UNIVERSITY OF SOUTHAMPTON

Inferring and Exploiting Compact Models of Evolutionary Problem Structure

by

Chris R. Cox

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Physical Sciences and Engineering
School of Electronics and Computer Science

May 2015

Academic Thesis: Declaration Of Authorship

I, ...CHRIS R. COX..... [please print name]

declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

[title of thesis] Inferring and Exploiting Compact Models of Evolutionary Problem Structure

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. 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;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Either none of this work has been published before submission, or parts of this work have been published as: [please list references below]:

   Cox, C. R., & Watson, R. A. (2014). Solving building block problems using generative grammar. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*. ACM.

   Cox, C. R., & Watson, R. A. (2014). Inferring and Exploiting Problem Structure with Schema Grammar. In *Parallel Problem Solving from Nature–PPSN XIII* (pp. 404–413). Springer International Publishing.

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Chris R. Cox

In both natural and artificial evolution, populations search a space of possibilities using the mechanisms of natural selection and random variation. However, not all variations are equally likely. The directions which variation can take are themselves a key part of the evolutionary machinery, determining the ability of evolution to create diversity whilst obeying the constraints of phenotype space. To be effective they must reflect the structure of the selective environment in which they exist. Evolutionary algorithms are often designed with a priori assumptions about this structure, but it can also be learned on the fly using "model-building" algorithms. However, there are many open questions: what information do populations contain about their selective environment? How can it be extracted from a population and represented? And how can it be exploited to facilitate more effective evolutionary search?

In this thesis, a novel type of lossless compact model called Schema Grammar is introduced. Schema Grammar overcomes the current limitations of compact models by enabling intrinsically non-sequential data to be compressed. It offers a number of advantages over existing model-building approaches. In particular, the model is able to infer a hierarchy of genetic schemata that is consistent with the compositional structure of the selective environment, and has a strong predictive quality with respect to fitness. By using this structure to facilitate variation at many different levels of scale, instances of well-known test problems are shown to be solvable in low-order polynomial time, matching the performance of state of the art methods.

The information-theoretic qualities of Schema Grammar also enable evolutionary information to be quantified in novel ways. Building on recent advances in information and complexity theory, the model is used to quantify mutual information between populations and their selective environment, including environments containing complex epistatic structure. It is also used to predict the fitness of individuals by measuring their information distance to a fit population.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

BBO	Black Box Optimisation
BBH	Building Block Hypothesis
BOA	Bayesian Optimisation Algorithm
CFG	Context Free Grammar
CSS	Compositional Schema Structure
DTP	Distance To Population
ECGA	Extended Compact Genetic Algorithm
EA	Evolutionary Algorithm
EDA	Estimation of Distribution Algorithm
ENS	Evolution by Natural Selection
FOS	Family Of Subsets
G-P	Genotype to Phenotype (Map)
GA	Genetic Algorithm
GE	Grammatical Evolution
GP	Genetic Programming
GRN	Gene Regulation Network
hBOA	Hierarchical Bayesian Optimisation Algorithm
HDF	Hierarchically Decomposable Function
HIFF	Hierarchical If-And-Only-If
KC	Kolmogorov Complexity
LT	Linkage Tree
LTGA	Linkage Tree Genetic Algorithm
MD	Mahalanobis Distance
MDL	Minimum Description Length
MSS	Multi Scale Search
NFL	No Free Lunch
NID	Normalised Information Distance
PCA	Principal Component Analysis
PMBGA	Probabilistic Model Building Genetic Algorithms
PMIC	Population Model Information Content
SG	Schema Grammar
TSP	Travelling Salesman Problem

Acknowledgements

I would like to give the warmest of thanks to the students and staff of the Institute of Complex Systems Simulation at the University of Southampton. You gave me a welcome home in which to explore my ideas, taught me things I didn't think I was capable of learning, gave me support beyond the call of duty, and left me with a collection of lifelong friendships. To my supervisor Richard Watson: thank you for listening to my nutty ideas, helping me shape them into something resembling science and encouraging me to get this thing finished! I wish you the best of luck with your grand theory of everything. My deepest gratitude goes to my wife Narissa, for unconditionally encouraging and supporting me throughout the whole journey.

Dedicated to Grandma

Chapter 1

Introduction

The motivation for this work is the pursuit of evolutionary search algorithms that can effectively solve a wide range of relevant problems. We know that in the space of *all* possible problems neither Evolution by Natural Selection (ENS), nor any other search algorithm, can offer advantages over any other, including random trial and error (Wolpert & Macready, 1997). However, the problems that ENS faces in nature and the hard computational problems we consider in this thesis are not all possible problems. They are in fact representative of particular subclasses of problem and they contain regularity. By tacitly presupposing the nature of this regularity it is possible for an algorithm to learn and then exploit the intrinsic constraints of each problem, without requiring specific a priori knowledge. This ability can hold a powerful advantage over non-adaptive search.

