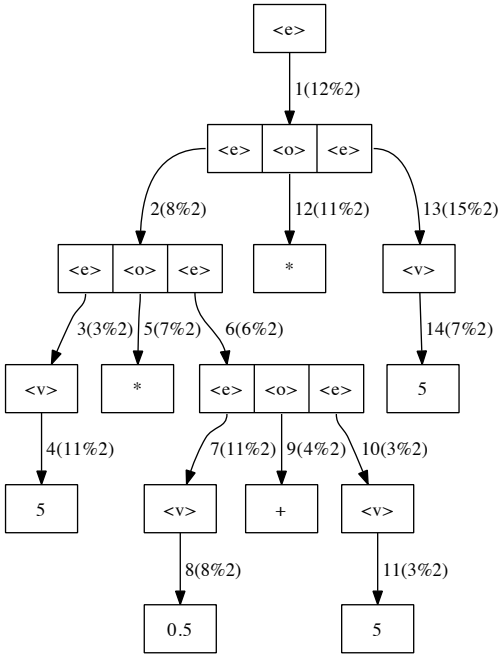


Figure 2.5: Standard GE Genotype to Phenotype Mapping, based on the example grammar shown in Figure 2.4, the order of expansion is indicated by the sets of numbers on the arrows between the blocks on the diagram, in the form of 1(12%2) where 1 is the expansion order and 12%2 is the application of Equation 2.1.

2.3.2 Mapping Termination

The mapping process in GE provides no guarantee that all chromosomes will lead to evaluable derivation trees. There are many factors that can lead to an incomplete mapping. Poor grammar design can lead to recursive derivation sequences that never finish, and a derivation sequence can use up all available codons in a chromosome resulting in an

incomplete mapping. Canonical GE has methods to try and alleviate this termination problem. Codon reuse, or wrapping as it is called in GE, allows for the mapping process to wrap back around to the start of the chromosome and reuse the chromosome again to try to complete mapping. There is however a user defined limit in place to prevent the wrapping from entering an infinite loop, and thus never completing the mapping process [115]. One drawback to wrapping is that a single codon mutation results in multiple mutations.

A second approach to aid with termination is that of chromosome tails. Chromosome tails consist of a certain percentage of random codons added by the initialiser, at the tail of the chromosome. While the tails may not be expressed at that specific point in time, during variation they may become expressed codons. Nicolau et al. [115] noted that the addition of tails to GE, provided a higher probability of generating terminating individuals. Finally, if after reaching the wrapping limit a chromosome still does not map to a complete tree, the chromosome is set as an invalid chromosome and will receive the worst fitness possible during evaluation. Nicolau et al. [115] performed an in-depth investigation into termination in GE, noting a link between poor grammar design and termination issues in GE, and also the benefits of tail usage.

2.4 Operators

Operations in evolutionary algorithms are the driving forces behind exploration and exploitation of the search space of the problem in question [49]. Initialisation strives to provide the initial population with a diverse sampling of possible individuals. Variation operations enable the algorithms to vary the population, thus exploring the search space. Selection and replacement operations, on the other hand, allow the algorithm to guide the population of solutions to perceived fitter locations in the search space to explore. Next is an overview of the most widely used Operations in Grammatical Evolution is given.

2.4.1 Initialisation

The quality of the population at the start of an evolutionary run can have a huge affect on the success of the run. If a population consists of individuals too small to represent the desired solution to a problem, the EA will fail to find valid solutions, unless a variation operation such as crossover has had time to increase the size of the individuals. Similarly if the population is located in too narrow a region of the solution space, then convergence to a local optimum can be increased significantly, and thus degrade performance. Harper [52, 56] has performed a very comprehensive investigation of initialisation in GE. Harper found that initialising to certain distributions of tree shape and size, proved beneficial to GE's performance. A brief introduction follow next, to the two main initialisation methods used in canonical GE.

Random Initialisation

Random Initialisation represents one of the most basic initialisation methods available. Before the run an initial length of chromosome is specified by the end user for the individuals. The initialiser then randomly generates chromosomes to that length and assigns them to the individuals. The strength of a random initialiser rests with the random number generator used for the initialisation. The number generator used should guarantee a uniform distribution of codon values from all possible values. While random initialisation provides a population of varied chromosomes, it has no consideration of the phenotype space or the size and shape of trees generated.

Ramped Half and Half Initialisation

While random initialisation can provide a good variety of chromosomes, it doesn't guarantee a widespread distribution of derivation trees. To address this, GE has borrowed from the GP literature and can use what is called Ramped Half and Half Initialisation

(RHH) [89]. RHH provides GE with a population consisting of a varied distribution of individuals with different sized and shaped derivation trees. To do this GE first partitions the population into two pools of individuals. Using a predetermined maximum and minimum depth, the algorithm proceeds to further divide each pool into pools of individuals with different depth limits, increasing up to the maximum depth (called ramping). For one half of the population these pools of individuals are created using the *Full* method of tree growth, where the derivation tree is constructed to be as fully expanded as possible, with all recursive branches reaching the depth limit (Figure 2.6). The other half of the population is then created using the *Grow* method of tree generation, where the derivation tree is randomly created with no constraint on expanding all branches to the maximum depth (Figure 2.7). The genotype, or chromosome, for each derivation tree is created during the construction of the derivation trees, in the full and grow methods. Using RHH results in a population with trees of a variety of sizes, shapes, and node densities. RHH also shows no consideration for the phenotype space in its initialisation.

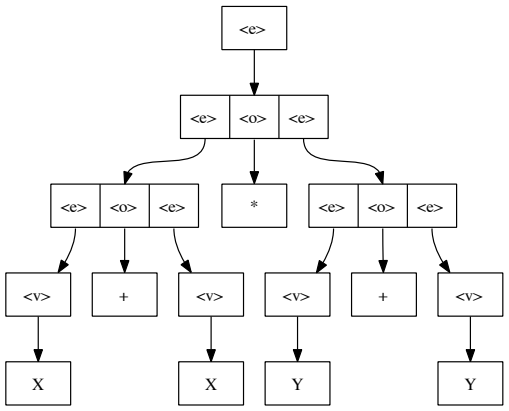


Figure 2.6: The full method of tree growth is displayed. Note that all recursive branches have reached the depth limit of 4, unlike the grow method in Figure 2.7

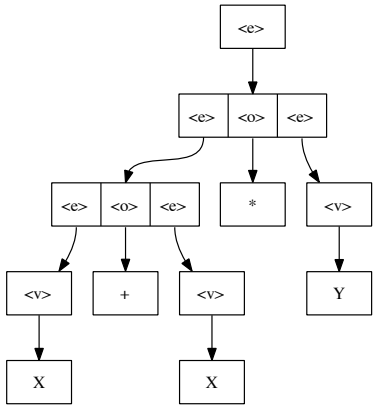


Figure 2.7: The grow method of tree growth is displayed. Note that not all recursive branches have reached the depth limit of 4, unlike the full method in Figure 2.6

2.4.2 Crossover

Crossover is the process of exchanging information between members of the population, with the intention that this exchange of information will result in fitter offspring. Crossover in GE can also be viewed as an operation that can vary the size of the chromosome. In traditional GP, which uses program trees, crossover is simply the exchanging of sub-trees. Two parents are selected for crossover from the population and then two exchangeable sub-trees are located in the respective programs trees. These trees are then exchanged and the individuals re-evaluated.

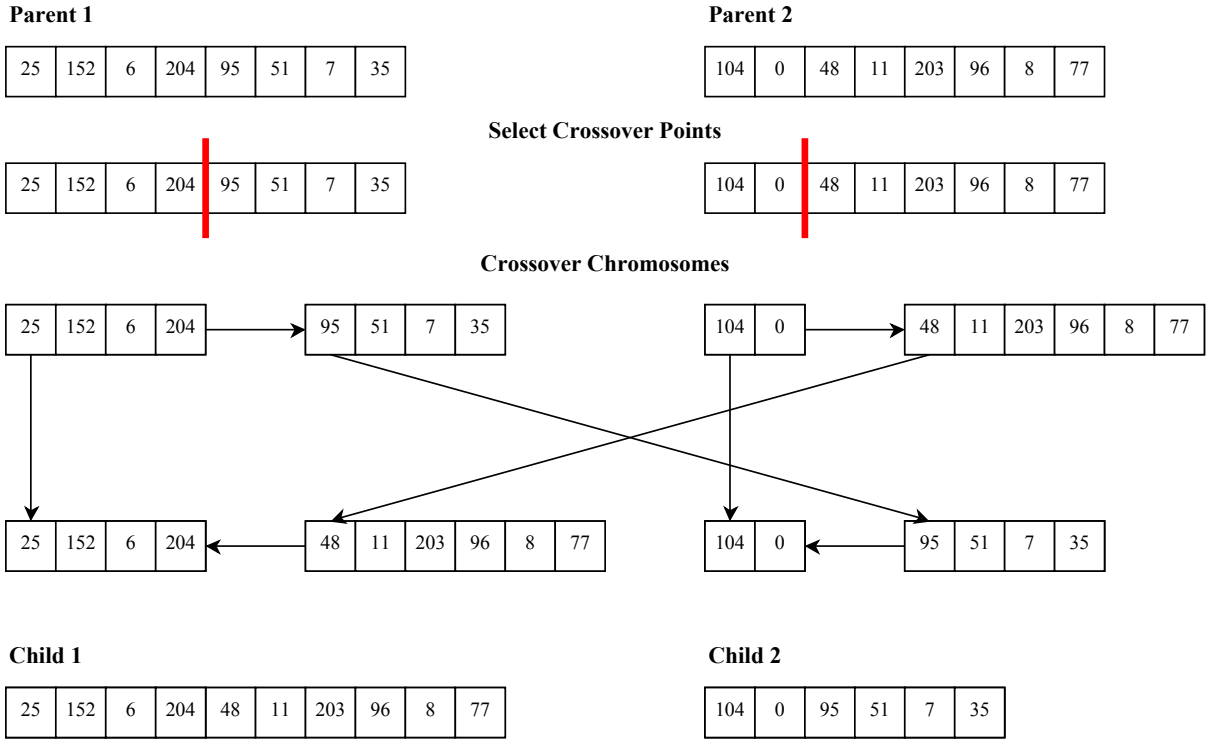


Figure 2.8: Single Point Crossover in GE where the selection point is not fixed, to allow for growth and shrinkage of the chromosome.

GE on the other hand performs all of its operations on the chromosome. Crossover in GE is handled in the same manner as in a GA. In GE two parents are selected from the population. Once each parent is selected, a point on the chromosome of each is chosen,

and then the chromosomes are split at these points and separated sections of chromosome are exchanged by the parents resulting in two new individuals, or children. This process is described visually in Figure 2.8. As GE uses a variable length chromosome the selection points for single point crossover can be allowed to vary in order to promote variation in the size of the chromosomes.

What has been described above is single variable point crossover. This method is the sole method of crossover used in the experimental sections of this thesis. Several other variants of crossover exist for GE [54, 55, 56, 81, 122, 134, 135]. These methods are briefly mentioned in Section 2.5.1.

2.4.3 Mutation

Mutation allows for slight variation of an individual, this variation intends to allow for individuals to take a small step towards the best solution [49]. In GP this process is done by selecting a node from the program tree and then replacing it with another node from the function or terminal set of nodes as required. This process can result in a single node being changed or at most the subtree that is rooted at the selected node will be changed from the resulting mutation.

Figure 2.9: Integer Mutation in GE

In GE mutation is again heavily influenced by GA. As GE uses a chromosome of integer values the main mutation type is the so-called Integer Mutation, where every codon in the chromosome changes to a new randomly generated integer, depending on the probability of mutation. This mutation probability is set by the end user before the start of a GE

run. An example of this can be seen in Figure 2.9. The codon highlighted has simply been changed, this change is mutation in GE. However unlike GP the scope for change in a single mutation is very large in GE. Due to the linear nature of the chromosome in GE if the mutation event changes the number of non terminal tree nodes, this will result in a different expression of all codons following the mutation site. This is known as the GE ripple effect [13, 15, 25, 66, 68, 69, 81, 133]. Integer mutation is the only method of mutation explored in the experimental sections of this thesis. However other methods of GE mutation exist and are briefly mentioned in Section 2.5.1.

2.4.4 Selection

Selection is the mechanism used by EC to select parents for reproduction. The selection of parents for offspring is a delicate matter that requires some thought. Selecting parents, in nature, from too similar a gene pool results in inbreeding and a weakening of the genetic pool, that can lead to problems such as deformities. In EC just selecting the fittest individuals for reproductions is not desirable as this leads to premature convergence and is not a very efficient way to search the genotypic space. While it is possible that two fit parents may produce a fitter offspring, it is also possible that a fit parent and a less fit parent will also produce a fitter offspring. The main reason for allowing imperfect selection, such as between a fit and unfit individual, is that it helps to maintain genetic diversity within the population. With this idea of imperfect selection in mind the following is a brief overview of two popular selection methods in EC, which are also the main selection methods used with GE.

Roulette Wheel Selection

Roulette Wheel Selection, or Fitness Proportionate Selection to use the more formal name, is a selection method whereby the relative fitness of the candidate individual in relation to the sum of the population's fitness values is used to derive a selection probability. Specifically an individual i has a probability of selection P_i, where $P_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$ and f_i represents the fitness of the individual i. By assigning each individual in the population a probability of selection, this probability conceptually represents a wedge on the roulette wheel. By spinning this conceptual wheel an individual is selected for variation, where individuals with the larger wedges on the wheel are more likely to be selected.

Tournament Selection

Tournament selection is a simpler and faster method of selection to that of roulette wheel selection, in that it does not require knowledge of the fitness of the whole population. In tournament selection a fixed number of individuals, referred to as the tournament size, are randomly selected from the population. From this tournament, or subset of the population, the fittest solution is selected for use. This process is repeated every time an individual is required to be selected. Tournament Selection has the ability to vary its selection pressure. Selecting a tournament size of one equates to just randomly selecting a member of the population, whilst a tournament size equal to the population size guarantees selection of the fittest individual in the population. Tournament selection is viewed as a good selection method as it prevents the best individual from immediately dominating the population causing premature convergence. Finally it is a tuneable and easy to implement method of selection [140].

2.4.5 Replacement

Applying variation operations to the population brings with it the issue of an expanding population. In EC the population is generally a fixed size. Due to this, a mechanism is needed to decide how to construct a new population from all the available parents and offspring, that fit within this size constraint. The two most common replacement methods used in GE are outlined below.

Steady State Replacement

The steady state replacement strategy is simply survival of the fittest. After each new individual is generated via the variation operations, it is immediately evaluated. If the new individual's fitness is better than the worst in the population, it is inserted into the population and the population's worst individual is deleted. This method does carry with it a very high selection pressure and can cause premature convergence to local optima. Originally GE used steady state replacement as a way to combat invalid individuals in the population [150].

Generational Replacement

In Generational Replacement the population is made up of new individuals every generation, with the children replacing the parents. This allows for more wide spread exploration of the search space as the majority of the children generated during variation are included in the new population. The new individuals may not be as fit as those in the previous generation and thus impede convergence. To counter the fact that the population's fitness values can worsen, the generational approach uses an elite population to maintain the best individuals that have been encountered during evolution, that may be used to improve the population. While generational replacement has a slower convergence rate than steady state replacement, the size of the elite population can vary this convergence rate greatly.

2.5 Advances in GE

GE has established itself as a widely used [166] EC method within a very diverse ecosystem of methodologies. Previous sections in this chapter have outlined how the algorithm works. In this section an overview of state of the art in GE is given. Firstly, previous research focused on examining the behaviour of GE is described in Section 2.5.1. Following this an overview of extensions to GE is given in Section 2.5.2. Finally a summary of applications and implementations of GE, in Sections 2.5.3 and 2.5.4 respectively, concludes this section.

2.5.1 Examinations

The examination of any EA is of interest as researchers strive to find out how a system works, so as to then improve its performance. GE is no different and has had many aspects of the algorithm fall under the scrutiny of researchers. The initial examination of GE was performed by O'Neill [124] and covered the canonical form of GE. Presented in this section is an overview of some of the other areas examined.

Crossover is one area where GE and GP differ significantly, and has fallen purview to in depth analysis. Works by O'Neill et al. [122, 134, 135] and Keijzer [81] have examined GE crossover and established the term Ripple Crossover. Ripple Crossover is similar to ripple mutation in that the crossover produces a change that effects everything that follow from the point of crossover. Ripple Crossover was found to provide a slower rate of convergence than canonical GP subtree crossover. Harper [54, 55] examined many crossover variants and established a non destructive method of GE crossover, and went on to further examine a self-selecting crossover mechanism, whereby the system itself can decide the best type of crossover to use.

Mutation is another area that has been examined in much detail. Murphy et al. [108] have examined and visualised the mutation landscape of canonical GE, and other gram-

matical variants of GE. Byrne et al.[13, 15] examined canonical GE mutation and identified two types of mutation, nodal and structural, before carrying out an extended examination using the two types of mutation. Hugosson et al. [68, 69] examined different chromosome representations and how they affect mutation, with the hopes of finding a smooth mutation operation. Fagan et al. [34, 33] examined the possibility of adding a self-adapting mutation parameter to GE, that reacts to the rate of change in the fitness of the population.

Initialisation methods also fell under scrutiny by Harper [52]. Harper found that poor initialisation can lead to a reduction in the performance of GE, and that poor grammar construction can lead to a serious explosion in tree sizes. In a similar vein, Murphy et al. [105] examined how different grammars and initialisation methods can produce significantly different results, and looked to match the distribution of individuals that the initialisers provided regardless of the grammar type. In related work Nicolau et al. [115] examined the issue of poor grammar construction and how it can significantly reduce the probability of successfully mapping a chromosome.

Grammars have been examined in some depth. Keijzer et al. [79] examined the usage of logic grammars. Murphy et al. [105, 107, 108, 109] explored the usage of tree-adjoining grammars that always guarantee a successful mapping from genotype to phenotype. Hemberg investigated probabilistic grammars and meta-grammars [66] that allowed for the grammars to evolve over time. Attribute grammars have been examined by Cleary [18, 19, 128] and more recently by Karim et al. [74, 75, 76, 77, 78].

Finally, some other research to note with respect to GE is that of Keijzer et al.'s examination of the GE mod rule and an alternate approach called the bucket rule [80]. O'Neill et al. [136] examined grammars, introns and bias in GE. O'Neill found that GE was susceptible to a bias towards certain rule productions. However allowing for a grammar defined intron to skip codons, or by changing the arity of the production rules in the grammar, this bias could be negated. Hemberg et al. [62] furthered this research by studying the

effect of grammars, implementing different arithmetic notation types and examining their effect on bias. Hemberg et al. [65] also performed an investigation into representing ADF's, Automatically Defined Functions, in GE in different ways. Hemberg [66] also presented a formal theory of disruption in GE. Hemberg investigated every type of change that could be made via GE's operators, and formally defined the disruption to the phenotype that occurred with each change. Swafford extensively examined the idea of modularity in GE [158, 159, 160, 161, 162]. O'Neill et al. [131] have also begun examining GE and its performance in dynamic environments.

2.5.2 Extensions

The modular design present in GE allows for easy manipulation and extension of the system. What follows is an overview of extensions of GE that reinforce the idea that GE is a very adaptable and expandable platform.

SEOIGE

Fenton et al. have built on the foundational work of the addition of shape grammars to GE by O'Neill et al. [130] and the doctoral studies of Byrne [11, 12], in interactive evolutionary design, to produce the Structural Engineering Optimisation in GE system. The system allows users to generate structures and use real world objectives such as minimising material costs, using certain material types, and maximising structural soundness of designs. There is also an interactive component to the design where the user is able to design a structure through an interactive GE process before handing the design off to be fine tuned by multi objective search criteria, such as the real world objective mentioned above.

PODI

Program Optimisation with Dependency Injection is the work of McDermott et al. [98]. McDermott uses the ideas behind GE to produce a very general EA that can be easily adapted to suit many needs. Instead of the fixed grammar derivation process of GE, PODI can use any non-deterministic program (NDP). The NDP's possible outputs are the feasible solution space, i.e. the possible phenotypes. The NDP can then be viewed as a mapping from an integer-array genotype to a phenotype.

TAGE

Tree Adjunct GE is a system designed by Murphy et al. [105, 107, 108, 109]. TAGE uses Tree Adjunct Grammars instead of canonical GE's context-free grammars to construct solutions to problems. This work has since been extended further to incorporate a Gene Regulatory Network (GRN) to guide the construction of TAGE solutions and allow for a developmental approach to TAGE [106].

CGE

Constituent Grammatical Evolution (CGE) by Georgiou [45, 47] is a version of GE that has been extended to incorporate constituent genes and conditional behaviour switching. These features are claimed to reduce genotype bloat, narrow the search space during evolution to more favourable sites and reduce the impact of destructive crossover events.

mgGE

Incorporating meta-grammars into GE was the doctoral work of Hemberg [57, 58, 63, 64, 66]. Hemberg presented two systems, Grammatical Evolution of Grammatical Evolution (GE^2) and mGGA. GE^2 used GE to evolve the grammar that was then passed into GE to be used to solve the problem specified. mGGA used a meta-grammar approach to modularity in a GA system similar to GE.

Modular GE

Swafford et al. [158, 159, 160, 161, 162] focused on extending GE to identify modules during a GE run. Once these modules have been identified they are assessed and the best modules are incorporated into the grammar for use in the run.

πGE

Position Independent GE by O'Neill et al. [127] strives to remove the positional dependency in the canonical GE mapping process. In πGE two codons are used for each expansion of the derivation tree. One codon is used to determine the position in the current tree where expansion will take place, while the second codon is then used in the standard GE way to determine what the expansion will be. Further study of πGE makes up a significant portion of this thesis and is explained in much further detail in Chapters 3, 4, 5 and 6.

Chorus

Chorus is a position independent encoding system for grammar based EA's. Unlike πGE above, where the position of the mapping varies during construction of the derivation tree, in Chorus the reading of codons from the chromosome is what is subject to positional change. Chorus uses a modified mod rule, where every production choice is considered, unlike GE where only the relevant production choices are considered. This modified mod rule, in conjunction with a concentration table, is used to construct derivation trees in a depth first manner similar to GE but using the concentration table to jump around the chromosome selecting codons as needed. The use of this table allows for Chorus to not leave any unused genetic introns in the chromosome that would occur if using the modified mod rule exclusively. This work is explained in detail in the doctoral work of Azad [1, 146, 147].

GAuGE

Genetic Algorithms using Grammatical Evolution was the doctoral work of Nicolau [26, 113, 114, 116, 117, 118, 119, 148, 149]. GAuGE uses GE and attribute grammars to evolve what each gene positions in the GA codes for. The system features position independent genetic algorithms, and uses the mod operation on codon values allowing for redundant coding, to mention but a few features.

2.5.3 Applications

GE has been applied to a wide variety of fields. The following overview of some of these applications serves to highlight the adaptability of GE to any problem domain.

The financial problem domain has received a lot of attention from a GE point of view. Dempsey [24] used the financial domain as a way to examine GE in dynamic environments. Brabazon et al. have published a book on the application of GE to financial problems [8]. GE has been used on a wide variety of financial problems, including identifying corporate stability [7], evolving trading strategies [22, 23], bankruptcy predictions [5], and generating credit ratings [6].

Subjective Design is another area where GE has been used in a significant body of work. Hemberg et al. [67] used GE to design Digital Surfaces using Genr8. O'Neill et al. [125] applied GE to the task of logo design. GE has also been applied to architectural design. O'Neill et al. [130] introduced shelters designed using GE. Byrne et al. [11, 12] designed bridges and other structures, whilst Ortega et al. [137] evolved fractal curves of given dimensions. GE has also been applied to the field of music generation. Reddin et al. [141] evolved elevator music whereas Shao et al. [155] used GE in his JIVE system to interactively evolve music.

Computer games represents another area of exploration. Murphy et al. [110, 111, 112] focused on using GE to evolve horse gait animation for graphical models. Galvan-Lopez

et al. [42, 43] explored designing Ms Pacman game controllers using GE. More recently the Super Mario Bros game has been the focus of GE applications in games. Perez et al. investigated evolving decision trees to play the game [138], before examining a hybrid A*-GE approach [139]. Moving away from controlling the player, Shaker et al. used GE to evolve levels for Mario [153] and personalised content for the game [154]

The above represents three areas of significant GE application. However GE has been applied to many more domains. Some application of note in the hardware domain are, Hemberg et al. [59, 60, 61] used GE to manage femtocell coverage. O'Neill et al. [132] evolved caching algorithms for computer processors. Colmenar et al. [20] evolved multi-objective dynamic memory managers. Jones et al. [70] evolved electronic circuit designs.

GE has also been used in the software domain. Cebrian et al. [16] evolved plagiarism tool validation methods. Sen et al. [151] and Wilson [167] both evolved techniques for intrusion detection but on differing network topologies. McIntyre et al. [99] evolved a multi classifier through crowding. Harper [53] evolved Robocode tank controllers. Tavares et al. [163] evolved ant colony optimisation algorithms. Drake et al. [27] applied GE to vehicle routing, evolving the variable neighbour search. Burke et al. [10] evolved hyper heuristics for the bin packing problem. Drchal et al. [28] evolved weights and topologies for neural networks. Escula et al. [31] captured protein structures. Georgoulas et al. [48] focused on classifying fetal heart rate data. Nicolau et al. [120] used GE to evolve models of the Net Ecosystem CO2 Exchange. Chen [17] used GE to aid in reservoir inflow forecasting.

In summary this section provided a brief overview of GE applications and demonstrates its versatility as a problem solving methodology.

2.5.4 Implementations

There are now many open source implementations of GE available for use by the general public. What follows is a brief summary of some of those available.

ECJ

Developed by Luke [96] and recently reviewed by White [165]. ECJ provides an implementation of GE, and many other EC methods, in Java and is freely available from `http://cs.gmu.edu/~eclab/projects/ecj`.

GEVA

Grammatical Evolution in Java (GEVA) by O'Neil et al. [129], developed by the UCD Natural Computing Research and Applications group is an implementation of GE in Java. GEVA is available from `http://ncra.ucd.ie/Site/GEVA.html`.

JCLEC

The Java Class Library for Evolutionary Computation (JCLEC) by Ventura et. al. [164] provides a suite of EC methods including GE. This library is freely available from `http://jclec.sourceforge.net`

jGE

jGE by Georgiou and Teahan [46] is another implementation of GE in Java. jGE can be downloaded from `http://pages.bangor.ac.uk/~eep201/jge`

libGE

libGE by Nicolau and Slattery [121] is a C++ implementation of GE. libGE is available at `http://bds.ul.ie/libGE`.

GEM

Grammatical Evolution in Matlab (GEM) by Hemberg is an implementation of GE that runs inside Matlab. The software is available from `http://ncra.ucd.ie/GEM`.

ponyGE

ponyGE is a very compact implementation of GE in Python. The software by Hemberg et al. is available from `http://code.google.com/p/ponyge`.

PyNeurGen

Python Neural Genetic Algorithm Hybrids (PyNeurGen) contains an implementation of GE in Python. PyNeurGen is freely available for download at `http://pyneurgen.sourceforge.net`.

DRP

Directed Ruby Programming (DRP) by McKeon, contains an implementation of GE in Ruby. The software is available from `http://drp.rubyforge.org`.

GERET

The GE Ruby Exploratory Toolkit (GERET) by Suchmann is an GE implementation in Ruby. The software is available from `http://geret.org`.

PODI

Program Optimisation with Dependency Injection (PODI) by McDermott et al. [98] provides an implementation of GE, and other EC methods, in Python. The software is available from `http://www.github.com/jmmcd/PODI`.

For the most up to date and comprehensive list of GE implementations available, the reader can consult `http://www.grammatical-evolution.org` and `http://ncra.ucd.ie/Site/Software`.

2.6 Summary

What has been presented in this chapter serves as an introduction to GE, which is exclusively used in the experimental part of this thesis. The main idea behind GE was introduced, and compared and contrasted against GP. Following this the workings of the algorithm were presented before finishing the chapter with an overview of the research, applications and implementations of GE that exist.

Part II

Experimental Research

Chapter 3

Exploring the Genotype-Phenotype Map in GE

This chapter presents an analysis of the genotype-phenotype map (GPM) in Grammatical Evolution (GE). The standard map adopted in GE is a depth-first, left-to-right, expansion of the non-terminal symbols during the derivation sequence. Earlier studies have indicated that allowing the order of the derivation tree expansion to be evolved during evolution produces performance gains [127]. This study extends the previous study to include a breadth-first and random approach to derivation tree construction. Investigation of the different approaches is performed on a selection of benchmark problems. It is concluded that it is possible to improve the performance of grammar-based Genetic Programming, such as GE, by the manner in which the genotype-phenotype map is performed. This chapter presents a more in-depth examination of work presented by Fagan et al. [40].

The chapter is structured as follows. An introduction motivating this research is presented in Section 3.1. Various approaches to the GPM are described in Section 3.2. Section 3.3 outlines the experimental setup, and the results are outlined in Section 3.4. Section 3.5 presents a discussion, before the chapter concludes with a summary in Section 3.6.

3.1 Introduction

Within the field of Genetic Programming (GP) [91, 140] the use of a genotype-phenotype map is not new [2, 41, 51, 82, 83, 85, 97, 157], and a number of variants to the standard tree-based form of GP exist, amongst which some of the most popular are Linear GP [9], Cartesian GP [103] and Grammatical Evolution (GE) [25, 133]. GE is a grammar-based form of GP which adopts a mapping from a linear genotype to phenotypic GP trees. O'Neill [124] 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.

Previous studies into enhancing the GPM in GE have focused on how the genotype is interpreted. Nicolau [26] and Azad [1] put forward methods that enhanced the GPM by determining how each codon would be interpreted, whilst maintaining a fixed order of derivation tree expansion. πGE first presented by O'Neill et al. [127] provided GE with a GPM that was not constrained by the linearity of the GE genotype. πGE used the linear GE genome, but allowed for variation of the location of derivation tree expression. This variation of location was left under the control of evolution. With πGE, O'Neill aimed to break the dependency of the linear GE genome, and instead provide smaller fragments of genetic material that may be exchanged with different area's of the derivation tree by using the evolvable expansion positions in πGE. This process would have an effect akin to that of sub tree crossover in GP, where small building blocks of a tree structure can be changed.

This study aims to verify the claims that πGE's GPM presents a performance gain over the traditional GE GPM. The study also sets out to investigate if other expansion orders can provide a similar performance benefit over the traditional GPM.

3.2 Genotype-Phenotype Maps

Four alternative mappers 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. [127] 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 sequence. 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.

A sample grammar is outlined in Figure 3.1, including an example chromosome. Figure 3.2 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 of an individual structure. Alternatively if a breadth-first expansion was adopted, Figure 3.3 illustrates how the order changes and thus the focus of evolutionary search takes a different direction towards broader tree structures. With the πGE approach [127] 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 = 2,12,7,9,3,15,23,1,11,4,6,13,2,7,8,3,35,19,2,6
```

Figure 3.1: An example grammar and chromosome. This grammar and chromosome will be used in all the GPM examples in this chapter.

3.2.1 GE - Depth-First GPM

Depth-First GPM represents the first mapping that will be examined. This GPM is important as it is the method of mapping used in canonical GE. In GE the derivation tree is expanded in a depth first manner by always selecting the leftmost NT to expand. The GE mapping process was explained in detail in Section 2.3. The genotype-phenotype map of GE operates as follows. An example of a depth-first mapping using the example grammar and chromosome in Figure 3.1 is displayed in Figure 3.2. A pseudo-code outline of the depth-first mapping process is also shown in Algorithm 3.1.

3.2.2 Breadth-First GPM

The breadth-first approach to mapping presents a slight variation to that of the traditional GE GPM. In breadth-first mapping as in depth-first, the mod rule is utilised to dictate what each NT will be expanded to. Breadth-first however will expand the derivation tree level by level in a left to right manner. This is where the mapping differs to GE's depth-first approach.

Figure 3.3 shows the breadth-first GPM mapping the example grammar and chromosome shown in Figure 3.1. The figure shows that the tree is expanded level by level until the tree is completed. Changing the mapping order of a derivation tree can have some large effects. In the canonical GE mapping the codons that specifically control the leaf nodes

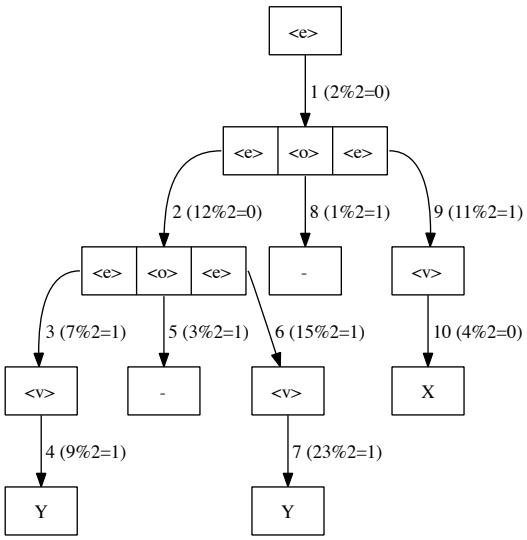


Figure 3.2: Example of the depth-first mapping process, which will be referenced ta as the GE mapping process. In the depth-first GPM the derivation tree is expanded in a depth-first left-most order. In this figure the expansion order is indicated on the arrowed edges between the tree nodes. $3(7\%2 = 1)$, indicates that this was the third expansion in the mapping and that $7\%2 = 1$ dictates what the third expansion will entail.

can be distributed evenly over the chromosome. In the breadth-first approach however all these codons will tend to be clustered towards the end of the chromosome. This makes it more likely that a crossover or mutation event will result in the changing of all the leaf nodes, due to the ripple effect of these operators in GE. Changing the GPM ordering can also affect the expression of the grammar. Hemberg [66] investigated this and found that some problem domains may benefit from the usage of a certain grammar notation and a certain mapping order, while others might not, due to biases that exist in the grammars and mappings. A pseudo-code outline of the breadth-first mapping process is also shown in Algorithm 3.2.

Algorithm 3.1 GE Depth-First Genotype-Phenotype Map Algorithm. The approach shown builds a derivation tree by always expanding the left-most NT, by always adding new productions to the start of the list we can guarantee this ordering. The generation of the phenotype has been extracted to a single method call at the end of the pseudo-code for clarity.

```
listNT {List to store NT's seen}
Add start symbol from grammar to listNT
wraps = 0
while listNT is not empty do
  if reached end of chromosome then
    wraps++
    if wraps > max wraps allowed then
      return false
    reset chromosome iterator
  currentNT = get head of listNT
  currentCodon = get next codon value
  newProduction = currentCodon % number of productions for currentNT
  set currentNT's children = newProduction
  {This is the key to depth first mapping}
  add newProduction to head of listNT {Only adds NTs}
Generate Phenotype by traversing the leaf nodes of the derivation tree.
return true
```

3.2.3 πGE - Position Independent GPM

The πGE GPM [127] differs from the traditional GE mapping in one way. While the expansion of the NTs is performed identically in both approaches the order in which these expansions take place is different. GE adopts a fixed order mapping, while πGE uses evolution to control the order of NT expansions.

Before any mapping can be done in πGE, there are some changes that need to be made to the 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 a chromosome of pair values. The first codon of the pair (The Order Codon), is used to choose which NT to expand and the second (The Content Codon), is used to choose what the production, based on the rules available for a NT of that type, just like in GE.

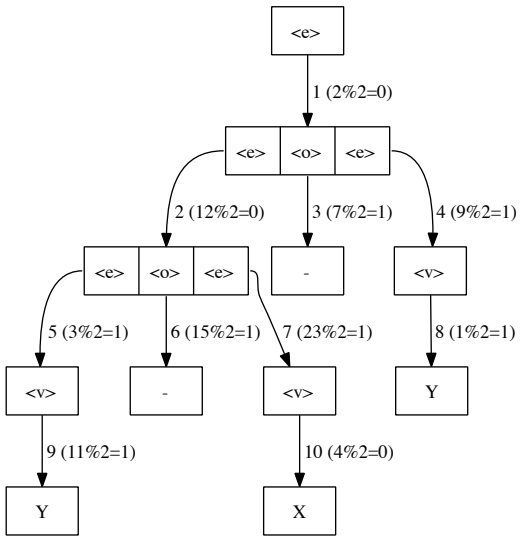


Figure 3.3: Example of the breadth-first mapping process. The derivation tree is expanded level by level in a left to right direction. In this figure the expansion order is indicated on the arrowed edges between the tree nodes. $2(12\%2 = 0)$, indicates that this was the second expansion in the mapping and that $12\%2 = 0$ dictates what the second expansion will entail.

The chromosome shown in Figure 3.1 can be viewed as a list of paired values such as `((2,12),(7,9)...)`.

πGE Mapping Example

The mapping process begins from the embryonic start symbol of the grammar. Taking the simple grammar provided in Figure 3.1 this is $< e >$. $< e >$ is then added to the list of possible expansions, $[< e >]$. Selecting the first πGE codon from the chromosome (Figure 3.1) yields the codon $(2, 12)$. The order codon, 2 is then passed to Equation 3.1, that results in selecting the NT to be expanded from the NT list, $[(< e >)]$ as $2\%1 = 0$. Now for the second half of the πGE expansion we have to perform the standard GE expansion on the selected NT. In this case there are two possible transformations which can be applied to $< e >$. Either it will be replaced with $< e >< o >< e >_0$ or with $< v >_1$.

Algorithm 3.2 Breadth-First Genotype-Phenotype Map Algorithm. This algorithm is similar to the depth-first algorithm (Algorithm 3.1). The only difference is that when adding the new productions to the NT list, the new production is added to the tail of the list. This simple step guarantees a breadth-first ordering, instead of the depth-first ordering that appending to the head of the list provides.

```
listNT {List to store NT's seen}
Add start symbol from grammar to listNT
wraps = 0
while listNT is not empty do
  if reached end of chromosome then
    wraps++
    if wraps > max wraps allowed then
      return false
    reset chromosome iterator
  currentNT = get head of listNT
  currentCodon = get next codon value
  newProduction = currentCodon % number of productions for currentNT
  set currentNT's children = newProduction
  {This is the key to breadth first mapping}
  add newProduction to tail of listNT {Only adds NTs}
Generate Phenotype by traversing the leaf nodes of the derivation tree.
return true
```

To decide what rule is taken, the content codon 12 and the number of choices available are used in conjunction with Equation 3.2. In this case $12\%2 = 0$ so $< e >$ will be transformed into $< e >< o >< e >$, and the NT list will be updated, $[< e >, < o >, < e >]$.

$$NT\ to\ expand\ =\ Order\ Codon\ \%\ |NT\ list| \tag{3.1}$$

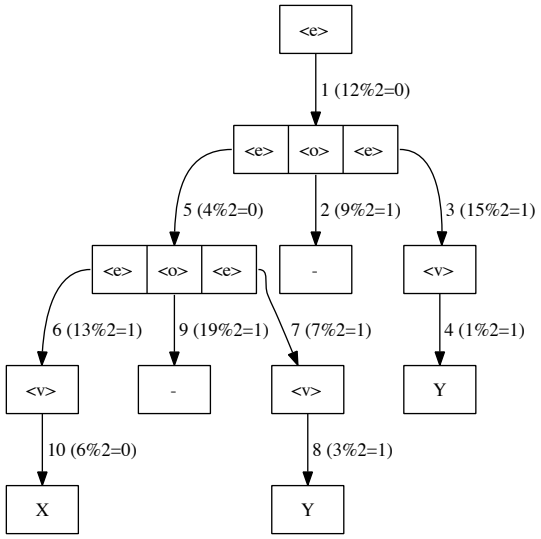
$$Expansion\ Choice\ =\ Content\ Codon\ \%\ Number\ of\ rules\ for\ NT \tag{3.2}$$

The second expansion of the πGE derivation tree follows a similar process, first the next πGE codon is read, $(7, 9)$. The order codon of the pair is used to select the next NT to expand, $< o >$ is chosen as $7\%3 = 1$. Next the content codon is used to select what the expansion becomes. Similarly to the first expansion $< o >$ has two possible productions,

$+_0$ or $-_1$, and $-$ is chosen in this case as $9\%2 = 1$. As $-$ is a terminal it is not added to the NT list and so the list now consists of $[< e >, < e >]$. When a NT production results in the generation of new NTs, these NTs are placed at the same location in the NT list of possible expansions, that the initial NT was selected from.

```
1.    [(e)]  => 2%1=0
2.    [e,(o),e]  => 7%3=1
3.    [e,(e)] => 3%2=1
4.    [e,(v)] => 23%2=1
5.    [(e)] => 11%1=0
6.    [(e),o,e] => 6%3=0
7.    [v,o,(e)] => 2%3=2
8.    [v,o,(v)] => 8%3=2
9.    [v,(o)] => 35%2=1
10.   [(v)] => 2%1=0
```

(a) πGE order choice list.



(b) πGE derivation tree example

Figure 3.4: Example of the πGE mapping process. The derivation tree is expanded in the order that is dictated by the chromosome and Equation 3.1. This process is outlined in Figure 3.4a. In this figure the expansion order is indicated on the arrowed edges between the tree nodes. $5(4\%2 = 0)$, indicates that this was the fifth expansion in the mapping and that $4\%2 = 0$ dictates what the fifth expansion will entail.

This expansion process is repeated until the tree is completed or the derivation process reaches the end of the chromosome. If all the codons have been used the mapper will either return an invalid individual or else wrap around to the start of the chromosome and continues mapping (if wrapping is enabled). Figure 3.4 provides the complete derivation example, with Figure 3.4a showing the NT list at each step of the derivation, and Figure 3.4b showing the completed derivation tree. The number associated with each branch of the tree is a reference to the numbered steps shown in Figure 3.4a, which show how each

Algorithm 3.3 πGE Genotype-Phenotype Map Algorithm. The addition of the order codons are highlighted. It can now be seen how the order codons are used to pick an index in the NT list. It is also worth noting how the new productions are then added to the index where the parent NT was taken from originally. This preserves the ordering of the derivation string, which is of importance in relation to the experiments performed in Chapter 5.

