Locality in the Evolutionary Optimisation of Programs

by

Thomas A. Seaton

PhD Thesis

UNIVERSITY OF YORK

Department of Electronics

February 2013

Abstract

The development and optimisation of programs through search is a growing application area for computational intelligence techniques. Evolution-inspired search heuristics, such as genetic programming, provide methods for autonomously generating programs within the constraints of a program representation. Genetic programming is a machine learning approach to producing programs represented using executable or interpreted structures. However, despite theoretical advances, choosing a suitable representation remains a basic concern for designers. Choice of representation affects search space size, structure and accessible solutions, as well as engineering considerations such as ease of implementation.

Locality is a property of evolutionary search spaces derived from the representation and search operators, that relates genotype and phenotype distances. The interaction between search space locality and search performance under different representations is not well understood. The objective of this thesis is to broaden the present understanding of locality to encompass more complex representations, for example graphs and grammars, as well as non-traditional coevolutionary approaches.

This thesis presents four main original contributions. Firstly, a statistical approach to measuring locality is defined that incorporates the Mantel test, a method adapted from numerical ecology. The method is assessed empirically in a series of case studies over two established forms of genetic programming, Grammatical Evolution and Cartesian Genetic Programming. Secondly, a new approach to visualising locality is provided. The technique uses force-layout algorithms derived from the field of graph-drawing to construct fitness landscapes in genetic programming. The technique is applied to produce visualisations that demonstrate structural characteristics across regions of the search space. Thirdly, the effect of locality on performance is assessed in model coevolutionary problems. A framework to analyse performance in a coevolutionary context is provided, followed by an examination of the response to locality and coupled algorithm parameters. The final contribution explores the interaction between locality and two 'pathological' dynamics in coevolutionary algorithms, disengagement and cycling. The analysis demonstrates that locality can influence the likelihood of coevolutionary pathologies, when using executable representations. Results are provided for new constructed problems and a coevolutionary pursuit and evasion task.

In the conclusions, directions for future analysis of the role of locality in evolutionary search are considered, as well as the relationship between these findings and other outstanding general issues in the field of genetic programming.

Contents

List of Figures

List of Tables

List of Symbols

	Distances and Metrics
d_X	A metric on the space X (Definition 3.1)
d_X^+	The maximum distance in the space X under the metric d.
d_X^-	The minimum distance in the space X under the metric d.
$\bar{M}$	The expected variation distance (Definition 3.3).
	Evolutionary Algorithm Composition and Search Space
R	Representation (Definition 2.1).
G	Genotype space (Definition 2.1).
P	Phenotype space (Definition 2.2).
m	Genotype to phenotype map (Definition 2.2).
f	Fitness function (Definition 2.3).
s	Search operator (Definition 2.4).
F	Function set.
T	Terminal set.
n	Genotype length.
q	Mutation probability.
	Genotype to Phenotype Map Properties
L	Locality (Definition 2.6)
Υ	Redundancy (Definition 2.8).
L_N	Normalised locality (Definition 6.3).
L_N^{-1}	Inverse locality (Definition 6.4).
	Neighbourhood Functions
$adj(x)$	Adjacent neighbours of x (Section 2.7.1).
$adj_m(x)$	Adjacent neighbours of x, transformed under m (Section 2.7.1).
$adj_m^{-1}(x)$	Adjacent neighbours of x, transformed under the inverse of m (Section 2.7.1).

Statistical

r_M	Mantel correlation coefficient (Section 3.4.1).
$G(\mu, \sigma)$	Gaussian distribution.

Graph Theory

V	Set of Vertices.
E	Set of Edges.
H	Hypergraph (Definition 4.1).
$\mathcal{L}$	Fitness Landscape (Definition 4.2).
K^x	A complete graph with x vertices.

Coevolutionary Algorithm

u	Coevolutionary interaction function (Definition 5.1).
C	Set of coevolutionary candidates.
T	Set of coevolutionary tests.

Problem and Operator Specific Parameters

μ	Number of selected parents in an evolutionary strategy.
λ	Number of offspring produced in an evolutionary strategy.
α	Mutation bias slope parameter (Section 3.6.2).
β	Mutation bias offset parameter (Section 3.6.2).
ν	Mutation bias for the Greater-Than game (Section 6.4.2).
θ	Angular heading of pursuer trajectory in the Game of Tag (Section 7.3.4).
γ	Angular heading of line of sight between pursuer and evader (Section 7.3.4).
N	Proportional navigation constant (Section 7.3.4).

Acknowledgements

Throughout my PhD, there have been a number of people who played a role and to whom I owe considerable thanks. My supervisor, Dr. Julian F. Miller, has been a source of great support and inspiration. Thank you Julian, for your guidance and willingness to spare so much time and energy even under sometimes difficult circumstances. I am also indebted to Tim Clarke, for stepping in during my second year. Tim has provided a wealth of advice and many enjoyable discussions.

That year I was fortunate enough to spend time at the University of Mainz. I am very grateful to Prof. Franz Rothlauf and his team for the excellent opportunity and useful conversations. Thanks especially to Dr. Jella Pfeiffer, Wolfgang Steitz and Malte Probst for making me feel very welcome in Germany.

Here in York, I would like to thank members/former members of the Intelligent Systems Group and York Centre for Complex Systems Analysis. In particular, Dr. Simon Hickinbotham, Simon Poulding, Dr. Martin Trefzer, Dr. David White and Dr. Dan Franks have all provided valuable input. Specifically thanks to Simon H., Simon P. and Martin for looking through my work and posing some very good questions, to Dave for his many insights into GP and detailed reviews, to Dan for our discussions on coevolution and the Mantel statistic. To my fellow lab mates, Mungo, Alex, Luis and Alfonso, for making the lab an enjoyable place to work with a fun atmosphere, as well as for talking through those inevitable day to day coding and maths problems that crop up. Thanks also to my internal examiner Prof. Susan Stepney and external examiner Dr. Michael O'Neill, University College Dublin, for agreeing to review my research and providing a thorough but enjoyable critique of this thesis.

Finally, but by no means least, a tremendous thanks to all my friends and family. Mum, Dad, Pip, your love and support has provided me with the strength to complete this. Pauline, John and the Owens, you have given me a second home 'up North'. Laura, this is for you - thank you for being with me, every step of the way.

For Laura

Declaration of Authorship

I declare that this thesis titled, 'Locality in the Evolutionary Optimisation of Programs' and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a research degree at the University of York.
- Where I have consulted published work, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

Signed: Thomas A. Seaton 

Date: 18.02.2013

"In the physical universe the effect that one event has on another tends to decrease with the distance in time or in space between them... Locality of action shows itself in the finite speed of light, in the inverse square law of fields, and in macroscopic statistical effects, such as rates of reaction and the speed of sound.

In computation, or at least in our old models of computation, an arbitrarily small event can and often does cause an arbitrarily large effect. A tiny program can clear all of memory. A single instruction can stop the machine. In computation there is no analog of distance. One memory location is as easily influenced as another."

The Connection Machine, Hillis (1985)

Part I

Motivation and Foundations

Chapter 1

Introduction

Contents

1.1 Evolutionary Search in Programs

The term *heuristic* is used in engineering to describe a guideline or rule of thumb, derived from prior experience. *Randomised search heuristics* (RSH) refer to heuristic approaches to finding solutions that are at least partly non-deterministic. Useful randomised search heuristics may not be guaranteed to converge on an optimal solution, but are expected to perform well in practice on certain classes of problems. Evolutionary Computation (EC) concerns the study of RSH that are inspired by biological evolution to solve computational problems. Numerous alternative classes of randomised search heuristic exist, for example particle-swarm optimisation or simulated annealing, whose inspiration lies in the observation of other natural search processes.

An application within Evolutionary Computation that has attracted much attention is the optimisation or development of programs through search (Friedberg, 1958; Fogel *et al.*, 1966; Smith, 1980; Forsyth, 1981; Cramer, 1985; Koza, 1992). The use of EC

approaches is motivated by the requirement to efficiently explore the large and complex spaces typifying this domain. The field of Genetic Programming (GP) encompasses a specialised collection of EC techniques designed for this task (Banzhaf *et al.*, 1998; Langdon and Poli, 2002; Poli *et al.*, 2008). The approach can be considered to be a form of automatic programming. GP techniques have been used to address a diverse range of problems, including notable human competitive successes (Koza *et al.*, 2003).

Despite the established applications of GP, theoretical progress explaining the performance of the technique on particular classes of problem has been more limited (Poli *et al.*, 2010). A major contributing factor is uncertainty over the influence and role of program representation in evolutionary search. In natural evolving systems, a mapping - the *genotype to phenotype map* (GPM) - exists between the biological structures storing inherited or genetic information and the expressed traits of an organism. The structure of GPMs and the breadth of changes required to exhibit new characteristics is a basic concern in genetics and molecular biology. Analogously, the representation of a program and the mapping between its stored form and executable behaviour also plays a fundamental role in the performance of GP heuristics.

1.2 Program Representations and Locality

Designing an Evolutionary Algorithm (EA) to develop or optimise programs requires the choice of an appropriate representation and associated mapping. Within this context, a broad range of options are available to the practitioner. These vary from representations based on tree data structures (the original formulation of GP) to configurations that use grammars, graphs or stacks. At present, there are few theoretical guidelines available for GP to support the selection of a particular configuration. Typically, the choices of users are motivated by empirical evidence, for example prior successes on similar problems, or practical considerations, for example ease of implementation. However, a range of qualitative properties of the artificial GPMs used in program evolution have been identified that have parallels with the GPM properties of natural systems. Similarities between artificial and natural GPM may include many to many relationships between represented programs and their characteristics, a modular composition, redundant components with no explicit functionality and a disproportional response to change. Establishing the significance of these characteristics can inform the decision between different representations and search mechanisms when artificial evolution is applied to a new problem.

Biological systems are remarkably resilient to mutations in genetic structure, but also have the capacity to express large changes under small variations. Many human genetic

disorders such as Haemophilia, Cystic Fibrosis and Sickle-cell disease are now known to be closely tied to local variations in individual and well isolated genes (Antonarakis and Beckmann, 2006). Conceptually, this is a familiar phenomenon for any user of modern programming languages; under the wrong conditions, variation of a single instruction can drastically alter the functionality of a program. In artificial GPM, the term *locality* has been coined to describe a property that addresses the extent to which small variations in the evolving structure (*local* changes) correspond to small variations in the expressed traits of that structure (Rothlauf, 2006). For the context of a heuristic search on programs, locality describes whether neighbouring configurations under the search are likely to have similar functional properties.

In recent years, it has been suggested that measures to preserve locality in artificial GPM can provide performance gains in GP (Nguyen, 2011; Krawiec, 2011). These studies have primarily focused on standard tree representations. However, the role of locality in other classes of program representation designed by practitioners, for example more general graph or grammar based structures, is largely not understood. A representation independent view of locality for GP would therefore be highly desirable. This will require the construction of novel analytical tools that are designed for a range of GP techniques and which can subsequently be applied to the more complex methods currently in use.

1.3 Coevolutionary Systems

One emerging area of application for such tools is coevolutionary search. Evolution in nature does not occur in isolation. *Coevolution* refers to the mutual influence of species upon one another's survival. The presence of one or more species affects the evolution of another, which in turn affects the first (Janzen, 1980). Artificial evolutionary systems have long sought to exploit coevolution, which enables the application of EC heuristics to problem domains where the quality of evolving members are defined through their interaction with others. *Coevolutionary algorithms* (CEA) have significantly different properties when compared with evolutionary algorithms that have access to an objective measure of solution quality. Designers have historically had to take into account these distinctions, which include pathological modes of behaviour not observed in standard EAs, that result in distinct forms of search (Ficici and Pollack, 1998; Watson and Pollack, 2001; Bucci and Pollack, 2003; Wiegand and Sarma, 2004).

Coevolutionary principles have been incorporated into GP techniques to provide methods of coevolutionary program development (Lichodzijewski and Heywood, 2007; Tay *et al.*, 2008; Schmidt and Lipson, 2008; McIntyre and Heywood, 2008; Sikulova and Sekanina, 2012). An advantage of this class of algorithm is their application to cooperative

or competitive scenarios, where programs are required to share resources, control conflicting agents or otherwise interact. As in traditional approaches, the requirement to define a suitable representation during the design process remains. However, no dedicated studies exist exploring locality in coevolutionary forms of GP. From the evidence in traditional evolutionary artificial systems, it is reasonable to propose that it is a contributing factor. Conversely, it is also possible that the effect of GPM properties may differ from those observed in standard GP systems. There is therefore a strong motivation to extend current analysis to include these methods.

1.4 Research Goals

The overarching aim of this research is to examine the effects of GPM locality in artificial evolutionary systems capable of developing programs. The intention is to broaden the present understanding of locality in GP by addressing systems using non-traditional program representations for EC, such as graphs and linear representations. The scope of the following work will encompass methods including both evolutionary and coevolutionary search mechanisms, with the objective of establishing a more unified framework that incorporates the distinct properties of these approaches.

1.5 Research Questions and Hypotheses

To address this aim, I pose a series of four research questions that will be considered sequentially. Questions A and B are by formulation engineering issues. Nonetheless, they are necessary steps in the following analysis, against which demonstrable progress can be shown. Questions C and D can be framed as falsifiable hypotheses.

A: "How can a statistical measure of genotype to phenotype locality be developed for different genetic programming representations?"

Locality is a difficult property to characterise with a range of previously stated definitions (Rothlauf, 2006; Chiam *et al.*, 2008; Galvan-Lopez *et al.*, 2011a). Quantifiable measurements of properties of the GPM are inherently challenging because of the size of EC search spaces and ensuing statistical uncertainty under sampling. Designing a statistically rigorous approach that can encompass non-traditional program representations carries additional complexity; the process of mapping to semantic behaviour may be structurally very distinct between different methods of storing programs. To extract useful general principles it is desirable to abstract methods of measurement away from single forms of representation. The value of developing a general method for quantifying

locality across different representations is that the property can be compared over the variety of GPM that are used by GP practitioners and evaluated experimentally.

B: "How can locality be visualised over regions of the GPM?"

Qualitative techniques for examining GPM locality are a necessary complement to the quantitative statistical approaches implied by Question A. Genotype to phenotype maps are not generally homogeneous and regions of the search space can exhibit distinct connectivity and neighbourhood structure. Visualising this structure in program representations requires suitable methods of multi-dimensional scaling. The goal of this question is to consider techniques to examine locality directly, with the intention of providing a more intuitive understanding of the property. As in A, the approaches developed should be applicable to different representations.

C: "Does genotype to phenotype locality contribute to coevolutionary algorithm performance?"

The significance of locality in evolutionary algorithm performance has been established empirically (D'haseleer and Bluming, 1994; Raidl and Gottlieb, 2005; Nguyen, 2011). However, as will be highlighted in this thesis, coevolutionary systems make several fundamentally different assumptions to traditional EAs. It is therefore not sufficient to assume that GPM locality will have the same effect on search that has been observed in standard techniques. The aim of this question is to verify whether coevolutionary algorithms exhibit similar responses to changes in this property of the GPM and identify contributing factors that may be unique to these systems.

D: "Does semantic locality interact with the occurrence of pathological behaviours in program coevolution?"

Algorithms using coevolutionary techniques introduce behaviours that lead to game theoretic states and interactions not seen in non-coevolutionary EAs. An example is cycling, where an algorithm becomes 'trapped' in a sequence of intransitive configurations. Dedicated studies have been carried out to investigate and mitigate these behaviours in coevolutionary algorithms using traditional bit string representations (Cartlidge, 2004; Wijngaarden and Jong, 2008). The factors that influence their occurrence in systems using more complex representations are not well understood. This question proposes that GPM locality is coupled to the occurrence of pathological behaviours in CEA. It is posed to open the unexplored issue of whether, in genetic programming, relationships can be drawn between coevolutionary pathologies and general properties of the GPM.

1.6 Thesis Structure

The thesis is divided into three parts. Part I contains background material on Evolutionary Computation, GP and the necessary foundations for the rest of the thesis. Part II addresses research questions A and B, developing statistical and visualisation tools that form an element of the main thesis contributions. Part III then tests hypotheses C and D. The focus in the second part is on standard evolutionary forms of GP. The third part introduces the more general subject of coevolutionary systems to extend the analysis developed in the first two parts.

- Chapter 2 (**Genetic Programming and the Genotype to Phenotype Map**) defines key terms, reviewing the literature concerning program representations in Evolutionary Computation and properties of genotype to phenotype maps.
- Chapter 3 (**Measuring Locality in GP Representations**) builds on the quantitative definitions in Chapter 2, proposing a statistical approach. The chapter explores the use of the Mantel test, a technique drawn from numerical ecology.
- Chapter 4 (**Methods of Visualising Locality**) develops a graph-drawing method to display variations in locality across program search spaces. The chapter reviews existing approaches, then describes the technique and provides examples of its application under different GP representations.
- Chapter 5 (**Coevolution and Representations**) reviews the literature on coevolutionary algorithms, focusing on the fundamental distinctions between EA and CEA. The chapter then addresses how the framework described in Chapter 2 applies to these systems.
- Chapter 6 (**Locality in Test-based Coevolution**) describes an empirical methodology for coevolutionary algorithms and presents experiments in bit string representations. The experiments examine how locality is coupled with algorithm parameters and its affect on performance over selected benchmarks.
- Chapter 7 (**Locality in Coevolutionary Genetic Programming**) extends the work in Chapter 6 by considering locality in coevolutionary forms of GP. Experiments are provided that investigate the relationship between locality and pathological coevolutionary behaviours in GP systems.
- Chapter 8 (**Conclusions**) closes the thesis by re-examining the findings of the previous chapters in relation to the proposed research questions. The remaining sections then detail potential future directions and relate the work back to more general open issues in the field of GP.

1.7 Major Related Work and Themes

There exist several general studies concerning locality and properties of artificial genotype to phenotype maps in GPM closely related to this thesis. Within the field of Evolutionary Computation, Rothlauf (2006) gives an introduction and overview of GPM properties that forms the basis of the frameworks and methodology considered in Chapters 2, 3 and 4. Nguyen (2011) and Galvan-Lopez *et al.* (2011a) have conducted empirical work defining and analysing locality in genetic programming based on tree data structures, which these chapters are also intended to complement. The framework for coevolutionary algorithms reviewed in Chapter 5 is based on the principles described by Ficici (2004) and Popovici *et al.* (2012). The concept of systematically analysing pathological behaviours in coevolution follows a line of inquiry emerging from the studies of Watson and Pollack (2001), which provides the starting point for Chapters 6 and 7.

Two further themes are explored in addition to the role of locality in evolutionary algorithms. These include: benchmarking of coevolutionary systems and biological parallels. In Chapters 6 and Chapter 7, we consider issues in measuring progress and designing benchmarks for coevolutionary algorithms. Finally, Chapter 8 also briefly addresses how the findings in Chapter 7 compare to observations of coevolution in natural systems.

1.8 Publications

Some of the material incorporated in this thesis has been presented in previous publications, primarily through conference proceedings. Use of the Mantel statistic as a measure of program locality in Chapter 3 was published in Seaton *et al.* (2012a). The visualisation method that is established in Chapter 4 was demonstrated in Seaton *et al.* (2012b). Experiments given in Chapter 6 are an expanded version of those provided in Seaton *et al.* (2011). The results in Chapter 7 have been reviewed and accepted for publication in Seaton *et al.* (2013).

Chapter 2

Genetic Programming and the Genotype to Phenotype Map

Contents

2.1 Chapter Motivation

Combinatorial optimisation is the problem of minimising or maximising a target function $f(x)$, where $\{x_1, x_2...x_n\}$ are elements in a set of feasible combinatorial solutions X. Classical and highly successful approaches exist for many examples of combinatorial optimisation problems. The methods of mathematical programming can be applied to determine the local maximum or minimum of a convex target function, subject to constraints. However, in the more general global combinatorial optimisation situation the objective is to find the globally optimal solution set, allowing that the target function may be non-convex and possibly multi-modal. Global optimisation problems may therefore be hard and not necessarily solvable in polynomial time (given $P \neq NP$). This motivates the use of modern heuristic approaches such as Evolutionary Computation.

A significant body of Evolutionary Computation (EC) methods exist that have been engineered to directly search a space of programs. The first motivation of this chapter is to provide necessary background on EC, outlining the most prevalent search heuristics and their application to the optimisation of programs. A short overview of the field of genetic programming and its precursors will be provided. The emphasis of the review is on distinguishing between the different methods of representing programs used in evolutionary search. The second motivation is to define common EC terms, which enables a more formal discussion of locality and the properties of the genotype to phenotype map (GPM) referenced throughout this thesis.

2.2 Chapter Outline

In Section 2.3, we consider the context of EC in optimisation, making reference to the No Free Lunch theorems. Section 2.4 then briefly summarises the principal areas within EC and their historical differences. Section 2.5 contrasts established techniques from four of the main paradigms within genetic programming which use tree, graph, grammar and linearly structured representations. Basic definitions are given in Section 2.6. The concept of locality and the genotype to phenotype map is introduced in Section 2.7, alongside a brief qualitative description of representational characteristics such as redundancy, robustness and modularity. Section 2.8 considers the current perspectives on GP locality and presents conclusions.

2.3 Evolutionary Computation as Stochastic Search

EC techniques can be seen as part of a wider collection of iterative random processes that non-deterministically select elements from X to evaluate under the target function. As such, the performance of EC techniques is stochastic, which permits solutions to hard combinatorial optimisation problems to be found in less than exponential time, at the expense of guaranteed convergence. The probability distribution used to select new elements may be implicitly defined by some selection mechanism or explicitly specified. A fundamental distinction between these methods is the procedure through which the implicit or explicit probability distribution over X is determined at each timestep.

2.3.1 No Free Lunch Theorems

The No-Free Lunch (NFL) theorems are a seminal set of results in statistics and machine learning that have been applied to stochastic search (Wolpert, 1996; Wolpert and Macready, 1996). Informally, within the context of optimisation, the no free lunch theorems imply that expected performance of an optimisation algorithm over all possible function choices is equivalent. The performance of a given algorithm for a particular problem is contingent on finding a good coupling between the selection procedure and the function to be optimised. The term 'Free Lunch' is used to describe an exception to the general No Free Lunch theorems for a specific subcase. For example, Free lunches have been shown between function representations and for selected subsets of functions (Droste and Wiesmann, 1998; Whitley, 1999). More recently, Poli and Graff (2009) demonstrated that a Free Lunch exists when searching the space of optimal heuristic algorithms; a context that can encompass Genetic Programming.

One consequence of the No Free Lunch theorems is that it is inappropriate to claim that any optimisation algorithm conforming to the original definitions set out in (Wolpert, 1996) has inherently superior performance for a random choice of function. However, in practice many randomly generated functions are of little interest. Therefore, combined with the existence of free lunches, the No Free Lunch theorems do not greatly restrict either the design of new EC optimisation algorithms or the engineering benefits of understanding which stochastic optimisation algorithms are likely to perform better on particular classes of problem.

2.4 Classes of Evolutionary Computation

The field of Evolutionary Computation has been divided into several broad sub-areas, which each encompass a common methodology or metaheuristic. As the field continues to mature it should be noted that these historically significant classes are becoming increasingly difficult to differentiate as particular techniques are hybridised or adopt common concepts and terminology.

2.4.1 Genetic Algorithms

Genetic algorithms (GA) were originally introduced by Holland (1975) as a model of natural evolutionary change and adaptation. The canonical genetic algorithm acts on a population of binary strings initialised randomly. Members of the population are assigned a *fitness value* that is used to determine the members selected as parents for subsequent generations. Operations analogous to *mutation* and *crossover* in biological systems are applied to selected strings, to produce new strings (*offspring*) introducing variation into the population. In the former, one or more bits are flipped with fixed probability. In the latter, segments of each string are exchanged between selected members of the population. Retaining fit members progressively improves the fitness of the whole population over evolutionary time.

2.4.2 Evolution Strategies

Rechenberg (1973) and Schwefel (1977) invented Evolutionary Strategies (ES) as a method of solving real vector numerical optimisation problems of the form $f : \mathbb{R}^n \to \mathbb{R}$. Evolutionary strategies define the selection mechanism by reference to a rank ordered population. A $(\mu + \lambda)$ strategy generates λ new points from μ selected parent points. Parents are selected by truncating the whole population to the best μ individuals by fitness. In a (μ, λ) strategy selection of the new parents is carried out only from the previous offspring. Offspring are produced by mutation, perturbing each point by a value drawn from the multivariate normal distribution with standard deviation ('step size') σ. The step size may be updated using a fixed rule or evolved simultaneously with the population, controlling the distance traversed by the population of vectors. More sophisticated ES adjust the direction of the distribution by updating the covariance between components (Hansen and Ostermeier, 1996).

2.4.3 Evolutionary Programming

Evolutionary Programming (EP) was proposed by Fogel *et al.* (1966) as a technique acting on finite state machines capable of predicting changes in the future state of their environment. Machines are assigned a fitness value based on the error in their production of unobserved states (symbols) given an input set of previously known states. Operations can be made to alter the structure of the finite state machine, such as the addition or subtraction of state nodes or changes to the connections between states. Parents are selected from a population of finite state machines based on their fitness. Later variations of Evolutionary Programming were abstracted from the finite state representation, though the emphasis was retained on reproduction through mutation rather than recombination (Fogel, 1994). Canonical EP is also distinct from canonical GP by including mutation as a distribution of possible operators.

2.4.4 Genetic Programming

Genetic Programming (GP) is a form of evolutionary computation popularised by Koza (1992), combining concepts from genetic algorithms and evolutionary programming. The basic premise of GP is to consider the evolution of executable program structures. The technique can therefore be considered to be a method of automatic programming. In its original formulation, genetic programming evolved a single population of tree-based programs. Since this early work GP has been extended to consider many different variations - see for examples the surveys of Poli *et al.* (2008), Kouchakpour *et al.* (2008) - amongst the most prominent of which are graph, grammar and linear systems. GP carries out a heuristic search over a space of possible programs to achieve a program which satisfies some set of objectives. An initial population of programs is generated according to a particular *initialisation scheme* and members of that population are selected according to a *fitness function.* Analogous search operators to those in GAs are then applied to selected candidates to generate new programs over successive generations, which are constructed based on the old parents. The central assumption of GP is that through the recombination of old, fit programs, progressively superior offspring will be generated. Figure 2.1 gives an example of high-level pseudocode for canonical genetic programming.

2.5 Genetic Programming: Paradigms

GP has many similarities with the older field of genetic algorithms (GA). The main distinction between genetic algorithms and genetic programming is that GP explicitly

```
1. Initialise a population of programs.
2. Evaluate fitness.
3. Select from amongst the best programs.
4. Apply search operators to selection to breed the next generation.
5. If termination criteria not reached, repeat from 2.
```

FIGURE 2.1: *Generic GP pseudocode for evolution in a single population.*

focuses on higher-level structures that can be parsed to generate executable programs. As a consequence, GP is often less tractable to theoretical analysis, because differences induced in the fitness of candidate solutions by the action of search operators are more complex. The following sections describe how program representation has been handled in GP, illustrated using examples from four standard approaches based on tree, graph, grammatical and linear representations.

2.5.1 Tree-based Genetic Programming

Tree-based genetic programming methods are the oldest class of GP program representation. Programs are constructed as syntax trees from a set of primitive functions and a set of terminals. Each interior node of the tree is associated with a function and the leaves are associated with terminals. The *arity* of each interior node is the number of arguments of the function (the number of inputs it admits). Figure 2.2 illustrates an example tree program, which is executed by recursive evaluation from leaves to root node. The initialisation method determines the structure of tree programs in the first generation of the search (initial population). In canonical Tree GP, programs are initialised using the *full*, *grow*, or *ramped half-and-half* methods. In the full method, trees are produced where all branches are the same length (equivalently, all terminals appear at the same depth). In the grow method, terminals can appear at any depth, producing trees with variable length branches. The ramped half-and-half method attempts to provide a greater diversity of structures in the initial population than either approach, by generating trees using both methods with equal probability.

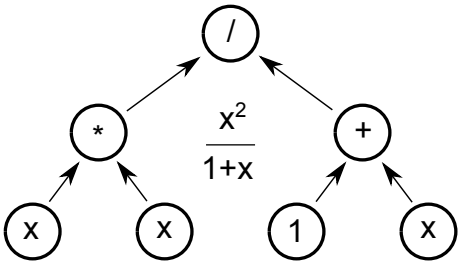

FIGURE 2.2: *An example full tree program representing a simple algebraic expression (parsed recursively from the left).*

At the end of each generation, programs are executed and a fitness function applied (*evaluation*) that maps each candidate program onto a real value, according to its performance on the particular problem. The population can then be ordered and some probabilistic selection method used based on the fitness values. *Tournament* selection methods chooses potential subsets of candidates randomly and compares their fitness to determine winners for the next generation. *Fitness proportionate* methods select candidates in proportion to their fitness compared to that of the whole population. A further, important and frequently applied concept is that of *elitism*, where one or more of the best candidates in the population are automatically selected.

A *crossover* operator, as commonly defined for Tree GP, accepts two selected parents from a previous generation and interchanges elements between parents based around a crossover point chosen in each tree. Depending on the specific strategy and location of crossover, large sub-trees (*sub-tree crossover*) or single leaves may be exchanged. The resulting *offspring* are compositions of each parent. The *mutation* operator accepts a single parent. Mutation operators may be defined to act on a single primitive function or terminal, randomly changing it (*point mutation*) or by replacing whole sub-trees with new, randomly generated variants (*sub-tree mutation*). Crossover and mutation are applied exclusively and at different rates, though the convention in canonical Tree GP is to adopt a high rate of crossover and low rate of mutation. Selected parents which undergo neither crossover nor mutation are copied directly into the next generation (*reproduction*). The process repeats for each new generation until a threshold number of generations is reached or a candidate emerges with the desired level of fitness. Typically, a large number of runs are executed, each initialising a different population of program trees and following a different trajectory through the space of possible programs.

The phenomenon of *bloat* is the tendency for the population of program trees to progressively increase in depth as the search continues without necessarily a corresponding improvement in fitness. Numerous strategies have been proposed in Tree GP to combat bloat; see the surveys of Luke and Panait (2006); Silva and Costa (2009). For example, a common technique is to constrain the maximum tree depth to some fixed number of nodes. Program trees that exceed this depth are then eliminated from the search.

2.5.2 Graph-based Genetic Programming

The concept of extending Tree GP to a graph based representation was proposed by Poli (1996) and Miller *et al.* (1997) through the *Parallel Distributed Genetic Programming* (PDGP) and *Cartesian Genetic Programming* (CGP) frameworks. In both frameworks, the constraint restricting function nodes to connections between distinct sub-branches

is relaxed. Edges can connect to nodes on other branches, enabling reuse of previously defined structures.

Parallel Distributed Genetic Programming (PDGP) defines graphs using a link set which directly specifies the connections between all function nodes. Each node in the graph is located on a uniform 2D grid. Connections are constrained to be directed upwards on the grid, in feed-forward fashion and between alternate rows. However, all nodes on the grid are associated with functions. This introduces the important concept of *redundancy* (see Section 2.7.2), where parts of the individual are not expressed in the resulting code. The search operators applied to PDGP include crossover and mutation, as in classical Tree GP. However, in order to preserve syntax under the new representation, several different variations were implemented. These acted on subgraphs within the grid and distinguished between 'active' and 'inactive' nodes.

Prior to PDGP, the PADO or Parallel Algorithm and Discovery Orchestration framework was proposed by Teller and Veloso (1996) which was later classified as a graph-based framework using an indexed memory. PADO was applied to the task of object recognition. More recent examples of graph-based frameworks include Genetic Network Programming (GNP) (Mabu *et al.*, 2007) for the control of agents, which was hybridised using a reinforcement learning technique, and GRAPE (Graph Structured Program Evolution) (Shirakawa and Nagao, 2009). GRAPE focused on providing features such as loops and handling of multiple data types.

Cartesian Genetic Programming (CGP) also represents programs as directed graphs, but uses a compact indirect mapping onto an integer string (Miller, 2011). As in PDGP, connections between nodes are labelled and defined with reference to rows and columns. Functions may only connect to previous nodes on the same row, or to terminals. The integer array representation that encodes this uses an integer representing the function associated with each node and a set of integers corresponding to the connected inputs. The canonical CGP approach uses a $\mu + \lambda$ truncation style of selection and applies a mutation operator directly to the integer array. A list of nodes determines where the outputs are selected from. An example CGP array representing the tree given in Figure 2.2 is shown below, where output is taken from the right most node (Figure 2.3).

The connectivity of the CGP array is specified by a set of constants - the number of rows n_r, number of columns n_c and 'levels-back' parameter l. In the single row or linear version of CGP, n_r is set to 1 and n_c to a total number of nodes (controlling the maximum possible graph depth). The levels-back parameter constrains the maximum number of previous rows which can be used as valid input connections for a node. In the unconstrained feed-forward case, each node can freely connect to any previous column. The action of the point mutation operator in CGP is to exchange one integer value

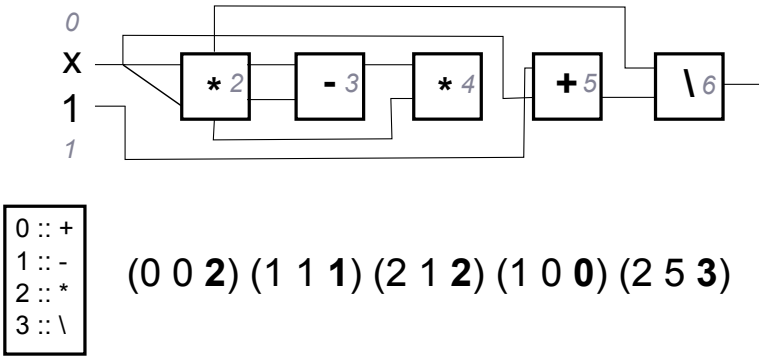

FIGURE 2.3: *Example Cartesian GP program that maps to the expression* $\frac{x^2}{1+x}$*, encoding a single row directed acyclic graph. Nodes 3 and 4 are redundant. Bold numbers in the array indicate function choices.*

in the array for an integer representing any other valid connection or function at a single point on the string. The *uniform mutation* operator carries out point mutation on each element of the array with a fixed (small) mutation probability. Notably CGP incorporates a large number of redundant 'junk' or neutral coding elements in the form of unused connections and functions. Studies of neutrality in CGP have suggested the property may be advantageous on certain classes of problem (Vassilev, 2000; Yu and Miller, 2006; Miller and Smith, 2006).

2.5.3 Genetic Programming using Grammars

A key issue in evolving executable programs is preserving the syntactic and semantic constraints implied by the programming language and problem domain. Grammar-based GP approaches (McKay *et al.*, 2010) use grammars to define the constraints during program evolution. As in Tree GP and graph-based representations, programs can be generated directly, or indirectly. Amongst the earliest examples of directly generating programs subject to grammatical constraints is the approach of Whigham (1995). Parse Trees were generated directly with reference to the grammar's production rules, derived by applying the rules repeatedly until a maximum depth was reached. Crossover and mutation operators were then applied to the stored trees. Whigham constrained these operators to ensure syntactic compatibility, such that crossover between parse trees was only permitted between nodes sharing the same non-terminal symbol.

By contrast, indirect approaches map the *decisions* required to evolve a parse tree. In Grammatical Evolution a context-free grammar is defined in Backus-Naur Form (BNF) that defines the possible relationships between each non-terminal and terminal symbol in the evolved code (O'Neill and Ryan, 2003). The syntax of evolved programs conforms to the grammar. An example of a constrained BNF grammar that could give rise to the expression shown in Figure 2.2 using postfix notation is given in Figure 2.4

```
<Tree>     ::= <Terminal> | <Function> ( <Tree>,<Tree>)

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

<Terminal> ::= x | 1
```

FIGURE 2.4: *An example BNF grammar capable of representing bi-arity trees using arithmetic operators.*

This mapping is carried out by storing a sequence of values that provide information on which decision should be taken as each non-terminal in the grammar is visited. Programs are therefore represented as a sequence of integers. To generate the program, production rules are visited in turn beginning with the start symbol for the grammar. When a choice between symbols in encountered, the next value in the sequence is looked up. The modulus of that value indicates which subsequent symbol is visited in the grammar. Conventional search operators can then be used to modify the decision sequence.[1] Grammatical evolution generates programs deterministically using a variable number of decisions, depending on the content of the decision sequence. If the end of a decision sequence is reached, material is reused by 'wrapping around' to the start. An upper limit is typically imposed to prevent too many cycles through the decision sequence, where individuals that exceed the limit are discarded. The integer sequence in grammatical evolution may be transcribed as a binary string, where each subset of transcribed bits is referred to as a codon. This transcription can take place using a natural or Gray encoding. Table 2.1 shows an example of how the left hand branch of the tree in Figure 2.2 can be mapped using this process.

String	[0101]	[1011]	[1000]	[0110]	[1010]	[0100]
Codon	5	11	8	6	10	4
Mapping	5%2=1	11%4=3	8%2=0	6%2=0	10%2=0	4%2=0
Decision	Function	×	Terminal	x	Terminal	x

TABLE 2.1: *A GE configuration corresponding to a component of the tree program in Figure 2.2. Substrings are mapped to an integer that determines the selection when a production rule is visited. The production rule then generates the corresponding symbol.*

2.5.4 Linear Genetic Programming

Encoding programs as trees, graphs or grammars requires some degree of interpretation on current processors. Reducing the interpretation required can provide gains in execution time. Linear Genetic Programming (Brameier and Banzhaf, 2007) acts at a

[1]Stochastic variants on grammar GP representations exist, where information stored in the decision sequence defines a probability that a particular symbol is selected. In the present work we will only consider deterministic forms of GE encodings.

lower level of abstraction, providing sequences of instructions to manipulate data within registers or memory. These may include branching instructions which enables the representation of graph or tree like flows of execution. Linear GP can be used to generate machine code directly or produce byte code for interpreted execution (Nordin *et al.*, 1999a,b).

Constructing programs at this level requires several departures from the Koza tree-based approach. Mutation operators are applied to introduce variation into elements of individual instructions (for example the opcode or input/output registers). Crossover operators exchange sets of instructions within the sequences comprising pairs of programs. Similarly to the higher level interpreted graph-based representations discussed in Section 2.5.2, Linear GP also includes unused instructions that constitute redundant material.

2.5.5 Other GP Variations

Another notable paradigm within GP includes systems designed such that the representation enables programs to store some or all of the information required to generate new programs. The identifying component of such a system is the capability of modifying its own mechanism for introducing variation; that is the genetic operators are at least partially constructed through evolutionary processes. To permit this requires a sufficiently flexible representation to describe programs that not only reproduce and alter their own structure, but also the facility for introducing these changes. Spector and Robinson (2002) termed these variations *autoconstructive* and provided an experimental system *Pushpop*. Pushpop uses a representation which enables the construction of complex evolutionary operators by the composition of primitive instructions within its own Lisp-like programming language, Push. The core of Push is a set of stacks to store different data types, which can be acted on by operators that push or pop their arguments to the correct stack. Other stack-based representations for GP have been constructed (Perkis, 1994).

2.5.6 Summary

This section has briefly described four of the canonical methods of representing programs in GP. Each of these forms of GP is associated with a detailed set of heuristics for searching the program space: a set of search operators, standard parameters such as population size or maximum program length and methods of evaluating each generation of output programs. To survey these comprehensively is outside the scope of this work; instead this section has focused on highlighting the principal differences between the

method of program representation used by each class. Tree-based genetic programming stores programs in the form of parse trees generated from a fixed set of function and terminal elements. Programs are executed by evaluating trees recursively. Graph-based forms of GP apply a similar approach that enables the flow of execution to pass between sub-branches. Grammar-based GP applies a set of semantic constraints to the search space based on a chosen grammar. Linear GP generates programs at a lower level of abstraction, as collections of linearly arrayed instructions.

2.6 Genetic Programming: Terminology

There is considerable variation in the interpretation of basic terms in common usage between the different EC communities. The analogous biological definitions are also far from standardised (Mahner and Kary, 1997). To avoid any ambiguity, this section formally defines the interpretations used throughout this thesis. We will then review some of the concepts used in the GA and GP literature to distinguish between different classes of representation and address the subject of locality.

2.6.1 Basic Definitions

Definition 2.1 (Genotype Representation)**.** A *genotype* g is a collection of primitive data. The *genotype representation* R is the type of data structure used to store the genotype. The set of all distinct genotypes that can be represented given R is G_R.

Remark: We term G_R to be the *genotype space* for that representation. For example, in the simple GA, the representation R is a binary string of length n, which gives a genotype space of cardinality 2^n. We will generally omit the subscript R when it is not necessary or is clear from context. Abstractly, the genotype space may be finite or infinite (though in practice the latter is constrained by machine precision and available memory.)

Definition 2.2 (Phenotype, Genotype to Phenotype Map)**.** A *phenotype* p is the decoded expression of a genotype. For each genotype representation R there exists a *genotype to phenotype map* $m : G_R \to P_R$, which is a surjective function to the set of all phenotypes P_R supported by R. We term P_R the *phenotype space*.

Remark: The term phenotype in EC is used diversely and can refer to multiple layers situated between genotype and evaluation. The definition above applies to the concept of a single intermediate layer between genotype and fitness. We will refer to this as a

single layer GPM. This issue will be returned to in Chapter 3. In a single layer genotype to phenotype map we say that two representations R and R' are strictly equivalent if they are equal in genotype space, phenotype space and genotype to phenotype mapping. Formally, if R and R' are equal then the set G_R is equal to G'_R, the set P_R is equal to P'_R and for each genotype $g \in G_R$, the phenotype $p = m(g)$ is equal to $p' = m'(g')$. We say that in a *direct* GPM, as exists in standard tree genetic programming, the map is bijective, such that there is a one to one mapping between genotype and phenotype. An *indirect* GPM has an associated genotype space which is not equal in size to its phenotype space. The mapping is said to be *fixed* if the mapping function does not change with evolutionary time.

Definition 2.3 (Fitness Function)**.** A fitness function is a function $f : P_R \rightarrow \mathbb{R}$ which specifies the objective quality of a phenotype for a given combinatorial optimisation problem. The function may be *static* or *time-dependent* (with respect to evolutionary time).

Remark: In the first part of this thesis, we will consider only static fitness functions. The second part will consider coevolutionary systems where fitness cannot be framed as a static objective measure, addressed in Chapter 5.

Definition 2.4 (Search Operator)**.** A search operator s is a relation $s_{\mu\lambda}$ between G^μ and G^λ, where $G^\mu = \{g_1, g_2...g_\mu\} \subseteq G_R$ is a subset of parent genotypes and $G^\lambda = \{g'_1, g'_2...g'_\lambda\} \subseteq G_R$ is a subset of offspring genotypes.

Remark: For example, the point mutation search operator acting on a binary string representation is a bit-flip acting on a single element of one genotype. The operator transforms one genotype onto one other member of the set of genotypes at Hamming distance of 1 from it.

Definition 2.5 (Population)**.** A population is a finite collection of genotypes. A population is said to be *homogeneous* if all genotypes in the population can be derived from the same genotype representation.

Remark: By convention, populations of genotypes in EC are homogeneous. However, it is common for populations which are separate but coevolving (see Chapter 5) to use different genotype representations.

2.7 Properties of Genotype to Phenotype Maps

Genotype to phenotype maps in biological systems describe complex, many-many relationships between genotype composition and the expressed traits in an organism. Establishing causal relationships in the map for a particular species is a difficult task, which can require direct observation of accumulated genetic change amongst a population. The aim of this section is to review, at a high level, the key common properties that have historically been identified in artificial GPM. Many of the terms used in EC have parallels in biological systems, but have been adapted by the Evolutionary Computation field. Fortunately, the vastly greater simplicity of biologically-inspired heuristics means that these concepts are easier to characterise.

Perhaps the most consolidated framework addressing the characteristics of genotype to phenotype maps and representations in Evolutionary Computation is the work of Rothlauf (2006). Rothlauf considered three qualities associated with GPM, *locality*, *redundancy* and *scaling*. The concepts were initially defined within the context of binary strings, but later extended to tree structures. The following sections summarise these concepts, in addition to describing the related issues of *robustness* or genetic canalisation, *dimensionality* and *modularity*.

2.7.1 Locality

Early Work

Informally, the *locality* of a genotype to phenotype map describes how the local neighbourhood structure of the genotype space corresponds with that of the phenotype space. In a *strongly local* genotype to phenotype map, genotypes that are defined as neighbours also possess neighbouring phenotypes.[2] Selecting a neighbouring genotype results in a new phenotype that is similar to the old, according to chosen measures of distance in each space. References to bias due to transformation of the local neighbourhood can be found in the early GA literature (Goldberg, 1989). Attention to this issue arose primarily out of the realisation that GA performance is significantly affected by the selection of different bitstring encodings. The term 'Hamming cliff' was introduced to describe the situation where, under a binary encoding, a small increase in the integer value of a bitstring could require changes to many bits simultaneously. This was addressed by mapping GA strings using a Gray encoding, initially investigated empirically

[2]There has been some difference in the conventions for qualitative descriptions of locality (Rothlauf, 2006; Galvan-Lopez *et al.*, 2011a). The terms *strong* and *weak* locality are used in this thesis, after Sendhoff *et al.* (1997a).

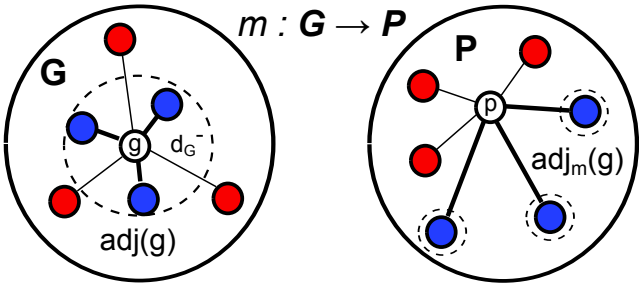

FIGURE 2.5: *A genotype to phenotype map with weak locality. The relative distance of neighbouring genotypes to* g *increases under the phenotype distance measure, such that they are no longer local neighbours in the phenotype space.*

(Caruana *et al.*, 1998), leading to later theoretical analysis (Rowe *et al.*, 2004). Droste *et al.* (2002) used distance metrics on the genotype and phenotype space to preserve the neighbourhood structure for EAs on Boolean functions. Efforts to formalise locality as a general property of GPM appeared over the same period (Sendhoff *et al.*, 1997a; Raidl and Gottlieb, 2005).

Quantifying Locality in GA Representations

The interpretation of locality in Rothlauf (2006) considers a finite set of discrete genotypes G and set of phenotypes P, with a surjective mapping function, over an arbitrary representation. A discrete measure of distance is assumed to be paired with each set, d_G and d_P. We write the shortest and largest non-zero distance in each space as d_G^-, d_P^-, d_G^+ and d_P^+ respectively:

$$\begin{aligned} d_G^- &= min\ d_G(g,g'),\ \forall g,g' \in G \text{ such that } d_G(g,g') > 0 \\ d_P^- &= min\ d_P(p,p'),\ \forall p,p' \in P \text{ such that } d_P(p,p') > 0 \\ d_G^+ &= max\ d_G(g,g'),\ \forall g,g' \in G \text{ such that } d_G(g,g') > 0 \\ d_P^+ &= max\ d_P(p,p'),\ \forall p,p' \in P \text{ such that } d_P(p,p') > 0 \end{aligned} \tag{2.1}$$

The *local neighbourhood* of a genotype g is then the subset of adjacent genotypes to g, for which $d_G(g,g') = d_G^-$. The function $adj(g)$ returns the local neighbourhood of g. Equivalently, the function $adj(p)$ returns the local neighbourhood of a phenotype p. Let $adj_m(g)$ return the phenotypes that correspond to the local neighbourhood of g, transformed into the phenotype space under the mapping function m. This function is illustrated in Figure (2.5). Note that in general $adj_m(g) \neq adj(m(g))$. Written using these definitions, Rothlauf quantified locality as:

Definition 2.6 (Locality)**.** Rothlauf (2006)

$$L = \sum_{g \in G} \sum_{p' \in adj_m(g)} d_P(m(g), p') - \bar{d_P}$$

where $p = m(g)$. This summation provides information on the distance of neighbours when mapped to the phenotype space. In a mapping function where all neighbouring genotypes correspond to neighbouring phenotypes, $L = 0$. A mapping with this property will be termed *fully local*. Weakly local mappings correspond to values $L >> 0$.