What, then, is the nature of regularity in the space of relevant problems? This thesis follows the work of Simon (1962), who observed that natural systems are often decomposed into a hierarchy of semi-independent parts. He referred to these as *nearly-decomposable systems*. The concept of near-decomposability can be observed in biological systems, in which life forms are typically constituted from a nested hierarchy of parts. It can also be observed in human social systems, where people are often organised in a hierarchy of groups functioning at various levels of scale. In the context of problem solving, near-decomposability relates to problems that can be decomposed into a hierarchy of semi-independent sub-problems. This means that good solutions to sub-problems at one level of detail often form component parts of good solutions at larger scales. Many relevant and interesting problems are structured in this way. If problem solvers are able to identify sub-solutions at the most granular level of detail, they can be recombined to find sub-solutions at larger scales. The idea is that by repeating a cycle of search, problem decomposition and recombination recursively, relevant problems can be solved "from the bottom up" in sub-exponential time, without requiring specific details of the problem instance to be known a priori.

In this thesis we study discrete optimisation problems in which each solution is defined by a fixed number of dimensions over a finite alphabet. We follow a long line of research in Genetic Algorithms (GAs) and Evolutionary Algorithms (EAs), which shows that the decomposition of these problems can be understood in terms of *schema structure*. Schemata are combinatorial patterns representing solutions to semi-independent sub-problems. Schemata are equivalent to hyperplanes. Their fitness is defined as the average fitness of all the solutions they contain. In the nearly-decomposable problems we consider here, fit schemata at one level of specificity can be composed together to form even fitter schemata at a more specific scale. For algorithms that are able to detect and recombine relevant schemata, this can create a type of compositional fitness gradient, leading from solutions with some fitness advantage to the global optimum.

The key contribution of this thesis is a novel type of evolutionary model building algorithm. Model building algorithms attempt to infer the schema structure implied by a selective environment and use this knowledge to facilitate evolutionary variation. The literature has suggested a variety of ways to do this, both implicit and explicit. These will be reviewed in Chapter 2. Whilst this work has provided algorithms that are highly effective on a variety of problems, some key challenges remain. Model building algorithms must be able to infer schema structure that is multivariate in nature and often organised in irregular, overlapping forms. However, current techniques are often not capable of capturing this structure without loss of information. Algorithms must also be able to use their models to guide search. Recent work has suggested how evolution might best exploit knowledge of its selective environment. Developing these ideas and combining them with effective models remains an important and open direction for scientific investigation.

This context leads us to the research questions that will be investigated here:

- How can complex schema structure be inferred without loss of information?
- What is the most effective way of exploiting schema structure in evolutionary search?

1.1 Lossless Compression as a Modelling Strategy

Our starting point for investigating these questions is the Minimum Description Length (MDL) principle (Rissanen, 1978), which holds that the best explanation of a data set is the *most compact* representation of it. This principle relates to exact rather than approximate representations of data and thus describes compact models of the *lossless* kind. There are two key advantages of adopting the MDL perspective for model building. Firstly, it is impossible to overfit the model to the data, as the model must describe the data exactly. Secondly, given a formal equivalence between compression and probabilistic

prediction, MDL drives models towards a superior predictive performance (Grünwald, 2007).

Where the MDL Principle doesn't help is explaining *how* data should be compressed. In fact, there is no right answer to this question. There is an equivalence between the idealised MDL model of data and its Kolmogorov Complexity (KC), which is defined as the shortest computer program that can generate the data (Kolmogorov, 1968). The idea is that the more regularity there is in the data, the lower the KC, and in our case the smaller the idealised MDL model. For two reasons, however, this quantity is unknowable: it is formally uncomputable, and it depends upon the language used to implement the program (Li & Vitányi, 2009). What this means in practical terms is that the MDL Principle is a relative concept: it doesn't tell us how to implement a lossless model of the data, it just tells us that the smaller that model can be made, the better.

It is easy to see the appeal of the principle with respect to the research questions. In short, the intuition is that the most compact description of an evolutionary population must implicitly capture the regularities within it, and is therefore the best model. MDL-inspired algorithms have been used in evolutionary model building before, but only to infer compact *statistical* models of a population, not to model the population itself (Pelikan, 2002; Harik et al., 2006). As we will see, an ability to do the latter provides powerful advantages. We can now formulate an overarching research hypothesis that this thesis will investigate:

> **Research hypothesis: compact models of fit populations can reflect the compositional structure of the selective environment and have strong predictive qualities with respect to fitness.**

In investigating this hypothesis we build directly on the work of Toussaint (Toussaint, 2006; Toussaint & von Seelen, 2007), who first suggested the idea of "compact models as a search strategy" and the concept of *Compression Evolutionary Algorithms* (Compression EAs). Toussaint's work provides inspiration and theoretical grounding for our hypothesis. It shows formally that one way of minimising the distance between a model of a population and the population itself is to find a *compact representation* of the population that minimises description length. It shows how compact representations can be implemented by using complex G-P maps that encode repeating structure. Moreover, it describes the potential power of compact representations as a search strategy. If complex variable interactions from phenotype space are captured in a G-P map, what is left is a new compact genetic encoding in which variables are independent. This scheme enables random exploration in genotype space, whilst respecting complex dependencies in phenotype space.