```
listNT {List to store NT's seen}
Add start symbol from grammar to listNT
wraps = 0
while listNT is not empty do
  if reached end of chromosome then
    wraps++
    if wraps > max wraps allowed then
      return false
    reset chromosome iterator
  {This is where the πGE order comes in}
  currentOrderCodon = get next codon value
  nextProductionIndex = currentOrderCodon % size of listNT
  currentNT = get listNT[nextProductionIndex]
  currentContentCodon = get next codon value
  newProduction = currentCodon % number of productions for currentNT
  set currentNT's children = newProduction
  {The new NT's are added where the parent NT was removed from}
  insert newProduction at listNT[nextProductionIndex] {Only adds NTs}
Generate Phenotype by traversing the leaf nodes of the derivation tree.
return true
```

choice of NT to expand comes about. A pseudo-code outline of the πGE mapping process is also shown in Algorithm 3.3.

3.2.4 Random Order GPM

Random Order GPM is a mapping where the expansion order of the derivation tree is under no form of ordering control at all, unlike the previous mapping examples. At each expansion step the mapping algorithm is free to choose from any of the NTs available. This random mapping approach is non deterministic, running the GPM twice will result in two different orders of derivation tree expansion.

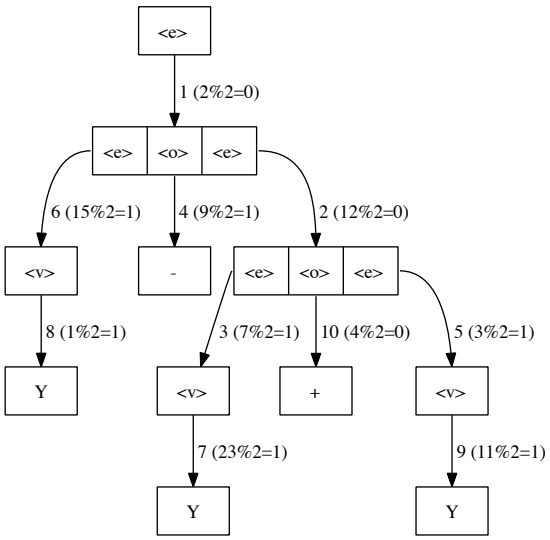


Figure 3.5: Example of the random position mapping process. In the random GPM the choice of what NT to expand next is selected at random, resulting in a non deterministic GPM. In this figure the expansion order is indicated on the arrowed edges between the tree nodes. E.g. $7(23\%2 = 1)$ indicates that this was the seventh expansion in the mapping and that $23\%2 = 1$ dictates what the seventh expansion will entail.

Figure 3.5 shows an example of a random GPM. The example uses the grammar and chromosome shown in Figure 3.1. The random GPM serves two purposes in this study. Firstly it provides for a base level of performance of GE, as with the random GPM the algorithm is randomly selecting derivation trees and is essentially random search. Secondly the random GPM provides a control for πGE. The πGE and random mappings differ only in that πGE has its ordering under the control of evolution whilst the random GPM does not. A pseudo-code outline of the random order mapping process is also shown in Algorithm 3.4.

Algorithm 3.4 Random Genotype-Phenotype Map Algorithm is very similar to the πGE algorithm. In the random mapping instead of using the second codon for the order, a random index is selected for each expansion. The generated NT's are again inserted into the list at the same position as the parent NT was removed from.

```
listNT {List to store NT's seen}
Add start symbol from grammar to listNT
wraps = 0
while listNT is not empty do
  if reached end of chromosome then
    wraps++
    if wraps > max wraps allowed then
      return false
    reset chromosome iterator
  {This is where the random order comes in}
  nextProductionIndex = random index within bounds of listNT size
  currentNT = get listNT[nextProductionIndex]
  currentCodon = get next codon value
  newProduction = currentCodon % number of productions for currentNT
  set currentNT's children = newProduction
  {The new NT's are added where the parent NT was removed from}
  insert newProduction at listNT[nextProductionIndex] {Only adds NTs}
Generate Phenotype by traversing the leaf nodes of the derivation tree.
return true
```

3.3 Experimental Setup

We wish to test the null hypothesis that there is no difference in performance when alternative mapping strategies are adopted with GE. Performance will be measured both in terms of the number of successful solutions found to each problem instance, and by examining the average best fitness. The problems examined are outlined in Section 3.3.1.

GEVA v2.0 [129] was adopted for the experiments conducted in this study. The evolutionary parameters adopted on all problems are presented in Table 3.1. Note that a relatively small population size of 100 was deliberately used, 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 provide the possibility to

Table 3.1: Parameter settings adopted on all problems examined.

Parameter	Value
Generations	100
Population size	100
Replacement strategy	generational with elitism (3 individuals)
Selection	tournament (tsize=3)
Mutation probability	0.01 (integer mutation)
Crossover probability	0.0 & 0.9 (ripple)
Initial chromosome length	200 codons (random init), πGE 400 codons

more precisely distinguish performance differences on these benchmark problems. Elitism was restricted to a size of 3 to prevent the population from converging too quickly. For all approaches except one an initial chromosome length of 200 codons was selected to provide all mappings with a similar amount of randomly created genetic material. πGE uses two GE style codons for a single πGE codon, because of this it was provided an initial chromosome length of 400.

3.3.1 Problems Examined

Four standard GP benchmark problems [166] were examined, and 250 independent runs performed for each setup on each problem. The grammar adopted in each case appear in Figure 3.6.

Even-5-Parity

Even-5-Parity [89] is a 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

Symbolic Regression [89] is a type of problem where the goal is to evolve a function that fits to a set of test cases. The classic quartic polynomial function is used here $x + x^2 + x^3 + x^4$ with 20 input-output test cases evenly spaced between the range -1 to 1. Fitness is the root mean square error. Success on the problem is measured using the notion of hits, where a hit is achieved when the error is less than 0.000001. A bivariate version of the problem, $x_0^4 + x_1^2$ is also examined over the same range.

Santa Fe Ant Trail

In the Santa Fe Ant Trail problem [88] 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 ahead to determine if there is food present.

Max

The aim of the max problem [44] is to evolve a tree that evaluates to the largest value within a set depth limit (6 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 the population tends to converge on suboptimal solutions which can be difficult to escape from as is shown by Langdon et al. [95].

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

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

<op> ::= + | *

<var> ::= 0.5
```

(a) Max.

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

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

<op> ::= + | - | * | /

<var> ::= x0 | x1 | 1.0
```

(b) Symbolic Regression A.

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

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

<op> ::= + | - | *

<var> ::= x0 | 1.0
```

(c) Symbolic Regression B

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

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

<pre-op> ::= not

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

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

(d) Even 5 Parity.

```
<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>
```

(e) Santa Fe Ant.

Figure 3.6: Grammars used for the problems in this chapter are presented above. There were five grammars used during the course of this investigation.

3.3.2 Statistical Tests

The quantities of interest are average best fitness and number of successful solutions found. For the successful solutions a simple head count approach will be used. Examination of the average best fitness values will require the addition of a statistical test.

In this thesis the Wilcoxon Rank-Sum test will be used. The Wilcoxon Rank-Sum was chosen as it is valid on normally and non-normally distributed data. The Wilcoxon Rank-Sum test is used to determine if two samples are drawn from the same distribution. The null hypothesis is that the samples are the same, and the test returns a p-value indicating the probability that this data could have arisen by chance, when drawing from the same distribution. For this thesis a p-value < 0.05 will be taken to indicate that the samples

are drawn from different distributions. As multiple samples of data are being compared pairwise, a Bonferroni correction will be applied to the threshold. This correction entails dividing the threshold by the number of pairwise tests. In this chapter there are four datasets, hence six pairwise tests. This results in a new threshold of 0.0083.

3.4 Results

The following section presents the findings of the study. Section 3.4.1 presents a comparison of the performance of the different GPM approaches on the benchmark problems. Section 3.4.2 investigates the effect the crossover operation has on the different GPMs, before the discussion is focused on the performance of the random GPM in Section 3.4.3.

3.4.1 Overall Performance

Table 3.2 presents the general findings for this study, while Tables 3.3 and 3.4 display the p-values of each GPM compared to the other GPM approaches, with and without crossover respectively. Plots of the average best fitness, with and without crossover, across the 250 runs are displayed in Figures 3.7 and 3.8 respectively.

Examining the tables there are 10 problem instances displayed (five problems with and without crossover). Looking at the number of successful solutions it can be seen that πGE finds a larger number of successful solutions than the other GPM approaches, in seven of the ten instances, and in one other case is only a single solution away from the best GPM approach. The closest competitive GPM only achieved the best in two of the ten instances. Examining solely the average best fitness it can bee seen that πGE is the most consistent GPM as it has displays the best average best fitness, or is statistically similar to the best fitness, in nine of the ten problem instances.

Table 3.2: This table shows the results for the different genotype-phenotype mapping approaches on a range of benchmark problems. 250 runs for each setup were performed and the results are shown below. In the table SS denoted the number of solution found during the 250 runs. The average of the best fitness is presented with the standard deviation shown in brackets, and also the average of the average population fitness is also presented. The highlighted cells indicate the best performing approach. In the case where multiple approaches are highlighted, there is no statistical difference between the approaches.

Approach	SS	Avg. Best	Avg. Avg.	SS	Avg. Best	Avg. Avg.
	Crossover 0.0			Crossover 0.9		
Even 5 Parity						
BF	123	3.596 (3.853)	6.274 (3.256)	158	2.464 (3.596)	5.639 (3.191)
GE	157	2.348 (3.403)	5.449 (3.057)	166	2.108 (3.192)	5.444 (2.746)
πGE	156	2.66 (3.612)	8.085 (3.189)	170	2.392 (3.595)	8.341 (2.759)
Random	83	4.928 (3.736)	15.691 (0.121)	90	4.628 (3.796)	15.698 (0.113)
Max 6						
BF	0	11.93 (1.71)	12.12 (1.51)	1	9.49 (2.35)	10.26 (1.93)
GE	0	11.83 (1.62)	12.03 (1.46)	0	9.93 (2.49)	10.60 (2.09)
πGE	0	11.67 (1.39)	12.35 (0.98)	0	11.39 (1.22)	12.38 (0.75)
Random	0	12.69 (0.45)	15.28 (0.08)	0	12.46 (0.47)	15.30 (0.07)
Santa Fe Ant						
BF	2	41.09 (13.13)	45.94 (12.02)	6	34.11 (13.46)	41.55 (11.75)
GE	2	39.41 (13.23)	44.60 (12.05)	8	31.39 (13.99)	39.09 (12.38)
πGE	9	31.11 (14.46)	46.02 (11.53)	8	26.86 (14.30)	44.79 (10.65)
Random	3	28.66 (11.18)	82.74 (0.814)	2	27.12 (13.09)	82.98 (0.79)
Symbolic Regression A						
BF	8	0.584 (0.348)	0.795 (0.778)	13	0.37 (0.237)	2.36 (11.93)
GE	8	0.576 (0.329)	1.76 (14.969)	34	0.291 (0.219)	1.88 (12.78)
πGE	20	0.453 (0.328)	2.59 (12.34)	37	0.285 (0.246)	16.98 (166.5)
Random	1	0.627 (0.215)	3.235 (15.39)	0	0.642 (0.189)	3.59 (21.78)
Symbolic Regression B						
BF	143	0.087 (0.113)	0.347 (0.245)	200	0.033 (0.067)	1.97 (17.49)
GE	120	0.115 (0.146)	0.347 (0.221)	195	0.037 (0.072)	185.9 (29.25)
πGE	200	0.035 (0.074)	0.64 (0.318)	223	0.017 (0.052)	1.73 (9.77)
Random	100	0.111 (0.104)	1.301 (0.133)	95	0.115 (0.102)	1.36 (0.33)

Table 3.3: The table displays the p-values resulting from comparing each pair of setups using the Wilcoxon Rank-Sum test. p-Values < 0.0083 are highlighted. The setups compared are the for the results seen in Table 3.2 without crossover.

Crossover 0.0				
Even 5 Parity				
Approach	BF	GE	πGE	Random
BF	-	5.56E-05	0.0045	0.0001
GE	5.56E-05	-	0.2222	4.93E-12
πGE	0.0045	0.2222	-	8.03E-10
Random	0.0001	4.93E-12	8.03E-10	-
Max				
Approach	BF	GE	πGE	Random
BF	-	0.4157	0.0024	4.00E-10
GE	0.4157	-	0.1877	6.53E-11
πGE	0.0024	0.1877	-	9.26E-23
Random	4.00E-10	6.53E-11	9.26E-23	-
Santa Fe Ant				
Approach	BF	GE	πGE	Random
BF	-	0.1675	3.41E-15	3.28E-22
GE	0.1675	-	3.64E-10	2.27E-17
πGE	3.41E-15	3.64E-10	-	0.0405
Random	3.28E-22	2.27E-17	0.0405	-
Symbolic Regression A				
Approach	BF	GE	πGE	Random
BF	-	0.9783	1.28E-05	7.02E-02
GE	0.9783	-	8.33E-05	3.94E-02
πGE	1.28E-05	8.33E-05	-	1.89E-10
Random	7.02E-02	3.94E-02	1.89E-10	-
Symbolic Regression B				
Approach	BF	GE	πGE	Random
BF	-	0.0371	1.14E-08	2.00E-04
GE	0.0371	-	5.79E-14	3.65E-01
πGE	1.14E-08	5.79E-14	-	3.57E-20
Random	2.00E-04	3.65E-01	3.57E-20	-

Table 3.4: The table displays the p-values resulting from comparing each pair of setups using the Wilcoxon Rank-Sum test. p-Values < 0.0083 are highlighted. The setups compared are the for the results seen in Table 3.2 with crossover.

Crossover 0.9				
Even 5 Parity				
Approach	BF	GE	πGE	Random
BF	-	0.2828	0.8741	5.49E-09
GE	0.2828	-	0.1753	2.57E-12
πGE	0.8741	0.1753	-	1.53E-09
Random	5.49E-09	2.57E-12	1.53E-09	-
Max				
Approach	BF	GE	πGE	Random
BF	-	0.0445	3.63E-21	1.19E-39
GE	0.0445	-	9.35E-13	5.42E-35
πGE	3.63E-21	9.35E-13	-	1.79E-28
Random	1.19E-39	5.42E-35	1.79E-28	-
Santa Fe Ant				
Approach	BF	GE	πGE	Random
BF	-	0.0253	1.40E-09	9.06E-09
GE	0.0253	-	0.0002	2.00E-04
πGE	1.4E-09	0.0002	-	7.49E-01
Random	9.06E-09	2.00E-04	7.49E-01	-
Symbolic Regression A				
Approach	BF	GE	πGE	Random
BF	-	6.80E-05	2.30E-05	2.27E-26
GE	6.80E-05	-	0.5039	2.73E-35
πGE	2.30E-05	0.5039	-	1.00E-30
Random	2.27E-26	2.73E-35	1.00E-30	-
Symbolic Regression B				
Approach	BF	GE	πGE	Random
BF	-	0.7362	0.001	1.95E-19
GE	0.7362	-	0.0023	6.13E-18
πGE	0.001	0.0023	-	2.07E-25
Random	1.95E-19	1.13E-25	2.07E-25	-

Taking Even Five Parity without crossover, GE and πGE present very similar performance levels, with GE slightly out performing πGE. Examining Table 3.3 adds that GE doesn't significantly outperform πGE, however both setups are significantly better than the BF and Random approaches. The addition of crossover to the algorithms improves performance with πGE now finding more successful solutions and GE maintaining the best fitness. BF has improved and by examining Table 3.4 it is evident that now all setups are significantly better than the Random approach.

The Max problem presents an interesting result. πGE has marginally the best fitness in the no crossover setup, however there is only a significant difference with respect to the Random GPM. The addition of crossover results in BF achieving the best result and significantly outperforming the other approaches.

On the Santa Fe Ant problem, without crossover, the random GPM achieves the best fitness, in contrast with its relatively poor performance on most of the other problem instances. This may be down to the fact that random sampling has been shown to be good at solving the ant problem [94]. πGE presents as finding the most successful solutions. The πGE and Random approaches are significantly better than the GE and BF approaches when examining the p-values. The addition of crossover to the problem results in πGE achieving a tie with GE for the most solutions found, but has significantly better best fitness. The result is significant when compared to the GE and BF approaches, however the Random approach again shows good performance.

The Symbolic Regression problems represent a clean sweep for πGE, significantly outperforming all other approaches. On all four problem instances πGE has the most successful solutions, and statistically the lowest average best fitness.

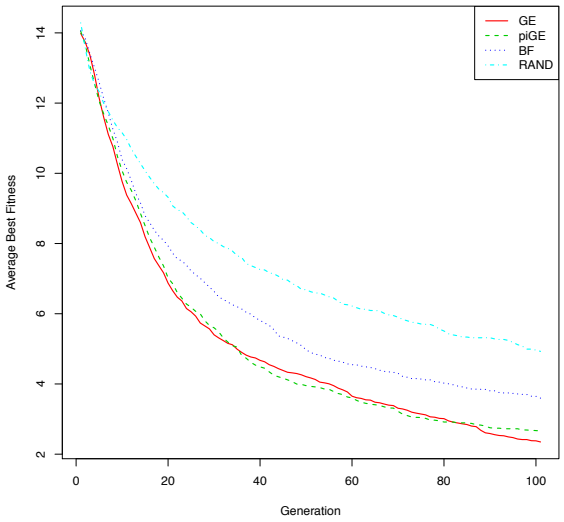
Looking back at the results as a whole, a trend can be seen by comparing the difference between the average best fitness and the mean average fitness in Table 3.2. πGE seems to maintain a much wider difference between the two then those of its closest rivals, GE

and BF. This gap gives an indication of how converged a population is fitness wise. The correlation between this and the performance gains seen cannot be ignored. πGE appears to provide a larger difference between best and average fitness during a run, indicating a less converged population fitness wise.

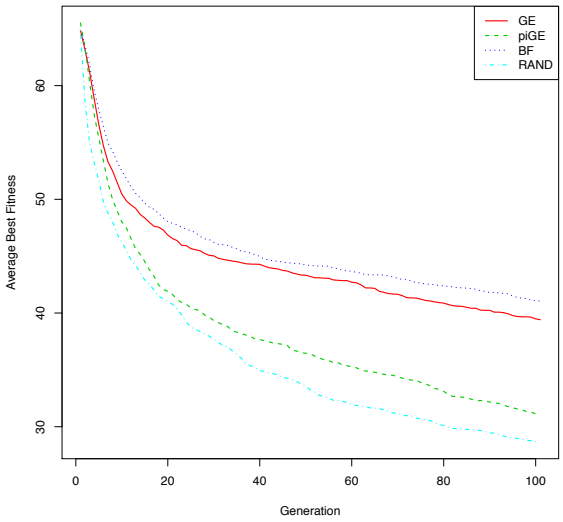
3.4.2 Crossover Performance

GE has been shown to take very good advantage of the crossover operation [135]. This is due to GE being able to take advantage of the genetic tails of the GE genotype. These sequences of unmapped codons, at the end of the genotype, provide the crossover operation with an increased probability of generating an individual that has a complete mapping sequence [115]. Consulting Table 3.2 it is evident that the fixed order mappings of GE and BF exhibit more of a performance increase with the use of crossover when compared to their variable order counterparts. In every problem domain both mapping approaches show marked improvement with crossover in terms of fitness and successful solutions found. πGE does not get as much of an increase in performance, for example in the Even 5 Parity and Max problems the πGE setups shows little improvement in average best fitness. However it can be seen that the number of successful solutions does increase with crossover in the majority of cases.

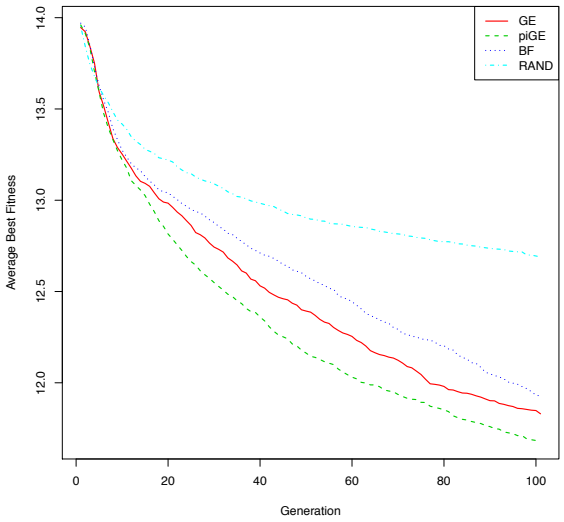
Figure 3.7 shows the average best fitness without crossover for four of the problem instances. Figure 3.8 shows the performance for the same problems with crossover. By comparing the graphs the increase in performance that GE and BF achieve can be seen. Figures 3.7c and 3.8c really emphasise this increase in performance.



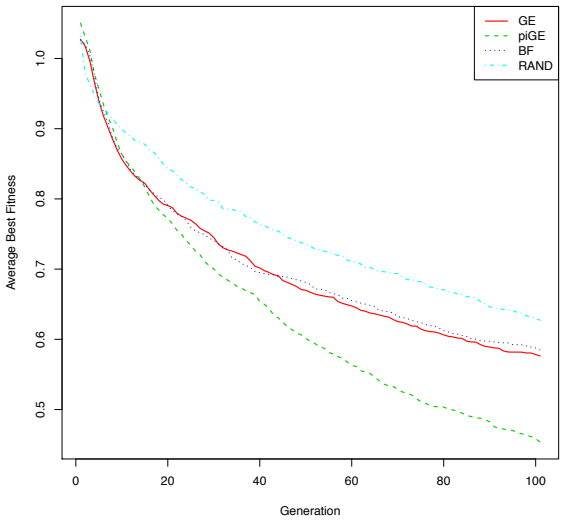
(a) Even 5 Parity - No Crossover



(b) Santa Fe Ant - No Crossover

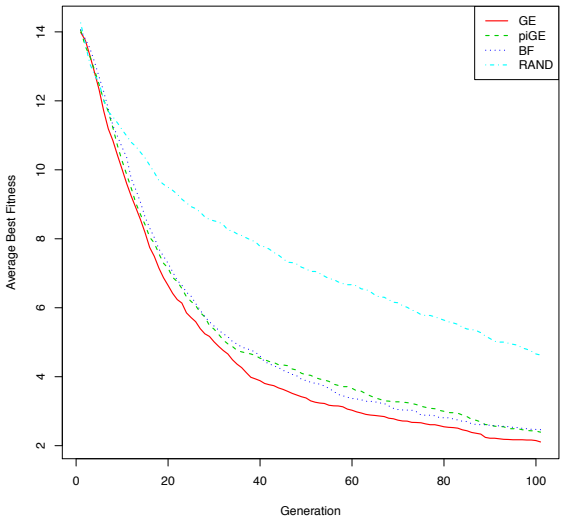


(c) Max - No Crossover

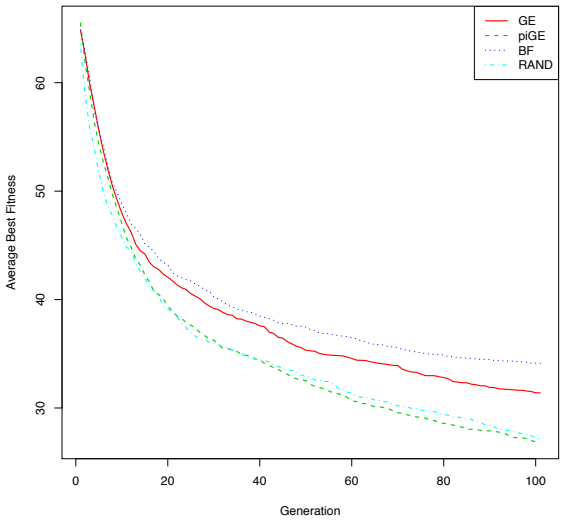


(d) Symbolic Regression A - No Crossover

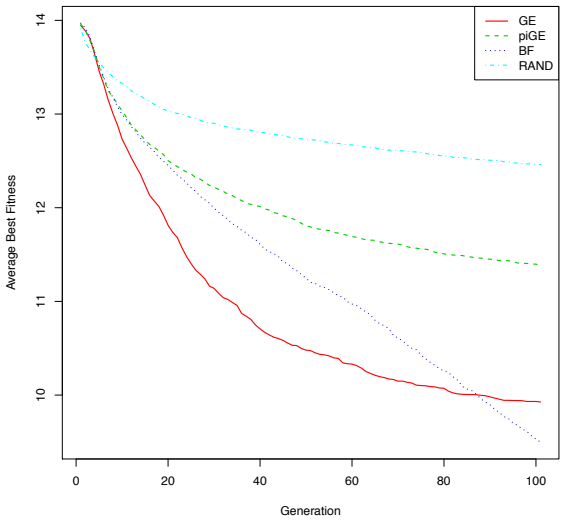
Figure 3.7: This figure shows the graphs for the average best fitness of the different GPM's on the problems examined without crossover.



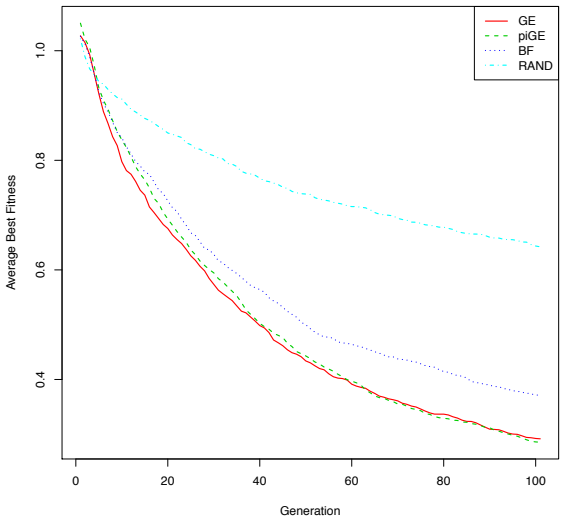
(a) Even 5 Parity - Crossover



(b) Santa Fe Ant - Crossover



(c) Max - Crossover



(d) Symbolic Regression A - Crossover

Figure 3.8: This figure shows the graphs for the average best fitness of the different GPM's on the problems examined with crossover

3.4.3 Random Performance

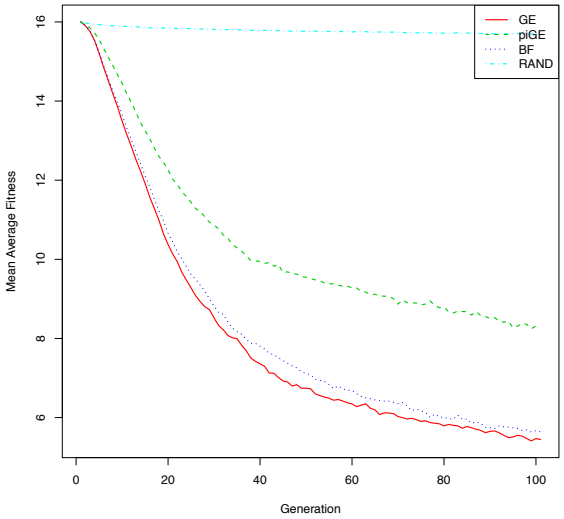
The random GPM has provided some interesting insight. The random GPM has provided a baseline for the study. However by looking at the mean average fitness and the used codon lengths of the GPMs used in this study, we can see the benefit for all other GPMs of retaining the mapping order in the GPM. Figure 3.9 shows the mean average fitness of the GPMs on two of the problem instances. Figure 3.10 shows the average used genes for the same two problem instances.

These figures show how the structure of the expansion orders, that is maintained by the genotype in πGE and inherently in the fixed order mappings, allows for the GPM to retain knowledge. The random GPM discards all this information every time it maps an individual, so it reaches a level of performance similar to that of random sampling.

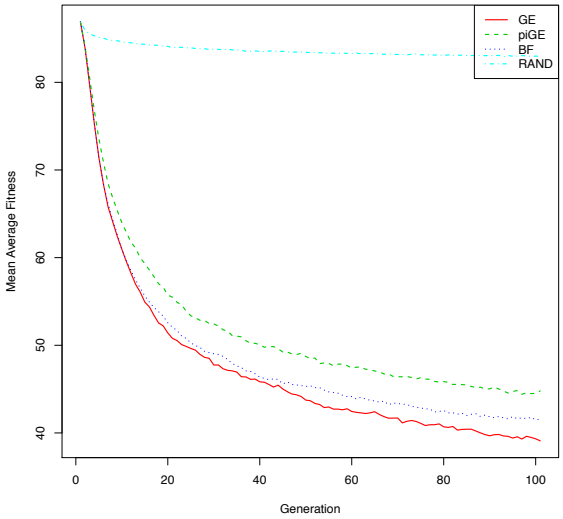
The Santa Fe Ant problem is an exception to this pattern of poor performance by the random GPM. Langdon [94] noted than randomly generating solutions provided very good performance on the ant problem. In Figure 3.8b it can be seen that the random GPM achieves a very good best fitness value. However, Figure 3.10b shows that random does not improve the average fitness of the population. This is because individuals created using the random GPM are effectively inheriting little genetic information, so mean performance can be very poor even while best performance is good. The good performance of the random GPM on the Santa Fe problem has negated the need for the GPM to retain the order information.

3.5 Discussion

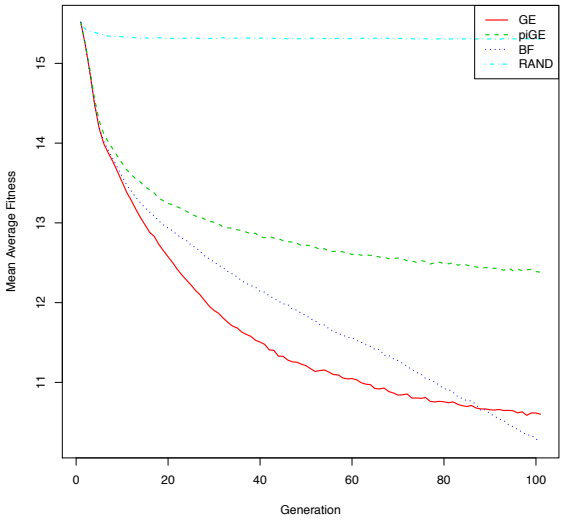
Overall πGE has distinguished itself as viable alternative to the traditional GE GPM. This is extremely encouraging considering the added search space that it has to overcome. The ability of this GPM to find solutions forms the bulk of the next two experimental chapters



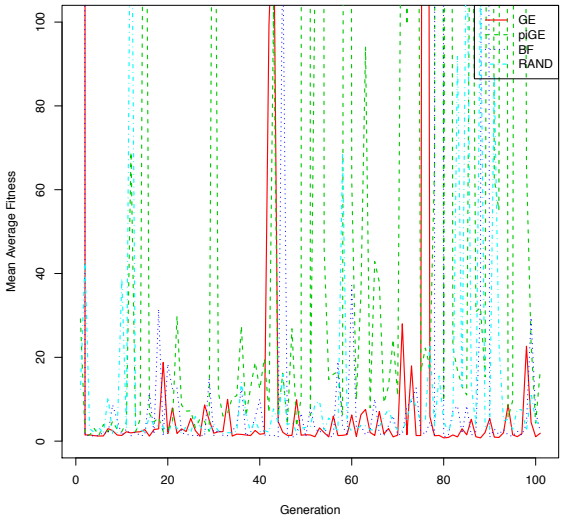
(a) Even 5 Parity - Crossover



(b) Santa Fe Ant - Crossover

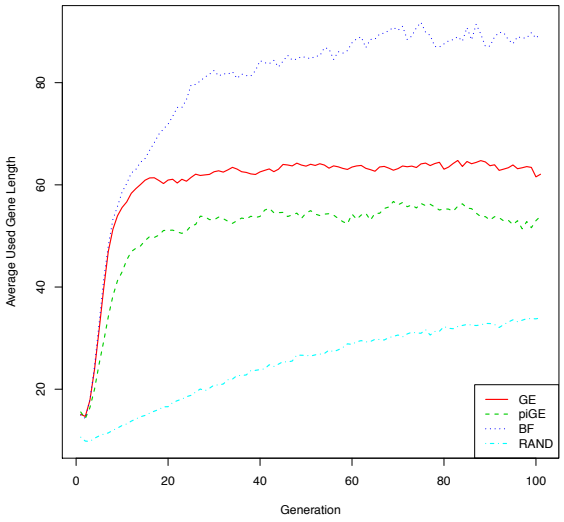


(c) Max - Crossover

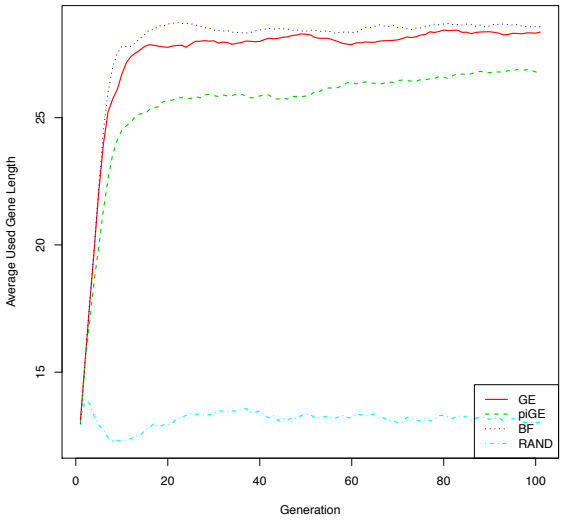


(d) Symbolic Regression A - Crossover

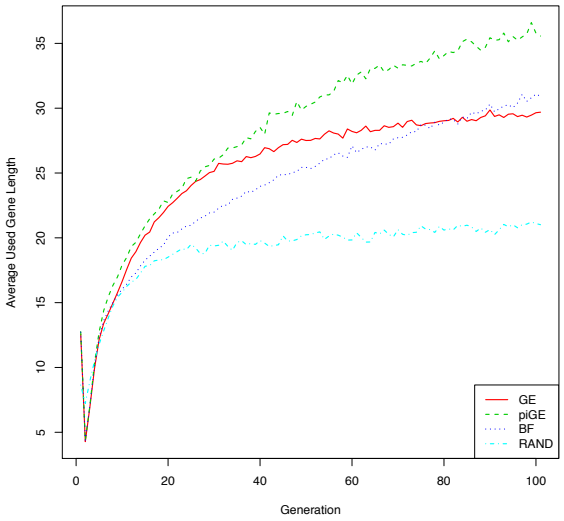
Figure 3.9: This figure shows the graphs for the mean average fitness of the different GPM's on the problems examined. Examining the graphs it can be seen that πGE has a worse mean average fitness than GE and BF. Random in contrast has a very large mean average fitness, indicative of its random sampling approach to search. The graph for symbolic regression is included for completeness, however the variation in fitness, due to outlier individuals of very bad fitness, makes it unreadable.



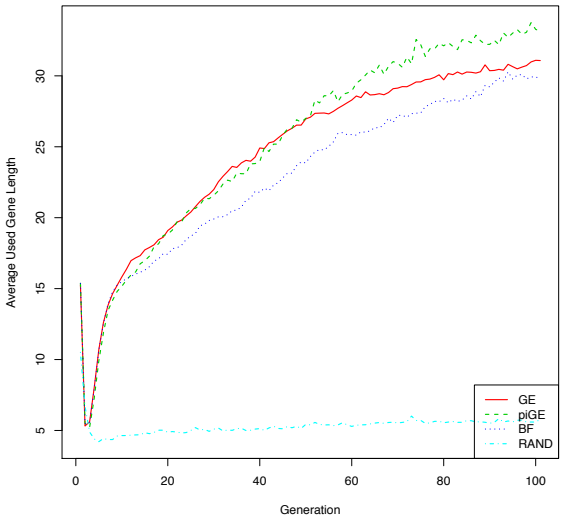
(a) Even 5 Parity - Crossover



(b) Santa Fe Ant - Crossover



(c) Max - Crossover



(d) Symbolic Regression A - Crossover

Figure 3.10: This figure shows the graphs for the average number of used genes of the different GPM's on the problems examined. It is interesting to note how the random mapping uses less genes than the other approaches, and seems to have difficulty building larger individuals.

of this thesis, where the inner workings of πGE will be explored.

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 [14]. The Max problem is more suited to a systematic pre-order (Depth-first) or level-order (Breadth-first) traversal of the tree, leading to better results faster than πGE.

3.6 Summary

An analysis of the genotype-phenotype map in Grammatical Evolution was presented, comparing performance of the standard *depth-first* approach to *breadth-first*, π*GE*, and *random* variations. Across the benchmark problems analysed we observed that the adoption of the more flexible πGE map, which is under the control of evolution, provided a viable alternative to the GE GPM. In some cases πGE presented significant performance gains.

The πGE GPM increases the genotypic search space, as the derivation sequence is incorporated into the genotype. This results in πGE having to search for orders of derivation as well the actually derivation expansions, therefore the results are even more impressive. The effect of crossover on different mappings was also investigated and found to vary in effectiveness with different GPMs and problem domains. The fundamental effect of a fixed or preserved order of expansion was also investigated. The following chapters aim to investigate the inner workings of πGE and present new insights into our understanding of the algorithm.

Chapter 4

Phenotypic Connectivity in πGE

This chapter explores how πGE can remain a competitive search method when compared to GE (as explored in Chapter 3), considering that it has to overcome the additional search space of expansion orders. Understanding how an algorithm navigates its search landscape can aid in explaining the performance of the algorithm. Furthermore, visualising this landscape has been shown to aid in this understanding [71, 92, 108]. The phenotypic landscape is selected for examination in this chapter. A phenotype is defined as the concatenation of the leaf nodes of the derivation tree in GE and πGE. A phenotype represents candidate solution to a problem, and is evaluable to produce a fitness value for the phenotype. By examining how the connections between phenotypes differ in GE and πGE, it becomes clearer how πGE can match GE's performance, given the added search overhead of πGE. This chapter presents an in depth examination of work presented by Fagan et al. [35].

The chapter is structured as follows. Section 4.1 outlines the increase in search complexity πGE adds, and introduces the landscape model used. Section 4.2 explains the methods used to conduct this experiment. Section 4.3 presents the visualisations of the phenotypic landscapes and discusses how the connectivity differs between the setups, before concluding the chapter with a summary in Section 4.4.

4.1 Introduction

Chapter 3 presented an exploration of genotype-phenotype map (GPM) variants for GE. The results of this study highlighted πGE as a viable alternative to the traditional GPM. However this finding raised many questions. In GE the ideal outcome is to find the correct genotype that generates a solution to a given problem. πGE also has to find this genotype, but in addition to the correct node expansions in the derivation tree, the πGE genotype also has to encode the correct ordering of NTs to expand in the derivation tree. GE does not encounter this issue as it possesses a fixed-order GPM process. The addition of this order search drastically increases the size of the search space that needs to be traversed. Does this also carry with it some benefits not available to GE's fixed order mapping? This forms the key question of this chapter: how does the addition of variable order, and an increased search space, still allow for the competitive search performance of πGE?

The addition of variable order was stated above to lead to a drastic increase in the number of possible genotypes to search. Consider a simple demonstration: a GE derivation tree that starts off with three expansions. At each of these expansions, a grammar allows for two possible expansion choices, and each choice produces two NT's and consumes one NT, e.g., $E ::= EE|EE$. This results in a branching factor (the number of possible expansions at a tree node) at each expansion of two for GE, that leads to a requirement of 2^3 genotypes that map to the trees needed to fully explore the first three expansions, as shown in Figure 4.1.

πGE for these same three expansions presents a different situation. The first expansion has a single NT so πGE has a choice of one NT to expand. This NT then presents two possible choices for the grammar exactly like GE. This choice results in the consumption of one NT and the creation of two new NTs. Now for the second expansion πGE has the choice of two NTs, and then from this it will then have two choices from the grammar for

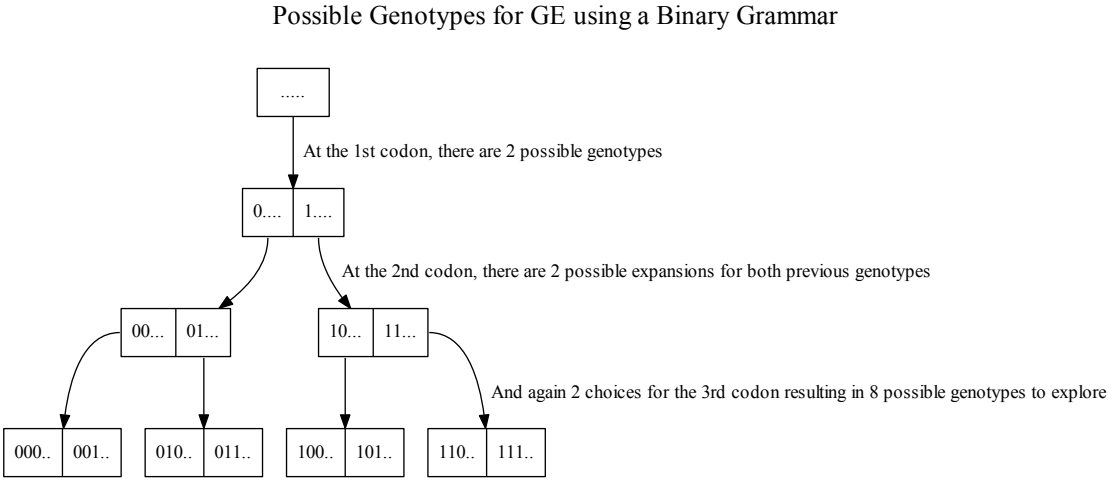


Figure 4.1: This figure shows the minimum possible genotypes needed by GE, to map to every possibility for the first 3 expansions of a grammar, consisting of the rule $E ::= EE|EE$. Note that with 3 expansions 8 (2^3) genotypes are needed.

whichever NT it selected. The second expansion consumed one NT but produced two new ones so there is now three NTs. The final of the three expansions for πGE results in a choice between the three unexpanded NT's. Finally the expansion of the NT results in a choice of two from the grammar. To model these three expansions πGE has to cover 48 ($1*2*2*2*3*2$) possible combinations of genotypes that can generate the eight possible derivation trees, as demonstrated in Figure 4.2. πGE has 40 more genotype combinations than GE, and if we were to carry this on to a fourth expansion GE would have 16 (2^4) possible trees that requires 16 genotypes, while πGE is facing 384 ($1*2*2*2*3*2*4*2$) possible genotypes to generate the same 16 trees.