Rothlauf's original expression is not normalised to the search space size. A variation of Rothlauf's definition was put forward by Chiam *et al.* (2006). Chiam noted that:

> *"Besides considering the mean genotype distance for all neighbouring phenotypes as in Rothlauf's locality measure, the converse which corresponds to the mean phenotype distance of all neighbouring genotypes should be taken into account also."*
>
> **- Chiam *et al.* (2006)**

This statement acknowledges that it is possible to define an 'inverse' measure of locality, by considering the local neighbourhood in the phenotype space. Chiam introduced two similar measures based on Rothlauf's original definition of locality, which he termed *proximity preservation*, to measure the locality of binary and Gray bitstring encodings. For each $g \in G$ and $p \in P$, these measures were defined as (our notation):

Definition 2.7 (Proximity Preservation)**.** Chiam *et al.* (2008)

$$\begin{aligned} \bar{d_{Pg}} &= \frac{1}{|adj_m(g)|} \sum_{p' \in adj_m(g)} d_P(m(g), p') \\ \bar{d_{Gp}} &= \frac{1}{|adj_m^{-1}(p)|} \sum_{g' \in adj_m^{-1}(p)} d_G(m^{-1}(p), g') \end{aligned}$$

where $adj_m^{-1}(p)$ gives $adj(p)$, transformed back into the genotype space. The first definition $\bar{d_{Pg}}$ is just the average difference over adjacent genotypes to g, measured in the phenotype space. The second definition $\bar{d_{Gp}}$ is similarly calculated in the genotype space for adjacent phenotypes to p. Chiam was able to compute the value of $\bar{d_{Gp}}$ for this case because the mapping function was assumed to be bijective (i.e. there exists a one-to-one mapping between genotypes and phenotypes and $|G| = |P|$).

TABLE 2.2: *Previous aggregative definitions used to measure the locality between genotype and fitness spaces in GP (shown for discrete fitness values)* From Galvan-Lopez *et al.* (2011a), in the notation of this thesis.

Definition	Note
$L_0 = \frac{\sum_{i=1}^{N} \lvert d_P(g_i, s(g_i)) - d_P^- \rvert}{N}$	A direct extension of Rothlauf's definition.
$L_1 = \frac{\sum_{i=1}^{N} \lvert d_P(g_i, s(g_i)) \rvert}{N}$	Average fitness distance including neutral changes.
$L_2 = \frac{\sum_{i=1:d_P(g_i,s(g_i))\geq 2}^{N} d_P(g_i, s(g_i))}{N}$	Only fitness changes greater or equal to unit 2.

Quantifying Locality in GP Representations

Recent studies of locality in GP have extended these concepts to define computable measures of locality in program spaces (Galvan-Lopez *et al.*, 2011a; McDermott *et al.*, 2011; Krawiec, 2011). Galvan-Lopez *et al.* (2011a) considered a set of three definitions, similar to Definition 2.6. These are summarised in Table 2.2 (our notation). The definitions were presented for a direct tree GP map, where $|G| = |P|$. Measurements of locality were defined with respect to the mapping to fitness values, such that phenotype differences were measured as the absolute difference in phenotype fitness:

$$d_p(p, p') = |f(p) - f(p')| \tag{2.2}$$

Each measure was computed over a number of samples N on the genotype space. The measure $d_p(g_i, s(g_i))$ is the difference in fitness between the indexed genotype $g_i \in G_R$ and that genotype following a mutation operator of the form $s : G_R \to G_R$. Galvan-Lopez *et al.* (2011a) defined the minimum phenotype distance d_P^- to be a small difference in fitness. In first definition L_0, Equation 2.6 was extended in a direct fashion. For integer fitness values, L_0 assumed $d_P^- = 1$, such that mutations which did not change fitness contributed to the summation. The second definition, L_1, removed this contribution (in this notation, assuming $d_P^- = 0$). The third definition, L_2, considered only larger fitness changes. For integer fitness values, differences of at least two were chosen. In problems with real numbered fitness, these boundaries were parametrised to continuous values. After comparison over a set of standard Tree GP benchmark problems and operators, the authors concluded that the variant L_0 closest to the original definition presented by Rothlauf (2006) (Definition 2.6), gave a better overall correlation with performance.

Generally, we will refer to the methods of Rothlauf (2006); Chiam *et al.* (2006); Galvan-Lopez *et al.* (2011a) as *aggregative* approaches to measuring locality. The common property throughout these approaches is that they rely on an explicit summation over pairs within a given distance interval, to characterise how the neighbourhood of genotypes is preserved under transformation.

2.7.2 Other Properties

Redundancy

Redundancy in representations is the concept that elements of a program's representation may not be expressed in the output code. Such redundant material may be of significance to the heuristic search, being reintroduced by operators as active code at a later stage, but does not affect the fitness of a program at time of evaluation. A *redundant* genotype to phenotype map possesses more possible distinct genotypes than phenotypes, $|G| > |P|$. The redundancy of a representation R can be characterised by the expression:

Definition 2.8 (Redundancy)**.** Rothlauf (2006)

$$\Upsilon = \sum_{p \in P} \frac{1}{2} \sum_{g \in G_p} \sum_{g' \in G_p} d_G(g, g')$$

where $G_p \subseteq G$ is the subset of genotypes that correspond to the phenotype p (i.e. $m(g) = p$ for all $g \in G_p$). This corresponds to summing up all the distances between genotypes which map to p, for all distinct phenotypes in the phenotype space.

The terms *synonymous* and *non-synonymous* were used by Rothlauf to label different kinds of redundant mapping. Synonymous implies that for the majority of phenotypes, the corresponding genotypes equivalent to that phenotype are close together in the genotype space ($\Upsilon \to 0$). Conversely, in the non-synonymous case the set of equivalent genotypes are far apart, giving a high total ($\Upsilon \gg 0$). As for Definition 2.6, the requirement is again to adequately define the measure d_G, which in turn depends on the particular data structure and search operators.

Scaling and Dimensionality

Scaling is a qualitative attribute which has been defined in linear representations to describe the variation in response to changing the genotype with respect to different positions within the data structure. Rothlauf divides representations as *uniformly* and *non-uniformly* scaled. A uniformly scaled genome is one in which changes at any point in the genome lead to equally significant changes in the phenotype. A binary genotype that is encoded directly onto the corresponding set of integers is non-uniformly scaled, because flipping a bit at the beginning of the genotype induces a greater change than at the end. Scaling is not necessarily coupled to the mapping between phenotype and

fitness. In general, trees, graphs and grammar-based GP all have the property that they are non-uniformly scaled. Efforts to remove position dependency have been made, for example by changing the order in which the genotype is interpreted. A method of achieving this, 'πGE', was demonstrated for grammatical evolution with reported performance gains (O'Neill *et al.*, 2004). To the author's knowledge, however, no general quantitative definitions of scaling have been presented in GP.

A closely related issue to scaling is the concept of dimensionality in the genotype and phenotype. Transforming a genotype to a phenotype can alter the number of components able to vary throughout the structure. Mapping from an n bit binary string to an integer value projects a vector of n components onto a single variable, reducing the number of degrees of freedom from n to 1. A change in the number of degrees of freedom may occur in any genotype to phenotype map where the genotype representation is different to the representation of the phenotype. This is distinct from changing the cardinality (size) of the space. In GP genotype to phenotype maps, the dimensionality of the genotype is generally at least as large as that of the phenotype, because operations are applied at the genotype level. Dimensionality is significant in GP because the degrees of freedom in the phenotype are often associated with identifiable subcomponents (for example subtrees) that define program fitness. A mapping in which the genotype space has a much higher dimensionality than the corresponding phenotype space implies that the contribution of many subcomponents of the genotype to fitness will be either coupled or redundant. Conversely, if there are fewer degrees of freedom in the genotype, then variations to the genotype may perturb the phenotype in several dimensions.

Modularity

From the inception of GP it has been recognised that the software engineering principle of encapsulating code in common functional units (modules) is a key component in the evolution of complex programs (Koza, 1992). A form of modularity may be introduced into a genotype to phenotype map 'internally' through the use of a particular representation which reproduces common elements, such as a graph or grammar. Alternatively, 'external' mechanisms explicitly designed to capture and reuse complex components can be constructed. Variations on current techniques to introduce modularity externally have been described for all the classes of GP detailed in the previous section. Examples include Koza's automatically defined functions (Koza, 1992), modular Cartesian Genetic Programming (Walker and Miller, 2008) and modular forms of Grammatical Evolution (Swafford *et al.*, 2011).

Robustness and Neutrality

Robustness - analogous to genetic canalisation in a biological system - describes the resilience of a phenotype to change, whether derived from genetic or environmental factors (Waddington, 1942). The effect increases the stability of established phenotypic traits, which may present an evolutionary edge. It has been claimed that similar qualities in artificial evolutionary systems are advantageous in particular cases, for example Hu *et al.* (2011). Given the highly simplified model of evolution used in genetic algorithms or genetic programming, where the source of phenotypic change is solely 'genetic', there is an implied link between robustness and phenotypic *neutrality*. The significance and contribution of neutrality is a complex unresolved issue and has been the subject of considerable debate (Yu and Miller, 2006; Collins, 2006; Wilson and Kaur, 2009; Galvan-Lopez *et al.*, 2011b). This thesis will not directly study the role of neutrality in the GPM. For convenience, we will define neutral changes to the genotype to be those which have no effect on the phenotype after mapping, $m(g) = m(g')$. A subset of genotypes will be described as *phenotypically neutral* if they map onto the same phenotype. A pair of genotypes will be said to be *fitness neutral* if they are distinct but map onto the same fitness value.

2.8 Discussion and Concluding Remarks

Despite the relative simplicity of the relationship between genotype and phenotype considered in GP compared with biological systems, it can be seen from the previous section that a range of different attributes have nonetheless been defined. Whilst each of these elements has been investigated within GP systems, a unified view of their contribution to search performance is yet to emerge for GP and EAs using executable program representations. This has been due in part to historical divisions between the different classes of EAs and the relatively recent migration of quantifiable measures of the different GPM properties from the GA to the GP literature.

Defining locality in GP is more complex than for conventional genetic algorithms because GP representations are likely to include multiple levels of mapping. Present studies of locality in GP have been primarily concerned with introducing stronger locality into GP genotype to phenotype maps (Krawiec, 2011). This view is driven by the understanding that weak locality when mapping between genotype and phenotype reduces the causal link between genotypic change and fitness (Sendhoff *et al.*, 1997a), and the prediction that this reduces performance on 'easy' problems in the present GP paradigms. Empirical studies improving GP locality and testing this assumption have supported this

hypothesis, for some problem cases. These studies are currently limited to direct, tree-based representations (Galvan-Lopez *et al.*, 2011a) and Grammatical Evolution (Rothlauf and Oetzel, 2006). A further consideration is the distinction between syntactic and semantic levels of expression in GP (Nguyen *et al.*, 2011a). We will return to this last issue in the next chapter.

In addition, choice of a given GP representation cannot be isolated from the assumptions that have been made regarding search operators. The mutation operator (in various forms) is a ubiquitous component of GP techniques. Crossover, by contrast, is not universal and its benefits are disputed (White and Poulding, 2009). Furthermore, the definitions of locality reviewed in Section 2.7.1 do not naturally extend to crossover using pairs of genotypes without some modification. For these reasons the analysis in the following chapters will concentrate on the locality of GP representations when using the mutation operator only.

A final consideration is to note that particular GP representations and GPM can over-represent portions of the phenotype space in some cases. This leads to a bias that is less transparent for GP than GAs. Therefore, to gain a more satisfactory understanding of locality in GP in general, its effects should be examined over a range of different GP paradigms and problem domains.

2.9 Chapter Summary

This chapter provided a foundational overview of the historical classes of evolutionary algorithms, the principal GP paradigms and properties of artificial genotype to phenotype maps. The review included:

- An outline of the representational differences between tree, graph, grammar and linear GP algorithms.
- The concept of GPM locality, summarising previous aggregative definitions and current assumptions regarding the role of locality in GP.
- A summary of other known GPM characteristics including redundancy, scaling, dimensionality, modularity, robustness and neutrality.

The next chapter elaborates on the issues highlighted in the discussion and builds on these concepts to develop a rigorous statistical measure of locality across different GP paradigms.

Part II

Statistical Tools and Visualisation

Chapter 3

Measuring Locality in Genetic Programming Representations

Contents

3.1 Chapter Motivation

The previous chapter provided an overview of the different paradigms used in current genetic programming techniques. Moving away from the relatively transparent properties of bit string to tree, graph or grammar representations presents several challenges. In Chapter 1, it was noted that the theoretical properties of different genotype to phenotype maps in genetic programming are poorly understood. We can consider four factors that have contributed to this situation:

1. Ambiguity over what constitutes the genotype and phenotype for a problem.
2. Multiple levels of mapping.
3. Choice of appropriate metrics when using complex search operators.
4. Unknown contributions from correlated GPM properties.

The first and second points concern lack of clarity when assigning the terms genotype and phenotype. Multi-level maps are common in the genetic programming paradigms; note the number of intermediate steps mapping between genotype and phenotype in the different classes of genetic programming described in Chapter 2. The third point derives from the diversity of different representations, operators and reproduction methods which have been applied in Evolutionary Computation. This situation is summarised eloquently by Graff and Poli:

> *"Despite the simplicity of EAs, sound theoretical models of EAs and precise mathematical results have been scarce and hard to obtain, often emerging many years after the proposal of the original algorithm... A key reason for this is that each algorithm, representation, set of genetic operators and, often, fitness function requires a different theoretical model. In addition, the randomness, non-linearities and immense number of degrees of freedom present in a typical EA make life very hard for theoreticians."*
>
> **- Graff and Poli (2010)**

Exact analysis of all of these heuristics is prohibitively time consuming. The problem is also furthered by the issue that practitioners typically customise a particular set of operators and GPM to address individual problems, which leads to difficulty when gathering evidence on general properties of representations. Finally, studies of GP genotype-phenotype maps generally consider the effect of one property in isolation. No formal understanding exists of the relationship between, for example, locality and modularity (beyond strictly qualitative impressions).

In order to make progress against this backdrop of uncertainty new approaches must be developed and tested that bridge different GP representations. The direction pursued here is to consider some standard examples of well established indirect GP maps that have been proven empirically to be successful for a relatively wide range of problems. Clearly, judgement regarding what has constituted a 'successful' GPM is highly subjective. This only reinforces the requirement for more comparative research and a better understanding of the effects of different classes of mapping.

3.2 Chapter Outline

This chapter addresses the question: **'How can a statistical measure of genotype to phenotype locality be developed for different genetic programming representations?'**. The aim is to develop a statistically rigorous method of measuring locality, which can be applied to GP representations in general. Section 3.3 explores the rationale for adopting a statistical approach, directly correlating genotype and phenotype distances. We then consider an underlying issue in the application of these approaches to program evolution, which is the selection of appropriate distance metrics in the genotype and phenotype space. In Section 3.4, an appropriate technique is identified and adapted, the Mantel statistic, derived from numerical ecology. The extensions required to apply it to artificial GPM are provided. Section 3.5 describes methods of biased sampling on the genotype space that can be used in conjunction with the statistic. The technique is then applied in Section 3.6 through a set of empirical case studies that demonstrate the generality of the approach under three mappings: a test *Weighted Integer* encoding, canonical *Cartesian GP* and *Grammatical Evolution*. The remaining sections analyse the outcomes of our case studies and evaluate the advantages and limitations of this method, relative to existing approaches.

3.3 Measuring Locality in Genetic Programming

In classical tree GP there is no explicit, intermediate state between genotype and phenotype. Operators act on the GP trees directly. One drawback of the general aggregative methods of measuring locality described for GP in the previous chapter is that we may lose insight into where in the mapping process the causal relationship between genotype distances and phenotype distances is sustained and where it becomes more random. The requirement for EAs to support such a relation is highlighted in McKay *et al.* (2010):

> *"The underlying assumptions of all evolutionary applications about the search space include... that there is sufficient correlation between fitness and semantics, so that non-random search is useful... that there is sufficient correlation between distance in the genotype and phenotype (semantic) spaces that evolutionary search is able to take advantage of the first correlation... that these relationships are nevertheless sufficiently uneven (that is the fitness landscape is sufficiently rough) that deterministic search methods do not perform well"*
>
> **- McKay *et al.* (2010)**

Another consideration is that the currently proposed measures of GP locality address all genotypes within a single mutation with equal priority. However, the probability of producing two genotypes from a parent that are both structurally the same distance away may differ significantly in some non-traditional representations. Caution is therefore called for when extending this assumption to other forms of GP. The method presented in this chapter considers an alternative to the previous aggregative approaches and explores a technique to directly correlate primitive genotype distances and phenotype distances. Statistical approaches to analysing GP based on distance metrics have been attempted previously, particularly in the context of measuring the diversity of EA populations (Vanneschi, 2004). There exists some commonality between the method described here and *fitness distance correlation* (FDC), extensively addressed in both the GA and GP literature (Jones and Forrest, 1995). Fitness distance correlation develops a measure of problem difficulty by considering distances to the generally unknown optimum in fitness space. Similarly, the family of techniques using *autocorrelation* (for example on random walks) can provide measures of the overall 'smoothness' of a given fitness space (Weinberger, 1990; Stadler, 1996). However, these methods are quite distinct from approaches that analyse the relationship between genotype and phenotype prior to fitness evaluation.

3.3.1 Metrics for Genotype and Phenotype Spaces

In developing a better understanding of locality at the syntactic and semantic levels, metrics are required that capture the distance between phenotypes at each stage of the mapping. Recall that a metric space is given by four criteria,

Definition 3.1 (Metric Space)**.** A *metric space* is a pair $Q = (X, d)$ consisting of a set, X and *metric function*, d, satisfying the constraints:

1. $d(x, y) \geq 0$
2. $d(x, y) = 0$ iff $x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, y) \leq d(x, z) + d(z, y)$ where $x, y, z \in X$.

The conventional term *semimetric* will be adopted when the triangle-inequality (condition 4) does not necessarily hold. In a *pseudometric*, the triangle inequality holds, but the coincidence axiom (condition 2) is relaxed, such that there may exist some cases $d(x, y) = 0$ when $x \neq y$.

When considering metrics on GP programs, a fundamental difficulty arises because the closer a measurement of similarity is to the obtained fitness difference, the more problem dependent it becomes. This is related to the statement that any realistic measure of similarity needs to capture the difference in program behaviour within the context of the problem domain. In phenotype spaces, this measure is determined by the program's semantics. By comparison, in the genotype space, selection of the metric is informed by the particular representation and variation operators.

3.3.2 Genotype Metrics

Numerous proposals have been put forward for appropriate metric distance measures in GA/GP spaces under different operators; see for example the reviews in Hien and Hoai (2006). These can be divided into four broad classes, based on the measure used:

1. *Edit distances* e.g. Hamming, Levenstein in the case of strings and tree or graph edit distances respectively (Payne and Stepney, 2009; Galvan-Lopez *et al.*, 2011a)
2. *Subtree distance* e.g. Keijzer, Tree Alignment (Keijzer, 1996; Vanneschi, 2004)
3. *Information compression* e.g. Normalised Compression Distance (Galvan-Lopez *et al.*, 2011a)
4. *Probabilistic measures* e.g. the Subtree Crossover Operator (Gustafson S., 2008)

Although it is normal practice to define strict metrics for the genotype space under the assumptions made in categories 1, 2 and 3, measures based directly on probabilities usually violate one or more of the metric criteria.

Metrics under Mutation

For genotype spaces with discrete genotypes (as will be analysed in the following sections) one convenient measure is the expected number of independent attempts that would be required to generate a genotype from another through a single mutation. Assuming that the two genotypes are mutually reachable (Definition 3.2), then this is just the inverse of the probability of mutating between both genotypes. This semimetric will be referred

to as the *expected variation distance* $\bar{M}$. The measure has the advantage that it defines distance based on the actual transition probability.

Definition 3.2 (Mutually reachable genotypes)**.** *A genotype g' is reachable from a genotype g under some variation operator s, given that the probability of deriving g' in a single operation $s(g)$ is greater than zero. The pair of genotypes (g, g') are mutually reachable if g is reachable from g' and g' is reachable from g.*

Definition 3.3 (Expected variation distance)**.** *A semimetric $\bar{M} : G \times G \to \mathbb{Z}$, that acts on pairs of mutually reachable genotypes (g, g'). $\bar{M}(g, g', s)$ gives the expected number of independent operations on g such that there is an instance $s(g) = g'$. If $g = g'$, we define $\bar{M}(g, g, s) \equiv 0$.*

Similar, simple measures based on the expected number of mutations have appeared previously in the evolutionary computation literature (Sendhoff *et al.*, 1997a,b), though have not to the author's knowledge been employed as a metric to quantify locality.

Example A: Computing the Expected Variation Distance

To illustrate Definitions 3.2 and 3.3, first consider the case of a genotype space composed of length n bitstrings, $G_{BIT} = \{0, 1\}^n$. Contrast the calculation of $\bar{M}$ on genotypes in G_{BIT} for two operators i) uniform mutation and ii) point mutation:

i) The uniform mutation operator flips each bit with a fixed probability q. The probability of transforming g into g' is $q^h(1-q)^{n-h}$, where h is the hamming distance between g and g'. All pairs of genotypes, $(g, g') \in G_{BIT} \times G_{BIT}$, are mutually reachable, since the action of the operator may transform any genotype into any other genotype and back with non-zero probability. Therefore the expected variation distance is given by the inverse of the mutation probability, $\bar{M} = (q^h(1-q)^{n-h})^{-1}$. For example, for the bit strings $g = (0, 1, 0)$ and $g' = (1, 0, 0)$, we have $n = 3$ and $h = 2$. If $q = \frac{1}{3}$, this gives $\bar{M} = 13.5$ (it takes on average between 13 and 14 independent operations on g to produce g'.)

ii) The point mutation operator only a single bit each operation. Therefore only genotypes at a distance of $h = 1$ are mutually reachable. The expected variation distance for these cases is just $\bar{M} = n$, since there are n hamming neighbours. For all other cases, the expected variation distance is not defined.

Now consider the expected variation distance between a pair of standard feed-forward CGP genotypes $(g, g') \in G_{CGP} \times G_{CGP}$ (details of the CGP genotype space are given in Chapter 2). The derivation is as follows (Seaton *et al.*, 2012a):

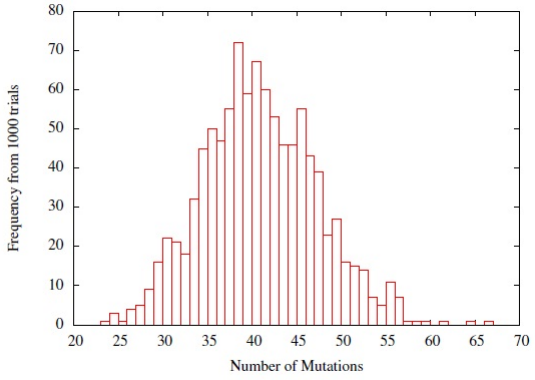

FIGURE 3.1: *The distribution of the number of transitions between CGP genotypes* $(000, 022)$ *and* $(002, 000)$ *in* 10^6 *mutations* (*over* 1000 *trials*)

Cartesian Genetic Programming with Uniform Mutation Operator

CGP genotypes are assumed to be equal sized integer strings of length n, which represent a single row with feed-forward connections, where output is derived from the right-most node. Assume there are x matching integers and $n - x = y$ different values. The genotype is split into integers corresponding to connections and functions. From the y different integer values, we have two subsets, one of size y_F containing different values corresponding to functions and one of size y_C containing values corresponding to connections. Assume a mutation operator that acts on all values with uniform probability q, where a mutation changes the integer to any other feasible value. Then, the probability of x values remaining the same is $(1-q)^x$. The probability of y_F values from g mutating to the same function as that in g' is $(\frac{q}{|F|-1})^{y_F}$, where $|F|$ is the size of the function set. Let $\mathbf{y_C}$ be the set of different integer values corresponding to connections. Each connection $i \in \mathbf{y_C}$ has $c_i - 1$ possible alternatives (where c_i is in general the total number of inputs, plus all previous nodes). Thus the probability of obtaining the same set of connections is $q^{y_C} \prod_{i \in \mathbf{y_C}} \frac{1}{c_i - 1}$. The total probability of mutating from one CGP genotype to another P_{CGP} is therefore:

$$P_{CGP}(g, g') = (1-q)^x \cdot \left(\frac{q}{|F|-1}\right)^{y_F} \cdot q^{y_C} \prod_{i \in \mathbf{y_C}} \frac{1}{c_i - 1} \tag{3.1}$$

Taking the inverse and collecting terms gives the expected number of independent mutations required, $\bar{M} = \frac{1}{P_{CGP}}$:

$$\bar{M} = \frac{(|F|-1)^{y_F}}{q^y \, (1-q)^x} \prod_{i \in \mathbf{y_C}} c_i - 1 \tag{3.2}$$

Equation 3.2 was verified experimentally using an example based on two simple CGP

genotypes, (000,022) and (002,000). The associated function set is $\{+, -, \times, \div\}$ and input set is {1,0}. The two genotypes therefore correspond to the expressions 1*(1+1) and (1+1). The mutation probability between the two is 4.0501×10^{-5} to 5 sig. fig. giving a distance $\bar{M} \approx 24690$. Hence, for a set of 10^6 mutations we would expect around $10^6/24690 \approx 40.5$ mutations from (000,022) to (002,000) on average. Figure 3.1 illustrates the corresponding distribution obtained over 1000 trials. The experimental mean matches the predicted value.

3.3.3 Phenotype Metrics

Measurement of the distance between complex phenotypes, such as programs, can be approached differently depending on the stage of the genotype to phenotype mapping process under discussion (McDermott *et al.*, 2011). To illustrate this, consider the classic GP *lawnmower problem* proposed by Koza (1992). In the lawnmower problem, programs are evolved that provide instructions to steer a 'lawnmower' agent on a n x m square lawn. Visiting a square 'mows' the grass on that square. Fitness is assigned based on the number of squares uncut at the end of a program, thus the goal is to ensure all squares are visited at least once. Squares may be revisited freely and the lawn wraps around such that when the agent moves off one side of the grid, it re-enters on the other. Thus a program maps deterministically to a route through the grid. The term phenotype could therefore encompass both syntactic and semantic states: the program, the route defined by the program, the state of the lawn at program completion, or the corresponding fitness value (sketched in Figure 3.2). Redundancy exists at each stage of this process - multiple programs may define the same route, many routes may achieve the same lawn-state and there are a number of permutations on the cut squares that give the same fitness value. This process is complicated further when we consider that there may exist additional layers of encoding and redundancy when mapping between the genotype and executable structure, as described for CGP and Grammatical Evolution in Chapter 2.

General approaches in the GP literature to measuring differences at the phenotypic level have concentrated either on the truth tables (Koza, 1992) or absolute differences in program fitness, for example Galvan-Lopez *et al.* (2011a). Given that deriving the output for all possible input states to a program is generally prohibitively expensive, it can be necessary to approximate the differences by some suitable sampling process. Examples of this method include the recently introduced approach of Nguyen (2011) under the term *sampling semantics* and Krawiec (2011) in his description of semantic embeddings. In this chapter we will concentrate on measuring differences at the syntactic level. Specific examples of characterising differences at the fitness and semantic levels will be considered in Chapters 4 and 7.

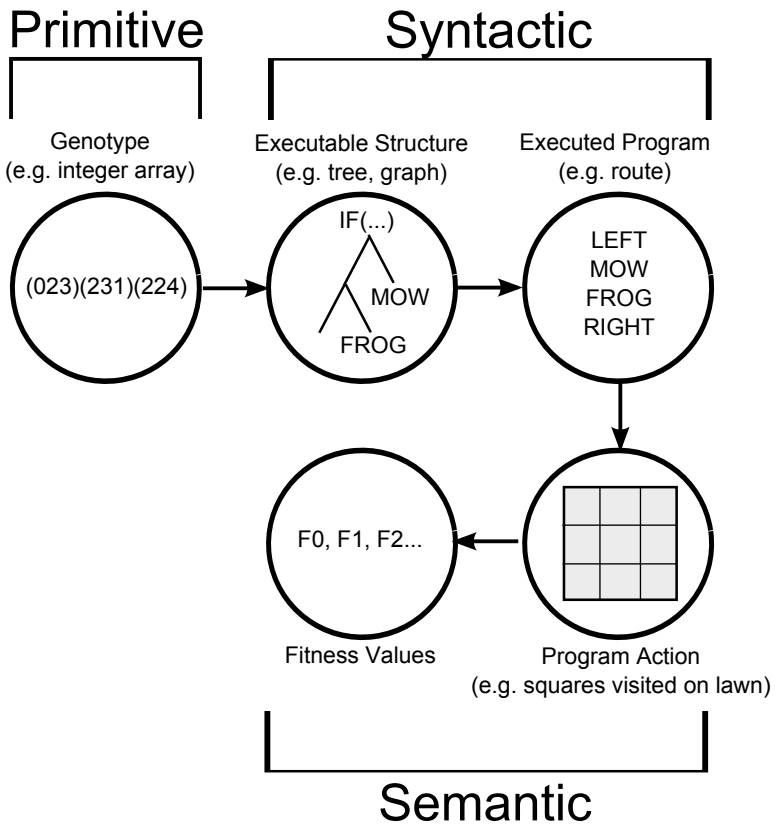

FIGURE 3.2: *Abstract genotype to fitness mapping for the classic GP lawnmower problem. GP genotype to fitness maps can exhibit several intermediate stages.*

3.4 Correlating Genotype and Phenotype Distances

3.4.1 The Mantel Test

The Mantel test is a general, non-parametric statistical resampling technique, used to explore the correlation between two triangular distance matrices (Mantel, 1967). Historically, the test was designed to address the analysis of spatial and temporal data from disease clustering. It has seen considerable application in numerical ecology (Oden and Sokal, 1986; Legendre and Fortin, 1989; Legendre *et al.*, 1994; Lichstein, 2006) and on genetic and linguistic data (Legendre and Fortin, 2010). In a mathematical sense, the Mantel test provides a permutation-based method of determining the statistical significance of linear or monotonic relationships between data. The test is applicable in situations when it is desirable to determine whether a correlation exists in the distances between elements sampled between two metric spaces. A key point is that standard, parametric significance tests cannot be employed for this situation, because in general distances derived from the same element in a space cannot be considered independent of each other. The technique is applied between two square distance matrices of size k, labelled $\mathbf{X}$ and $\mathbf{Y}$. The matrices contain the pair-wise differences calculated between all

elements of a sample under two measures of distance d_X and d_Y. For example, in the ecological context $\mathbf{X}$ might represent the geographical distances between samples of a species at particular sites and $\mathbf{Y}$ corresponding measured genetic distances. The differences are assumed to adhere to the symmetry property of a metric, so both matrices are symmetric with zeros along the diagonal. The original, 'standardised mantel statistic' is then given by the expression (Legendre and Legendre, 1998):

$$r_M = \frac{1}{s-1} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \left(\frac{X_{i,j} - \bar{X}}{\sigma_X} \right) \left(\frac{Y_{i,j} - \bar{Y}}{\sigma_Y} \right) \tag{3.3}$$

where r_M is the linear correlation coefficient obtained, $\bar{X}$, $\bar{Y}$ and σ_X, σ_Y are the mean and standard deviation calculated for the elements of $\mathbf{X}$ and $\mathbf{Y}$ respectively and $s = k(k-1)/2$. This is equivalent to calculating the Pearson-product moment (linear correlation) over the upper-half of the matrix.

3.4.2 Significance Testing for Genotype to Phenotype Maps

For r_M to be a useful statistic under sampling, significance testing should be carried out against the null hypothesis, H_0 that the distances in $\mathbf{X}$ and $\mathbf{Y}$ are uncorrelated. This takes place each time the Mantel statistic is calculated. Recall that in classical significance testing, a p-value is derived that denotes the probability of accepting the null hypothesis. *The principal realisation of the Mantel test is that, under the null hypothesis, rows and columns of the matrix are exchangeable.* That is we expect to be able to freely rearrange the labels of each set of distances. By permuting the rows (and corresponding columns) of $\mathbf{X}$ and recalculating r_M, a permutation distribution can be constructed from which the significance of correlations in the unpermuted data is obtained. Given that the null hypothesis is true, we would expect that the unpermuted data should lie somewhere in the centre of this range. The test proceeds by obtaining the original unpermuted coefficient r_M^0 and a set of coefficients under permutation of $\mathbf{X}$, denoted $r = \{r_M^1 ... r_M^N\}$, where N is the total number of permutations. Let $\chi \subseteq r$ such that $x \in \chi \geq r_M^0$. The probability of accepting the null hypothesis in the presence of an apparent positive correlation is then given by the one sided test

$$p(H_0|r) \approx \frac{||\chi||}{N} \tag{3.4}$$

that is the number of instances in which the recalculated coefficient equals or exceeds r_M^0, divided by the total number of permutations. A similar test can be carried out for the case of negative correlation. The test does not necessarily have to support a linear model: it may be appropriate to compute r_M using an alternative statistic, such as Spearman

rank-based correlation, using the permutation test in exactly the same fashion. The result converges monotonically on the true significance at large N. The number of permutations recommended in the literature varies, but a value in the range of 1000-10000 permutations is typically suggested (Legendre and Fortin, 2010).[1] *In all statistics quoted in this work, the minimum level assumed to be significant is* $p(H_0|r) = 0.005$.

Example B: Computing the Mantel statistic in a linear function

Consider the case of linear correlation between a square matrix of size 100 with elements $X_{i,j} = |i - j|$, the absolute difference between indices, and a corresponding noisy matrix $Y_{i,j} = X_{i,j} + G(\mu = 0, \sigma)$ (Figures 3.3, top left and top right respectively). Figure 3.3, bottom left, shows the calculated correlation as a function of increasing noise. Figure 3.3, bottom right displays the convergence in significance value as the number of permutations increase.

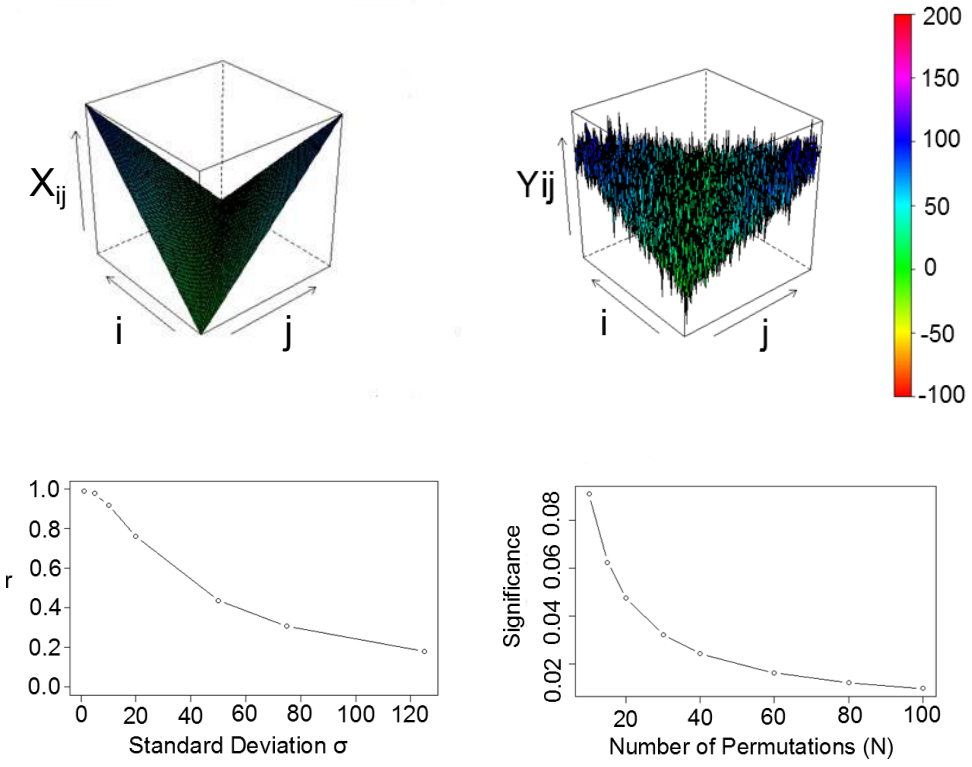

FIGURE 3.3: *Example B: Linear correlation calculated using the Mantel test between a pair of 100 × 100 distance matrices. As the number of permutations on the distance matrix is increased, the calculated significance converges on the true p-value.*

[1]Standard methods for carrying out the Mantel test are supported in numerical ecology statistical packages such as `ecodist` and `vegan` (Goslee and Urban, 2007), in *R*.

3.4.3 Extension to Multiple Distance Classes

To apply the Mantel statistic to an artificial genotype to phenotype map, a method is required to calculate it over particular distance intervals. This is to establish whether a correlation exists only for closer, or more distant, genotypes. A similar situation arises in numerical ecology, where correlations may be limited by time, or by geographic distance. Previous derived techniques of the Mantel statistic have considered correlations over particular ranges, such as the 'Mantel correlogram', which applies a model matrix to examine correlations over particular distance classes (Oden and Sokal, 1986). A simplified approach is used here, explicitly sub-dividing the distance matrix. Let $\mathbf{X_U}$ be the upper triangle of $\mathbf{X}$. A set of distance classes are selected such that each distance class $\mathbf{D}_{a,b}$ is a subset of the elements of $\mathbf{X_U}$ where $a \leq X_{i,j} < b$. Hence, a distance class contains the elements over which r_M is computed which fall within the range (a, b). The corresponding set of distances at the same index positions in $\mathbf{Y_U}$ are also found. The coefficient r_M is calculated separately for each distance class. Significance values are derived as before for each calculated statistic, by permuting the original matrix and recomputing r_M over that interval.

Example C: Computing the Mantel statistic in a periodic function

This procedure was tested using a noisy, periodic function between two artificial distance matrices. Matrix $\mathbf{X}$ was configured as before, $X_{i,j} = |i - j|$. Distance elements in $\mathbf{Y}$ were mapped according to the expression

$$Y_{i,j} = X_{i,j} + 10sin(\frac{\pi X_{i,j}}{10}) + 5G(\mu = 0, \sigma) \tag{3.5}$$

A uniformly distributed set of distance classes of size 5 were selected (one quarter of the period of Equation 3.5). Figure 3.4 illustrates Equation 2 and the correlation coefficients determined over each distance class. Under linear correlation, we expect the sign of each coefficient to reflect the overall change in gradient for each distance class, as observed. All statistics were significant at $p(H_0|r_M) < 0.005$, except the final distance category. This correctly reflects the sparsity of data in the last interval, validating the process.

3.5 Sampling the Genotype to Phenotype Map

For realistic problems, there is generally no prior knowledge of the distribution of genotype distances under a given metric. To inform an accurate measure of locality, a sufficiently representative sample is required across the genotype space. The effect of sampling on locality measurements in GP is poorly understood, an issue highlighted in

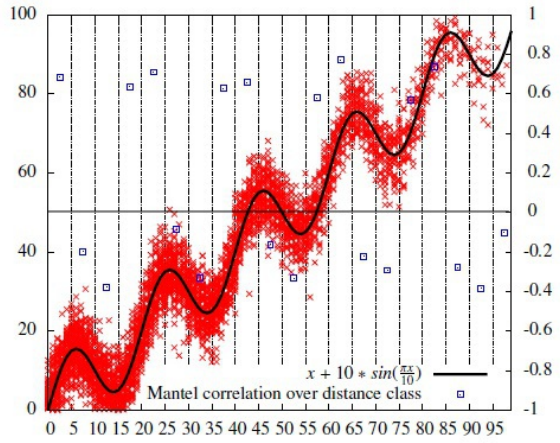

FIGURE 3.4: *The Mantel statistic on a sinusoidal test function mapping between two distance matrices. The matrices are subdivided over each distance interval such that a separate correlation coefficient and p-value can be obtained for each class. All coefficients above except for the* [95:100] *class were determined to be significant under permutation.*

Galvan-Lopez *et al.* (2011a). The most basic sampling method applied throughout the EC literature is to sample without bias across the whole genotype space (simple random sampling). However, where the genotype space is large, the probability of sampling two or more genotypes in close proximity at random is usually very small. Random sampling will provide a very low sampling density over one or more distance intervals. Application of the Mantel statistic to this problem requires that each distance interval must be populated with a sufficiently large set of measurements. Therefore it is necessary to explore methods of biased sampling.

3.5.1 Hamming Sampling

The most basic method of obtaining local samples is to enumerate all elements of a space inside a fixed distance. This requires the specification of a metric, such as the Hamming distance and a fixed point of interest in the search space, such as a global optimum. However as noted earlier metrics based on the number of different positions in a genotype such as the Hamming distance may not accurately reflect the distance of genotypes under some evolutionary operators. Furthermore, sampling local regions via the Hamming distance does not account for the scaling of the representation. By way of example, consider the Cartesian GP transition probability derived in Equation 3.2. The transition probability between genotypes is derived from the number of possible connections at each position, which scales with the genotype length. Transitions between genotypes which differ at positions closer to the output node are less likely.

3.5.2 Metropolis Sampling

The problem described above is related to the general question of sampling from a skewed distribution. This was addressed for GP representations by Vanneschi (2004), who employed the Metropolis-Hastings algorithm from statistical physics. Metropolis-Hastings is a generalisation of the Metropolis algorithm (Metropolis *et al.*, 1953), which provides a method of simulating a probability distribution when sampling from that distribution directly is difficult to accomplish. The Metropolis algorithm is a Markov process, where samples are drawn sequentially, only dependent on the previous sample. The probability of drawing each new sample is dependent solely on the previously obtained value.[2] Each newly drawn sample is accepted according to an *acceptance ratio*, defined as the probability of the new sample under the desired distribution divided by the probability of the old sample. In the limit, this converges on the distribution required.

Whilst the Metropolis family of algorithms can be applied to sample fitness values from a genotype space in a particular distribution, they cannot be used in this case. This limitation occurs because measurements under the Mantel statistic use pair-wise measures of distance. Given a set of sampled genotypes, distances are measured between all pairs. Therefore, generating the target distribution in this fashion is inappropriate because the derived distances cannot be obtained independently of each other.

3.5.3 Chain-Referral Sampling

One natural alternative to Hamming sampling that overcomes this issue is to construct a sampling strategy based directly on the mutation operator. A useful technique can be developed from the concept of *chain-referral sampling*, also referred to as 'snowball' sampling (Biernacki, 1981). Chain-referral sampling is used extensively in sociological research to sample for human participants from minority groups. From an initial subset of participants in a population each member is requested to recommend a collection of individuals who share a similar background or skill-set to themselves. By iterating the process, a network of individuals with the desired characteristics can be found. The input is a root genotype chosen from the unbiased distribution. The output of is a biarity tree which spans part of the neighbourhood under mutation for that genotype. Each level of the tree is generated by applying the mutation operator a fixed number of instances to the previous depth. This can be carried out recursively. The depth of the search tree can be varied to permit sampling of genotypes which correspond to more distant pairs under the mutation operator. Distance measurements are then derived

[2]The Metropolis-Hastings algorithm extends the Metropolis algorithm to the situation where the Markov chain can be asymmetric, that is $prob(stateX) \rightarrow X' \neq prob(stateX') \rightarrow X$

pair-wise between all genotypes in the sample tree. For convenience, we will refer to samples obtained using this method as 'ancestral-trees'.

3.6 Case Studies

This section details three experiments that explore the use of the Mantel statistic as a measure of locality, summarised in Table 3.1. Locality in genetic programming is a function of both the GPM and the search operators. The first experiment uses a simple, artificial mapping between integer arrays (genotypes) and real numbers (phenotypes), where the mapping is controlled using a fixed array of weights. The experiment contrasts a weakly local and strongly local mapping, by applying two different non-uniformly scaled genotype to phenotype maps. The second experiment uses a graph-based, Cartesian Genetic Programming (CGP) configuration, mapping between integer genotypes and the syntax of a derived program. Differences in syntax are measured using the Normalised Compression Distance (NCD) (detailed in Section 3.6.2). Here locality is varied by introducing a continuous mutation bias scaled using a sigmoid function, to vary distances in the genotype space. The third experiment applies a similar bias, but tests the approach using a basic Grammatical Evolution (GE) configuration.

3.6.1 Experiment 1: Weighted Integer Model

A simple model was first constructed based on a finite space of integer genotype vectors, $G = (0, 1...k)^n$. For a given (vector) genotype $\mathbf{g} \in G$, the phenotype is a real number specified by the product $\mathbf{w} \cdot \mathbf{g}$, where $\mathbf{w}$ is a fixed vector of positive real weights $\mathbf{w} = (w_1...w_n)$. A non-uniform mutation operator is used that randomly varies the components of $\mathbf{g}$, where the probability of varying each component is $\mathbf{q} = (q_1...q_n)$. Components can be mutated to any other valid integer value between 0 and k. Our expectation is therefore that by altering the components of $\mathbf{q}$ with respect to the weights

TABLE 3.1: *A summary of experiment representations and metrics in each case study. Genotype distances are measured using the expected variation distance. Phenotype distances are measured either as the absolute difference in fitness (weighted integer model), or between program syntax (CGP and GE examples).*

#	Representation	Mapping	Genotype Metric	Phenotype Metric
1.	$\mathbb{Z}^n$	Vector Product	$d_G = \bar{M}(g, g')$	$d_P = \lvert f(p) - f(p') \rvert$
2.	CGP	Graph Encoding	$d_G = \bar{M}(g, g')$	$d_P = NCD(p, p')$
3.	GE	Grammar Encoding	$d_G = \bar{M}(g, g')$	$d_P = NCD(p, p')$

(or geometrically, varying the angle between $\mathbf{q}$ and $\mathbf{w}$), this models an artificial map with scalable locality (Figure 3.5). The model is intentionally related to maps that have been applied in other combinatorial optimisation problems of basic importance, such as the Knapsack problem. The model is also qualitatively similar to the typical situation found in GA or GP with linear genotypes, where particular genes produce a distinct contribution to the overall phenotype.

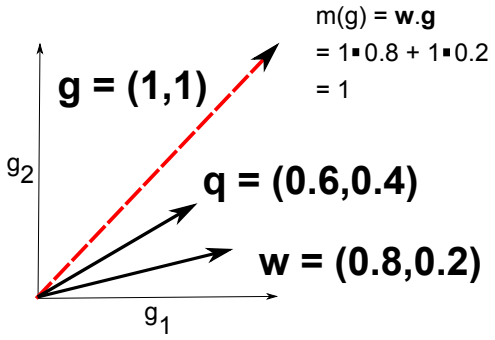

FIGURE 3.5: *Weighted integer model example for $n = 2$. The mapped phenotype $m(\mathbf{g})$ of each integer vector genotype $\mathbf{g}$ is given by the dot-product with the corresponding weight vector $\mathbf{w}$. A non-uniform mutation operator varies each component of $\mathbf{g}$ with the corresponding probability $\mathbf{q}$. Fitness values are directly mapped $f(\mathbf{g}) = m(\mathbf{g}) = \mathbf{w}.\mathbf{g}$*

Consider an instance where $n = 8$ and $k = 10$. Fix a set of positive, linearly decreasing weights $\mathbf{w} = (100, 87.5 \ldots 12.5, 0)$. Now define two vectors of mutation probabilities, $\mathbf{q_A}$ and $\mathbf{q_B}$, each scaling between a minimum value $q_{MIN} = 0.1$ and a maximum $q_{MAX} = 0.2$. For the strongly local case, let the mutation probabilities increase linearly, $\mathbf{q_A} = (0.1, 0.1125 \ldots 0.1875, 0.2)$. Using $\mathbf{q_A}$, components that make a large contribution to the phenotype are mutated with low probability. Conversely, a weakly local map can be introduced, $\mathbf{q_B} = (0.2, 0.1875, \ldots 0.1125, 0.1)$, where the order of mutation probabilities is reversed. These two instances are compared in Figure 3.6.