Toussaint demonstrated the concept of Compression EAs by inferring and exploiting G-P maps using sequential lossless compression techniques. The problem with these techniques is that they can only capture structure that is sequentially contiguous: they are unable to capture order-independent *combinatorial* interactions. To use sequential compression as the basis of a model building EA, biased sequential representations are required that arrange related alleles together on the genome (the idea of *tight linkage*, see Section 2.1.1). Given that the state of the art in evolutionary model building has successfully overcome this limitation, sequential compression is not fit for purpose as a contemporary modelling strategy. This shortcoming has inhibited adoption of Toussaint's ideas in evolutionary model building (Yu et al., 2009).

1.2 Schema Grammar

In this thesis we introduce a novel type of compact grammatical model that overcomes the constraints of sequential compression, which we refer to as ***Schema Grammar (SG)***. We show that this model offers a principled framework for describing and quantifying the information content of a population. Moreover, we show that SG is able to infer a *language* that reflects the schema structure of the underlying problem. We then use an SG-based Compression EA to show the power of this structure as an exploration distribution.

The cornerstone of the SG solution is a representation that encodes solutions as an *unordered set* of symbols, and production rules that specify recursive *set membership* relations. The symbology of the representation defines each allele and its functional context (locus) as a unique *terminal symbol*. As an example, consider an 8-dimensional bit string written as 10010101. In the SG representation, terminal symbols exist for each allele in a particular functional context. We use the notation x_l^a to define a terminal symbol representing an allele a at location l. Using this representation, the example individual can be redefined as the set $\{x_1^1, x_2^0, x_3^0, x_4^1, x_5^0, x_6^1, x_7^0, x_8^1\}$, which has no inherent sequential order. This representation makes explicit the combinatorial information represented by individuals and schemata, whilst removing the possibility of positional bias.

Inference of SG models is performed using a type of lossless compression. The algorithm recursively identifies co-occurring patterns of two or more symbols, representing epistatic interactions occurring anywhere on the genome. If we imagine an epistatic interaction in the selective environment, say between x_6^1 and x_8^1, then in a population of fit individuals these symbols are likely to co-occur together frequently. A lossless compression algorithm working on frequency of co-occurrence in the manner just described may detect this co-occurrence and form a new symbol s with a rewriting rule of $s \rightarrow \{x_6^1, x_8^1\}$. Not only can this schema be used to compress the original population, it tells us something about the

epistatic structure of the problem. Relating this back to the search space, what each of these symbols really represents is a *hyperplane*, and the compression algorithm is using these hyperplanes to describe regularity in the search space more compactly. Essentially, the G-P map inferred by SG decomposes information in the population into a hierarchy of schemata.

The fact that the SG model is a compact, lossless model of a population also creates new capabilities that are beyond existing approaches to evolutionary model building. The size of any such model can be regarded as an estimate of its KC. Essentially, compressing a population with SG and measuring the size of the result estimates the *absolute* information contained by the population. In this thesis we will combine this property with recent theoretical work to show how the absolute mutual information between a population and its selective environment can be quantified using SG. In other words, we show how it is possible to measure the amount of information contained in a population about fitness. The ability of SG to detect epistatic interactions of many orders differentiates it from existing attempts to do the same thing. As well as breaking new ground for evolutionary model building, this is a significant contribution in its own right.

1.3 Claims

In this thesis we use *compact* to mean a lossless description of a population that approximates its KC. Compact does not necessarily mean a globally minimal description. Rather, it means a description that is *compressed enough* to approximate KC. The SG grammatical formalism we will introduce in Section 3.3.4 does not satisfy this definition of compactness in itself. Indeed, it would be perfectly possible to evolve, infer or design SG grammar instances that are *not* compact. In what follows we use the term "SG model" to mean the grammatical formalism *combined with* the offline inference algorithm of Section 3.4. This algorithm is one possible means of generating compact instances of SG. It would also be possible to evolve, infer or design compact instances as an alternative. The SG model is not fully deterministic nor guaranteed to infer the smallest possible representation of a population. However, the work in this thesis demonstrates that it is compact *enough*. This includes quantitive evidence that SG approximates KC better than alternatives.

The usefulness of compact models is dependent upon the degree that we accept the MDL Principle. In other words, that the smallest representation is the "best" representation. This thesis provides good reasons for supporting this principle. It does this by showing that compact models can infer a schema structure that is consistent with the intrinsic structure of a decomposable problem and be used to achieve start of the art problem solving performance.

We can now specify some specific claims of the thesis:

- **SG can infer a compact model of a sample population without sensitivity to positional bias.**
- **By modelling a representative population of fit individuals, SG can infer a schema structure that is consistent with the compositional structure of the selective environment.**
- **By using SG to infer an exploration distribution that reflects the constraints of phenotype space, global function optimisation performance can be achieved that is comparable with the state of the art in evolutionary algorithms.**
- **By exploiting the information-theoretic properties of SG as a compact model, the information content of evolutionary populations and individuals can be quantified with more accuracy than alternative methods.**

1.4 Thesis Structure

The chapters of this thesis are used to provide evidence for these claims and prove the overarching hypothesis on a variety of different problem classes. Chapter 2 examines the nature of evolutionary problem structure. It shows how many relevant problems can be decomposed into a schema structure and describes the different forms this structure can take. It then reviews the field of model building EAs, both implicit and explicit, and describes the state of the art. It introduces an emerging class of evolutionary variation, *Multi-Scale Search* (MSS), that offers distinct advantages over alternative approaches. MSS is used in Chapter 5 to define an SG-based Compression EA.