Given that GE and πGE share a common grammar, the phenotypes that can be derived by both systems are identical, given the same amount of codons, where a πGE codon is a pair. This shared phenotypic space, called the phenotypic landscape, allows for the direct comparison of how each respective system can search the landscape. Modelling each algorithm's interaction with this landscape may shed light on how πGE can overcome the

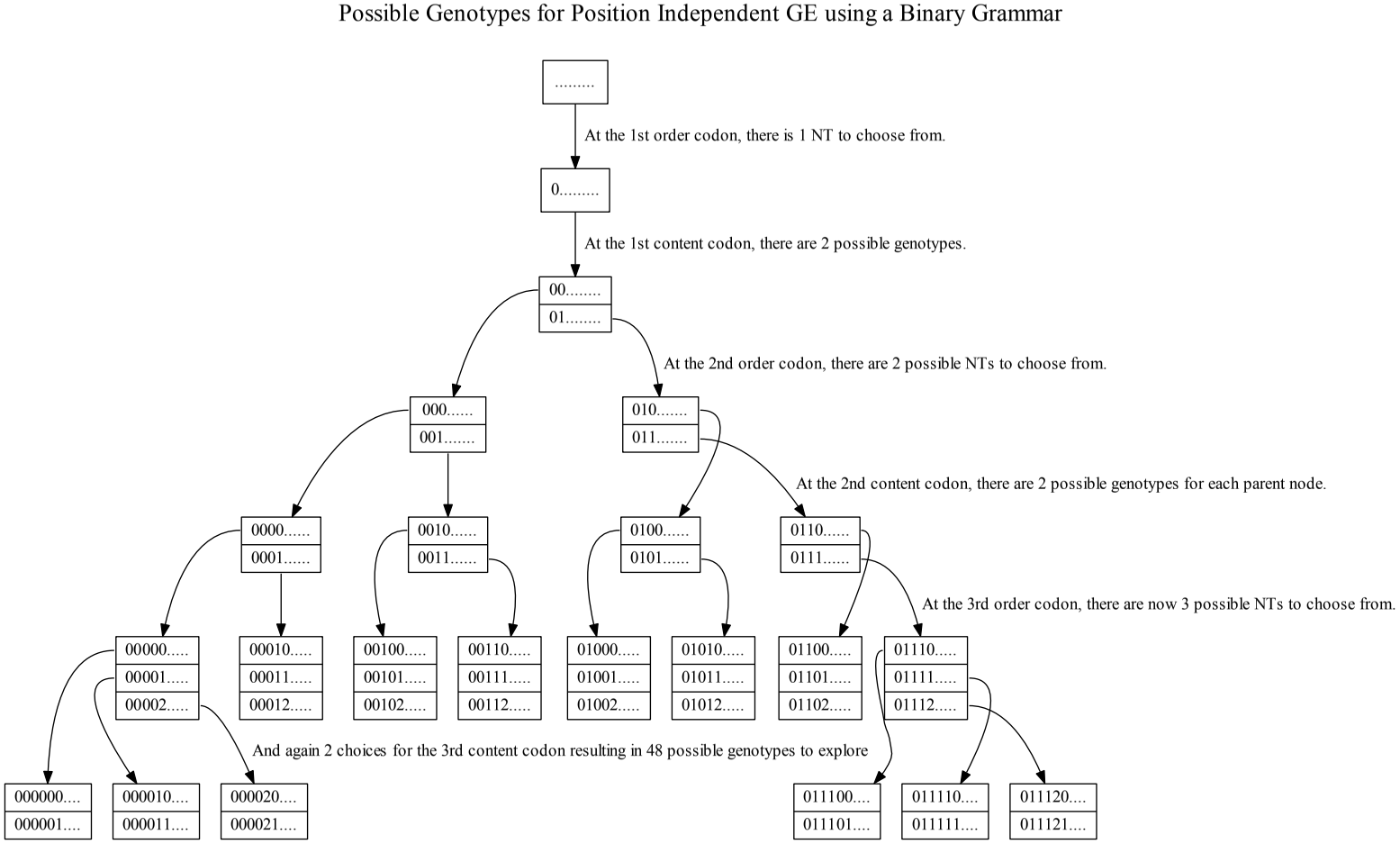


Figure 4.2: This figure shows the minimum number of genotypes needed by πGE to cover all possible mapping sequences for the first 3 expansions of a grammar, consisting of the rule $E := EE|EE$. Note that the grammar consumes 1 NT and produces 2 NTs to the possible locations πGE can choose for expansion. This leads to the NT list increasing by 1 per expansion. Due to this, 48 $(1 * 2 * 2 * 2 * 3 * 2)$ genotypes are needed to model the first 3 expansions. This results in only 8 derivation trees, but πGE has 48 possible paths to these trees.

massive overhead of order search. Koza and Poli [92] noted that understanding how an algorithm operates can be aided by visualising the program space, i.e., the phenotype space in GE and πGE.

4.1.1 Search Landscapes

Landscapes are an idea that can be employed to aid in the understanding of complex systems [72, 92]. Good visualisation of a landscape can facilitate study of an aspect of interest in the system. This visualisation may allow the user to gain increased understanding of the process in question, thus allowing for the deduction of solutions to observed behaviours in the system being examined. The doctoral thesis of Jones [71], focused on trying to understand and define landscapes in terms of EC methods. Jones stated:

> "A landscape is a metaphor by which we hope to imagine some aspect of the behaviour of an algorithm [71]."

He put forward the idea that an algorithm contains many landscapes dependent upon the operator performing search, such as mutation, and crossover in EC. Jones hypothesised that there was not one landscape, but rather a combination of these individual landscapes that provides a complete model of the system [72].

The landscape model, as outlined in detail in Jones' thesis [71], was employed for this study. What follows is a brief description. In the model, a landscape can be described as a five tuple (Equation 4.1).

$$L = (R, \phi, f, F, >_F) \tag{4.1}$$

- R denotes the representation space of the search algorithm.
- ϕ denotes the operator acting on the landscape.
- f a function that maps a multiset of R ($M(R)$) to F, the fitness space, $f : M(R) \rightarrow F$.

- F the fitness space
- $>_F$ represent a partial ordering over the fitness space.

The landscape L can be visualised as a labelled directed graph $G_L = (V, E)$, where the vertices V are a subset of $M(R)$, $V \subseteq M(R)$, and the edges E are a subset of the cross product of V, $E \subseteq V \times V$. An edge E between two vertices, v and w, can be said to exist if and only if there is a connection between v and w via an application of ϕ, $(v, w)\ \exists E \iff \phi(v, w) > 0$.

This model was further defined by Murphy et al. [108] for usage in a comparison of grammars in GE, and a similar definition is used for this experiment. The landscapes to be examined in this study are defined by the representation space R, that combines the chromosome space, and a GE GPM or πGE GPM resulting in the phenotype space. The phenotype space represents all the valid phenotypes that can be derived from the grammar within a given chromosome length, this is the object space O. Single int-flip mutation represents ϕ, and f is the GE fitness function. For this landscape the graph $G_L = (V, E)$ can be viewed as having set of vertices V, where $V \subseteq M(R)$ and $V \subseteq O$, meaning the vertices are genotypes, but also valid phenotypes. Given that GE and πGE share the same phenotype space, V will be the same regardless of what GPM is used. The edges between the vertices may differ, due to how ϕ interacts with the representation space R. Through these differences in E, GE and πGE will be compared.

4.2 Experimental Design

The aim of this experiment is to compare the phenotypic landscapes of GE and πGE with respect to the mutation operator. The experiment is to ascertain how πGE is able to display comparable performance to GE, even though it has to search a substantially larger search space, owing solely to the addition of a position independent GPM.

In this experiment GEVA [129] is extended to incorporate the "mutate and store" operation, as described in Section 4.2.1. This operation allows for the construction of the phenotypic landscapes. In EC an explicit solution to a problem is not known a priori. This results in GE using grammars that can recurse and grow solutions as large as needed. This recursiveness in the grammars leads to an infinitely large search space. Due to this, modelling of the landscape will require a chromosome limit to be set, to guarantee a complete landscape is achieved. Another factor that needs to be catered for is the increase in search space size πGE is subject to, as noted in Section 4.1. To deal with these issues both GE and πGE will be given sufficient chromosome length to express the same phenotypic space. GE and πGE will be given the same number of respective GE and πGE codons. Note that a πGE codon consists of two parts. Finally both GE and πGE will have their respective phenotypic landscapes compared using two example grammars outlined in Section 4.2.2.

4.2.1 Mutate and Store

Mutate and Store (MS), originally introduced by Murphy et al. [108], and modified to meet the requirements of this study, allows for exploration and mapping of any grammar's phenotypic landscape. MS maps the phenotypic landscape via single int flip mutation events, where by exactly one codon is mutated per mutation event. MS requires a fixed length chromosome of all zero codons. MS then takes the desired grammar and this initial chromosome and builds the phenotypic landscape. MS does this by starting at the first codon, and finding all the possible choices for that codon, by checking the grammar. Once all the possible choices for the codon are known, MS generates new genotypes for each choice and stores them in a neighbourhood. Having stored all possible neighbours, MS evaluates these neighbours and records what mutations resulted in chromosomes with valid phenotypes. These valid phenotypes are then added to the population for mutation at the next codon index. This process is repeated until all codon indices in the genotype

have been fully explored. Once this process is done, all the individual neighbourhoods of valid phenotypes are compressed into a single neighbourhood of phenotype connections. This final neighbourhood is then represented as a graph for analysis. Pseudocode for the algorithm can be seen in Algorithm 4.1.

Algorithm 4.1 The Mutate and Store algorithm. The algorithm starts at the first codon of the chromosome and finds all possible neighbours. For each of these neighbours the algorithm then find all the possible neighbours at the second codon and so on until all possible genotypes have been found. The algorithm then constructs a graph from all the unique phenotypes found in the neighbourhood.

```
Population pop {Population to store individuals with valid phenotypes}
Population neighbours {Population to store neighbours}
Individual init {Individual with all zero chromosome of size N}
List edges {Container to Store connections}
Add init → pop
for i = 0 → init chromosome length do
  for j = 0 → pop size do
    current_ind = pop get individual j
    Reset neighbours
    Generate and Store all possible valid genotypes for codon at index i → neighbours
    Get edges from neighbours → edges
  Add neighbours → pop
Generate graph info file from pop
```

MS 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 degeneracy and neutral mutations that GE can take advantage of. Degeneracy in GE is provided by the mod rule. Consider the following: a GE codon valued 62 is mutated to 64. When this codon is applied to a binary grammar rule, the mutation results in no change to the expansion of the tree. Removing the degeneracy is important as it significantly limits the number of possible phenotypes. If MS allowed codons values between 0 and 255, and a chromosome was limited to a size of just 3 codons, that would results in over 16 million possible genotypes that would need to be explored regardless of the arity of the grammar. MS when investigating a grammar with an arity of 2, leads to only 8 possible genotypes

to explore 3 codons.

πGE presented an extra layer of exploration that needed to be added to MS, for the mapping of πGE phenotypes. In the above explanation, at each codon of the genotype, the grammar was consulted for the possible expansion choices, for that codon. For πGE this process took place at every even valued codon index. πGE required that for every odd codon index the NT list size was consulted, so that every possible expansion point in the partial πGE derivation tree be explored. This resulted in an increase in the number of possible genotypes, and restricted the size of genotype that could be explored in this study. The degeneracy for the expansion order codons was removed, as it was for standard GE codons.

4.2.2 Grammars

Initial setup tests highlighted a computational constraint for grammar usage with πGE. Usage of grammars with high arity production rules resulted in MS not being able to model the phenotype landscape of πGE. This was due to the increase in possible genotypes that πGE has to explore, as explained in Section 4.1, and in Figures 4.1 and 4.2. A simple grammar was designed to enable modelling to take place. This grammar (Figure 4.3a) is a binary grammar that has two choices for every rule. The grammar shares the same core rules as the commonly used symbolic regression grammar (Figure 4.3c). Finally in order to provide an understanding of how the landscapes scale, an enhanced version of the initial grammar is applied (Figure 4.3b). This grammar adds a third variable to the initial grammar. All grammars are displayed in Figure 4.3.

```
<expr> ::= <op> <expr> <expr>  (0)
         | <var>               (1)

<op> ::= +                     (0)
       | *                     (1)

<var> ::= x0                   (0)
        | 1.0                  (1)
```

(a) Example grammar 1, binary grammar.

```
<expr> ::= <op> <expr> <expr>  (0)
          | <var>              (1)

<op> ::= +                     (0)
       | *                     (1)

<var> ::= x0                   (0)
         | x1                  (1)
         | 1.0                 (2)
```

(b) Example grammar 2, 3 variable variant.

```
<expr> ::= <op> <expr> <expr>  (0)
          | <var>              (1)

<op> ::= +                     (0)
       | *                     (1)
       | -                     (2)
       | \                     (3)

<var> ::= x0                   (0)
        | x1                   (1)
        | 1.0                  (2)
```

(c) Example grammar for Symbolic Regression.

Figure 4.3: The figure presents the grammars used for the study. Figure 4.3a shows the initial binary style grammar used in the first experiment. Figure 4.3b displays the extended three variable grammar used in experiment 2 to show how the phenotypic connectivity scales. Figure 4.3c show a grammar used for a symbolic regression, used in other chapters of this thesis. This grammar is provided to show that the example grammars used in the experiment are related to actual grammars used in GE to solve problems.

4.3 Phenotypic Landscape Visualisations

This section visualises the phenotypic landscapes of both GE and πGE. Firstly both representations are examined from the compressed viewpoint of adjacency matrices. These matrices show which of the possible phenotypes are connected via a single mutation event. The adjacency matrix representation disregards any possibility of phenotypes being connected to another phenotype in more than one way. Following this the graphs of the phenotypic landscapes for both are displayed and discussed. The graph representation allows for visualisation of multiple connections between phenotypes, e.g., where a pair of phenotypes are connected via both order, and content codon mutation events in πGE. The two visualisation methods are performed on the example grammars shown in Figures 4.3a and 4.3b.

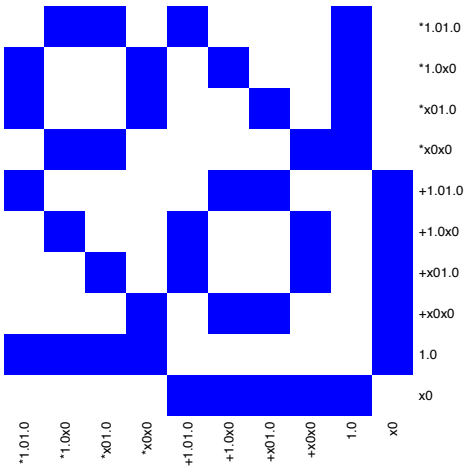
4.3.1 Experiment 1 - Simple Grammar

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. Adjacency matrices are good for showing clearly what each phenotype is connected to, as graphs can be cluttered and difficult to interpret. An algorithm whose phenotype space has a densely populated adjacency matrix will have a greater amount of freedom moving from phenotype to phenotype. This freedom can aid in increasing the search performance.

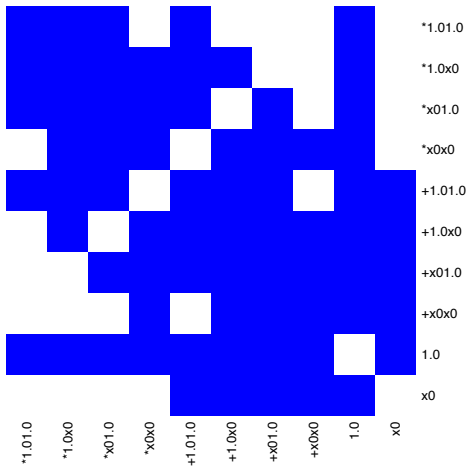
Figure 4.4 shows the adjacency matrices for all possible phenotypes achievable with a maximum derivation tree of 7 NTs, for the simple grammar (Figure 4.3a), for both types of GPM. This limit of derivation tree size was the highest common tree size that could be generated for GE and πGE on the hardware available for this experiment. Figure 4.4a displays the connections between phenotypes for GE, while Figure 4.4b shows the connectivity of πGE for the same phenotypic landscape. Comparing Figures 4.4a and 4.4b it can be seen that πGE's phenotype space is more densely connected than GE's phenotype space. GE presents a total of 28 connections, whilst πGE has 70 connections among the same phenotypes.

It is interesting to see how GE has no neutral mutation due to MS, but with the addition of order to the GPM, πGE exhibits neutral mutation on nearly all phenotypes. The neutral mutations are evident along the diagonal, from the bottom right corner of the matrix to top left corner. Figure 4.4c shows the connections between phenotypes exclusive to πGE, and Figure 4.4d shows these 42 connections highlighted amongst all the πGE connections including the 42 connections common to GE.

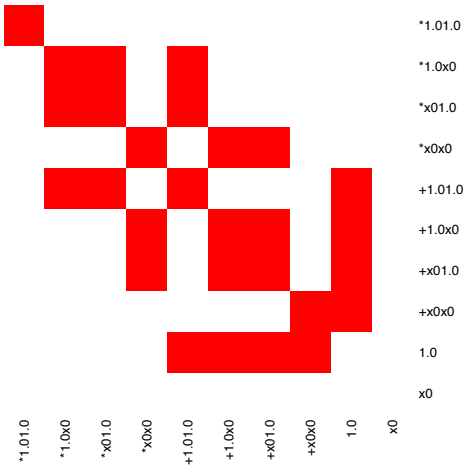
Phenotypes of a single variable, such as $(x1)$, cannot exhibit neutral mutations other than via the mod rule, eliminated earlier. This is due to the NT list for such a tree never exceeding a size of one, thus the left-most non terminal is always picked. When the tree



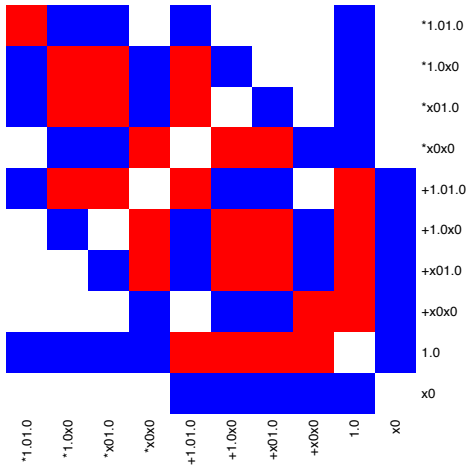
(a) Adjacency matrix showing connections between phenotypes possible with GE.



(b) Adjacency matrix showing connections between phenotypes possible with πGE.



(c) Adjacency matrix showing connections only possible with πGE.



(d) Adjacency matrix for πGE. The connections exclusive to πGE highlighted in blue.

Figure 4.4: Comparison of phenotype adjacency matrices of GE and πGE on the simple binary style grammar shown in Figure 4.3a. In 4.4a the adjacency matrix for GE is shown, whilst 4.4b shows the matrix for πGE. Figure 4.4c displays a matrix containing the phenotype connections unique to πGE, note the presence of neutral mutations along the diagonal of the matrix, that are only possible by πGE's evolvable order. Finally 4.4d shows the adjacency matrix for πGE with connections exclusive to πGE highlighted in red.

sizes of individuals grow to permit a varied ordering in πGE, it is seen that a mutation in the order codons results in neutral mutations. This behaviour is not possible in GE, as MS removes codon degeneracy, and GE doesn't possess variable orderings in the GPM.

The adjacency matrix representation is good for quickly showing connectivity but it lacks the ability to show multiple connections between the same phenotypes. In order to gain a more in-depth understanding of the landscape, and how each GPM interacts with it, we must be able to visualise the whole landscape and all its connections. This is important in the case of GE and πGE, as πGE can mutate from phenotype to phenotype using both the mutations that effect the productions of the NT like in GE, and also the connections made possible via mutating the order of the tree expansion in the GPM. These mutations may not be represented on the adjacency matrix, as it only takes into account the existence of a connection between a pair of phenotypes, and not how many connections exist.

Figure 4.5a shows the phenotypic landscape graph for GE. Each vertex represents a valid phenotype and each edge represents a connection between two phenotypes via a single mutation event. It is interesting that every vertex has a degree of four and that there are no extra paths between vertices. There are no neutral mutations visible as this is not permitted via the MS operator.

Figure 4.5b shows the πGE landscape over the same phenotypic space. The first thing that stands out when comparing the two setups is the increased number of edges between vertices. These vertices are purely from the addition of the variable ordering in πGE. The neutral mutations seen previously in Figure 4.4b, added by the variable ordering can be seen in vertices like $(*1.01.0)$, with a little loop edge shown. The increased number of edges lead to an increased degree for each vertex. This comparison is summarised in Table 4.1 and also shown in Figure 4.6, where the edges common to GE and πGE are displayed in blue, while the πGE exclusive edges are displayed in red.

(a) The connectivity of GE using the grammar in Figure 4.3a.

(b) The connectivity of πGE using the grammar in Figure 4.3a.

Figure 4.5: The connectivity of GE (Figure 4.5a) and πGE (Figure 4.5b), using the grammar in Figure 4.3a are displayed. Each vertex represents a phenotype and the edges represent the ability for πGE to move from one phenotype to another in a single mutation. When compared to GE (Figure 4.5a) it can be seen that πGE has many more edges between vertices. It is clear that πGE has more possible connections between phenotypes. Comparing vertices such as $(*1.01.0)$, in πGE edges loop back to these vertices denoting a neutral mutation that is not seen in GE.

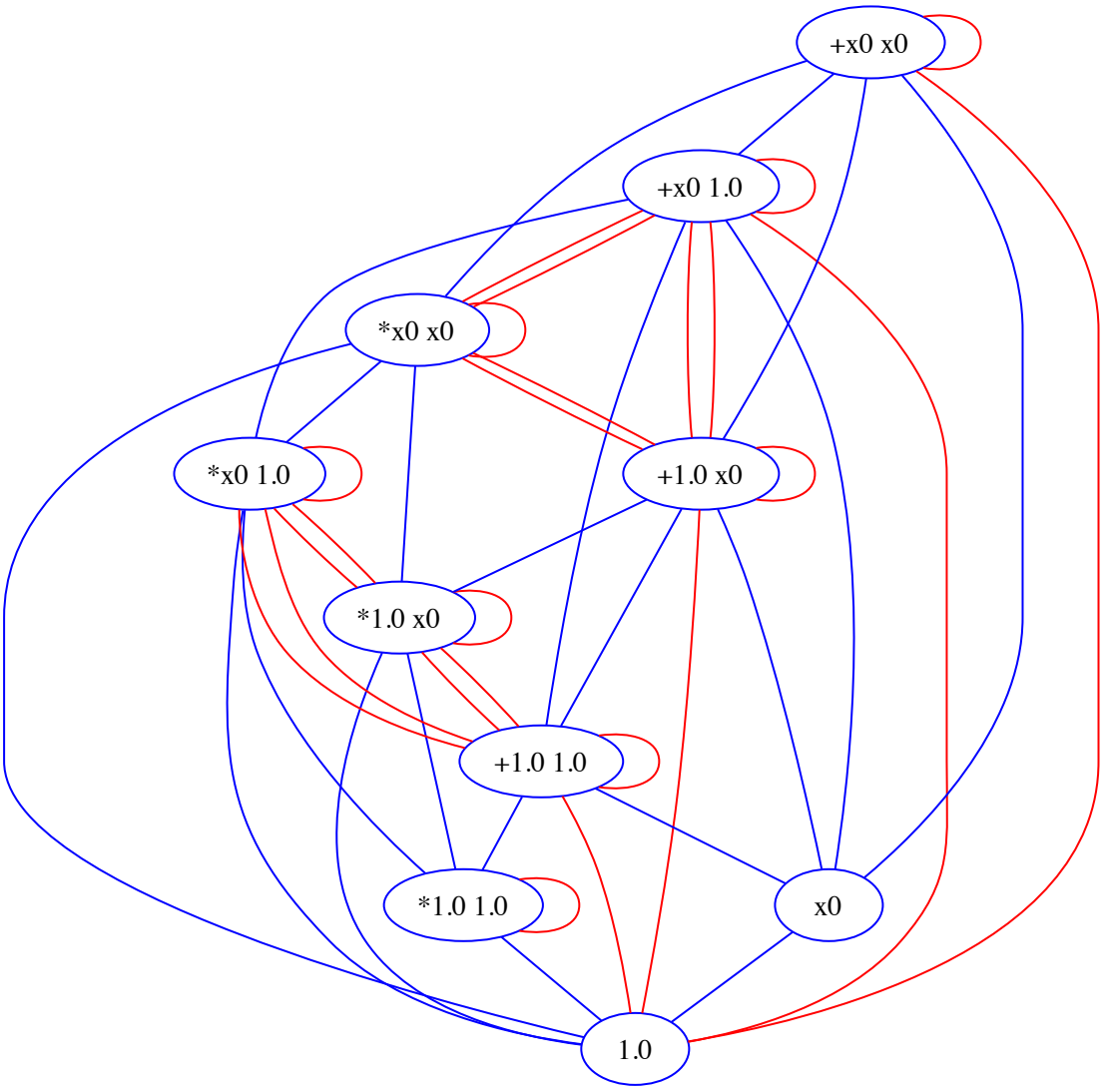


Figure 4.6: The connectivity of πGE using the grammar in Figure 4.3a is displayed. Each vertex represents a phenotype and the edges represent the ability for πGE to move from one phenotype to another in a single mutation. The edges in red represent the connections that are only available to πGE. The blue edges shown the common paths shared by πGE and GE. This graph is useful to fully understand the addition to connectivity that a variable ordering in the GPM adds. In Figure 4.4d a total of 70 connections are visible for πGE, but this increases to 90 connections in this figure. These extra connections are a direct result of the πGE GPM, as the same phenotypes being connected in multiple ways is due to the addition of evolvable order. The number of connections common to GE is the same in both figures.

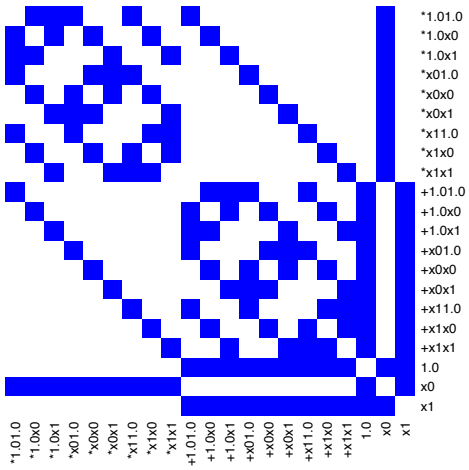
The reason for examining the actual graphs of the adjacency matrices becomes evident when the number of connections are compared. In Figure 4.4b a total of 70 connections are visible for πGE, but this increases to 90 connections in Figure 4.5b. These extra connections are the result of mutations to the tree content like GE, or a mutation of the ordering of the GPM, resulting in the same phenotypes being connected in multiple ways.

4.3.2 Experiment 2 - Expanded Grammar

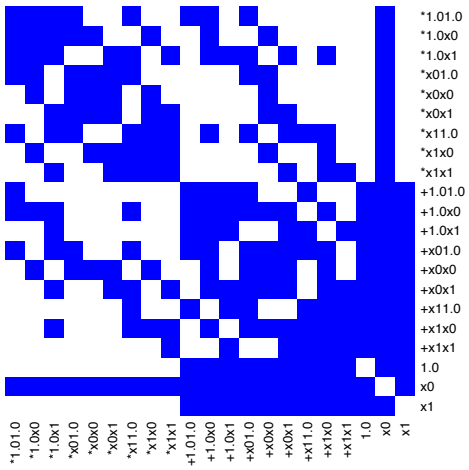
The second examination of the connectivity of πGE versus GE was performed using the three variable version of the binary grammar (Figure 4.3b). This grammar variant was used to show how the landscapes scale. The addition of just one extra variable, increasing the total number of variables from two to three, results in a 110% increase in the number of vertices needed to represent the phenotypic landscape (from 10 to 21).

Fig. 4.7 shows the adjacency matrices for the expanded grammar (Figure 4.3b), for both types of GPM. Figure 4.7a shows the connections between phenotypes for GE. Figure 4.7b shows the connectivity of πGE. Figure 4.7c shows the connections between phenotypes exclusive to πGE, and Figure 4.7d shows these connections highlighted amongst all the πGE connections including the ones common to GE. The findings from the initial study showed an increased number of connections in πGE over GE. Also πGE was shown to add neutral mutations via its variable order. These finding are again present with the enhanced grammar with πGE having 42 more connections than GE.

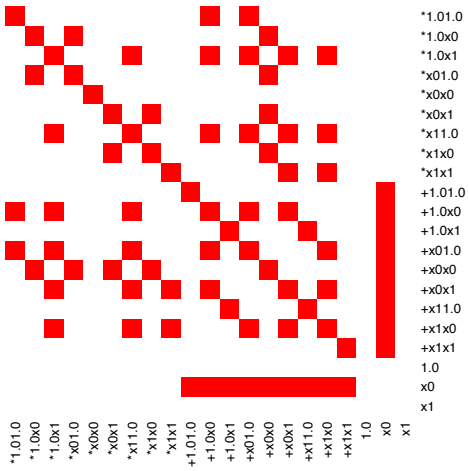
Focusing on the graph representation of the landscape, it can be seen how the size of the landscape has increased. Figure 4.8 shows the phenotypic landscape graph for GE. Each vertex represents a valid phenotype and each edge represents a connection between the phenotypes via a single mutation event. Figure 4.9 shows the πGE landscape over the same phenotypic space. The two setups still produce a vastly different number of edges between vertices. The neutral mutations and multiple paths between the same



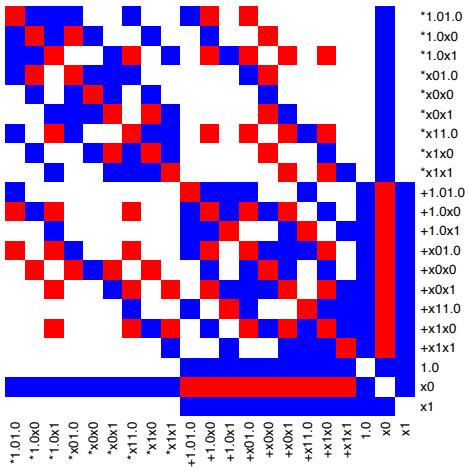
(a) Adjacency matrix showing connections between phenotypes possible with GE.



(b) Adjacency matrix showing connections between phenotypes possible with πGE.



(c) Adjacency matrix showing connections only possible with πGE.



(d) Adjacency matrix for πGE. The connections exclusive to πGE highlighted in blue.

Figure 4.7: Comparison of phenotype adjacency matrices of GE and πGE on the expanded binary style grammar shown in Figure 4.3b. In 4.7a the adjacency matrix for GE is shown, whilst 4.7b shows the matrix for πGE. Figure 4.7c displays a matrix containing the phenotype connections unique to πGE, note the presence of neutral mutations along the diagonal of the matrix. Finally 4.7d shows the adjacency matrix for πGE with the connections exclusive to πGE highlighted in red.

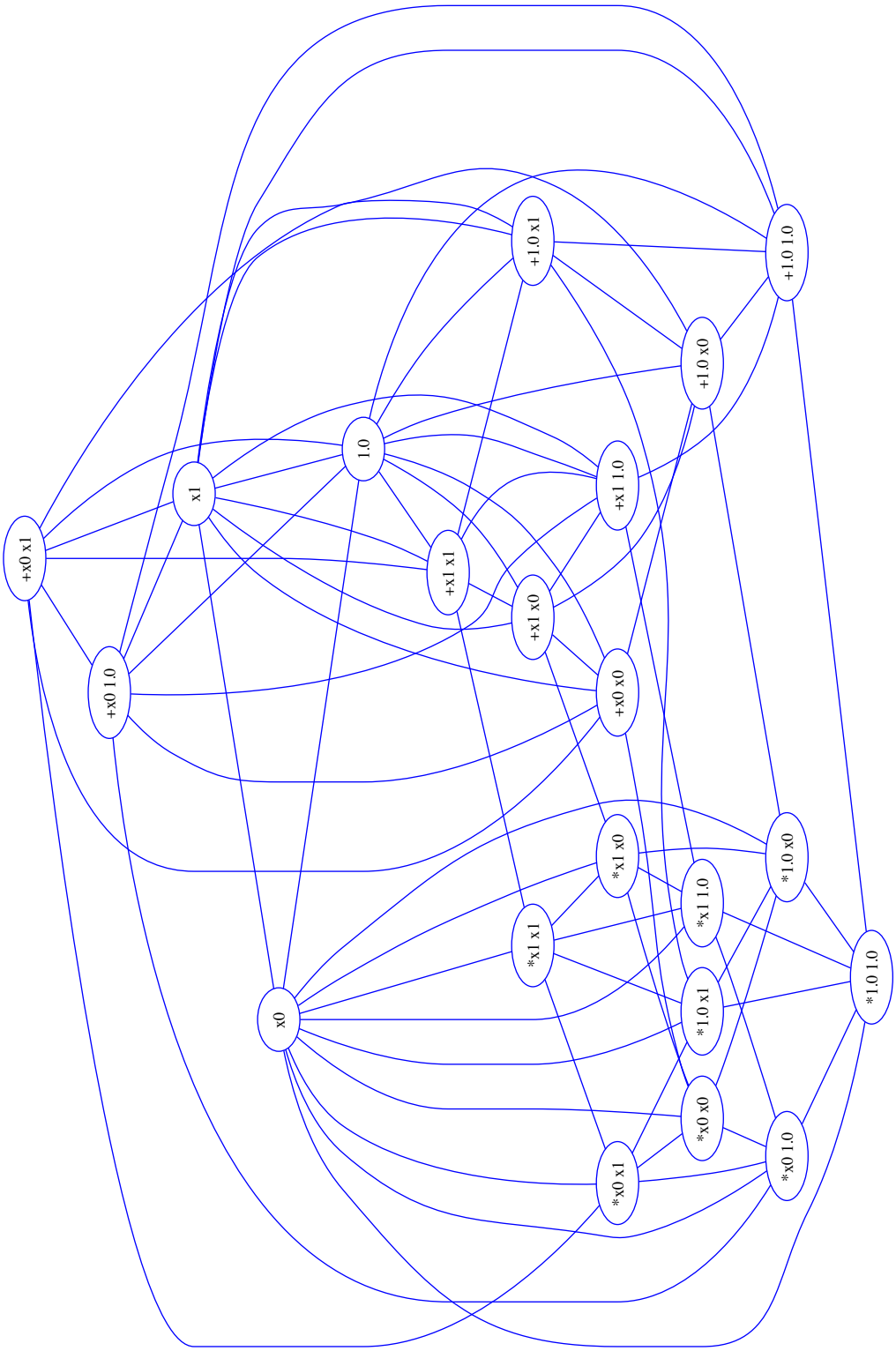


Figure 4.8: The connectivity of GE using the grammar in Figure 4.3b is displayed. Each vertex represents a phenotype and the edges represent the ability for GE to move from one phenotype to another in a single mutation.

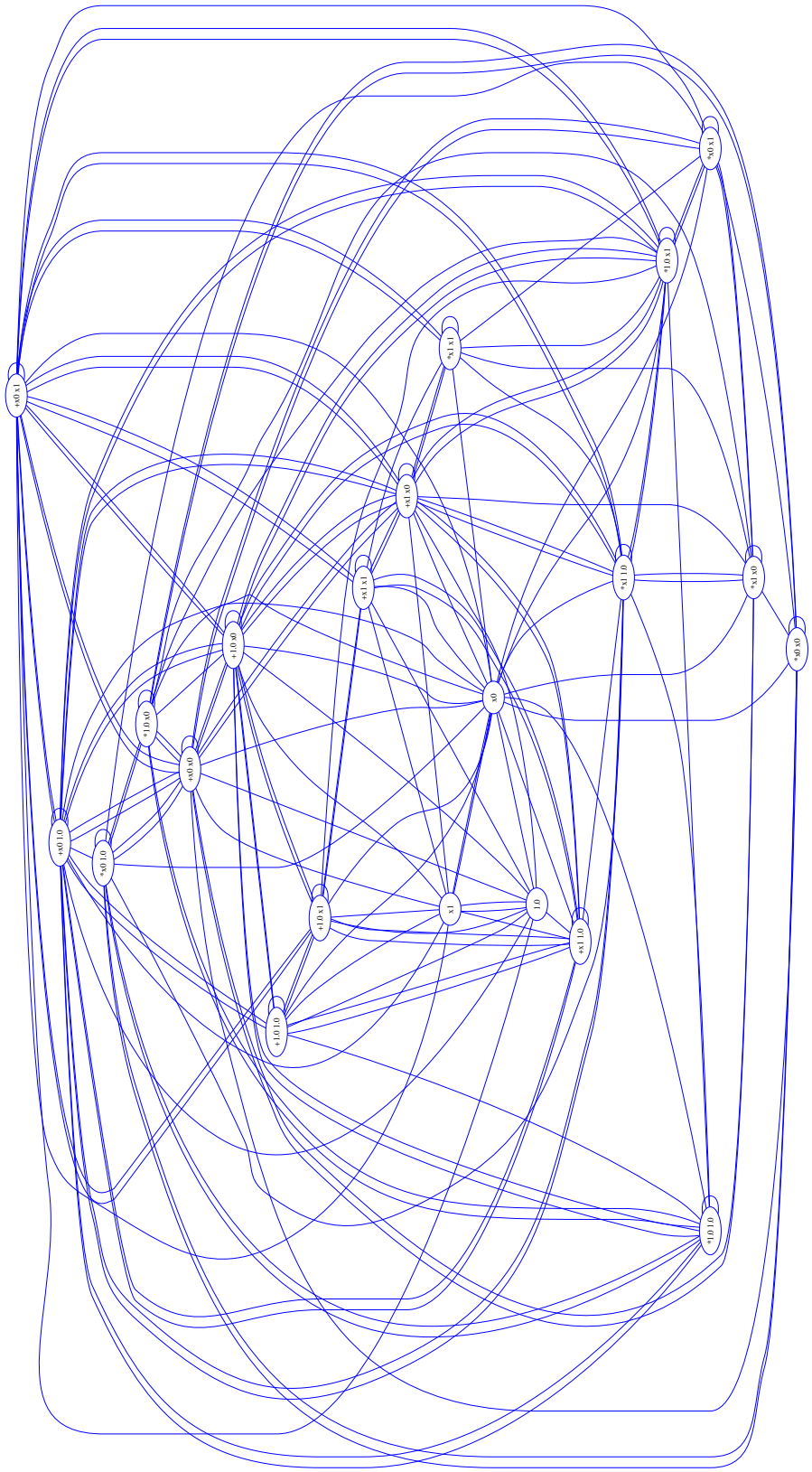


Figure 4.9: The connectivity of πGE using the grammar in Figure 4.3b is displayed. Each vertex represents a phenotype and the edges represent the ability for πGE to move from one phenotype to another in a single mutation. When compared to GE (Figure 4.8) the findings of the previous study in Section 4.3.1 are repeated. It can be seen that πGE has many more edges between vertices. Neutral mutations are evident at nodes such as (* 1.0 1.0).