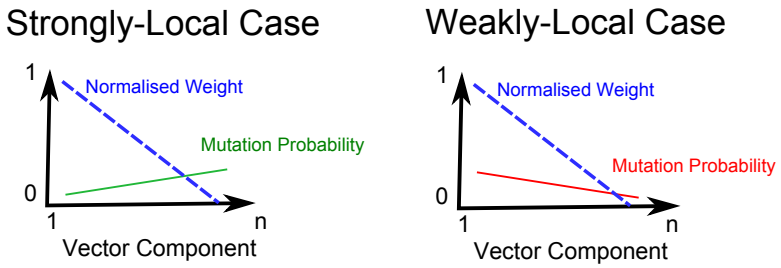

FIGURE 3.6: *Difference in scaling of mutation probabilities over the weighted-integer vector components, which gives rise to the strongly local and weakly local cases.*

Figure 3.7 illustrates the relationship between genotype and phenotype distances for this instance, using sample trees generated under chain referral to a depth of nine mutations. The distance between genotypes is measured under the expected mutation distance, using a logarithmic scale. The distances between phenotypes are taken as the absolute

difference in fitness. The images on the left show this for the strongly local mapping, using $\mathbf{q_A}$. The images on the right show the weakly local mapping, using $\mathbf{q_B}$.

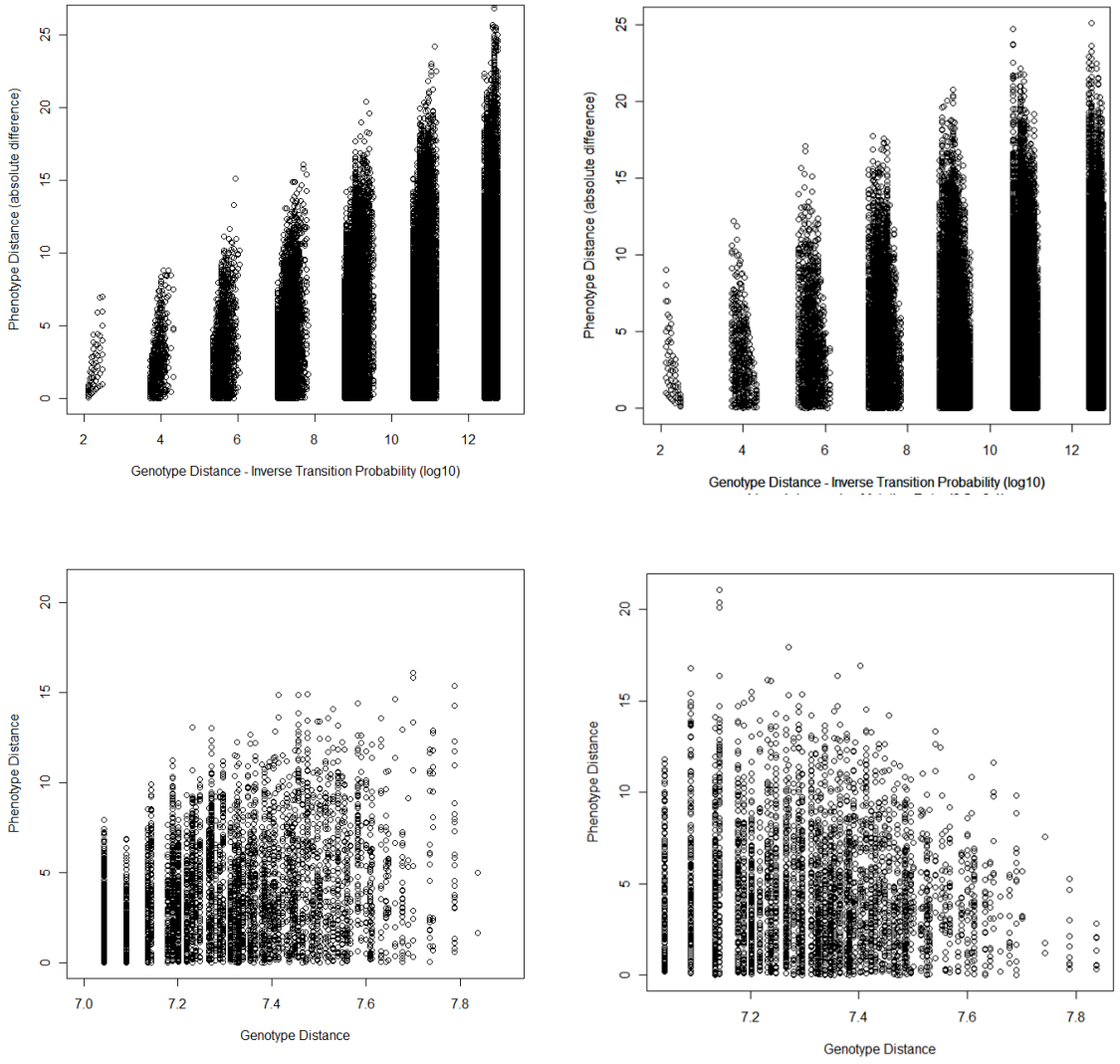

FIGURE 3.7: *Genotype versus phenotype distances on a weighted integer test representation. Top left: A strongly local mapping using* $\mathbf{q_A}$. *Top Right: A weakly local mapping using* $\mathbf{q_B}$. *Bottom left: Close up of an individual band from the strongly local mapping. Bottom right: Close up of an invididual band from the weakly local mapping.*

The top pair of images correspond to the whole range of genotype distances. The bottom pair of images give an example of a band of distances between $7 \leq log_{10}(\bar{M}) \leq 8$. The discrete individual bands in the top pair of images correspond to the $n-1$ possible hamming distances between each pair of genotypes. In general, mutating multiple genes will occur with a low probability and results in a larger net change to the phenotype (large difference in dot product). Therefore, across the whole range, both images show an overall positive correlation between genotype and phenotype. However, focusing in

highlights the difference between the two mappings, which is observable over each band. In each individual band, the relationship between genotype and phenotype distances is positive for the strongly local mapping and negative for the weakly local mapping.

Figure 3.8 contrasts the corresponding Mantel correlation calculated for each distance band. The statistic was derived for 50 sets of samples, shown for random (uniform) sampling and chain-referral (snowball) sampling. Over all bands in the strongly local map, a weakly positive correlation is correctly identified by the Mantel statistic. Over all bands in the weakly local map, a negative correlation is identified, showing similar correlation coefficients for both sampling techniques. Significance calculations were obtained under permutation to test the hypothesis that a correlation exists in each distance interval. The significant results, $p(H_0|r_M) < 0.005$, are labelled with the (*) symbol.

Biased sampling using the chain-referral method increases the number of samples in the first two classes (4.5 - 6.5, 6.5 - 8.0). This provides a sufficient sampling density to give better significance levels than uniform sampling over these distances. The statistic can then be used to discriminate between the strongly local case, where small weights correspond to large mutation probabilities and the weakly local case.

Effect on Performance

Of particular interest is how this change in locality affects the resulting performance of an evolutionary algorithm based on the different genotype to phenotype maps. Consider a basic scenario where the phenotype vectors give a fitness proportional to their length. Figure 3.9 contrasts the corresponding performance under the strongly local and weakly local maps, for a simple 1+1 EA. Performance is measured as the probability of deriving the fittest, maximal length vector $\mathbf{g}$ by 100 generations. The results are shown over a range of settings for the minimum component of each mutation vector (in all instances, $q_{MAX} = q_{MIN} + 0.1$). The weakly local case gives a reduced performance compared to the strongly local case, for all settings of q_{MIN}. However, this reduction is comparatively small (the difference in probability of success is approximately 10-15% at the optimum). Qualitatively, the change is a minor difference in performance and the base choice of value for the mutation parameter dominates the effects of locality. It could be argued that this is in line with our expectations, because the maps have introduced a relatively subtle variation to the observed genotype to phenotype correlation.

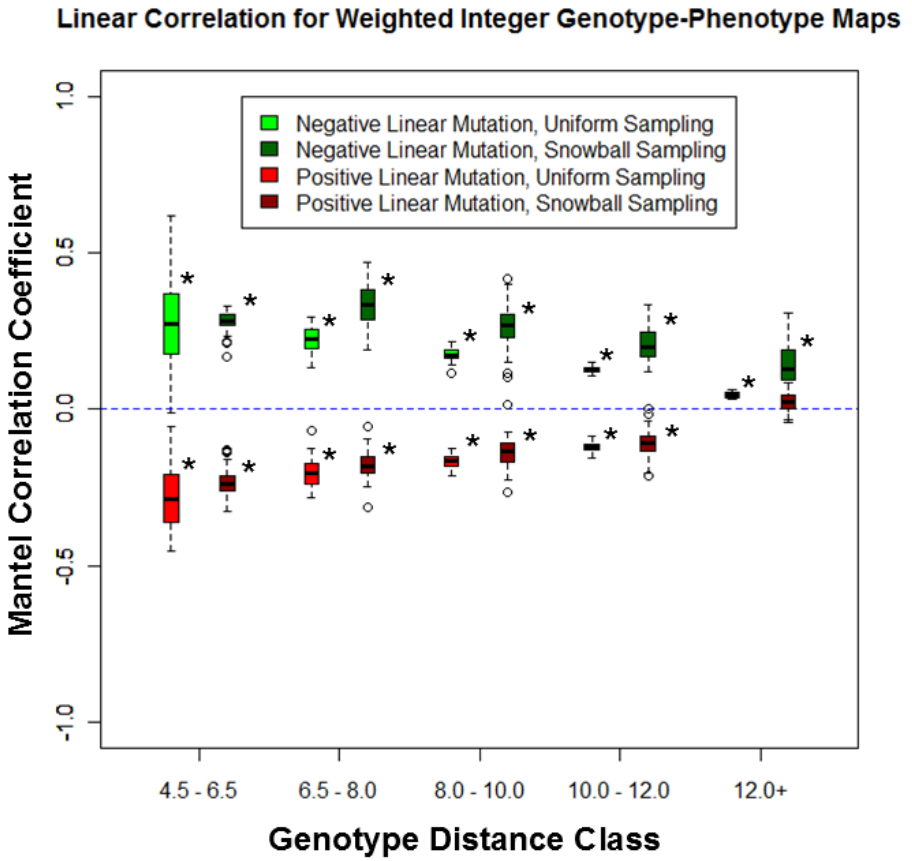

FIGURE 3.8: *The calculated Mantel correlation coefficients in the weighted-integer test case. The symbol (*) denotes a significant correlation under the Mantel test. The correlation coefficient discriminates between the strongly local mapping (negatively scaled mutation rates, positive correlation) and weakly local mapping (positively scaled mutation rates, negative correlation). Results given for uniform and chain-referral sampling.*

Result Summary

The weighted integer genotype to phenotype map provided a very controllable, if artificial, approach to observing the effects of locality. Changes to the locality of the mapping were evident under the Mantel statistic, supporting this method of measurement. We can therefore proceed to more realistic maps with greater confidence.

3.6.2 Experiment 2: Cartesian Genetic Programming

A variable locality mutation operator

In the weighted integer example, controlling locality was straightforward, because the GPM is well-understood. To change the locality of the CGP representation, the mapping is fixed and changes are made to the operator. A technique is used directly based on the

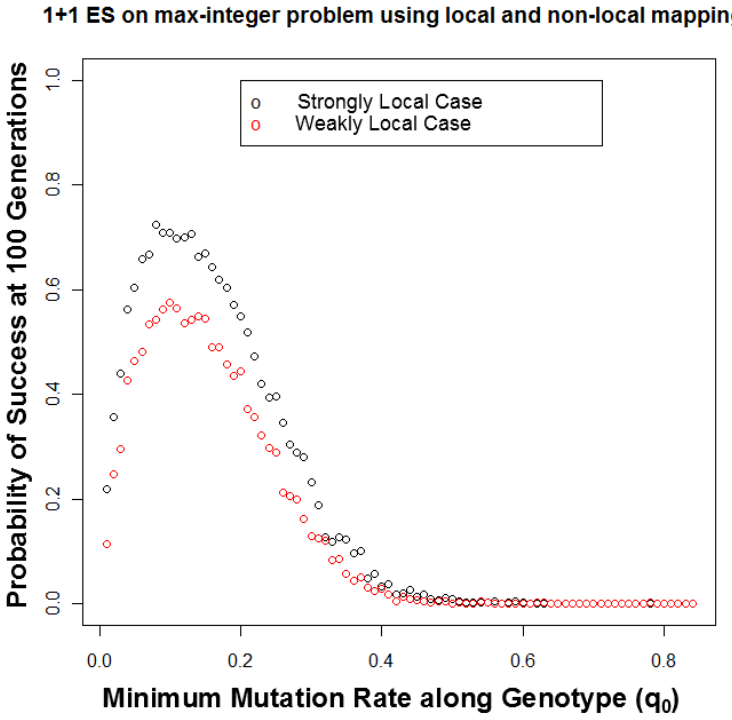

FIGURE 3.9: *Change in performance for local and non-local weighted integer maps.*

approaches of Beadle (2009a,b) and Nguyen *et al.* (2011b) to control syntactic locality. An intermediary bias $u_{\alpha\beta}$ was introduced to the uniform mutation operator:

$$u_{\alpha\beta}(p, p') = \left(1 + e^{-\alpha(d_P(p,p')-\beta)}\right)^{-1} \tag{3.6}$$

The bias $u_{\alpha\beta}$ is a standard sigmoid function and has the effect of changing the expected mutation distance between each pair of genotypes. The parameters α and β specify the slope and offset of the bias respectively such that $\alpha, \beta \in \mathbb{R}$ and $\beta \geq 0$. Pseudocode for this biased operator, `SIGMUTATE`, is provided in Algorithm 1. The function `phenotypeDistance()` gives the distance between two phenotypes under a given phenotype metric. The function `rand()` returns a random number uniformly chosen between zero and one and `mutate(g)` corresponds to a base mutation operator such as uniform or point mutation. `SIGMUTATE` therefore returns a mutated genotype which is biased by the selection probability given by α and β, where $\alpha = 0$ corresponds to a uniform bias. Offspring which are accepted will be close syntactic neighbors of the parent under the phenotype metric when $\alpha \ll 0$ and $\beta \approx 0$. An upper bound on the number of iterations was fixed, $maxCount = 100$.

The advantage of using this method of producing variable locality is that it provides a tunable encoding, whilst also meeting the requirements for the application of the Mantel

test, which can be used to validate this statistically. For each application of the mutation operator to a genotype, $u_{\alpha\beta}$ defines the probability that a proposed set of mutations will be accepted. The process is repeated until an acceptable mutation is found and returned by the operator or *maxCount* is exceeded. To first order, this gives an adjusted expected variation distance of

$$\bar{M}_{\alpha\beta} \approx u_{\alpha\beta}^{-1} \times \bar{M} \tag{3.7}$$

which scales the locality of the mapping.

Application to CGP

It was contended in the previous section that the Weighted Integer genotype model provides a good approximation to the scaling (Section 2.7.2) that is present in complex GP encodings, such as Cartesian Genetic Programming, by controlling the contribution made by each element of the genotype. Recall the standard CGP encoding, described in Chapter 2. The encoding between the integer array genotype and program relates closely to the simpler weighted integer map. A variable contribution is made to the evaluated program by each integer, by changing the corresponding graph connectivity, or function choice.

There is no priori knowledge of which syntactic distance measures are most appropriate to describe differences between programs in CGP. To compute the Mantel statistic requires a phenotype metric which is symmetric because of the requirement to obtain pair-wise distances (Legendre and Legendre, 1998). Measurements of the syntactic distance between CGP phenotypes (d_P) were derived using the *Normalised Compression*

```
Data: Genotype g, α, β
Result: Mutated genotype g'
count ← 0;
found ← false;
g' ← g;
while ¬ found ∧ c < maxCount do
    g' ← mutate(g);
    d_P ← phenotypeDistance(m(g), m(g'));
    acceptChance← sigmoid(d, α, β);
    if rand() < acceptChance then
        found ← true;
    end
    count ← count+1;
end
return g';
```

Algorithm 1: `SIGMUTATE`

Distance (NCD) (Equation 3.8). The theoretical basis of the NCD is the similarity in information contents between each object. Thus, the metric is a justifiable starting approach because it makes no assumptions regarding the importance of different syntactic features in each program. These syntactic distances were obtained by the same procedure used in the study of locality in tree-based GP by Galvan-Lopez *et al.* (2011a), such that

$$d_P(p, p') = \frac{C(pp') - min(C(p), C(p'))}{max(C(p), C(p')} \tag{3.8}$$

where C is a function giving the length in bits of the string representation (assuming UTF8) of the argument for a particular compressor. Each phenotype was decoded into the prefix string representing the corresponding encoded arithmetic expression. This expression excludes neutral nodes (junk) that do not contribute to the phenotype. Pairwise application of the NCD gives a measure of similarity between phenotypes in the range of $\{0.0 : 1.0\} + \epsilon$, using the `gzip` algorithm (where $\epsilon \approx 0.1$, an error term introduced because the compression is not ideal). For genotypes that map to very similar string representations of that program, $d_P(p, p') \to 0$. Conversely, in phenotypes that are proportionally more distinct and share fewer common schema, $d_P(p, p') \to 1$.

Fifty biarity tree samples were obtained from a representative CGP genotype space using an arithmetic function set, including the protected division operator, using chain-referral sampling. Table 3.2 summarises the parameters used to initialise each genotype. Sample biarity trees were produced recursively under the `SIGMUTATE` operator to a depth of 7 mutations, containing 511 genotypes. The expected variation distance $\bar{M}$ was obtained pair-wise for all members of each sample. This method provided a set of 50 matrices each containing ∼150000 genotype distances.

The threshold value assumed in the sigmoid function was set to an intermediate level of similarity under the NCD, $\beta = 0.2$, shown in Figure 3.10. An example plot (for one CGP sample) is given in Figure 3.11. The graphs are scatterplots, binned into hexagons, illustrating qualitatively the distribution observed between the log-scaled expected variation distance $\bar{M}$ and normalised compression distance d_P. The result is shown between two similar maps at low locality ($\alpha = 20, 10$) and at high locality ($\alpha = -10, -20$). This directly compares the change in locality induced by the bias. Inspecting the scat-

TABLE 3.2: *CGP search parameters.*

Representation	CGP	Crossover	No
Nodes	10	Selection Strategy	$(\mu + \lambda) = (4 + 6)$
Structure	Single row feed-forward	Population Size	10
Function Set	{+,-,*, ÷}	Fitness Samples	$10 \in \{-2 : 2\}$
Terminal Set	{0,1}	Max Generations	2000
Mutation Rate	0.15	Runs	500

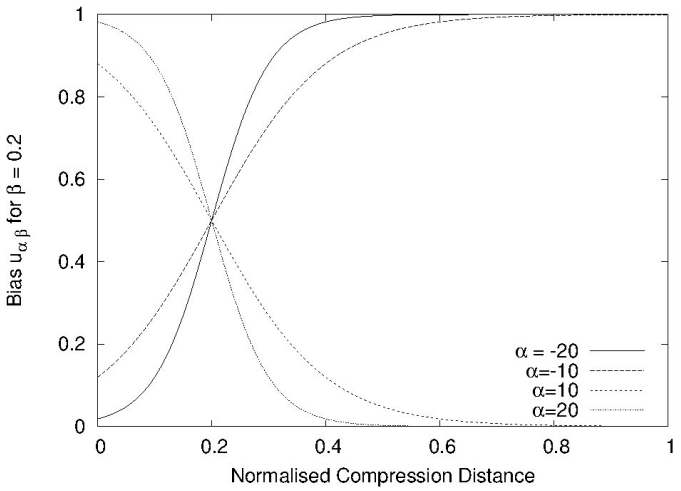

FIGURE 3.10: *Probability of accepting a new offspring via. uniform mutation operator.*

ter graphs appears to indicate a weakly positive trend, apparent over short distances. This follows from the decreasing likelihood of making larger syntactic changes to the phenotype, under the uniform mutation operator. Relatively probable mutations, $\bar{M} \sim$ [0 - 20] correspond to smaller changes in compression distance, $d_P \sim [0.1 - 0.3]$. The majority of distances observed in the region $\bar{M} \sim$ [20 - 40] (between genotypes on lower branches of the sample) occur with lower probability and correspond to greater variation in syntactic change.

Using the Mantel test, we can validate these qualitative observations. Figure 3.12 shows the range of corresponding Mantel coefficients r_M calculated over all samples, for linear correlation, as a function of distance. It can be inferred that an overall weak positive correlation exists in the CGP mapping, which falls off as a function of genotype distance. A set of 1000 permutations was then generated for each distance matrix to test significance at $P(H_0) < 0.005$, for a set of 8 distance classes from $log_{10}(\bar{M}) = 0.0 : 40.0$. The correlations found to be significant under permutation are labelled (*). Inclusion of the Mantel test therefore gives a firm basis from which to reject the null hypothesis and accept the correlation. The effects of the mutation bias are also apparent (contrast positive α with negative α).

Symbolic Regression Performance

To explore the relationship between syntactic locality and performance in CGP, a randomly generated selection of 40 symbolic regression problems was produced. Symbolic regression is an extensively employed benchmarking tool in GP, which has provided notably successful applications. Although the use of symbolic regression as a benchmark has been criticised in recent years problems from this class have nonetheless remained

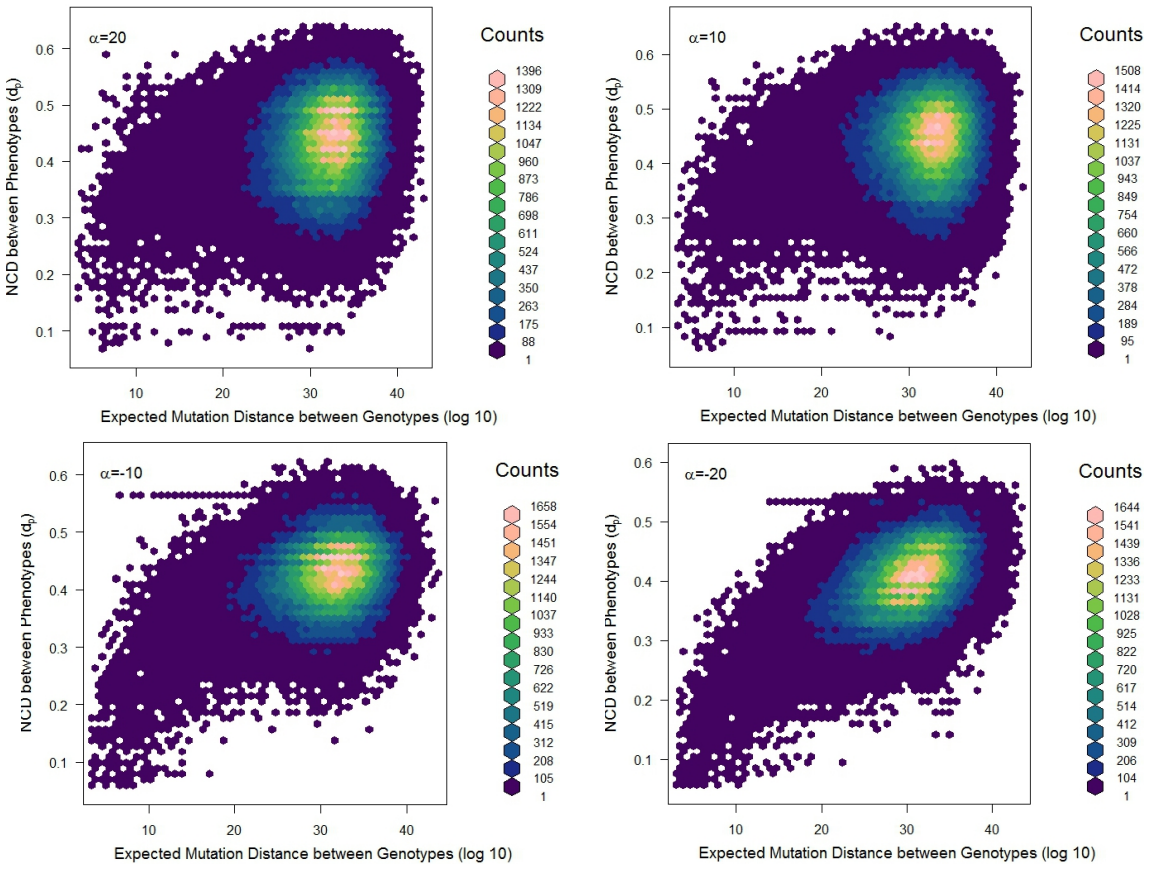

FIGURE 3.11: *Illustration of the effect of the NCD mutation bias on genotype to phenotype correlation. Top left: Lowest locality. Bottom right: Highest locality*

staples for initial tests of new GP systems and methods (McDermott *et al.*, 2012). The objective of a symbolic regression problem is to fit a model expression M that is as close as possible to a hidden true function x, given a set of N samples points on x. A standard approach in GP is to minimise the Euclidean norm d_2 between the known samples of the true function and the output of the model:

$$d_2 = \sqrt[2]{\Sigma_{i=0}^{N}(x(i) - M(i))^2} \tag{3.9}$$

Samples are obtained at uniform or randomly generated intervals. In this work, unless otherwise stated uniform sampling over 20 points will be employed. The quality of models will be tested over 100 unseen points against the true function.

The problem instances considered here were restricted in complexity to simple 5th order polynomials with integer coefficients in the range of {-2:2}. These are basic problems known to be solvable consistently using only the standard CGP representation. Five instances of each problem were considered, applying the mutation bias with $\alpha = \{-20, -10, 0, 10, 20\}$ and $\beta = 0.2$. Fitness was evaluated by deriving the euclidean distance over the set of uniformly distributed sample points. Other parameters (Table

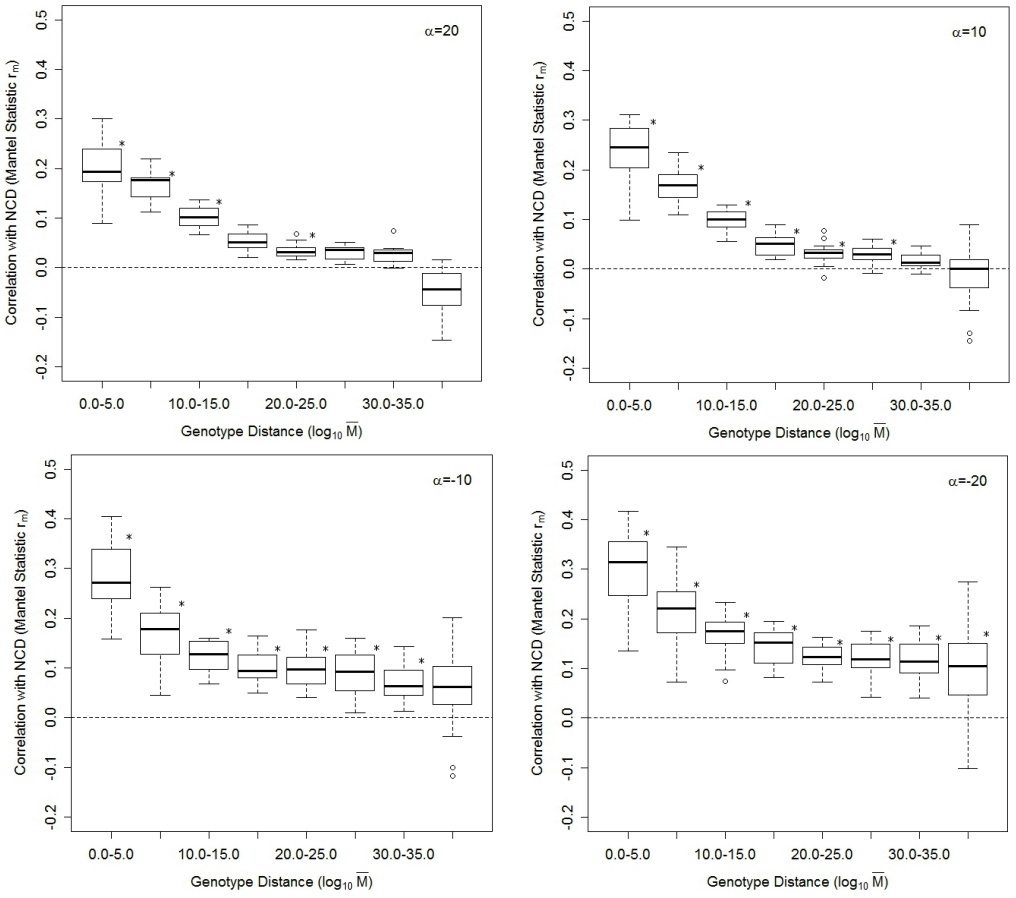

FIGURE 3.12: *Measured Mantel correlation for CGP with different levels of mutation bias. Top left: Lowest locality. Bottom right: Highest locality*

3.2) were informed per common previous estimates in CGP (Miller, 2011). These parameters have not been optimised to account for interaction with the mutation bias, therefore comparison can only be made in a relative fashion.

Tables 3.3 and 3.4 at the end of this section detail the corresponding probability of success η at each locality level after 2000 generations, estimated in each instance from the fraction of $k = 500$ runs that successfully recovered the expression. The measured value of η is binomially distributed. Following the approach of Walker *et al.* (2007), confidence intervals were assigned to each measurement based on Wilson's score method (Newcombe, 1998):

$$\frac{2k\eta + z_{norm}^2 \pm z_{norm}\sqrt{z_{norm}^2 + 4k\eta(1-\eta)}}{2(k + z_{norm}^2)} \tag{3.10}$$

where $z_{norm} = 1.96$ is the conventional normal score for 95% confidence. For the 23 polynomial problems with an average success probability $\bar{\eta}$ greater than 10% (denoted with a †), a general tendency can be observed towards better performance at lower levels

of locality. Of this subset, shown in Figure 3.13, in 21 of the 23 cases the probability of success was higher for $\alpha = 20$ than $\alpha = -20$. In the remaining cases with success probability below 10%, no measurable trend is observable outside of experimental error. All problems were solved successfully over at least one set of runs.

Result Summary

The weak correlation observed between genotype and phenotype distances in Figures 3.11 and 3.12 is consistent with the variation in structure that small mutations can impose in this representation. Altering a single node connection in CGP may cause a large number of functions to be disconnected. Similarly, if the same node is connected to many neighbours, then adjusting it will tend to produce a disproportionate change to the syntax of a program. Because the mapping can support many neutral transitions, there is a tendency towards parents producing syntactically similar offspring, but this is offset by the potential for large structural change. The relatively small impact of the α scaling parameter suggest this bias is difficult to adjust, given that it is a direct consequence of using a graph-based structure.

The trend of the symbolic regression results under truncation selection implies that higher levels of correlation between genotype and phenotype distance led to poorer performance in CGP. We can consider three feasible explanations. Firstly, it is likely that the constraints imposed by high locality have restricted the diversity of the search, which may render intermediary schema difficult to reach (Pham *et al.*, 2013). Secondly, in Rothlauf's model of locality, poorer performance under higher locality can be associated with fitness landscapes which are misleading or deceptive (such as GA 'trap' functions) (Rothlauf, 2006). Further investigation of the fitness landscapes for these specific problem instances would be required to determine whether this is the case for this representation. Thirdly, it is unclear how features of the CGP genotype to phenotype map not addressed here, such as high redundancy, or structural bias (Payne and Stepney, 2009) contribute to the trend. To explore this counter-intuitive result, a comparison was made by running an identical experiment using a conventional tournament selection strategy. Tournament selection was carried out in a binary fashion, choosing parents through comparison between pairs of genotypes drawn uniformly from the population. The highest fitness individual in each tournament was always selected and the best individual between each generation was retained. In principle, because of the random element in selecting competitors, the tournament selection strategy provides a weaker selection pressure.

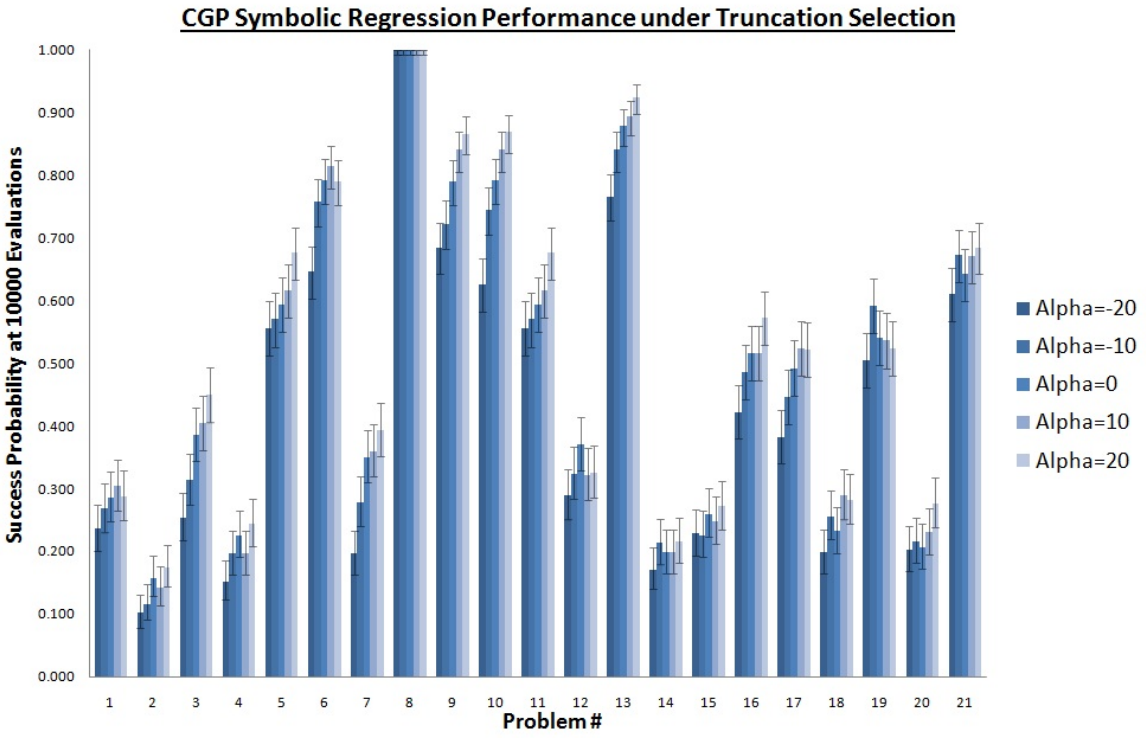

FIGURE 3.13: *CGP performance on Symbolic Regression problem set under Truncation selection. Measured at bias* $\alpha = \{-20 : 20\}$

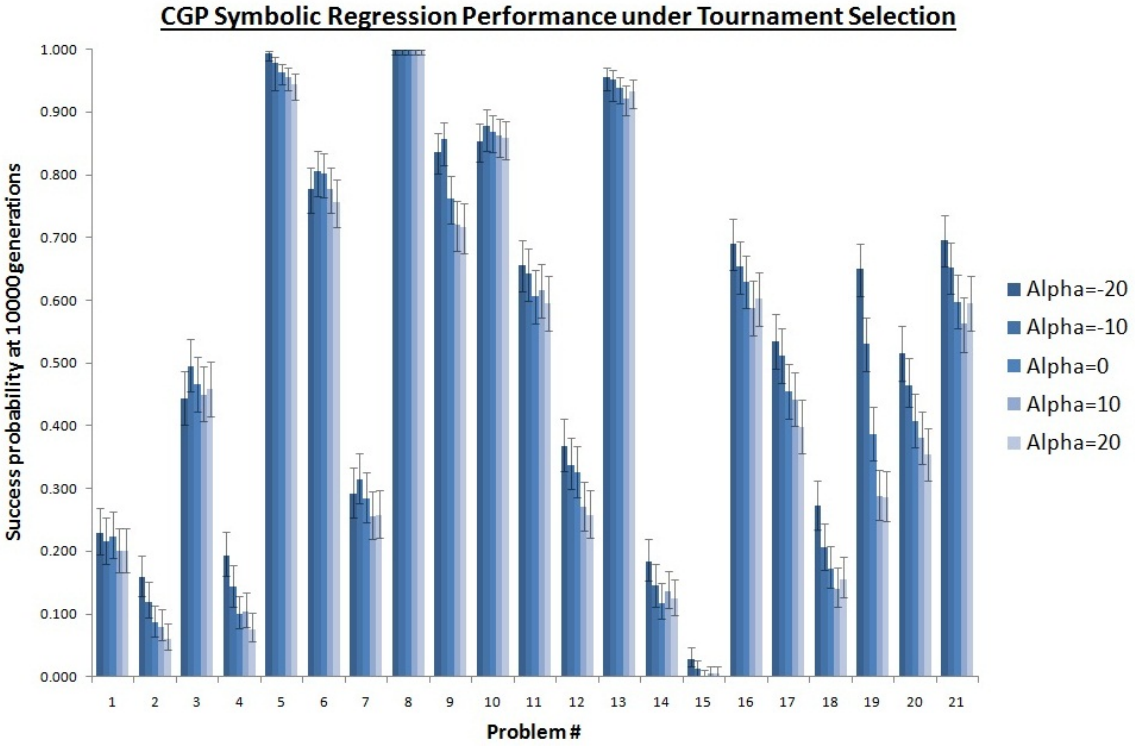

FIGURE 3.14: *CGP performance on Symbolic Regression problem set under Tournament selection. Measured at bias* $\alpha = \{-20 : 20\}$

TABLE 3.3: *CGP success probability η with respect to locality, using truncation selection.*

Polynomial Expression	α					$\bar{\eta} \geq 0.1$?
	-20	-10	0	10	20	
$-1 - 2x^3 + x^4$	0.236	0.268	0.286	0.304	0.288	†
$-2 - 2x - x^2 - x^4$	0.102	0.116	0.158	0.142	0.174	†
$-2 - x^3$	0.254	0.314	0.386	0.404	0.450	†
$-2x^2 + 2x^4 + 2x^5$	0.152	0.196	0.226	0.196	0.244	†
$-2x^2 - 2x^3$	0.556	0.570	0.594	0.616	0.676	†
$-2x^3 + x^4$	0.646	0.758	0.792	0.814	0.790	†
$-2x^3 + x^5$	0.196	0.278	0.350	0.360	0.394	†
$-x$	1.00000	1.00000	1.00000	1.00000	1.00000	†
$-x^2 + x^3 + 2x^4$	0.684	0.722	0.790	0.840	0.866	†
$-x - x^4$	0.626	0.744	0.792	0.840	0.868	†
$1 - 2x^2 - 2x^3$	0.556	0.570	0.594	0.616	0.676	†
$1 - x + 2x^3 + x^4$	0.290	0.324	0.370	0.322	0.326	†
$1 - x + x^2 - x^3$	0.766	0.840	0.878	0.894	0.924	†
$1 + x^3 + 2x^5$	0.170	0.214	0.198	0.198	0.216	†
$2 + 2x - x^2 - 2x^3$	0.228	0.226	0.260	0.248	0.272	†
$2 + x^2 - x^3$	0.422	0.486	0.516	0.516	0.572	†
$2x^2 - x^3 + 2x^4$	0.382	0.446	0.492	0.524	0.522	†
$2x^2 - x^3 + x^4 - x^5$	0.198	0.256	0.232	0.290	0.282	†
$2x^2 + x^3 - 2x^4 - x^5$	0.504	0.592	0.540	0.536	0.524	†
$2x + x^2 - 2x^3$	0.202	0.216	0.206	0.230	0.276	†
$x + 2x^2 - 2x^3$	0.610	0.672	0.642	0.670	0.684	†
$x + 2x^2 - 2x^3$	0.610	0.672	0.642	0.670	0.684	†
$-2 - 2x^2 - 2x^5$	0.034	0.042	0.038	0.014	0.028	-
$-x + 2x + x^4 - 2x^5$	0.122	0.122	0.112	0.110	0.116	†
$-2 + 2x - x^2 - 2x^3 + 2x^5$	0.002	0.006	0.000	0.000	0.000	-
$-2 + x - x^2 - 2x^4 - 2x^5$	0.001	0.014	0.004	0.004	0.040	-
$-2x^5 - 2x^3 - 2x^2 - 2x - 1$	0.016	0.016	0.004	0.040	0.008	-
$-2x + 2x^3 + 2x^4 + 2x^5$	0.064	0.050	0.040	0.048	0.040	-
$-2x + 2x^3 + 2x^4 + 2x^5$	0.064	0.050	0.040	0.048	0.040	-
$-2x + 2x^3 + x^4 - 2x^5$	0.006	0.020	0.006	0.006	0.006	-
$-2x + 2x - 2x^2 - x^4 - 2x^5$	0.002	0.002	0.000	0.000	0.002	-
$-2x + x^2 - 2x^3 + 2x^4 - 2x^5$	0.006	0.010	0.004	0.006	0.008	-
$1 - 2x + x^2 - x^3 - x^5$	0.120	0.098	0.092	0.084	0.046	-
$1 - x^3 + 2x^4 - 2x^5$	0.046	0.036	0.030	0.038	0.022	-
$1 + 2x^2 - x^3 + 2x^4 - 2x^5$	0.020	0.044	0.024	0.018	0.006	-
$1 + 2x - x^2 + 2x^5$	0.106	0.112	0.092	0.086	0.082	-
$2 - 1x - 2x^2 + 2x^3 - x^4 - x^5$	0.008	0.002	0.002	0.002	0.002	-
$2 + 2x^2 + x^3 - 2x^4 - x^5$	0.070	0.068	0.050	0.048	0.042	-
$2 + x - 2x^2 - 2x^3 + x^5$	0.038	0.034	0.030	0.030	0.042	-
$2 + x - x^2 - x^4 - x^5$	0.052	0.024	0.032	0.032	0.028	-
$2 + x - x^2 + x^4 - x^5$	0.052	0.024	0.032	0.032	0.028	-

TABLE 3.4: *CGP success probability η with respect to locality, using tournament selection.*

Polynomial Expression	α					$\bar{\eta} \geq 0.1$?
	-20	-10	0	10	20	
$-1 - 2x^3 + x^4$	0.230	0.216	0.224	0.200	0.200	†
$-2 - 2x - x^2 - x^4$	0.158	0.120	0.086	0.080	0.060	†
$-2 - x^3$	0.444	0.494	0.466	0.450	0.458	†
$-2x^2 + 2x^4 + 2x^5$	0.194	0.144	0.100	0.104	0.076	†
$-2x^2 - 2x^3$	0.994	0.978	0.964	0.956	0.944	†
$-2x^3 + x^4$	0.777	0.806	0.802	0.778	0.756	†
$-2x^3 + x^5$	0.292	0.314	0.284	0.256	0.258	†
$-x$	1.00000	1.00000	1.00000	1.00000	1.00000	†
$-x^2 + x^3 + 2x^4$	0.836	0.856	0.762	0.720	0.716	†
$-x - x^4$	0.854	0.878	0.868	0.862	0.858	†
$1 - 2x^2 - 2x^3$	0.656	0.642	0.606	0.616	0.596	†
$1 - x + 2x^3 + x^4$	0.368	0.338	0.326	0.270	0.258	†
$1 - x + x^2 - x^3$	0.956	0.952	0.938	0.922	0.932	†
$1 + x^3 + 2x^5$	0.184	0.146	0.118	0.136	0.124	†
$2 + 2x - x^2 - 2x^3$	0.028	0.012	0.002	0.006	0.006	-
$2 + x^2 - x^3$	0.690	0.654	0.630	0.588	0.602	†
$2x^2 - x^3 + 2x^4$	0.534	0.512	0.454	0.442	0.398	†
$2x^2 - x^3 + x^4 - x^5$	0.272	0.206	0.172	0.140	0.156	†
$2x^2 + x^3 - 2x^4 - x^5$	0.650	0.530	0.386	0.288	0.286	†
$2x + x^2 - 2x^3$	0.516	0.464	0.408	0.380	0.354	†
$x + 2x^2 - 2x^3$	0.696	0.652	0.598	0.562	0.596	†
$-2 - 2x^2 - 2x^5$	0.020	0.018	0.008	0.002	0.000	-

Figure 3.14 summarises the results over the first 21 problems. A qualitative similarity is apparent in the success on individual problems, a general trend with respect to locality is also evident - in the opposite direction to that shown in Figure 3.14. Under the relaxed selection pressure increases in locality provide a small improvement in performance. The result highlights an apparent trade-off between selection and variation for this problem set. Reducing the locality of the mapping increases the likelihood of random variation in phenotype distances, which in turn increases variation in the population as a whole. Conversely, a higher selection pressure counter-acts this by selecting fitter individuals more greedily. This suggests that to provide better performance for this representation over this problem set, it is necessary to find a balance between the locality of the mapping and strength of the selection procedure.

3.6.3 Experiment 3: Grammatical Evolution

The approach adopted in the previous section can in theory be extended to any GP representation for which a satisfactory distance metric can be defined on the genotype and phenotype space and genotypes are mutually reachable. Grammatical Evolution, as discussed in Chapter 2, is a well-established GP technique using an indirect mapping with a large body of existing analysis. Previous work has indicated that simple forms of

```
<expr> ::=   <expr> <op> <expr>
           | ( <expr> <op> <expr> )
           | <var>
<op>   ::=   +
           | -
           | *
           | %
<var>  ::= x
```

FIGURE 3.15: *Simple arithmetic function BNF grammar*

GE can exhibit weak locality (Rothlauf and Oetzel, 2006; Byrne *et al.*, 2010; Hugosson *et al.*, 2010). However, it should be emphasised that the intention of this experiment is not to provide a thorough, direct comparison between GE and Cartesian Genetic Programming or apply the most recent innovations in both methods. To do so would require a comprehensive, full-factorial analysis of all parameters in both representations, over a more diverse problem and operator set. From an engineering standpoint, such an analysis would quickly be rendered obsolete, because both representations have been refined significantly since their original inception. Therefore the selected parameters (detailed in Table 3.5) are notional only. The purpose of this experiment is to demonstrate the generality of the Mantel statistic as a method of measuring locality, by applying it to another distinct GP paradigm.

TABLE 3.5: *Grammatical Evolution search parameters.*

Representation	Grammatical Evolution	Crossover	No
Length	45	Selection Strategy	$(\mu + \lambda) = (50 + 450)$
Structure	Arithmetic BNF Grammar	Population Size	500
Function Set	{+,-,*, ÷}	Fitness Samples	$20 \in \{-2 : 2\}$
Terminal Set	{x}	Max Generations	100
Mutation Rate	0.10	Runs	500

The configuration used here is based on seminal work in GE and uses a fixed length vector of 45 integers in {0,1,2,3}, initialised uniformly at random. The BNF grammar used is given in Figure 3.15, where % is the protected division operator, which is a more constrained version of an example grammar from Ryan *et al.* (1998), including only the basic arithmetic function set. Expression tree size was constrained to a maximum depth of 10 nodes. Each tree was derived by sequentially parsing elements along the integer vector, wrapping around the vector if further decisions were required, as described in Section 2.5.3. A limit of at most one wrap around the vector was permitted (only terminal values were selectable after that point).