Chapter 3 describes and then formally defines SG as a compact model based on *context free grammar* (CFG). It defines a symbolic representation for discrete search problems using the model. Different types of schema structure are then considered, and it is shown how they can be modelled using the grammar. A lossless offline compression algorithm is then defined, with a theoretical time complexity of $O(n^2p)$, where n is the number of problem dimensions and p is population size.

Chapter 4 demonstrates that the model inferred by SG can accurately capture the schema structure of the selective environment. Model instances are inferred from populations of fit individuals, visualised, and compared to the original problem. This work shows that not only does the model accurately reflect the structure of the environments on which it is tested, the model's schema structure reveals a compositional fitness gradient to the fittest solutions. This appeals to the intuition behind most model building

EAs – that problems can be solved by recursively inferring and recombining fit schemata together.

In Chapter 5, the power of the exploration distribution inferred by SG is explored. Several classes of test problem with important theoretical properties are introduced. The mutant spectra of the SG distribution is analysed for these problems. This reveals that by using simple variation on the SG representation, more fit novel offspring can be generated than by using alternative methods. This is then put to the test in global function optimisation experiments, using a Compression EA that combines SG with MSS. The results show that SG is able to solve each of the tested problem classes efficiently, with performance comparable or superior to state of the art methods.

Chapter 6 shows how the SG model can be usefully applied to create new measures of evolutionary information. Although quantitative measures of evolutionary information have been suggested in the past, these are limited by their insensitivity to epistatic interaction. The measures are used to show that SG is able to quantify the information that a population has inferred about its selective environment, even when the environment contains complex structure. The work also shows that the measures have a strong predictive quality with respect to fitness. By parsing previously unseen individuals and measuring their information content, the model is able to accurately predict their fitness, matching state of the art classification methods from machine learning.

The insights and contributions of this work are discussed in Chapter 7. The chapter considers the implications for evolutionary model building, theoretical biology and machine learning.

Chapter 2

Model Building in Evolutionary Computation

This chapter reviews the field of model building algorithms in evolutionary computation. The first part of the chapter introduces evolutionary problem structure and the key concepts used to describe it. This provides a conceptual framework for comparing different approaches to model building and understanding the contributions of this thesis. Topics in this section include unbiased black box optimisation (BBO), near decomposability, schema structure and modularity. In the second part of the chapter, different approaches to implicit and explicit model building are reviewed from the literature. The theory and practice of implicit model building using crossover-based GAs is described, along with the work that highlight its limitations. The start of the art in explicit model building algorithms is then described, covering probabilistic models and more recent alternative techniques.

This chapter does not review all the literature relevant to the contributions of this thesis. In the interests of readability and relevance, some review topics have been moved to the chapters to which they are most pertinent. Concepts that have specific relevance to compact models, including lossless compression, Kolmogorov Complexity (KC) and subspace clustering methods, are introduced in Chapters 3 and 6. Some schema-related theory, including deception and Walsh Analysis is included in Chapter 4. Finally, some search algorithm specifics, which are used to create comparison methods in Chapter 5, are covered in that chapter.

2.1 Evolutionary Problem Structure

2.1.1 Unbiased Black Box Optimisation

It is known from the *No Free Lunch* (NFL) theorems for search and optimisation that in the space of *all* possible problems, no search process is better than any other, including random search (Wolpert & Macready, 1997). These theories do not mean that non-random search processes (like evolution) offer no advantages. Rather, they show that this is only the case if the subset of problems to which they are applied is so wide that over all the problems in the subset, there are no observable correlations between problem inputs and outputs. More formally, this mean subsets of problems that are *closed under permutation* (Schumacher et al., 2001). This may seem like an extreme case, and indeed it is: Igel & Toussaint (2003) show that the fraction of problem subsets in which this criteria holds is vanishingly small. Within any narrower range of problems, specific algorithms *can* offer comparative advantages over others – something that Droste et al. (2002) describe as "free appetisers".

The practical implication of free appetisers is that if search algorithms employ some intrinsic knowledge of a particular problem domain, then they will offer better than random performance on problems within that domain. Algorithms can be highly specific, targeting relatively small subsets of problem instances of a particular class and achieving very good search performance as a result. This is the case with many of the heuristic and metaheuristic problem solving methods that are used in operations research. In contrast, many workers in the field of evolutionary computation have attempted to find evolution-inspired search algorithms, perhaps even *the* algorithm, that can effectively generalise to as many "interesting" problems as possible. In a computational setting this is known as black box optimisation (BBO), and "interesting" problem scope usually translates to NP-hard problems[1].