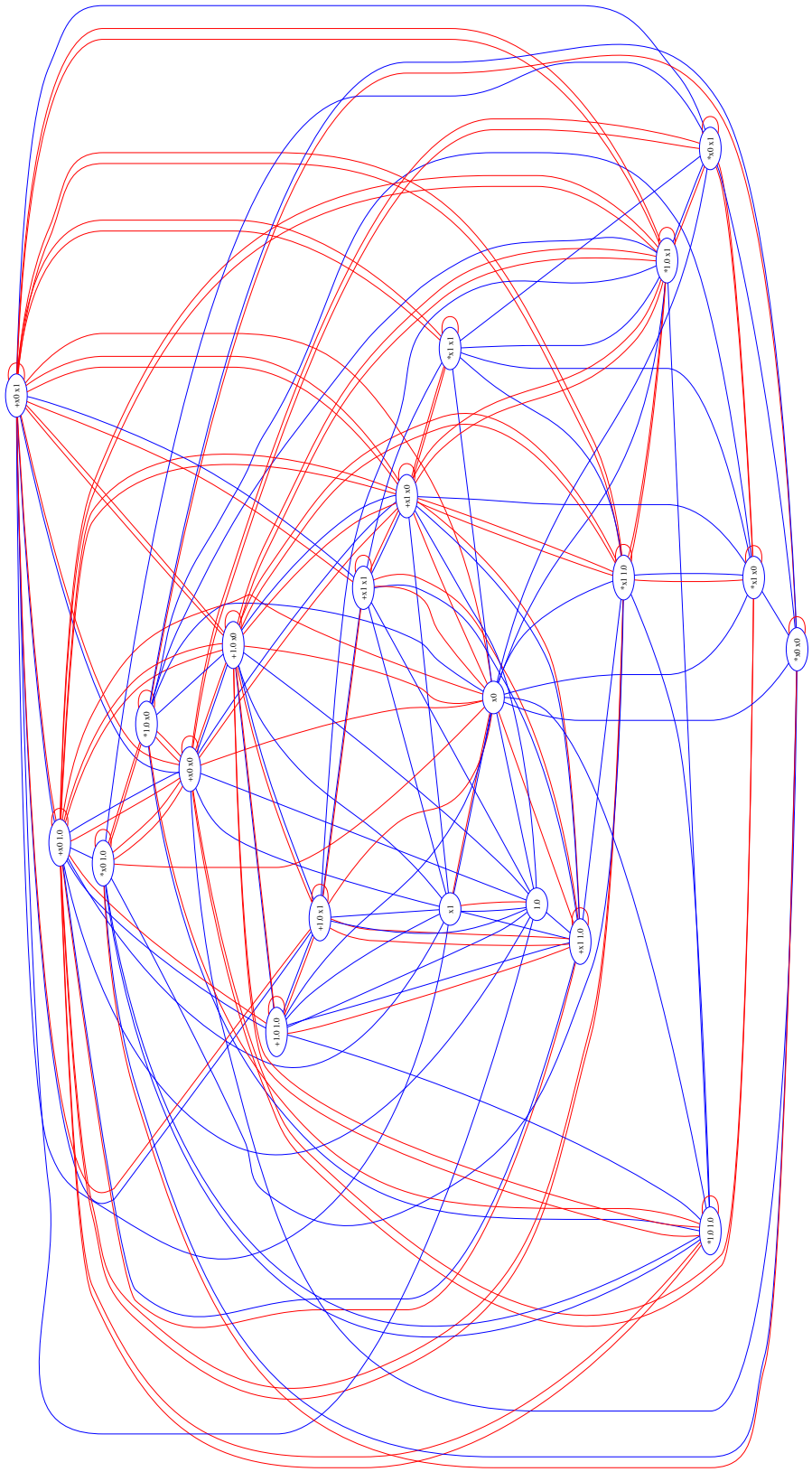


Figure 4.10: The connectivity of πGE using the grammar in Figure 4.3b is displayed. Each vertex represents a phenotype and the edges represent the ability for πGE to move from one phenotype to another in a single mutation. The edges in red represent the connections that are only available to πGE. The blue edges shown the common paths shared by πGE and GE. The graph once again displays the increase in connectivity that the variable order GPM in πGE provides.

vertices also remain present as expected. This comparison is summarised in Table 4.1 and also compared against the initial setup. The table shows that the addition of position independence to the mapping process increases the connections between phenotypes in the landscape. With both the grammars there is slightly more than a 100% increase in the number of connections between phenotypes. The average degree of a vertex in the graph does't quite match the increase in nodes, indicating that some nodes get more than a doubling of edges. Figure 4.10 displays a hybrid graph to help show how the addition of variable order to GE produces a more connected landscape. In the graph the edges common to GE and πGE are displayed in blue, while the πGE exclusive edges are displayed in red.

Table 4.1: Table outlining features of the connectivity graphs shown for both experiments. The table notes the total number of vertices and edges, the total degree of the graph and the average degree for a vertex for each GPM approach on both grammars used. From the number it is clearly evident that πGE has more connectivity than GE.

Graph Features	**GE Grammar 1**	**πGE Grammar 1**
$\#\ Vertices$	10	10
$\sum_{i=1}^{n} DegreeVertex(i)$	42	90
$\#\ Edges$	21	45
$\overline{VertexDegree}$	4.2	9.0
	GE Grammar 2	**πGE Grammar 2**
$\#\ Vertices$	21	21
$\sum_{i=1}^{n} DegreeVertex(i)$	98	198
$\#\ Edges$	49	99
$\overline{VertexDegree}$	4.67	9.43

4.3.3 Limitations

This study is limited to the exploration of the phenotypic connectivity in terms of the mutation operation. The decision to focus only on mutation was made based on several factors. The most important reason behind this was complexity. The methods used to

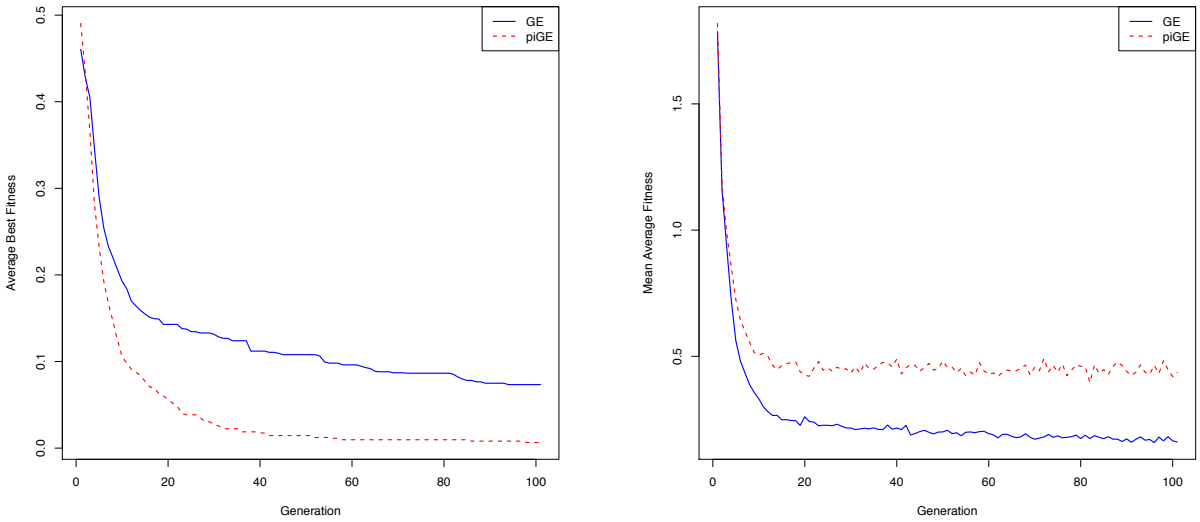
generate the landscapes above pushed the available resources to the limits. MS when given more than a chromosome length used in this experiment would consume the $24GB$ of ram available when trying to store possible genotypes to explore. Similarly if MS was given a grammar of higher arity than the ones used in this experiment, the ram resources were again exhausted with the increase in genotypes possible, that need to be investigated.

Crossover, being the other main operation used by GE, was not explored due to the fact that it represents such a complex landscape to model. It has been shown that πGE has a significantly increased search space with the addition of the evolvable order (Section 4.1). Modelling crossover in this space presents a challenge not only in computational resources but also in complexity. A crossover event in πGE can happen at any codon, order or content, leading to large combinatorial increase. Consider two genotypes of 10 codons in size. A single point crossover with variable crossover points, such as the method used in GE, using these two genotypes would lead to the exploration of nearly 100 possible individuals. The average vertex degree in this study is between 4 and 10 for just mutation. The addition of crossover to MS would exceed the computational power available, so a reimplementation of MS would be required.

4.3.4 Discussion

The goal of this experiment was to examine phenotypic landscapes of πGE and GE. Visualising the landscapes was done with the intention of gaining further insight into how the πGE algorithm works. πGE introduces an increase to the search space of genotypes with its variable ordered GPM. With this increase in the search space how can πGE still maintain good search performance? πGE and GE were compared using a simple grammar. From this comparison certain aspects of the πGE algorithm became evident. πGE has a much more connected phenotypic landscape, that allows for a better ability to explore the search space. πGE also introduces pure neutral mutations via the varying of the expansion

order of the GPM. These neutral mutations are not possible with GE except via codon degeneracy.



(a) Average Best Fitness.

(b) Mean Average Fitness.

Figure 4.11: This figure displays the relative performance of both πGE and GE on the quartic polynomial symbolic regression problem. The graphs show performance over 100 runs using the grammar in Figure 4.3a. The problem was performed with no crossover to ensure that only the mutation connectivity seen in this chapter would be used for search. Figure 4.11a shows the average best fitness of the population averaged over the 100 runs. πGE is showing performance advantages over GE. Figure 4.11b shows the mean average fitness of the 100 runs. It can be seen that πGE maintains a higher average fitness within its population providing greater fitness diversity. GE's mean average fitness is much closer the mean best fitness, while πGE maintains a mean average fitness of around 0.5, while its mean best fitness tends to 0. This can be linked to how connected the πGE phenotypic space is as it shows the population does not converge to a local optimum. πGE also shows a 96% success rate in comparison to GE's 90%.

πGE increases the genotypic search space that it must navigate by introducing an evolvable expansion order as shown in Section 4.1. This evolvable expansion order also brings with it some unforeseen benefits. The addition of variable order to the GPM has resulted in an algorithm that exhibits far more connectivity in the phenotypic space than standard GE. πGE's variable order GPM allows for phenotypes to neutrally mutate back to

themselves. Neutral mutations of this nature are only seen in GE through the degeneracy gained via the use of codon values and the mod rule in the GPM. πGE will not only enjoy the benefit of standard GE's degeneracy, but will also gain additional degeneracy via the use of a mod rule to control the variable order GPM, and the natural neutral mutations it gains from the usage of the variable order GPM.

Figure 4.11 shows the performance of πGE and GE on 100 runs of the quartic polynomial symbolic regression problem, using the reduced binary grammar of Figure 4.3a. GE and πGE where given 200 and 400 codon respectively to allow for that same landscape to be generated. The rest of the setting where in line with the setting used in Chapter 3. The runs were performed using only mutation so as to use only the connectivity shown in this chapter for search. The graphs show that πGE exhibits better performance than GE in terms of average fitness and also maintains a greater range of fitness values in the average population fitness. πGE also shows a 96% success rate in comparison to GE's 90%. These runs were performed to demonstrate that the grammar used to model the phenotypic landscapes is capable of solving a problem. The runs also are in keeping with the results reported in Chapter 3.

Redundancy, neutral mutation, and degeneracy can have a drastic effect on the performance of an EA [4, 86, 87, 124, 142, 143, 144, 145]. This may provide more insight into how πGE can maintain performance under the strain of order search. πGE presents a very redundant GPM, with order redundancy, and the codon redundancy on both the NT choice and tree expansion choice. Ebner et al. [29] have shown that a GPM that exhibits high redundancy increases evolvability (the ability for random variation to produce a fitness improvement). The added redundancy in the πGE GPM also leads to a substantial amount of neutral mutation, as seen in the number of potential genotypes that MS explored to generate the phenotypic landscapes. This explosion of genotypes is the reason why πGE could only be modelled on such a limited landscape.

Kimura [86, 87] argued that in the neutral theory of evolution, most mutation events are neutral mutations, and that only a small number of non-neutral mutations are actually beneficial. Kimura also studied neutral networks in nature, and noted how most mutations simply navigate this neutral network until a beneficial mutation occurs. Ebner et al. [30] examined the idea of neutral mutations and neutral networks in GPM. Shackleton et al. [152] and Shipman [156] also explored this avenue of research into redundancy and neutral mutations, highlighting again how a many to one GPM was beneficial over a one to one mapping due to redundancy.

Many others have investigated redundancy in GPM and found it to be a key component to driving evolution. Kargupta [73] provided a theoretical examination of a simple redundant mapping in a search space, that underpins a lot of the research in this area. Rothlauf [142, 143, 144, 145] has also undertaken significant research in the area as have Banzhaf [4] and O'Neill [124]. Rothlauf believed that redundancy was a good this only if the optimum solution is over represented in the search space. This represents a sampling of research in the area, and in the majority of cases redundancy and neutral mutation is viewed as a beneficial property in a mapping from genotype to phenotype. πGE has a larger search space than GE to explore, and the added connectivity, redundancy and neutral mutations that the evolvable ordering of the GPM provided πGE is the only difference between the two algorithms. Therefore it is argued that the increased connectivity and redundancy in πGE, must be responsible for πGE's ability to achieve comparable performance to GE.

4.4 Summary

Given the search overhead the evolvable order GPM gives to πGE, it was decided to investigate this to see what impact it had on πGE. Did the order add anything to πGE? This chapter undertook the study of the phenotypic connectivity of πGE to gain further understanding into how πGE works.

Visualising the phenotypic landscape of single mutation events in GE and πGE using a graph representation, it was shown that the addition of order led to a significant increase in connectivity for πGE. A more densely connected algorithm has the benefit of easier movement within the search space.

The addition of order also added degeneracy and neutral mutation, unlike GE that relies upon the mod rule to provide this. 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 does not represent a problem for πGE to search the solution space. In fact the increased redundancy allows for added neutral mutation which is the driving force behind evolution.

In the next chapter we go on to explore the orders present in a πGE population during evolution. Some measures are presented that monitor the orders present in the population and examples of the orders seen are also presented.

Chapter 5

Examining Order in πGE

πGE uses 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 chapter investigates the behaviour of the orders in a πGE population during evolution. By comparing the expansion order of the πGE derivation process to known orders it is possible to see if πGE exhibits a bias towards any sort of fixed ordering in the derivation tree expansion process. It is concluded that within πGE we do not evolve towards a specific order but rather as distribution of orders, and that πGE is using its variable order to sample the solution space. This chapter presents a more in depth examination of work presented by Fagan et al. [35].

The chapter is structured as follows. Section 5.1 provides an introduction to the problem outlining the need for investigation into the order of expansion in πGE. In Section 5.2, the suite of metrics used to measure order in πGE are introduced before being outlined in detail in Sections 5.2.1 and 5.2.2. Following on from the metrics, Section 5.3 outlines the experimental setup for the study, before the results are shown in Section 5.4. Finally the chapter is concluded with a summary of the work carried out in Section 5.5.

5.1 Introduction

O'Neill et al. [127] proposed in their original πGE paper that introducing an evolvable expansion order 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 [66]. O'Neill et al. hypothesised that by breaking the strict left to right expansion linkage in GE, and instead adopting an approach where the expansion order was not known a priori, πGE would allow for movement of better building blocks within the derivation tree than what is possible within GE. This might mean using an order that only slightly deviates from the fixed GE order or it might come about from an order with no distinguishable known ordering.

Chapter 4 investigated how πGE could still perform on par with GE, considering the massive search overhead πGE has to overcome due to this order. It was observed that the addition of order to the derivation process led to an increase in connectivity in the search space allowing for more neutral mutation and easier traversal of the search space. While there have been several studies pertaining to πGE [25, 38, 39, 40, 42, 127], this chapter presents the first in-depth look into the behaviour of the expansion order in πGE.

This chapter aims to take the investigation of πGE into the domain of what is happening to the orders during a run of the πGE system. 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 some form of metrics must be used to determine the distance from a known order. A suite of Order Bias Distance Metrics are proposed for this undertaking, where πGE orders are monitored and compared against known expansion orders, and then used to examine how πGE behaves.

5.2 Order Bias Distance Metrics

Before any metric can be defined it must first be decided what expansion orders are possible for a derivation tree in GE. A taxonomy of expansion orders is presented below. Expansion orders can be classified as deterministic, where the behaviour is known a priori, or non-deterministic, where the order cannot be determined before the expansion process. There are four main types of expansion orders possible in GE.

- *Depth-First Order* is a deterministic tree expansion order where the tree is expanded as deep as possible before coming back up the tree to go down all remaining branches of the tree. This method can be performed in two variations. The Tree can be expanded with left-most first priority, or with right-most first priority. GE adopts a left-most first, depth-first mapping approach. GE starts at the root and then recursively goes down the left-most child node, until no child nodes are present at which point it goes up a level and tries the next left most child. This process of expansion was outlined in detail in Section 2.3.1.

- *Breadth-First Order* is another deterministic tree expansion order. In breadth-first order the tree is expanded at each level as fully as possible before moving on to the next level of the tree. This method is similar to depth-first in that the expansion can be done with left-most or right-most priority. A left-most breadth-first mapping approach was outlined in detail in Section 3.2.2.

- *Heuristic Order* is an ordering that can be deterministic or non deterministic. The expansion of the tree is under the control of some user defined rule that dictates the expansion order of the tree, such as πGE. These heuristics can take the from of expanding certain types of nodes first, or perhaps switching between depth-first and breadth-first every couple of expansions for example. Some elements of randomness can be added that would make the process a non deterministic one.

- *Random Order* is a non-deterministic ordering of tree expansion. Random order randomly picks any possible expansion site, expands the derivation tree. This process is repeated until no expansion sites remain. This approach retains no knowledge of previous orders. Section 3.2.4 outlines in detail a random order approach to derivation tree expansion order.

For the purpose of investigating what orders πGE is using, this study will focus on determining a distance from known orders. This dictates that only Depth-First and Breadth-First orderings will be considered from the above taxonomy, as they present the only guaranteed deterministic orderings of tree expansion. In the following sub-sections the two metrics used in this study to monitor these orderings are presented and explained in detail.

5.2.1 Depth First Order Bias

We propose Depth First Order Bias (DFOB) as a measure that shows how far away from a depth first derivation order a πGE order is. Depth first order of expansion is important as it represents the standard GE expansion order of a derivation tree. The measuring of a distance from a depth first ordering is possible, by exploiting a feature of the πGE algorithm's implementation.

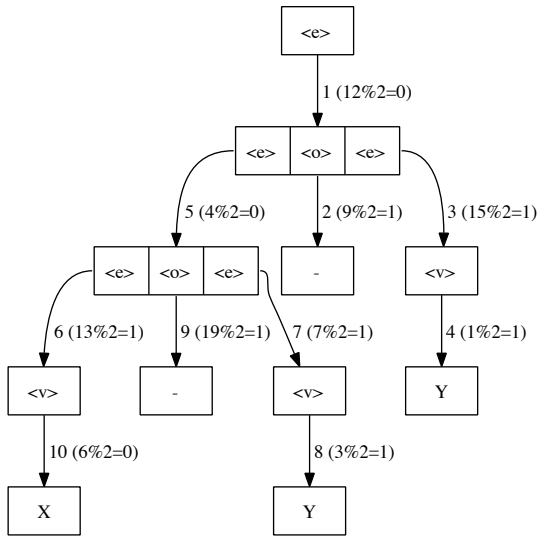
In πGE, all non terminals encountered during the derivation process are added to a list of possible expansion sites. At each step in the derivation process a selection from this list is made, and this selection is controlled by the chromosome. When a non terminal is expanded any non terminals generated from the expansion are then placed in the list in the position the parent NT was taken from, as seen in Figure 5.2a. A reminder of the πGE derivation process is explained in Figure 5.2.

$$NT\ to\ expand\ =\ Codon\ value\ \%\ Number\ of\ NT's \tag{5.1}$$

Figure 5.2a shows the state of the πGE NT queue during each step of the tree expansion process. This process can be viewed as a string construction process, disregarding the tree representation. Figure 5.2b shows the same NT queue, represented as a derivation string. In the figure the terminal symbols have been added to the NT's in the queue. This process provides the current state of the derivation string at each expansion, and shows how the NT queue maintains the left to right ordering of the NT's as seen in Figure 5.1b. Figure 5.2b has been formatted in such a way as to display the approximate tree structure for easier

```
<e> ::= <e> <o> <e> | <v>
<o> ::= + | -
<v> ::= X | Y
Chromosome ::= 2,12,7,9,3,15,23,1,11,4,
               6,13,2,7,8,3,35,19,2,6
```

(a) Example grammar and chromosome



(b) Example of the πGE mapping process

Figure 5.1: This figure shows the πGE mapping process as seen in detail in Section 3.2.3. The chromosome shown in Figure 5.1a 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 Equation 5.1 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 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. 5.2a and 5.1b show the expansion of the example grammar in Figure 5.1a using the πGE mapping process. The number associated with each branch of the tree is a reference to the numbered steps shown in Fig. 5.2a which show how each choice of NT to expand comes about.

```
1.   [(e)]          => 2%1=0
2.   [e,(o),e]      => 7%3=1
3.   [e,(e)]        => 3%2=1
4.   [e,(v)]        => 23%2=1
5.   [(e)]          => 11%1=0
6.   [(e),o,e]      => 6%3=0
7.   [v,o,(e)]      => 2%3=2
8.   [v,o,(v)]      => 8%3=2
9.   [v,(o)]        => 35%2=1
10.  [(v)]          => 2%1=0
```

(a) NT selection process in πGE.

```
1.          [<e>]
2.          [<e>,<o>,<e>]
3.          [<e>,  -  ,<e>]
4.          [<e>,  -  ,<v>]
5.          [<e>,  -  ,  Y  ]
6.   [<e>,<o>,<e>,  -  ,  Y  ]
7.   [<v>,<o>,<e>,  -  ,  Y  ]
8.   [<v>,<o>,<v>,  -  ,  Y  ]
9.   [<v>,<o>, Y ,  -  ,  Y  ]
10.  [<v>, - , Y ,  -  ,  Y  ]
```

(b) NT selection process as a derivation string.

Figure 5.2: Figure 5.2a shows the state of the NT queue in πGE during the derivation tree expansion process. The NT selected in each step is underlined. Figure 5.2b shows the same thing as Figure 5.2a, represented as a derivation string. From this we can see how it is possible to track a depth-first ordering using the πGE NT queue.

comparison. The queue can now be used to determine how close to a traditional GE mapping order (left-most depth-first) a πGE ordering is.

DFOB is measured in terms of the average percentage distance away from a traditional GE left-most depth-first ordering. To ascertain how far from a depth-first order the current order is, the position selected by Equation 5.1, NT Choice, at each step of the πGE derivation is converted into a percentage using Equation 5.2.

$$ExpansionDistance = \frac{100}{\mid NT\ list \mid} \times NT\ Choice \tag{5.2}$$

The idea is that the left-most depth-first order, or 0% distance, is always to select the first item 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. A distance of 0% represents a left-most depth first, while a value of 100% for represent a right-most depth first ordering. A value around 50% would represent an ordering that was not biased towards either left-most or right-most expansion, and also deviated from a depth first approach.

Accuracy of Depth First Order Bias

Upon examination of the metric there was one noted issue. As the metric measures distance from the left of the list there is a slight bias the metric places as the first and last expansion have to be 0% as when there is only one element in the list the left most is always selected. This means that the maximum possible distance away from the ideal order is dependent on the number of nodes. Equation 5.3 displays a formula that can show a maximum percentage value based on the number of nodes in the derivation tree. A tree cannot have a distance unless there is more than two nodes in the derivation tree.

$$Max\ \%\ Distance = \frac{(Num\ Nodes - 2) \times 100}{Num\ Nodes} \tag{5.3}$$

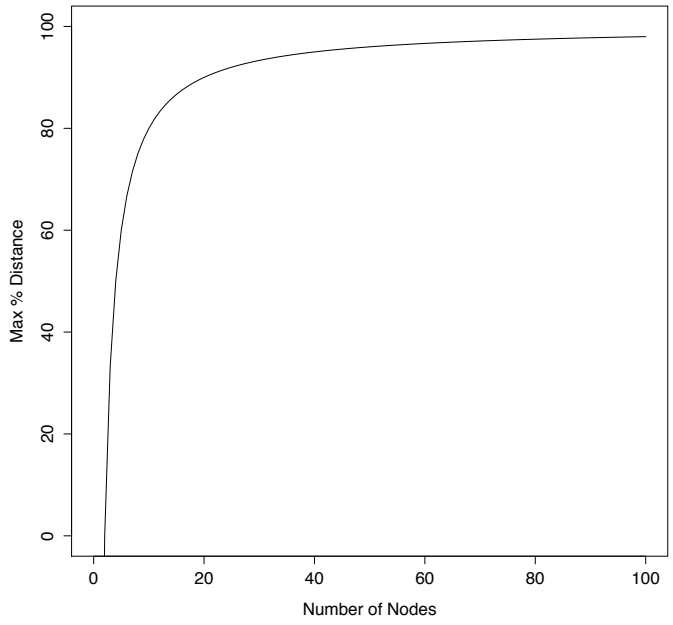


Figure 5.3: This figure displays the maximum distance that can be observed. Due to the first expansion and last expansion always recording a zero value, the maximum distance value increases with the number of nodes. The inclusion of these values is done to fairly represent all the expansions in the derivation process.

From Equation 5.3 it is clear that as the number of nodes increases the maximum

percentage approaches 100 quickly. This presents an important characteristic that needs to be considered when interpreting results. One can choose to leave these two nodes out of the metric thus removing the slight bias, or include them and represent a true picture of the order. For the study that follows it was decided to include all expansions as solutions to the problems to be examined would require significant number of nodes in the derivation tree thus minimising the effect of the first and last expansion values.

5.2.2 Breadth First Order Bias

The second expansion order that can be used as measure for πGE is the breadth-first order expansion. Recording of the tree depth at each step of the πGE mapping process allows for a distance from a pseudo breadth-first ordering to be obtained, called Breadth-First Order Bias (BFOB). Consider the πGE derivation tree in Figure 5.1b. The derivation tree has four levels. Level 0 contains the root node and then the level increases by one with each subsequent depth, up to a maximum value in this example of 3.

The first step required for BFOB is that each node is looked at in order of expansion. Using Figure 5.1b as an example, at each expansion the tree depth is noted. Once this process is finished what remains is an ordered list of expansion depths, $[0, 1, 1, 2, 1, 2, 2, 3, 2, 3]$. A breadth first order of this same tree would have resulted in the GPM visiting nodes in the following order, $[0, 1, 1, 1, 2, 2, 2, 2, 3, 3]$. By sorting the πGE list from smallest depth to highest depth value, it is possible to achieve a similar result. Sorting the depths in this way, results in the loss of all left to right ordering at each depth as seen in a breadth first mapping. What can be said to be produced from such a sorting of the list is a level-first ordering. This level-first order serves as an excellent indicator that a breadth-first order has been evolved.

Figure 5.4 shows a graph that displays the ideal order of tree levels a breadth first mapping would take in blue. The places where the observed expansion tree levels differs

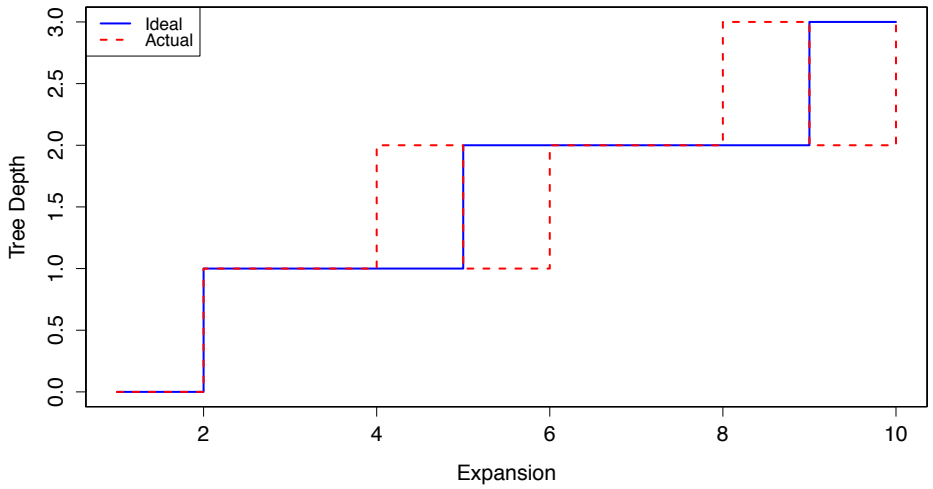


Figure 5.4: This figure shows the difference between a πGE expansion order and a breadth first expansion order. By taking the depth of each expansion in a πGE tree and sorting them, a breadth first ordering can be seen. Comparing the actual order and the ideal order using a Hamming distance approach, it is possible to observe if πGE is tending to a breadth first ordering. The ideal order is displayed in blue, while the red dashed line shows where πGE deviates from the ideal.

from this is shown in dashed red. Assigning a distance based on these regions in red will provide an indication towards how far from a breadth first ordering a πGE order is. Three methods for assigning this distance were explored during the course of this study. The following sections outline each approach and outline the pros and cons of each method.

Manhattan Distance Approach

The Manhattan Distance [93] approach was to take each expansion, and then use the absolute value of the difference between the expansion tree level and the sorted ideal expansion level, for each expansion and sum the differences. This total would then be divided by the number of points to provide the average distance per point from a level first order (Equation 5.4).

$$Average\ Manhattan\ Distance = \frac{\sum_{i=1}^{n} |i_{actual} - i_{ideal}|}{n} \tag{5.4}$$

The drawback of this approach was that it did not normalise across different tree sizes. A small tree with many errors would be considered closer then a large tree with one error that involved a larger distance for that error. Also there exists no upper bound for this type of distance.

Next Best Distance Approach

Next Best Distance (NBD) tried to compensate for the problems of the Manhattan approach by penalising the current choice to what should be the next correct expansion, e.g. with breadth first that would mean the NT with the lowest depth value. This approach meant that previous incorrect choices would not be penalised doubly. NBD used a stack approach where each level was provided a stack and then each occurrence of an expansion would be pushed onto the stack for its level. Then for each actual expansion, by comparing the current expansion level to the lowest level stack that had an expansion it could be determined what the next breadth first expansion should be. Summing these errors, and then dividing by the total number of expansions, produced an average distance per expansion.

This approach provided a more complex method of measuring a distance. The mistakes of previous expansions had little bearing on the current expansion's error. This method however still suffered from the same issues as the Manhattan distance approach above, suffering from the lack of ability to normalise the data across all tree sizes, and the lack of a distance limit.

Hamming Distance Approach

The Hamming Distance [50] approach was the final approach investigated. The Hamming distance is essentially a binary distance measure. In the Hamming distance approach if the expansion is at the correct level then a value of 0 is added to the total. An expansion that has the incorrect tree depth will result in a 1 being added to the total (Equation 5.5). The total distance would then be divided by the total number of expansions to provide an average distance per expansion for the individual being monitored, as shown in Equation 5.6.

$$Hamming(i) = \begin{cases} 0, & \text{if } i_{actual} == i_{ideal} \\ 1, & \text{else} \end{cases} \tag{5.5}$$

$$Average\ Hamming\ Distance = \frac{\sum_{i=0}^{n} Hamming(i)}{n} \tag{5.6}$$

The Hamming approach was chosen as it provided a distance that was limited between 0 and 1 that allowed for normalised reporting of distances on all tree sizes and problems. A distance of 0 for an individual will indicate an expansion order in a level first expansion manner, while a value near 1 represents a random level ordering. When used in conjunction with DFOB it will be possible to see if the orders in a πGE run evolve towards either of the known fixed orders of tree expansion for GE.

5.3 Experimental Setup

For all experiments reported here GEVA v2.0 [129] was used and modified as needed to produce the required output.

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 undertaken and the information was outputted to a file for post processing after the run had finished. For each individual both DFOB and BFOB were calculated. Population distance histograms were generated for every generation of the run.

The general settings for the experiments is displayed in Table 5.1. These setups were then applied to the four problem domains from Chapter 3, *Santa Fe Ant Trail*, *Even 5 Parity*, *Max* and *Symbolic Regression*. The experiments were then repeated using a fixed order initialisation as explained in Section 5.3.1, and then examined to see how the order would change starting from a fixed order. Would it maintain the order or remain close to it, or would it follow the behaviour of standard πGE?

Table 5.1: Parameter settings adopted for the order experiments.

Parameter	**Value**
Generations	100
Population	100
Replacement strategy	Generational with elitism (10%)
Selection	Tournament size=3
Mutation probability	0.01 (integer mutation)
Crossover probability	0.0 & 0.9 (variable single point)
Initial chromosome length	200 codons (random init & GE order init)
Runs	100 per setup & problem

5.3.1 Initialisation Methods - GE Order Initialisation

Random initialisation was initially chosen for the study to observe how the orderings on πGE tree changed during evolution. However after the initial results for the study were examined it was determined that a fixed order initialisation method was needed for πGE. This fixed order initialisation would allow for comparisons to be drawn between a known order, and also allow for investigation into πGE evolving away from a known order. Both questions that couldn't be answer by random initialisation alone.

GE Order Initialisation (GOI) was devised to provide πGE with such an initialisation method. By taking advantage of the ordered queue that πGE uses, as discussed in Section 5.2.1, it is possible to initialise the population to a GE-style derivation order. Setting πGE to a GE order requires that every order codon in the chromosome be set to 0. This guarantees that the πGE mapping process will always select the left most NT to expand next, just like traditional GE. GOI can be applied to both RHH (Section 2.4.1) and random (Section 2.4.1) initialisation methods, allowing πGE to be initialised to a GE order for each. For the purposes of this study only the random initialisation method was used.

5.4 Results

The initial focus of the study is to observe the population over 100 runs and examine the behaviour of the DFOB, Section 5.4.1. Following this the study moves onto the results observed using the BFOB, Section 5.4.2. Section 5.4.3 presents the results of the investigation of how grammar complexity may affect orders, while Section 5.4.4 examines crossover's effect. Finally the section is concluded with a trio of discussions in Section 5.4.5.

5.4.1 DFOB Results

Figure 5.5 displays the distance histogram from a depth first ordering for a randomly initialised πGE population over 100 runs at five fixed intervals during evolution. 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 due to the grammar structure. It can be observed by following the sub-figures from left to right in chronological order, that there are fewer of these small individuals after 100 generations. This trend was seen across all setups and problems initialised using the Random ordering. In general the population tends to maintain a distribution of orders centred heavily in the $30\% - 40\%$ range of the histogram. The speed at which this distribution is reached varies with the problem and could be linked to the different grammars. This is investigated in more detail in Section 5.4.3.

Figure 5.6 reports the results for DFOB on a population that has been initialised to a known order, GE depth-first. This setup was proposed to see if providing πGE with a known order would result in the algorithm maintaining the order, or would it diverge. Examining the figure it can be seen how the whole population now starts in the left-most column of the histograms denoting a depth-first order. Over time it can be seen that the population starts to adapt the initial order and the population can be seen to slowly diverge. The population's drift away from the GE order was slow in the problems examined but there is no way to stop this drift in πGE. This drift is due to the ripple effect in πGE, that is discussed further in Section 5.4.5. In the setups shown again the speed of divergence varies slightly between setups but the diverging trend is very evident.

It can be seen from these results that πGE maintains a distribution of orders during evolution and does not converge to a specific order. Providing a known fixed order, via the GE order initialisation, shows that πGE will modify and explore around an order, and

over time will resist order convergence and continue to explore.

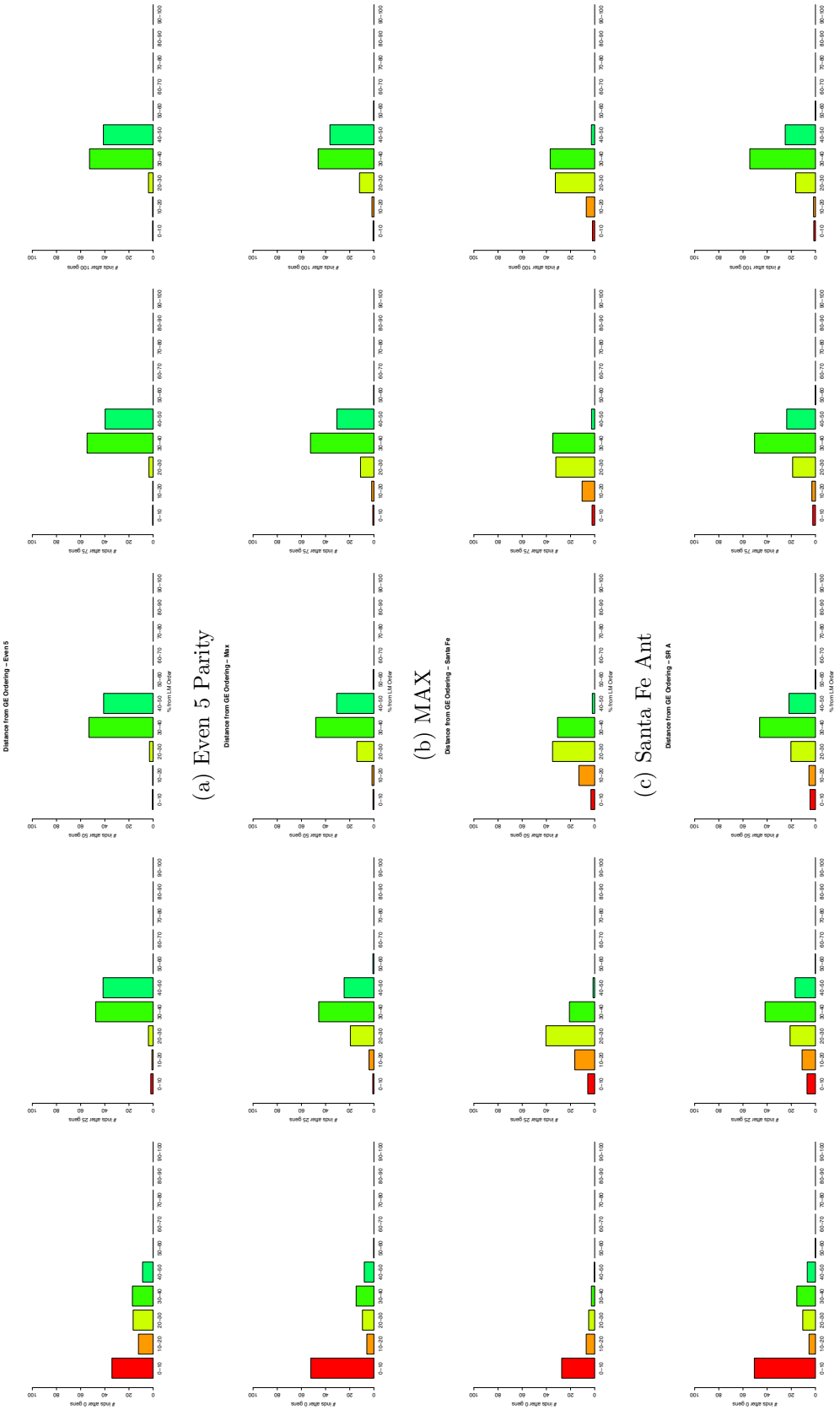
Further examination was done into DFOB by examining the top 10% of the population during evolution, and the results are in Appendix A. The trends observed in the whole population of drifting towards a distribution of orders, from a random and fixed order initialisation, are seen in those figures. This proves that the population dynamics observed here, provide a good indication for the elite populations behaviour. The trends were also seen in the successful runs reported in Appendix B.

5.4.2 BFOB Results

Figures 5.7 and 5.8 display the distance from a breadth-first mapping order, for random initialisation and GE order initialisation respectively. The main difference between the two figures can be seen in the initial population of each run. The GE order initialisation does not provide πGE with as varied a range of orders in relation to breadth-first ordering. The distributions are heavily biased toward the left and right. The left most column of the histogram can be discounted in this case as it contains short individuals as was seen in the previous section.

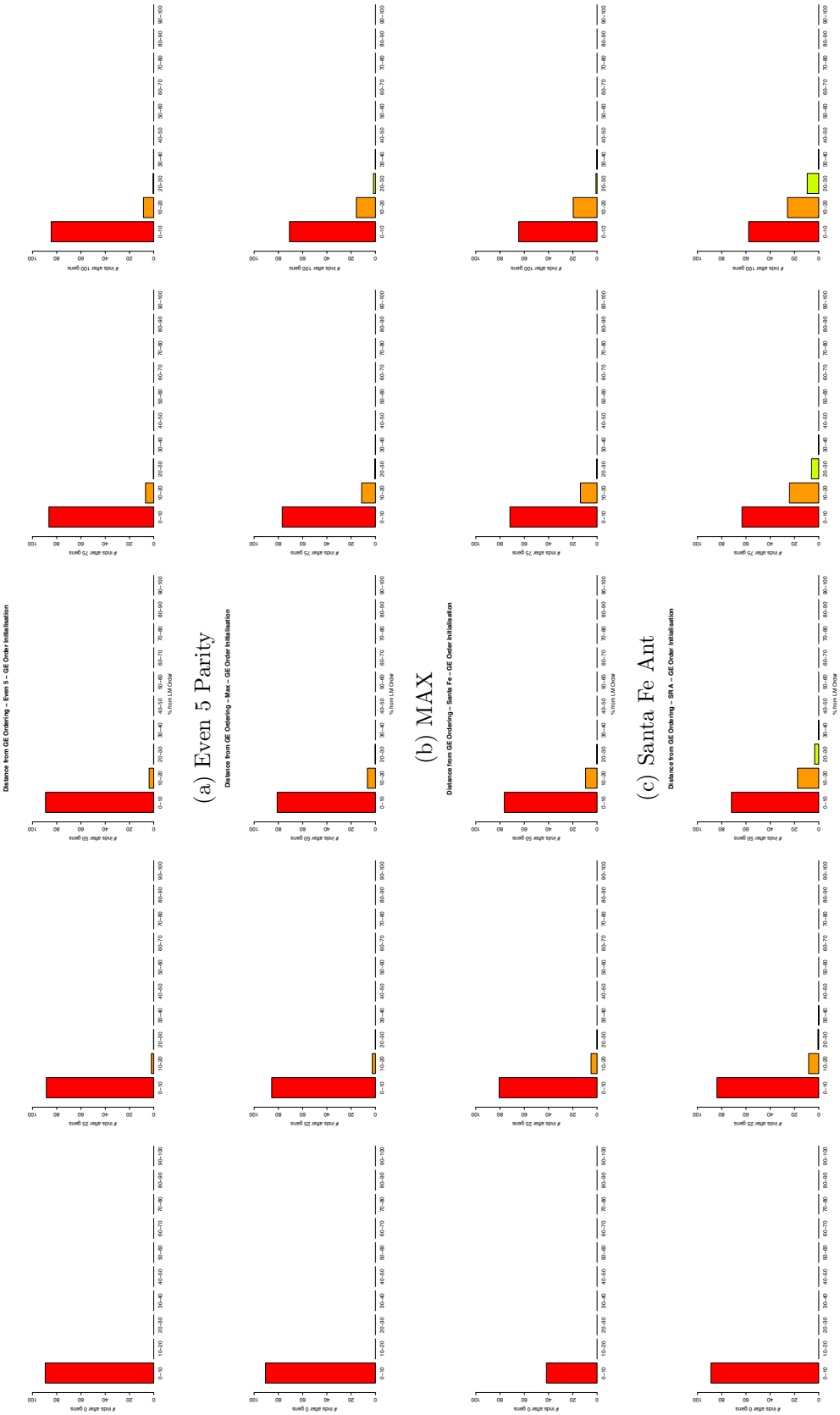
Another interesting observation can be made by examining the populations at generation 100. In the case of the Santa Fe and Symbolic Regression A problems, there is a trend towards a wider range of orders at the final generation then seen in the other problems. This could be due to the grammars and how they affect order distributions. This will be explored further in Section 5.4.3.