Changes in performance over the symbolic regression problem set are presented in Figures 3.16 and 3.17. The results are summarised in Tables 3.6 and 3.7. Note that the

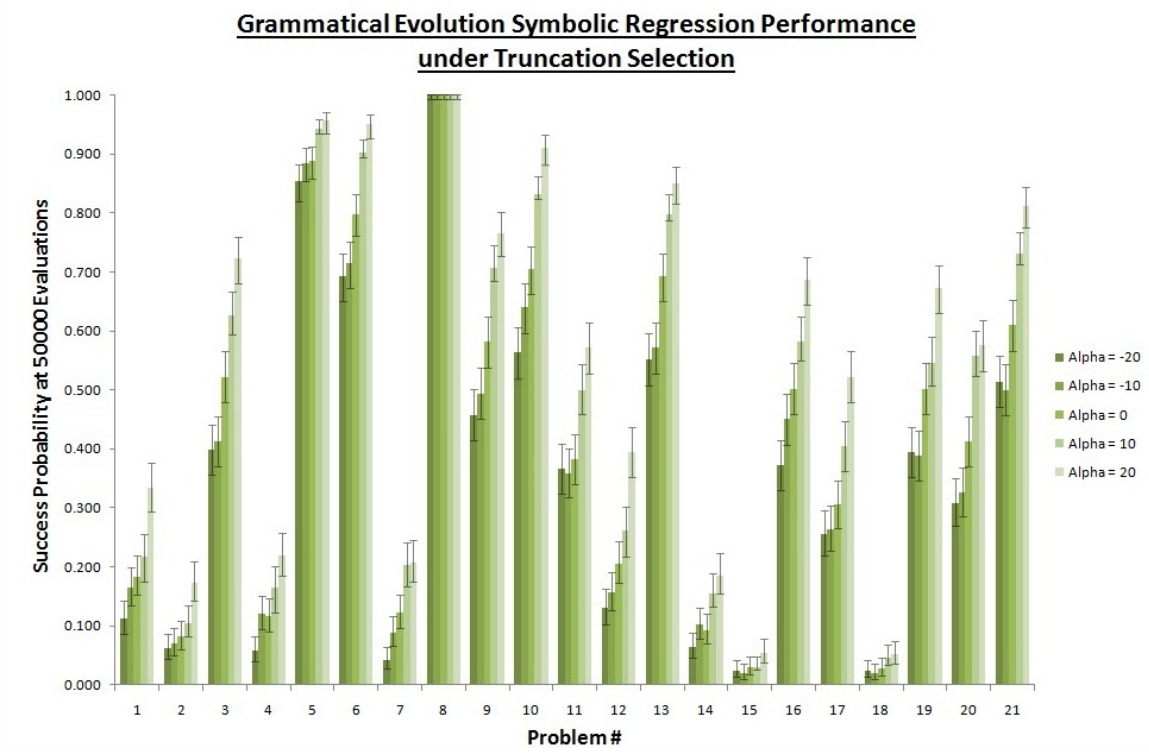

FIGURE 3.16: *Grammatical Evolution performance on Symbolic Regression problem set under Truncation selection. Measured at bias* $\alpha = \{-20 : 20\}$

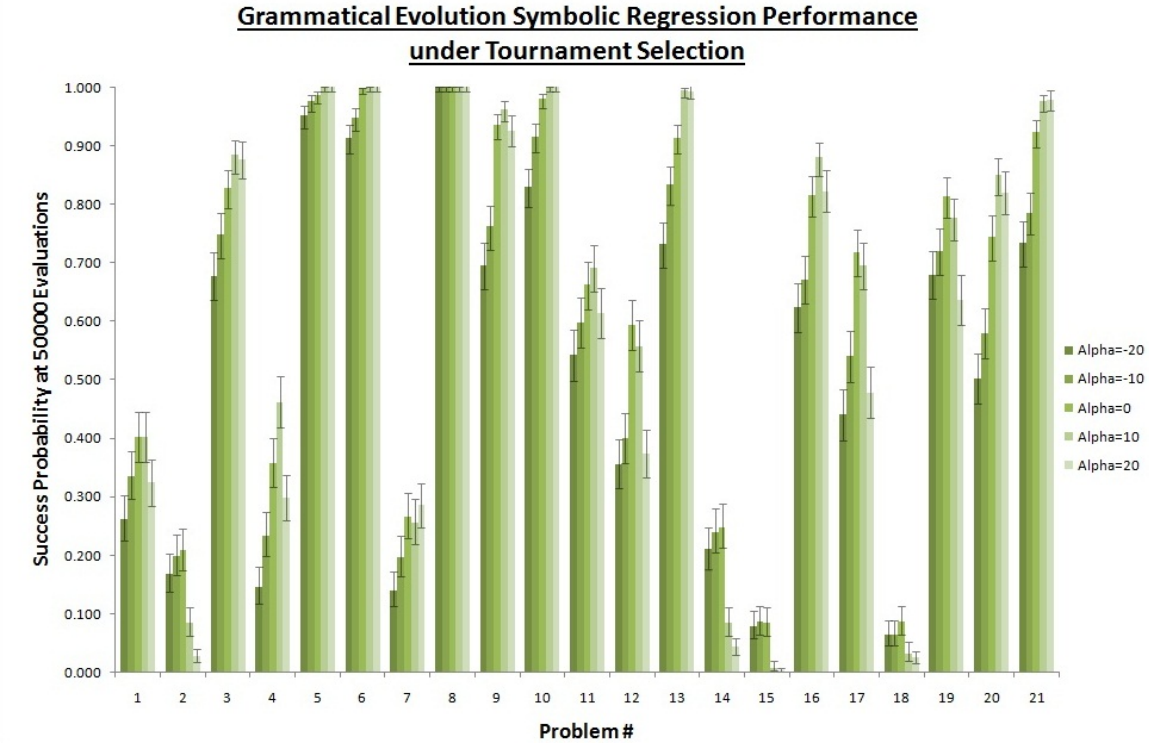

FIGURE 3.17: *Grammatical Evolution performance on Symbolic Regression problem set under Tournament selection. Measured at bias* $\alpha = \{-20 : 20\}$

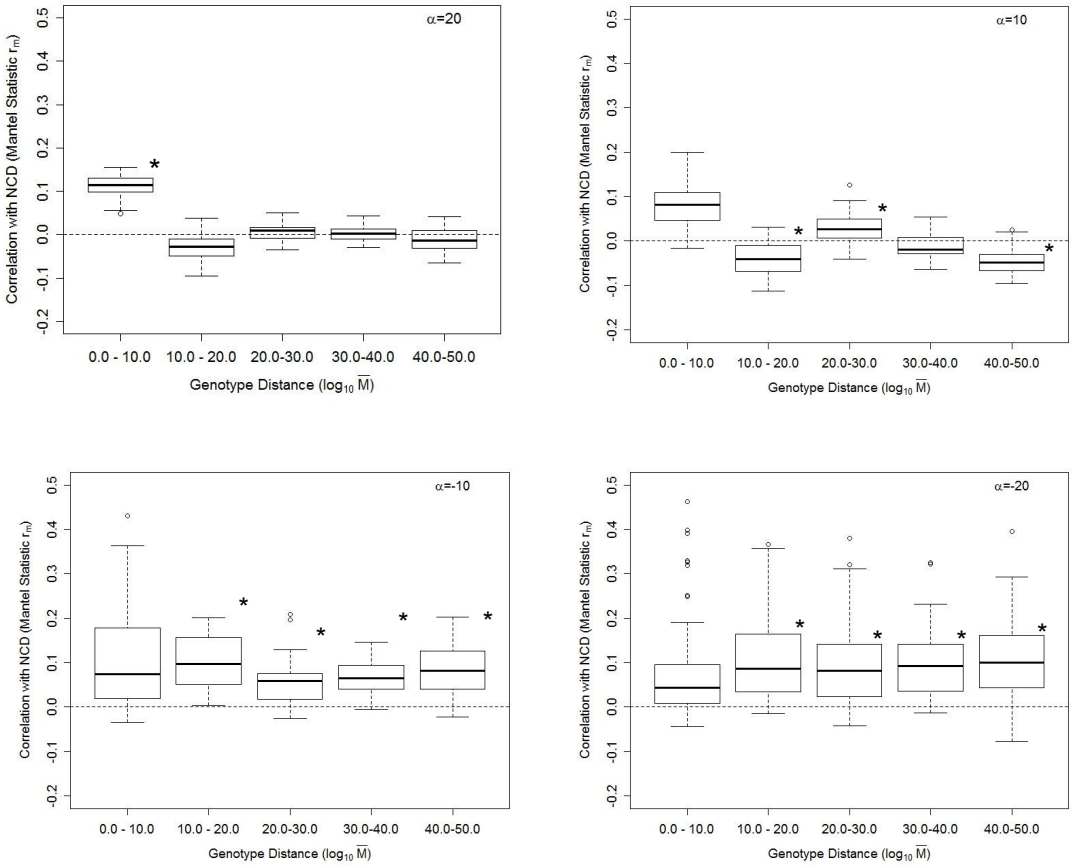

FIGURE 3.18: *Measured Mantel correlation for Grammatical Evolution with different levels of mutation bias. Top left: Lowest locality. Bottom right: Highest locality*

problem set was downsized based on those randomly generated instances which were useful in the CGP experiment (i.e. those which were solved with probabilities outside of experimental error, the first 21 instances). Figure 3.18 shows the corresponding Mantel statistics computed for each value of α.

Result Summary

Despite using a significantly different representation, under truncation selection a similar response to changes in syntactic locality is apparent for the GE case. Increasing the locality of the representation reduced performance across all of the 21 instances. The effects of the mutation bias on the calculated Mantel statistic are also consistent with those observed in the CGP representation. However, the response evident in CGP when switching between truncation and tournament selection was not observed in Grammatical Evolution.

TABLE 3.6: *GE success probability η with respect to locality, using truncation selection.*

Polynomial Expression	α					$\bar{\eta} \geq 0.1$?
	-20	-10	0	10	20	
$-1 - 2x^3 + x^4$	0.112	0.164	0.184	0.218	0.334	†
$-2 - 2x - x^2 - x^4$	0.062	0.070	0.082	0.104	0.174	†
$-2 - x^3$	0.398	0.412	0.522	0.626	0.722	†
$-2x^2 + 2x^4 + 2x^5$	0.058	0.120	0.116	0.166	0.220	†
$-2x^2 - 2x^3$	0.854	0.884	0.888	0.942	0.956	†
$-2x^3 + x^4$	0.692	0.714	0.798	0.902	0.950	†
$-2x^3 + x^5$	0.042	0.088	0.122	0.204	0.208	†
$-x$	1.000	1.000	1.000	1.000	1.000	†
$-x^2 + x^3 + 2x^4$	0.458	0.494	0.582	0.706	0.766	†
$-x - x^4$	0.564	0.640	0.704	0.832	0.910	†
$1 - 2x^2 - 2x^3$	0.366	0.358	0.382	0.500	0.572	†
$1 - x + 2x^3 + x^4$	0.130	0.156	0.206	0.262	0.394	†
$1 - x + x^2 - x^3$	0.552	0.572	0.692	0.798	0.850	†
$1 + x^3 + 2x^5$	0.064	0.102	0.092	0.154	0.186	†
$2 + 2x - x^2 - 2x^3$	0.02	0.02	0.03	0.03	0.054	†
$2 + x^2 - x^3$	0.372	0.450	0.502	0.582	0.686	†
$2x^2 - x^3 + 2x^4$	0.256	0.264	0.305	0.404	0.522	†
$2x^2 - x^3 + x^4 - x^5$	0.024	0.020	0.028	0.046	0.052	†
$2x^2 + x^3 - 2x^4 - x^5$	0.394	0.388	0.502	0.546	0.672	†
$2x + x^2 - 2x^3$	0.308	0.326	0.412	0.558	0.576	†
$x + 2x^2 - 2x^3$	0.514	0.500	0.610	0.730	0.812	†

TABLE 3.7: *GE success probability η with respect to locality, using tournament selection.*

Polynomial Expression	α					$\bar{\eta} \geq 0.1$?
	-20	-10	0	10	20	
$-1 - 2x^3 + x^4$	0.262	0.336	0.402	0.402	0.324	†
$-2 - 2x - x^2 - x^4$	0.168	0.198	0.208	0.084	0.028	†
$-2 - x^3$	0.678	0.748	0.828	0.884	0.876	†
$-2x^2 + 2x^4 + 2x^5$	0.146	0.234	0.358	0.462	0.298	†
$-2x^2 - 2x^3$	0.952	0.976	0.986	1.000	1.000	†
$-2x^3 + x^4$	0.914	0.948	0.998	1.000	1.000	†
$-2x^3 + x^5$	0.140	0.196	0.266	0.256	0.286	†
$-x$	1.000	1.000	1.000	1.000	1.000	†
$-x^2 + x^3 + 2x^4$	0.696	0.762	0.936	0.962	0.926	†
$-x - x^4$	0.830	0.916	0.980	1.000	1.000	†
$1 - 2x^2 - 2x^3$	0.542	0.598	0.662	0.692	0.614	†
$1 - x + 2x^3 + x^4$	0.356	0.400	0.594	0.558	0.374	†
$1 - x + x^2 - x^3$	0.732	0.834	0.914	0.994	0.992	†
$1 + x^3 + 2x^5$	0.210	0.240	0.248	0.084	0.044	†
$2 + 2x - x^2 - 2x^3$	0.078	0.086	0.084	0.008	0.004	†
$2 + x^2 - x^3$	0.624	0.672	0.816	0.880	0.822	†
$2x^2 - x^3 + 2x^4$	0.440	0.540	0.718	0.696	0.478	†
$2x^2 - x^3 + x^4 - x^5$	0.064	0.064	0.086	0.032	0.026	†
$2x^2 + x^3 - 2x^4 - x^5$	0.680	0.720	0.814	0.776	0.636	†
$2x + x^2 - 2x^3$	0.502	0.580	0.744	0.850	0.820	†
$x + 2x^2 - 2x^3$	0.734	0.786	0.924	0.976	0.978	†

3.7 Discussion and Concluding Remarks

The occurrence of a similar, weak correlation between genotype and phenotype distances in both CGP and Grammatical Evolution is an interesting consequence of general properties shared by both mappings. Application of the `SIGMUTATE` operator generated maps with comparable syntactic locality when measured under the Mantel statistic. Furthermore, despite only a superficially similar choice of parameters, both representations demonstrated a similar response to increases in locality under truncation selection. This is suggestive about the relationship between selection pressure and mapping locality.

One possible explanation for the different response to switching to tournament selection between the two populations is the relative change in selection pressure. In the GE case, the larger population size implies a lower initial selection pressure than the CGP case. This may contribute to the observation that no significant change occurs when relaxing the selection pressure further. A more detailed study of the relationship between selection pressure, locality and performance would be required to determine this.

Overall, the results appear to support the intuition that the good performance of both mappings can be in part attributed to an intermediate balance between exploitation and exploration in the program space. However, the relationship between locality and other GPM properties theorised to influence performance (neutral networks, capacity for large structural change, modality of the search space) has not been assessed in this analysis. In addition, it remains open what contribution is made to performance by mutations that take place over larger genotype distances. The probability of mutations occurring between particular genotypes at large $\bar{M}$ is very small for all the mappings considered here. This is partly offset by the combinatorial increase in the number of possible transitions, as can be observed in Figures 3.11.

From these experiments it is apparent that directly correlating genotype and phenotype distances can provide detailed information concerning the locality of GP genotype to phenotype maps. The approach is therefore a viable alternative to the aggregative measures previously described. This can be validated statistically by the assignment of significance levels using the Mantel test. However, the technique is currently limited to representations which use unary (one-to-one) operators. Defining an exact distance metric under crossover is problematic (see for example Moraglio (2007) for discussions of crossover and metric spaces).

The Mantel statistic was demonstrated here for differences between phenotype measured at the syntactic, rather than semantic level. The computational cost of computing the phenotype metric (here, the normalised compression distance) between strings has not been considered and is relatively high for these representations. Therefore, the benefit of

adjusting indirect mappings using this form of syntactic bias is dependent on the cost of a fitness evaluation for the domain of interest. In problems where the fitness evaluation is comparatively cheap, the cost of obtaining genotype distances may outweigh the benefit. Conversely, such an approach would be more appropriate in situations where fitness evaluation is very expensive.

Using syntactic differences in phenotype either as a general performance predictor, or as a method of tuning existing genotype to phenotype maps, remains a subtle issue when applied to variants of GP employing indirect maps. A distinction can be drawn between different sources of diversity in the population; diversity introduced by weak locality in the genotype to phenotype map and diversity from the selection method. Despite these outstanding problems, it has been shown that the technique described in this chapter is a viable alternative to existing approaches and provides a method of determining whether measures of GPM locality are statistically significant.

3.8 Chapter Summary

This chapter proposed a statistical framework to measure locality in indirect genetic programming genotype to phenotype maps.

- The Mantel test, a technique obtained from numerical ecology, was introduced and a novel application given in artificial genotype to phenotype maps.
- The technique was validated using a constructed weighted integer representation.
- Case studies were carried out using two indirect GP encodings, *Cartesian Genetic Programming* and *Grammatical Evolution*.
- Distance metrics on genotype and phenotype spaces were considered. A suitable metric was defined and derived for CGP.
- It was established that a weak correlation between genotype and phenotype distances exists in both maps at the syntactic level, for arithmetic function sets and uniform mutation.
- The effect of varying syntactic locality on performance was measured over a set of randomly generated polynomial symbolic regression problems.
- Stronger syntactic locality was associated with a trend towards improved CGP performance under weaker, tournament selection, and reduced performance under truncation selection. The trade-off between selection pressure and locality in GP representations was discussed.

Having examined mapping locality from a statistical perspective, the next chapter will explore how visualisation techniques can be used to gain a more direct impression of locality in different GPM.

Chapter 4

Methods of Visualising Locality

Contents

4.1 Chapter Motivation

Visualisation is a significant consideration when exploring complex search spaces, such as those of Genetic Programming. Graphical presentation is an essential component in the development of randomised search heuristics, required to give a practitioner a more

intuitive understanding of algorithm behaviour. Particular importance can be attached to visualisation when insight is desired into the interaction between different elements of the system. Therefore, to construct a coherent picture of the role of locality in evolutionary and coevolutionary algorithms, this chapter will develop tools to visualise the genotype to phenotype map.

A drawback of the quantitative measures of locality reviewed in Chapters 2 and 3 is that information provided about a map's quality is obtained over the whole search space, or a large representative sample. The concept of locality is informally associated with a search space possessing a topology that is in some sense locally *smooth*, at the phenotype or fitness level. By that rationale, it is possible for operators to induce a search space that incorporates *regions* with strong and weak locality. Identifying these regions in a mapping and understanding their contribution is therefore of practical importance. From this perspective, it can be seen that whilst characterising the locality of a mapping with a single value provides a convenient overall view, it may obscure more subtle distinctions between particular mappings. Hence, it is appropriate to consider methods of presenting the relationship between genotype and phenotype directly. This will enable an examination of the changes in regional locality induced by choosing different representations and the differences in algorithm behaviour that ensue.

4.2 Chapter Overview

The focus of this chapter is on identifying and providing tools to visualise the locality of artificial genotype to phenotype maps. The intention is to address the question: *'How can locality in different regions of the search space be visualised?'*. In Section 4.3 existing methods are surveyed including, where relevant, approaches that have been applied to visualise other properties of the genotype to phenotype map, such as neutrality. These are captured in a taxonomy which sets the context of available techniques. Section 4.4 follows this discussion by presenting a new framework for graph-based visualisation of the fitness landscapes of indirect GPM. A short review is provided of graph layout algorithms and their utility is explored when examining the locality of genetic programming search spaces. Section 4.5 then applies this approach in a set of experiments using simple example problems, shown for basic versions of Cartesian Genetic Programming and Grammatical Evolution. The final section summarises these observations and findings across both representations.

4.3 Visualising Structure in Genotype-Phenotype Maps

Extensive effort has been applied within the field of Evolutionary Computation to display data regarding the general state and progress of search. The subject matter is a broad area and is the object of at least one dedicated international workshop series.[1] However, when considered from first principles, tools to visualise information about an evolutionary algorithm can be assigned amongst three categories: those applied *prior* (before running a search) *concurrently* (displaying data whilst an algorithm is ongoing) and *post* (retrospectively applied to understand what has occurred).

The first type (*prior*) addresses the provision of visual information that is used to predict expected behaviour when the algorithm is run. Examples include the visualisation of data derived from approaches such as sampling the effect of search operators, selection mechanisms or models of the structure of the search space. The second category (*concurrent*) presents relevant information about the current state. Examples include how particular phenotypes interact with the system within which they are evaluated, how that system is changing and derived properties of populations such as fitness. The practitioner may be interested in whether a search has stagnated and no progress is being made, or in the diversity of the current sample. By contrast, visualisations derived after the algorithm has terminated (*post*) are typically concerned with providing a picture of the changes obtained over the history of the search. These might include the lineages of particular offspring, how particular building blocks are transmitted throughout the search or a convergence diagram showing the quality of the best solution over generational time.

As noted in Chapter 2, when genotype to phenotype maps are incorporated in evolutionary algorithms they can be considered to be *fixed* or *variable*, dependent on whether the mapping can be changed at run time. Recall that a fixed mapping was defined as one where the relationship between genotype and fitness is independent of evolutionary time (Definition 2.2). An example is the standard polynomial symbolic regression problem employed in genetic programming by Koza (1992). Here, where the target function is static over generational time, fitness evaluations are determined according to the same method for all instances. To clarify, whilst there may be variance or noise in the assigned fitness values, the underlying function that is employed to map from genotype to phenotype does not change. By contrast, in a variable mapping the target function has a dependency on time, potentially mandating that visualisation be carried out concurrently.

[1] The workshop 'Visualisation Methods for Genetic and Evolutionary Computation' has been held annually at the Genetic and Evolutionary Computation Conference from 2010.

A second consideration is the data type of the representation and the level of mapping to which the visualisation approach may apply. Given that much of the underpinning requirement for visualisation techniques in Evolutionary Computation originally grew from the genetic algorithm literature, it is perhaps unsurprising that visualisation techniques aimed at mappings based from fixed binary strings are most prevalent. The comparatively recent context of evolution in more complex structures is therefore less extensively populated. Similarly, visualisations may be applied to GPM at different levels, ranging from genotype to fitness to intermediate states. We now review a number of the existing approaches, before classifying them using these criteria.

4.3.1 Historical Techniques

A Remark on Scope

This section comments on previous visualisation techniques or mathematical methods which have particular bearing on the issue of locality in genotype to phenotype maps. The intention is not to provide general coverage of visualisation methods in Evolutionary Computation; for broader reviews, the reader is referred to the work of (Collins, 1998b; Hart and Ross, 2001; Pohlheim, 2006).

Fitness Landscapes

Perhaps the oldest and best known tool for visualising the genotype to phenotype map is the concept of a *fitness landscape*. Fitness landscapes are a mainstay in evolutionary computation and evolutionary biology, adapted from their original presentation in genetics (Wright, 1988; Jones, 1995). Informally, fitness landscapes embed the topology of a genotype search space and corresponding fitness evaluations onto a surface. The placement of the fitness values on the surface is expected to be mapped such that that the adjacent neighbours in genotype space are adjacent on the surface. This concept of a neighbourhood is provided by a measure of distance on the genotype space, informed by the choice of search operator. For example, in the context of a binary string representation, as in classical genetic algorithms, the metric of choice is typically the Hamming distance, such that adjacent points on the fitness landscape differ by one bit (often termed a *locus*).

Adapting the definitions of Vassilev (2000), we state that:

Definition 4.1 (Hypergraph)**.** A *hypergraph* is a finite set of vertices $V = \{v_1, v_2 ... v_n\}$ and a family $\{E_i\}_{i \in I}$ of subsets of V, for which $E_i \neq \emptyset$ and $\bigcup_{i \in I} E_i = V$. The couple $H = (V, E)$ is then called a *hypergraph* of order n.

Definition 4.2 (Fitness Landscape)**.** A *fitness landscape* is the tuple $\mathcal{L} = (H_G, f, s)$. H_G is a hypergraph whose set of vertices correspond to the members of the genotype space G. The fitness function $f : G \to \mathbb{R}$ is used to label the vertices of H and s is the applicable search operator acting on G. Each valid action of s on a subset of G corresponds to an edge $e \in H_G$.

Remark: Vassilev's definitions use hypergraphs to express the concept of a fitness landscape. The definition may appear more complex than standard presentations, but has the advantage that by generalising the concept of an edge using hypergraphs, it facilitates a very flexible concept of neighbourhood. The definitions are rephrased to fit the notation used in this work.

Fitness landscapes provide both qualitative and quantitative information concerning the properties of a search space. Examples include the presence of local minima, the distance to optimal values or concepts such as smoothness or plateaus. Convention differs concerning whether optimal values in the search space correspond to maximising or minimising fitness. Therefore, if a fitness landscape is visualised on a two-dimensional surface, these may be represented as peaks or troughs respectively. A search can be informally seen as a set of points moving across this surface.

Despite their success and prevalence in the evolutionary computation literature, a number of drawbacks have been identified in the use of fitness landscapes when employed on more complex representations. This will be returned to in section 4.4.

Multidimensional Scaling Approaches

Multi-dimensional scaling involves the transformation of multi-dimensional data down to a smaller number of dimensions whilst preserving features or structure present in the higher dimensional space. The process can enable the properties of complex functions to be visualised more readily. Collins (1998a) and Hart and Ross (2001) give a more comprehensive review of these techniques in the context of visualising a genotype to phenotype mapping. However, for completeness, a short summary is provided here.

Most notably, within the framework of genetic algorithms and binary representations, multidimensional scaling techniques have included *Principal Component Analysis* (PCA) (Collins, 1999), *Sammon Mapping* (Dybowski *et al.*, 1996; Pohlheim, 2006), *Distance Maps*(Shine and Eick, 1997) , *Search Space Matrices* (Collins, 1998a) and *Self-Organising Maps* (Romero *et al.*, 2002). Each method has been applied to visualise the relationship between genotype and phenotype across either the whole search space, or some proportion of the searched area.

Principal Component Analysis (PCA) has the advantage of being computationally straightforward and possesses a rigorous mathematical foundation. The PCA-based approach applied by Collins (1999) was applied only to direct genotype to phenotype maps and was derived from images of the state of the population at a particular time, rather than the whole search space. However, it is not clear whether reduction to an orthogonal basis is appropriate for more general genotype to phenotype maps.

By comparison with linear methods such as PCA, *Sammon Mapping* is a non-linear method of dimensional reduction. The technique can be outlined as an optimisation process, where the objective is to maintain a similar configuration of distances between the elements of a higher dimensional space and the elements of the lower dimensional space. Discrepancies between each set of distances are accumulated as an error function, giving the quality of a map. This is then minimised, typically through methods such as gradient descent. In Evolutionary Computation, Sammon mapping was first applied to binary genotypes generated during an evolutionary run (Dybowski *et al.*, 1996). A similar error minimisation approach is the *Distance Map* described by Shine and Eick (1997), which explored the use of linear programming and GAs to derive a two dimensional version of the genotype space.

Search Space Matrices (Collins, 1998a), visualise the genotype space on a two dimensional lattice, using a fixed mapping specialised to bit string representations. This approach is computationally fast, but is difficult to extend to other representations. A potentially more flexible alternative is to consider learned embeddings. *Self Organising Maps* are a well known machine learning method, consisting of a lattice of nodes with associated weights, embedded onto a plane. This network is trained such that input vectors from the training data (the higher dimensional space) are presented to it, classically at random. By iteratively adjusting the weights of the closest matching node and nodes within a specified radius, the map is optimised to provide a form of unsupervised classification, spatially clustering nodes with similar features. In Romero *et al.* (2002), the map was trained directly, using a subset of the possible binary genotypes as the input vectors. Later work by Amor (2005) integrated this into search, using the trained map to improve diversity.

In summary, from these examples it can be observed that multi-dimensional scaling approaches are a useful tool to address the size and complexity of genotype to phenotype maps. Their application to more complex representations, rather than bit strings, is an area of current research (McCandlish, 2011).

Connection Maps and Frequency Maps

An alternative technique to address the density of information present in a genotype to phenotype map is to present that data on a shaded grid or heat map. Murphy *et al.* (2011) recently proposed two applications of heat maps, termed *Connection Maps* and *Frequency Maps*, demonstrated in the context of grammar-based genetic programming. Cells of each map correspond to pairs of distinct phenotypes. In Connection Maps, each pair of phenotypes in the phenotype space is enumerated and it is determined whether the distance between the associated genotypes is less than or equal to the smallest feasible change (for example, in integer-based Grammatical Evolution, this is mutating one integer value). The corresponding cell is coloured where genotypes are neighbours in this sense. A Frequency Map enumerates phenotype pairs similarly, but colours cells in proportion to the number of connected neighbours, thus visualising redundancy.

Heat maps are an interesting method of capturing large data sets and when arranged in this fashion can potentially capture much of the variation in locality across a genotype space. However, a modification to the above methods would be required to simultaneously represent genotypes at distances further apart than the local neighbourhood, or for which the transition probability is position dependent (true in some graph-based encodings such as CGP).

Quotient Graphs

Wilson and Kaur (2009) proposed an original approach to visualising properties of small genotype to phenotype maps, as a tool to analyse neutrality in indirect mappings. The method employed the mathematical notion of a *Quotient Graph* (derived from the definition of a quotient set) to visualise how mutational drift could be biased by the presence of neutrality in integer string genotypes. The work was applied in a case study on Cartesian Genetic Programming and Grammatical Evolution, using the uniform mutation operator.

Informally, a quotient graph can be defined from a partition on the vertices of an undirected graph. Members of each partition share the same adjacency with respect to other partitions. Wilson combined this concept with a carefully defined notion of search neutrality: genotypes which not only share the same phenotype, but that give rise to the same distribution of phenotypes on the application of a search operator. Wilson's visualisations convey compactly how the genotype to phenotype map can give rise to a bias in a search, over different neutral regions in the genotype space. Although the approach was not established with the aim of investigating locality, it further demonstrates that graph based visualisations can be used to effectively present complex GPM properties.

TABLE 4.1: *Techniques used to visualise the structure of genotype to phenotype maps. Mapping refers to the representations employed in the original application. A distinction is made between mappings that display fitness or an intermediate phenotype.*

Visualisation Technique	*Mapping Originally Considered*	*Point of Use*
Fitness Landscape	Genotype → Fitness	Prior
Connection Maps	Integer Genotype → Phenotype	Prior
Frequency Maps	Integer Genotype → Phenotype	Prior
Quotient Graph	Bit String, Integer Genotype → Phenotype	Prior
PCA	Bit String → Fitness	Concurrent
Sammon Mapping	Bit String → Fitness	Concurrent
Self-Organising Maps	Bit String → Fitness	Concurrent
Distance Maps	Bit String → Fitness	Concurrent
Search Space Matrices	Bit String → Fitness	Prior

Summary

Table 4.1 summarises the techniques that have been outlined in this section. Multi-dimensional scaling techniques have primarily been applied to binary strings, using direct maps to fitness. Connection Maps, Frequency Maps and Quotient Graphs originate in the genetic programming literature and are therefore applicable to more general notions of genotype and phenotype. Fitness Landscapes are ubiquitous but by definition do not conventionally display intermediate phenotypes between genotype and fitness. Similarly, with the notable exception of Search Space Matrices, the multi-dimensional scaling techniques described here have been used concurrently with the search process, rather than as a tool applied before search to understand the structure of the search space. The next section considers an alternative method of multi-dimensional scaling, using graph-layout algorithms.

4.4 Fitness Graphs

Section 4.3 considered a number of existing approaches to visualising the structure of genotype to phenotype maps. However, it was observed that the majority of these techniques are applicable to direct maps and binary string representations, rather than indirect maps or program representations. Furthermore, even in the comparatively simple context of binary strings, displaying large numbers of bits and correspondingly large neighbourhood spaces can present difficulties as the number of dimensions in the genotype increases. The problem is more acute when visualising GP data types, such as trees or graphs.

As an example, Langdon and Poli (2002) criticise the application of fitness landscapes to classical tree genetic programming. Key issues they highlight include the lack of a *natural neighbourhood relationship*, the *asymmetry in search operators* and *variation in the transition probability* between genotypes which are ostensibly neighbours. The first point relates to the complexity of selecting distance metrics on tree or graph structures, which has been addressed in section 3.3.1. An example of operator asymmetry is crossover, which typically produces a single offspring from two parents. The third point notes that, dependent on the search operators, genotypes with apparently similar structures may be distant when considering the actual transition probability.

To counter to these issues, they advocate visualising fitness landscapes in programs as weighted graphs. This provides a more general structure on which to draw the search space topology:

> *"The lack of natural ordering may prevent us from being able to plot the landscape, even for one or two dimensional search spaces. However, we can still visualise the landscape by representing it as a graph where the nodes represent the points in the search space and the links indicate which points are neighbours. In order to visualise the fitness of the points in the search space, the nodes in the graph could be labelled or coloured on the basis of their fitness."*
>
> **- Langdon and Poli (2002)**

For convenience and to help distinguish references to the drawing from the abstract mathematical object, we will term this drawing to be a *fitness graph*.

4.4.1 Drawing Fitness Graphs using Graph Layout Algorithms

Langdon and Poli's description gives a starting point from which to build an alternative approach to visualising the locality of a mapping. Consider a drawing that contains a vertex for each member of the genotype space G. Scale the radius of each vertex according to the normalised fitness of the corresponding phenotype and relate edge lengths to the distances between genotypes. Because landscapes drawn with geometric distances exactly proportional to their edge-lengths do not in general embed onto a plane in a simple fashion, a compromise must be sought. One approach is to seek a useful embedding by employing *graph-layout algorithms*, which optimise the layout of nodes given this constraint, as well as aesthetic considerations. The design of such an algorithm falls within the extensive field of graph-drawing. Whilst a full introduction to graph-drawing is outside the scope of this work, a brief discussion is worthwhile to place our experimental choices in their proper context.

The review of Eades and Hong (2012) describes two principal classes of algorithm in graph-drawing, *Force Directed* and *Planarity* approaches. In the former approach, a graph is defined to have a total energy, under an 'energy model', which is given as a function of the distances between vertices when these are assigned positions in a drawing. Iteratively minimising this function by allowing vertices to move to a local optimum improves the derived layout.[2] This can be seen equivalently as converging on an equilibrium, such that the total force applied to each vertex is zero. Forces comprise a mixture of attractive and repulsive components, based on a selected physical model. By contrast, Planarity layout algorithms construct drawings more directly, proceeding by placing vertices explicitly within a planar embedding. The layout is accomplished by geometric methods, such as triangulation of the faces of the graph. Fitness graphs used to visualise genotype to phenotype maps are large and typically not planar, that is they cannot be embedded directly onto a plane without crossing edges. Force Directed approaches are therefore the most appropriate for this case, in order to generate a qualitatively useful layout and expose the structure of the search space.

Force Directed Layout

In the experiments described in this chapter, three force directed algorithms were explored, applied using the **Gephi** (Jacomy *et al.*, 2009) graph visualisation software: *Fruchterman-Reingold* (Fruchterman and Reingold, 1991), *Yifan-Hu* (Hu, 2005) and *Force Atlas 2* (Jacomy *et al.*, 2011). The former are well-established algorithms that

[2]The reader will note the similarity to techniques described in the previous section on multi-dimensional scaling. Though the fields are usually regarded as distinct, graph-drawing could also be classified as a kind of dimensional reduction, albeit one specialised to a particular structure.

have been shown to be particularly effective for the analysis of large social networks (Carrington *et al.*, 2005; Socievole and Marano, 2012). The latter is a state-of-the-art addition, a fast implementation which employs an approach similar to that of Fruchterman-Reingold, the so-called '*spring-electric*' model.

The original *spring* model derived attractive forces between vertices based on the ideal behaviour of mechanical springs and therefore linearly proportional to distance (Hooke's law). The *spring-electric* model is a variation on this applying repulsive forces using a heuristic analogous to that occurring between between electrically charged particles in classical electromagnetism (the inverse square law). The Fruchterman-Reingold algorithm modifies these relations to include different scaling relations with distance, based on properties of the graph and layout area. More recent algorithms have altered these relations further, to accommodate different computational requirements and aesthetic factors. Notably, in the Force Atlas 2 algorithm the repulsive force is modified to increase with the number of edges connecting to each vertex. Thus, vertices with a larger number of edges connections will tend to be laid out further apart, which improves the clarity of presentation.

Time Complexity

When visualising fitness-graphs that represent genotype spaces directly, efficient scaling with respect to the search space size is necessary. Because edge numbers $|E|$ increase with the number of neighbouring genotypes (vertices $|V|$), this is the main criterion that must be considered. Of the algorithms described, the basic Fructerman-Reingold algorithm scales as $\theta(|V^2| + |E|)$ (Fruchterman and Reingold, 1991). The Yifan-Hu algorithm has been shown to be $O(|V|log|V|)$ in time per iteration (Hu, 2005). No formal theoretical bounds on the efficiency of Force Atlas 2 have yet been provided, but in the original submission the authors presented an empirical comparison across four benchmark data sets that suggests it is highly competitive. These bounds refer to time to update the position of all vertices. Table 4.2 summarises the force components and theoretical performance of each algorithm.[3]

In practice, two further issues must be addressed when employing force directed layout algorithms. Firstly, because the algorithms find a local minimum, they are sensitive to the initial layout of the graph. It may therefore be necessary to randomise this and apply the heuristic over a number of instances to derive a satisfactory graph drawing. Secondly, it should be determined whether the algorithm incorporates the original edge weights. The Fructerman-Reingold and Yifan-Hu algorithms optimise the layout to

[3]For conciseness, in the force component summary shown in Table 4.2 some constant tuning parameters are neglected. Full details are available in the cited papers for each algorithm.

TABLE 4.2: *Properties of the Force Layout Algorithms; where d is the distance between each pair of vertices, k is a precalculated value giving the desired vertex separation (also referred to as the natural spring length), $j \geq 1$ an integer parametrising scaling of the repulsive force and D_1, D_2 is the degree of each vertex respectively.*

Force Layout Algorithm	Force Components		*Complexity (per iteration)*	*Scaled by edge weight?*
	Attractive	*Repulsive*		
`Fructerman Reingold`	$\frac{-d^2}{k}$	$\frac{k^2}{d}$	$\theta(\|V^2\| + \|E\|)$	No
`Yifan-Hu`	$\frac{-d^2}{k}$	$\frac{k^{1+j}}{d^j}$	$O(\|V\|log\|V\|)$	No
`Force Atlas 2`	$-d$	$\frac{(D_1+1)(D_2+1)}{d}$	Not disclosed	Yes

give close to uniform edge lengths. This enables a useful view of the neighbourhood structure of each vertex, but neglects variations in distance within the neighbourhood. Force Atlas 2 addresses this by allowing an optional weighting parameter $\delta \in [0.0 : 1.0]$ which modifies the attractive force (F_a) between vertices as

$$F_a = -w(e)^{\delta} d(v_1, v_2) \tag{4.1}$$

where $d(v_1, v_2)$ is the geometric distance drawn between the pair of vertices v_1 and v_2, $w(e)$ is the weight assigned to the corresponding edge (Jacomy *et al.*, 2011). From the perspective of the current work, this permits the graph layout to more appropriately reflect the distance metric used to set the length of the graph edges. Figure 4.1 illustrates this, showing the application of the force layout algorithms to a simple randomly weighted network. The Fructerman-Reingold and Yifan-Hu algorithms result in a relatively uniform placement of vertices across the drawing, regardless of their corresponding edge weighting. In Force Atlas 2, the most strongly weighted edges (thicker lines) correspond to a stronger attractive force and the source and target vertices are proportionally closer.

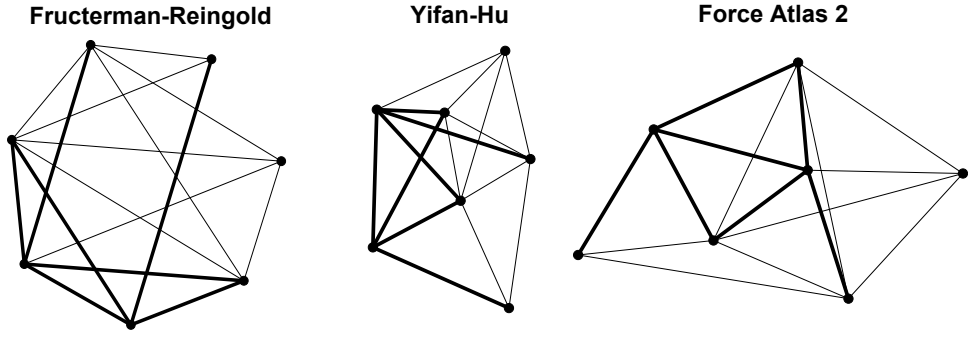

FIGURE 4.1: *Examples of graph layout algorithms applied to a simple randomly generated weighted network with 7 vertices. Vertices in the Fructerman-Reingold and Yifan-Hu algorithms achieve clarity of presentation, but are laid out independently of edge weight. Vertices connected by strongly weighted edges are closer under Force Atlas 2.*

4.4.2 Displaying Locality in Fitness Graphs

Graph Preparation

Assume a genotype space where all genotypes are mutually reachable, using the expected mutation distance metric. A complete fitness graph is formed from the genotype space with $|G| = |V|$. As a complete graph, the space has $|E| = \frac{|G|(|G|-1)}{2}$ edges, which is too dense to be practical even for highly constrained representations. Each edge between the genotypes is assigned a weight scaled inversely proportional to the logged expected mutation distance, $w(e) = 1/log_{10}(\bar{M})$ (Definition 3.3). This implies that large probabilities of transition give a large attractive force.

To present a more convenient view of the local neighbourhood (the high probability transitions), pruning can be carried out, starting with the edges with the highest weight, which correspond to most distant genotype-pairs. At each stage the weight is reduced by a constant Δw, selected to remove the next equivalent set of edge weights. The graph is reduced to the lowest weight such that it is still fully connected. This procedure is described in the `SimplePrune` algorithm overleaf (Algorithm 2) and illustrated in Figure 4.2. The significance of retaining a fully connected graph is to avoid separation into multiple components, which will then be inappropriately aligned by the force layout algorithm.[4]

Assigning Locality Values

Assigning an appropriate value to represent the locality of a vertex depends on the stage of the mapping under consideration and the definition of locality that is accepted. For the purpose of these visualisations, each vertex was assigned a value equivalent to the mean phenotypic distance over the local neighbourhood:

$$\bar{d_p} = \sum_{g' \in adj(g)} \frac{d_p(g, g')}{N_g} \tag{4.2}$$

where N_g is the number of adjacent vertices for genotype g after pruning. This does not exclude neutral or zero value changes to the genotype. The distance d_p may be derived from the differences at the intermediate stage, given for example by the metrics in section 3.3.1. It may also refer to the change in fitness, in which case this becomes

[4]Experience with the Force Atlas 2 algorithm suggested that when laying out graphs with multiple components, the distance between components increased such that drawing them on a scale that could still display component detail became difficult. This is a consequence of the balance between attractive and repulsive forces in the algorithm. An alternative approach would be to adjust the layout parameters controlling these, but retaining a single component was simpler than this in practice.

Data: A Weighted Graph G
Result: A Pruned Graph
w ← maximum weight of G;
while G *is connected* **do**
    subtract all edges with weights of value greater than w from G;
    w ← w - Δ w;
end
return G

Algorithm 2: `SimplePrune`

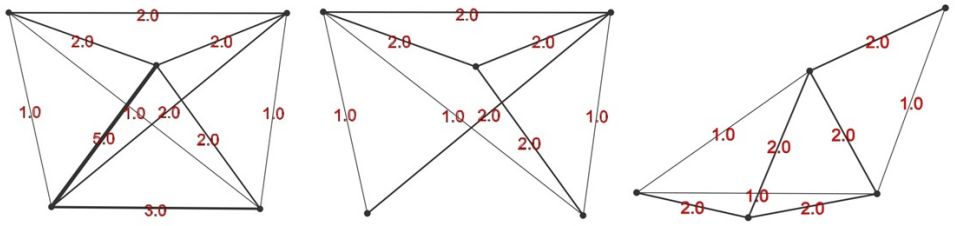

FIGURE 4.2: *Application of* `SimplePrune` *to (weighted)* K^5*. Equivalent weight edge-sets are successively removed, from the largest weight to the smallest, until the lowest weight, fully connected graph is obtained. The drawing is then mapped using Force Layout 2.*

similar to the definitions of Galvan-Lopez *et al.* (2011a), described in Chapter 2. As with any average, aggregating in this fashion can potentially be deceptive, depending on the characteristics of the true distribution (it is not robust to outliers). Nonetheless it is useful to generate a first impression of the locality of a mapping across different regions of the search space.

A Note on Colour Mapping

Figure 4.3 above continues the example followed in the previous section, showing graphs coloured by fitness and locality respectively. To assign colours to locality and fitness values in the visualisation, both sets of values were first normalised to the range [0:1].[5] For locality, the hue of each vertex was then mapped according to the equation:

$$H = 90 - 90 \times \sqrt{(1 - (L_g - 1)^2)} \tag{4.3}$$

which was found to be suitable after a process of trial and error. Saturation and brightness were both set to 100. The expression given by Equation 4.3, then provides a mapping such that HSB (90,100,100) corresponds to bright green, associated with the lowest value of L_g (strongest locality) and (0,100,100) with a bright red, the highest

[5]For problems where the range of possible fitness values was continuous, such as symbolic regression, normalisation was carried out with respect to minimum and maximum values generated across the genotype space.

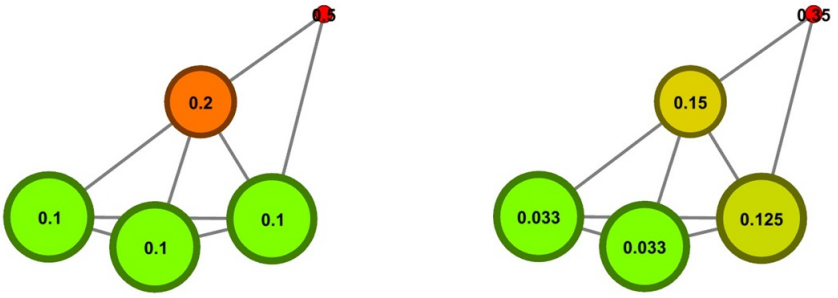

FIGURE 4.3: *Left: An example graph with the vector of normalised fitnesses* (0.5, 0.2, 0.1, 0.1, 0.1). *Vertices are labelled and coloured by fitness. Right: Graph re-labelled and coloured by the corresponding locality of each vertex. Note that for reference the radius of the vertex represents fitness in both cases.*

value of L_g (weakest locality). Fitness was mapped in a similar fashion (we take the convention of minimising fitness). The resulting colour maps are monotonic in both the locality and fitness, whilst also providing a clear distinction between genotypes in regions with different properties.

4.5 Application to Indirect Genotype Phenotype Maps

4.5.1 Selected Test Problems

Two simplified problem instances were selected to verify the technique described in the previous section. Because GP search spaces scale rapidly with representations of increasing length, the test problems were constrained to those solvable using a restricted number of dimensions, terminals and functions. These instances are trivial in practice, because the search space is small enough to be enumerated. However, studying them can help to provide a better understanding of the variations in structure induced by different kinds of GPM, with the intention of providing insight into the biases present in larger problems.

MAX-GP

MAX is a basic analogy of the GA One-MAX problem, which has been employed in the analysis of canonical Tree based GP and indirect representations (Langdon and Poli, 2002; Murphy *et al.*, 2011). For clarity, since different classes of representation will be addressed in this thesis, instances here will be referred to as MAX-GP. The goal of MAX-GP is to generate a program that provides the largest possible real number output given a fixed set of real valued terminals and arithmetic function set.

MAX-GP is a suitable point to begin to assess this visualisation technique, because solutions can be defined for relatively small search spaces, which it is possible to show in their entirety. Another attraction of the problem is that (for some limited cases) it is possible to calculate theoretical bounds on the performance of GP. An example of this is provided for CGP in Appendix A.

Simple Binomial ($x^2 + x$)

The second problem is a symbolic regression instance with the form $x^2 + x$. Symbolic regression problems of practical significance can require a large terminal and function set and correspondingly large search space sizes. The simplicity of this function enables the search space to be fully enumerated. The visualisation in the next section shows the fitness of the model solutions and their corresponding neighbourhood.

4.5.2 Representation and Search Operator Details

Cartesian Genetic Programming with Bi-arity functions

Integer genotypes in Cartesian Genetic Programming have a position dependent range of values. In Seaton *et al.* (2010), the scaling for the single row version of CGP with bi-arity functions was derived as

$$|G| = |F|^n \Pi_{i=0}^{n-1} (|T| + i)^2 = |F|^n \left(\frac{(|T| + n - 1)!}{(|T| - 1)!} \right)^2 \tag{4.4}$$

where n is the number of nodes, $|T|$ is the size of the terminal set, $|F|$ the size of the function set, assuming unconstrained feed-forward input to each node. The position dependency results in a factorial increase in search space size with the number of terminals and nodes.

Given this rate of increase, in the MAX-GP problem, a restricted function and terminal set was adopted using only the addition operator + and a single constant input 1.0. Each CGP node can therefore at most double the output of the previous node, such that the maximum output at node n is 2^n. This occurs when both inputs for each node connect to the previous layer. The sign of the output is always positive. Individual genotypes are therefore assigned a normalised fitness as $f = \frac{k}{2}^n$ where k is the value of the output. There is a single optimal value at $f = 1.0$. From Equation 4.4, the search space size for this case is 576 genotypes. In the Binomial problem, the function set was expanded to $\{+, *\}$, over three nodes. This gives a search space with a single optimal phenotype,

which can be mapped by 8 distinct genotypes, of size 288. A uniform mutation rate of $m = 0.15$ was assumed in both cases.