A useful additional distinction is between variation methods that have an a priori bias towards a certain arrangement of problem dimensions or input values and those that do not. Lehre & Witt (2012) formally categorise these methods of variation as *biased* and *unbiased* respectively. This distinction is particularly relevant in the context of GAs, as the bias inherent in most crossover methods requires the dimensions of solution space to be arranged into a sequential representation, with any related dimensions in a close physical proximity. This is known as *tight linkage*. The efficacy of single and multi-point crossover is drastically reduced on problems that do not adhere to this kind of *positional bias* (Eshelman et al., 1989; Spears & De Jong, 1990). As we will see, the state of the art in evolutionary model building, including the algorithms introduced by this thesis, has moved well beyond this limitation. The field can now offer *unbiased* BBO algorithms

[1] Arguably, if any algorithm assumes that problems fall within a limited scope then the box can no longer be described as black. Nonetheless, we adopt the term here and use it as it is commonly understood.

with no positional bias, that have proven to be highly effective on a range of NP-hard problems.

2.1.2 Near Decomposability

NFL theory tells us that in order for any unbiased BBO algorithm to be effective (on average), then some regularity must exist across the space of problems to which it is applied. Compelling evidence of such a regularity is provided by unbiased algorithms that offer *some* added value across many different classes of NP-hard problem (e.g., hill-climbers, simulated annealing, GAs with uniform crossover). Inspiration can also be taken from the natural world, where the often spectacular performance of ENS on diverse problems suggests, by the same theory, a common order to them. The extent and the precise nature of the regularity in both natural systems and hard computational problems is not currently known. The school of thought followed here is inspired by Simon (1962), who observed that complex systems often contain regular structural properties. Simon argued that systems are often naturally *decomposable* into a set of semi-independent sub-systems, where there are strong dependencies *within* sub-systems and weak dependencies *between* sub-systems. He called these systems *nearly decomposable*, and illustrated the idea with examples drawn from biology, physicochemical systems and human social systems. He also suggested that these systems are often arranged into a hierarchy – in the words of Simon, "subsystems that, in turn, have their own subsystems, and so on". These observations are also consistent with recent work in evolutionary biology, which shows that modular organisation is central to developmental processes (Schlosser & Wagner, 2004), and the current understanding of topological modularity in networks (Newman, 2006).

Simon's original concepts have led to new ways of thinking about evolutionary problem structure. They have shown that good solutions to many interesting problems can be composed from a combination of good solutions to smaller sub-problems. This has also provided a new way of thinking about evolutionary problem solving. In contrast to the traditional perspective of evolution, which views adaptation as a gradual accumulation of individual traits, a *compositional* perspective suggests evolutionary adaption can work by recursively identifying and recombining partial solutions together (Watson, 2006). This philosophy can be seen at the heart of the implicit and explicit model building algorithms reviewed in Section 2.2, and the Schema Grammar (SG) approach that will be introduced by this thesis.

The performance of model building algorithms is largely dictated by their ability to identify the relevant partial solutions for a given problem and recombine them in an effective way. Given NFL, it is perhaps unlikely that a universal form to these relevant partial solutions can exist for all problems of interest, let alone an algorithm that could consistently learn it. In this thesis we concentrate on structural forms that can be expressed

in terms of epistatic interactions between alleles. These interactions and their fitness effects can be formalised using *schemata*, which are described in the following section. Algorithms that can infer and exploit these structures, either directly or indirectly, have been proven effective on a variety of interesting and relevant NP-hard problems.

The form that decomposed schema structure can take depends on the problem. We will consider classes of problem that decompose into regular, modular forms, and classes of problem that decompose into irregular, randomised forms. This allows us to understand and compare the capabilities of different types of model building algorithm. It also allows the SG model, which is capable of inferring regular and irregular schema structure of many different orders, to be fully understood. Model building algorithms may not infer schemata directly. For example, they may first learn modular decomposition through *linkage learning* and then sample module configurations from the population, or from a probabilistic model. These approaches are reviewed in Section 2.2. Regardless of whether inference of schema structure is direct or indirect, the ability of a model to capture the structure of a problem is reflected in its ability to *recall* these patterns.

2.1.3 Schemata

Schemata are the central formalism we will use in this thesis for describing problem structure. Every candidate partial solution can be understood as a pattern containing a set of individual solution features. The fitness of each pattern can also be understood in terms of the average fitnesses of all solutions which match that pattern. Holland (1975) formalised this idea for discrete solution spaces with his concept of *schemata*. Essentially, a collection of alleles that constitute a partial solution can be defined in terms of a hyperplane over the partition of the problem space (collection of loci) on which the alleles are defined. Each hyperplane is called a schema. The effect on fitness of each schema is defined using the average fitness of solutions contained by the hyperplane. If the alleles in each schema do not all interact with respect to fitness, then it is possible to describe their fitness effects using lower order schemata, and the information represented by the schema is redundant. If they do interact, then the schema represents potentially useful information about problem structure that cannot be further decomposed. This is particularly important for efficient solving if the interaction has superadditive effects on fitness.

We use the conventional notation for schemata found in the evolutionary computation literature (e.g., Holland, 2000). This notation assumes a bit-string representation for n-dimensional solutions over a binary alphabet. We can represent all possible schemata in a solution space using the set $H = \{1, 0, *\}^n$. Each schema $h \in H$ can be represented by an n-bit pattern, for example $h = 10{**}10{**}$. Each $*$ is a wildcard that signifies that no allele is specified in this dimension. The 1s and 0s define a hyperplane in the partition they are specified. From the representation, it can be seen that there are 3^n possible

schemata in an n-dimensional binary problem space. The *order* of a schema refers to the number of specified alleles; the *defining length* refers to the distance between the first and last specified alleles (in a sequential bit string interpretation). The order and defining length of the previous example are 4 and 6 respectively.