As was the case in the previous section the whole population results provided a very good indication for performance of the elite individuals and the successful run only orders. These results can be found in Appendix A and B respectively.



(a) Even 5 Parity (b) MAX (c) Santa Fe Ant (d) Symbolic Regression A

Figure 5.5: The figure shows the distribution of distances from a depth-first order on four problems with Random Order Initialisation. The subfigures show how the distribution changes during the course of the run with the left-most graph being generation zero, and the right-most being the final generation. In the four graphs it is evident that πGE evolves towards a distribution of orders centre around 30% - 40%.



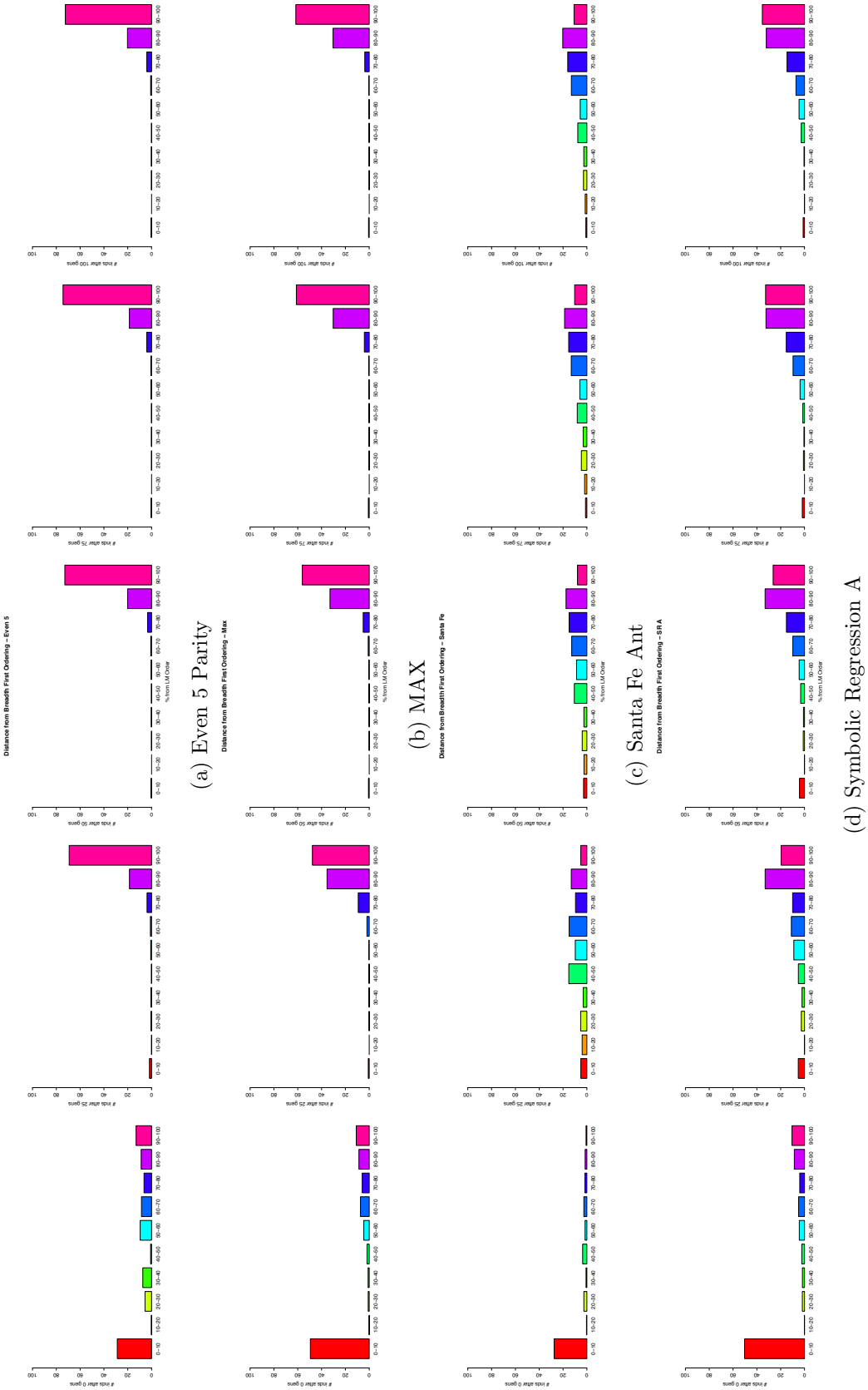
(a) Even 5 Parity

(b) MAX

(c) Santa Fe Ant

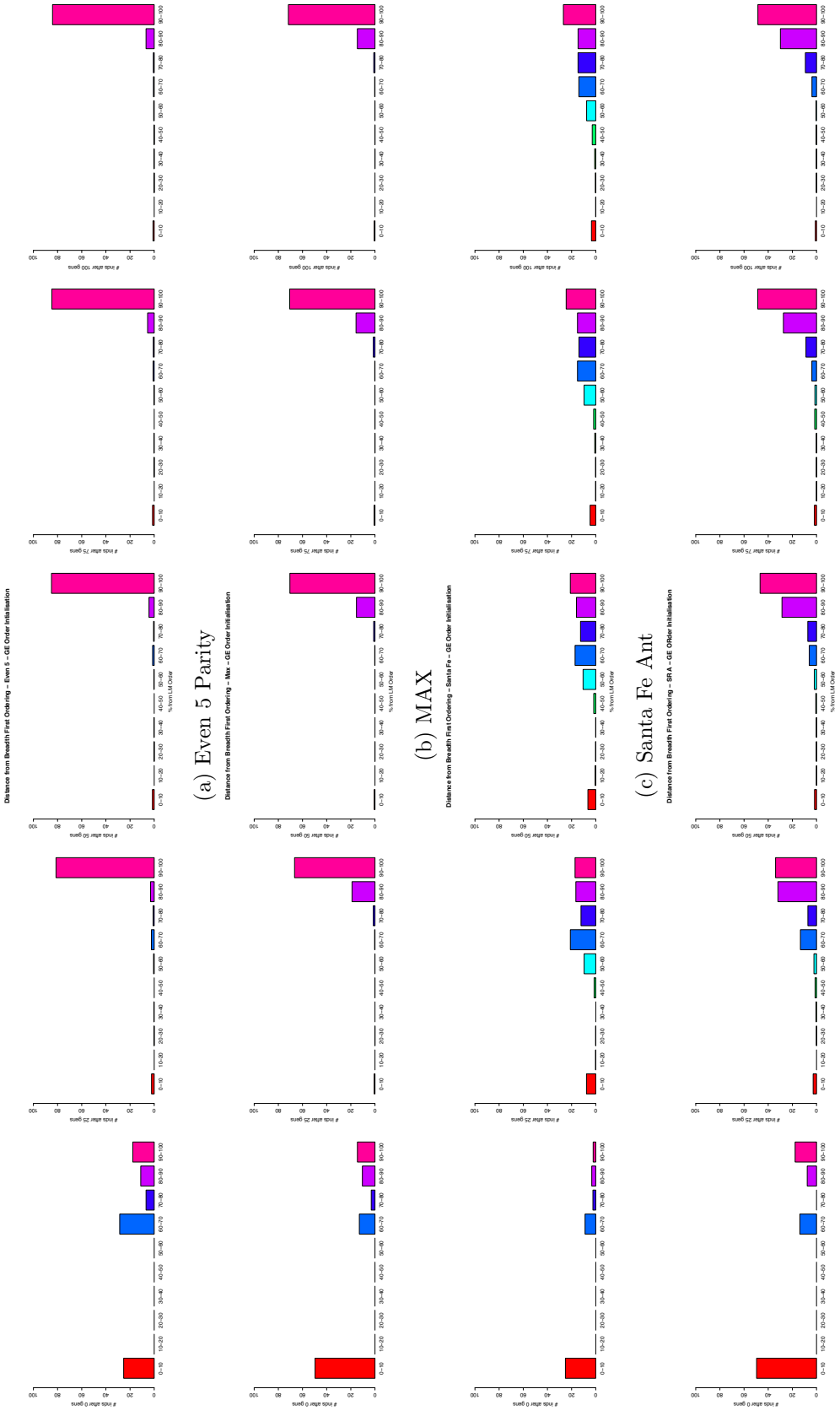
(d) Symbolic Regression A

Figure 5.6: The figure shows the distribution of distances from a depth-first order on four problems with GE Order Initialisation. The subfigures show how the distribution changes during the course of the run with the left-most graph being generation zero, and the right-most being the final generation. In the four graphs it is evident that πGE evolves away from the fixed order at varying rates, but there is a strong presence of GE style orders present in the populations.



(a) Even 5 Parity (b) MAX (c) Santa Fe Ant (d) Symbolic Regression A

Figure 5.7: The figure shows the distribution of distances from a breadth-first order on four problems using random order initialisation. The figures show that the population starts with a wide distribution of values. For Even 5 Parity and Max the population is in a much tighter distribution at the end of the runs, that is focus at 100%. Santa Fe Ant and Symbolic Regression A, however, have a much wider tail to their distributions at the end of the runs. The figure indicates that there is no indication of a significant breadth-first order being present.



(a) Even 5 Parity (b) MAX (c) Santa Fe Ant (d) Symbolic Regression A

Figure 5.8: The figure shows the distribution of distances from a breadth-first order on four problems using GE order initialisation.The figures show that GE order initialisation actual pushes the final distribution further to 100%, than was seen in Figure 5.7. Also the generation 0 distribution has a different shape that was seen with random order initialisation.

5.4.3 The Grammar Effect

In Chapters 3 and 4 it has been shown how the changing of the complexity of the grammar used for a problem can have an effect on the performance and connectivity of πGE. In Sections 5.4.1 and 5.4.2 some concerns were raised that suggested perhaps the complexity of the grammars used for the problems could be effecting the distribution of orders. To answer this question a second variant of the symbolic regression grammar was used that had reduced arity in some of the production rules. This second grammar variant (SR B) was then used to solve the same target function, and the orders analysed. The grammars used are displayed in Figure 5.9.

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

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

<op> ::= + | - | * | /

<var> ::= x0 | x1 | 1.0
```

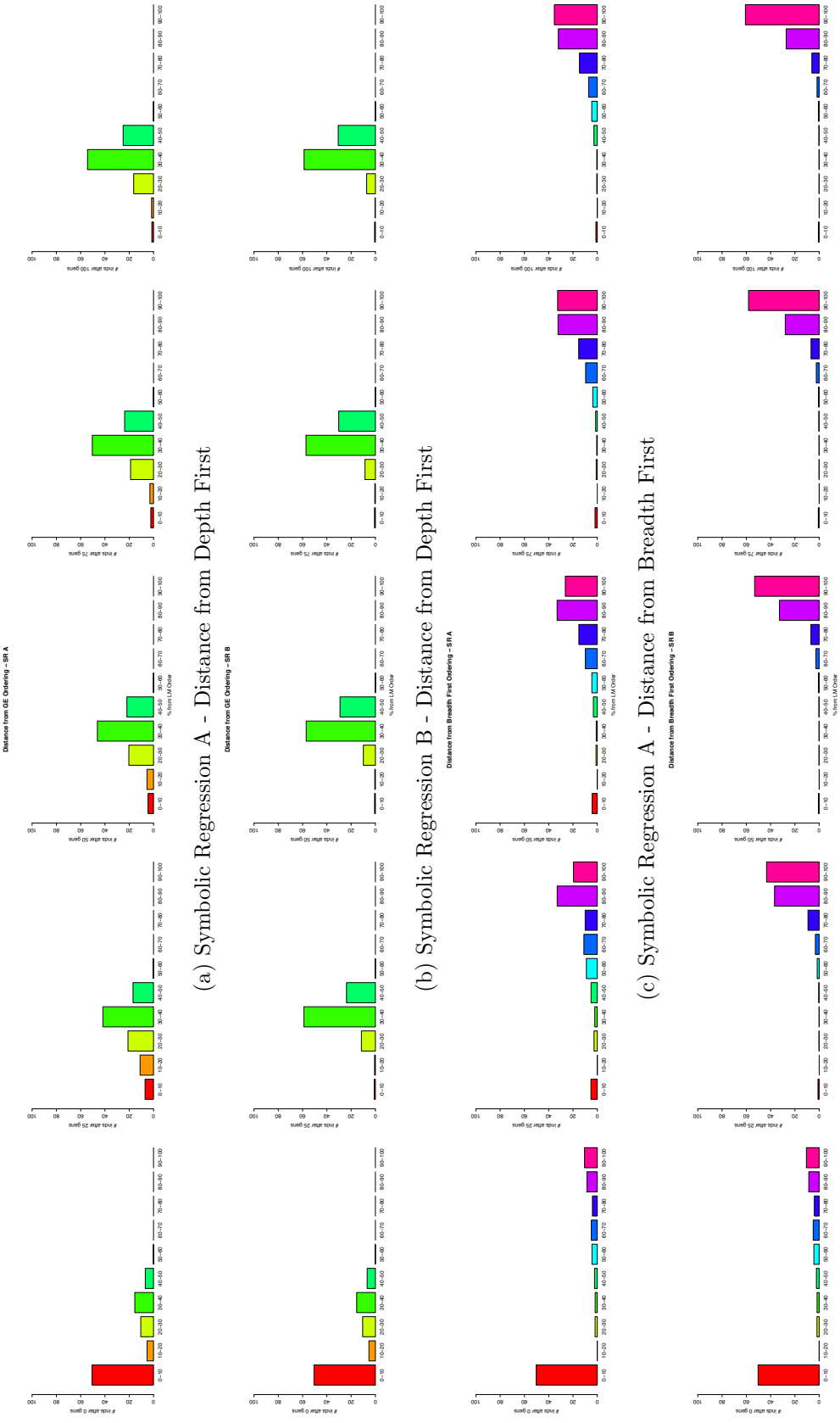
(a) Symbolic Regression A.

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

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

<op> ::= + | - | *

<var> ::= x0 | 1.0
```

(b) Symbolic Regression B

Figure 5.9: Grammars used for the grammar complexity comparison are presented above.

Figures 5.10 and 5.11 present the comparison of the two grammar variants for random initialisation and GE order initialisation respectively. Looking at each figure it is evident that the reduction in the grammar complexity, and as a result the search space explorable, does in fact narrow the distribution of orders for both metrics. This helps to explain the difference seen in previous sections where a difference was noted between problems and suggests that it could be due to the difference in grammar. However the general trends observed in the previous sections can be seen in this example, with a distribution of orders evolving over time.



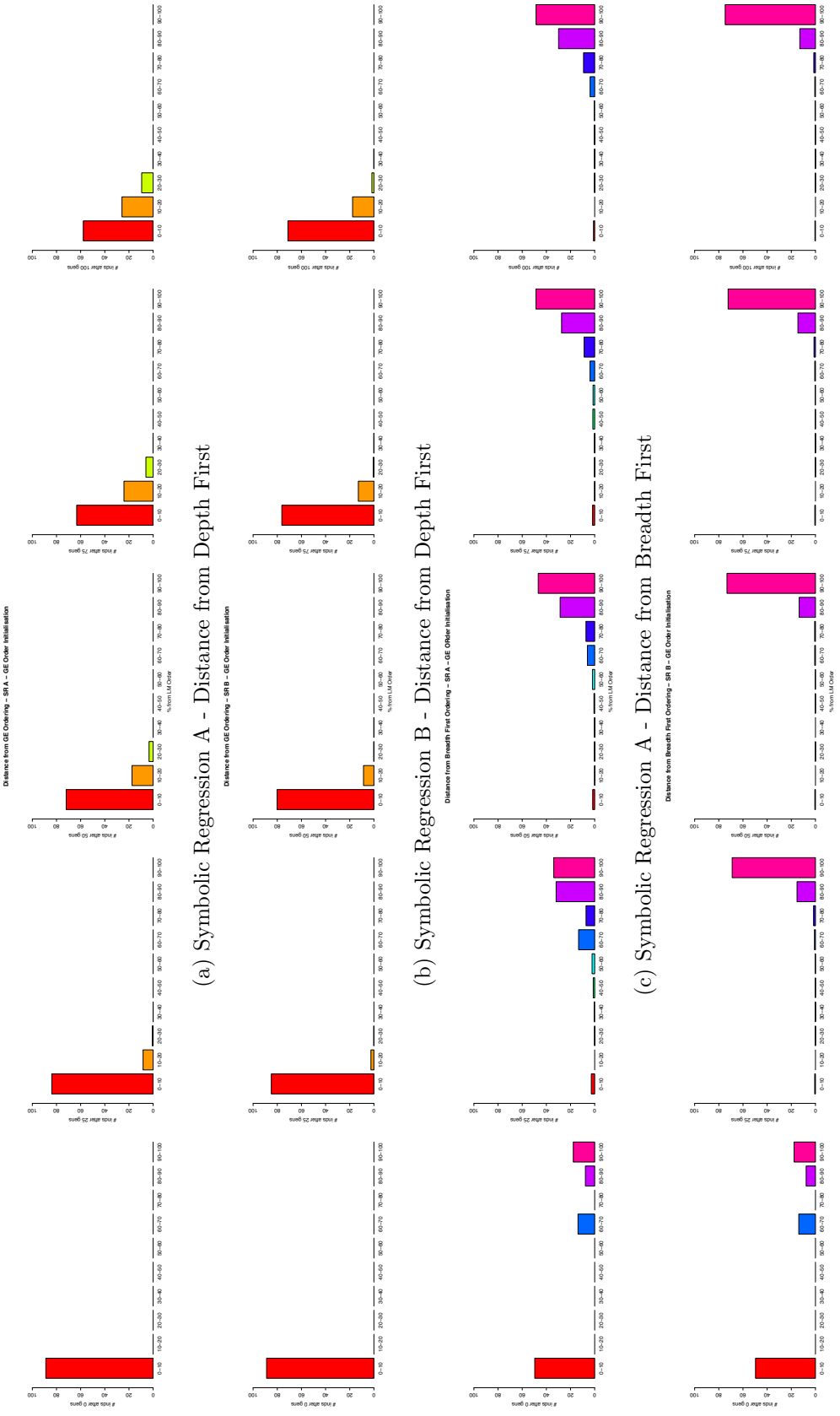
(a) Symbolic Regression A - Distance from Depth First

(b) Symbolic Regression B - Distance from Depth First

(c) Symbolic Regression A - Distance from Breadth First

(d) Symbolic Regression B

Figure 5.10: The figure shows the difference a change in grammar induces on the orders in πGE using a random order initialisation. The reduction in grammar complexity has lead to a reduction in the tail size of the distributions shown.



(a) Symbolic Regression A - Distance from Depth First

(b) Symbolic Regression B - Distance from Depth First

(c) Symbolic Regression A - Distance from Breadth First

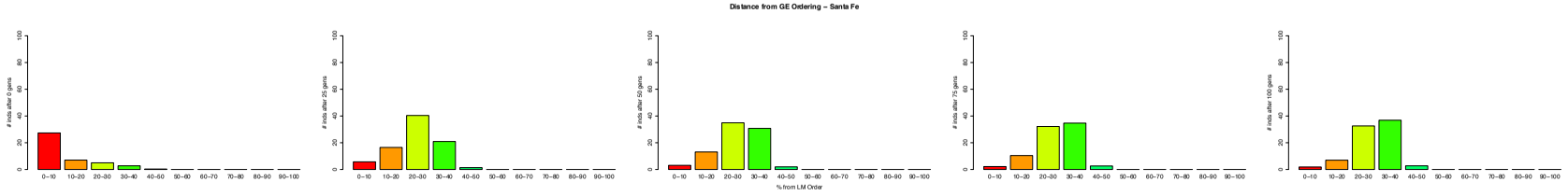
(d) Symbolic Regression B - Distance from Breadth First

Figure 5.11: The figure shows the difference a change in grammar induces on the orders in πGE using GE order initialisation. The reduction in grammar complexity has lead to a reduction in the tail size of the distributions shown as in Figure 5.11. An interesting observation is how the reduction in grammar complexity has also retarded the drift away from a GE order.

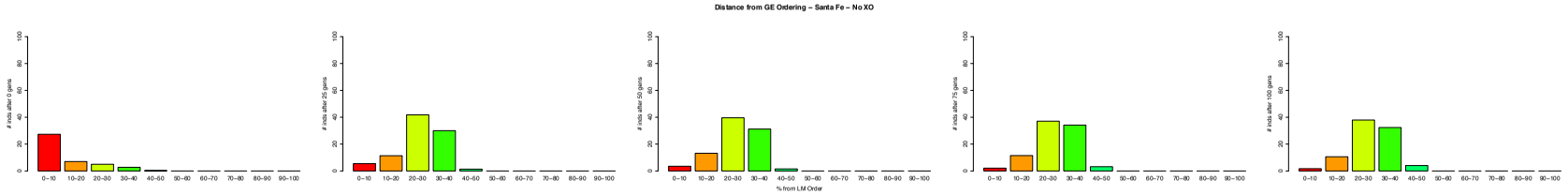
5.4.4 The Crossover Effect

In Chapter 3 it was seen that πGE did not get as significant a performance boost from crossover, as was observed in GE. Crossover in πGE looked to provide additional search but did not contribute much to overall best fitness or average population fitness. This section examines what impact crossover may have on order in πGE. Does crossover increase the rate at which πGE's order diverges? Does it help to maintain a diversity of orders?

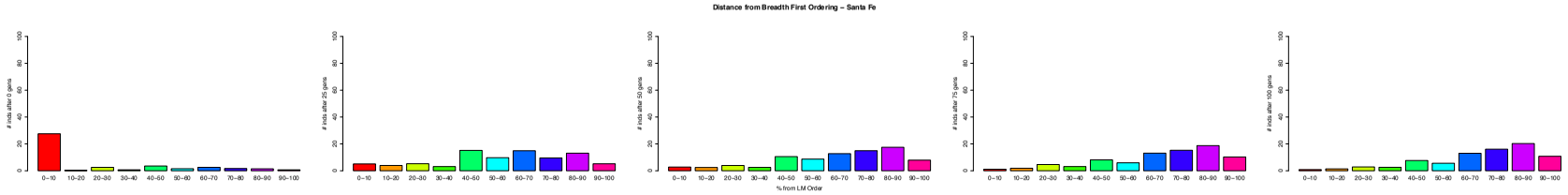
Figures 5.12 and 5.13 show results for both initialisation methods on the Santa Fe ant problem. In Figure 5.12 the DFOB and BFOB is displayed with and without crossover. There is only a very slight difference in the order histograms when crossover is turned off. In Figures 5.12a and 5.12b crossover leads to the distribution being centred in $30 - 40\%$ rather than $20 - 30\%$ without crossover, and no discernible increase in divergence can be seen in Figures 5.12c and 5.12d. This leads to the conclusion that switching crossover on or off does not have much of an effect on order in πGE. The effect of crossover in πGE could be being limited by the fact that mutation can have such a destructive ripple effect in πGE.



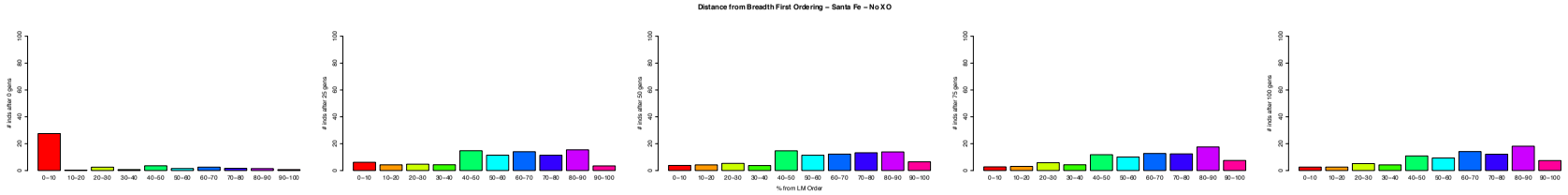
(a) Santa Fe Ant - Distance from Depth First



(b) Santa Fe Ant without Crossover - Distance from Depth First

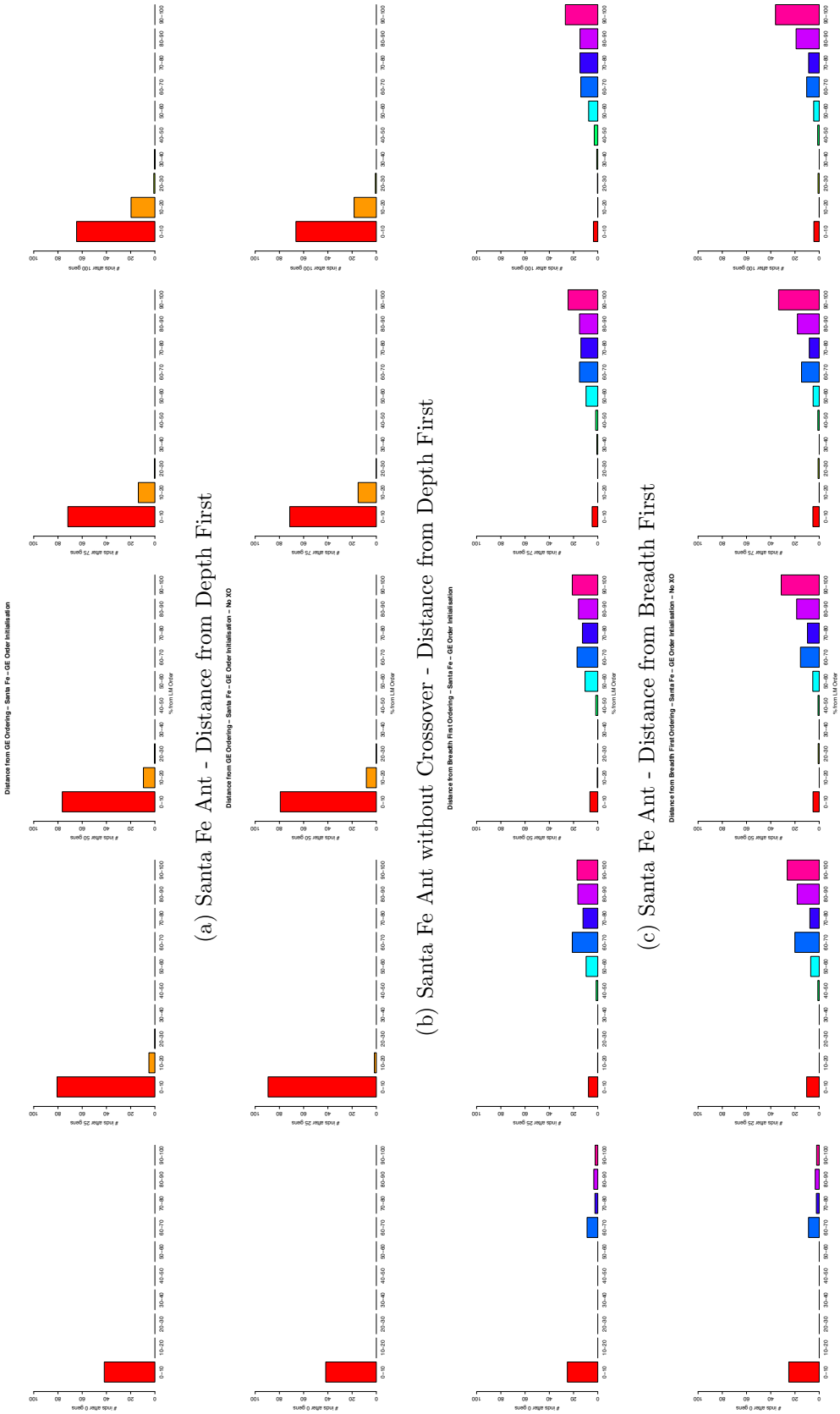


(c) Santa Fe Ant - Distance from Breadth First



(d) Santa Fe Ant without Crossover - Distance from Breadth First

Figure 5.12: The figure shows the effect crossover has on the orders in πGE on the Santa Fe Ant problem. In Figure 5.12a the distribution of orders is centred at $30-40\%$ rather than at $20-30\%$ in 5.12b. Figure 5.12c the distribution after 100 generations is much smoother than compared to Figure 5.12c. In both DFOB and BFOB crossover can be said to push the distribution further away from a known order in both cases.



(a) Santa Fe Ant - Distance from Depth First

(b) Santa Fe Ant without Crossover - Distance from Depth First

(c) Santa Fe Ant - Distance from Breadth First

(d) Santa Fe Ant without Crossover - Distance from Breadth First

Figure 5.13: The figure shows the effect crossover has on the orders in πGE on the Santa Fe Ant problem using GE order initialisation. As was seen in Figure 5.12, there is only a slight variation in order with the addition of crossover.

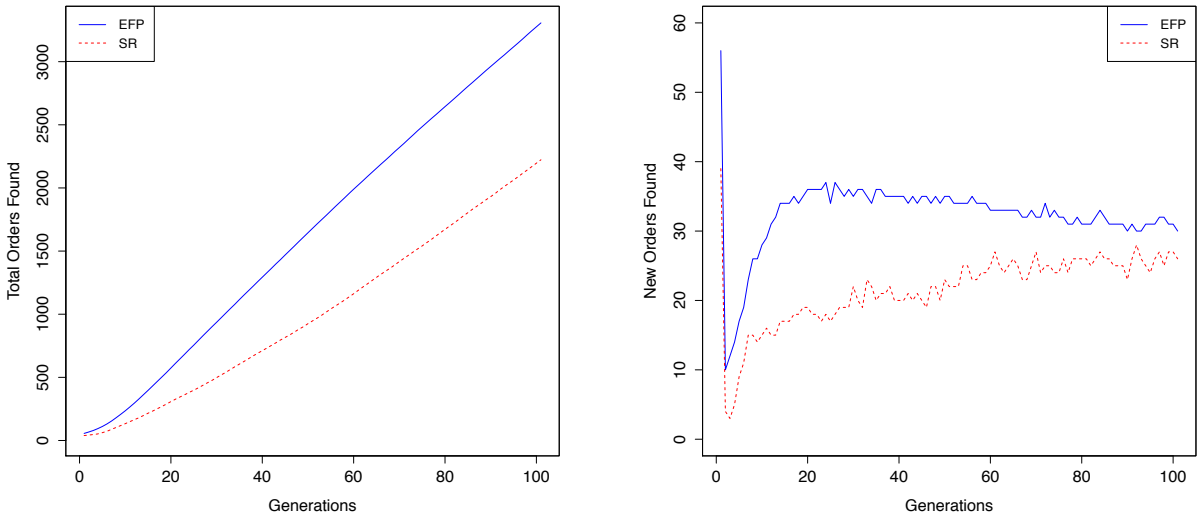
5.4.5 Discussion

The experiments in this chapter have raised questions that warrant further discussion. The following sections aim to answer these question. Section 5.4.5 takes a further look at the orders being discovered during evolution for uniqueness. Section 5.4.5 examines the tree bias in πGE derivation trees, comparing against GE's tree bias. Finally, Section 5.4.5 ends the discussion section with a look at limiting order drift in πGE.

Orders

The results section have displayed how πGE tends towards a distribution of orders. This distribution approach does not show if the orders in each bar of the graphs are the same orders or different. It was decided to perform an additional experiment that would analyse the orders, and identify the unique orders being discovered by πGE

Taking the data from the 100 runs used in previous sections, the data was examined and the following reported. Figure 5.14 displays the orders encountered by πGE during evolution. Figure 5.14a shows the average total number of orders found over 100 generations. Figure 5.14b displays the average new orders found per generation. After the initial generation there is a sharp decline in the number of new orders found. The behaviour of the two problems differ slightly. Symbolic Regression shows a sharp decline in the number of new orders found but this is then followed by a continual increase in the number of new orders found. Even 5 Parity though suffers a sharp decline, followed by a quick increase in the number of orders found, before peaking and then showing signs of a decline in the number of new orders seen. These results show that πGE is continually finding new orders every generation, with an average of $20 - 30\%$ of the orders per generation being never before encountered orders.



(a) Total orders found

(b) New orders found per generation

Figure 5.14: This figure displays the orders encountered by πGE during evolution. Figure 5.14a displays the total number of orders found averaged over 100 runs for two problems. Figure 5.14b displays the new orders found per generation, again averaged over 100 runs on the same two problems. It can be seen that after the initial generation there is a sharp fall in new orders found before this figure climbs over the following generations. These figures show that πGE continues to find new orders during the course of the run.

Tree Bias

Hemberg [66] has reported that grammar design can lead to a certain bias in the derivation trees produced by GE. Does the addition of a position independent mapping change the bias? Up to this point orderings in πGE have been compared to a fixed order-initialised πGE. However the change in initialiser could result in the two populations starting out with differently biased initial populations. This presents an interesting question that must be investigated to see if the bias in the initial population could be affecting the results above.

To answer this question a small experiment was designed to compare the initial populations of GE and πGE, from the perspective of tree shape. Does one approach result in trees with a certain bias towards a left branch (where the difference between the depths of

the right branches of the roots subtree is smaller than the left branches), a right branch bias, or a balanced tree shape?

$1,000,000$ derivation trees were generated using random initialisation. The initialiser was modified to exclude derivation trees of a single variable (x), and a single expression (e.g. $x + 1.0$). This limitation was imposed as the binary nature of the grammar would produce too many of these types of trees, generally only present in the initial population, as noted above. The chromosome length was set at 50 codons.

The symbolic regression grammar seen in Figure 5.9b was also changed to an infix notation. This change was imposed to provide a neutrally biased grammar. The change consisted of replacing `<expr> ::= <op> <expr> <expr>`, that adds a right tree expansion bias, with `<expr> ::= <expr> <op> <expr>`.

The bias of a tree was calculated by first translating the leaf nodes of the derivation tree to points $p = (x, y)$. If there was a tree with n leaf nodes, the leftmost leaf would receive an x value of 1, and next left most would be given a value of 2, and this value would increment by 1 until leaf node n in reached. y represented the depth of the leaf node in the tree. Once the nodes have been translated to points, the slope (S) of the line of best fit is calculated for the points using Eq. 5.7. This slope will indicate the bias of the tree shape. If a tree is left-heavy it would have a negative slope value.

$$S = \frac{\sum_{i=1}^{n} x_i y_i - ((\sum_{i=1}^{n} x_i) \times \overline{y})}{\sum_{i=1}^{n} x_i^2 - ((\sum_{i=1}^{n} x_i) \times \overline{x})} \tag{5.7}$$

Figure 5.15 shows the distribution of slopes for both GE and πGE. From the figure there are two things of note. Firstly both GE and πGE display the exact same distribution of tree shapes. The addition of position independence results in no change in derivation tree bias. The second interesting observation is the symmetry that is visible in the plot. It indicates that for every tree that has a right-heavy there is a similarly left-heavy tree.

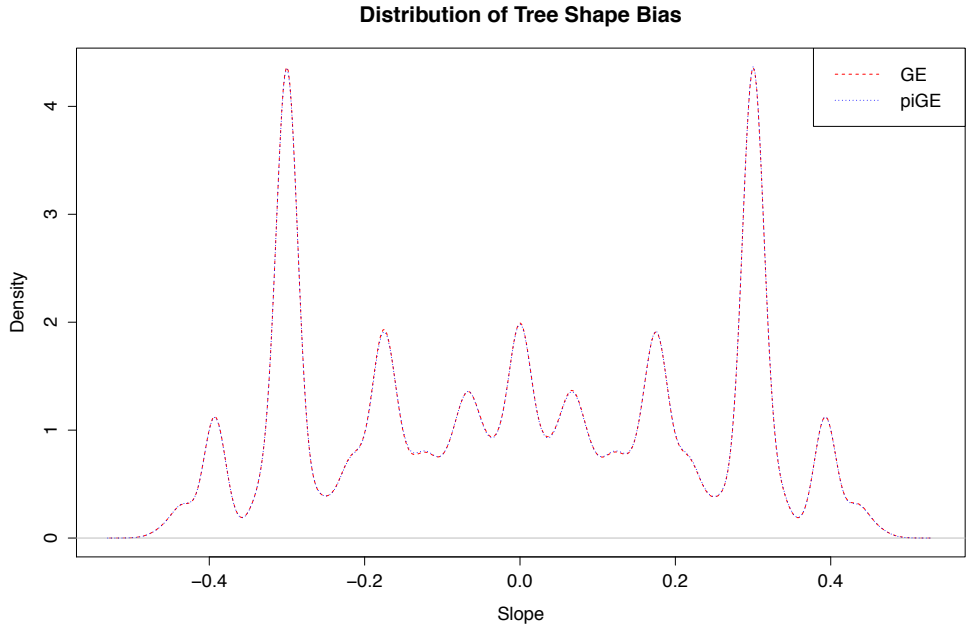


Figure 5.15: This figure displays the distribution of derivation tree slopes for 1,000,000 derivation trees of both GE and πGE. The figure indicates there is no difference in the bias of trees generated in the initial populations of GE and πGE

The initial study by Hemberg [66] examined the bias present in GE with respect to performance. In the study a preferential bias could be found that impacted performance. This study examined bias within the initial population of GE and πGE. In this examination no left-right bias was observed in either algorithm. Also no difference in bias was observed between the two algorithms.

Restricting Order Drift in πGE

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

In previous work [39], 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 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. If a content mutation changes the number of NT's in the πGE NT 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 [11], 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 affected.

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. This mutation operation and ripple is further explored in Chapter 6.

5.5 Summary

The main aim of this chapter was to further investigate what happens within πGE with regards to the expansion orders used in the algorithm. The orders of πGE individuals during evolution were 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. Two different metrics where defined and used to explore order in πGE. The effect that grammar complexity has on the distribution of orders was discussed and it was found that more complex grammars lead to a wider distribution of orders. Crossover was examined and found to have limited impact on the orders observed within πGE. The orders discovered during evolution were examined to verify that πGE continued to search the order space, and encountered new orders during evolution. Derivation tree bias was also examined and it was found that the addition of the variable expansion order in πGE did not change the bias in the initial population. Finally the idea of restricting the order drift in πGE was discussed and the ripple effect of πGE was outlined.

In the following chapter we will explore a new mutation operation that may help deal with this ripple effect.

Chapter 6

Focused Mutation with πGE

Chapter 3 has shown πGE to exhibit comparable performance, and in some cases slightly better performance, to standard Grammatical Evolution (GE). The chapters that followed investigated certain aspects of πGE in the quest to understand the inner workings and behaviours of πGE. These studies all relied upon the traditional GE int flip mutation operation. GE uses a leftmost non terminal expansion GPM, while πGE evolves the order of mapping as well as the content. In this chapter, the idea of focused mutation search is introduced and used to examine which aspect of the πGE mapping process provides the lift in performance over standard GE. By examining different setups of focused mutation on a set of benchmark problems, a purely content focused int flip mutation was shown to exhibit a performance gain over the other setups.This chapter presents a more in depth examination of work presented by Fagan et al. [39].

The chapter is structured as follows. Section 6.1 outlines the motivation for this vein of research. Section 6.2 describes the focused mutation operation, before the experimental setup is stated in Section 6.3. The results are reported and discussed in Section 6.4. The chapter is concluded with a discussion in Section 6.5 and a summary in Section 6.6.

6.1 Introduction

The earlier studies in this thesis have relied upon the application of standard GE mutation operations in πGE. Chapter 3 examined performance across a range of GPM using the GE mutation operation. Chapter 4 mapped the phenotypic connectivity of the GE mutation operation and highlighted order and content mutations. Finally Chapter 5 looked at the orders being evolved during a πGE run, with the GE mutation operation being used again.

Through careful examination of the πGE's mapping process, questions have arisen as to what effect mutation has on the mapping process of πGE and if there exists a better solution to applying mutation in πGE, than the int flip mutation used so far. These are:

- Should mutation occur on the part of the chromosome that controls which of the unexpanded Non-Terminal Nodes (NT) will be selected for expansion next? If so what effects does this have?
- Should mutation focus only on what a selected NT expands to?
- Is some form of adjustable dual mutation rate required to extract maximum performance from the πGE mapper?

This study presents an examination of performance of πGE, and through the idea of focused mutation introduces new ideas as to how best to utilise mutation for finding the optimum solution in the search space by adjusting the mapping process. Through these new approaches it is hoped to gain a deeper insight into the behaviour of πGE, firstly by establishing a baseline of performance with only Content Focused Mutation and Order Focused Mutation, and then by examining if certain mixtures of mutation can yield performance increases not previously encountered.

6.2 πGE Focused Mutation Operation

The addition of the evolvable order with πGE presents a new degree of freedom in the GPM process, that is not currently utilised with GE operations. πGE has a codon that controls the order of derivation, and also the subsequent NT expansion. By focusing on each of these codons it is possible to produce a mutation operation that can be tuned as desired. In the standard πGE setup, the rate of search performed on the order and content codons relative to each other in general is on a basis of 1:1. If this search is allowed 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. This may allow the current order's content to be evolved for a number of generations before the order is changed. In standard πGE, an unknown amount of order and content codons are mutated, as GE's mutation operation is noisy and relies on probabilities and a random number generator, thus giving a fluctuating mix of mutation between codon types across the population. This can have a drastic effect on the 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). 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 the order of the search being undertaken is explicitly defined.