Grammatical Evolution with Bi-arity Functions

In fixed length forms of Grammatical Evolution, where genotypes comprise a fixed number of z integers, $0 \leq z < k$, the search space scaling is simpler than for Cartesian GP, given by z^k. For the MAX-GP problem a highly constrained grammar ($k = 2$) was first defined, to provide a comparable - though not equivalent - phenotype space to the CGP representation (Figure 4.4). The genotype space represents members from the set of possible trees up to a predetermined depth. The exact trees represented are dependent on the maximum permitted genotype length. For the Binomial problem, the grammar was modified to permit the multiplication operator (Figure 4.5).

```
<expr> ::= <term> |                    (0)
           <func> (<expr>,<expr>)      (1)
<func> ::=     +                       (0)
<term> ::=     1                       (0)
```

FIGURE 4.4: *Simple constrained grammar for the MAX-GP problem.*

```
<expr> ::= <term> |                    (0)
           <func> (<expr>,<expr>)      (1)
<func> ::=      + |                    (0)
                *                      (1)
<term> ::=      x                      (0)
```

FIGURE 4.5: *Simple constrained grammar for the Binomial problem.*

4.5.3 Visualisations of CGP and GE Search Spaces

Figures 4.6 to 4.13 show the fitness graphs computed for each search space and laid out using the Force Atlas 2 algorithm. In the first pair of graphs for each problem (Figures 4.6, 4.7, 4.10 and 4.11), colour is mapped to fitness. Edges are omitted in the drawing, but their weight is accounted for in the layout algorithm per Equation 4.1. In the second pair of graphs for each problem (Figures 4.8, 4.9, 4.12 and 4.13) colour is mapped to mean fitness difference, calculated according to Equation 4.2. The radius of nodes is proportional to the normalised fitness in both cases to permit a relative comparison (though note that the overall dimensions of each image have been scaled). The largest nodes are optimal solutions.

FIGURE 4.6: *Genotype space for a four node bi-arity CGP representation, on the MAX-GP problem. Red: Low fitness genotype. Green: High fitness genotype.*

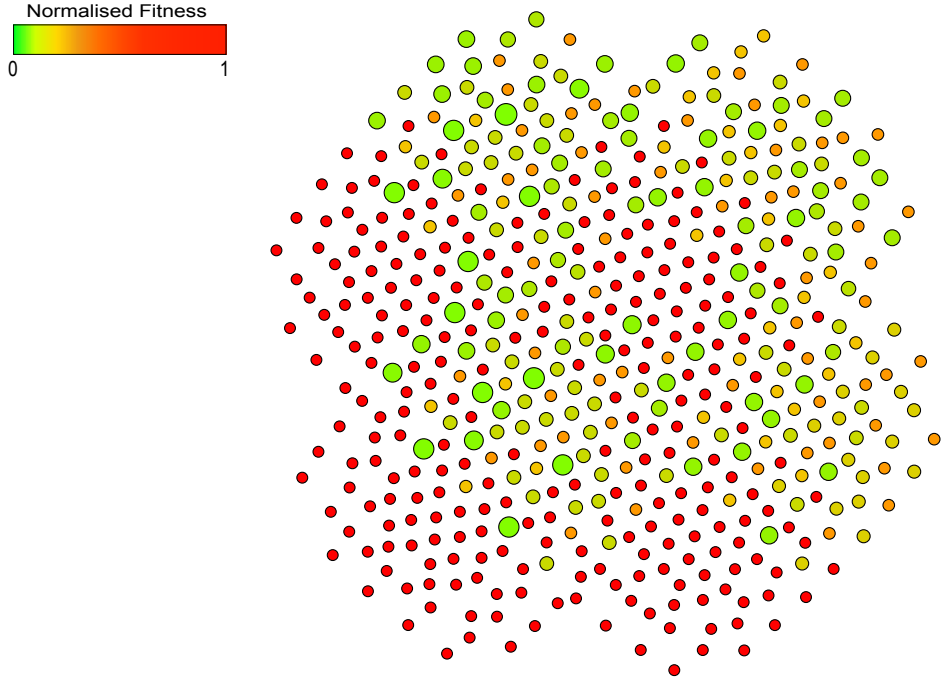

FIGURE 4.7: *Genotype space for a Grammatical Evolution representation using an array of nine integers, on the MAX-GP problem. Red: Low fitness genotype. Green: High fitness genotype.*

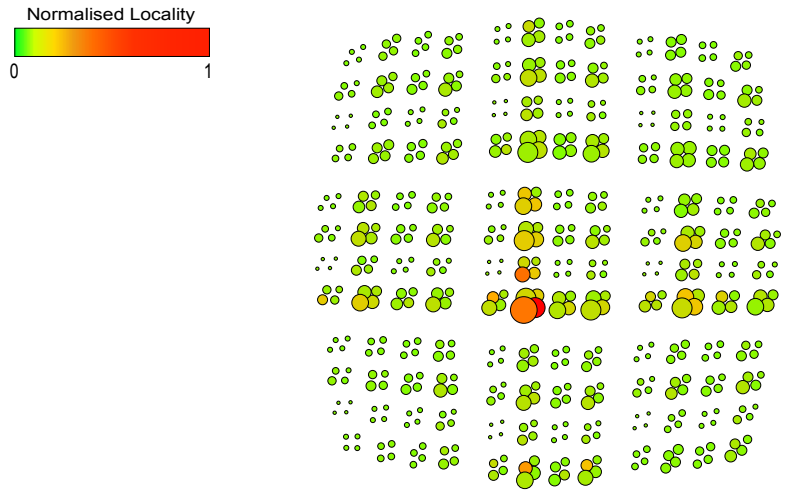

FIGURE 4.8: *Genotype space of a four node bi-arity CGP representation on the MAX-GP problem. Red: Weak locality genotype. Green: Strong locality genotype.*

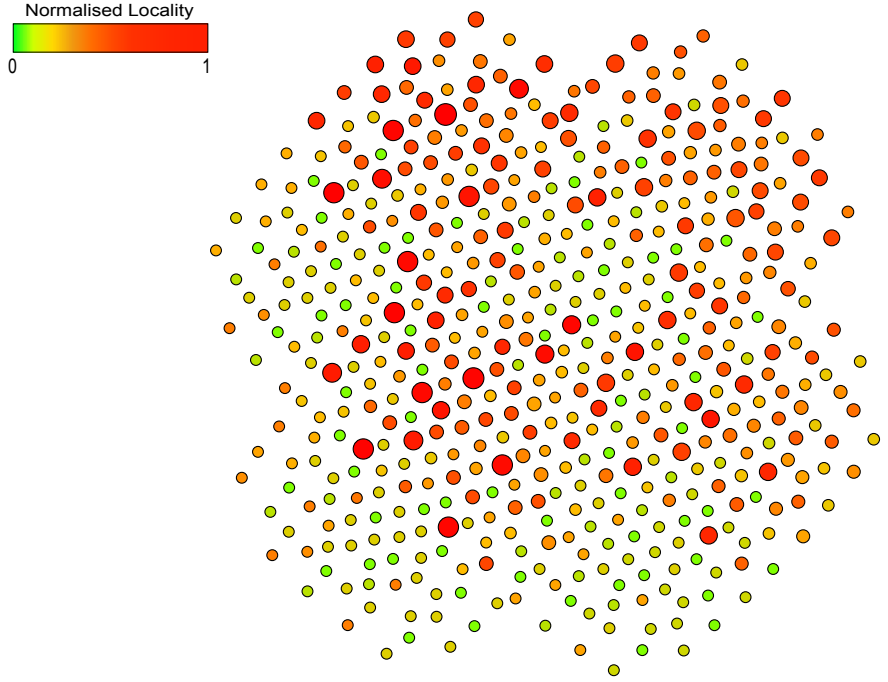

FIGURE 4.9: *Genotype space of a Grammatical Evolution representation using an array of nine integers, on the MAX-GP problem. Red: Weak locality genotype. Green: Strong locality genotype.*

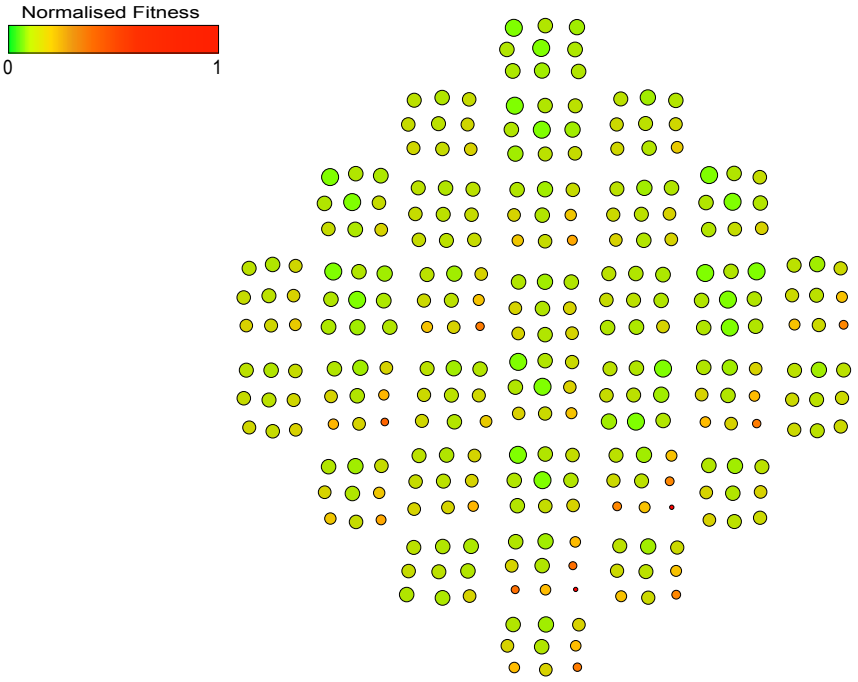

FIGURE 4.10: *Genotype space of a three node bi-arity CGP representation, on the Binomial $x^2 + x$ problem. Red: Low fitness genotype. Green: High fitness genotype.*

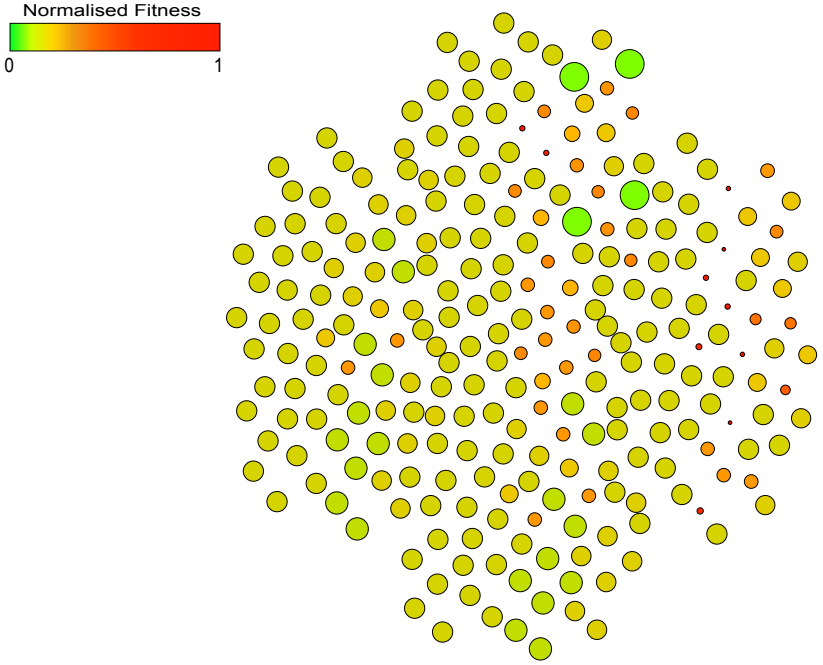

FIGURE 4.11: *Genotype space of a Grammatical Evolution representation using an array of nine integers, on the Binomial $x^2 + x$ problem. Red: Low fitness genotype. Green: High fitness genotype.*

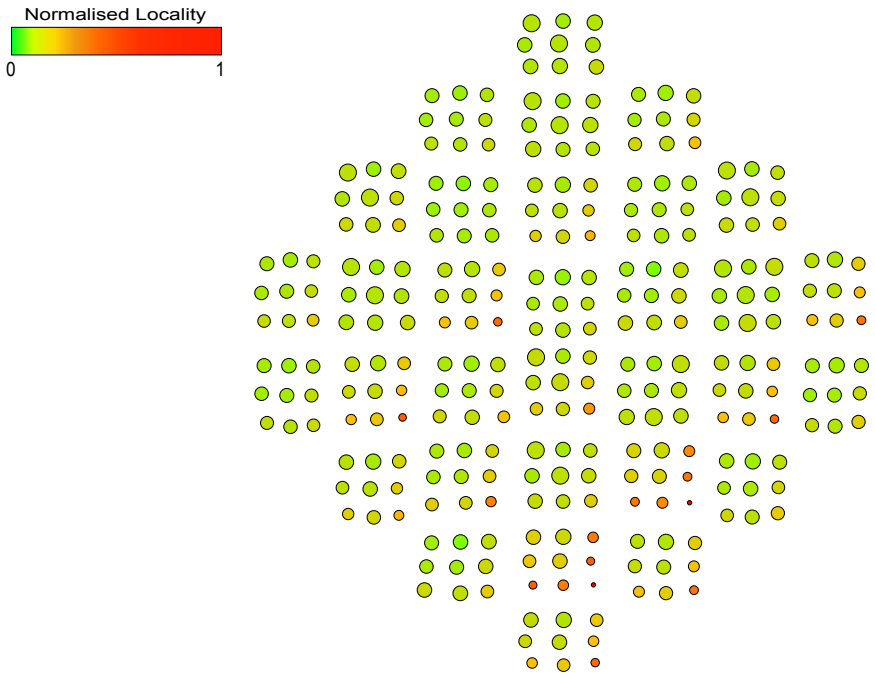

FIGURE 4.12: *Genotype space of a three node bi-arity CGP representation,on the Binomial $x^2 + x$ problem. Red: Weak locality genotype. Green: Strong locality genotype.*

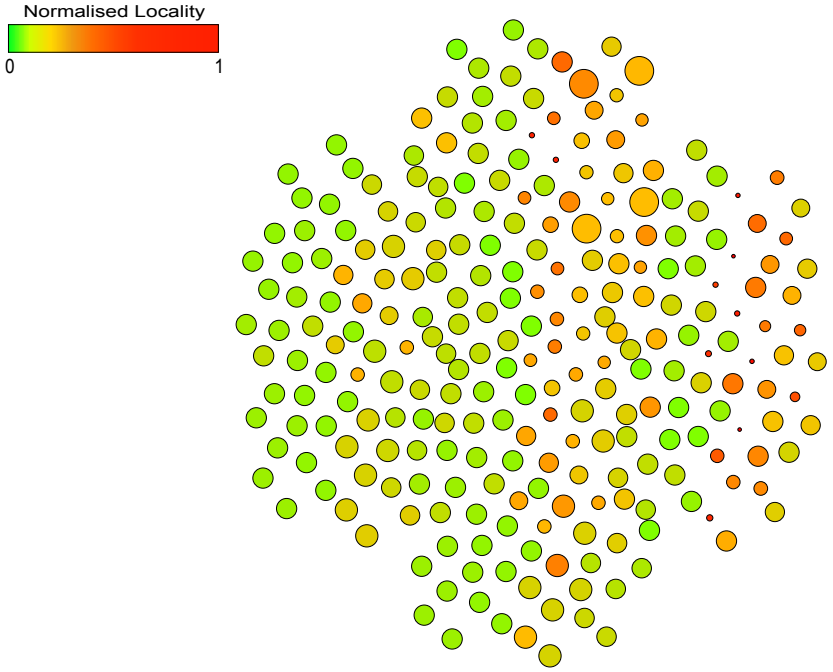

FIGURE 4.13: *Genotype space of a Grammatical Evolution representation using an array of nine integers, on the Binomial $x^2 + x$ problem. Red: Weak locality genotype. Green: Strong locality genotype.*

4.6 Analysis of Obtained Visualisations

4.6.1 MAX-GP

The CGP search space shows a high degree of structural symmetry. In the MAX-GP case considered, the genotype space is clustered into neighbourhoods of four individuals (Figures 4.6, 4.8). This is intuitively correct, based on the constraints placed on the CGP genotype representation. A node in the arity two CGP genotype used consists of two integer connections and a function value. The latter value cannot change because only one function is available. The node with the smallest number of connections is node two (the left most node can only connect to the fixed input.) Each of the two input connections has two possible alternatives. This gives three equidistant nearest neighbours. These highest probability offspring are correctly shown as the closest neighbours in the visualisation. The characteristically high neutrality of the GPM is also evident; many genotypes have an equivalent output and hence equivalent fitness.

By contrast, the Grammatical Evolution mapping under MAX-GP shows no clearly defined structural symmetry, but high fitness individuals are still closely connected. One half of the search space is composed of terminal values. This is a consequence of the constraints imposed on the specific grammar; when the grammars given in Figure 4.4 and Figure 4.5 are parsed, genotypes initialised with an odd value in their first position will evaluate to x. An EA operating on such a grammar would first eliminate these low fitness values from the population. The grammars used by GE in practice typically feature a much more diverse terminal set, which would lead to a correspondingly higher degree of diversity amongst these basic elements.

4.6.2 Simple Binomial

The properties observed in the Simple Binomial problem are similar to the MAX-GP examples in both cases. For CGP, the fitness graph displays symmetry around each set of nine clustered genotypes. The main distinction is the presence of multiple optimal genotypes, twenty of which are displaced across the search space. In the grammar-based representation, the optimal values are more tightly connected, but situated within a region that has lower locality on average.

Of particular interest is that the CGP and GE representations, despite possessing a superficially similar genotype structure and use of the same search operator, lead to genotype to phenotype maps with quite distinct structural properties. Regions with

strong and weak locality are visible in both cases, but the distribution of these is dependent on quite specific aspects of each representation.

4.7 Visualisation of Large Fitness Graphs

The sampling approaches described in Section 3.5 can be applied to plot subgraphs of larger genotype to phenotype maps. Fitness graphs were obtained for a standard application of CGP, digital circuit synthesis using boolean functions (Miller, 2011). Two basic example circuits were selected, a 1:4 demultiplexer and 2-bit multiplier. In both cases, fitness is derived from the truth table for that circuit, accumulated from the total number of correctly predicted outputs. The truth tables are given in Tables 4.3 and 4.4 below respectively.

TABLE 4.3: *1:4 Demultiplexer Truth Table*

Inputs			Outputs			
X_1	X_2	X_3	Y_1	Y_2	Y_3	Y_4
X_1	0	0	X_1	0	0	0
X_1	0	1	0	X_1	0	0
X_1	1	0	0	0	X_1	0
X_1	1	1	0	0	0	X_1

TABLE 4.4: *2-bit Multiplier Truth Table*

Inputs				Outputs			
X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Examples are presented for a linear-feed forward representation with 30 nodes and the boolean function set $\{AND, OR, NOT, IF\}$. This number of inputs, functions and nodes

implies the search space is far too great to enumerate. Figures 4.14 and Figure 4.15 illustrate fitness graphs obtained using the CGP representation for each problem, sampling using the chain referral approach, shown without and without edges respectively.

The figures illustrate the fitness graphs of ancestral trees derived from a selected optimal point. Regions of high and low fitness, as well as fitness neutral plateaus, are visible in both instances. Because the graph represents only a sample of the whole space, the symmetries evident when visualising a completely enumerated genotype space are not evident. Each tree is plotted to a depth of 10 mutations from the optimal point. The islands (separated components) visible correspond to branches that contain genotypes at a larger expected mutation distance from the main clusters.

4.8 Discussion and Concluding Remarks

The intention of this chapter was to establish a technique to visualise the locality of program representations, across different regions of the genotype to phenotype map. Representing fitness landscapes as graphs is an appropriate method to accomplish this, but projection onto a low-dimensional space is problematic. It has been shown that this can be addressed through pre-processing of the graph and the application of force-layout algorithms.

Choice of the expected variation distance as a measure of the difference between two genotypes was prompted by a requirement to select metrics that are more representative of the underlying search operator. A limitation of the metric is that it assumes a transition between any pair of genotypes can be made in a single evolutionary step. It is therefore most applicable to indirect representations that can use simple operations on a linear genotype, such as CGP or GE, but could in principle be adapted to any EA where this criterion will be met.

In both the Grammatical Evolution and CGP representations, a consistent observation for these problems is that optimal solutions have several neighbours with significantly poorer fitness. More generally it can be seen that, qualitatively, better fitness genotypes were located in regions of the fitness landscape that are not smooth. This is because the majority of genotypes have mediocre fitness and are therefore more likely to accumulate smaller values under Equation 4.2.

Despite the low locality of both the CGP and GE mappings observed in the fitness graphs, fit genotypes are still relatively tightly clustered and therefore in general connected by high probability transitions. For these particular problems and configurations, the optima in GE compose a larger portion of the search space and appear to be more

FIGURE 4.14: *Illustration of a fitness graph sampled for the 1 to 4 demultiplexer problem using a CGP representation. Edges not shown. Colour indicates fitness.*

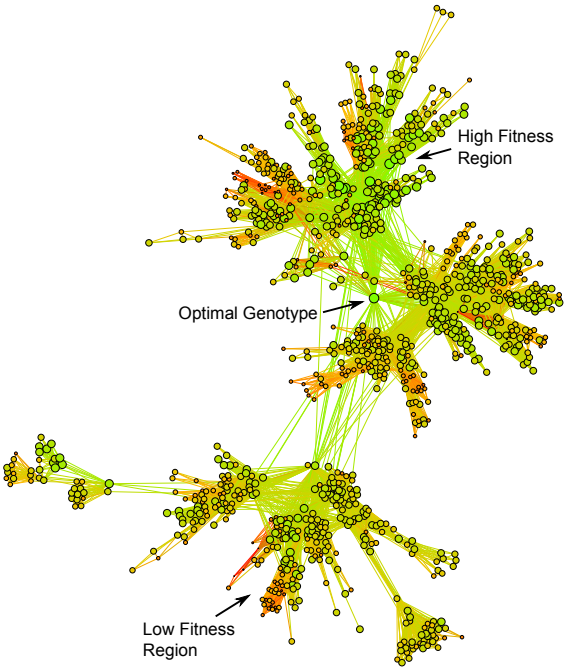

FIGURE 4.15: *Illustration of a fitness graph sampled for the 2 bit multiplier problem using a CGP representation. Edges shown. Colour indicates fitness.*

closely connected to low fitness genotypes than in CGP. These observations are somewhat in agreement with the traditional heuristics that have been established for each technique. CGP typically uses a stronger selection pressure than GE. In principle, this should prove advantageous on a smoother landscape; whilst a more relaxed selection pressure may provide performance gains on a rougher, multi-modal surface.

Finally, the CGP digital circuit visualisations in Section 4.7 were provided as examples to illustrate the technique on larger problems using sampling. As sampling density in a region increases, one would expect structural symmetries similar to those observed in the completely enumerated cases to become more evident.

4.9 Chapter Summary

This chapter put forward a novel method of visualising regions of strong and weak locality in indirect genotype to phenotype maps, using graph-layout algorithms.

- Existing methods of visualising genotype to phenotype maps were summarised and the scope of each technique considered. Specific issues related to the visualisation of program representations were discussed.
- A brief introduction to graph-layout algorithms was given. It was proposed that the force-layout class of algorithms can be used to draw fitness landscapes in GP.
- The approach was tested on two simple problems, MAX-GP and Binomial, using basic forms of CGP and GE.
- The locality of each representation was examined using the resulting visualisations. Strongly local and non-local regions could be identified in both cases. A high degree of symmetry was observed in the CGP case.
- Examples were presented of the application of the force-layout approach to plot larger scale fitness graphs in digital circuit evolution, for a 2-bit multiplier and 1:4 demultiplexer.

Chapters 3 and 4 have focused on providing statistical and visualisation tools to analyse locality in standard evolutionary systems that consist of a single population of genotypes. In the following part, we will move to consider a new context, coevolutionary systems. The following chapters will extend the current approaches to locality to explore how the property can be studied in multiple populations, in the absence of a method of objective fitness evaluation. Chapter 5 will introduce this area, providing a general background on coevolution in EAs.

Part III

Coevolutionary Systems

Chapter 5

Coevolution and Representations

Contents

5.1 Chapter Motivation

Coevolutionary algorithms (CEA) generalise evolutionary algorithms to problems where the quality of candidate solutions can only be expressed as a relative property. Examples arise in interactive situations where the value of individual sub-components cannot be objectively measured. One analogy is the scenario of ranking tennis players. It might be assessed that an individual player serves accurately, or has excellent stamina. However the outcome of a game will be determined by the worth of these abilities evaluated against a particular competitor. Furthermore, if the player is paired in a game

of doubles, the outcome will not only depend on the player's own capability, but will include collaboration with a partner in a team. Standard single population evolutionary algorithms carry out optimisation based on an objective measure of fitness that defines the quality of a solution. In a coevolutionary algorithm, such objective evaluation cannot be carried on a single individual and must instead incorporate two or more. This distinction has ramifications for concepts such as progress and convergence, as will be described throughout this chapter.

Studies of the genotype to phenotype map in the coevolutionary literature have primarily been carried out in the context of adaptive representations - that is, on coevolving the structure or parameters of a representation (Jong, 2003). Few analyses exist that directly consider the effect of biases due to genotype to phenotype mapping in a coevolutionary system (Wiegand *et al.*, 2002). A feasible explanation for this situation is that, as a relatively new technique, the majority of theoretical work on coevolutionary algorithms has addressed biases in more established bit string and direct encodings rather than complex representations such as trees and graphs. Coevolution is frequently *applied* using complex representations, for example through neuroevolution (Moriarty and Miikkulainen, 1998; Stanley and Miikkulainen, 2004) and genetic programming (Angeline and Pollack, 1993a; Haynes *et al.*, 1995). However, from the perspective of the user of evolutionary algorithms, there is little guidance to predict the behaviour that will occur in such systems. To begin to address this gap, we now move to examine the role of locality in artificial coevolution.

Because coevolutionary algorithms exhibit a number of fundamentally different behaviours to their evolutionary counterparts it is necessary to first provide some additional background. This chapter will highlight these differences and provide a review of previous work. The discussion is intentionally general, aiming to clarify concepts and avoids all but high level references to the architecture or design of particular algorithms.

5.2 Chapter Outline

The purpose of this chapter is to introduce the subject of artificial coevolution, then to discuss whether the ideas from Rothlauf's framework for EC representations described in Chapter 2 still apply. In the first section, an overview is provided of coevolutionary algorithms, from their historical origins to the current perspective. The second section summarises some of the key points of failure that have been identified in coevolutionary systems: the so-called 'coevolutionary pathologies'. The third part then provides relevant mathematical definitions, referencing a recent theoretical framework by Popovici *et al.* (2012), and addresses the topic of representation in coevolving programs.

5.3 What are Coevolutionary Algorithms?

In common usage, coevolution in biology refers to evolution between strictly separate populations, where the evolutionary change of individuals in one population affects another, which in turn affects the first (Janzen, 1980). In standard artificial coevolution, entities are divided amongst discrete populations in the same way. Figure 5.1 gives high level pseudocode for the structure of a canonical coevolutionary algorithm in multiple populations. In the example given, coevolution occurs simultaneously (all members are evaluated concurrently).[1] The interacting genotypes are drawn from separate populations and interactions with the same population are precluded. Coevolution then results in a set of interactions, the outcomes of which determine those genotypes from which the next generation is evolved.[2] *Thus, the aim of a well-designed coevolutionary algorithm is to provide an efficient search, when the space of possible interactions is large.*

```
1. Initialise a set of populations containing coevolving genotypes.
2. Interact a subset of genotypes drawn from the populations.
3. Assign fitness values from the interaction outcomes.
4. Select from amongst the best genotypes.
5. Apply search operators to selection to breed the next generation.
6. If termination criteria not reached, repeat from 2.
```

FIGURE 5.1: *Pseudocode for a canonical coevolutionary algorithm.*

5.3.1 Historical Development: Biological Origins

Formal algorithms to exploit the concept of coevolution for problem solving were not widely popularised until the pioneering work of Hillis using a genetic algorithm on the development of minimal sorting networks (Hillis, 1990). Hillis's stated intent was to increase the diversity of evolved solutions by including multiple coevolving populations, deriving inspiration from Hamilton's earlier work on parasitism in biology (Hamilton, 1980). As in biological systems, coevolving entities possess relations ranging between diametric opposition, to contexts which are fully mutualistic. The two traditional viewpoints in coevolution follow these extremes. In *competitive coevolution*, calculating fitness conventionally involves defining the performance of an entity over one or more tasks, relative to that of adversaries. In *cooperative coevolution*, entities collaborate such that their fitness is formed by some *collective* performance over a problem.

[1] More generally *asynchronous* coevolution also exists where populations interact at different rates.

[2] It is important to recognise that this usage is conventional and should not limit our thinking of what granularity in interactions are appropriate for a particular problem. An argument can be made for usage of the term coevolution when internal to a population, or between sub-components of a particular entity - in artificial coevolution, what matters is where and when partitioning and evaluations occur. In this work, we will restrict our usage to coevolution across multiple populations.

One of the most cited historical inspirations within the field of biological coevolution is the work of Dawkins and Krebs (1979). The lecture, held at the Royal Institute, advocates the role of arms races in the directional increase in complexity apparent between species. In particular, the work provides a classification of arms races, addressing their importance both between species, and internal to species between similar and dissimilar members. The emphasis placed by biologists at this time on *arms races leading to more complex adaptations* has been cited as justification for initial work on competitive coevolutionary algorithms and is an argument still prevalent in the current literature, see for example Cartlidge and Bullock (2004), Kim *et al.* (2004), Drezewski and Obrocki (2009). By contrast, the design of cooperative coevolutionary algorithms can be suggested to take its origins in the interface between evolutionary biology and game theory, in particular in the theory of evolutionary stable strategies and the evolution of cooperation (Axelrod, 1987).

Other motivations for competitive coevolutionary algorithms were put forward in the field of machine learning (Angeline and Pollack, 1993b). Foremost amongst these arguments was the familiar difficulty of defining an objective fitness function in a complex domain. CEA promised a method of circumventing this issue, by stipulating an evolutionary framework that enabled problems to be solved purely through competitive learning. Flagship results at this time such as the neuro-evolution of a human-competitive backgammon player, '*Neurogammon*', illustrated the potential of self-play as an unsupervised learning technique (Tesauro, 1989). Initial theoretical work was carried out by Kauffman, in his treatment of coevolutionary algorithms as a dynamical system (Kauffman and Johnsen, 1991). Also of note are early developments by Paredis addressing the issue of coevolutionary lifetime memory and learning, where individuals improve over the course of a single generation (Paredis, 1994, 1996, 1997). Subsequently, the first application of cooperative coevolution to function optimisation was established by Potter and De Jong, using the idea of multiple co-evolving entities to separately develop different variables (Potter and Jong, 1994). They assessed performance over four benchmark functions from standard GA optimisation, assigning function variables to distinct, coevolving sub-populations. The fitness of these sub-components was then evaluated collectively, considering their contribution as a whole to the complete solution.

It is apparent from this body of early work that the notions of competitive and cooperative evolution are deeply-embedded in the literature. This has had a significant impact on the direction taken by the field; competitive and cooperative coevolution have historically been seen as distinct approaches. Authors typically presented qualitative arguments based on either the *arms race effect* (the initial motivation behind competitive coevolution) or *sub-dividing the problem domain* (the initial motivation behind cooperative coevolution). However, in practice the whole range of mixed 'social'

behaviours exists between these two problem solving approaches. Problems treated by artificial systems may include interactions that are neither exclusively cooperative nor competitive. The terms cooperative and competitive coevolution are therefore merely convenient labels to describe the prevailing form of interaction that takes place in a particular system.

5.4 Coevolutionary Pathologies

Historical applications of coevolutionary algorithms rapidly diversified beyond function optimisation into areas such as competitive robotics (Floreano and Nolfi, 1997; Floreano *et al.*, 1998), strategy development (Lindgren and Johansson, 2001), artificial life (Sims, 1994) and predator-prey pursuit simulations under GAs (Cliff and Miller, 1995b). Interest in the effectiveness of cooperating agents subsequently led to studies on the development of coordination strategies governing co-evolved behaviour (Nolfi and Floreano, 1998), gaining ground within the wider context of coordination problems in multi-agent learning systems.

Despite this initial promise and some success stories, it became evident that coevolutionary algorithms exhibit complex dynamic behaviour and a number of 'pathologies' that hamper their application in optimisation and problem solving. These difficulties parallel the problems introduced by the '*Red Queen Effect*', derived from Valen's hypothesis in coevolutionary biology (Valen, 1973). Valen's hypothesis implies that because fitness in a biological species is a function of that of competing species, improvements to fitness must necessarily come at the expense of others. This implies a 'zero-sum' type interaction:

> *"We can think of the Red Queen's Hypothesis in terms of an unorthodox game theory. To a good approximation, each species is part of a zero-sum game against other species. Which adversary is most important for a species may vary from time to time, and for some or even most species no one adversary may ever be paramount. Furthermore, no species can ever win, and new adversaries grinningly replace the losers."*
>
> **- Valen (1973)**

In coevolutionary algorithms, the interactive nature of the fitness function leads to uncertainty over how to define a solution, how to measure progress and subsequently how to design a consistently useful algorithm.

This difficulty in defining progress arises because the relationships between solution quality are frequently intransitive. Informally, an intransitive relationship in this context

can be illustrated as the idea that, given three potential solutions A, B and C, the relations B>A and C>B do not imply C>A. Thus coevolutionary runs may break down into *cycling* behavior. The result is populations that repeatedly revisit points on the search space without reaching any fixed equilibrium point. One of the oldest approaches to measuring this is that of Cliff and Miller, frequently cited for presenting metrics and techniques to analyse the progress of a competitive coevolutionary algorithm with respect to previous archived solutions. Notably, this includes the CIAO plot (Current Individual versus Ancestral Opponents), a method of visualising performance against all previous opponents (Cliff and Miller, 1995b). However, over many generations the number of evaluations required to support this becomes impractical.

Further issues in coevolutionary performance were presented by Watson and Pollack, defining the notions of *loss of gradient* and *focusing* (Watson and Pollack, 2001). Loss of gradient refers to the issue that competitive coevolutionary algorithms stagnate when candidates are presented with no challenging opponents. This stagnation, or *disengagement* between populations, occurs where no new selection requirement is provided and hence there is no pressure to improve. The term *focusing* was introduced to describe an effect where adaptation is specialised against the known set of visited adversaries, creating a 'fragile' solution-strategy unable to compete over the whole search space. The problem is analogous to overfitting in machine learning and is a particular issue when there are multiple (phenotypic) dimensions along which an algorithm must optimise. Practical difficulties in coevolutionary computation continued with the identification of issues such as *relative overgeneralisation* (Wiegand, 2003). The term *relative overgeneralisation* refers to an effect that can occur in cooperative problems. Consider a 'general purpose' candidate, which provides a small outcome when evaluated (paired) alongside many others in the search space. The candidate can appear to outperform alternatives that are more specialised, but that give a higher average outcome. Therefore, a candidate that performs exceptionally alongside a small subset of partners may be outperformed by a candidate with poorer average performance, but which has a wider range of partners with which it cooperates acceptably well. Consequently, the general purpose candidate is more survivable under selection than the specialised candidate, yet does not yield the best outcome overall. Wiegand notes that, subject to the *solution concept* desired, either property (robustness vs. specialisation) can be useful for the problem at hand (Wiegand and Sarma, 2004).

In summary, we can see that coevolutionary algorithms were originally motivated by two key advantages: *to enable competitive learning for problem domains where objective fitness function definitions are difficult or impossible*, and *to take advantage of structure in problems by using a framework which decomposed them into coevolved sub-elements.*

Despite a wide range of potential applications, practical implementations have been fraught with difficulties, the so-called coevolutionary pathologies:

- *Cycling*
- *Loss of Gradient and Disengagement*
- *Focusing*
- *Mediocre Stable States*
- *Relative Overgeneralisation*

These issues are known to arise because fitness is only derived relative to a particular subset of individuals, giving no explicit ordering on what constitutes a good or bad solution. The next section reviews the theoretical foundations and approaches developed to address these pathologies.

5.5 Coevolutionary Algorithms and Solution Concepts

In order to resolve the pathologies that were observed in coevolutionary algorithms, authors moved to consider a more robust range of *solution concepts*. In this context, a solution concept is *a partition on a discrete search space, that defines a subset of solutions*. Importantly, the solution subset is only defined 'relative' to the space to which that concept is applied. For CEA, this search space is the space of possible interactions between evolving genotypes. In the early days of CEA, it was observed that algorithms converged on sub-optimal attractors, typically associated with solution concepts in game theory such as *Nash equilibria*. Therefore, solution concepts have been historically linked to the idea of ensuring *monotonic solution progression*, that is avoiding the problem of revisiting and cycling between inferior solutions.

5.5.1 Types of Solution Concept

In the context of Evolutionary Computation, coevolutionary solution concepts are defined with respect to the outcomes of interactions between the subset of genotypes that comprise a solution. Popovici *et al.* (2012) recently drew together a list of the competitive solution concepts that have been considered in EC, which we summarise in Table 5.1. These terms are closely related to those in formal game theory.

TABLE 5.1: *Solution concepts in competitive coevolutionary algorithms (Popovici* et al.*, 2012)*

Solution Concept	**Description**
Best Worse case	Maximise minimum outcome over interactions with all opponents.
Simultaneous Maximisation of All Outcomes	Maximise outcome over all possible opponents simultaneously.
Maximization of Expected Utility	Maximize expected outcome against a randomly selected opponent.
Nash Equilibrium	Equilibrium such that neither mixed strategy can change without receiving lower outcome.
Pareto Optimal Set	Non-dominated front over all tests.
Pareto Optimal [Minimal] Equivalence Set	As Pareto-Optimal Set, but reduced to discard candidates which receive the same outcome over all tests.

Simultaneous Maximisation of All Outcomes

The simplest competitive solution concept is referred to as *Simultaneous Maximisation of All Outcomes*. In test-based coevolutionary algorithms[3], candidate solutions coevolve alongside one or more populations of 'tests'. Simultaneous maximisation of outcomes simply refers to the set of solutions where, over all possible tests the outcome of that test is to be maximised. That is, moving to any other candidate will result in an explicitly lower outcome on one or more tests. Popovici et. al. note that although this is an intuitively useful idea, it has limited applicability. There exist many problems that do not possess a solution, where a candidate cannot be found that achieves the best outcome against all tests. The solution concept assumes the existence of a universally 'best' strategy.

Maximisation of Expected Utility

Maximisation of Expected Utility originates from the concept in competitive coevolution of maximising performance against a random strategy (an opponent). Members of this set have the highest average outcome when considered against all other evolvable strategies. This idea is constrained by the assumption that all tests are equally valuable (Ficici, 2004). In practice, for many situations we can say that a solution that performs well on average against all tests is not the most valuable. It may provide a high outcome against mediocre candidates, but fail against a smaller subset of more competitive, specialised opponents. Algorithms have been designed with the intention

[3]Test based coevolutionary algorithms will be considered in more detail in Chapter 6

of ensuring monotonic progress in these simple cases: *Competitive Covering* (Rosin, 1997) and *MaxSolve* (Jong, 2005). The covering competitive algorithm acts on two populations of strategies. The algorithm includes new learned strategies only when they beat or 'cover' all previous opponents in the other population. Similarly, the MaxSolve algorithm was engineered to generate an archive of strategies which solve an increasing number of tests from the opposing population.

Nash Equilibrium and Pareto-Optimality

More sophisticated concepts were introduced to work around these disadvantages. The existence of a *Nash Equilibrium* for any finite game is guaranteed by John Nash's famous proof (Nash, 1950). Informally, a Nash Equilibrium is formed from a set of strategies, when the strategy selected by each player is the best response given knowledge of all opposing strategies. Convergence on the Nash Equilibrium can be ensured in coevolutionary algorithms by the use of *Nash memory*, an archiving mechanism which increasingly approximates the mixed Nash strategy (Ficici and Pollack, 2003). Ficici advocated the use of Nash Equilibria as a monotonic solution concept in coevolutionary algorithms, though drawbacks are apparent: Nash equilibrium may be strictly dominated by other solutions and do not necessarily possess the highest objective outcome over any subset of candidates in the solution space. One alternative is to ensure that solutions are *Pareto-Optimal*. The Pareto-Optimal solution concept is defined in coevolutionary systems relative to subsets of coevolved test candidates, where each equivalence solution set is given by those candidates which exactly 'solve' the same subset of tests. Thus, a Pareto-Optimal equivalence set contains those solutions which have an identical outcome over all known tests. Archives have been introduced which populate the Pareto-Optimal front of solutions and this concept has been shown to ensure a form of monotonic progression (Jong, 2007). De Jong also defines a separate solution concept, the *Pareto-Optimal Minimal Equivalence Set*, where the constraint to contain all candidates solving a given subset is relaxed. The equivalence set then contains a single instance from the 'identical' candidates which solve a particular set of tests.

Best Worse Case

A further, more recently introduced solution concept for coevolutionary algorithms is the familiar game-theoretic idea of maximising minimum outcome. The minimum *outcome* in the competitive case is the smallest outcome obtained by a candidate solution over an associated set of coevolved tests. Finding candidates with the largest minimum outcome can be considered important under situations where the impact of obtaining low values

is high (Popovici *et al.*, 2012). The less common inverse situation, or '*maxmin*', is to consider the set of solutions providing the lowest, maximum outcome (that is ensuring a certain level of return is *potentially* possible).

5.5.2 Criticism of Solution Concepts and Coevolutionary Progress

The idea that it is necessary to ensure monotonic progress towards a particular solution concept has been challenged. Miconi (2009) questions the lack of distinction drawn between *'historical'* and *'global'* progress in the application of coevolutionary algorithms to problem solving. He defines historical progress as progress delivered solely relative to past opponents (previous candidates visited in the search space) and global progress as progress measured objectively to the solution concept. The argument is that past historical progress, that is evolving strategies which are sequentially better against previous opponents, does not necessarily guarantee global progress. Miconi defines a *globally monotonic solution concept* as a situation in which this condition does hold.

In more recent work, Popovici and Jong (2009) summarises several issues in striving for *global* monotonic improvement. Firstly, Ficici has shown that globally monotonic progress is only possible for a subset of the known solution concepts characterised in coevolutionary computation (Ficici, 2004). For a problem with a fixed solution concept (as is typically the case) monotonic progress may not be achievable. Secondly, even for those solution concepts for which monotonic progress is theoretically possible, an algorithm to achieve this may not be known. Finally, such an algorithm may not be computationally useful, due to memory requirements or slow convergence.

It is also worth a brief aside to address the question of how the No Free Lunch theorem for optimisation applies to these notions of coevolutionary solution concept. Wolpert and Macready extended their seminal work to define a framework, generalisation optimisation (GO), that demonstrates that coevolutionary free lunches exist between pairs of algorithms (Wolpert and Macready, 2005). Their result did not extend to distinguishing between free lunches over algorithms using particular solution concepts - more recent work pursued by Travis and Tauritz describes this as an open question, identifying a free lunch for the maximisation over all outcomes solution concept (Service and Tauritz, 2008) and Pareto-coevolution. For these solution concepts it was demonstrated that *there do exist algorithms which have, on average, better global coevolutionary progress in terms of their number of fitness evaluations* (Service, 2009). Theoretical work in this area is still active.

5.5.3 Summary

We have reviewed the notions of solution concepts, progress and monotonicity in coevolutionary algorithms. A solution concept is a partition that can be applied to any subset of the whole search space. When that subset is not equal to the whole space, progress towards the solution concept may be described as local progress, rather than global. The solution concepts currently addressed within the coevolutionary literature are:

- *Best Worse Case*
- *Simultaneous Maximisation of all Outcomes*
- *Maximisation of Expected Utility*
- *Nash Equilibrium*
- *Pareto Optimal Set*
- *Pareto Optimal Minimal Equivalence Set*

Properly specifiying which solution concept applies for a problem is one of the main considerations when designing a coevolutionary algorithm.

5.6 Terminology and Mathematical Definitions

Attempts to classify coevolutionary algorithms in the context of this work are complicated by differences in terminology. For example, the usage of the term representation in coevolutionary algorithms often refers not only to the data structure used in genotype to phenotype mapping, but can refer to how individuals are distributed throughout coevolved populations. In cooperative coevolutionary literature, the term has been applied to mean 'problem representation', that is how sub-components (which may be separate individuals) can be aggregated up to give a whole solution. The following definitions are adapted from Popovici *et al.* (2012), using the notation set out in Chapter 2.

5.6.1 Coevolutionary Problems

Coevolutionary problems can be said to take place on an *interactive domain*. Diverse examples of such domains exist, such as classical board games, collaborative robot teaming, auctions and bidding contests.

Definition 5.1 (Interactive Domain)**.** An *interaction* is a tuple of phenotypes $(p_1, p_2...p_n)$. An interaction function is a function $u : P_1 \times P_2 \times ... \times P_n \rightarrow \mathbb{R}$, where $p_i \in P_i$.. The *outcome* of an interaction is the value $u(p_1, p_2...p_n)$. An *interactive domain* is a set of *interaction functions* that act on n *phenotype sets*, $P_1, P_2...P_n$.

Remark: The interaction function is similar in form to a payoff matrix in standard game theory. Payoff matrices store the outcomes of interactions between strategies used by the players of a game. Some coevolutionary algorithms use only a single interaction function and one set of outcomes, but there is no formal restriction (for asymmetric situations, more than one interaction function may be required).

Following from this, two classes of coevolutionary problem can be described, *co-search* and *co-optimisation*:

Definition 5.2 (Co-search Problem)**.** A *co-search* problem is a set of potential solutions to an interactive domain S derived from the phenotypes $p \in P$, for which there is a *solution concept*. The solution concept is a binary partition on S into solutions O and non-solutions.

Remark: How solutions are obtained from the phenotypes depends on the problem. A solution may comprise phenotypes across several different phenotype sets, or even mixed combinations of phenotypes (a selection of phenotypes chosen at certain probabilities). This enables the inclusion of more complex solution concepts, such as Nash Equilibrium.

Definition 5.3 (Co-optimisation Problem)**.** A *co-optimisation* problem is a set of potential solutions S derived from the phenotypes $p \in P$, for which there is a partial order $\leq$ on S

Remark: This definition is a specialisation of the co-search problem. Ordering the potential solutions in S using $\leq$ implies that there exists a subset of maximal elements in S. In Figure 5.2, an illustration is given to clarify Popovici's definitions. The figure shows a set of potential solutions constructed from a number of phenotype sets, according to a set of interaction functions. For a co-search problem, potential solutions are partitioned using a solution concept. For a co-optimisation problem, an ordering applies that determines the desired solutions.

The usefulness of this framework in the context of the current work is that it gives a robust description of the classes of problem that coevolutionary algorithms are designed to solve. It also helps to clarify the main distinctions between evolutionary and coevolutionary algorithms (interaction functions that can act over distinct sets of phenotypes).