The fitness of a schema $f(h)$ is the average fitness of solutions that are contained by that schema, or equivalently the average fitness of solutions that match the pattern specified by the schema. For example in a 4-bit problem the fitness of the schema $f(1{**}1)$ is calculated using the following equation:

$$f(1{**}1) = \frac{1}{4}\left[f(1001) + f(1011) + f(1101) + f(1111)\right] \tag{2.1}$$

The number of solutions contained by an order k schema is then $2^{(n-k)}$. We say a higher-order schema h_2 (which can be a complete solution) is *contained* by a schema h_1 if h_2 matches the pattern specified by h_1. For example the schema 1*** contains the schema 1**0, both of which contain the schema 11*0. We can also say that high-order schema 11*0 is contained by both 1*** and 1**0 (and *1** etc). Note that containment here runs contrariwise to the idea of strings and substrings. For example the string "hello world" contains the substring "world". However, even though the solution 11110000... appears as though it contains the schema 1111****..., it does not: the solution is a single point in the space and the schema is a (potentially vast) hyperplane than contains $2^{(n-4)}$ such solutions. As schemata get larger in order they thus specify an increasingly smaller subspace, and this can be a confusing aspect of the string representation.

The comparative fitness of any schema depends upon the fitnesses of all schemata in that partition – in other words, the alternative substates. Therefore, we can say that each schema is a competitor in a *hyperplane competition* (Whitley, 1991). For example, consider the order-2 schema 11***. The other competitors in this competition are the schemata 00***, 01*** and 10***. Each order-k competition will have 2^k competitors, each with their own average fitnesses. The *winner* of the hyperplane competition is the schema with the highest fitness. It is also beneficial for us to describe schema in each competition as being of *above-average* or *below-average* schema fitness. The average schema fitness is constant for all hyperplane competitions and is equal to the average fitness of all solutions in the space.

2.1.4 Building Blocks

The term *building blocks* is used frequently in the evolutionary computation literature when referring to problem structure. It has both a generic and a specific meaning. The specific meaning comes from the Schema Theorem and Building Block Hypothesis (BBH) (see Section 2.2.1) and refers to schemata of above-average fitness with a short defining

length. This meaning is used to describe features that could be feasibly recombined using single or multipoint crossover without disruption. The generic meaning of building blocks in the literature is fit sub-solutions to decomposable problems. Intuitively, the best solutions are constructed by combining together building blocks. In this context, a building block can be interpreted as a schema with above-average or competition-winning fitness and is independent of order and defining length. Unless otherwise specified, this is the meaning we will adopt here.

2.1.5 Modularity

Since Simon first introduced his abstract concept of near decomposability, researchers in the field of evolutionary computation have attempted to create more concrete definitions of modularity and hierarchy. The concepts we describe here are based on a series of works by Watson and his collaborators (Watson et al., 1998; Watson, 2002; de Jong et al., 2004; Watson & Pollack, 2005; Watson, 2006). This work categorises the relationships between different subparts of a system, although it does not prescribe any method for decomposing a problem into these parts. Along with a number of associated texts, it has spawned a number of synthetic test problems that evaluate the ability of algorithms to infer and exploit modular and hierarchically modular structure, some of which we will use in the thesis. These test problems are commonly known as *building block problems*, as the fitness function is typically defined in terms of modular building block composition.

Consider a decomposition of a problem space into a set of disjoint partitions, each of which we refer to as *modules*. The hyperplane competitors in each module are known as *module configurations*, the number of which are denoted as $C = 2^k$, where k is the order of the partition. For any given module, we regard any module configuration that is optimal in some context to be of potential value in composing a solution. We denote the number of these module configurations as C'. If $C' = 1$ (for all modules), we say the problem is *separable*, as the optimal configuration of each module does not depend on any other. If $1 < C' < C$ then interdependency exists between subproblems. In this case, we say that the problem is decomposable but not separable. If $C' = C$, then the problem is not decomposable.

This definition of separability does not necessarily imply independence, rather that fitness must monotonically increase when sub-solutions are combined together, creating a type of compositional fitness gradient. This is more general, and thus arguably more useful, than definitions of separability that are based on *additively decomposable functions* (Pelikan & Hauschild, 2012). If a problem is separable down to the level of individual variables then we say it is *fully separable*. This is functionally equivalent to the idea of *convexity* in real-valued optimisation problems (Bertsekas, 2003). Fully separable problems can be solved trivially by algorithms that use a one-bit variation neighbourhood, such as simple hill-climbing.

If problems are either decomposable or separable, then there is an opportunity for model building algorithms to reduce the combinatorial complexity of the search. Firstly, this requires them to learn (either implicitly or explicitly) the correct decomposition of modules and module configurations. Even if a problem is separable this can be difficult, as the optimal module configurations may only be inferred amongst many potentially interesting configurations. Secondly, it requires them to recombine them in an efficient way. The difficulty posed by this step is often overlooked, and random recombination of module configurations will often only yield the optimal answer in exponential time. Recent work has shown that often the most efficient way to facilitate recombination is using incremental variation where the variation neighbourhood is adapted using the inferred structure of the problem. This is the idea behind Multi-Scale Search (MSS) (see Section 2.2.5).