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 exchange 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. **Order : Content**: 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.

By undertaking this investigation it is hoped 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.

6.3 Experimental Design

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.

GEVA v2.0 [129] was adopted and tailored 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 6.1. Note that a relatively small population size of 100 was deliberately used, 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 provide the possibility to more precisely distinguish performance differences on these benchmark problems. Elitism was restricted to a size of 3 to prevent the population from converging to quickly. For all approaches an initial chromosome length of 200 was selected to provide all mappings with a similar amount of randomly created genetic material.

Table 6.1: Parameter settings adopted on all problems examined.

Parameter	**Value**
Generations	100
Population size	100
Replacement strategy	generational with elitism (3 individuals)
Selection	tournament (tsize=3)
Mutation probability	0.01 (integer mutation)
Crossover probability	0.0 & 0.9 (ripple)
Initial chromosome length	200 codons (random init)

The problems used for this study mirror those seen in previous chapters. The Even Five Parity Problem, Max, Santa Fe Ant Trail and Quartic Polynomial Symbolic Regression, represent the problem domains examined. The grammars used are displayed in Figure 6.1.

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

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

<op> ::= + | *

<var> ::= 0.5
```

(a) Max.

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

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

<op> ::= + | - | * | /

<var> ::= x0 | x1 | 1.0
```

(b) Symbolic Regression A.

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

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

<op> ::= + | - | *

<var> ::= x0 | 1.0
```

(c) Symbolic Regression B

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

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

<pre-op> ::= not

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

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

(d) Even 5 Parity.