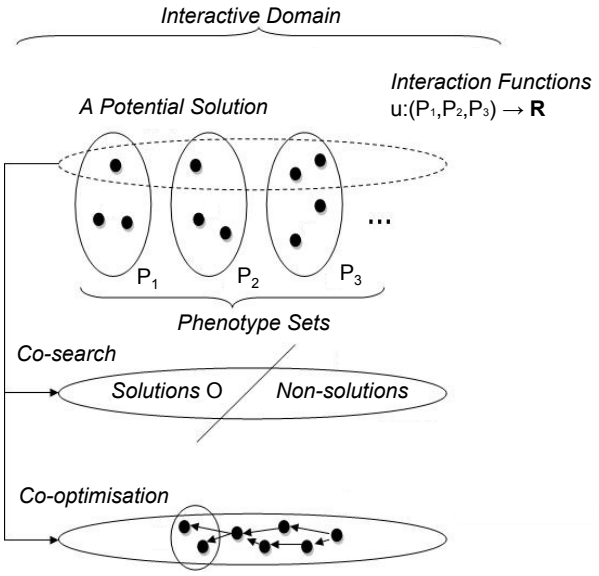

FIGURE 5.2: *A diagram of a generic coevolutionary problem. Illustrates the distinction between co-search and the specialised case, co-optimisation.*

5.7 Discussion and Concluding Remarks

What do these differences between evolution and coevolution imply for our understanding of the GPM? Using the concepts cited in the previous section, it can be seen the definitions given in Chapter 2 remain valid. This is the case because the definitions in Chapter 2 make no assumptions about the mapping between phenotype and fitness. Therefore concepts such as redundancy and locality can still be expressed when working with coevolutionary algorithms, providing they use metrics defined before phenotypes interact.

One limitation of this viewpoint is that representations used in co-search may possess an extra level of complexity beyond classical evolutionary search, because of the potential use of multiple representations simultaneously. Each representation used in a co-search problem will have its own properties. For example, a strongly local genotype to phenotype map may coevolve together with a weakly local genotype to phenotype map, or vice versa. In the following chapters, we will restrict our scope to symmetric representations, where only a single class of genotype to phenotype map is present. Finally, although similar properties of the genotype to phenotype map can be defined in coevolutionary algorithms, there is no firm basis to believe that the responses observed in single population systems can be inferred for coevolution. For example, in problems with a positive fitness gradient to the optimum, weak locality may randomise a search (Rothlauf, 2006). However, locality in this sense only occurs in a CEA if a small phenotypic change induces a small change for all interactions. To the best of the author's knowledge, there exists

no work extending the notion of 'smoothness' in GP fitness landscapes to co-search. An alternative framework for locality in CEA is discussed in Chapter 8.

5.8 Chapter Summary

This chapter has provided an overview of the subject of coevolutionary algorithms, covering the notion of solution concepts, coevolutionary pathologies and the issues surrounding progress in coevolutionary systems. Current definitions in artificial coevolution (Popovici *et al.*, 2012) were related to the concepts introduced in Chapter 1.

- Coevolutionary problems are defined over an *interactive domain*.
- Problems can be divided into two classes, *co-search* and *co-optimisation*.
- Potential solutions are aggregated from a collection of *phenotype sets* using a set of *interaction functions*.
- Goal conditions are determined by selection of an appropriate *solution concept* for the problem.
- Concepts from the theory of representations (locality, neutrality etc.) in single population evolutionary systems can be described within this framework.

The ideas reviewed here may appear quite abstract, but form the basis of a necessary context to analyse genotype to phenotype maps in artificial coevolution. In the following chapters, we will make use of these definitions and more concrete examples will be explored.

Chapter 6

Locality in Test-based Coevolution

Contents

6.1 Chapter Motivation

The previous chapter reviewed some of the fundamental distinctions that exist between coevolutionary and evolutionary algorithms. Having provided tools to analyse locality in evolutionary systems in Part I, it is now possible to extend our study of genotype to phenotype locality to CEA. Because there is little theoretical evidence to indicate directly how locality affects progress in CEA, it is necessary to first select a class of coevolutionary problems in which the property can be examined empirically. One frequently encountered form of co-search problem in engineering is the aim of optimising a solution against a large collection of tests. For example, designing a novel system component may require many tests in different environmental contexts, typically too many to allow for an exhaustive set of trials. In the case of search-based software development, tests might literally comprise unit tests for the functionality of a particular segment of code (White *et al.*, 2011). On a more abstract level, in a scientific experiment tests can be presented with the aim of refuting a particular hypothesis. Tests act in opposition to other elements, therefore test-based coevolution is a strictly competitive situation.

A more precise description of test-based problems can be stated using the co-search framework described in the previous chapter. The interactive domain (Definition 5.1) contains a set of *candidates* C and set of *tests* T. The goal in a test-based problem is to attain a set of candidates that satisfy the specific solution concept appropriate to that interactive domain. Candidates are evaluated using an interaction function against a set of tests. We say that *single outcome* problems have an interaction function of the form $u : C \times T \to \mathbb{R}$. A *binary outcome* problem is a specialisation of this in which a test evaluates only to a pass or failure $u : C \times T \to \{1, 0\}$.

Solving test-based co-optimisation problems using coevolutionary algorithms has been a central focus for CEA practitioners and theorists (Rosin, 1997; Jong and Pollack, 2004; Jaskowski and Krawiec, 2009). Analogous to competitive coevolution in nature, the intention is to generate genotypes that represent successively more capable candidates and more challenging tests. Test-based coevolution therefore captures much of the complexity of the general problem of co-search and provides a set of appropriate problems. To give proof of principle, this chapter will concentrate on locality in coevolutionary problems that use binary string representations. Representations suitable for the coevolution of programs will be considered in Chapter 7.

6.2 Chapter Outline

A hypothesis of this thesis is that the locality of artificial genotype to phenotype maps will significantly influence the behaviour of coevolutionary algorithms. This chapter tests this assumption in detail. The aim is to confirm whether representation locality has an effect on performance in co-search problems and, if so, to what degree. This is addressed by posing the question, *"Does genotype to phenotype locality contribute to coevolutionary algorithm performance?"* The domain used to explore this question is *test-based competitive coevolution.* Section 6.3 of this chapter addresses experimental design considerations when analysing coevolutionary systems. Section 6.4 then describes two test-based benchmark problems, the Compare-on-one and Greater-than number games. Section 6.5 details the measures of progress and locality used for these problems. This includes some necessary background on Pareto-optimality and archives. Sections 6.6 and 6.7 then use the games to analyse the interaction between mapping locality, performance and a coevolutionary pathology, disengagement. The remaining sections discuss and conclude.

6.3 On Experimental Design in Coevolutionary Algorithms

Empirical research on the contribution of any given factor in the performance of a coevolutionary algorithm is complicated by several issues. These include:

1. *Coevolutionary pathologies.*
2. *Selection of adequate benchmarks.*
3. *Measurement of objective progress.*

Coevolutionary pathological behaviour, such as cycling, can occur even in simple coevolutionary systems. From the review in Chapter 5, it can be seen that it is not clear from the present literature how biases from representation affect pathological behaviours. To ascertain how a given factor in a coevolutionary algorithm interacts with the pathological behaviours shown by coevolutionary algorithms is a major challenge; see for example the thesis of Cartlidge (2004) on coevolutionary disengagement. Some pathologies can be mitigated by the addition of components intended to ensure progress towards a particular form of solution, such as archives; this consideration reinforces the need for experiments to be conducted with reference to a clearly defined solution concept.

Isolating where poor performance can be attributed to a particular factor or pathology requires carefully designed benchmarks. It is likely that the real response of a CEA's

performance to factors such as changes in redundancy, locality or scaling is coupled: again, the extent to which factors are co-dependent is not clear *apriori*. Finally, it is necessary to define how to objectively measure progress. By definition, problems in coevolutionary algorithms lack a suitable objective external measure against which to measure performance.

6.3.1 Proposed Empirical Framework

Three steps are necessary to measure how a factor such as the genotype to phenotype mapping affects the performance of a coevolutionary algorithm:

1. A benchmark and algorithm are identified in which measurable progress under a given solution concept is known to occur. A metric is defined that approximates progress on the underlying objectives.
2. Observations are made of how approximate progress changes with respect to that factor independently, in the absence of variation in any other contributing factors.
3. The sensitivity of progress under that factor is assessed relative to other explanatory factors and parameters.

Performance and Progress Metrics

The convention in Evolutionary Computation concerning monitoring progress is to measure performance with respect to fitness evaluations. The assumption in this measure is that fitness evaluation is the dominating, most computationally expensive component of the search algorithm. The analogous measurement in a coevolutionary system is the number of evaluations required in computing the interaction function. Given these measures of performance, one can either consider the state of the population after a fixed number of evaluations, or the number of evaluations required to converge on a solution of particular quality.

Because coevolutionary algorithms may return to poorer states, performance will be measured over fixed intervals. Overall performance can then be determined by considering the likelihood of the algorithm succeeding in reaching the desired state. This goal state is defined by reference to the solution concept and tested after a known amount of computational effort, over a sufficient number of independent runs to give statistically significant results. Measuring the rate of improvement towards a solution concept can be accomplished by defining a measure of distance to the solution set, if it is known. If the solution set is not known, then comparison must be made against a collection of

exemplary solutions or tests, for example in competition with a human expert. To avoid the subjective component this introduces, the benchmarks considered in this thesis will be restricted to those with known optimal solutions or where comparison can be made against close to optimal solutions.

Choice of Benchmarks

Where possible, benchmarks will be selected from previous literature. Benchmarks in CEAs are often associated with a particular class of representation, so variations may be necessary to examine benchmarks within the context of different kinds of genotype to phenotype map. Additionally, benchmark problems are usually associated (explicitly or implicitly) with a particular kind of solution concept, as defined in Chapter 5. It will made clear what alterations, if any, have been made to benchmarks when referenced and which solution concept applies.

It should also be noted that there is an inherent trade-off in benchmark choice between realism and transparency. Benchmarking using complex problems may help to ensure techniques are tested on a domain 'closer' to those of practical interest. However, this comes at the expense of our ability to interpret results and isolate how particular properties of the system have affected performance. Because the focus in this study is on the latter this dictates a preference for simpler benchmarks, so that GPM properties can be studied in detail.

6.4 Test-based Benchmarks: Number Games

Number games (Watson and Pollack, 2001) are constructed coevolutionary problems that use a sufficiently simplified representation and interaction function so that detailed analysis and comparison of different algorithms is practical. In the following test-based problems, the coevolving entities are represented by n dimensional fixed length vectors $c = [c_1...c_n], t = [t_1...t_n]$ of binary, integer, or floating point values. There exists a known target vector or set of vectors that comprises the solution set. The progress of the search is therefore straightforward to interpret. By varying the interaction function u that describes the outcome of interactions between candidates and tests, different coevolutionary scenarios can be modelled. Candidate vectors receive a outcome of $u(c, t)$ and test vectors receive a outcome of $1 - u(c, t)$. The experiments in this chapter use modified versions of two established numbers games, *Compare on One* (CO1) and *Greater Than* (GT).

6.4.1 Compare-on-one game (CO1)

Compare-on-one (CO1) is a binary outcome number game proposed by Jong (2007). The objective of CO1 is to maximise all dimensions of the candidate vector, which is guided by a binary interaction function. A candidate is awarded a score of 1 if it has a greater value in the largest dimension ($maxComponent$) of the test vector:

$$u(c,t) = \begin{cases} 1 & c_i > t_i \\ 0 & c_i \leq t_i \end{cases} \qquad i = maxComponent(t) \tag{6.1}$$

CO1 is known to be challenging for canonical coevolutionary algorithms because progress must be achieved in all dimensions, but comparison is only permitted on single element of a vector in each interaction. CO1 therefore models a property of larger scale problems, where a test may only examine a particular aspect of a system's performance. This promotes overspecialisation on a sub-optimal set of candidates or tests.

The approach in this chapter uses a modified instance of the original CO1, that has been altered to include an indirect genotype to phenotype map. Each genotype is represented as a 30 bit word. The word is divided into two parts and the corresponding substrings mapped to a pair of 15 bit unsigned integers (the phenotype), by a binary to integer encoding (the genotype to phenotype map). Therefore, each candidate or test corresponds to a pair of integer values in the range 0 to 32767.

6.4.2 Greater-than game (GT)

The Greater-than game (GT) was originally introduced by Watson and Pollack (2001) and later adapted by Cartlidge (2004) to explore the pathology of disengagement in coevolutionary systems. In the original single outcome game, genotypes were fixed length binary strings and the interaction function is given by

$$u(c,t) = \begin{cases} 1.0 & \Sigma_c > \Sigma_t \\ 0.5 & \Sigma_c = \Sigma_t \\ 0.0 & \Sigma_c < \Sigma_t \end{cases} \tag{6.2}$$

where Σ_c and Σ_t denote the sum of elements in the candidate and test vector respectively. Cartlidge then applied a mutation bias to the test population that increased the likelihood of generating larger values by mutation, by increasing the probability of generating a one during each bit flip. In the presence of such a bias, tests increased

in difficulty much faster than the candidate population can adapt (Cartlidge and Ait-boudaoud, 2011). Disengagement then occurs when no distinction can be made between the set of evolving candidates, because all candidates perform poorly across the test set.

GT was modified to include a bit-string to integer mapping identically to CO1 in Section 6.4.1. However, using the same procedure to bias candidates as Cartlidge and Ait-boudaoud (2011) is not feasible, because larger genotypes may not correspond to larger phenotypes under the genotype to phenotype map. As an alternative method of producing disengagement, a negative bias was introduced to the test population. Before evaluation, each member of the test population is updated with probability ν. The updated tests are replaced by an equivalent test with both integer values reduced by a fixed penalty of 1000. In the absence of a positive selection gradient, the quality of tests will deteriorate over time, until disengagement occurs (the quality of candidates cannot be distinguished). Larger penalties and values of ν correspond to a stronger *asymmetric bias*.

6.5 Factors and Progress Metrics

6.5.1 Locality Measures for CO1 and GT

Measuring locality in CO1 and GT is more straightforward than is the case for the program representations described in Chapter 3, because of the simplicity of the fixed length binary to integer GPM. Recall that Rothlauf's original definition of locality (Equation 2.6) is not normalised. A normalised aggregative measure of locality is used here, L_N, based on Definition 2.7:

$$L_N = \frac{1}{|G|} \sum_{g \in G} \frac{1}{|adj_m(g)|} \sum_{p' \in adj_m(g)} \frac{d_P^+ - d_P(p, p')}{d_P^+ - d_P^-} \tag{6.3}$$

where d_G^+, d_P^+ and d_G^-, d_P^- are the maximum and minimum distances in genotype space and phenotype space respectively (Equation 2.2). The constant $d_P^+ - d_P^-$ is the largest possible difference in the phenotype space minus the smallest possible difference. The function $adj_m(g)$ returns the local neighbourhood of g, mapped into the phenotype space.

The sum given in Equation 6.3 is a more convenient metric for locality in these particular discrete phenotype spaces because it is normalised to assume the value 1 when all neighbouring genotypes have neighbouring phenotypes and 0 when all neighbouring genotypes give maximally different phenotypes. Values between 0 and 1 indicate intermediate levels of locality in the mapping.

TABLE 6.1: Inverse locality for a 3-bit binary, Gray and randomised encoding. Rows show the partial sum of L_N^{-1}, for each phenotype.

p	*Binary*		*Gray*		*Random*	
0	000	1.00	000	1.00	010	0.00
1	001	0.75	001	1.00	101	0.25
2	010	0.75	011	1.00	000	0.50
3	011	0.50	010	1.00	011	0.25
4	100	0.50	110	1.00	100	0.50
5	101	0.75	111	1.00	110	0.50
6	110	0.75	101	1.00	001	0.25
7	111	1.00	100	1.00	111	0.50
$\mathbf{L_N^{-1}}$	0.75		1.00		0.34	

For this case, where m is a bijective function, it is also possible to define the corresponding expression in phenotype space. We term this the inverse locality, L_N^{-1}:

$$L_N^{-1} = \frac{1}{|P|} \sum_{p \in P} \frac{1}{|adj_m^{-1}(p)|} \sum_{g' \in adj_m^{-1}(p)} \frac{d_G^+ - d_G(g, g')}{d_G^+ - d_G^-} \tag{6.4}$$

Equation 6.3 defines locality with respect to neighbours in the genotype space and Equation 6.4 with respect to neighbours in the phenotype space. This reflects the concept that it is possible to have a mapping that is strongly local in the inverse direction (phenotype to genotype) but not in the forward direction (genotype to phenotype).

Example: Locality in Gray Codes

For small genotype spaces, the measures of locality given in Equations 6.3 and 6.4 can be calculated directly. By way of example, consider the calculation for a reflected binary or Gray code. Gray codes provide a useful test case, because by definition all phenotypes (integers) have neighbouring genotypes. Under Equation 6.3, the inverse locality of a Gray code (i.e. for integer to bit string) is equal to one, by definition. If we make a change to the code the inverse locality may decrease accordingly. Table 6.1 contrasts L_N^{-1} computed for a three bit word in binary, Gray and a random permutation. The distance metric d_G is the Hamming distance and d_P is the absolute difference between integers. The maximum and minimum distance d_G^+ and d_G^- are 3 and 1 respectively. The partial sums in Table 6.1 can be seen as mean 'scaling factors' for each local neighbourhood.

From the values of L_N^{-1} in Table 6.1, it can be seen that the Gray encoding preserves the local neighbourhood structure when mapping from phenotype to genotype (i.e. L_N^{-1}), but the random and binary encodings disrupt it.

6.5.2 Measuring Performance

Performance and Solution Concepts

The solution in the GT game is an example of simultaneously maximising all outcomes: the aim is to find the candidate with maximum value in both dimensions (32768, 32768). For CO1, the solution concept is Pareto-optimality. A Pareto-optimal solution can be defined for test-based problems by considering each test as a separate objective.[1] The solution concept induces a strong dominance relation $\succ$ between each pair of candidates $(c, c') \in C \times C$, with respect to each subset of tests $T_s \subseteq T$:

$$\begin{aligned} c \succ c' \text{ such that} \quad & \forall t \in T_s,\ u(c,t) \geq u(c',t) \\ \text{and} \quad & \exists t \in T_s,\ u(c,t) > u(c',t) \end{aligned} \tag{6.5}$$

Removing the second condition gives the corresponding weak relation:

$$c \succeq c' \text{ such that } \forall t \in T_s,\ u(c,t) \geq u(c',t) \tag{6.6}$$

When the whole test set T is referred to, Equation 6.6 induces a partial order on the candidate set.[2] The dominance relation orders candidates into layers or *Pareto fronts*. Members of each Pareto-front are non-dominated $c \not\succ c'$ with respect to each other, over all tests.

The performance of an algorithm over a run is assessed by considering the objectively best set of candidates with respect to the solution concept. In these constructed problems, this is straightforward because the objective quality of candidates is known. As constructed here, the objective performance of a candidate can be evaluated as the minimum value in the pair of integers to which that candidate corresponds (Jong, 2007).

6.5.3 Iterated Pareto Coevolutionary Archive

In the CO1 experiment, the *Iterated Pareto Coevolutionary Archive* (IPCA) described by Jong (2007) was included. IPCA is an unbounded archive that maintains successive

[1] The situation can be compared with conventional multi-objective optimisation, where multiple objectives exist that must be satisfied. The distinction is that the number of tests is large, not necessarily known priori and that evaluating against all tests simultaneously is infeasible.

[2] Bucchi (2007) examined order in test-based problems, ordering tests using an alternative, geometric approach. Informally, a coordinate system was assembled that enables candidates and tests to be ordered with respect to a set of underlying objectives. However extraction of this underlying coordinate system is complex and has recently been shown to be NP-Hard (Jaskowski and Krawiec, 2011).

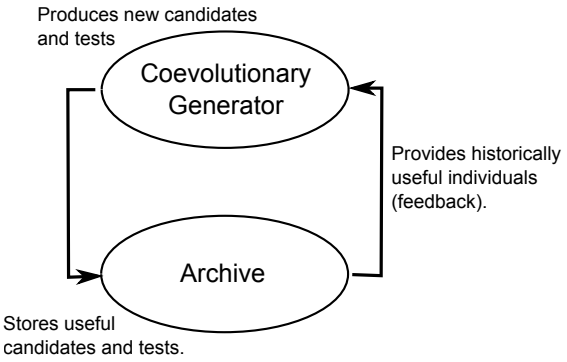

FIGURE 6.1: *A generic coevolutionary archive system.*

layers of Pareto-optimal candidates, with respect to currently known tests. The justification for including IPCA is that it ensures that monotonic progress will occur on this test problem.

Conceptually, under De Jong's model, the coevolving system has two coupled components: a generator, which features two distinct populations of candidates and tests, coupled to an archive for each population, which accumulates the Pareto-optimal front of solutions. The purpose of the generator is to iteratively produce new samples of candidates and tests. The purpose of the archive is store dominating candidates, and tests that are determined to be useful to establish this order.

Figure 6.1 gives a generic impression of this system. As more tests are accumulated, the quality of individuals in the candidate and test archives improves. Stored candidates converge on the objective best solution, which is the Pareto front of candidates defined with respect to the whole test set. The test archive retains those tests that have been required to distinguish the evolved candidates. The archive provides useful historical candidates and tests back to the generator to incorporate as potential parents in future iterations. This feedback mechanism is described in Section 6.6.3.

6.6 Experimental Setup

6.6.1 Mapping Construction

The effect of varying locality on progress in GT and CO1 was explored by generating sample mappings with different inverse locality. Following from the example in section (6.5.2), a simple greedy search was applied to generate sample encodings (Algorithm 3) from the standard reflected binary Gray code. The algorithm's input is a target locality T and a maximum tolerance (error margin) *tol* on T. The algorithm begins with a local Gray encoding and perturbs it until locality falls to the vicinity of the desired level.

Data: Target locality T, maximum tolerence tol
Result: Encoding m
Generate a binary code m;
$l \leftarrow L^{-1}(m)$;
while $|T - l| > tol$ **do**
  Swap a randomly selected pair of integer values in m to give m';
  if $|T - L^{-1}(m)| < |T - l|$ **then**
    $m \leftarrow m'$;
  end
end
return m

Algorithm 3: `GreedyPerturb`

Sets of 5 sample encodings were considered between $L_N^{-1} = (0.6 : 1.0)$, to a tolerance of ± 0.01.

6.6.2 Algorithm Configuration

A $\mu + \lambda$ coevolutionary algorithm was used as the generator for each experiment. The generator included a population of 20 candidates and a population of 20 tests, initialised uniformly at random. The simplest method of cross-evaluation, *complete mixing*, was applied to determine fitness. Fitness values were accumulated from the sum of evaluating all members of both populations against each other:

$$\begin{aligned} f(c) &= \sum_{t \in T} u(c,t) \\ f(t) &= \sum_{c \in C} 1.0 - u(c,t) \end{aligned} \tag{6.7}$$

Truncation selection was applied to the μ fittest genotypes. Offspring were then generated using a uniform mutation operator, with mutation probability q, to produce the remaining set of λ samples. The initial parameter choices, informed by Jong (2007); Cartlidge and Ait-boudaoud (2011) are given in Table 6.2. Sensitivity of the experimental outcomes to these choices is assessed in Section 6.7.

6.6.3 Archive Feedback

In the CO1 problem, at each generation before selection or operators are applied, the candidate and test populations are submitted to update the IPCA. Details of the archive updating procedure can be found in Jong (2007). In the system studied here, feedback was included by randomly selecting members of the archive to act as parents. Each new sample has a chance to be produced from an archive member, rather than the current

TABLE 6.2: *Initial algorithm and parameter configuration for CO1 and GT.*

Property	**CO1**	**GT**
Population Size	2×20	2×20
Mutation Rate q	0.1	0.1
Selection	$(\mu = 5) + (\lambda = 15)$	$(\mu = 5) + (\lambda = 15)$
Generations	100	100
Runs	500	500
Archive	IPCA	N/A
Archive Feedback	Linearly Increasing	N/A
Test Bias ν	N/A	0.5

population. The likelihood of this event is initially zero, but increases at a rate of 1% each generation. This ensured that in the later stages of search new genotypes are likely to be generated by the mutation of members of the archived Pareto front. The strategy will be referred to as linearly increasing feedback.

6.6.4 Measuring Disengagement in GT

Cartlidge and Ait-boudaoud (2011) advocated that if no distinction can be made in fitness between any members of the test set - the variance of fitness values is zero - the candidate and test populations can be said to be disengaged. Disengagement occurs when a population enters a state such that no search gradient can be determined with respect to the other coevolving components. To analyse whether there is a relationship between disengagement and locality, disengagement between the candidate and test populations was monitored throughout the GT game. At each generation, the likelihood of disengagement is estimated as the proportion of runs in a disengaged state at that point. The probability of disengagement $P(Disengaged)$ was estimated at each generation by taking the fraction of the total set of runs for which the variance of the candidate population fitness values was equal to zero at that point.

6.7 Results in Binary Number Games

6.7.1 Compare-on-one Results

Figure 6.2 illustrates the expected (arithmetic mean) performance in CO1 for two sample encodings ($L_N^{-1} = 0.51$ and 0.76) at the initial fixed mutation rate of 0.1. A comparison is shown against the performance using the reflected binary code, with and without archive feedback. The expected performance of a control run using random selection and no archive is also shown. Best performance was achieved by the Gray code and

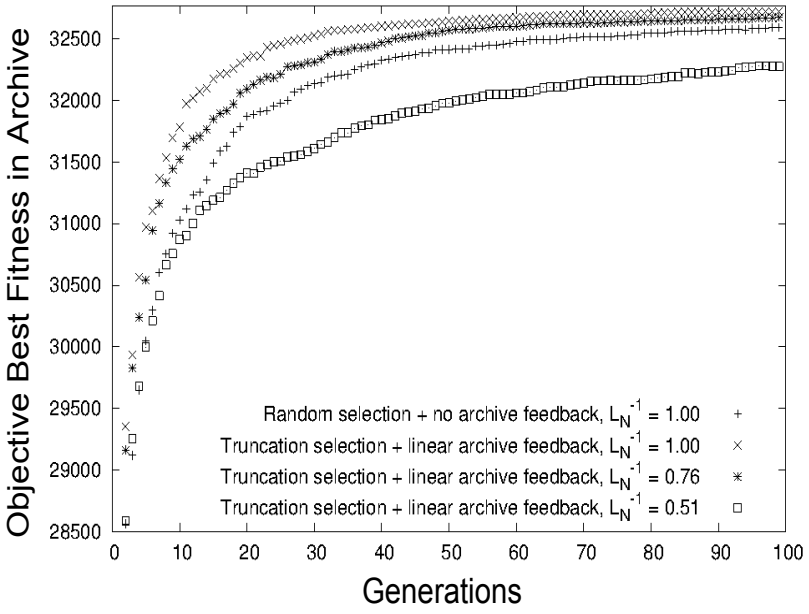

FIGURE 6.2: *Example convergence plots for CO1, with and without archive feedback.*

archive. Figure 6.3 then shows the response to varying encoding locality independently of other parameters. Algorithm 3 was run 5 times, to generate 5 sample sets each containing 11 encodings sampled at intervals from each run. The points shown represent the performance of a distinct encoding, measured at 50 generations. The trend indicates a strong correlation between locality and performance (Spearman coefficient $\rho = 0.98$, p-value 0.005, exact) which was observable over all the sample encodings. This suggests a nearly monotonic decrease in performance as the mappings were perturbed, for this parameter configuration.

Of interest is whether this trend is preserved for different parameter configurations. As noted in Section 6.3, the response to varying parameters is likely to be coupled. This was addressed by the construction of a response surface showing the effect of varying pairs of parameters simultaneously. Results are shown for locality with mutation rate and locality with selection pressure. The response surface in Figure 6.4 (left) illustrates the trend over mutation rates between 0.01 and 0.2. Both best and worst performance are evident when using the most local (Gray) encoding ($L_N^{-1} = 1$). Best performance in the candidate population occurred at the mutation rate $q = 0.13$. Performance decreased to a minimum at $q = 0.01$. Decreasing the encoding locality resulted in a trend towards higher performance in the limit of low mutation rates and poorer performance at higher mutation rates. Figure 6.4 (right) shows the response to changing locality for different selection pressures, μ, where μ is the number of individuals retained as parents from each population. The surface is displayed for a mutation probability $q = 0.1$. The

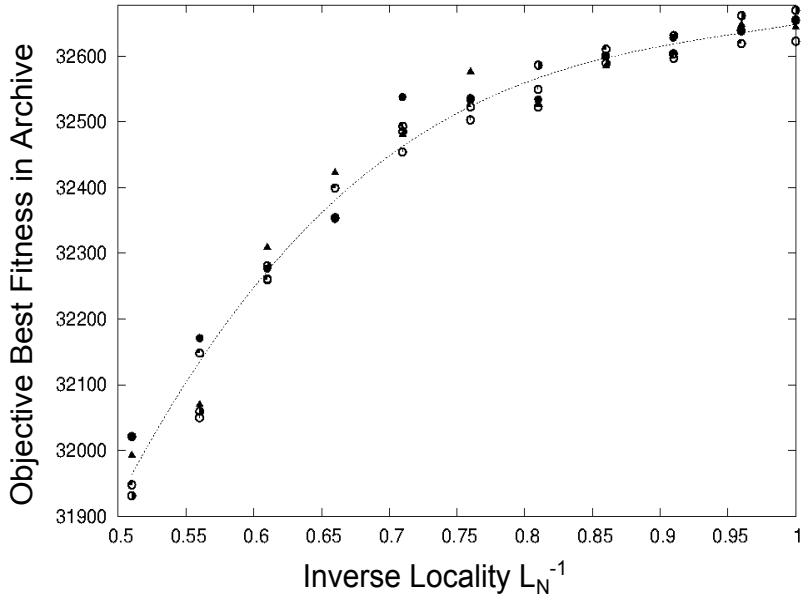

FIGURE 6.3: *Response in CO1 to varying mapping locality using initial parameter settings, repeated for 5 sets of sample encodings (symbols denote encodings from the same sample set). Performance measured at 50 generations.*

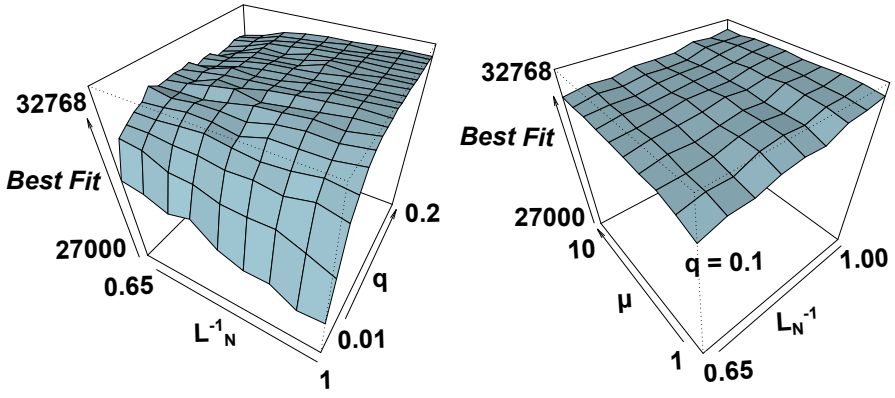

FIGURE 6.4: *Response surfaces for the CO1 game. Expected performance measured at 50 generations (500 runs/vertex). Left: Mutation rate. Right: Selection pressure.*

trend towards higher performance in the stronger locality encodings is retained over all settings of μ. The system was relatively insensitive to this parameter.

6.7.2 Greater-than game Results

The surfaces shown in Figure 6.5 give the results of a factorial experiment to investigate the effect of varying locality on performance and disengagement in GT. Sample results are shown at three mutation levels $q = \{0.01, 0.03125, 0.1\}$ whilst simultaneously varying the asymmetric bias ν in the coevolving test population. As in CO1, the best performance occurs when using the Gray encoding with a moderate mutation rate, $q = 0.1$.

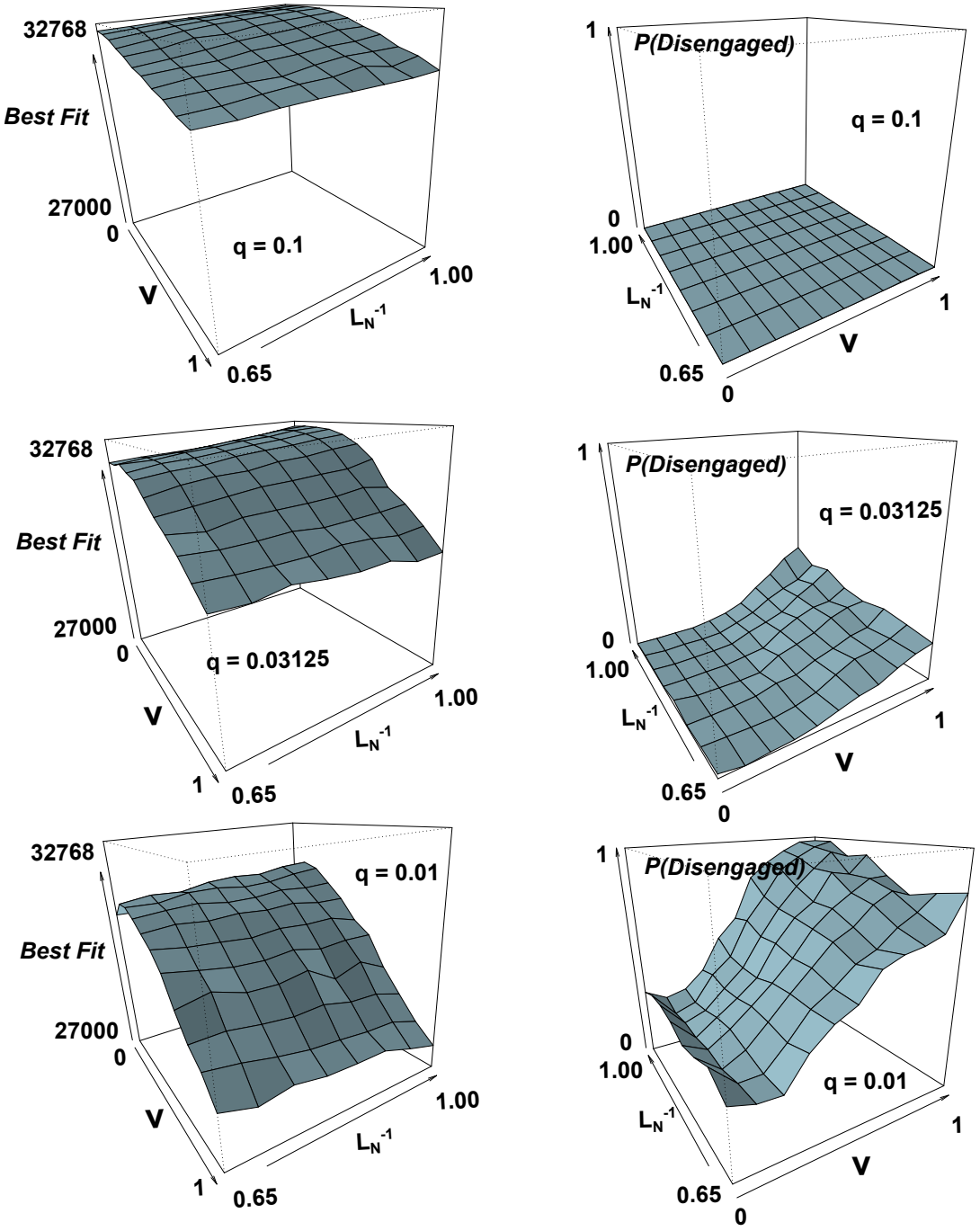

FIGURE 6.5: *Response surfaces for the GT game. Shown at three mutation rates* $q = \{0.1, 0.03125, 0.01\}$*, with respect to variation in bias setting and encoding locality. Measured at 50 generations (500 runs / vertex).*

Disengagement is not evident at this level. However, the probability of disengagement increases up to a 98% likelihood at the 0.01 mutation rate, when using Gray codes. At lower mutation rates, decreasing the encoding locality decreased the likelihood of disengagement. A commensurate performance increase is also seen for that case.

6.8 Discussion and Concluding Remarks

The factor that most strongly affects performance in these simple constructed systems is the probability of mutation. The response surfaces observed in CO1 suggest that the use of stronger locality mappings is only of benefit in this problem at moderate to high mutation rates. The trend reverses at lower mutation rates. A probable explanation for this is loss of phenotypic diversity. If that explanation is correct, the problem is an example of a situation where using a weaker locality mapping can increase performance when search diversity is low. In contrast, selection pressure in the CEA had only a marginal effect on performance. We note that a form of 'implicit' elitism is introduced to the algorithm through use of the IPCA, which only retains dominating solutions. This may account for the weak response to changes in selection pressure.

In GT, the test bias and disengagement pathology also significantly influenced performance. Increasing the asymmetric bias ν led to disengagement. For large bias, the objective fitness of the candidate population will still initially increase, but selection in the test population does not overcome the applied penalty. Loss of diversity in the tests then in turn halts the progress of candidates. At the q=0.1 mutation rate, disengagement falls, which we suggest occurs because a sufficiently diverse set of tests has been provided. The effects of changing mapping locality are then similar to those observed in CO1. Weak locality encodings outperform higher locality encodings at small ν and q, corresponding to a lower probability of disengagement.

The strong correlation between inverse (phenotypic) locality and performance measured in both CO1 and GT therefore supports our hypothesis, verifying that mapping locality is a factor in the performance of a coevolutionary algorithm on these problems. As observed in the non-coevolutionary GP systems in Chapter 3, the contribution of strong or weak locality encodings to performance is coupled with other algorithm properties. In particular, these results suggest that a complex interplay exists for this coevolutionary configuration between locality, mutation rate and disengagement. Locality was not the most significant contributing factor. However, it was observed to have an effect on performance in both problems throughout the parameter configurations. The effect of locality on the disengagement pathology is interesting, because it raises the possibility that certain mappings may be inherently more robust to pathological behaviours.

6.9 Chapter Summary

This chapter addressed the question "*Does genotype to phenotype locality contribute to coevolutionary algorithm performance?*". Experimental design considerations were discussed. Two constructed binary problems were then examined from the domain of test-based competitive coevolution. In both cases, mapping locality was shown to be strongly correlated with the performance of a canonical coevolutionary algorithm.

- The Compare-on-one (CO1) and Greater-than (GT) games were modified to incorporate a binary to integer mapping.
- A set of encodings were constructed by perturbation of Gray codes using a normalised measure of locality.
- A factorial analysis was carried out to examine the sensitivity of performance in CO1 and GT to changes in locality across the set of encodings.
- Encoding locality was observed to be a factor in performance under all parameter configurations.
- A relationship was observed between mapping locality and the probability of disengagement in GT.

The CO1 and GT binary number games are highly simplified coevolutionary systems, albeit ones that use different solution concepts. The next chapter will study whether these trends also hold in coevolutionary genetic programming and further investigate the interaction that has been observed here between mapping locality and coevolutionary pathologies.

Chapter 7

Locality in Coevolutionary Genetic Programming

Contents

7.1 Chapter Motivation

An issue of increasing interest is the construction of programs designed for competitive environments, where there exists a requirement to adapt to an adversary itself capable of intelligent action. Synthesising the concepts introduced in coevolutionary algorithms with genetic programming brings the technique into closer proximity with fields traditionally concerned with collective behaviour, such as multi-agent systems. However, coevolving the structures needed to support such applications, from simple state machines to neural networks, requires the use of increasingly complex representations. Therefore a better understanding is needed of how to design genotype to phenotype maps that are effective in such systems.

The aim of this chapter is to take a step in this direction by analysing the effect of semantic locality on the coevolution of programs. In the previous chapter we confirmed that GPM locality can be a factor in the performance of simple binary coevolutionary algorithms. A relationship was also suggested with a common pathological coevolutionary behaviour, disengagement. It is a logical progression to examine whether these findings generalise to coevolutionary algorithms when working with more complex interactive GP domains. A similar empirical method will be used to that constructed for binary representations in the previous chapter. The analysis will be supported using the methods developed in Chapters 3 and 6.

7.2 Chapter Outline

This chapter considers the question: *'Does semantic locality interact with the occurrence of pathological behaviours in program coevolution?'*. To address this, in Section 7.3 a set of coevolutionary GP problems is outlined. In each case, the interactive domain, measures of semantic difference and the relevant solution concepts are defined. Section 7.4 then uses these problems to continue the exploration of pathological behaviour and locality commenced in the previous chapter, presenting empirical results on the relationship between semantic locality, disengagement and cycling in coevolutionary GP. To complement this, Section 7.5 then analyses a more realistic historical GP problem, the *Game of Tag*. Methods of measuring progress and changes in performance are addressed for this case. The remainder of the chapter discusses these results and concludes.

7.3 A Problem Set for Coevolutionary GP

Few established benchmarks exist for coevolutionary forms of GP. Historically, one natural source of problems has been competitive pursuit and evasion games, where the objective is to develop strategies to intercept or escape an opponent. An example is the *Serengeti* problem, first analysed in GP by Haynes *et al.* (1995), which presents a classic predator-prey scenario. Strategies are developed for multiple predators ('lions') to capture a prey-agent (a 'gazelle') on a simulated grid-world. *Serengeti* is considered to be difficult to solve without a degree of cooperation between predators (Luke and Spector, 1996). Pursuit and evasion is frequently used in the coevolutionary literature, but has been criticised as a method of benchmarking (Cliff and Miller, 1995a), primarily due to the complexity of interactions between different strategies and the ensuing difficulties when defining measures of progress. However, given the breadth of practical applications, it remains an attractive area within which to analyse CEA provided a sufficiently simplified problem instance can be defined.

Another commonly used class of coevolutionary GP problem is a 'game versus environment', in which programs are coevolved to control an agent in conjunction with an increasingly challenging structure, such as a maze or series of obstacles. The *Tartarus* grid-world game proposed by Teller (1993) presents a situation in which an agent must manoeuvre a series of blocks onto positions around the edges of the world. Notably the *Tarturus* game requires that solutions must include a method of referencing previous states (memory). A more recent example can be found in Cartlidge and Ait-boudaoud (2011), in which a maze navigation problem was defined where strategies to control robots to escape a maze are coevolved simultaneously with increasingly difficult maps.

Although the occurrence of coevolutionary pathologies when employing GP has been reported (Lipson *et al.*, 2005), to our knowledge no explicit studies of the effect of properties of the genotype to phenotype map have been carried out. Of the coevolutionary pathologies described in Chapter 5, we postulate that disengagement and cycling have particular significance for GP systems. Asymmetric situations are a frequent occurrence in program coevolution. For example in the *Tartarus* problem evolving a solution such as a better finite state machine is combinatorially a harder task than generating a more challenging world. Cycling is also a particular concern because (in general) the evaluation of GP structures is more expensive than their binary counterparts. Revisiting a prior state has a larger associated cost.

Two drawbacks limit the utility of established problems such as *Serengeti* and *Tartarus* when investigating coevolutionary pathologies. As observed in the previous chapter, to measure progress or performance in a coevolutionary algorithm it should be possible to

provide an approximate metric of solution quality. *Serengeti* and *Tartarus* both lack a clearly defined notion of optimal behaviour or solution concept. In addition the mappings required to solve these problems are considerably more involved than those that have been analysed in previous chapters. The following sections therefore introduce two new minimal interactive domains, the *GP Greater Than Game* (GP-GT) and the *Simple Cycler* (SC), which are constructed specifically to explore disengagement and cycling in a GP setting. The problems are generalised from the concept of number games analysed within binary representations. To complement this a more realistic historical problem the *Game of Tag* (GoT) is described, which will then be used to explore the effect of locality on performance.

7.3.1 GP Greater-Than Game (GP-GT)

To investigate the pathology of disengagement in coevolutionary GP, a game was defined based on the the GA *Greater-Than* (GT) game described in the previous chapter. The GP-GT uses two symmetric populations of programs. Each program operates on real values formed from a constrained function set {+,-} accepting a single terminal input fixed at unity. Programs have a single output, derived by evaluating the expression at the root node of the program[1]. The outcome of comparing a pair of programs (p, p') is given as a function of the program outputs u. The interaction function is:

$$u(p,p') = \begin{cases} 1.0 & o(p) > o(p') \\ 0.5 & o(p) = o(p') \\ 0.0 & o(p) < o(p') \end{cases} \tag{7.1}$$

where $o(p)$ and $o(p')$ are the real valued outputs of each program. The expressions are constrained to a maximum depth n, measured from root to terminal nodes. The game is solved after a program is found from the subset of programs that maximises the output.[2]

7.3.2 Simple Cycler (SC)

Informally, we define cycling behaviour in coevolutionary GP as exiting and revisiting the same phenotypic state in a program search space. *Simple Cycler* is an elementary game that is designed to simulate measurable, irregular cycles in a coevolutionary GP algorithm. Evolved programs operate on boolean values and are constructed from the function set {AND,OR,NOT,IF}, where the IF function accepts three arguments: a

[1] In graph representations, such as CGP, we define the root as the last node in the integer genotype.

[2] GP-GT is superficially similar to the GP 'MAX' problem considered in Chapter 4. The key distinction is that programs are evaluated using only their relative rather objective fitness.

condition, response if the condition is true and response if the condition is false. The input is a fixed terminal with value TRUE. Programs are required to output a string of n boolean values (for example, from n selected output nodes). This output is mapped onto an unsigned integer $o \in \{1 : 2^n\}$, using a binary encoding. The interaction function is computed as:

$$u(p, p') = \begin{cases} 1.0 & o(p) > o(p') \text{ except } o(p) = 2^n \text{ and } o(p') = 1 \\ 1.0 & o(p) = 1,\ o(p') = 2^n \\ 0.5 & o(p) = o(p') \\ 0.0 & o(p) < o(p') \text{ except } o(p) = 1 \text{ and } o(p') = 2^n \\ 0.0 & o(p) = 2^n,\ o(p') = 1 \end{cases} \tag{7.2}$$

This expression states that the program corresponding to the greatest integer wins, except for the cases where the maximum integer (2^n) value is compared to the minimum integer, 1. Therefore under this function all programs can be positioned on a single intransitive cycle of length 2^n.

For example, for $n = 2$, programs may output values in the range {1:4}. The cycle implied is $o(p) = 1 < o(p) = 2 < o(p) = 3 < o(p) = 4 < o(p) = 1$. A program is said to have traversed the cycle when it has changed from a structure corresponding to the smallest integer to the largest and back, via the intermediate states. Cycling behaviour is monitored by measuring the average period of these transitions over a fixed number of generations.

7.3.3 The Game of Tag (GoT)

The *Game of Tag* is a conceptually straightforward two dimensional pursuit and evasion game introduced by Reynolds (1994). Games consist of an idealised scenario in which control programs are developed which provide pursuit and evasion strategies. Analogous to the children's game, the objective for each control program is to minimise the length of time during which a program is designated as 'it' (the pursuer). Play occurs between pairs of competitors, which are point objects able to move at a fixed speed over a number of discrete timesteps. No account is made for momentum or limitations in change of heading. If the competitor designated as 'it' enters a certain capture radius of its opponent, the opponent is 'tagged' and the roles are exchanged. Consequently, successful programs must simultaneously evolve pursuit and evasion behaviours, depending on their role at that timestep. Effective algorithms should increase the capability of competitors in both roles progressively over time. An example screenshot showing a simulation a game in progress is given in Figure 7.1. At the start of a game, one competitor is placed at the centre of the two dimensional play area. The other competitor is placed

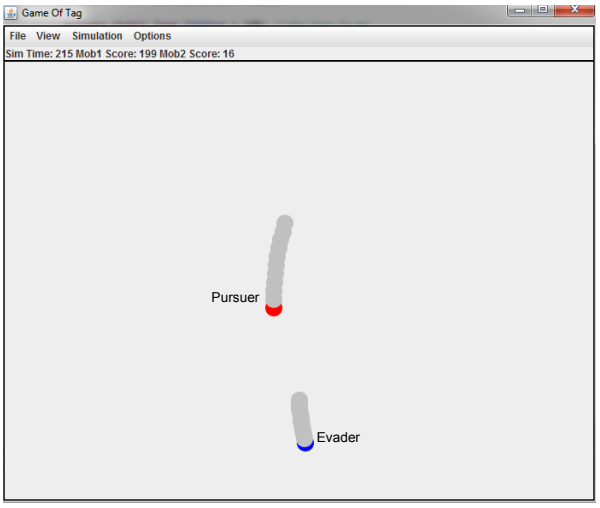

FIGURE 7.1: *A simulation of the game of tag between two adversaries.*

uniformly at random in a square region with width w centred on this position. Whilst a competitor is in pursuit, it is set to move at twice the speed of the evader. Inputs to each program are restricted to a vector in the local coordinate system of the competitor and a boolean value, which specifies whether the competitor is in pursuit or not at that timestep. Programs provide a single real number output, which is interpreted as an updated heading. A score z is awarded to each competitor at the end of the game, equal to the number of timesteps spent as the evader. During training, the subjective fitness of each program is evaluated as the average score obtained over Z games against a subset of coevolving adversaries. The interaction function between programs is given using the mean score obtained over all games:

$$u(p, p') = \frac{1}{Z} \sum_{i \in Z} z_i(p, p') \tag{7.3}$$

To ensure symmetry, for each set of games in the first half of the set the first competitor begins as the evader and in the second half the roles are swapped.