Hierarchical modular interdependence describes modular interdependency that simultaneously exists at multiple levels of scale. Essentially, in problems with this structure, the best solutions to sub-problems at the lowest levels are composed to form the best solutions to sub-problems at the next level and so forth. This creates an analogy to Simon's original concept of hierarchy in the form of an evolutionary problem. The idea is best characterised by the HIFF problem (Watson et al., 1998), which is detailed in Section 4.3.3. In order to overcome hierarchical modular interdependency efficiently, it is not just necessary to infer fit modular configurations at one level, but recursively as search progresses. This introduces the idea of variation neighbourhoods having to adapt dynamically to reflect emerging knowledge of problem structure.

2.1.5.1 Modular Schema Structure

In Chapter 4, SG models are used to infer a schema structure from modular and hierarchically modular problems. This begs the question, which schema structure correctly represents the problem? In other words, what does modularity *look like* in terms of schemata? Although this depends on the fitness function structure, it is useful to provide a general illustration, which we do here using a binary GA representation.

The partition occupied by a module directly translates to a schema partition. Module configurations of interest (i.e., the configurations that make up C') are schema instances specified over that partition. As an example, consider a fitness function with a modular composition, with a module covering the first 2 dimensions (on a sequential bit string representation) of an 8-dimensional solution space. Let us say there are two module configurations that are optimal in alternative contexts – an all-1s and all-0s configuration. This structure is represented by two schemata: 11****** and 00******, which we would expect an accurate model to be able to recall.

The presence of modular interdependence in a problem means that combinations of module configurations interact with respect to fitness. For a model to capture relevant structure, it should be able to recall interactions that are optimal in some context. Consider an identical module present in the third and fourth dimensions of the previous example. Let us say there are two combinations that meet these criteria, 1111***** and 0000****, which we would expect an accurate model to be able to recall. To effectively promote such interactions when they are the result of (modular) sign epistasis or reciprocal sign epistasis (Weinreich et al., 2005), where the individual module configurations may be less favourable, requires the model to capture structure at a higher level of organisation. This is also the case if synergistic interactions are to be effectively exploited by search.

Hierarchical modular interdependence in a problem means the presence of modular interdependence at multiple levels of scale. If the form of the hierarchy in the fitness function is a tree, as is the case with HIFF (Watson et al., 1998), HXOR (ibid.), hierarchical trap functions (Pelikan & Goldberg, 2000) and H-SBB (Mills et al., 2014), then the modules are arranged as a strictly nested structure. In these problems, it is also the case that the hierarchical epistatic interactions are synergistic. In other words, there is a fitness bonus to be gained from co-ordinating modules. Following the reasoning in the previous paragraph, to effectively exploit these synergistic interactions in search, these interactions need to be captured by the model. In schema terms, this translates to a nested hierarchy of optimal module configurations. Using an 8-bit (unshuffled) HIFF function as an example, the relevant module configurations are the following:

$$00******, 11******, **00****, **11****, ****00**, ****11**, ******00, ******11$$
$$0000****, 1111****, ****0000, ****1111$$
$$00000000, 11111111$$

2.1.6 Irregular Forms

Outside of the building block problems that are used to test algorithms on modular structures, the arrangement of epistasis in real problems is rarely neat. In reality, beneficial interactions may occur on partitions of many different orders, which can overlap in an irregular way throughout the problem space. Structure of this nature is usually referred to simply as as *overlap*. Its defining characteristic is an unpredictability in the form of interactions, which makes it distinct from hierarchical modularity, in which modules also overlap, but usually in a strictly nested tree (Yu et al., 2009). Overlap complicates modelling considerably as modular structure at each level of scale cannot typically be captured using a disjoint linkage model. Many of the explicit modelling methods we will consider are unable to model overlapping problem structure. Overlapping, irregular structures are not inconsistent with the formal definitions of modularity

provided by Watson, but most problems that have been designed to exhibit modularity do not have irregular, overlapping structures.

It has been suggested that irregular structure in problems originates from what is known as *backbones*. The term was first coined by Monasson et al. (1999), and refers to the subset of variables that are common in all globally optimal solutions, or in the case of graph colouring problems, the subset of order-2 interactions that are common. Prugel-Bennett (2007) refined this idea by suggesting *critical backbones* as the "small critical subset of variables necessary to be in the basin of attraction of the global solution". The fittest solutions to a problem are likely to exist on multiple backbones that relate to each other in a complex way. Therefore, in a population under selection, inference of the relevant backbones is likely to involve inference of complex overlapping structure. In the words of Prugel-Bennett: "a successful search algorithm is one that can effectively search the space of critical backbones".

2.1.6.1 Irregular Schema Structure

Irregular structure is manifested in fit schemata that do not follow a predictable topology. Interactions may overlap in the space of alleles, for example 1111**** and **00***, in terms of loci, for example 1111**** and **111**, or both, for example 1111**** and **011*1*. This presents a particular difficultly for linkage-learning algorithms, as fit interactions often cannot be recalled by recombining a set of disjoint module configurations. The SG model that we introduce in Chapter 3 can infer a hierarchical schema structure containing complex overlapping forms, and this is demonstrated in Chapter 4.