```
<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>
```

(e) Santa Fe Ant.

Figure 6.1: Grammars used for the problems in this chapter are presented above. There were five grammars used during the course of this investigation.

6.3.1 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 rates are needed to normalise the effective rate of mutation per individual. These have been calculated using the following equations:

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

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.

Table 6.2: Mutation Rates for Experimental Setups

Setup	**Order Rate**	**Content Rate**
1:1 ratio (GE)	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

Note that these equations are 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 6.2.

6.4 Results

The results for the four experimental setups described in Section 6.3 are now presented. We divide their exploration 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. The Wilcoxon rank-sum test was again used to test for statistical significance between setups. As in Chapter 3, a Bonferroni correction will be applied to the threshold. In this chapter there are three datasets, hence three pairwise tests. This will result in a new$p$-value threshold of < 0.01666.

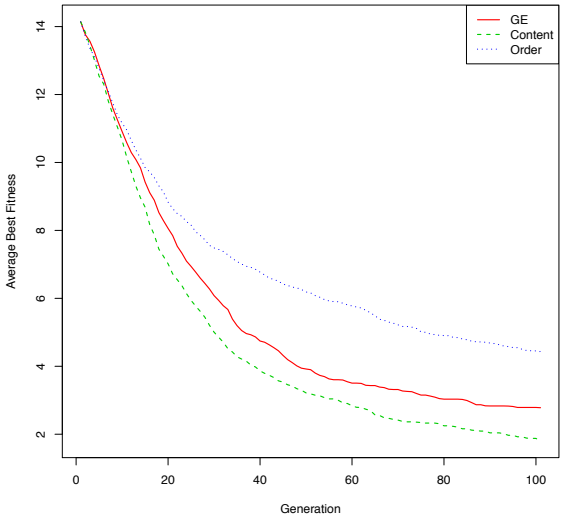
6.4.1 Results for Standard GE Mutation, Order-only Mutation and Content-only Mutation

Table 6.3 displays the number of successful solutions found to the four problems, as well as the mean best fitness, and the standard deviation, over the 250 runs performed for each setup. Table 6.4 shows the p-values for each setup compared against the other setups. Figure 6.2 plots the mean best fitness for the results displayed in Table 6.3.

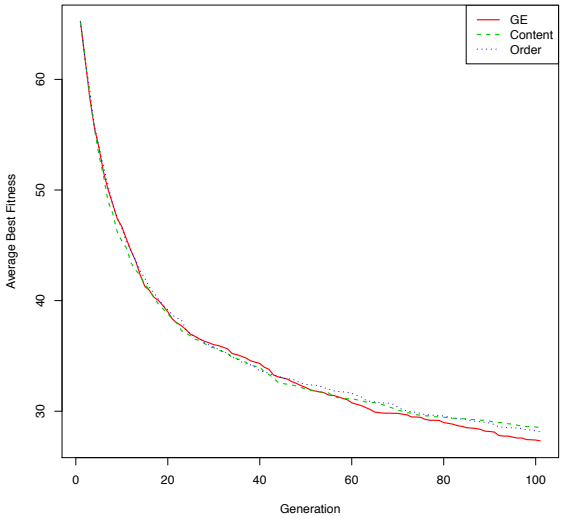
Table 6.3: This table shows the results for the content and order only mutation approaches on a range of benchmark problems. 250 runs for each setup were performed and the results are shown below. In the table success denoted the number of solution found during the 250 runs. The average of the best fitness is presented with the standard deviation shown in brackets. The highlighted cells indicate the best performing approach. In the case where multiple approaches are highlighted, there is no statistical difference between the approaches.

Problem	Mutation	Mean Best Fitness (stdev)	Successes
Even 5	GE	2.78(3.82)	156
	Order	4.43(3.86)	90
	Content	1.80(3.06)	179
Santa Fe	GE	27.32(14.48)	11
	Order	28.16(13.35)	13
	Content	28.54(15.00)	14
Sym Reg A	GE	0.27(0.21)	37
	Order	0.41(0.28)	18
	Content	0.27(0.21)	37
Max	GE	11.42(1.39)	0
	Order	11.94(0.89)	0
	Content	11.44(1.33)	0

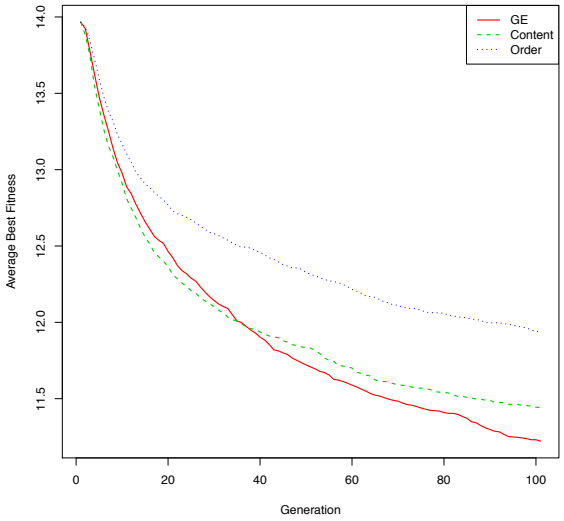
In the Even 5 Parity problem content mutation outperforms the other setups in both measured categories, and this result can be viewed as significantly better when the p-values of Table 6.4 consulted. The Santa Fe Ant problem displays varied results. Content



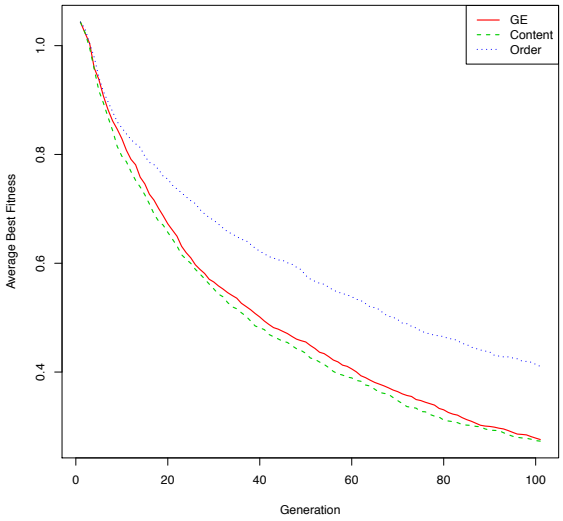
(a) Even 5 Parity - Crossover



(b) Santa Fe Ant - Crossover



(c) Max - Crossover



(d) Symbolic Regression A - Crossover

Figure 6.2: This figure shows the graphs for the average best fitness of the different mutation approaches on the problems examined with crossover. These graph represent the average of the best fitness at each generation over 250 runs. It can be seen that the order only approach to mutation suffers in performance from an early stage in the runs, in three out of four test problems. While the content only approach actually improves early search performance.

Table 6.4: The table displays the p-values resulting from comparing each pair of setups using the Wilcoxon Rank-Sum test. p-Values < 0.01666 are highlighted. The setups compared are the for the results seen in Table 6.3.

Even Five Parity			
	GE	Content	Order
GE	-	1.50E-03	1.09E-05
Content	1.50E-03	-	7.19E-13
Order	1.09E-05	7.19E-13	-
Santa Fe Ant			
	GE	Content	Order
GE	-	0.422	0.405
Content	0.422	-	0.843
Order	0.405	0.843	-
Symbolic Regression A			
	GE	Content	Order
GE	-	0.944	2.49E-08
Content	0.944	-	2.47E-05
Order	2.49E-08	2.47E-05	-
Max			
	GE	Content	Order
GE	-	0.068	5.12E-11
Content	0.068	-	8.52E-06
Order	5.12E-11	8.52E-06	-

mutation has the most number of successful solutions. However, the p-values indicate that the results for best fitness are too similar to draw significant conclusions from, thus all setups are deemed equal with respect to average best fitness performance. Looking at the Symbolic Regression problem it is impossible to distinguish a difference between the GE and content mutation approaches. However both are significantly better than the order approach. Finally the Max problem presents another problem where there is no identifiable difference between GE and content mutations, but once again the slightly worse performance of order mutation is deemed significantly worse.

These results show that focusing the search purely on the order codons (1:0 setup) tends to significantly give the worst results, with the Santa Fe Ant problem being the

exception. 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) due to the ripple effect of πGE.

The results obtained with content-only mutation (0:1 setup) tend to give the best results in terms of solutions found and fitness. When mutation events are 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 to the fixed GE order.

6.4.2 Results for 2:1 and 1:2 Ratios of Mutation

In order to investigate the contribution of the order search to the success of πGE, different ratios of order:content mutation have been explored, and are presented in this section. Table 6.5 (with crossover) shows the results obtained, while Table 6.6 displays the p-values indicating if there is significant difference between the different setups.

Examining the results, it is seen that content mutation outperforms the two mixed ratio setups on the Even Five problem, in terms of fitness and successful solutions found. The result is significantly better when Table 6.6 is consulted. On the Santa Fe Ant problem it is apparent that the order biased setup performs the best, but not significantly better than the other setups. It is worth noting that when comparing the 2:1 setup, to the order only approach in Table 6.3, that the addition of some content mutation has resulted better performance. Symbolic Regression presents a set of very similar results. There is little to separate the setups with respect to fitness and this is backed up by the p-values. However one thing to note is how the content biased setup found more successful solutions.

One final observation from the results has to come from the performance on the Max problem. In both the 1:2 and 2:1 setups, the average best fitness observed is competitive with the best results seen in Chapter 3. The surprising observation is that both the setups

Table 6.5: This table shows the results for the 2:1 and 1:2 balanced mutation approaches on a range of benchmark problems. 250 runs for each setup were performed and the results are shown below. In the table success denoted the number of solution found during the 250 runs. The average of the best fitness is presented with the standard deviation shown in brackets. The highlighted cells indicate the best performing approach. In the case where multiple approaches are highlighted, there is no statistical difference between the approaches.

Problem	Setup	Mean best fitness (stdev)	Successes
Even 5	Content	1.80(3.06)	179
	1:2	2.32(3.45)	150
	2:1	2.31(3.43)	123
Santa Fe	Content	28.54(15.00)	14
	1:2	24.74(14.33)	11
	2:1	21.43(13.33)	16
Sym Reg A	Content	0.27(0.21)	37
	1:2	0.24(0.22)	43
	2:1	0.24(0.22)	36
Max	Content	11.44(1.33)	0
	1:2	9.95(2.32)	0
	2:1	9.20(1.75)	0

exhibit such good performance, when the previous content and order only setups failed to make the same impact.

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 ratios both show good performance levels. However the variance in performance between the two ratio setups suggest, that trying to find the universal best setup for a set of problems using a ratio technique of search is quite hard. It is evident in that the 1:2 ratio setup outperforms the 2:1 setup, and vice versa, in different problems. This volatility however is not present in the purely content based setup, that has what can be deemed consistent good performance across all examined problems.

Table 6.6: The table displays the p-values resulting from comparing each pair of setups using the Wilcoxon Rank-Sum test. p-Values < 0.01666 are highlighted. The setups compared are the for the results seen in Table 6.5.

Even Five Parity			
	Content	1:2	2:1
Content	-	0.0015	1.09E-05
1:2	0.0015	-	7.19E-13
2:1	1.09E-05	7.19E-13	-
Santa Fe Ant			
	Content	1:2	2:1
Content	-	0.014	0.473
1:2	0.014	-	0.130
2:1	0.473	0.130	-
Symbolic Regression A			
	Content	1:2	2:1
Content	-	0.861	0.339
1:2	0.861	-	0.438
2:1	0.339	0.438	-
Max			
	Content	1:2	2:1
Content	-	0.574	0.095
1:2	0.574	-	0.237
2:1	0.095	0.237	-

While the 1:2 and 2:1 ratio setups were designed to see if more biased order search and content search would be advantageous, both setups are similar to a standard GE mutation operation (1:1). It would be of interest to see what the addition of a small amount of 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.

6.5 Discussion

In Chapter 5 the idea of trying to limit the order drift seen in πGE was broached. The mutation operator described in this chapter, provides a way to focus the mutation. With this in mind the content only and order only mutation operations were tested on the Santa

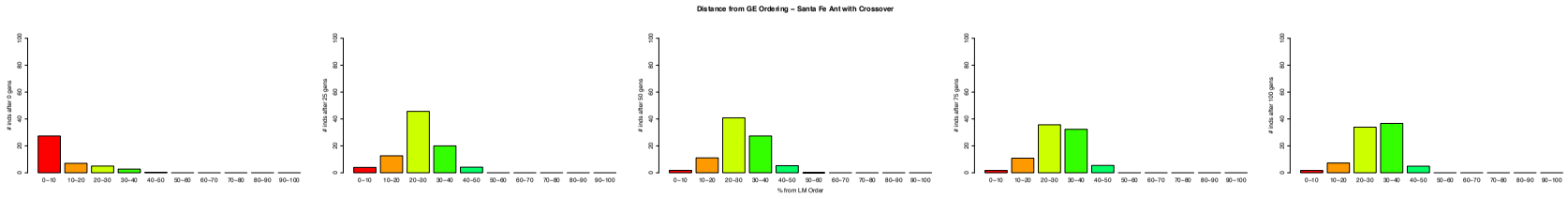
Fe Ant problem. Using the DFOB from Chapter 5 the orders were monitored over 100 runs, Figure 6.3 displays the results of this experiment.

Looking at the random initialisation setup, there appears to be no discernible difference between the distributions observed. However when the GE order initialisation runs are investigated some trends are seen. It can be clearly seen that with the content only setup there is little to no divergence from the initial ordering. This shows that the πGE ripple effect hasn't had a big effect on the orders observed. The order only setup shows a distribution that diverges slightly more than that seen in Chapter 5. Both setups however show good performance as was seen in Table 6.3. From this it is evident that the balancing of the mutation rates does allow for some control over πGE's orderings, that was previously not available to the system.

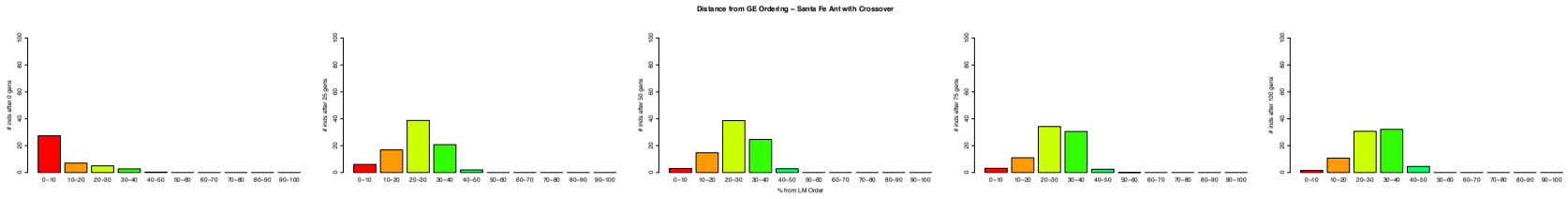
6.6 Summary

πGE provides GE with a GPM that exhibits a degree of freedom in its representation that has not been investigated for possible exploitation. Knowing this, a mutation operation was proposed that could focus mutation on a specific type of codon, or rebalance the standard GE mutation. This rebalancing can provide πGE with a more favourable mutation balance between order and content codons.

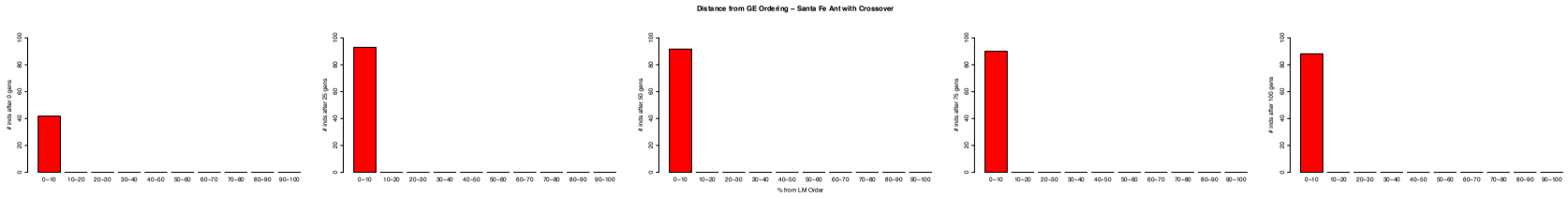
The results provide evidence that a better mutation operation has been found for πGE. The results support a rebalancing of the search towards the content codons. This rebalancing provides an comparable performance to standard mutation, while also offering increased performance on some problem domains. 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. It was also observed that the use of a rebalanced mutation can



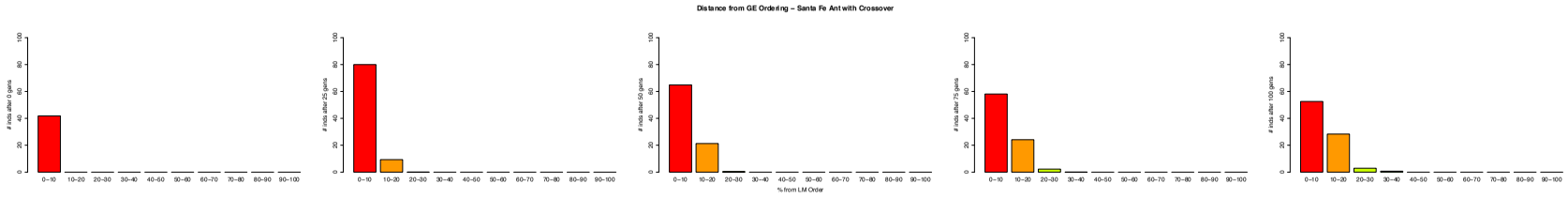
(a) Santa Fe Ant - Using Content Only Mutation



(b) Santa Fe Ant - Using Order Only Mutation



(c) Santa Fe Ant - Using Content Only Mutation and GE Initialisation



(d) Santa Fe Ant - Using Order Only Mutation and GE Initialisation

Figure 6.3: The figure shows the distribution of distances from a depth-first order on the Santa Fe Ant Problem using the focused mutation operation. Figures 6.3a and 6.3b show the distribution of orders used by πGE during evolution with random initialisation, no observable different can be seen when using the different mutation operations. When πGE is initialised to a fixed GE order it becomes clear that the content only mutation operation (Figure 6.3c) prevents the distribution of orders from drifting away from a GE ordering. Unlike the order only mutation (Figure 6.3d), where we begin to see the distribution drift away form a purely GE ordering.

address two issues noted in previous chapter. Firstly, rebalancing the mutation rate has narrowed the performance drop off seen in the Max problem. Secondly, the new mutation operation can control the order drift seen in Chapter 5.

This chapter brings to an end the experimental section of this thesis. The next chapter will address the conclusions of this thesis and outline future work.

Part III

Fin.

Chapter 7

Conclusions and Future Work

Having examined a variety of GPM approaches in GE, and investigated the reasons why certain GPMs may exhibit good performance even against an increasing search space, the conclusions are now discussed. A summary of the thesis is presented in Section 7.1. The contributions of the thesis are presented in Section 7.2, and future work is outlined in Section 7.3.

7.1 Thesis Summary

The aim of this thesis was to perform an investigation into the GE GPM. Specifically this research focused on two core mapping approaches, GE and πGE. To aid in the investigation some ancillary mapping approaches are also examined to give a spectrum of GPM expansion orders. The research questions tackled are outlined below.

Do alternative GPMs exist for GE that provide comparable or better performance? - Chapter 3 deals with this question in full. A comparison of four GPM approaches was performed, and from this it was deemed that πGE could be seen as a competitive with GE. The study displayed a slight issue for πGE with its performance on the Max problem.

A solution to this was seen in Chapter 6 through the use of focused mutation. Also of note was the fact the πGE exhibited very good performance without crossover, although it was not able to take as full an advantage of crossover as GE.

How does πGE present good performance given the added search space of having an evolvable GPM order? - Chapter 4 covered this question. The increase in search space size caused by πGEs evolvable order was quantified. A large increase in search space size was found. To understand πGEs good performance despite this increase, the search space connectivity of πGE and GE were compared. The addition of an evolvable order in πGE results in πGE exhibiting neutral mutations through order and showing a much higher degeneracy than GE. The number of connections between individuals via single mutations that πGE has when compared to GE is also significantly larger. As the only difference between GE and πGE is the increase in connectivity and degeneracy that the evolvable order in πGE adds to the algorithm, it is argued that this is what allows πGE to maintain competitive performance to GE.

What GPM orders is πGE actually using during evolution? - Chapter 5 provided the answer to this question. Two metrics were developed and used to monitor how far a πGE population was away from depth-first and breadth-first orders. It was found that πGE does not evolve towards a known order but rather a distribution of orders. Following this, a fixed order initialisation method was used to determine whether, if the population was initialised to a specific order, the population would diverge from this order. It was seen that the population did diverge, but at a slow rate.

Do genetic operations exist that may take advantage of these new GPMs? - Chapter 6 answered this question. The idea of focusing mutation on a specific type of πGE codon was implemented. This new operator was applied to the test problems with a variety of setups. It was seen that a focused mutation on only content codons provided πGE with a performance boost. This operator also allowed πGE to limit its ripple effect to content

codons. This is akin to GE's ripple effect and tries to allow πGE to maintain a close to fixed order during evolution. Some of the other setups with a ratio of order to content codon mutations showed promise in closing the performance gap πGE exhibited on the Max problem in Chapter 3.

7.2 Contributions

The contributions directly emanating from the experimental section of the thesis are outlined below. The research questions they address are noted at the end of each contribution.

Literature Review

An extensive review of Grammatical Evolution is presented in Chapter 2. The topics cover: investigations into the algorithm's behaviour, extensions to GE, application of GE, before a comprehensive list of implementations is finally compiled.

Suite of Genotype-Phenotype Maps (GPM) for GE

Several variants to the GE GPM are required for comparison of mapping approaches in this thesis. To this end a suite of GPMs for GE are detailed in Chapter 3 (Research Question 1).

Performance Comparison of Different Approaches to the GPM

Chapter 3 presents a detailed comparison of mapping approaches performed on a range of benchmark problems. The comparison is the foundation on which this thesis is based (Research Question 1).

Analysis of Genotype-Phenotype Map connectivity

Connectivity of a GPM relies heavily on the representation underpinning the approach. Chapter 4 provides an in-depth analysis of the two most promising GPMs

from this thesis, examined in Chapter 3. Visualisation of the comparison was aided by the usage of graphs and adjacency matrices (Research Question 2).

Methods of Monitoring Expansion Order in πGE

πGE has a variable order of expansion in the genotype-phenotype map. Understanding what orders are being used by the population or a subset of its individuals can provide valuable insight into its behaviour. Chapter 5 presents two metrics that can measure distance from known mapping orders (Research Question 2).

Analysis of Population Order Dynamics

Some GPMs use a variable expansion order during the mapping from genotype to phenotype. The order dynamics of one such mapping, πGE, are explored over a set of benchmark problems in Chapter 5 (Research Question 3).

Identification and Analysis of an Advanced Mutation Operation

Through investigation of other variants of the GPM, it is possible to discover added degrees of freedom in the mapping approach. These degrees of freedom may be exploited to produce improved performance of the GPM. Chapter 6 sees the realisation of one such operator, Focused Mutation Operation. The operator's behaviour is analysed, and different setups of the operator are compared across a range of problems (Research Question 4).

7.3 Future Work

The work carried out in this thesis has presented some interesting avenues for future work. This section sets out to identify the three main areas of investigation that present the best prospects for fruitful research.

7.3.1 Further Work with Operators

Chapter 6 presented a focused mutation operator for πGE. Results indicated that rebalancing of the mutation events could lead to performance gains. With this in mind exploration of an adaptive focused mutation operator may prove beneficial. Using the data being generated during a run the mutation could be adjusted to focus on order or content codons as desired.

πGE suffers from being a variable order GPM, in that it is very difficult to perform complex genetic operations on individuals. This leads to the opportunity in the future to devise efficient ways to perform operations such as subtree crossover on πGE. GE achieves good performance gains by using crossover [115, 123]. These gains are not seen in πGE due to the lack of a linearly linked genotype and phenotype. Providing πGE with an equivalent operation that transfers information in a similar fashion may prove advantageous as in GE. Chapter 6 left a lot of possible avenues for exploration with the interaction of the new mutation operation and crossover.

7.3.2 GE Mapping Islands

No free lunch theory implies that no one search algorithm can be the best at all problems. With this in mind I would propose borrowing from island based EC and present a paradigm for GE. There now exists several different GPM variants for GE, the idea behind Grammatical Evolution - Mapping Islands (GEMI) is that the population would be partitioned and sent to islands where each sub population would use a different GPM. After a certain duration of evolution on islands, the population would be gather together, evaluated, and then repartitioned. The same genotype and derivation tree data structures are used in all mappings. However, a new algorithm for mapping a derivation tree derived under one GPM to the genotype required to produce it under another GPM would be required.

GEMI can provide the benefits of all GPMs. If a sub-population on a GE island has converged to a local optimum, relocation to a πGE island may result in escape from the optimum given the added connectivity of πGE. Likewise a certain grammar may be more suited to a breadth first approach. By polling all individuals at island recombination the size of each island could be adapted based on performance of its population. This would have the added benefit of preventing wasted exploration on a GPM that wasn't providing productive individuals.

A similar approach could be taken with different grammars. Murphy [109] has documented success with the use of TAGE (Tree Adjunct Grammatical Evolution) that uses tree adjunct grammars instead of context free grammars. Hemberg [66] had success with meta grammars. This form of adaptive population GE is ripe for exploration.

7.3.3 Visualisations

During the course of this thesis the usage of visualisation has proven a very good way to investigate and understand how an algorithm works. The problem with these visualisations has been the size of the data needed to generate some of the images. Investigation into better and more efficient ways to generate these images would be beneficial. A real time monitoring device akin to GenViewer by Murphy et al. [104] here would be a nice feature to have to help monitor the health of a population with regards to tree shapes, tree sizes, or convergence, to name a few measures. A monitoring device may prove beneficial in an EC system that would run for prolonged periods of time and allow user input during a run.

Bibliography

[1] Raja Muhammad Atif Azad. *A Position Independent Representation for Evolutionary Automatic Programming Algorithms - The Chorus System.* PhD thesis, University of Limerick, Ireland, December 2003.

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

[3] Wolfgang Banzhaf, Guillaume Beslon, Steffen Christensen, James A. Foster, Francois Kepes, Virginie Lefort, Julian F. Miller, Miroslav Radman, and Jeremy J. Ramsden. Guidelines: From artificial evolution to computational evolution: a research agenda. *Nature Reviews Genetics*, 7(9):729–735, August 2006.

[4] Wolfgang Banzhaf and Andre Leier. Evolution on neutral networks in genetic programming. In Tina Yu, Rick L. Riolo, and Bill Worzel, editors, *Genetic Programming Theory and Practice III*, volume 9 of *Genetic Programming*, chapter 14, pages 207–221. Springer, Ann Arbor, 12-14 May 2005.

[5] Anthony Brabazon and Michael O'Neill. Anticipating bankruptcy reorganisation from raw financial data using grammatical evolution. In *Applications of Evolutionary Computing*, pages 368–377. Springer, 2003.

[6] Anthony Brabazon and Michael O'Neill. Bond-issuer credit rating with grammatical evolution. In *Applications of Evolutionary Computing*, pages 270–279. Springer, 2004.

[7] Anthony Brabazon and Michael O'Neill. Diagnosing corporate stability using grammatical evolution. *International Journal of Applied Mathematics and Computer Science*, 14(3):363–374, 2004.

[8] Anthony Brabazon and Michael O'Neill. *Biologically Inspired Algorithms for Financial Modelling*. Natural Computing Series. Springer, 2006.

[9] Markus Brameier and Wolfgang Banzhaf. *Linear Genetic Programming*. Number XVI in Genetic and Evolutionary Computation. Springer, 2007.

[10] Edmund K. Burke, Matthew R. Hyde, and Graham Kendall. Grammatical evolution of local search heuristics. *IEEE Transactions on Evolutionary Computation*, 16(3):406–417, June 2012.

[11] Jonathan Byrne. *Approaches to Evolutionary Architectural Design Exploration Using Grammatical Evolution*. PhD thesis, University College Dublin, Ireland, 2012.

[12] Jonathan Byrne, Erik Hemberg, Michael O'Neill, and Anthony Brabazon. A methodology for user directed search in evolutionary design. *Genetic Programming and Evolvable Machines*, pages 1–28, 2013.

[13] Jonathan Byrne, James McDermott, Michael O'Neill, and Anthony Brabazon. An analysis of the behaviour of mutation in grammatical evolution. In Anna Isabel Esparcia-Alcazar, Aniko Ekart, Sara Silva, Stephen Dignum, and A. Sima Uyar, editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 14–25, Istanbul, 7-9 April 2010. Springer.

[14] Jonathan Byrne, James McDermott, Michael O'Neill, and Anthony Brabazon. An analysis of the behaviour of mutation in grammatical evolution. In *EuroGP 2010 the 13th European Conference on Genetic Programming*. Springer, 2010.

[15] Jonathan Byrne, Michael O'Neill, and Anthony Brabazon. Structural and nodal mutation in grammatical evolution. In Guenther Raidl, Franz Rothlauf, Giovanni Squillero, Rolf Drechsler, Thomas Stuetzle, Mauro Birattari, Clare Bates Congdon, Martin Middendorf, Christian Blum, Carlos Cotta, Peter Bosman, Joern Grahl, Joshua Knowles, David Corne, Hans-Georg Beyer, Ken Stanley, Julian F. Miller, Jano van Hemert, Tom Lenaerts, Marc Ebner, Jaume Bacardit, Michael O'Neill, Massimiliano Di Penta, Benjamin Doerr, Thomas Jansen, Riccardo Poli, and Enrique Alba, editors, *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1881–1882, Montreal, Québec, Canada, 8-12 July 2009. ACM.

[16] Manuel Cebrian, Manuel Alfonseca, and Alfonso Ortega. Towards the validation of plagiarism detection tools by means of grammar evolution. *IEEE Transactions on Evolutionary Computation*, 13(3):477–485, June 2009.

[17] Li Chen. Macro-grammatical evolution for nonlinear time series modeling-a case study of reservoir inflow forecasting. *Engineering with Computers*, 27(4):393–404, 2011.

[18] Robert Cleary. Extending grammatical evolution with attribute grammars: An application to knapsack problems. Master of science in computer science, University of Limerick, University of Limerick, Ireland, 2005.

[19] Robert Cleary and Michael O'Neill. An attribute grammar decoder for the 01 multiconstrained knapsack problem. In Gunther R. Raidl and Jens Gottlieb, editors,

Evolutionary Computation in Combinatorial Optimization – EvoCOP 2005, volume 3448 of *LNCS*, pages 34–45, Lausanne, Switzerland, 30 March-1 April 2005. Springer Verlag.

[20] J. Manuel Colmenar, Jose L. Risco-Martin, David Atienza, and J. Ignacio Hidalgo. Multi-objective optimization of dynamic memory managers using grammatical evolution. In Natalio Krasnogor, Pier Luca Lanzi, Andries Engelbrecht, David Pelta, Carlos Gershenson, Giovanni Squillero, Alex Freitas, Marylyn Ritchie, Mike Preuss, Christian Gagne, Yew Soon Ong, Guenther Raidl, Marcus Gallager, Jose Lozano, Carlos Coello-Coello, Dario Landa Silva, Nikolaus Hansen, Silja Meyer-Nieberg, Jim Smith, Gus Eiben, Ester Bernado-Mansilla, Will Browne, Lee Spector, Tina Yu, Jeff Clune, Greg Hornby, Man-Leung Wong, Pierre Collet, Steve Gustafson, Jean-Paul Watson, Moshe Sipper, Simon Poulding, Gabriela Ochoa, Marc Schoenauer, Carsten Witt, and Anne Auger, editors, *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1819–1826, Dublin, Ireland, 12-16 July 2011. ACM.

[21] F. H. C. Crick. Central Dogma of Molecular Biology. *Nature*, 227(5258):561–563, August 1970.

[22] Wei Cui, Anthony Brabazon, and Michael O'Neill. Evolving dynamic trade execution strategies using grammatical evolution. In Cecilia Di Chio, Anthony Brabazon, Gianni A. Di Caro, Marc Ebner, Muddassar Farooq, Andreas Fink, Jorn Grahl, Gary Greenfield, Penousal Machado, Michael O'Neill, Ernesto Tarantino, and Neil Urquhart, editors, *EvoFIN*, volume 6025 of *LNCS*, pages 192–201, Istanbul, 7-9 April 2010. Springer.

[23] Wei Cui, Anthony Brabazon, and Michael O'Neill. Evolving efficient limit order strategy using grammatical evolution. In *2010 IEEE World Congress on Computational*

Intelligence, pages 2408–2413, Barcelona, Spain, 18-23 July 2010. IEEE Computational Intelligence Society, IEEE Press.

[24] Ian Dempsey. *Grammatical Evolution in Dynamic Environments*. PhD thesis, University College Dublin, Ireland, 2007.

[25] Ian Dempsey, Michael O'Neill, and Anthony Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments*. Studies in Computational Intelligence. Springer, 2009.

[26] Miguel Nuno Fialho dos Santos Nicolau. *Genetic Algorithms using Grammatical Evolution*. PhD thesis, University Of Limerick, 2006.

[27] John H. Drake, Nikolaos Kililis, and Ender Ozcan. Generation of VNS components with grammatical evolution for vehicle routing. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu, editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 25–36, Vienna, Austria, 3-5 April 2013. Springer Verlag.

[28] Jan Drchal and Miroslav Šnorek. Tree-based indirect encodings for evolutionary development of neural networks. In *Artificial Neural Networks-ICANN 2008*, pages 839–848. Springer, 2008.

[29] Marc Ebner, Patrick Langguth, Juergen Albert, Mark Shackleton, and Rob Shipman. On neutral networks and evolvability. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1–8, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.

[30] Marc Ebner, Mark Shackleton, and Rob Shipman. How neutral networks influence evolvability. *Complexity*, 7(2):19–33, 2001.

[31] Gabi Escuela, Gabriela Ochoa, and Natalio Krasnogor. Evolving L-systems to capture protein structure native conformations. In Maarten Keijzer, Andrea Tettamanzi, Pierre Collet, Jano I. van Hemert, and Marco Tomassini, editors, *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of *Lecture Notes in Computer Science*, pages 74–84, Lausanne, Switzerland, 30 March - 1 April 2005. Springer.

[32] David Fagan. Genotype-Phenotype Mapping in Dynamic Environments with Grammatical Evolution. In Miguel Nicolau, editor, *GECCO 2011 Graduate students workshop*, pages 783–786, Dublin, Ireland, 12-16 July 2011. ACM.

[33] David Fagan, Erik Hemberg, Miguel Nicolau, Michael O'Neill, and Séan McGarraghy. Towards Adaptive Mutation in Grammatical Evolution. In Terry Soule, Anne Auger, Jason Moore, David Pelta, Christine Solnon, Mike Preuss, Alan Dorin, Yew-Soon Ong, Christian Blum, Dario Landa Silva, Frank Neumann, Tina Yu, Aniko Ekart, Wil Browne, Tim Kovacs, Man-Leung Wong, Clara Pizzuti, Jon Rowe, Tobias Friedrich, Giovanni Squillero, Nicolas Bredeche, Stephen Smith, Alison Motsinger-Rei, Jose Lozano, Martin Pelikan, Silja Meyer-Nienber, Christian Igel, Greg Hornby, Rene Doursat, Steve Gustafson, Gustavo Olague, Shin Yoo, John Clark, Gabriela Ochoa, Gisele Pappa, Fernando Lobo, Daniel Tauritz, Jurgen Branke, and Kalyanmoy Deb, editors, *GECCO Companion '12: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, pages 1481–1482, Philadelphia, Pennsylvania, USA, 7-11 July 2012. ACM.

[34] David Fagan, Erik Hemberg, Michael O'Neill, and Séan McGarraghy. Fitness Reactive Mutation in Grammatical Evolution. In Radomil Matousek, editor, *18th International Conference on Soft Computing, MENDEL 2012*, pages 144–149, Brno, Czech Republic, 27-29 June 2012. Brno University of Technology.

[35] David Fagan, Erik Hemberg, Michael O'Neill, and Séan McGarraghy. Understanding Expansion Order and Phenotypic Connectivity in πGE. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu, editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 37–48, Vienna, Austria, 3-5 April 2013. Springer Verlag.

[36] David Fagan, Miguel Nicolau, Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Dynamic Ant: Introducing a New Benchmark for Genetic Programming in Dynamic Environments. In Natalio Krasnogor, Pier Luca Lanzi, Andries Engelbrecht, David Pelta, Carlos Gershenson, Giovanni Squillero, Alex Freitas, Marylyn Ritchie, Mike Preuss, Christian Gagne, Yew Soon Ong, Guenther Raidl, Marcus Gallager, Jose Lozano, Carlos Coello-Coello, Dario Landa Silva, Nikolaus Hansen, Silja Meyer-Nieberg, Jim Smith, Gus Eiben, Ester Bernado-Mansilla, Will Browne, Lee Spector, Tina Yu, Jeff Clune, Greg Hornby, Man-Leung Wong, Pierre Collet, Steve Gustafson, Jean-Paul Watson, Moshe Sipper, Simon Poulding, Gabriela Ochoa, Marc Schoenauer, Carsten Witt, and Anne Auger, editors, *GECCO '11: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 183–184, Dublin, Ireland, 12-16 July 2011. ACM.

[37] David Fagan, Miguel Nicolau, Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Dynamic ant: Introducing a new benchmark for genetic programming in dynamic environments. Technical Report UCD-CSI-2011-04, School of Computer Science and Informatics, University College Dublin, `http://www.csi.ucd.ie/biblio`, 2011.

[38] David Fagan, Miguel Nicolau, Erik Hemberg, Michael O'Neill, Anthony Brabazon, and Séan McGarraghy. Investigation of the Performance of Different Mapping Orders for GE on the Max Problem. In Sara Silva, James A. Foster, Miguel Nicolau,

Mario Giacobini, and Penousal Machado, editors, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 286–297, Turin, Italy, 27-29 April 2011. Springer Verlag.

[39] David Fagan, Miguel Nicolau, Michael O'Neill, Edgar Galván-López, Anthony Brabazon, and Séan McGarraghy. Investigating Mapping Order in πGE. In *2010 IEEE World Congress on Computational Intelligence*, pages 3058–3064, Barcelona, Spain, 18-23 July 2010. IEEE Computational Intelligence Society, IEEE Press.

[40] David Fagan, Michael O'Neill, Edgar Galván-López, Anthony Brabazon, and Séan McGarraghy. An Analysis of Genotype-Phenotype Maps in Grammatical Evolution. In Anna Isabel Esparcia-Alcazar, Aniko Ekárt, Sara Silva, Stephen Dignum, and A. Sima Uyar, editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 62–73, Istanbul, 7-9 April 2010. Springer.

[41] Jose-Luis Fernandez-Villacanas Martin and Mark Shackleton. Investigation of the importance of the genotype-phenotype mapping in information retrieval. *Future Generation Computer Systems*, 19(1), 2003.

[42] Edgar Galván-López, David Fagan, Eoin Murphy, John Mark Swafford, Alexandros Agapitos, Michael O'Neill, and Anthony Brabazon. Comparing the Performance of the Evolvable PiGrammatical Evolution Genotype-Phenotype Map to Grammatical Evolution in the Dynamic Ms. Pac-Man Environment. In *2010 IEEE World Congress on Computational Intelligence*, pages 1587–1594, Barcelona, Spain, 18-23 July 2010. IEEE Computational Intelligence Society, IEEE Press.

[43] Edgar Galván-López, John Mark Swafford, Michael O'Neill, and Anthony Brabazon. Evolving a Ms. PacMan controller using grammatical evolution. In Cecilia Di Chio,

Stefano Cagnoni, Carlos Cotta, Marc Ebner, Aniko Ekart, Anna I. Esparcia-Alcazar, Chi-Keong Goh, Juan J. Merelo, Ferrante Neri, Mike Preuss, Julian Togelius, and Georgios N. Yannakakis, editors, *EvoGAMES*, volume 6024 of *LNCS*, pages 161–170, Istanbul, 7-9 April 2010. Springer.

[44] Chris Gathercole and Peter Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation*, GECCO '96, pages 291–296, Cambridge, MA, USA, 1996. MIT Press.

[45] Loukas Georgiou. *Constituent Grammatical Evolution*. PhD thesis, School of Computer Science, Bangor University, LL57 1UT, Gwynedd, UK, August 2012.

[46] Loukas Georgiou and William J. Teahan. jGE - A java implementation of grammatical evolution. In *10th WSEAS International Conference on Systems*, pages 534–869, Athens, Greece, July 10-15 2006.

[47] Loukas Georgiou and William J. Teahan. Constituent grammatical evolution. In Toby Walsh, editor, *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 1261–1268, Barcelona, Spain, 16-22 July 2011. AAAI Press.

[48] George Georgoulas, Dimitris Gavrilis, Ioannis G. Tsoulos, Chrysostomos Stylios, Joao Bernardes, and Peter P. Groumpos. Novel approach for fetal heart rate classification introducing grammatical evolution. *Biomedical Signal Processing and Control*, 2(2):69–79, 2007.

[49] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[50] Richard W Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[51] Simon Harding, Julian Francis Miller, and Wolfgang 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.

[52] Robin Harper. GE, explosive grammars and the lasting legacy of bad initialisation. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, Barcelona, Spain, 18-23 July 2010. IEEE Press.

[53] Robin Harper. Co-evolving robocode tanks. In Natalio Krasnogor, Pier Luca Lanzi, Andries Engelbrecht, David Pelta, Carlos Gershenson, Giovanni Squillero, Alex Freitas, Marylyn Ritchie, Mike Preuss, Christian Gagne, Yew Soon Ong, Guenther Raidl, Marcus Gallager, Jose Lozano, Carlos Coello-Coello, Dario Landa Silva, Nikolaus Hansen, Silja Meyer-Nieberg, Jim Smith, Gus Eiben, Ester Bernado-Mansilla, Will Browne, Lee Spector, Tina Yu, Jeff Clune, Greg Hornby, Man-Leung Wong, Pierre Collet, Steve Gustafson, Jean-Paul Watson, Moshe Sipper, Simon Poulding, Gabriela Ochoa, Marc Schoenauer, Carsten Witt, and Anne Auger, editors, *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1443–1450, Dublin, Ireland, 12-16 July 2011. ACM.

[54] Robin Harper and Alan Blair. A structure preserving crossover in grammatical evolution. In David Corne, Zbigniew Michalewicz, Marco Dorigo, Gusz Eiben, David Fogel, Carlos Fonseca, Garrison Greenwood, Tan Kay Chen, Guenther Raidl, Ali Zalzala, Simon Lucas, Ben Paechter, Jennifier Willies, Juan J. Merelo Guervos, Eugene Eberbach, Bob McKay, Alastair Channon, Ashutosh Tiwari, L. Gwenn Volkert, Dan

Ashlock, and Marc Schoenauer, editors, *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2537–2544, Edinburgh, UK, 2-5 September 2005. IEEE Press.

[55] Robin Harper and Alan Blair. A self-selecting crossover operator. In Gary G. Yen, Lipo Wang, Piero Bonissone, and Simon M. Lucas, editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 5569–5576, Vancouver, 6-21 July 2006. IEEE Press.

[56] Robin Thomas Ross Harper. *Enhancing Grammatical Evolution*. PhD thesis, School of Computer Science and Engineering, The University of New South Wales, Sydney 2052, Australia, 2009.

[57] Erik Hemberg. An exploration of learning and grammars in grammatical evolution. In Anna I. Esparcia, Ying ping Chen, Gabriela Ochoa, Ender Ozcan, Marc Schoenauer, Anne Auger, Hans-Georg Beyer, Nikolaus Hansen, Steffen Finck, Raymond Ros, Darrell Whitley, Garnett Wilson, Simon Harding, W. B. Langdon, Man Leung Wong, Laurence D. Merkle, Frank W. Moore, Sevan G. Ficici, William Rand, Rick Riolo, Nawwaf Kharma, William R. Buckley, Julian Miller, Kenneth Stanley, Jaume Bacardit, Will Browne, Jan Drugowitsch, Nicola Beume, Mike Preuss, Stephen L. Smith, Stefano Cagnoni, Jim DeLeo, Alexandru Floares, Aaron Baughman, Steven Gustafson, Maarten Keijzer, Arthur Kordon, Clare Bates Congdon, Laurence D. Merkle, and Frank W. Moore, editors, *GECCO-2009 Graduate student workshop*, pages 2705–2708, Montreal, 8-12 July 2009. ACM.

[58] Erik Hemberg, Conor Gilligan, Michael O'Neill, and Anthony Brabazon. A grammatical genetic programming approach to modularity in genetic algorithms. In Marc Ebner, Michael O'Neill, Anikó Ekárt, Leonardo Vanneschi, and Anna Isabel