7.3.4 Game Solutions and Conditions for Optimality

Table 7.1 summarises the solution and its corresponding class for each problem, following the framework for coevolutionary algorithms described in Chapter 5. Details are discussed in the following sections.

TABLE 7.1: *Selected coevolutionary GP problems and their solutions.*

Problem	**Goal State**	**Solution Concept**
GP-GT	Program p with $o(p) = 2^n$.	Maximisation of All Outcomes.
SC	No fixed goal state.	N/A, intransitive cycle.
GoT	Maximised performance vs. PP and PN.	Maximisation vs. Test Set.

Solutions in GP-GT and SC

In GP-GT, the game is considered to be solved when a program is found from the set which maximises the possible output, $o(p) = 2^n$. In the framework of Chapter 5, this corresponds to the *Simultaneous Maximisation of all Outcomes* solution concept. By contrast, the SC game does not have a globally optimal set of solutions. All programs in the Simple Cycler lie on the same intransitive cycle and are therefore of equal merit objectively.

Solutions in the Game of Tag

In the original description of the Game of Tag, it was noted that in the absence of momentum a simple but effective strategy is to move directly towards the opponent whilst the pursuer and directly away whilst the evader. This is a method referred to as 'pure pursuit' (PP), in the language of pursuit and evasion games. The solution was claimed informally to be optimal for the ideal situation described. A little consideration shows that this is only partially correct. Given two competitors executing this strategy, deviation by either to any other strategy (including mixed combinations) results in poorer performance, because a route other than the shortest path must be traversed. This solution can therefore be classified as a *Nash equilibrium*, under the definition given in Chapter 5. However, it does not satisfy the *Simultaneous Maximisation of all Outcomes* solution concept, because there exist cases where pure pursuit is less effective than more sophisticated guidance laws.

As an example, to measure progress in this work an alternative strategy *proportional navigation* (PN) will also be considered. Proportional navigation is a widely applied guidance law used extensively in missile systems, backed by a large body of analysis (Yuan, 1948; Shukla and Mahapatra, 1990). The strategy is based on the principle that an interception between the trajectories of two objects travelling with fixed speed will occur if the bearing between them is a non-zero constant (an example is sketched in Figure 7.2). Intuitively, proportional navigation is more effective than pure pursuit in situations where there is a requirement to 'lead' the target. In 'pure' proportional navigation, the change in heading of a pursuer $\dot{\theta}$ is proportional to the change in line of

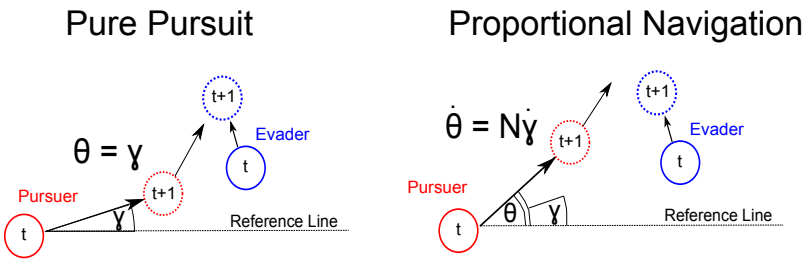

FIGURE 7.2: *Pure and proportional navigation pursuit strategies. The pursuer is shown against a fixed trajectory evader over a discrete timestep* $t \rightarrow t+1$.

sight to target $\dot{\gamma}$, where N is a unitless quantity referred to as the proportional navigation constant, which controls the magnitude of response (Equation 7.4). Real world applications typically assume a value of N in the range 3 - 5. A value of $N = 3$ is assumed in the present work, following from the analysis of Shukla and Mahapatra (1990).

$$\dot{\theta} = N\dot{\gamma} \tag{7.4}$$

In addition to PP and PN, we will also evaluate solutions against intermediate quality mixed opponents. Measuring progress exclusively against an optimal or near-optimal adversary will not usefully indicate progress over the whole evolutionary period, because the quality of mediocre players may not be distinguished. Therefore, mixed strategies will be generated by introducing variation into the pure pursuit strategy. The mixed strategies respond with a randomly selected angle with probability $P(noise)$ and the correct pure-pursuit response with probability $1 - P(noise)$ such that $P(noise) = 1$ corresponds to a strategy with random outputs. Testing strategies in this fashion provides an indication of the robustness of an evolved strategy against a broader range of phenotypes, whilst also giving a scalable measure of quality.

7.3.5 Metrics for Semantic Difference

Given this problem set, we specify a metric for the semantic difference between programs, defined for each interactive domain. The metrics are denoted as d_{GP-GT}, d_{SC} and d_{GoT} respectively:

$$d_{GP-GT} = |o(p) - o(p')| \tag{7.5}$$

$$d_{SC} = min \begin{cases} |o(p) - o(p')| \\ 2^n - |o(p) - o(p')|. \end{cases} \tag{7.6}$$

$$d_{GoT} = \sum_{K} |\theta(p) - \theta(p')| \tag{7.7}$$

For GT, Equation 7.5 uses the absolute difference in the output of each program. Because the winner of an interaction in GT is determined solely by program output, this is the natural measure for that problem. Similarly for SC, Equation 7.6 uses the smallest difference in output between programs, but takes into account intransitivity by wrapping around cycle. For example, in the $n = 2$ case, programs with output 1 and 4 are considered to be adjacent ($d_{SC} = 1$), because they are positioned next to each other on the cycle.

The Game of Tag measure, Equation 7.7, is the absolute difference between output headings, summed over a set of K x-y input vectors to both programs. The set of vectors point to a grid of uniformly distributed fixed positions across the square playing region in the Game of Tag. This approach is derived from the definition of semantic sampling distance used in Nguyen (2011). An example of the procedure is given in Figure 7.3.

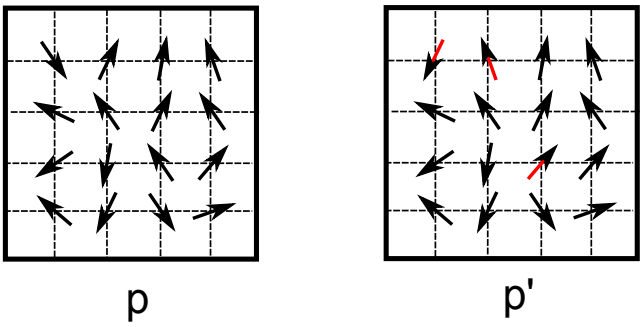

FIGURE 7.3: *Illustrates the calculation of semantic differences in the Game of Tag. The program p' differs from p at three sampled points. The metric d_{GoT} sums the absolute differences over each vector in the sampled region.*

During initial experiments, several alternative semantic metrics were considered for the Game of Tag (for example the Euclidean distance between outputs, the number of matching output angles). These were tested by measuring distances between the pure pursuit strategy and noisy variations. Equation 7.7 was accepted as the simplest approach that ensures the differences in the metric are linearly proportional to changes to the output. Each difference between samples contributes with equal weight to the total difference. Potential limitations of this assumption are discussed in Section 7.6.

7.4 Mapping Locality and Pathological Behaviours

7.4.1 Algorithm Configuration

Reproduction and semantic bias were carried out by using the `SIGMUTATE` operator and varying the parameter α, as described in Chapter 3. Recall that `SIGMUTATE` biases the

TABLE 7.2: *A summary of fixed algorithm parameters and game properties.*

Parameter	**GP-GT**	**SC**	**GoT**
Nodes	20	20	50
Function Set	{+,-}	{AND, OR, NOT, IF}	Reynolds (1994)
Terminal Set	{0,1}	{TRUE}	$\{x, y, isIt, [0.2{:}1.0]\}$
Mutation Rate	0.05	0.05	0.02
Selection	4+6 ES	1+1 ES	1+4 ES
Populations	2×10	2×1	2×5
Runs	500	100	500
Generations	500	1000	500
Offset β	1	1	0.05
Games/Opponent	1	1	Training: 5, Testing: 100
Game length	N/A	N/A	100 timesteps
Startbox Size w	N/A	N/A	7
Pursuer speed	N/A	N/A	2
Evader speed	N/A	N/A	1
Capture radius	N/A	N/A	1

locality of the mutation operator using a sigmoid function. A summary of the fixed algorithm parameters used in each problem case is given in Table 7.2. In preliminary experiments, each fixed parameter was tuned independently. A locally optimal set of parameters was found for the CGP system on each problem, under no semantic bias ($\alpha = 0$). The range of values given in Miller (2011) was used as a basis. The Game of Tag parameters are based on those originally fixed by Reynolds (1994).

Although a full-factorial analysis of all parameters is outside the feasible scope of this work, in Section 7.4.3 the sensitivity of our experimental outcomes in SC are tested with respect to mutation rate and length of CGP genotype. The offset β was fixed to the minimum semantic difference in the constructed problems and a representative small angular difference of 0.05 revolutions (18°) in the GoT.

7.4.2 Disengagement in the GP Greater-Than Game

Figure 7.4 shows the mean value of the largest program output in each population over the search period and the probability of disengagement at that generation, obtained using 500 runs. Examples of the dependency on α are also shown in detail reported at 250 generations. Performance and disengagement were strongly correlated at this point with the value of α (Spearman correlation, +0.99 and -0.95 respectively, using a p-value of 0.005 under an exact test). A similar response is evident throughout the search. Poorer performance and the highest probability of disengagement are measurable in the stronger locality mappings ($\alpha \to$ -1). Conversely, mappings with an unbiased or weak locality corresponded to a lower probability of disengagement. The results are therefore

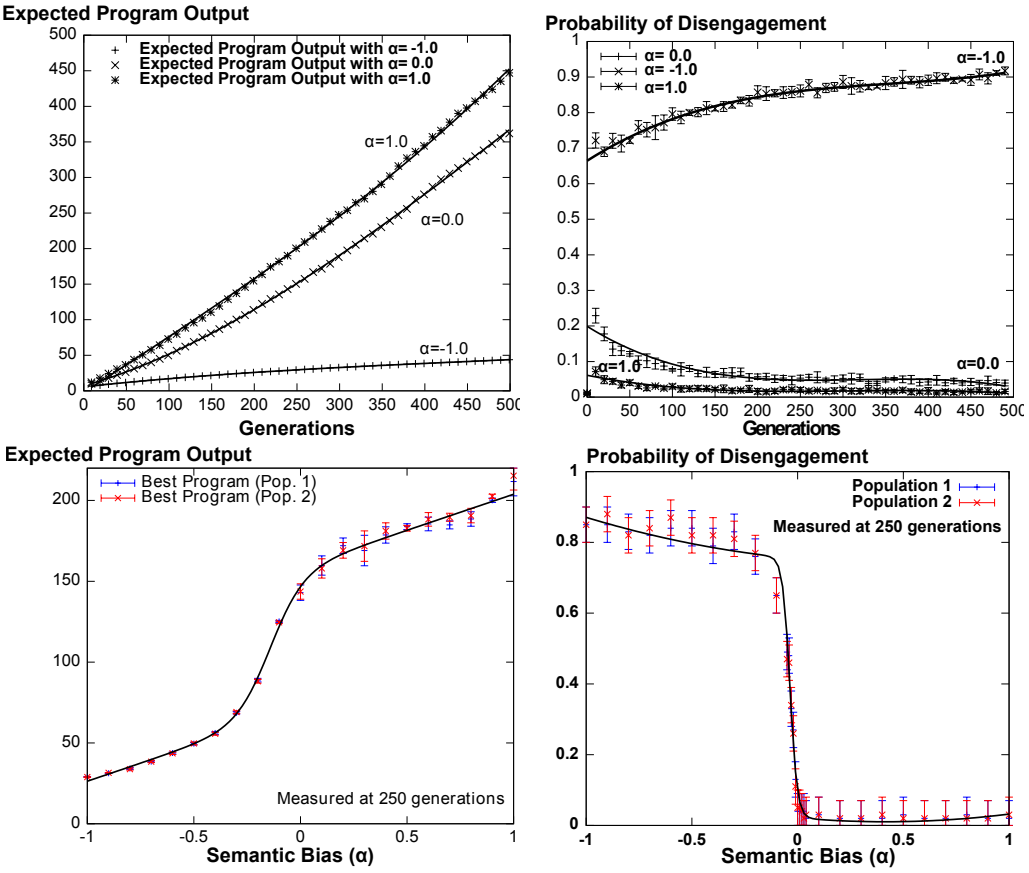

FIGURE 7.4: *The effect of locality on disengagement in GP-GT, using Cartesian GP. Top left: Expected output of best program. Top right: Probability of disengagement. Bottom left: Sensitivity of output to semantic locality. Bottom right: Sensitivity of probability of disengagement to semantic locality (reported at 250 generations).*

in agreement with the findings using the binary representation for GGT in Chapter 6. This supports the theory that higher locality mappings in which large changes in phenotype are less probable are more prone to disengagement.

7.4.3 Periodicity in the Simple Cycler

Given the relatively small number of phenotype states in Simple Cycler, it is convenient to examine the effect of changing the parameters of the operator directly by sampling. Figure 7.5 shows the change in phenotype distribution produced by varying α, for $\beta = 4$. Recall that β gives the offset of the sigmoid function in Equation 3.6, hence for the d_{CYCLE} metric a value equal to half the maximum, 2^{n-1}, is appropriate. Each distribution is obtained by application of 1000 mutations to a randomly initialised population of 100 genotypes. Characteristically for the CGP mapping, a high proportion ($\approx 42\%$) of the mutations are neutral. As α is increased, the distribution is skewed by the bias, reducing the number of neutral mutations. Mutations at larger phenotype distances

become more frequent. Conversely, negative values of α skew the distribution towards neutral mutations and a greater frequency of small phenotype changes.

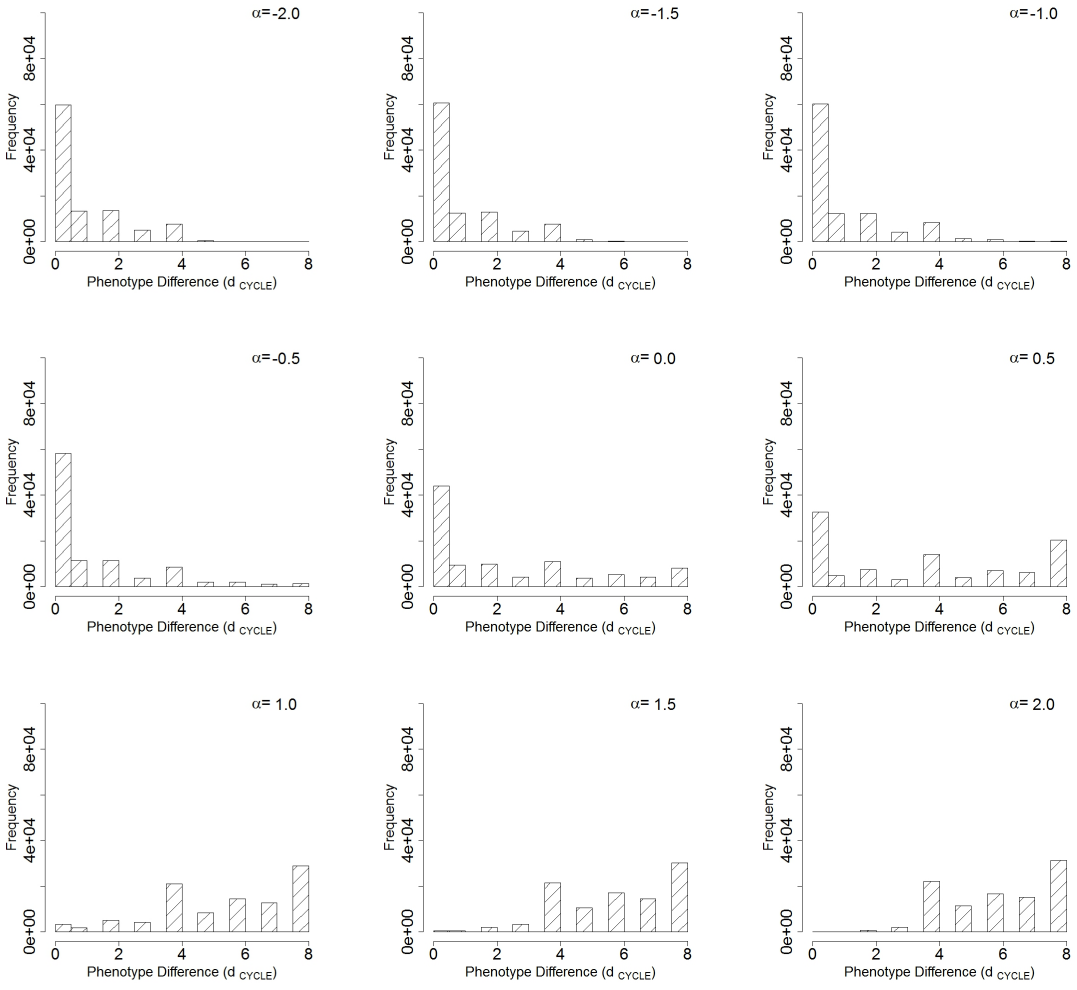

FIGURE 7.5: *The effect of SIGMUTATE on the distribution of phenotypes generated on mutation in the Simple Cycler problem. Results are shown using a uniform mutation operator in CGP with mutation probability* $q = 0.1$*, for* $\alpha \in \{-2 : 2\}$ *and* $\beta = 0.5$.

The average cycle period in SC can be characterised by measuring the mean number of generations for each population to cycle between the maximum state 2^n and back. Equivalently, this is the average number of mutations before the state is revisited. Figure 7.6 (top left) shows the period obtained for transitive chains of increasing length. Accordingly, as n is increased, the time to traverse the chain is larger. Figure 7.6 (top right) shows an extract from an example run using an unbiased mutation operator, for an intermediate value $n = 4$ (with mutation probability $r = 0.1$). Each genotype cycles irregularly through the 16 states. The unbiased distribution ($\alpha = 0$) shows a distribution of phenotypes whose frequency is not strongly correlated with distance.

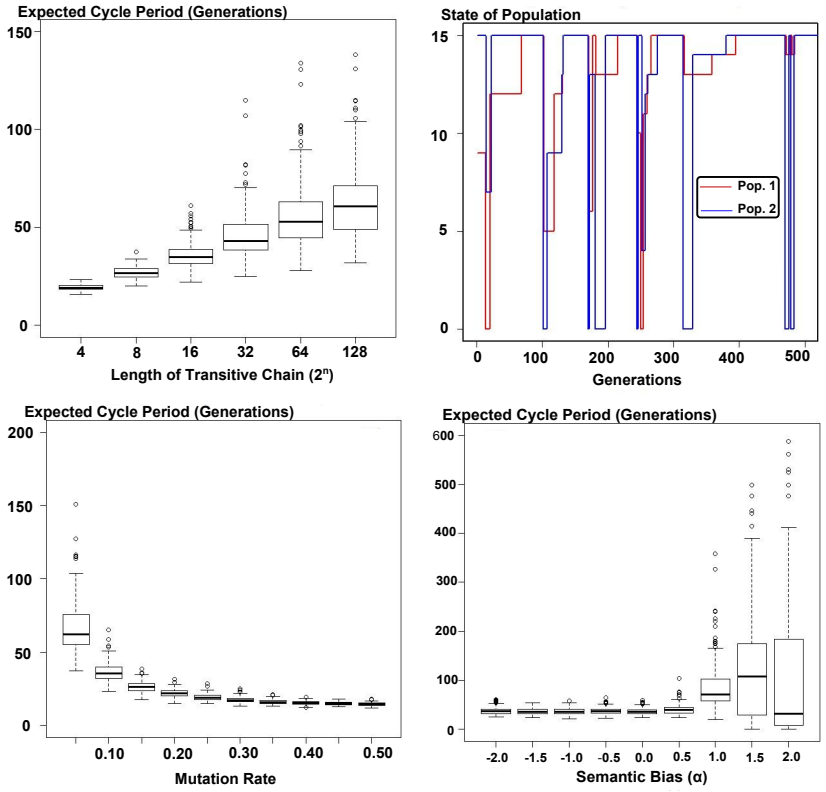

FIGURE 7.6: *Cycling in the Simple Cycler case. Top left: Scaling of unbiased cycle period with problem size. Top right: Extract of a coevolutionary run, illustrating irregular cycling in both populations. Bottom left: Response to mutation rate. Bottom right: Response to semantic bias.*

Figure 7.6 also shows the effect on cycle rate of changing the mapping, with respect to mutation rate (bottom *left*) and α (bottom *right*). At high mutation rates states are revisited with greater frequency $\sim \frac{1}{2^n}$. At low rates, the frequency tends to zero. Weak locality ($\alpha \geq 1$) corresponds to longer periods, though no change is evident between the stronger locality maps. The response to mutation rate can be explained by noting that the algorithm tends towards random search at high values. At low rates, cycling slows because neutral mutations are more prevalent and traversing the whole chain takes a longer period of time. Fast cycling is also apparent at $\alpha < 0$ because iterating through the intermediate states can occur more quickly in a localised mapping. Finally, at high values of α transitions to neighbouring states are more difficult, which decreases cycling in the system. The distribution of cycle periods also broadens in weak locality maps because phenotypes are fixed for longer periods, which gives rise to a larger number of outlying values.

7.5 Mapping Locality and Performance

7.5.1 CGP Implementation

Following the original work, the root node of all programs evolved in the Game of Tag is seeded with the 'IF-IT' function to provide a separate flow of execution for pursuit and evasion. In the CGP implementation, this was ensured by fixing the final integer of the genotype to the value that denotes this function. With that exception, during initialisation all other integer values were selected with equal probability. Because there is no standardised approach to providing constants in CGP, the simplest technique is adopted here: the introduction of a small array of fixed constants as terminal values {0.2, 0.4, 0.6, 0.8, 1.0}.[3] For convenience, Table 7.3 gives the original function set. The largest function arity is 4, therefore each node in the CGP representation corresponds to five integer values, such that the total genotype contains 250 integers.

TABLE 7.3: *The Game of Tag function set (Reynolds, 1994)*

Function		**Function**	
+	(a+b)	*if-it*	if-it then a else b
-	(a-b)	*min*	if a $\leq$ b then a else b
*	(a*b)	*max*	if a $\geq$ b then a else b
%	protected division	*if-lte*	if a $\leq$ b then c else d
abs	absolute value of a		

7.5.2 Progress Metrics

Performance in the Game of Tag was measured by relative evaluation against four strategies: pure pursuit (PP), proportional navigation (PN), a strategy which returned a random heading (R) and a mixed strategy (PR). The mixed strategy substituted a random response for pure pursuit 50% of the time. In addition vector field plots, as described by Reynolds (1994), were used to gain a qualitative impression of progress. The plots visualise the angular response of a program to different vector inputs. Examples are given in Figure 7.7 for the pure-pursuit and proportional navigation strategies. Each arrow in the plot shows the output angle of the program, with respect to a vector from the origin to that point. Therefore, in the pure-pursuit case, pursuit corresponds to all arrows pointing outwards from the origin and evasion to all arrows pointing inwards. This is expressible using the quadrant arctan function. In the proportional navigation

[3] More general methods of evolving constants, such as ephemeral random values (Koza, 1992), were considered in preliminary attempts to apply CGP to this problem. However, the inclusion of constants generated at runtime greatly complicates this analysis, by implying a variable, rather than fixed, genotype to phenotype map. Additionally, it introduces asymmetry into the distances between genotypes, which violates the assumptions made in Chapter 3.

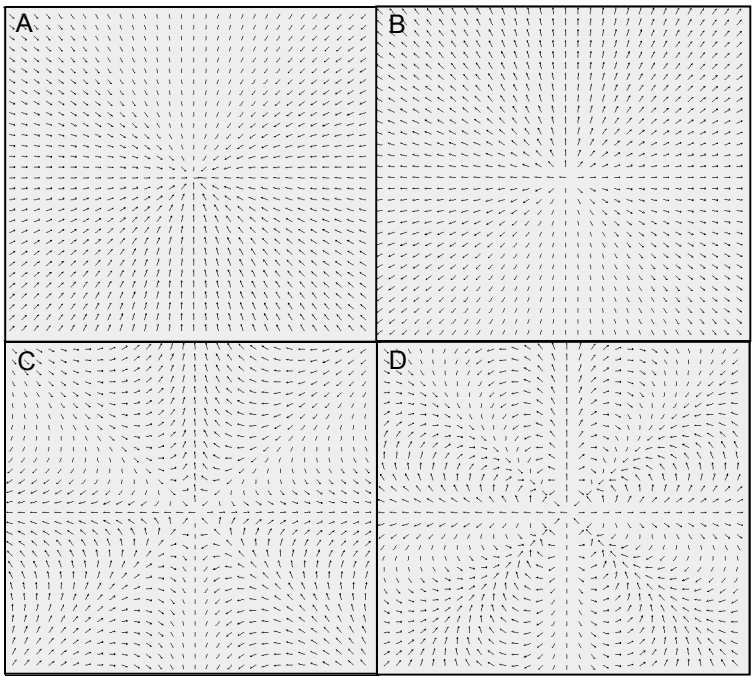

FIGURE 7.7: *Vector-field plots for ideal pursuit and evasion strategies. A: Pure Pursuit (evader). B: Pure Pursuit (pursuer). C: Proportional Navigation (N=3, pursuer). D: Proportional Navigation (N=3, evader).*

case, vectors show the direction adopted to intercept the expected path of the target, based on the current rate of change in line of sight.

7.5.3 Performance in the Game of Tag

Figure 7.8 (top left) shows the expected progress of unbiased CGP against these metrics.[4] Best performance was achieved against the randomised and proportional navigation strategies (evolved strategies are expected to win $\approx 90\%$ of games versus random opponents). Weakest performance is evident against the PP and PR strategies. A modest expected performance ($\approx 20\%$) was observed against the pure pursuit case, though higher success rates were achieved in individual cases. A direct comparison with the performance of Reynold's implementation is not possible because only 5 individual runs were reported in the original work. Figure 7.8 (bottom left, bottom right) shows an example set of co-evolved vector field plots over the course of a run. The responses exhibit similar characteristics to those described in the previous work and resemble the pure-pursuit strategy, approximating the quadrant arctan function.

[4] An additional cost is associated with computing the difference between phenotypes in `SIGMUTATE`. This is omitted in the results presented, because in practice it was found to be negligible relative to the large number of evaluations required to assign a fitness value to each individual in this system (200 evaluations per game, 5 games per fitness computation).

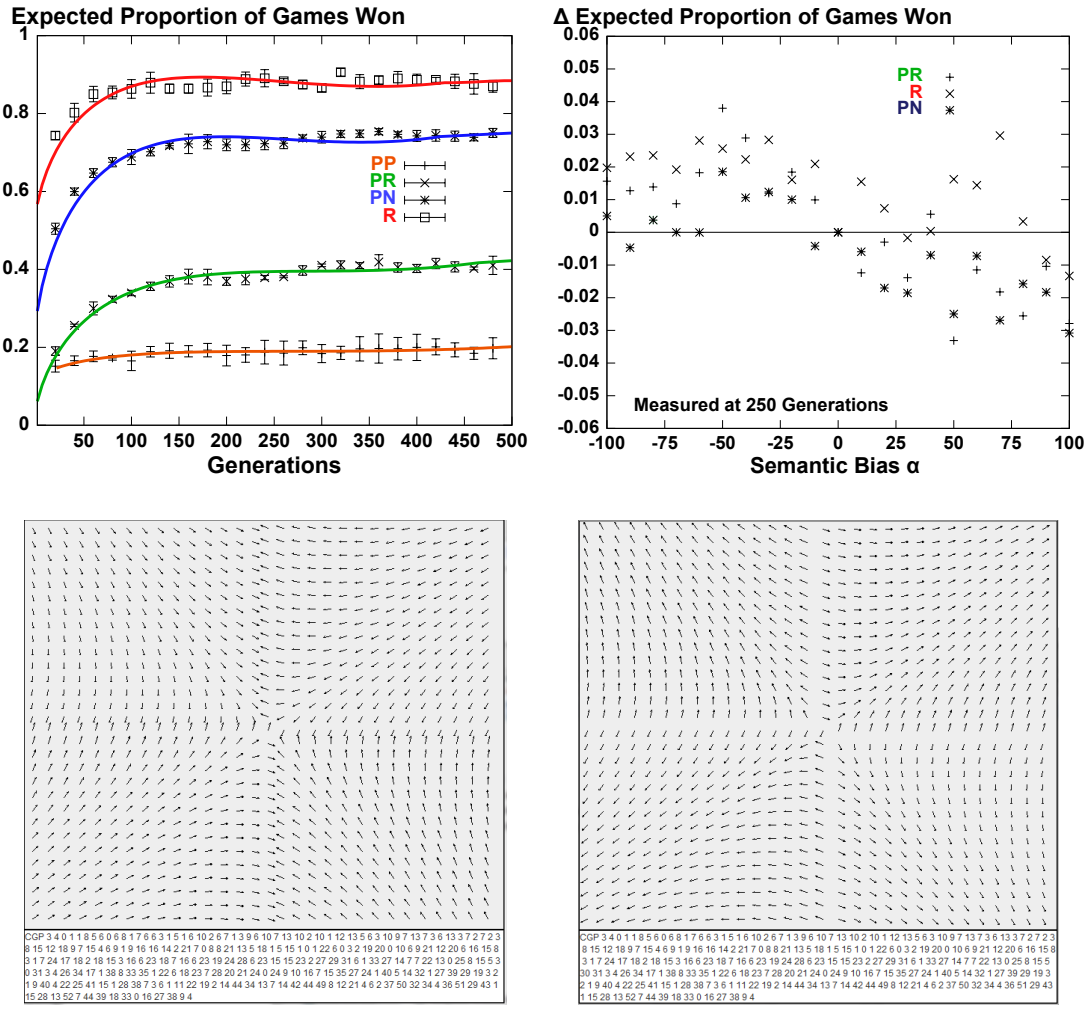

FIGURE 7.8: *Game of Tag Performance. Top Left: Unbiased CGP performance. Top Right: Sensitivity of performance to semantic bias. Bottom Left: Example coevolved evasion strategy. Bottom Right: Example coevolved pursuit strategy.*

The sensitivity of this result to semantic bias was examined. No significant change was observed when measuring against the PP strategy. A weak response was measurable against the PR, R and PN strategies. The measured Spearman correlation coefficients in each case are (PP = -0.17, PR = -0.82, PN = -0.79, R = -0.62) where the correlations in PR, PN and R are significant at $p \leq 0.005$ (exact). Figure 7.8 (top right) shows the change in expected performance at 250 generations for each of the significant results. Best performance was observed at $\alpha = -50$, which corresponds to a strong bias towards small phenotypic changes. However, the net performance change is small ($\approx 5 - 10\%$ difference in win ratio) and the overall effect of varying semantic locality in this case is marginal.

7.6 Discussion and Concluding Remarks

Evidence from the constructed problems we have examined confirms that semantic locality can be a factor in CEA performance, but as seen in the GA examples studied in the previous chapter, this is complicated by interaction with coevolutionary pathologies. The experiments have highlighted that performance is inter-dependent on each of these issues and provided an initial quantitative analysis for disengagement and cycling. The strong correlation between locality and disengagement in GP-GT supports the hypothesis that disengagement can be related to mapping locality in program coevolution. The probability of disengagement increases for small changes because programs are less likely to be distinguished using the coevolving population. Although we must be wary of generalising too far from this simplified example, the result suggests there may be a tradeoff between the use of semantically local mappings and operators which permit sufficient diversity to ensure populations of programs remain engaged.

An interesting result from the Simple Cycler example is the trend towards faster cycling in both random (high mutation rate) and highly local search (small α). Inspection of individual runs showed that this is a consequence of two distinct search behaviours. Under random search, programs do not traverse the chain of states and instead revisit each state with fixed probability. For small n this probability is high, therefore giving a short period. However when mutations are constrained to programs with a small difference in output the search tends towards hillclimbing through the intermediate states, also giving fast cycles. The result implies that because cycling in GP is a pathology that occurs when an algorithm follows a transitive chain of programs, mappings biased towards small semantic changes may worsen this behaviour.

Varying semantic bias in the Game of Tag problem introduced only a minor change to performance. Two issues may contribute to this. Firstly, relative to the constructed problems, the semantic mapping in the Game of Tag is significantly more complex. The relationship between the outcome of a game and changes to program output is not transparent. It is therefore not clear whether our metric for semantic difference is the most suitable for this case. For example, given that agents begin positioned at the centre of the space and spend a greater proportion of time in this area, it may be that different program responses in that region are more likely to skew the outcome of a game. Other, non-uniform weightings across the set of K samples may be more appropriate. Secondly, the random component in the starting configuration introduces noise into each outcome. Noise reduces the likelihood of disengagement under the definition given by Cartlidge and Ait-boudaoud (2011), because of the increased variation in fitness values.

At present, archiving techniques are the principal approach to providing monotonic progress in CEA. Examining methods in GP that have achieved good results without archives (Jaskowski and Krawiec, 2009) may provide an indication of how representations inherently more robust to coevolutionary pathologies can be developed. Although this result is sufficient to address our research question, future analysis could extend the scope of these observations to other GP representations and problem sets, such as GE. This analysis also motivates the study of other pathologies which have been characterised in binary representations but have been neglected in GP, such as overgeneralisation (Wiegand, 2003). In the next chapter, we will examine some of these issues and consider other areas of future work.

7.7 Chapter Summary

The goal of this set of experiments was to examine the role of the semantic mapping in coevolutionary GP. The chapter presented empirical work analysing the effects of locality on pathological behaviours and performance.

- Two constructed problems were introduced, the *GP Greater-Than-Game* (GP-GT) and the *Simple Cycler* (SC), to elicit quantifiable disengagement and cycling in a coevolutionary GP system.
- Performance in the GP-GT was found to be strong correlated with mapping locality. Use of an operator biased towards small semantic changes led to a higher probability of disengagement.
- Cycling was observed in the SC model and an analysis of the frequency response to changes in mapping locality was presented.
- Additional results were provided for coevolutionary CGP on the Game of Tag, a historical pursuit and evasion game. A weak response to mapping locality was observed under a locally optimal parameter set.

Chapter 8

Conclusions

Contents

8.1 Thesis Summary

This thesis set out a series of research questions to investigate the role of genotype to phenotype locality in evolution-inspired search heuristics. The work addresses the optimisation of programs by evolutionary and coevolutionary search, focusing on the techniques and paradigms of genetic programming. The first part laid the foundations for this analysis, outlining the principal classes of genetic programming representation

and defining the characteristics of the genotype to phenotype maps used in GP. The second part focused on establishing statistical and visualisation tools to analyse locality using this framework, for evolutionary systems. The third part then moved to empirically examine locality in the context of coevolutionary systems.

The purpose of this chapter is to analyse the findings of this work in relation to my original research questions and hypotheses. I will summarise the main thesis contributions, then provide suggestions for future research. The final sections relate these outcomes back to wider open research issues in genetic programming and program evolution.

8.1.1 Hypotheses and Research Questions Revisited

Measuring Locality in Genetic Programming

Q1: **"How can a statistical measure of genotype to phenotype locality be developed for different genetic programming representations?"**

In Chapter 3, it was demonstrated that the Mantel statistic from numerical ecology can be applied to determine the correlation between artificial genotype and phenotype distances. The statistic can be used to guide understanding of how changes to evolutionary search operators or representation affect correlations in the mapping. Because the technique is applied over different distance classes, this is extendible beyond the local neighbourhood to consider the strength of the mapping between more distant genotypes. The novel application of the technique to this context was validated in three case studies using different classes of representation. The experiments showed that the Mantel statistic can distinguish between strong and weak locality mappings. For the example problems and genetic operators selected, weak locality was shown at the syntactic level using the Mantel statistic in two leading GP methods, Cartesian Genetic Programming and Grammatical Evolution. Given a suitable choice of syntactic or semantic metrics, the method can be applied flexibly over the GP paradigms.

Visualising Locality

Q2: **"How can locality be visualised over regions of the GPM?"**

Mapping locality is not uniform throughout the genotype space. Chapter 4 provided a method of visualising locality across sampled regions of the mapping, by the application of a force-directed graph layout algorithm. Examples were provided for fully-enumerated GP search spaces under Cartesian Genetic Programming and Grammatical Evolution. It was possible to observe regions of strong and weak locality using the projected graphs in

both representations. The approach was found to be scalable to greater than 10^5 vertices on a current desktop machine. In addition, the projected graph drawings captured previously unseen features such as symmetry in the CGP genotype to phenotype map. Graph drawings were also illustrated for samples from the landscapes of more complex GP problems, such as search in digital circuits. These examples indicate that force-directed layout algorithms are a viable approach to displaying mapping locality for the complex multi-modal surfaces found in GP.

Locality and Coevolutionary Performance

Q3: **"Does genotype to phenotype locality contribute to coevolutionary algorithm performance?"**

Chapter 6 defined an empirical framework to examine the extent to which changes in mapping properties, such as semantic locality, affected progress in binary coevolutionary systems. In a factorial experiment using competitive, test-based coevolution, locality was shown to have small but significant effect on performance relative to other factors. In the number game benchmark problems considered, Compare-on-one and Greater-Than, the sign of the contribution of strong locality to performance was found to be contingent on mutation rate. Systems with a low mutation rate benefited from weaker locality mappings. The results confirm that locality measured at the semantic level can contribute to the performance of CEA on co-optimisation problems, though the magnitude of the effect varies dependent on where in the parameter space the algorithm is situated.

The Effect of Locality on Pathological Coevolutionary Behaviours

Q4: **"Does semantic locality interact with the occurrence of pathological behaviours in program coevolution?"**

Also in Chapter 6, it was then further shown in the Greater-Than experiment that the likelihood of a pathological coevolutionary behaviour, disengagement, was dependent on semantic locality. Strongly semantically local mappings were more prone to disengagement throughout the evolutionary runs. In the less local mappings, disengagement occurred at a reduced rate. To explore whether this effect was also evident in more complex systems, in Chapter 7 the response of disengagement to locality was studied using program representations. A strong coupling between disengagement and mapping locality was observed in a GP generalisation of the GT problem.

Chapter 7 then considered the effect of mapping locality on transitive cycling in coevolutionary genetic programming. A constructed problem was presented, Simple Cycler, that enabled the frequency of cycles to be monitored in a controlled fashion. It is evident that varying the mapping locality alters the cycling frequency for this problem. It follows that semantic locality can affect the form of transitive cycling in coevolutionary systems.

The contribution of locality to the performance of program coevolution was also addressed in Chapter 7 using a more complex pursuit and evasion problem, the Game of Tag. A statistically significant response to mapping locality was observed, but the overall magnitude of the change in performance was small.

Overall, the results from this sequence of experiments support the view that mapping locality - whilst contributing to performance in coevolutionary algorithms - is not necessarily the dominant factor. However, mapping locality does have bearing on the occurrence of coevolutionary pathologies such as disengagement and cycling, which can under certain circumstances influence whether a coevolutionary algorithm succeeds or fails.

8.1.2 Main Contributions

In summary, the main original contributions presented during the course of this research included:

1. A novel application of the Mantel statistic to measure locality in artificial genotype to phenotype maps.
2. The application of a force-directed graph layout algorithm, Force Atlas 2, to visualise GPM locality in GP search spaces.
3. An empirical framework to investigate performance factors in coevolutionary algorithms. The application of this framework to coevolutionary genetic programming.
4. The observation of a relationship between genotype to phenotype locality and the coevolutionary pathological behaviours disengagement and cycling.

8.2 Extensions and Future Work

8.2.1 Relating Locality to other GPM Properties (Chapter 2)

Locality and GP using Modules

Many modern GP systems incorporate some form of explicit modularity. However, modularity has not been addressed in this thesis. A useful goal of future work would be to clarify how explicit modularity can be included in the geometric view of locality outlined in Chapter 2. As an example, in Walker and Miller (2008), sections of a CGP genotype were encapsulated into modules available in a list for reuse. The action of forming a module using this type of mechanism, such that it is made available for reuse by some subsequent operation, changes the topology of the genotype to phenotype map.

To see this, consider the effect of forming a module on the structure of the genotype space. Forming a module increases the likelihood of transitioning to a genotype that includes the common functional elements in that module. Under a probabilistic measure of genotype distance, this decreases the distance to that genotype in the genotype space, from any genotype that can accept the module. The distances between phenotypes are unchanged. Therefore, if the genotypes that can accept the module have similar phenotypes (under some metric) then it can be seen that the overall mapping locality has been increased by this action. Conversely, if the phenotypes are dissimilar, then the overall mapping locality must have decreased.

The concept naturally leads to the notion of imposing some form of phenotypic similarity on module development. One method of accomplishing this might be to measure the semantic distance to all stored modules. Operators could be biased to select modules that had a small semantic effect on the code. In principle, this may provide a method of maintaining strong semantic locality when the reuse of modules is required.

Locality and GPM Neutrality

This thesis has drawn a distinction between mapping locality and mappings which permit neutral changes in phenotype. One view of locality is as a measure of the relationship between changes in structure at different levels in the mapping process. From that perspective, an event that produces no change is not indicative of strong or weak locality. However, the issue is complicated by the different forms of neutrality that occur in indirect maps. Phenotypically neutral changes can be derived through different mechanisms across the GP paradigms, whether by changes to a redundant part of the genotype or by a change which leads to a semantically or fitness equivalent program.

An empirical approach to this problem would be to consider a range of model mappings that contain quantified levels of (one form of) neutrality. A factorial investigation of the effects on performance could be carried out by varying the locality due to 'non-neutral' changes across the mappings simultaneously. Such a study might offer insights into the relationship between these properties, though it would probably be most practical initially to investigate this in simplified situations (for example the weighted integer model described in Chapter 3).

8.2.2 Refinements to the use of the Mantel Statistic (Chapter 3)

Experiments with the Mantel statistic in Chapter 3 suggested that it is a robust measure of locality in the GPM at the syntactic level. The statistic can also be applied to generate measurements of locality at other levels in GP maps, for example between phenotype and fitness. This requires the selection of semantic metrics, such as those in Chapter 7. The technique could be improved by developing a better understanding of the sensitivity to these decisions. Additional comparisons could be made with locality measured under this statistic in other classes of GP representations, such as linear GP, or alternative operators. Further validation of the technique on other problem domains would also be appropriate.

From an implementation perspective, testing the significance of the Mantel statistic is a relatively expensive task, because it is carried out by permutation. A serial implementation was used in this work. Because permutations are applied over separate distance classes, calculating the test statistic lends itself well to parallel execution. Fast calculations of other permutation-based test statistics have been achieved using GPU implementations (Eklund *et al.*, 2011). Combined with efficient management of the stored distance matrices, such an implementation could enable evaluation of representation locality using the Mantel statistic over the course of an evolutionary run. This would be advantageous in situations where the GPM is time-dependent.

8.2.3 Graph-based Visualisation Methods (Chapter 4)

Graph-layout algorithms are a powerful tool when visualising high dimensional data, which have not been well-explored within this context in the GP literature. This may be in part because software to analyse networks on the scale required to illustrate GP search spaces has only become readily available relatively recently. The explosion in growth of network analysis tools can be seen as an opportunity to provide methods of inspecting and comparing the characteristics of different representations more directly. This work has only considered only force-directed algorithms, whose performance (as a

form of metaheuristic search) carries uncertainty. An avenue for further work would be to consider deterministic methods, closer to classical multidimensional scaling, such as spectral layout algorithms. Spectral layout uses the eigenvectors of the graph adjacency matrix or Laplacian matrix to provide fast methods of optimal layout (Brandes *et al.*, 2004).

Selecting samples for fitness graphs in such a way that they are most representative of an evolutionary run is an open question. The emphasis as proposed in this chapter has been to analyse existing GPM, rather than those developed concurrently with an evolutionary run and to consider the whole genotype space of simple problems. One natural extension appropriate for novel GPM would be to construct a graph of the mapping based on the regions explored at runtime. Repeated runs would focus on the graph-drawings on regions of the GPM that are active in the search.

8.2.4 Solution Concepts and Progress in CEA (Chapter 5)

It is an interesting aside to note that, whilst providing a much stronger theoretical foundation for coevolutionary algorithms, the coevolutionary solution concepts reviewed in this chapter represent only a small proportion of this extensive field of study in game theory and multi-agent systems. For example, it is well known in game theory that the concept of a Nash Equilibrium can be extended to the idea of a *Correlated equilibrium* (Aumann, 1974). Hybrid concepts, such as the *fuzzy-nash-pareto equilibrium* also exist (Dumitrescu *et al.*, 2010). Algorithms associated with these more general notions do not appear to have been investigated in the main body of coevolutionary literature.

To obtain theoretically robust forms of coevolution using GP representations, it will be necessary to understand how properties of the genotype to phenotype map can be related to the CEA solution concepts. Are some GP paradigms more apt for different solution concepts? It would seem stronger ties are required between the CEA and GP literature before these questions can be satisfactorily answered.

8.2.5 Extending Locality to Interaction Functions (Chapter 6)

Coevolutionary algorithms explore a space of interactions between genotypes. This poses the question, is there an analogous form of locality in this space? The definition of locality used in Chapter 6 is based directly on the aggregative measures used by Rothlauf (2006) and Chiam *et al.* (2008). The framework was designed for evolutionary algorithms and therefore focuses on changes in individual genotypes. For CEA, it is possible to extend these concepts to construct measures between collections of genotypes.

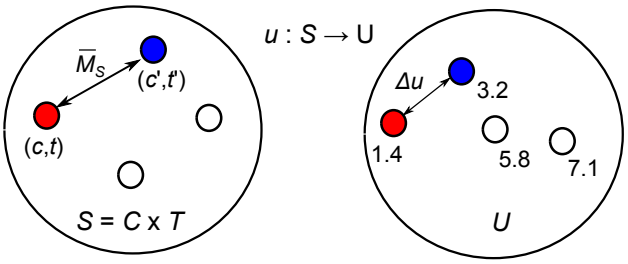

FIGURE 8.1: *Locality in a CEA interaction function for a test-based problem, mapping between pairs of candidate and test genotypes and interaction outcomes.*

In test based coevolution, search is carried out in the product space of potential solutions $S = C \times T$. Therefore, a neighbourhood structure exists for S that considers the probability of search operators transitioning between the interactions $(c, t) \in S$. Assume two pairs (c, t) and (c', t'). For an independently applied mutation operator s, this implies that the expected variation distance (Definition 3.3) can be extended to S as:

$$\bar{M}_S(c, t, c', t') = \frac{2}{Pr(c' = s(c)) \times Pr(t' = s(t))} \tag{8.1}$$

where $c \neq c'$, $t \neq t'$. The value $\bar{M}_S$ is then the expected number of mutation operations required when the mutation operator is applied simultaneously to a candidate and test genotype. The minimum of $\bar{M}_S$ is two, because the operator must be applied at least once to each individual. Assume a single outcome interaction function $u : (c, t) \to \mathbb{R}$. Let $U \subseteq \mathbb{R}$ denote the set of values over all $(c, t) \in S$. The natural measure for U is the absolute difference in outcome $\Delta u = |u(c, t) - u(c', t')|$. This gives a distance measure in both spaces.

If small $\bar{M}_S$ implies small Δu, then the mapping given by the interaction function between S and U could be described as having strong locality. The concept is illustrated in Figure 8.1. Future work would investigate whether characterising the interaction function in this fashion (in the 'landscape of interactions') provides a clearer relation with search performance.