Esparcia-Alcázar, editors, *Proceedings of the 10th European Conference on Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 1–11, Valencia, Spain, 11-13 April 2007. Springer.

[59] Erik Hemberg, Lester Ho, Michael O'Neill, and Holger Claussen. A symbolic regression approach to manage femtocell coverage using grammatical genetic programming. In Steven Gustafson and Ekaterina Vladislavleva, editors, *3rd symbolic regression and modeling workshop for GECCO 2011*, pages 639–646, Dublin, Ireland, 12-16 July 2011. ACM.

[60] Erik Hemberg, Lester Ho, Michael O'Neill, and Holger Claussen. A comparison of grammatical genetic programming grammars for controlling femtocell network coverage. *Genetic Programming and Evolvable Machines*, 14(1):65–93, March 2013.

[61] Erik Hemberg, Lester Ho, Michael O'Neill, and Holger Clausssen. Comparing the robustness of grammatical genetic programming solutions for femtocell algorithms. In Terry Soule, Anne Auger, Jason Moore, David Pelta, Christine Solnon, Mike Preuss, Alan Dorin, Yew-Soon Ong, Christian Blum, Dario Landa Silva, Frank Neumann, Tina Yu, Aniko Ekart, Will Browne, Tim Kovacs, Man-Leung Wong, Clara Pizzuti, Jon Rowe, Tobias Friedrich, Giovanni Squillero, Nicolas Bredeche, Stephen Smith, Alison Motsinger-Reif, Jose Lozano, Martin Pelikan, Silja Meyer-Nienberg, Christian Igel, Greg Hornby, Rene Doursat, Steve Gustafson, Gustavo Olague, Shin Yoo, John Clark, Gabriela Ochoa, Gisele Pappa, Fernando Lobo, Daniel Tauritz, Jurgen Branke, and Kalyanmoy Deb, editors, *GECCO Companion '12: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion*, pages 1525–1526, Philadelphia, Pennsylvania, USA, 7-11 July 2012. ACM.

[62] Erik Hemberg, Nic McPhee, Michael O'Neill, and Anthony Brabazon. Pre-, in- and postfix grammars for symbolic regression in grammatical evolution. In T. M. McGinnity, editor, *IEEE Workshop and Summer School on Evolutionary Computing*, pages 18–22, University of Ulster, Derry, Northern Ireland, 18-22 August 2008.

[63] Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Altering search rates of the meta and solution grammars in the mGGA. In Michael O'Neill, Leonardo Vanneschi, Steven Gustafson, Anna Isabel Esparcia Alcazar, Ivanoe De Falco, Antonio Della Cioppa, and Ernesto 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 March 2008. Springer.

[64] Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Grammatical bias and building blocks in meta-grammar grammatical evolution. In Jun Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 3775–3782, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.

[65] Erik Hemberg, Michael O'Neill, and Anthony Brabazon. An investigation into automatically defined function representations in grammatical evolution. In R. Matousek and L. Nolle, editors, *15th International Conference on Soft Computing, Mendel'09*, Brno, Czech Republic, 24-26 June 2009.

[66] Erik Anders Pieter Hemberg. *An Exploration of Grammars in Grammatical Evolution*. PhD thesis, University College Dublin, Ireland, 17 September 2010.

[67] Martin Hemberg and Una-May O'Reilly. Extending grammatical evolution to evolve digital surfaces with genr8. In Maarten Keijzer, Una-May O'Reilly, Simon M. Lucas, Ernesto Costa, and Terence Soule, editors, *Genetic Programming 7th European Con-*

ference, EuroGP 2004, Proceedings, volume 3003 of *LNCS*, pages 299–308, Coimbra, Portugal, 5-7 April 2004. Springer-Verlag.

[68] Jonatan Hugosson, Erik Hemberg, Anthony Brabazon, and Michael O'Neill. An investigation of the mutation operator using different representations in grammatical evolution. In *2nd International Symposium "Advances in Artificial Intelligence and Applications"*, volume 2, pages 409–419, Wisla, Poland, October 15-17 2007.

[69] Jonatan Hugosson, Erik Hemberg, Anthony Brabazon, and Michael O'Neill. Genotype representations in grammatical evolution. *Applied Soft Computing*, 10(1):36–43, January 2010.

[70] Eric A. Jones and William T. Joines. Genetic design of electronic circuits. In Scott Brave and Annie S. Wu, editors, *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pages 125–133, Orlando, Florida, USA, 13 July 1999.

[71] Terry Jones. *Evolutionary Algorithms, Fitness Landscapes, and Search*. PhD thesis, University of New Mexico, 1995.

[72] Terry Jones. One operator, one landscape. *Santa Fe Institute Technical Report*, pages 95–02, 1995.

[73] Hillol Kargupta. Gene expression: The missing link in evolutionary computation. Technical report, Los Alamos National Lab., NM (United States), 1997.

[74] Muhammad Karim and Conor Ryan. Degeneracy reduction or duplicate elimination? an analysis on the performance of attributed grammatical evolution with lookahead to solve the multiple knapsack problem. In David Pelta, Natalio Krasnogor, Dan Dumitrescu, Camelia Chira, and Rodica Lung, editors, *Nature Inspired Cooperative*

Strategies for Optimization (NICSO 2011), volume 387 of *Studies in Computational Intelligence*, pages 247–266, Cluj-Napoca, Romania, 2012. Springer.

[75] Muhammad Rezaul Karim and Conor Ryan. A new approach to solving 0-1 multiconstraint knapsack problems using attribute grammar with lookahead. In Sara Silva, James A. Foster, Miguel Nicolau, Mario Giacobini, and Penousal Machado, editors, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 250–261, Turin, Italy, 27-29 April 2011. Springer Verlag.

[76] Muhammad Rezaul Karim and Conor Ryan. A simple improvement heuristic for attributed grammatical evolution with lookahead to solve the multiple knapsack problem. In Geuk Lee, Daniel Howard, and Dominik Slezak, editors, *Proceedings of the 5th International Conference on Convergence and Hybrid Information Technology, ICHIT 2011*, volume 6935 of *Lecture Notes in Computer Science*, pages 274–281, Daejeon, Korea, September 22-24 2011. Springer.

[77] Muhammad Rezaul Karim and Conor Ryan. Attributed grammatical evolution with lookahead for the multiple knapsack problem. *Memetic Computing*, 4(4):279–302, 2012.

[78] Muhammad Rezaul Karim and Conor Ryan. Sensitive ants are sensible ants. In Terry Soule, Anne Auger, Jason Moore, David Pelta, Christine Solnon, Mike Preuss, Alan Dorin, Yew-Soon Ong, Christian Blum, Dario Landa Silva, Frank Neumann, Tina Yu, Aniko Ekart, Will Browne, Tim Kovacs, Man-Leung Wong, Clara Pizzuti, Jon Rowe, Tobias Friedrich, Giovanni Squillero, Nicolas Bredeche, Stephen Smith, Alison Motsinger-Reif, Jose Lozano, Martin Pelikan, Silja Meyer-Nienberg, Christian Igel, Greg Hornby, Rene Doursat, Steve Gustafson, Gustavo Olague, Shin Yoo,

John Clark, Gabriela Ochoa, Gisele Pappa, Fernando Lobo, Daniel Tauritz, Jurgen Branke, and Kalyanmoy Deb, editors, *GECCO '12: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 775–782, Philadelphia, Pennsylvania, USA, 7-11 July 2012. ACM.

[79] M. Keijzer, V. Babovic, C. Ryan, M. O'Neill, and M. Cattolico. Adaptive logic programming. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 42–49, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[80] Maarten Keijzer, Michael O'Neill, Conor Ryan, and Mike Cattolico. Grammatical evolution rules: The mod and the bucket rule. In A. G. B. Tettamanzi J. A. Foster, E. Lutton, J. Miller, C. Ryan, editor, *Proceedings of the Fifth European Conference on Genetic Programming (EuroGP-2002)*, volume 2278 of *LNCS*, pages 123–130, Kinsale, Ireland, 2002. Springer Verlag.

[81] Maarten Keijzer, Conor Ryan, Michael O'Neill, Mike Cattolico, and Vladin Babovic. Ripple crossover in genetic programming. In J. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tetamanzi, and W. B. Langdon, editors, *Proceedings of the Fourth European Conference on Genetic Programming (EuroGP-2001)*, volume 2038 of *LNCS*, pages 74–86, Lake Como, Italy, 2001. Springer Verlag.

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

[83] Robert E. Keller and Wolfgang 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.

[84] Robert E. Keller and Wolfgang Banzhaf. The evolution of genetic code in genetic programming. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert 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.

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

[86] Motoo Kimura. *The neutral theory of molecular evolution*. Cambridge University Press, 1985.

[87] Motoo Kimura. *Population genetics, molecular evolution, and the neutral theory: selected papers*. University of Chicago Press, 1995.

[88] John R. Koza. Genetically breeding populations of computer programs to solve problems in artificial intelligence. In *Proceedings of the Second International Conference on Tools for AI*, pages 819–827, Herndon, Virginia, USA, 6-9 November 1990. IEEE Computer Society Press, Los Alamitos, CA, USA.

[89] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[90] John R. Koza and Forrest H Bennett III. Automatic synthesis, placement, and routing of electrical circuits by means of genetic programming. In Lee Spector,

William B. Langdon, Una-May O'Reilly, and Peter J. Angeline, editors, *Advances in Genetic Programming 3*, chapter 6, pages 105–134. MIT Press, Cambridge, MA, USA, June 1999.

[91] John R. Koza, Martin A. Keane, Matthew J. Streeter, William Mydlowec, Jessen Yu, and Guido Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.

[92] John R. Koza and Riccardo Poli. Genetic programming. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 5, pages 127–164. Springer, 2005.

[93] Eugene F Krause. Taxicab geometry: An adventure in non-euclidean geometry, 2012.

[94] W. B. Langdon and R. Poli. Why ants are hard. Technical Report CSRP-98-4, University of Birmingham, School of Computer Science, January 1998. Presented at GP-98.

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

[96] Sean Luke. *The ECJ Owner's Manual – A User Manual for the ECJ Evolutionary Computation Library*, zeroth edition, online version 0.2 edition, October 2010.

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

[98] James McDermott and Paula Carroll. Program optimisation with dependency injection. In Krzysztof Krawiec, Alberto Moraglio, Ting Hu, A. Sima Uyar, and Bin Hu,

editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 133–144, Vienna, Austria, 3-5 April 2013. Springer Verlag.

[99] A. R. McIntyre and M. I. Heywood. A grammatical evolution multi-classifier through crowding. In Bart Rylander, editor, *Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 219–226, Chicago, USA, 12–16 July 2003.

[100] Robert I. Mckay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'Neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11:365–396, September 2010.

[101] Julian F. Miller and Riccardo Poli. Editorial to tenth anniversary issue on progress in genetic programming and evolvable machines. *Genetic Programming and Evolvable Machines*, 11(3/4):247–250, September 2010. Editorial: Tenth Anniversary Issue: Progress in Genetic Programming and Evolvable Machines.

[102] Julian F. Miller and Stephen L. Smith. Redundancy and computational efficiency in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 10(2):167–174, April 2006.

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

[104] Eoin Murphy and Erik Hemberg. Gen Viewer. `http://ncra.ucd.ie/GEVA/GenViewer.zip`, 2011.

[105] Eoin Murphy, Erik Hemberg, Miguel Nicolau, Michael O'Neill, and Anthony Brabazon. Grammar bias and initialisation in grammar based genetic programming. In Alberto Moraglio, Sara Silva, Krzysztof Krawiec, Penousal Machado, and Carlos

Cotta, editors, *Proceedings of the 15th European Conference on Genetic Programming, EuroGP 2012*, volume 7244 of *LNCS*, pages 85–96, Malaga, Spain, 11-13 April 2012. Springer Verlag.

[106] Eoin Murphy, Miguel Nicolau, Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Differential gene expression with tree-adjunct grammars. In Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone, editors, *Parallel Problem Solving from Nature, PPSN XII (part 1)*, volume 7491 of *Lecture Notes in Computer Science*, pages 377–386, Taormina, Italy, September 1-5 2012. Springer.

[107] Eoin Murphy, Michael O'Neill, and Anthony Brabazon. A comparison of GE and TAGE in dynamic environments. In Natalio Krasnogor, Pier Luca Lanzi, Andries Engelbrecht, David Pelta, Carlos Gershenson, Giovanni Squillero, Alex Freitas, Marylyn Ritchie, Mike Preuss, Christian Gagne, Yew Soon Ong, Guenther Raidl, Marcus Gallager, Jose Lozano, Carlos Coello-Coello, Dario Landa Silva, Nikolaus Hansen, Silja Meyer-Nieberg, Jim Smith, Gus Eiben, Ester Bernado-Mansilla, Will Browne, Lee Spector, Tina Yu, Jeff Clune, Greg Hornby, Man-Leung Wong, Pierre Collet, Steve Gustafson, Jean-Paul Watson, Moshe Sipper, Simon Poulding, Gabriela Ochoa, Marc Schoenauer, Carsten Witt, and Anne Auger, editors, *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1387–1394, Dublin, Ireland, 12-16 July 2011. ACM.

[108] Eoin Murphy, Michael O'Neill, and Anthony Brabazon. Examining mutation landscapes in grammar based genetic programming. In Sara Silva, James A. Foster, Miguel Nicolau, Mario Giacobini, and Penousal Machado, editors, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 130–141, Turin, Italy, 27-29 April 2011. Springer Verlag. Best paper.

[109] Eoin Murphy, Michael O'Neill, Edgar Galván-López, and Anthony Brabazon. Tree-adjunct grammatical evolution. In *2010 IEEE World Congress on Computational Intelligence*, pages 4449–4456, Barcelona, Spain, 18-23 July 2010. IEEE Computational Intelligence Society, IEEE Press.

[110] James Murphy, Michael O'Neill, and Hamish Carr. Exploring grammatical evolution for horse gait optimisation. In Leonardo Vanneschi, Steven Gustafson, Alberto Moraglio, Ivanoe De Falco, and Marc Ebner, editors, *Proceedings of the 12th European Conference on Genetic Programming, EuroGP 2009*, volume 5481 of *LNCS*, pages 183–194, Tuebingen, April 15-17 2009. Springer.

[111] James Murphy, Michael O'Neill, and Hamish Carr. Gait optimisation for distinct horse models using grammatical evolution. In R. Matousek and L. Nolle, editors, *15th International Conference on Soft Computing, Mendel'09*, Brno, Czech Republic, 24-26 June 2009.

[112] James E. Murphy. *Applications of Evolutionary Computation to Quadrupedal Animal Animation.* PhD thesis, School of Computer Science and Informatics, University College Dublin, Ireland, March 2011.

[113] Miguel Nicolau, Anne Auger, and Conor Ryan. Functional dependency and degeneracy: Detailed analysis of the GAuGE system. In Pierre Liardet, Pierre Collet, Cyril Fonlupt, Evelyne Lutton, and Marc Schoenauer, editors, *Evolution Artificielle, 6th International Conference*, volume 2936 of *Lecture Notes in Computer Science*, pages 15–26, Marseilles, France, 27-30 October 2003. Springer. Revised Selected Papers.

[114] Miguel Nicolau, Anne Auger, and Conor Ryan. Investigating degenerate code and gene dependency in the GAuGE system. In Alwyn M. Barry, editor, *GECCO 2003:*

Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, pages 150–157, Chigaco, 11 July 2003. AAAI.

[115] Miguel Nicolau, Michael O'Neill, and Anthony Brabazon. Termination in grammatical evolution: Grammar design, wrapping, and tails. In Xiaodong Li, editor, *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*, pages 2381–2388, Brisbane, Australia, 10-15 June 2012.

[116] Miguel Nicolau and Conor Ryan. How functional dependency adapts to salience hierarchy in the GAuGE system. In Conor Ryan, Terence Soule, Maarten Keijzer, Edward Tsang, Riccardo Poli, and Ernesto Costa, editors, *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 157–167, Essex, 14-16 April 2003. Springer-Verlag.

[117] Miguel Nicolau and Conor Ryan. Crossover, population dynamics, and convergence in the GAuGE system. In Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund Burke, Paul Darwen, Dipankar Dasgupta, Dario Floreano, James Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andy Tyrrell, editors, *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 1414–1425, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[118] Miguel Nicolau and Conor Ryan. Efficient crossover in the GAuGE system. In Maarten Keijzer, Una-May O'Reilly, Simon M. Lucas, Ernesto Costa, and Terence Soule, editors, *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, volume 3003 of *LNCS*, pages 125–137, Coimbra, Portugal, 5-7 April 2004. Springer-Verlag.

[119] Miguel Nicolau and Conor Ryan. Solving sudoku with the GAuGE system. In Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, and Anikó Ekárt, editors, *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 213–224, Budapest, Hungary, 10 - 12 April 2006. Springer.

[120] Miguel Nicolau, Matthew Saunders, Michael O'Neill, Bruce Osborne, and Anthony Brabazon. Evolving interpolating models of net ecosystem CO2 exchange using grammatical evolution. In Alberto Moraglio, Sara Silva, Krzysztof Krawiec, Penousal Machado, and Carlos Cotta, editors, *Proceedings of the 15th European Conference on Genetic Programming, EuroGP 2012*, volume 7244 of *LNCS*, pages 134–145, Malaga, Spain, 11-13 April 2012. Springer Verlag.

[121] Miguel Nicolau and Darwin Slattery. *libGE*, 2006. for version 0.27alpha1, 14 September 2006.

[122] M. O'Neill and C. Ryan. Crossover in grammatical evolution: A smooth operator? In R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin, and T. C. Fogarty, editors, *Proceedings of the Third European Conference on Genetic Programming (EuroGP-2000)*, volume 1802 of *LNCS*, pages 149–162, Edinburgh, Scotland, 2000. Springer Verlag.

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

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

[125] Michael O'Neill and Anthony Brabazon. Evolving a Logo Design Using Lindenmayer Systems, Postscript and Grammatical Evolution. In Jun Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 3788–3794, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.

[126] Michael O'Neill, Anthony Brabazon, Miguel Nicolau, Sean Mc Garraghy, and Peter Keenan. πgrammatical evolution. In *Genetic and Evolutionary Computation – GECCO-2004, Part II*, LNCS3103. Springer-Verlag, 2004.

[127] Michael O'Neill, Anthony Brabazon, Miguel Nicolau, Sean Mc Garraghy, and Peter Keenan. πgrammatical evolution. In Kalyanmoy Deb, Riccardo Poli, Wolfgang Banzhaf, Hans-Georg Beyer, Edmund Burke, Paul Darwen, Dipankar Dasgupta, Dario Floreano, James Foster, Mark Harman, Owen Holland, Pier Luca Lanzi, Lee Spector, Andrea Tettamanzi, Dirk Thierens, and Andy Tyrrell, editors, *Genetic and Evolutionary Computation – GECCO-2004, Part II*, volume 3103 of *Lecture Notes in Computer Science*, pages 617–629, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[128] Michael O'Neill, Robert Cleary, and Nikola Nikolov. Solving knapsack problems with attribute grammars. In R. Poli, S. Cagnoni, M. Keijzer, E. Costa, F. Pereira, G. Raidl, S. C. Upton, D. Goldberg, H. Lipson, E. de Jong, J. Koza, H. Suzuki, H. Sawai, I. Parmee, M. Pelikan, K. Sastry, D. Thierens, W. Stolzmann, P. L. Lanzi, S. W. Wilson, M. O'Neill, C. Ryan, T. Yu, J. F. Miller, I. Garibay, G. Holifield, A. S. Wu, T. Riopka, M. M. Meysenburg, A. W. Wright, N. Richter, J. H. Moore, M. D. Ritchie, L. Davis, R. Roy, and M. Jakiela, editors, *GECCO 2004 Workshop Proceedings*, Seattle, Washington, USA, 26-30 June 2004.

[129] Michael O'Neill, Erik Hemberg, Conor Gilligan, Eliott Bartley, James McDermott, and Anthony Brabazon. GEVA: Grammatical evolution in java. *SIGEVOlution*, 3(2), Summer 2008.

[130] Michael O'Neill, James McDermott, John Mark Swafford, Jonathan, Erik Hemberg, Anthony Brabazon, Elizabeth Shotton, Ciaran McNally, and Martin Hemberg. Evolutionary design using grammatical evolution and shape grammars: Designing a shelter. *International Journal of Design Engineering*, 3(1):4–24, 2010.

[131] Michael O'Neill, Miguel Nicolau, and Anthony Brabazon. Dynamic environments can speed up evolution with genetic programming. In Natalio Krasnogor, Pier Luca Lanzi, Andries Engelbrecht, David Pelta, Carlos Gershenson, Giovanni Squillero, Alex Freitas, Marylyn Ritchie, Mike Preuss, Christian Gagne, Yew Soon Ong, Guenther Raidl, Marcus Gallager, Jose Lozano, Carlos Coello-Coello, Dario Landa Silva, Nikolaus Hansen, Silja Meyer-Nieberg, Jim Smith, Gus Eiben, Ester Bernado-Mansilla, Will Browne, Lee Spector, Tina Yu, Jeff Clune, Greg Hornby, Man-Leung Wong, Pierre Collet, Steve Gustafson, Jean-Paul Watson, Moshe Sipper, Simon Poulding, Gabriela Ochoa, Marc Schoenauer, Carsten Witt, and Anne Auger, editors, *GECCO '11: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, pages 191–192, Dublin, Ireland, 12-16 July 2011. ACM.

[132] Michael O'Neill and Conor Ryan. Automatic generation of caching algorithms. *Evolutionary Algorithms in Engineering and Computer Science*, 30:127–134, 1999.

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

[134] Michael O'Neill, Conor Ryan, Maarten Keijzer, and Mike Cattolico. Crossover in grammatical evolution: The search continues. In J. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. G. B. Tetamanzi, and W. B. Langdon, editors, *Proceedings of the Fourth European Conference on Genetic Programming (EuroGP-2001)*, volume 2038 of *LNCS*, pages 337–347, Lake Como, Italy, 2001. Springer Verlag.

[135] Michael O'Neill, Conor Ryan, Maarten Keijzer, and Mike Cattolico. Crossover in grammatical evolution. *Genetic Programming and Evolvable Machines*, 4(1):67–93, March 2003.

[136] Michael O'Neill, Conor Ryan, and Miguel Nicolau. Grammar defined introns: An investigation into grammars, introns, and bias in grammatical evolution. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 97–103, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

[137] Alfonso Ortega, Abdellatif Abu Dalhoum, and Manuel Alfonseca. Grammatical evolution to design fractal curves with a given dimension. *IBM Journal of Research and Development*, 47(4):483–493, 2003.

[138] Diego Perez, Miguel Nicolau, Michael O'Neill, and Anthony Brabazon. Evolving Behaviour Trees for the Mario AI Competition Using Grammatical Evolution. In Cecilia Di Chio, Stefano Cagnoni, Carlos Cotta, Marc Ebner, Aniko Ekart, Anna I Esparcia-Alcazar, Juan J. Merelo, Ferrante Neri, Mike Preuss, Hendrik Richter, Julian Togelius, and Georgios N. Yannakakis, editors, *Applications of Evolutionary Computing, EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoIN-*

TELLIGENCE, EvoNUM, EvoSTOC, volume 6624 of *LNCS*, pages 123–132, Turin, Italy, 27-29 April 2011. Springer Verlag.

[139] Diego Perez, Miguel Nicolau, Michael O'Neill, and Anthony Brabazon. Reactiveness and Navigation in Computer Games: Different Needs, Different Approaches. In *Proceedings of the 2011 IEEE Conference on Computational Intelligence and Games*, pages 273–280, Seoul, South Korea, 31 August - 3 September 2011. IEEE.

[140] Riccardo Poli, William B. Langdon, and Nicholas Freitag 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).

[141] John Reddin, James McDermott, and Michael O'Neill. Elevated pitch: Automated grammatical evolution of short compositions. In Mario Giacobini, Ivanoe De Falco, and Marc Ebner, editors, *Applications of Evolutionary Computing, EvoWorkshops2009: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoPhD, EvoSTOC, EvoTRANSLOG*, volume 5484 of *LNCS*, pages 579–584, Tubingen, Germany, 15-17 April 2009. Springer Verlag.

[142] Franz Rothlauf. On the locality of representations. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1608–1609, Chicago, 12-16 July 2003. Springer-Verlag.

[143] Franz Rothlauf. *Representations for genetic and evolutionary algorithms*. Springer, second edition, 2006. First published 2002, 2nd edition available electronically.

[144] Franz Rothlauf and David E. Goldberg. Redundant representations in evolutionary computation, January 28 2003.

[145] Franz Rothlauf and Marie Oetzel. On the locality of grammatical evolution. In Pierre Collet, Marco Tomassini, Marc Ebner, Steven Gustafson, and Anikó Ekárt, editors, *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 320–330, Budapest, Hungary, 10 - 12 April 2006. Springer.

[146] Conor Ryan, Atif Azad, Alan Sheahan, and Michael O'Neill. No coercion and no prohibition, A position independent encoding scheme for evolutionary algorithms—the Chorus system. In James A. Foster, Evelyne Lutton, Julian Miller, Conor Ryan, and Andrea G. B. Tettamanzi, editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 131–141, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag.

[147] Conor Ryan and R. Muhammad Atif Azad. Sensible initialisation in chorus. In Conor Ryan, Terence Soule, Maarten Keijzer, Edward Tsang, Riccardo Poli, and Ernesto Costa, editors, *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 394–403, Essex, 14-16 April 2003. Springer-Verlag.

[148] Conor Ryan and Miguel Nicolau. Doing genetic algorithms the genetic programming way. In Rick L. Riolo and Bill Worzel, editors, *Genetic Programming Theory and Practice*, chapter 12, pages 189–204. Kluwer, 2003.

[149] Conor Ryan, Miguel Nicolau, and Michael O'Neill. Genetic algorithms using grammatical evolution. In James A. Foster, Evelyne Lutton, Julian Miller, Conor Ryan, and Andrea G. B. Tettamanzi, editors, *Genetic Programming, Proceedings of the 5th*

European Conference, EuroGP 2002, volume 2278 of *LNCS*, pages 278–287, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag.

[150] Conor Ryan and Michael O'Neill. Grammatical evolution: A steady state approach. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 180–185, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Stanford University Bookstore.

[151] Sevil Sen and John Andrew Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks. In *WiSec '09: Proceedings of the second ACM conference on Wireless network security*, pages 95–102, Zurich, Switzerland, March 16-19 2009. ACM.

[152] Mark Shackleton, Rob Shipman, and Marc Ebner. An investigation of redundant genotype-phenotype mappings and their role in evolutionary search. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 493–500, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 July 2000. IEEE Press.

[153] Noor Shaker, Miguel Nicolau, Georgios N Yannakakis, Julian Togelius, and Michael O'Neill. Evolving levels for super mario bros using grammatical evolution. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 304–311. IEEE, 2012.

[154] Noor Shaker, Georgios N Yannakakis, Julian Togelius, Miguel Nicolau, and Michael ONeill. Evolving personalized content for super mario bros using grammatical evolution. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012.

[155] Jianhua Shao, James McDermott, Michael O'Neill, and Anthony Brabazon. Jive: a generative, interactive, virtual, evolutionary music system. In Cecilia Di Chio,

Anthony Brabazon, Gianni A. Di Caro, Marc Ebner, Muddassar Farooq, Andreas Fink, Jorn Grahl, Gary Greenfield, Penousal Machado, Michael O'Neill, Ernesto Tarantino, and Neil Urquhart, editors, *EvoMUSART*, volume 6025 of *LNCS*, pages 341–350, Istanbul, 7-9 April 2010. Springer.

[156] Rob Shipman, Mark Shackleton, Marc Ebner, and Richard Watson. Neutral search spaces for artificial evolution: A lesson from life. *Artificial Life*, 7:162–169, 2000.

[157] 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.

[158] John Swafford, Miguel Nicolau, Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Comparing methods for module identification in grammatical evolution. In Terry Soule, Anne Auger, Jason Moore, David Pelta, Christine Solnon, Mike Preuss, Alan Dorin, Yew-Soon Ong, Christian Blum, Dario Landa Silva, Frank Neumann, Tina Yu, Aniko Ekart, Will Browne, Tim Kovacs, Man-Leung Wong, Clara Pizzuti, Jon Rowe, Tobias Friedrich, Giovanni Squillero, Nicolas Bredeche, Stephen Smith, Alison Motsinger-Reif, Jose Lozano, Martin Pelikan, Silja Meyer-Nienberg, Christian Igel, Greg Hornby, Rene Doursat, Steve Gustafson, Gustavo Olague, Shin Yoo, John Clark, Gabriela Ochoa, Gisele Pappa, Fernando Lobo, Daniel Tauritz, Jurgen Branke, and Kalyanmoy Deb, editors, *GECCO '12: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, pages 823–830, Philadelphia, Pennsylvania, USA, 7-11 July 2012. ACM.

[159] John Mark Swafford. *Analyzing the Discovery and Use of Modules in Grammatical Evolution.* PhD thesis, Ph.D. thesis, University College Dublin, Ireland, 2013.

[160] John Mark Swafford, Erik Hemberg, Michael O'Neill, and Anthony Brabazon. Analyzing module usage in grammatical evolution. In Carlos A. Coello Coello, Vincenzo Cutello, Kalyanmoy Deb, Stephanie Forrest, Giuseppe Nicosia, and Mario Pavone, editors, *Parallel Problem Solving from Nature, PPSN XII (part 1)*, volume 7491 of *Lecture Notes in Computer Science*, pages 347–356, Taormina, Italy, September 1-5 2012. Springer.

[161] John Mark Swafford, Erik Hemberg, Michael O'Neill, Miguel Nicolau, and Anthony Brabazon. A non-destructive grammar modification approach to modularity in grammatical evolution. In Natalio Krasnogor, Pier Luca Lanzi, Andries Engelbrecht, David Pelta, Carlos Gershenson, Giovanni Squillero, Alex Freitas, Marylyn Ritchie, Mike Preuss, Christian Gagne, Yew Soon Ong, Guenther Raidl, Marcus Gallager, Jose Lozano, Carlos Coello-Coello, Dario Landa Silva, Nikolaus Hansen, Silja Meyer-Nieberg, Jim Smith, Gus Eiben, Ester Bernado-Mansilla, Will Browne, Lee Spector, Tina Yu, Jeff Clune, Greg Hornby, Man-Leung Wong, Pierre Collet, Steve Gustafson, Jean-Paul Watson, Moshe Sipper, Simon Poulding, Gabriela Ochoa, Marc Schoenauer, Carsten Witt, and Anne Auger, editors, *GECCO '11: Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 1411–1418, Dublin, Ireland, 12-16 July 2011. ACM.

[162] John Mark Swafford, Michael O'Neill, Miguel Nicolau, and Anthony Brabazon. Exploring grammatical modification with modules in grammatical evolution. In Sara Silva, James A. Foster, Miguel Nicolau, Mario Giacobini, and Penousal Machado, editors, *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, volume 6621 of *LNCS*, pages 310–321, Turin, Italy, 27-29 April 2011. Springer Verlag.

[163] Jorge Tavares and Francisco B. Pereira. Automatic design of ant algorithms with

grammatical evolution. In Alberto Moraglio, Sara Silva, Krzysztof Krawiec, Penousal Machado, and Carlos Cotta, editors, *Proceedings of the 15th European Conference on Genetic Programming, EuroGP 2012*, volume 7244 of *LNCS*, pages 206–217, Malaga, Spain, 11-13 April 2012. Springer Verlag.

[164] Sebastian Ventura, Cristobal Romero, Amelia Zafra, Jose A. Delgado, and Cesar Hervas. JCLEC: a java framework for evolutionary computation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 12(4):381–392, 2008.

[165] David R. White. Software review: the ECJ toolkit. *Genetic Programming and Evolvable Machines*, 13(1):65–67, March 2012.

[166] David R White, James McDermott, Mauro Castelli, Luca Manzoni, Brian W Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una-May O'Reilly, and Sean Luke. Better GP Benchmarks: Community Survey Results and Proposals. *Genetic Programming and Evolvable Machines*, 14(1):3–29, 2013.

[167] Dominic Wilson. *Grammatical Evolution based Data Mining for Network Intrusion Detection*. PhD thesis, Electrical Engineering and Computer Science, University of Toledo, Toledo, OH, USA, April 2008.

Part IV

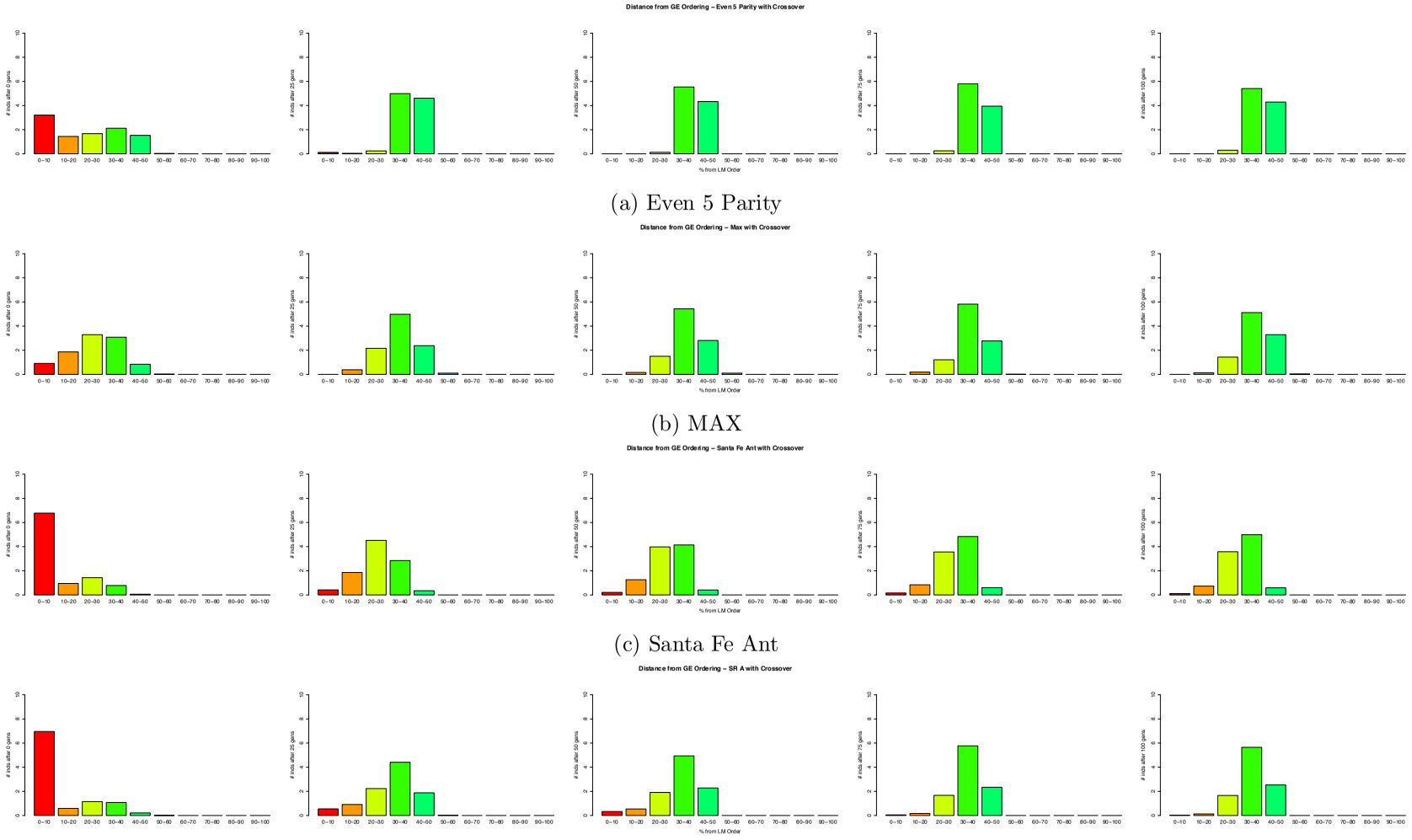
Appendices

Appendix A

Order Histograms - All Runs

This appendix contains the order histogram figures for Chapter 5. The figures show the top 10 individuals for each run. The figures mirror those shown in the Section 5.4 just from the perspective of the top individuals from the runs.

A.1 Order Histograms - Elite Individuals



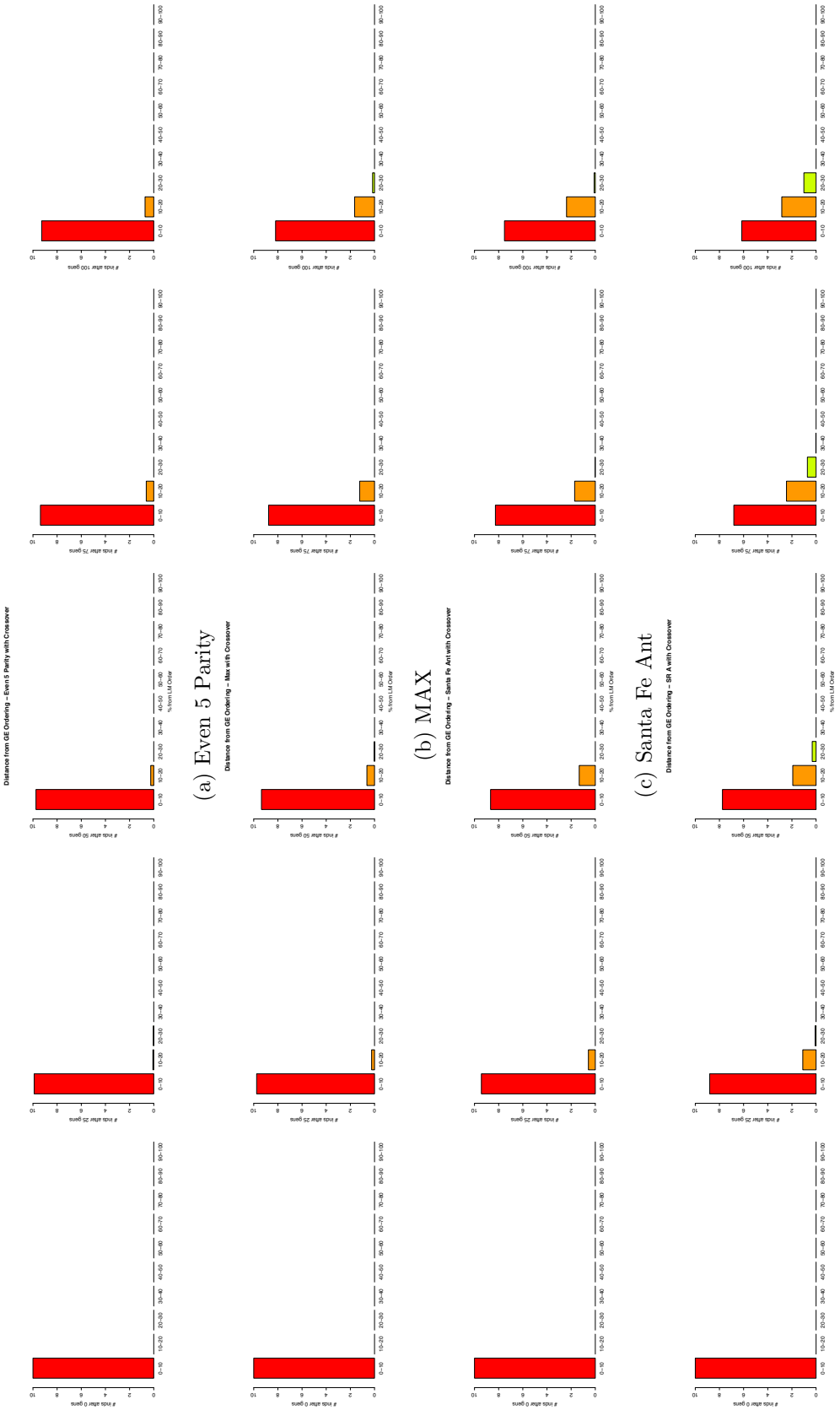
(a) Even 5 Parity

(b) MAX

(c) Santa Fe Ant

(d) Symbolic Regression A

Figure A.1: The figure shows the distribution of distances from a depth-first order on four problems with Random Order Initialisation, for the top 10 individuals in the population. The subfigures show how the distribution changes during the course of the run with the left-most graph being generation zero, and the right-most being the final generation. In the four graphs it is evident that πGE evolves towards a distribution of orders centre around 30% - 40%.



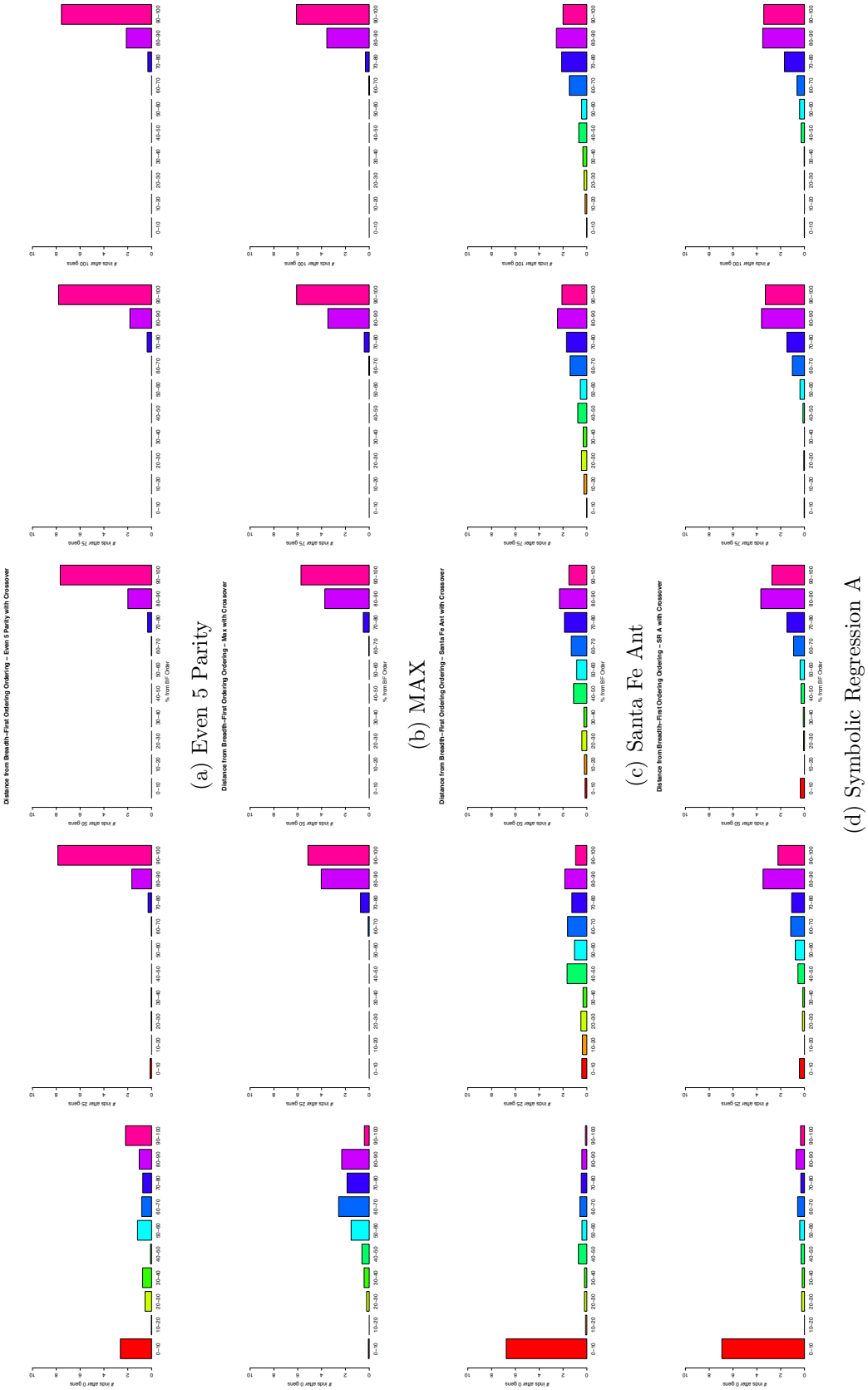
(a) Even 5 Parity

(b) MAX

(c) Santa Fe Ant

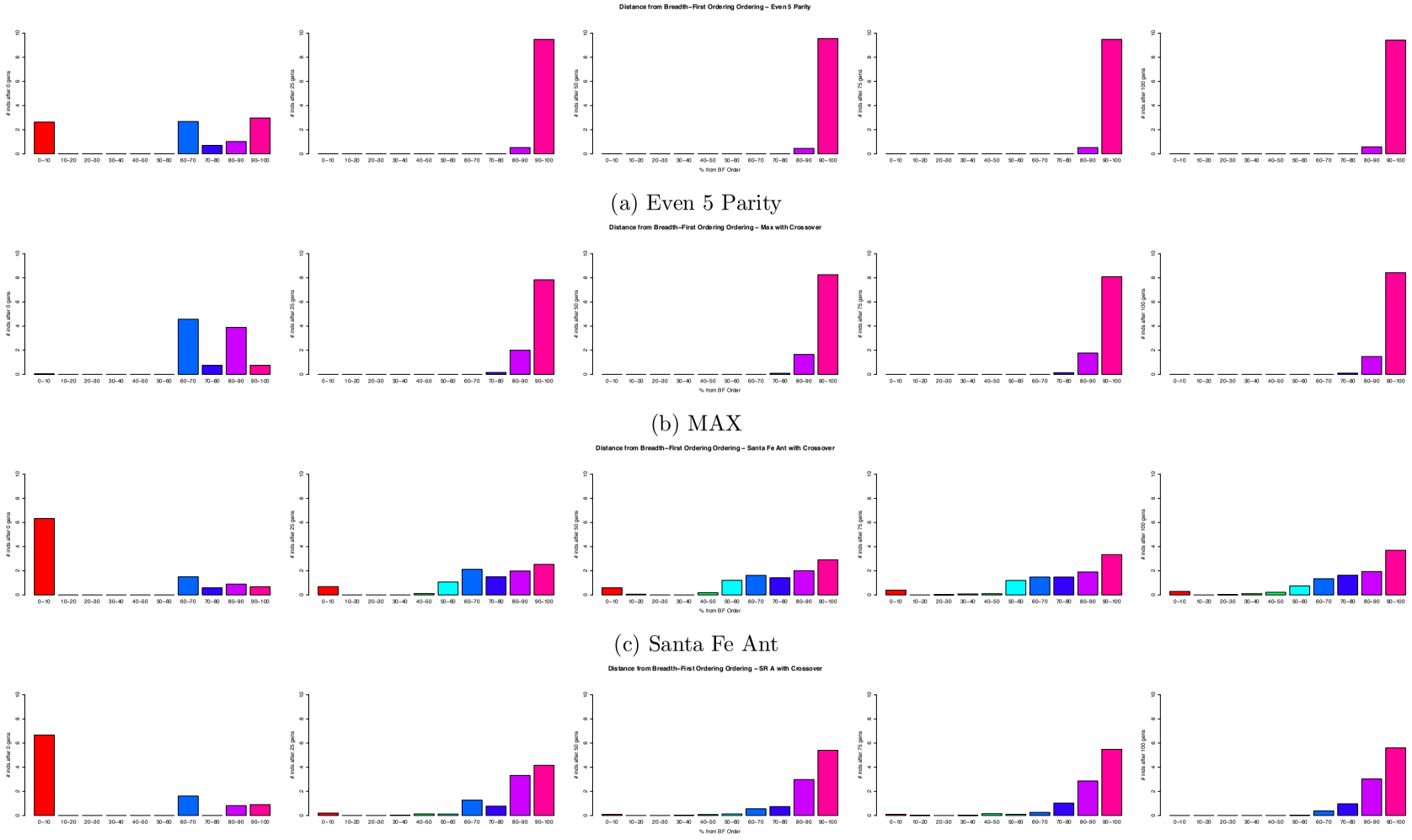
(d) Symbolic Regression A

Figure A.2: The figure shows the distribution of distances from a depth-first order on four problems with GE Order Initialisation, for the top 10 individuals in the population. The subfigures show how the distribution changes during the course of the run with the left-most graph being generation zero, and the right-most being the final generation. In the four graphs it is evident that πGE evolves away from the fixed order at varying rates, but there is a strong presence of GE style orders present in the populations.



(a) Even 5 Parity (b) MAX (c) Santa Fe Ant (d) Symbolic Regression A

Figure A.3: The figure shows the distribution of distances from a breadth-first order on four problems using random order initialisation, for the top 10 individuals in the population. The figures show that the population starts with a wide distribution of values. For Even 5 Parity and Max the population is in a much tighter distribution at the end of the runs, that is focus at 100%. Santa Fe Ant and Symbolic Regression A, however, have a much wider tail to their distributions at the end of the runs. The figure indicates that there is no indication of a significant breadth-first order being present.



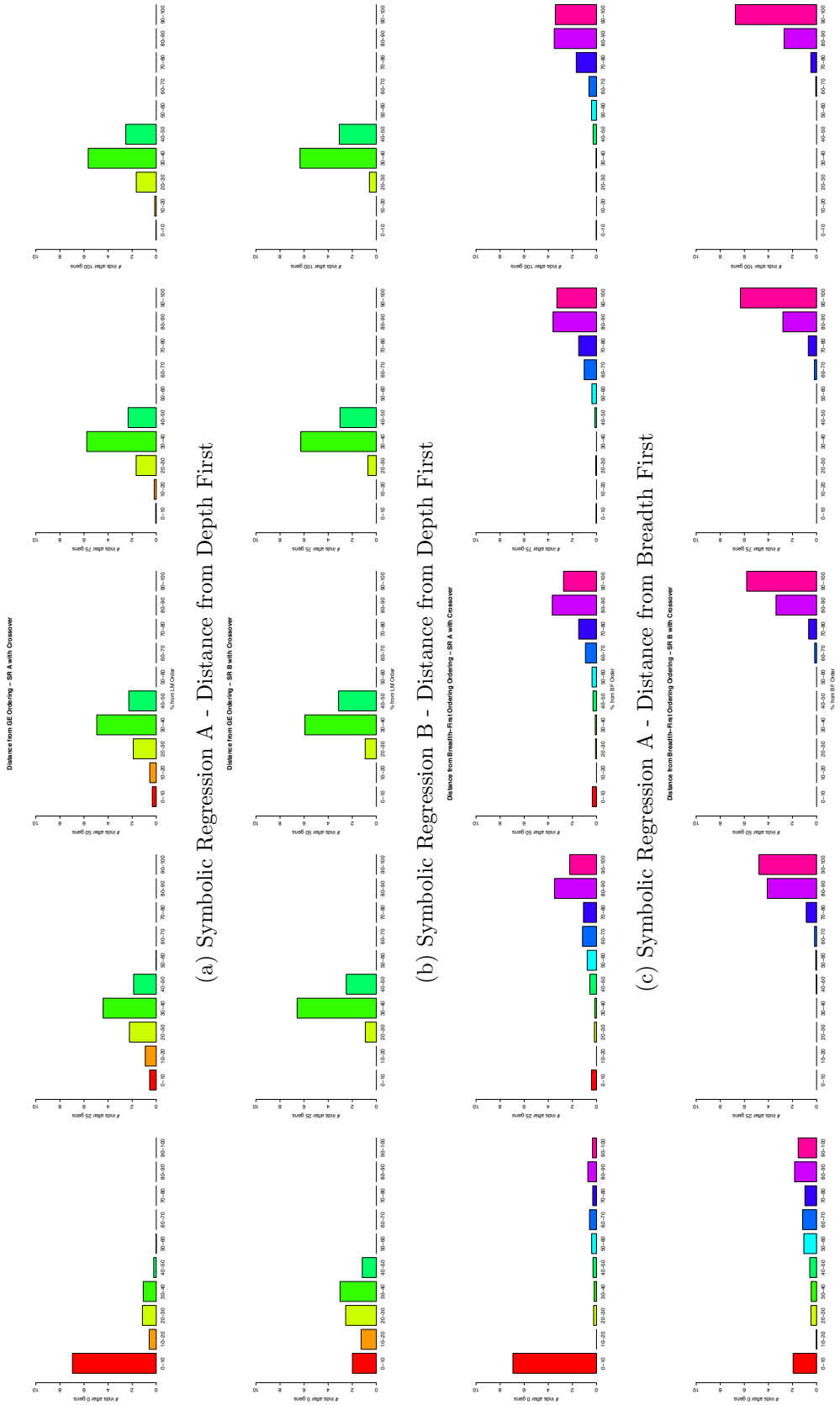
(a) Even 5 Parity

(b) MAX

(c) Santa Fe Ant

(d) Symbolic Regression A

Figure A.4: The figure shows the distribution of distances from a breadth-first order on four problems using GE order initialisation, for the top 10 individuals in the population.The figures show that GE order initialisation actual pushes the final distribution further to 100%, than was seen in Figure 5.7. Also the generation 0 distribution has a different shape that was seen with random order initialisation.



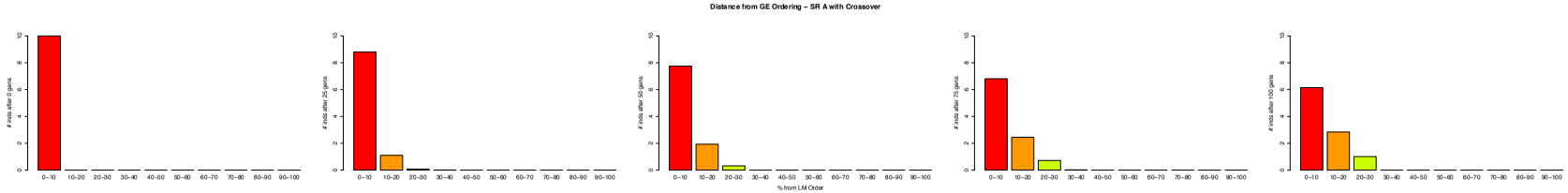
(a) Symbolic Regression A - Distance from Depth First

(b) Symbolic Regression B - Distance from Depth First

(c) Symbolic Regression A - Distance from Breadth First

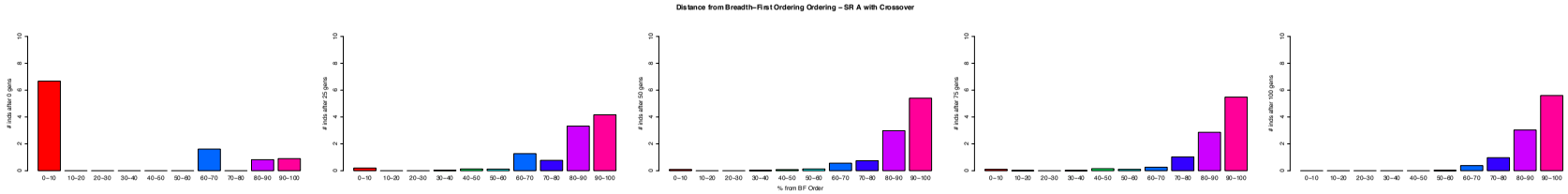
(d) Symbolic Regression B - Distance from Breadth First

Figure A.5: The figure shows the difference a change in grammar induces on the orders in πGE using a random order initialisation, for the top 10 individuals in the population. The reduction in grammar complexity has lead to a reduction in the tail size of the distributions shown.

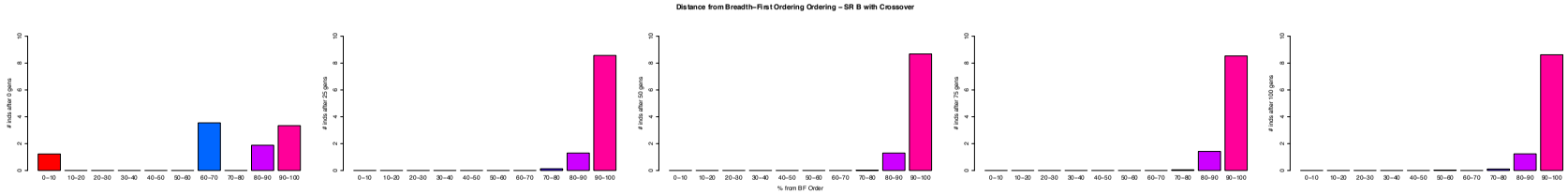


(a) Symbolic Regression A - Distance from Depth First

(b) Symbolic Regression B - Distance from Depth First

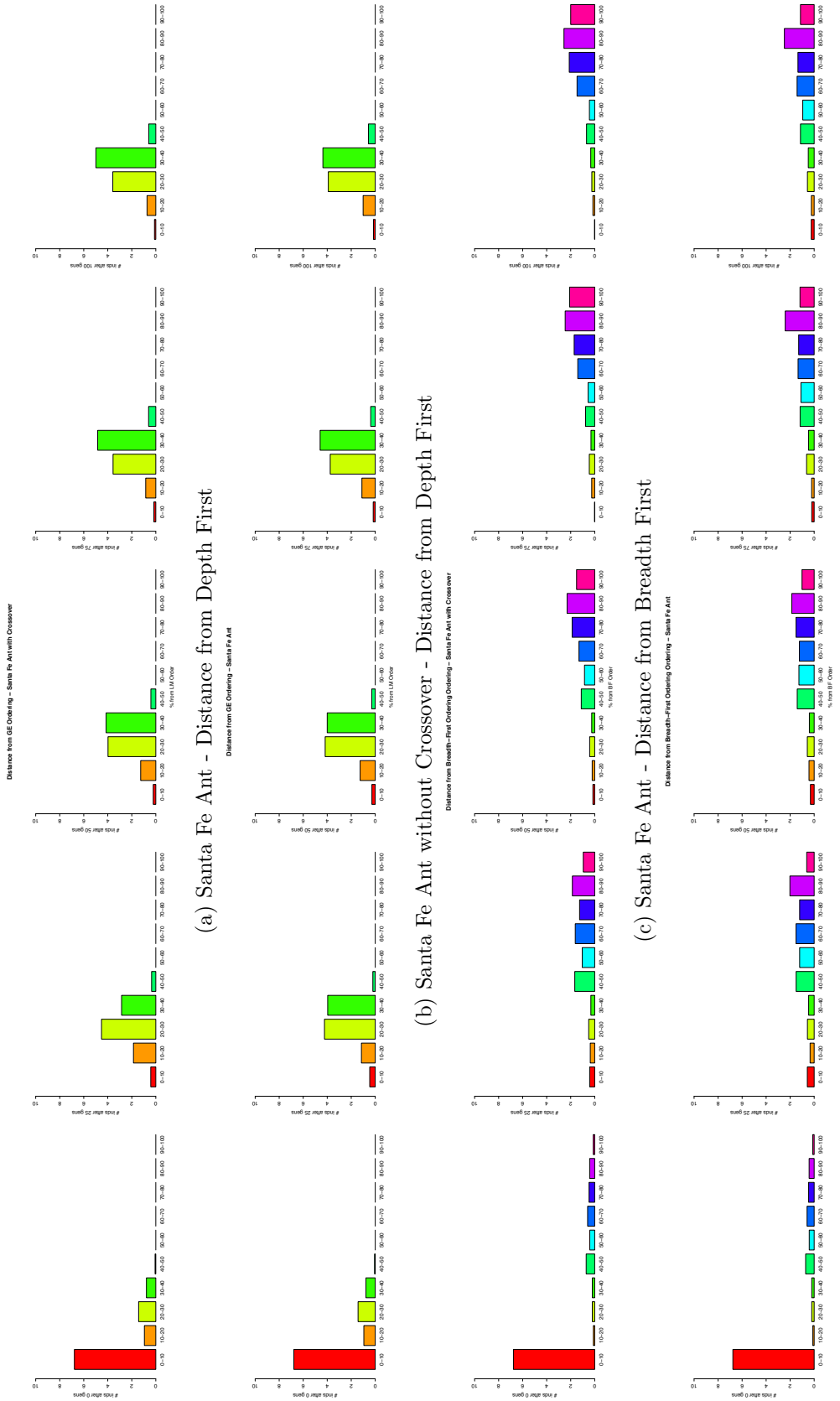


(c) Symbolic Regression A - Distance from Breadth First



(d) Symbolic Regression B - Distance from Breadth First

Figure A.6: The figure shows the difference a change in grammar induces on the orders in πGE using GE order initialisation, for the top 10 individuals in the population. The reduction in grammar complexity has lead to a reduction in the tail size of the distributions shown as in Figure 5.11. An interesting observation is how the reduction in grammar complexity has also retarded the drift away from a GE order.



(a) Santa Fe Ant - Distance from Depth First

(b) Santa Fe Ant without Crossover - Distance from Depth First

(c) Santa Fe Ant - Distance from Breadth First

(d) Santa Fe Ant without Crossover - Distance from Breadth First

Figure A.7: The figure shows the effect crossover has on the orders in πGE on the Santa Fe Ant problem, for the top 10 individuals in the population. In both DFOB and BFOB crossover can be said to push the distribution further away from a known order in both cases.

(a) Santa Fe Ant - Distance from Depth First

(b) Santa Fe Ant without Crossover - Distance from Depth First

(c) Santa Fe Ant - Distance from Breadth First

(d) Santa Fe Ant without Crossover - Distance from Breadth First

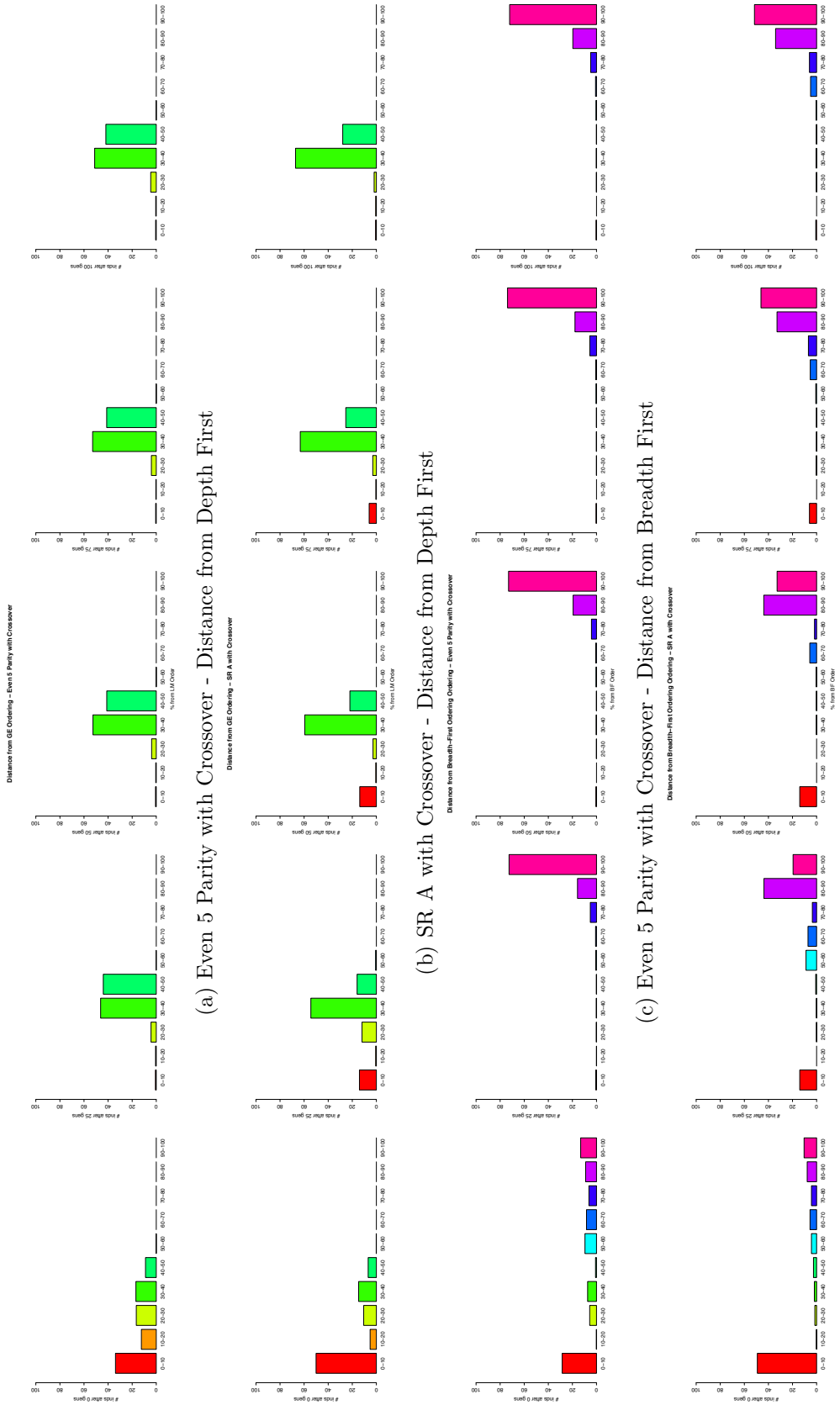
Figure A.8: The figure shows the effect crossover has on the orders in πGE on the Santa Fe Ant problem using GE order initialisation, for the top 10 individuals in the population. As was seen in Figure 5.12, there is only a slight variation in order with the addition of crossover.

Appendix B

Order Histograms - Successful Runs

This appendix contains the order histogram figures for Chapter 5. The figures show the orders observed for only the successful runs. The figures when compared to those in Section 5.4 display the same observed trends.

B.1 Order Histograms - Successful Runs Only



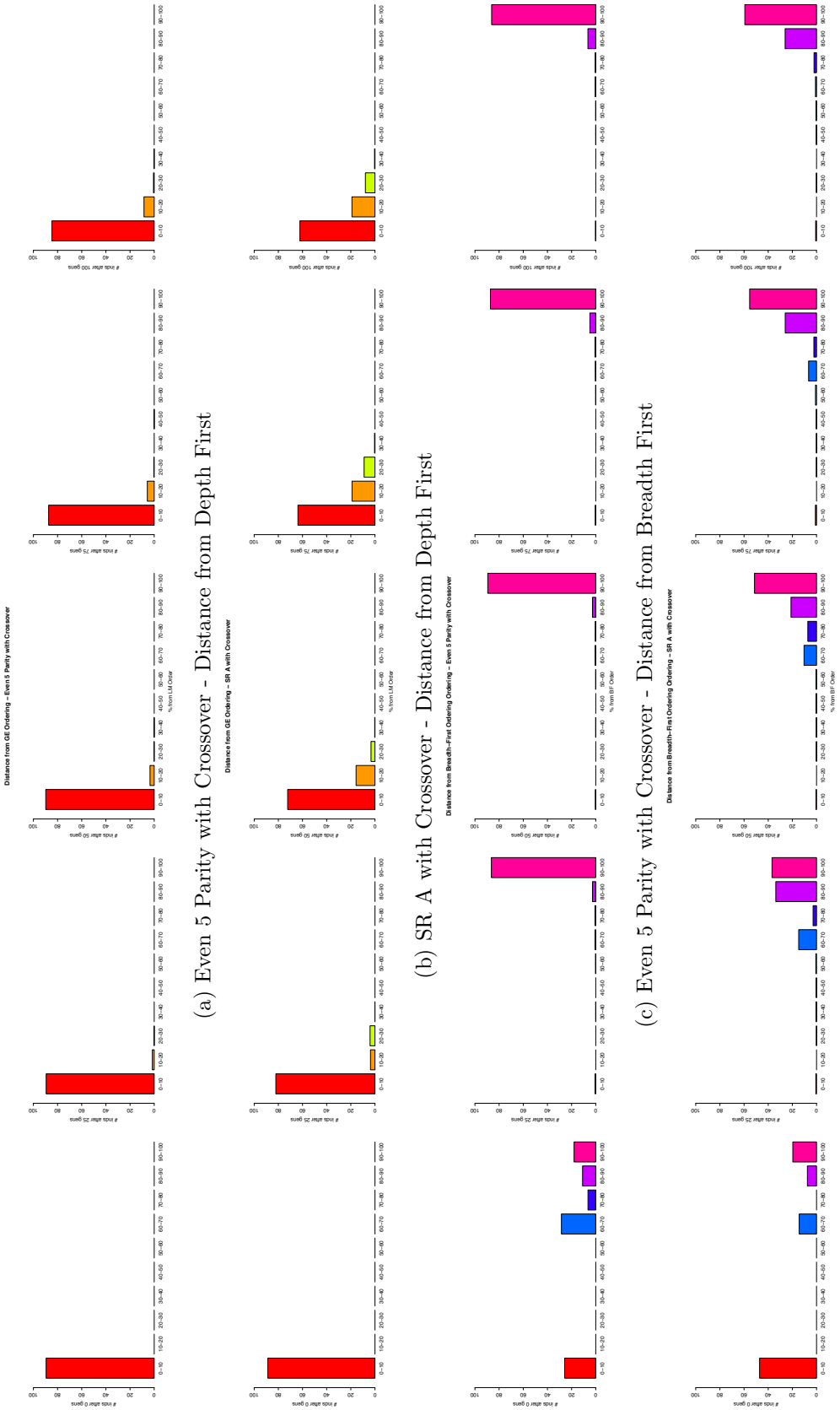
(a) Even 5 Parity with Crossover - Distance from Depth First

(b) SR A with Crossover - Distance from Depth First

(c) Even 5 Parity with Crossover - Distance from Breadth First

(d) SR A with Crossover - Distance from Breadth First

Figure B.1: The figure shows the distribution of distances from a depth-first and breadth-first order on two problems with Random Order Initialisation, for the successful runs. The subfigures show how the distribution changes during the course of the run with the left-most graph being generation zero, and the right-most being the final generation.



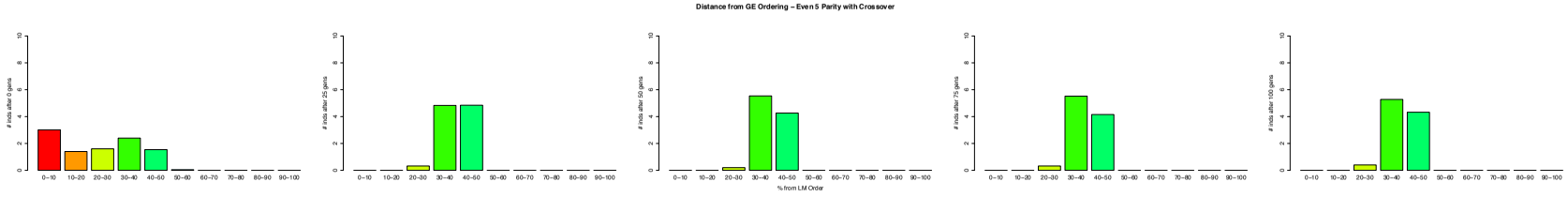
(a) Even 5 Parity with Crossover - Distance from Depth First

(b) SR A with Crossover - Distance from Depth First

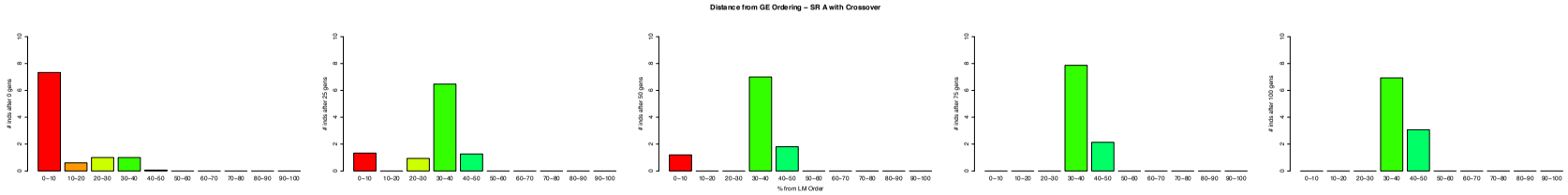
(c) Even 5 Parity with Crossover - Distance from Breadth First

(d) SR A with Crossover - Distance from Breadth First

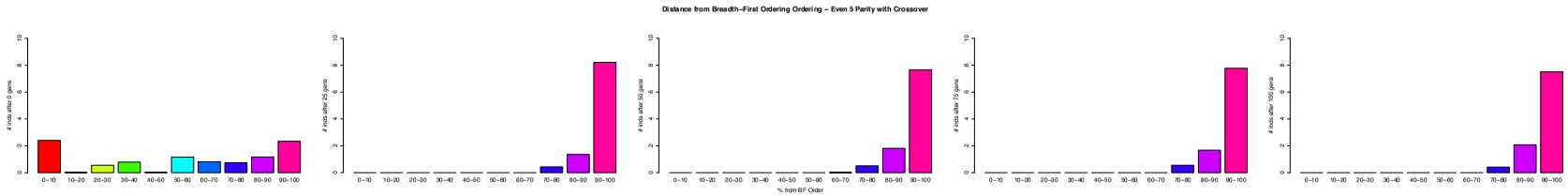
Figure B.2: The figure shows the distribution of distances from a depth-first and breadth-first order on two problems with GE Order Initialisation, for the successful runs. The subfigures show how the distribution changes during the course of the run with the left-most graph being generation zero, and the right-most being the final generation.



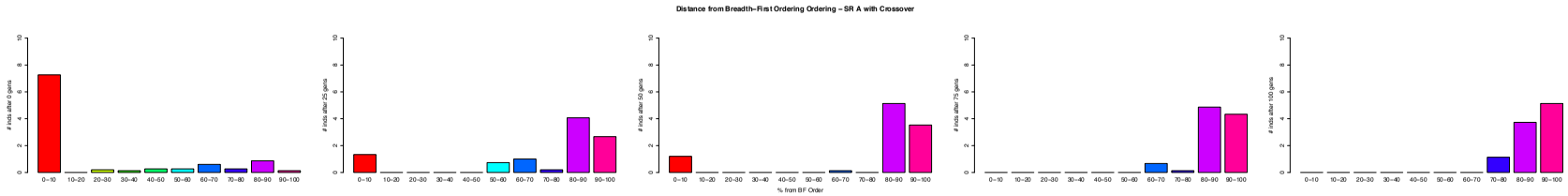
(a) Even 5 Parity with Crossover - Distance from Depth First



(b) SR A with Crossover - Distance from Depth First

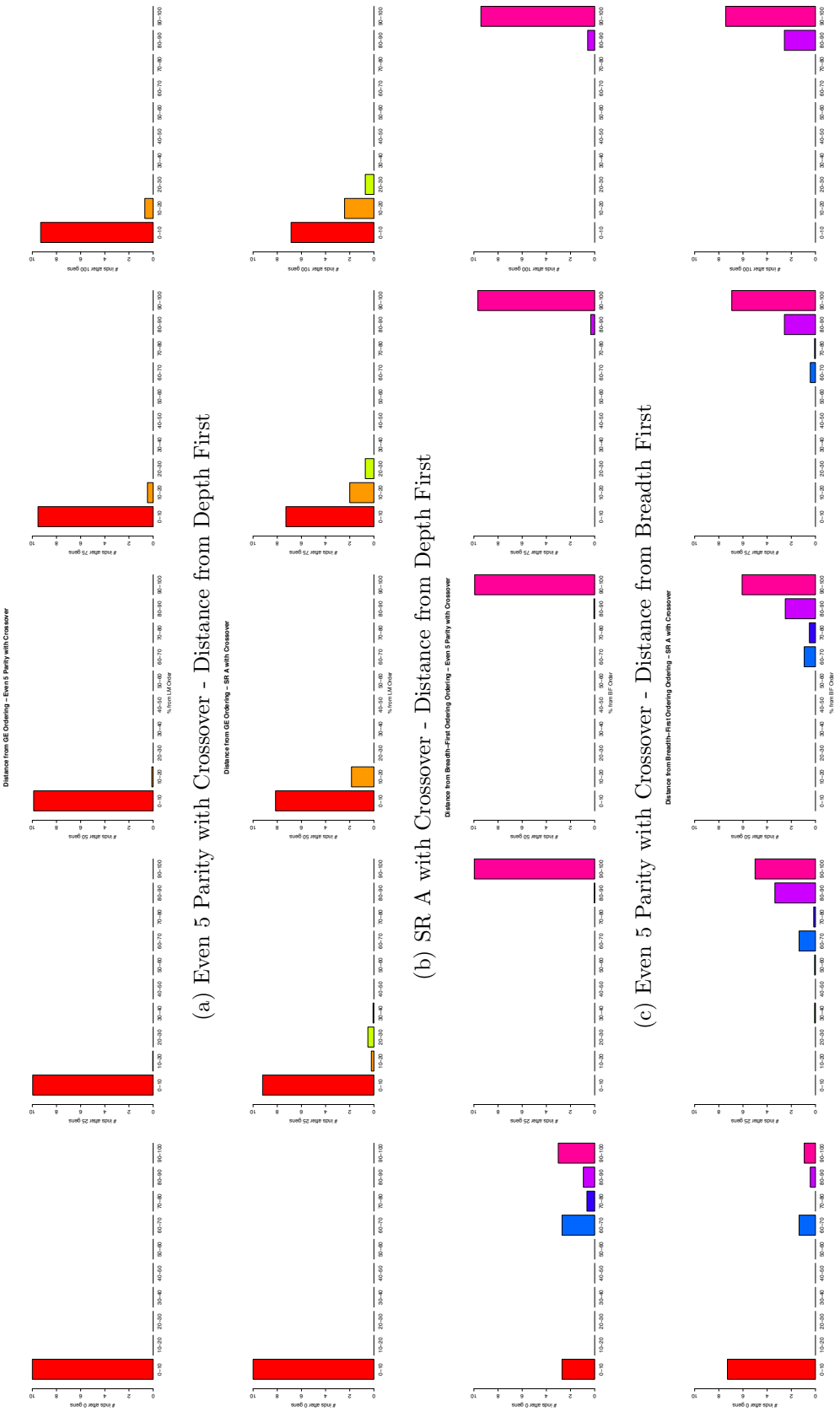


(c) Even 5 Parity with Crossover - Distance from Breadth First



(d) SR A with Crossover - Distance from Breadth First

Figure B.3: The figure shows the distribution of distances from a depth-first and breadth-first order on two problems with Random Order Initialisation, for top 10 of the population on the successful runs. The subfigures show how the distribution changes during the course of the run with the left-most graph being generation zero, and the right-most being the final generation.



(a) Even 5 Parity with Crossover - Distance from Depth First

(b) SR A with Crossover - Distance from Depth First

(c) Even 5 Parity with Crossover - Distance from Breadth First

(d) SR A with Crossover - Distance from Breadth First

Figure B.4: The figure shows the distribution of distances from a depth-first and breadth-first order on two problems with GE Order Initialisation, for top 10 of the population on the successful runs. The subfigures show how the distribution changes during the course of the run with the left-most graph being generation zero, and the right-most being the final generation.