8.2.6 CEA and Pathological Behaviours (Chapter 7)

Parallels in Natural Systems

The observations concerning disengagement and GPM locality in Chapter 7 bear some similarity to the reciprocal interactions observed in natural coevolving systems. Hanifin *et al.* (2008) report an example of a system where arms races on selected traits persist

until large phenotypic adaptations take place. The work studied the coevolutionary relationship between Garter snakes and their prey, species of toxic newt, examining newt toxicity against the resistance of snakes across regions of North America. An 'escalation zone' was characterised, within which traits in predator and prey are sufficiently competitive to increase both values:

> *"Initial increases in toxicity might be promoted by selection from interactions with other species as in other systems, including predators on early life stages. Some localities (e.g., Benton, Oregon) seem to persist in this escalation zone, while others (e.g., Omo, California) escape from the arms race due to rapid evolution of extreme resistance through simple genetic mechanisms. Such adaptive changes suspend reciprocal selection, and no counter escalation follows."*
>
> - **Hanifin *et al.* (2008)**

Disengagement observed in the GT variations has some parallels with this example. In the strongly local system, the probability of an objectively poor program adapting to an objectively good, faster adapting adversary (one with large output) was very low. Conversely, a weakly local GPM permitted some semantic transitions of sufficient magnitude such that adaptation was possible. The extent to which these tendencies are also true for biological systems (albeit acknowledging their far greater complexity) would be an original next step.

Metrics for Disengagement

A further open question concerning disengagement is whether the metric of Cartlidge and Ait-boudaoud (2011) (zero variance in fitness) is satisfactory for all situations. In problems where fitness has a noisy component, as given for example in the Game of Tag, an alternative might be to define disengagement as when fitness variance across the population falls below a certain threshold. The threshold value would distinguish between fitness gradients induced by the objective quality of opponents and gradients due to error in the interaction function.

Given the relationships observed between cycling, disengagement and mapping locality, it is reasonable to expect that properties of the genotype to phenotype map will also have an effect on the occurrence of other pathologies. For example, it can be conjectured that mapping locality will contribute to the likelihood of focusing in coevolutionary systems. Focusing is a consequence of overspecialisation on one or more phenotypic traits. In a strongly local mapping, changes to the phenotype are more likely to be incremental. This suggests a situation where the prevailing change is a small improvement to existing good

traits, which it can be inferred may lead to specialisation. The *focusing* number game is designed to analyse focusing in coevolutionary systems (Bucci and Pollack, 2003). The game could be adapted to examine algorithms using GPM to test this hypothesis, following similar principles to the experiments in this work.

8.3 Relationships with Open Issues in GP

Material covered in this thesis has partially contributed to a number of acknowledged outstanding issues in the field of genetic programming (O'Neill *et al.*, 2010).

8.3.1 Choice of Representation

Understanding the role of GPM locality in evolutionary search can directly inform choice of representation by a practitioner. The new methods of measuring and visualising locality described in this thesis are intended to assist GP practitioners with this process. Selecting a representation is as much an engineering issue as it is a theoretical one; algorithmic loss of performance due to a sub-optimal but simple GPM should be offset against the time to implement a more complex mapping. However, providing tools and theoretical guidelines will help to speed up this process. An emerging focus in GP is the distinction between syntactic/semantic levels of the mapping process and how to ensure indirect GP mappings retain locality given their greater complexity. The techniques set out here (the Mantel statistic and the application of Force-layout algorithms) have intentionally been constructed such that they can be applied flexibly to this problem using syntactic or semantic metrics.

Choosing representations in CEA is a largely unexplored area. The evidence from Chapters 6 and 7 suggests that, as is the case for non-coevolutionary forms of GP, mapping locality contributes to performance. Some recommendations can be inferred from the results in CO1, GT, GT-GP, Simple Cycler and the Game of Tag. Firstly, it has been seen that using a strongly local mapping can increase the likelihood of disengagement. Therefore, switching to a less local mapping whose properties are otherwise similar may improve performance when this pathology occurs. Secondly, in Simple Cycler it was noted that decreasing locality reduced cycling frequency. A possible explanation is that less local encodings can allow coevolutionary algorithms to escape intransitive cycles. Further experiments are required to determine whether this hypothesis is correct and the extent to which it may hold over other GP representations and problems.

8.3.2 Benchmarking

It has recently been acknowledged by the GP community that new benchmarks are required to progress the field (McDermott *et al.*, 2012). Two new constructed problems have been provided, the GP Greater-than-game and the Simple Cycler. The problems are scalable and designed to transparently test the occurrence of disengagement and cycling in coevolutionary GP. In addition, a framework for benchmarking in coevolutionary algorithms was set out in Chapter 6, that stressed the importance of measuring coevolutionary progress relative to a clearly defined solution concept. This is intended to help address the problem of making fair empirical comparisons between coevolutionary forms of GP.

Several further sources are available to generate additional coevolutionary GP benchmarks. One option is generalising existing GP evolutionary problems, such as symbolic regression, to a coevolutionary form. An alternative is to adapt existing coevolutionary GA benchmarks to GP (the approach used in Chapter 7). Both the MAX-GP and GT-GP problems are based on the concept of maximising program output; however GT-GP introduces coevolution by deriving fitness from the relative comparison of programs, rather than objectively.

8.3.3 Complexity Analysis

Introducing proofs of the time complexity for different GP paradigms on particular problem classes is a major goal of GP theory. In Appendix A, a proof is given for the expected time complexity of a simplified form of Cartesian Genetic Programming on the MAX-GP problem. Although this is a very constrained example, it is the first result of its kind for the CGP representation. Generalising the proof to a more realistic problem, such as a case from digital circuit evolution, could go some way to strengthening the theoretical basis of the technique.

As a longer term ambition, time complexity results could be related back to the properties of the GPM. It has been shown in genetic algorithms that for other factors such as mutation rate there exist critical boundaries between time complexity classes in response to changing parameters (Lehre and Yao, 2012). That such boundaries also exist for GPM properties seems probable. Mapping out how varying quantifiable GPM properties like locality change the complexity class of simple problems in a given GP algorithm, although a very challenging proposition, would be a significant contribution in this area.

8.4 Final Words

The tools for analysing locality presented in this thesis are intended to provide a foundation for the investigation and comparison of different evolutionary representations. In the field of genetic programming, these tools could be exploited to help assess newly proposed representations and search operators, particularly in conjunction with a communally maintained set of benchmarks. Locality is only one property of artificial GPM and no single set of tools is likely to provide a clear indicator of a maps suitability. However, by constructing a portfolio of methods to examine the range of possible GPM characteristics, a more principled approach to representation design could be achieved.

Presently, single population evolution is the most prevalent approach in genetic programming. As programs are required to incorporate further interactive, competitive and collaborative features, more extensive use of coevolution is likely to become a necessary advancement for the field. An understanding of the effects of GP representations on coevolutionary behaviours, as initiated here, will greatly help to realise this.

Appendix A

Performance of Cartesian GP on the Single Arity MAX problem

The MAX test case, used in Section 4.5.1 is the problem of maximising the output of a program given a fixed GP representation, function set and inputs. This appendix records calculations that show how the theoretical behaviour of a CGP representation on MAX can be bounded from above to expected polynomial performance in the problem size, for a simplified example. We will use the method of *artificial fitness levels*, as described in Lehre (2011). The basic concept of the method is to partition the search space into equivalent fitness levels consisting of subsets of genotypes whose phenotype's discrete fitness falls within the bounds of that level. The proof proceeds by bounding the expected number of levels and the number of evaluations required to transition between them. This work is based on theoretical approaches previously applied to the One MAX problem, the GA analogy of MAX (see for example the review of Oliveto *et al.* (2007)).

A.1 Assumptions

1) Assume a single row CGP genotype g with n nodes, with a terminal set containing a single input $T \in \mathbb{N}$ and function set $F = \{+1\}$. The function $+1$ is a single arity operation that adds one to its argument. With this simplification, the genotype can be written as a tuple of indexed nodes, $g = (g_1, g_2 ... g_n)$, where each node g_i is represented as a pair of integers $(j, 0)$ giving the index of the node input and function choice. The first integer j indicates a connection to any previous node or the input, $0 \leq j \leq g_i$. Given the single function choice, the second integer value is fixed at zero.

Remark: As an example, in the 4 node genotype (0 0)(0 0)(1 0)(1 0), nodes 1 and 2 are connected to the input, nodes 3 and 4 are connected to node 1. Given an input of 0, the corresponding outputs at each node are (1, 1, 2, 2) respectively. Each connected function adds one to the output along the encoded graph.

2) The phenotype $p = m(g)$ can be written as an ordered list of connected nodes $p = (p_1, p_2...p_l)$. In this representation, the MAX problem has a fitness function such that $f(p) = |p| \leq l$.

Definition A.1 (Fitness Level)**.** A *fitness level* is a subset $F_k \subseteq G$ such that for all $x, y \in F_k$, genotypes have equal fitness $f(x) = f(y) = k,\ k \in \mathbb{Z}$. Let F be the set of fitness levels $F = \{F_1 \cup F_2... \cup F_n\}$ in G.

Remark: The fitness of a genotype is given by the number of connected nodes. Each fitness level contains the subset of genotypes with equivalent, integer valued fitness.

3) Assume a mutation operator that operates on the genotype with a uniform probability of mutation $q = \frac{1}{n}$. Denote the probability that g mutates to g' under the uniform mutation operator s to be $Prob(s(g) = g') = Q_{g,g'}$. Then let $Q_g^k = \bigcup_{\forall g' \in F_k} Q_{g,g'}$.

Remark: The set Q_g^k contains those probabilities that correspond to all transitions between g and the genotypes in the fitness level F_k.

4) Assume a 1+1 EA using this configuration and mutation operator.

A.2 Proof for Single Arity MAX

Lemma A.2. *For all genotypes $g \in F_k$, $1 \leq k < n$, there exists a probability $Q \in Q_g^{k+1}$, such that $Q \geq \frac{1}{en^4}$.*

Proof. Let $g \in F_k$ and p be the corresponding phenotype. There exists at least one unconnected node in g, because $k < n$. Label two nodes g_a and g_b, where

g_a is the lowest index node in g, not in p.
g_b is the lowest index node in g, in p.

Consider the smallest number of edits to g required give a genotype $g' \in F_{k+1}$. This transition can always be made by editing connections to insert g_a. There are two cases

Case 1. Connect g_a to g_b, at the beginning of the phenotype, $(a < b)$.
Case 2. Connect g_a to g_n, in the middle of the phenotype, $(a > b)$.

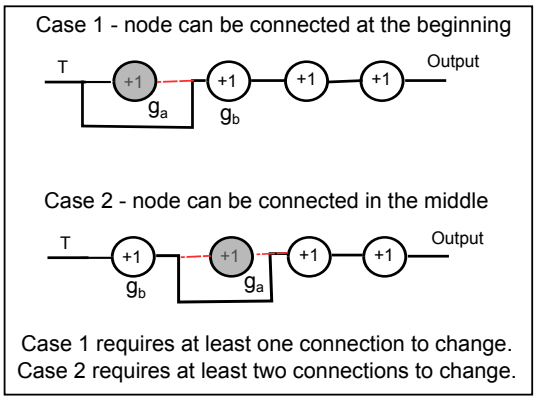

FIGURE A.1: *An illustration of the edits required to connect a redundant CGP node.*

At best one connection is changed in the first case, at best two connections are changed in the second case (see Figure A.1).

The probability of changing x nodes in g and keeping $n - x$ connections the same is $q^x(1-q)^{n-x}$, which given $q = \frac{1}{n}$ is smaller for $x = 2$ than $x = 1$. The probability of the x altered values mutating to the particular connections is bounded from below by $\frac{1}{n^x}$, which holds because the maximum number of potential connections for any node in g is n. Hence, for $x = 2$, combining these probabilities gives

$$\begin{aligned} Q_{g,g'} &\geq \left(\frac{q}{n}\right)^2 (1-q)^{n-2} \\ &\geq \frac{1}{n^4}(1-\frac{1}{n})^{n-2} \\ &\geq \frac{1}{en^4} \end{aligned}$$

Because $Q_{g,g'} \in Q_g^{k+1}$, this proves Lemma A.2. □

Theorem A.3. *The expected number of evaluations to optimise the single arity MAX problem using 1+1 CGP with a function set F and T has an upper bound $O(n^5)$.*

Proof. The proof follows directly from Lemma A.2. Let Q^k be the probability of transitioning from F_k to F_{k+1}, for all $g \in F_k$. From Lemma A.2, we have

$$Q^k \geq min(Q_g^k) \geq \frac{1}{en^4}$$

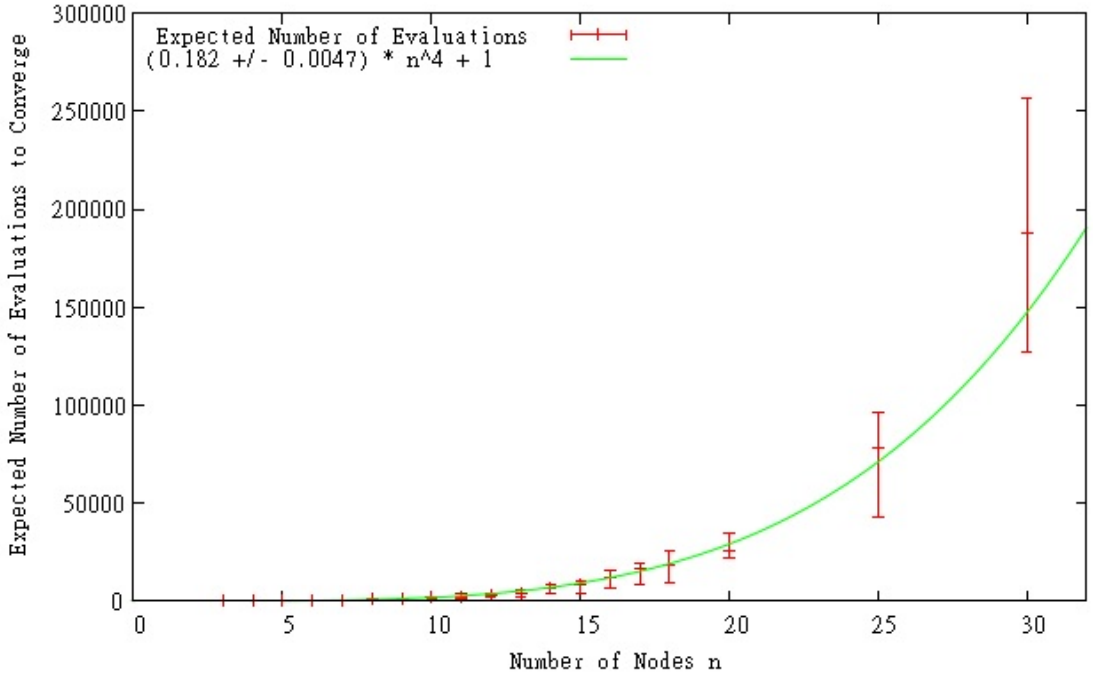

FIGURE A.2: *Scaling of the number of expected fitness evaluations for CGP on the single arity MAX-GP problem.*

Let the expected number of evaluations to transition in the 1+1 EA between levels F_k and $F_k + 1$ be $E(k)$. Then since $E(k) = \frac{1}{Q^k} \leq en^4$, we can sum to give

$$\begin{aligned} E(n) &\leq \sum_{k=1}^{k \leq n} E(k) \\ &\leq n \times en^4 \\ &\leq en^5 \end{aligned}$$

where $E(n)$ is expected number of evaluations to connect all n CGP nodes (the optimal fitness configuration). Therefore $E(n)$ can be bounded as $O(n^5)$. □

A.3 Experimental Comparison

Theorem A.3 can be validated experimentally. A 1+1 EA with the CGP representation as described was configured and run from a random starting configuration 500 times between $n = 1$ and $n = 20$. Figure A.2 shows the scaling of the expected number of evaluations to obtain the solution with respect to the number of nodes (problem size). Fitting by non-linear least squares gives the model $E(n) = (0.182 \pm 0.0047) \times n^4$, which supports Theorem A.3 as an upper bound for the expected performance on MAX-GP.

Abbreviations

BNF	**B**ackus-**N**aur **F**orm
CIAO	**C**urrent **I**ndividual vs. **A**ncestral **O**pponent
CEA	**C**o**e**volutionary **A**lgorithm
CO1	**C**ompare **O**n **One**
CGP	**C**artesian **G**enetic **P**rogramming
EC	**E**volutionary **C**omputing
EP	**E**volutionary **P**rogramming
FDC	**F**itness **D**istance **C**orrelation
GA	**G**enetic **A**lgorithm
GE	**G**rammatical **E**volution
GoT	**G**ame **o**f **T**ag
GO	**G**eneralisation **O**ptimisation
GP	**G**enetic **P**rogramming
GPM	**G**enotype to **P**henotype **M**ap
GT	**G**reater **T**han
HSB	**H**ue **S**aturation **B**rightness
IPCA	**I**terated **P**areto **C**oevolutionary **A**rchive
NCD	**N**ormalised **C**ompression **D**istance
NFL	**N**o **F**ree **L**unch
PP	**P**ure **P**ursuit
PN	**P**roportional **N**avigation
RSH	**R**andomised **S**earch **H**euristic
SC	**S**imple **C**ycler

Bibliography

Amor, H. (2005). Intelligent exploration for genetic algorithms: using self-organizing maps in evolutionary computation. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1531–1538, Washington D.C, USA.

Angeline, P. J. and Pollack, J. B. (1993a). Coevolving High-Level Representations. In Langton, C. G., editor, *Artificial Life III*, pages 55–71. Addison-Wesley.

Angeline, P. J. and Pollack, J. B. (1993b). Competitive Environments Evolve Better Solutions for Complex Tasks. In *Proceedings of the 5th Interational Conference on Genetic Algorithms*, pages 264–270, Urbana-Champaign, IL, USA.

Antonarakis, S. E. and Beckmann, J. S. (2006). Mendelian disorders deserve more attention. *Nature Reviews Genetics*, 7(4):277–282.

Aumann, R. J. (1974). Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96.

Axelrod, R. (1987). The Evolution of Strategies in the Iterated Prisoner's Dilemma. In *Genetic Algorithms and Simulated Annealing*, chapter 3, pages 32–41. Morgan Kaufmann, New York, USA.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann.

Beadle, L. (2009a). *Semantic and Structural Analysis of Genetic Programming*. PhD thesis, University of Kent.

Beadle, L. (2009b). Semantically driven mutation in genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1336–1342.

Biernacki, P. (1981). Snowball Sampling, Problems and Techniques of Chain Referral Sampling. *Sociological Methods And Research*, 10(2):141–163.

Brameier, M. F. and Banzhaf, W. (2007). *Linear Genetic Programming*. Genetic and Evolutionary Computation Series, Springer.

Brandes, U., Harel, D., and Koren, Y. (2004). Spectral Graph Drawing. In *Handbook of Graph Drawing and Visualisation*, chapter 15, pages 1–20. CRC Press, LLC.

Bucchi, A. (2007). *Emergent Geometric Organization and Informative Dimensions in Coevolutionary Algorithms*. PhD thesis, Brandeis University.

Bucci, A. and Pollack, J. (2003). Focusing versus Intransitivity. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 250–261, Chicago, IL, USA.

Byrne, J., McDermott, J., O'Neill, M., and Brabazon, A. (2010). An analysis of the behaviour of mutation in grammatical evolution. In *Proceedings of the 13th European Conference on Genetic Programming*, pages 14–25.

Carrington, P. J., Scott, J., and Wasserman, S. (2005). *Models and Methods in Social Network Analysis (Structural Analysis in the Social Sciences)*. Cambridge University Press, 1st edition.

Cartlidge, J. (2004). *Rules of Engagement : Competitive Coevolutionary Dynamics in Computational Systems*. PhD thesis, University of Leeds.

Cartlidge, J. and Ait-boudaoud, D. (2011). Coevolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 15(2):215–229.

Cartlidge, J. and Bullock, S. (2004). Unpicking Tartan CIAO Plots: Understanding Irregular Coevolutionary Cycling. *Adaptive Behavior*, 12(2):69–92.

Caruana, R., Eshelman, L. J., and Schaffer, J. (1998). Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. In *Proceedings of the 5th International Conference on Machine Learning*, pages 153–161, Hong Kong.

Chiam, S., Goh, C., and Tan, K. (2006). Issues of Binary Representation in Evolutionary Algorithms. In *Proceedings of the IEEE International Conference on Intelligent Systems*, pages 1–8, Bangkok, Thailand.

Chiam, S., Tan, K., Goh, C., and Al Mamun, A. (2008). Improving locality in binary representation via redundancy. *IEEE Transactions on Systems, Man, and Cybernetics (B)*, 38(3):808–25.

Cliff, D. and Miller, G. F. (1995a). Co-evolution of Pursuit and Evasion II: Simulation Methods and Results. In *From Animals to Animats: Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour*. University of Sussex.

Cliff, D. and Miller, G. F. (1995b). Tracking The Red Queen: Measurements of adaptive progress in co-evolutionary simulations. In *Proceedings of the 3rd European Conference on Advances in Artificial Life*, pages 200–218, London, UK.

Collins, J. (1999). Visualization of evolutionary algorithms using principal components analysis. In *Proceedings of the Genetic Algorithms and Evolutionary Computation Conference*, pages 99–100, Orlando, FL, USA.

Collins, M. (2006). Finding needles in haystacks is harder with neutrality. *Genetic Programming and Evolvable Machines*, 7(2):131–144.

Collins, T. (1998a). *Using Software Visualisation Technology to help Genetic Algorithm Designers*. PhD thesis, The Open University, UK.

Collins, T. D. (1998b). Understanding Evolutionary Computing : A Hands on Approach. In *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 564–569, Anchorage, AK, USA.

Cramer, N. L. (1985). A representation for the adaptive generation of simple sequential programs. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 183–187, Pittsburgh, PA, USA.

Dawkins, R. and Krebs, J. R. (1979). Arms Races between and within Species. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 205(1161):489 – 511.

D'haseleer, P. and Bluming, J. (1994). Effects of Locality in Individual and Population Evolution. In *Advances in Genetic Programming, vol. 2*, pages 177–198. MIT Press.

Drezewski, R. and Obrocki, K. (2009). Co-operative Co-evolutionary Approach to Multi-objective Optimization. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems:*, pages 277–284, Salamanca, Spain.

Droste, S., Jansen, T., and Wegener, I. (2002). Optimization with Randomized Search Heuristics - The (A)NFL Theorem, Realistic Scenarios, and Difficult Functions. *Theoretical Computer Science*, 287(1):131–144.

Droste, S. and Wiesmann, D. (1998). On Representation and Genetic Operators in Evolutionary Algorithms. Technical report, Collaborative Research Center (SFB) 531 : [249], University of Dortmund.

Dumitrescu, D., Lung, R., Mihoc, T., and Nagy, R. (2010). Fuzzy Nash-Pareto Equilibrium: Concepts and Evolutionary Detection. In *Proceedings of EvoApplications 2010*, pages 71–79, Istanbul, Turkey.

Dybowski, R., Collins, T., and Weller, P. (1996). Visualization of Binary String Convergence by Sammon Mapping. In *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 377–383, San Diego, CA.

Eades, P. and Hong, S. (2012). How to Draw a Graph, Revisited. In *Expanding the Frontiers of Visual Analytics and Visualization*, chapter 7, pages 111–126. Springer, London.

Eklund, A., Andersson, M., and Knutsson, H. (2011). Fast Random Permutation Tests Enable Objective Evaluation of Methods for Single-Subject FMRI Analysis. *International Journal of Biomedical Imaging*, 2011:627947.

Ficici, S. G. (2004). *Solution Concepts in Coevolutionary Algorithms*. PhD thesis, Brandeis University.

Ficici, S. G. and Pollack, J. B. (1998). Challenges in Coevolutionary Learning: Arms-Race Dynamics, Open-Endedness, and Mediocre Stable States. In *Proceedings of the 6th International Conference on Artificial Life*, pages 238–247, Pasadena, CA, USA.

Ficici, S. G. and Pollack, J. B. (2003). A Game-Theoretic Memory Mechanism for Coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 286–297, Chicago, IL, USA.

Floreano, D. and Nolfi, S. (1997). God save the red queen! Competition in co-evolutionary robotics. In *Proceedings of the 2nd Annual Conference on Genetic Programming*, pages 398–406, San Francisco, CA, USA.

Floreano, D., Nolfi, S., and Mondada, F. (1998). Competitive Co-Evolutionary Robotics : From Theory to Practice. In *From Animals to Animats, 5th International Conference on Simulation of Adaptive Behaviour*, volume 4, pages 515–524, Zurich, Switzerland.

Fogel, D. B. (1994). An introduction to simulated evolutionary optimization. *Neural Networks, IEEE Transactions on*, 5(1):3–14.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution*. John Wiley and Sons, Inc., New York, 1st edition.

Forsyth, R. (1981). Beagle: A darwinian approach to pattern recognition. *Kybernetes*, 10(3):159–166.

Friedberg, R. (1958). A learning machine. *IBM Journal of Research and Development*, 2(1):2–13.

Fruchterman, T. M. J. and Reingold, E. M. (1991). Graph Drawing by Force-directed Placement. *Software-Practice and Experience*, 21(11):1129–1164.

Galvan-Lopez, E., McDermott, J., O'Neill, M., and Brabazon, A. (2011a). Defining locality as a problem difficulty measure in genetic programming. *Genetic Programming and Evolvable Machines*, 12(4):365–401.

Galvan-Lopez, E., Poli, R., Kattan, A., ONeill, M., and Brabazon, A. (2011b). Neutrality in evolutionary algorithms what do we know? *Evolving Systems*, 2(3):145–163.

Goldberg, D. E. (1989). *Genetic Algorithms in Search and Optimisation*. Addison-Wesley, 1st edition.

Goslee, S. C. and Urban, D. L. (2007). The ecodist Package for Dissimilarity-based Analysis of Ecological Data. *Journal Of Statistical Software*, 22(7):1–19.

Graff, M. and Poli, R. (2010). Practical performance models of algorithms in evolutionary program induction and other domains. *Artificial Intelligence*, 174(15):1254–1276.

Gustafson S., Vanneschi, L. (2008). Crossover-Based Tree Distance in Genetic Programming. *IEEE Transactions on Evolutionary computation*, 12(4):506–524.

Hamilton, W. D. (1980). Sex versus Non-Sex versus Parasite. *Oikos*, 35(2):282 – 290.

Hanifin, C. T., Brodie, E. D. J., and Brodie, E. D. I. (2008). Phenotypic mismatches reveal escape from arms-race coevolution. *PLoS Biology*, 6(3):e60.

Hansen, N. and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 312–317, Nagoya, Japan.

Hart, E. and Ross, P. (2001). GAVEL - A New Tool for Genetic Algorithm Visualization. *IEEE Transactions on Evolutionary Computation*, 5(4):335–348.

Haynes, T., , Sen, S., Schoenfield, D., and Wainwright, R. (1995). Evolving multiagent coordination strategies with genetic programming. Technical Report UTULSA-MCS-95-05, The University of Tulsa.

Hien, N. T. and Hoai, N. X. (2006). A Brief Overview of Population Diversity Measures in Genetic Programming. In *Proceedings of the 3rd Asian-Pacific Workshop on Genetic Programming*, pages 128–139, Hanoi, Vietnam.

Hillis, W. D. (1985). *The Connection Machine*. PhD thesis, Massachusetts Institute of Technology.

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimisation procedure. *Physica D*, 42(1-3):228–234.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michegan Press, Ann Arbor, Michegan.

Hu, T., Payne, J. L., Banzhaf, W., and Moore, J. H. (2011). Robustness , Evolvability , and Accessibility in Linear Genetic Programming. In *Proceedings of the 14th European Conference on Genetic Programming*, pages 13–24, Torino, Italy.

Hu, Y. (2005). Efficient and high quality force-directed graph drawing. *The Mathematica Journal*, 10(1):37–71.

Hugosson, J., Hemberg, E., Brabazon, A., and O'Neill, M. (2010). Genotype representations in grammatical evolution. *Applied Soft Computing*, 10(1):36–43.

Jacomy, M., Heymann, S., Venturini, T., and Bastian, M. (2011). ForceAtlas2 , A Graph Layout Algorithm for Handy Network Visualization. http://www.medialab.sciences-po.fr/publications/.

Jacomy, M. B., Heymann, S., and Jacomy, M. (2009). Gephi: An Open Source Software for Exploring and Manipulating Networks. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, pages 361–362, San Jose, CA, USA.

Janzen, D. H. (1980). When is it Coevolution? *Evolution*, 34(3):611–612.

Jaskowski, W. and Krawiec, K. (2009). Formal analysis and algorithms for extracting coordinate systems of games. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 201–208, Milan, Italy.

Jaskowski, W. and Krawiec, K. (2011). Formal analysis, hardness, and algorithms for extracting internal structure of test-based problems. *Evolutionary Computation*, 19(4):639–71.

Jones, T. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, School of Computer Science, The University of New Mexico.

Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, USA.

Jong, E. (2003). Representation Development from Pareto-Coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 262–273, Chicago, IL, USA.

Jong, E. (2005). The MaxSolve Algorithm for Coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 525–536, Washington D.C, WA, USA.

Jong, E. (2007). A Monotonic Archive for Pareto-Coevolution. *Evolutionary Computation*, 15(1):61–93.

Jong, E. and Pollack, J. B. (2004). Ideal Evaluation from Coevolution. *Evolutionary Computation*, 12(2):159–192.

Kauffman, S. A. and Johnsen, S. (1991). Coevolution to the edge of chaos: coupled fitness landscapes, poised states, and coevolutionary avalanches. *Journal of Theoretical Biology*, 149(4):467–505.

Keijzer, M. (1996). Efficiently Representing Populations in Genetic Programming. In *Advances in Genetic Programming 2*, chapter 13, pages 259–278. MIT Press.

Kim, Y. K., Kim, J. Y., and Kim, Y. (2004). A Tournament-Based Competitive Coevolutionary Algorithm. *Applied Intelligence*, 20(3):267–281.

Kouchakpour, P., Zaknich, A., and Bräunl, T. (2008). A survey and taxonomy of performance improvement of canonical genetic programming. *Knowledge and Information Systems*, 21(1):1–39.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1st edition.

Koza, J. R., Keane, M. A., and Streeter, M. J. (2003). What's AI Done for Me Lately? Genetic Programming's Results. *IEEE Intelligent Systems*, 18(3):25–31.

Krawiec, K. (2011). Semantically Embedded Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1379–1386, Dublin, Ireland.

Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming*. Springer.

Legendre, P. and Fortin, M. J. (1989). Spatial pattern and ecological analysis. *Vegetatio*, 80(2):107–138.

Legendre, P. and Fortin, M.-J. (2010). Comparison of the Mantel test and alternative approaches for detecting complex multivariate relationships in the spatial analysis of genetic data. *Molecular Ecology Resources*, 10(5):831–844.

Legendre, P., Lapointe, F., and Cagrain, P. (1994). Modeling brain evolution from behavior: a permutational regression approach. *Evolution*, 48(5):1487–1499.

Legendre, P. and Legendre, L. (1998). *Numerical Ecology*. Developments in Environmental Modelling, Elsevier, 2nd edition.

Lehre, P. (2011). Fitness-levels for non-elitist populations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 2075–2082, Dublin, Ireland.

Lehre, P. K. and Yao, X. (2012). On the impact of mutation-selection balance on the runtime of evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 16(2):1–38.

Lichodzijewski, P. and Heywood, M. I. (2007). Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 464–471, London, England.

Lichstein, J. W. (2006). Multiple regression on distance matrices: a multivariate spatial analysis tool. *Plant Ecology*, 188(2):117–131.

Lindgren, K. and Johansson, J. (2001). Coevolution of strategies in the n-person Prisoner's Dilemma. In Crutchfield, J. and Schuster, P., editors, *Evolutionary Dynamics - Exploring the Interplay of Selection, Neutrality, Accident, and Function*, pages 5–9. Addison-Wesley, Reading, MA, USA.

Lipson, H., Bongard, J., and Zykov, V. (2005). Co-Evolutionary Methods in System Design and Analysis. In *Proceedings of the 15th International CIRP Design Seminar*, Shanghai, China.

Luke, S. and Panait, L. (2006). A comparison of bloat control methods for genetic programming. *Evolutionary Computation*, 14(3):309–44.

Luke, S. and Spector, L. (1996). Evolving Teamwork and Coordination with Genetic Programming. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 150–156, Stanford, CA, USA.

Mabu, S., Hirasawa, K., and Hu, J. (2007). A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and its Extension Using Reinforcement Learning. *Evolutionary Computation*, 15(3):369–398.

Mahner, M. and Kary, M. (1997). What exactly are genomes, genotypes and phenotypes? And what about phenomes? *Journal of Theoretical Biology*, 186(1):55–63.

Mantel, N. (1967). The detection of disease clustering and a generalized regression approach. *Cancer Research*, 27(2):209–20.

McCandlish, D. M. (2011). Visualizing fitness landscapes. *Evolution*, 65(6):1544–58.

McDermott, J., Galvan-Lopez, E., and O'Neill, M. (2011). A Fine-Grained View of GP Locality with Binary Decision Diagrams as Ant Phenotypes. In *Parallel Problem Solving from Nature (PPSN XI)*, pages 164–173, Krakov, Poland.

McDermott, J., White, D. R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., and O'Reilly, U. M. (2012). Genetic Programming Needs Better Benchmarks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 791–798, Philadelphia, PA, USA.

McIntyre, A. and Heywood, M. (2008). Cooperative problem decomposition in Pareto competitive classifier models of coevolution. In *Proceedings of the 11th European Conference on Genetic programming*, pages 289–300, Naples, Italy.

McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y., and ONeill, M. (2010). Grammar-based Genetic Programming: a survey. *Genetic Programming and Evolvable Machines*, 11(3-4):365–396.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., and Teller, A. H. (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21(1078):1087–1093.

Miconi, T. (2009). Why Coevolution Doesnt Work: Superiority and Progress in Coevolution. In *Proceedings of the 12th European Conference on Genetic Programming*, pages 49–60, Tubingen, Germany.

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

Miller, J. F., editor (2011). *Cartesian Genetic Programming*. Springer, 1st edition.

Miller, J. F., Thomson, P., and Fogarty, T. (1997). Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study. In *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications*. Wiley.

Moraglio, A. (2007). *Towards a geometric unification of evolutionary algorithms*. PhD thesis, University of Essex.

Moriarty, D. and Miikkulainen, R. (1998). Forming Neural Networks through Efficient and Adaptive Coevolution. *Evolutionary Computation*, 5(4):373–399.

Murphy, E., Neill, M. O., and Brabazon, A. (2011). Examining Mutation Landscapes in Grammar Based Genetic Programming. In *Proceedings of the 14th European Conference on Genetic Programming*, pages 130–141, Torino, Italy.

Nash, J. F. (1950). Equilibrium points in N-Person Games. *Proceedings of the national academy of sciences*, 36(1):48–49.

Newcombe, R. G. (1998). Two-sided confidence intervals for the single proportion: comparison of seven methods. *Statistics in medicine*, 17(8):857–72.

Nguyen, Q. U., , Hoai, N., O'Neill, M., and McKay, B. (2011a). The role of syntactic and semantic locality of crossover in genetic programming. In *Parallel Problem Solving From Nature (PPSN XI)*, pages 533–542, Krakow, Poland. Springer.

Nguyen, Q. U. (2011). *Examining Semantic Diversity and Semantic Locality of Operators in Genetic Programming.* PhD thesis, School of Computer Science and Informatics, University College Dublin.

Nguyen, Q. U., Nguyen, X. H., O'Neill, M., McKay, R. I., and Galvan-Lopez, E. (2011b). Semantically-based crossover in genetic programming: Application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119.

Nolfi, S. and Floreano, D. (1998). Co-evolving Predator and Prey Robots: Do "Arms races" Arise in Artificial Evolution? *Artificial Life*, 4(4):311–335.

Nordin, P., Banzhaf, W., and Francone, F. (1999a). *Advances in genetic programming*, volume 3, chapter 12 "Efficient Evolution of Machine Code for CISC Architectures Using Instruction Blocks and Homologous Crossover", pages 275–302. MIT Press.

Nordin, P., Hoffmann, F., Francone, F., Brameier, M., and Banzhaf, W. (1999b). AIM-GP and parallelism. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, pages 1059–1066, Washington D.C, USA.

Oden, N. and Sokal, R. (1986). Directional autocorrelation: an extension of spatial correlograms to two dimensions. *Systematic Biology*, 35(4):608.

Oliveto, P., He, J., and Yao, X. (2007). Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293.

O'Neill, M., Brabazon, A., Nicolau, M., Garraghy, S. M., and Keenan, P. (2004). PI Grammatical Evolution. In *Proceedings of the Genetic Algorithms and Evolutionary Computation Conference*, pages 617–629, Seattle, WA, USA.

O'Neill, M. and Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers, 1st edition.

O'Neill, M., Vanneschi, L., Gustafson, S., and Banzhaf, W. (2010). Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363.

Paredis, J. (1994). Coevolutionary Computation. *Artificial Life*, 2(4):355–375.

Paredis, J. (1996). Coevolutionary Life-time Learning. In *Parallel Problem Solving from Nature (PPSN IV)*, pages 72–80, Berlin, Germany.

Paredis, J. (1997). Coevolving Cellular Automata: Be Aware of the Red Queen! In *Proceedings of the 7th International Conference on Genetic Algorithms*, San Francisco, CA, UK.

Payne, A. and Stepney, S. (2009). Representation and Structural biases in CGP. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1064–1071, Trondheim, Norway.

Perkis, T. (1994). Stack-based genetic programming. In *Proceedings of the IEEE Conference on Evolutionary Computation.*, pages 148–153, Orlando, FL, USA.

Pham, T. A., Nguyen, Q. U., Nguyen, X. H., and O'Neill, M. (2013). Examining the diversity property of semantic similarity based crossover. In *Proceedings of the 16th European Conference on Genetic Programming*, pages 265–276.

Pohlheim, H. (2006). Understanding the Course and State of Evolutionary Optimizations using Visualization. *Artificial Life*, 12(2):217–227.

Poli, R. (1996). Parallel Distributed Genetic Programming. Technical Report CSRP-96-15, School of Computer Science, University of Birmingham.

Poli, R. and Graff, M. (2009). There is a free lunch for hyper-heuristics, genetic programming and computer scientists. In *12th European Conference on Genetic Programming*, pages 195–207, Tubingen, Germany.

Poli, R., Langdon, W., and McPhee, N. (2008). *A field guide to genetic programming*. Creative Commons, 1st edition.

Poli, R., Vanneschi, L., Langdon, W., and McPhee, N. (2010). Theoretical Results in Genetic Programming : The Next Ten Years? *Genetic Programming and Evolvable Machines*, 11(3-4):285–320.

Popovici, E., Bucci, A., Wiegand, R. P., and Jong, E. (2012). *Coevolutionary Principles*, chapter 31, pages 987–1034. Springer-Verlag, Berlin.

Popovici, E. and Jong, K. (2009). Monotonicity versus performance in co-optimization. In *Foundations of Genetic Algorithms*, pages 151–170, Orlando, FL, USA.

Potter, M. A. and Jong, K. (1994). A Cooperative Coevolutionary Approach to Function Optimization. In *Parallel Problem Solving from Nature (PPSN-'94)*, pages 249–257, Jerusalem, Israel.

Raidl, G. and Gottlieb, J. (2005). Empirical Analysis of Locality, Heritability and Heuristic Bias in Evolutionary Algorithms: A Case Study for the Multidimensional Knapsack Problem. *Evolutionary Computation*, 13(4):441–475.

Rechenberg, I. (1973). *Evolutionsstrategic: Optimierung technischer Systeme nach Prinzipen der biologischen Evolution*. Frommmann-Holzboog Verlag, Stuttgart, 1st edition.

Reynolds, C. W. (1994). Competition , Coevolution and the Game of Tag. In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 59–69, MIT, MA, USA.

Romero, G., Merelo, J., Castillo, P., Castellano, J., and Arenas, M. (2002). Genetic Algorithm Visualisation Using Self-organizing Maps. In *Parallel Problem Solving from Nature VII*, pages 442–451, Birmingham, UK.

Rosin, C. (1997). *Coevolutionary Search Among Adversaries*. PhD thesis, San Diego, California.

Rothlauf, F. (2006). *Representations for Genetic and Evolutionary Algorithms*. Springer Berlin Heidelberg.

Rothlauf, F. and Oetzel, M. (2006). On the Locality of Grammatical Evolution. In *Proceedings of the European Conference on Genetic Programming*, pages 320–330, Budapest, Hungary.

Rowe, J., Whitley, D., Barbulescu, L., and Watson, J.-P. (2004). Properties of gray and binary representations. *Evolutionary computation*, 12(1):47–76.

Ryan, C., Collins, J., and O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In *Proceedings of the First European Workshop on Genetic Programming*, pages 83–96, Paris, France.

Schmidt, M. D. and Lipson, H. (2008). Coevolution of Fitness Predictors. *IEEE Transactions on Evolutionary Computation*, 12(6):736–749.

Schwefel, H.-P. (1977). *Numerische Optimierung von Computer-modellen mittels der Evolutionsstrategie*. Birkhauser Verlag, Basel, 1st edition.

Seaton, T., Brown, G., and Miller, J. F. (2010). Analytic solutions to differential equations under graph-based genetic programming. In *Proceedings of the 13th European Conference on Genetic Programming*, pages 232–243, Istanbul. Springer Berlin.

Seaton, T., Clarke, T., and Miller, J. F. (2011). Locality in co-optimisation problems. In *Proceedings of the 4th York Doctoral Symposium*, pages 43–51, York, UK.

Seaton, T., Miller, J. F., and Clarke, T. (2012a). An ecological approach to measuring locality in linear genotype to phenotype maps. In *Proceedings of the 15th European Conference on Genetic Programming*, pages 170–181, Malaga, Spain.

Seaton, T., Miller, J. F., and Clarke, T. (2012b). Visualisation of fitness landscapes using graph-layout algorithms. In *Proceedings of the 5th York Doctoral Symposium*, page 84, York, UK.

Seaton, T., Miller, J. F., and Clarke, T. (2013). Semantic bias in program coevolution. In *Proceedings of the 16th European Conference on Genetic Programming*, Vienna, Austria. To appear.

Sendhoff, B., Kreutz, M., and Seelen, W. (1997a). A condition for the genotype phenotype mapping : Causality. In *Proceedings of the 7th International Conference on Genetic Algorithms*, San Francisco, CA, USA.

Sendhoff, B., Kreutz, M., and von Seelen, W. (1997b). Causality and the analysis of local search in evolutionary algorithms. Internal Report 97-16, Insitut fur Neuroinformatik, Ruhr-Universitat Bochum.

Service, T. C. (2009). Unbiased coevolutionary solution concepts. In *Foundations of Genetic Algorithms*, pages 121–130, Orlando, FL, USA.

Service, T. C. and Tauritz, D. R. (2008). A No-Free-Lunch Framework for Coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 371–378, Montreal, Canada.

Shine, W. and Eick, C. (1997). Visualizing the evolution of genetic algorithm search processes. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 367–372, Indianapolis, IN, USA.

Shirakawa, S. and Nagao, T. (2009). Graph Structured Program Evolution with Automatically Defined Nodes. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1107–1114, Montreal, Canada.

Shukla, U. and Mahapatra, P. R. (1990). The proportional navigation dilemma-pure or true? *IEEE Transactions on Aerospace and Electronic Systems*, 26(2):382–392.

Sikulova, M. and Sekanina, L. (2012). Coevolution in Cartesian Genetic Programming. In *Proceedings of the 15th European Conference on Genetic Programming*, pages 182–193, Malaga, Spain.

Silva, S. and Costa, E. (2009). Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10(2):141–179.

Sims, K. (1994). Evolving 3d morphology and behaviour by competition. *Artificial Life*, 4(1):353–372.

Smith, S. (1980). *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, University of Pittsburgh.

Socievole, A. and Marano, S. (2012). Exploring user sociocentric and egocentric behaviors in online and detected social networks. In *Proceedings of the 2nd Baltic Congress on Future Internet Communications*, pages 140–147, Vilnius, Lithuania.

Spector, L. and Robinson, A. (2002). Genetic Programming and Autoconstructive Evolution with the Push Programming Language. *Genetic Programming and Evolvable Machines*, 3(1):7–40.

Stadler, P. F. (1996). Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20(1):1–45.

Stanley, K. O. and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21(1):63–100.

Swafford, J. M., Hemberg, E., O'Neill, M., Nicolau, M., and Brabazon, A. (2011). A non-destructive grammar modification approach to modularity in grammatical evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1411–1418, Dublin, Ireland.

Tay, J. C., Tng, C. H., and Chan, C. S. (2008). Environmental effects on the coevolution of pursuit and evasion strategies. *Genetic Programming and Evolvable Machines*, 9(1):5–37.

Teller, A. (1993). Learning Mental Models. In *Proceedings of the 5th Workshop on Neural Networks*, San Francisco, CA, USA. Society for Computer Simulation.

Teller, A. and Veloso, M. (1996). PADO: A new learning architecture for object recognition. In *Symbolic Visual Learning*, pages 81–116. Oxford University Press.

Tesauro, G. (1989). Neurogammon Wins Computer Olympiad. *Neural Computation*, 1(3):321–323.

Valen, L. V. (1973). A new evolutionary law. *Evolutionary Theory*, 1(1):1–30.

Vanneschi, L. (2004). *Theory and Practice for Efficient Genetic Programming*. PhD thesis, University of Lausanne.

Vassilev, V. K. (2000). *Fitness Landscapes and Search in the Evolutionary Design of Digital Circuits*. PhD thesis, University of York.

Waddington, C. H. (1942). Canalization of development and the inheritance of acquired characters. *Nature*, 150(3811):563–565.

Walker, J. A. and Miller, J. F. (2008). The automatic acquisition, evolution and reuse of modules in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 12(4):397–417.

Walker, M., Edwards, H., and Messom, C. (2007). The reliability of confidence intervals for computational effort comparisons. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1716–1723, London, UK.

Watson, R. and Pollack, J. (2001). Coevolutionary Dynamics in a Minimal Substrate. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 702–709, San Francisco, CA, USA.

Weinberger, E. (1990). Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336.

Whigham, P. (1995). Grammatically-based Genetic Programming. In JP, R., editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41, Tahoe City, CA, USA.

White, D. and Poulding, S. (2009). A Rigorous Evaluation of Crossover and Mutation in Genetic Programming. In *Proceedings of the 12th European Conference on Genetic Programming*, pages 220–231, Tubingen, Germany.

White, D. R., Arcuri, A., and Clark, J. A. (2011). Evolutionary Improvement of Programs. *IEEE Transactions on Evolutionary Computation*, 15(4):515–538.

Whitley, D. (1999). A free lunch proof for gray versus binary encodings. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 726–733, Orlando, CA, USA.

Wiegand, R., Liles, W., and Jong, K. (2002). The Effects of Representational Bias on Collaboration Methods in Cooperative Coevolution. In *Parallel Problem Solving from Nature (PPSN VII)*, pages 257–268, Granada, Spain.

Wiegand, R. P. (2003). *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University.

Wiegand, R. P. and Sarma, J. (2004). Spatial Embedding and Loss of Gradient in Cooperative Coevolutionary Algorithms. In *Parallel Problem Solving from Nature (PPSN VIII)*, pages 912–921, Birmingham, UK.

Wijngaarden, R. P. T. and Jong, E. (2008). Evaluation and Diversity in Co-evolution. In *Parallel Problem Solving in Nature (PPSN X)*, pages 631–640, Dortmund, Germany.

Wilson, D. and Kaur, D. (2009). Search, Neutral Evolution, and Mapping in Evolutionary Computing: A Case Study of Grammatical Evolution. *IEEE Transactions on Evolutionary Computation*, 13(3):566–590.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390.

Wolpert, D. H. and Macready, W. G. (1996). No Free Lunch Theorems for Optimization. *IEEE Transactions On Evolutionary Computation*, 1(1):67–82.

Wolpert, D. H. and Macready, W. G. (2005). Coevolutionary Free Lunches. *IEEE Transactions On Evolutionary Computation*, 9(6):721–735.

Wright, S. (1988). Surfaces of Selective Value Revisited. *The American Naturalist*, 131(1):115–123.

Yu, T. and Miller, J. F. (2006). Through the Interaction of Neutral and Adaptive Mutations Evolutionary Search Finds a Way. *Artificial Life*, 12(4):526–551.

Yuan, L. C. (1948). Homing and Navigational Courses of Automatic Target-Seeking Devices. *Journal of Applied Physics*, 19(12):1122–1129.