The relative advantage conferred by modelling overlapping schema structure is not yet known. Interestingly, the algorithm currently considered to be the state of art in model building, LTGA (Section 2.2.5.1), can only model overlap in a strictly nested hierarchy, yet can outperform EDAs that are able to model more complex overlapping structures (Sadowski et al., 2013).

2.2 Algorithms

2.2.1 The Schema Theorem and Building Block Hypothesis

After genetic algorithms were first adopted and established as an effective solving method on a variety of problems, attention turned to how they worked. A widely-held belief was that crossover-based variation was able to implicitly infer the schema structure of the problem from a population under selection, allowing fit sub-solutions to be re-combined. A seminal contribution in this regard was provided by Holland (1975) and

the *Schema Theorem*. The Schema Theorem shows mathematically that under certain conditions, the frequency of a schema in an evolving population that is reproducing through crossover, will grow in proportion to its fitness. Goldberg advanced this idea with the *Building Block Hypothesis* (BBH) (Goldberg, 1989b). The idea is that if evolution is able to identify and recombine fit low-order blocks together, then fitter high order blocks can be discovered, and this process can be repeated recursively. The implication is that by using crossover, hierarchical problem structure of many different orders can be automatically inferred and exploited.

There are three key limitations with the Schema Theorem and BBH. Firstly, the theoretical foundations for the Schema Theorem have been criticised, and it has not been proven over more than one generation (Beyer, 1997). Because of this, it is regarded as an ansatz rather than a proof for the general behaviour of crossover. Secondly, the assumptions used to prove the Schema Theorem cause difficulty for the idea of GAs as *unbiased* BBO algorithms. The Schema Theorem holds (for one generation) when the probability of each schema being disrupted is relatively low. This means that schemata need to be "short" and of low order, which explains the BBH definition of a building block. The problem with this is that a high-dimensional solution space has no concept of "short" or "long" – these are only valid on problem representations where a sequential order is imposed on those dimensions by the algorithm designer. This means that variation is no longer random, but a function of sequence and thus has an intrinsic positional bias. In order then for crossover to be effective, a sequential representation with "good linkage" has to be designed that essentially preloads the algorithm with assumptions about where fitness-effecting interactions are likely to be found (Goldberg, 1991). Positional bias can be removed from GAs by using a *uniform crossover* operator rather than one-point or two-point sequential crossover. Research has shown, however, that once positional bias is removed, GAs struggle to identify and recombine schema structure at multiple levels of scale (Thierens, 1995, 1999). It also destroys their ability to identify and recombine modular structure (Watson, 2002, 2006). The third limitation is that on some test problems with good linkage, crossover has been expected to do well but has been shown to perform poorly. Of particular note are its performance on deceptive functions (Goldberg, 1987; Pelikan, 2002), and the "Royal Roads" problem (Mitchell et al., 1993; Forrest & Mitchell, 1993), where there is evidence to suggest that it offers no advantages over simple-hill climbing methods with a stochastic element. These results show that even in functions with a biased sequential representation, the two basic things that crossover is required to do – promote and recombine the fittest schemata – may not occur in practice.

Despite the limitations of the theory, simple GAs remain an effective search method for some applications. However, in the context of unbiased BBO methods, other techniques are proving to be far more effective.

2.2.2 Compositional Evolution

Using his framework of *Compositional Evolution*, Watson (2006) shows that if evolutionary variation is guided correctly by a model, then the central intuition behind the Schema Theorem and BBH can indeed be realised. In other words, he shows how it is possible for evolutionary search to efficiently decompose and recombine problem structure without positional bias, and how it can efficiently solve problems that gradualist approaches to search, such as hill-climbing, cannot. Central to the idea is model guided variation: evolution itself evolving an effective variation neighbourhood. The concept is demonstrated practically using the SEAM algorithm on shuffled HIFF problems (Watson & Pollack, 2000), which crossover methods are unable to solve efficiently. This *symbiotic* algorithm performs a type of recursive clustering to identify relevant module configurations, then recombines these to create offspring.

The key contribution of the work is not the SEAM algorithm itself, which is only effective on very contrived problems, but a conceptual framework that interprets Simon's original ideas in the context of evolutionary systems. In particular, it highlights the potential power of an evolutionary search process that is able to infer an exploration distribution that reflects the constraints of the selective environment. The contributions of this thesis lend further support for this framework.

2.2.3 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) originated in the late 1990s (Mühlenbein & Paaß, 1996) and are also known as Probabilistic Model Building Genetic Algorithms (PMBGAs). These algorithms have been shown to successfully overcome the shortcomings of crossover based GAs on a variety of different problem classes. In this section we consider EDA variants for discrete problem spaces. Across this field many different approaches have been developed and the literature continues to grow. Here, we focus on summarising the key themes and cornerstone works that characterise the development of the field since its inception, and their capabilities with respect to inferring and exploiting schema structure. Additional information about each of the described methods can be found in the review papers of Pelikan et al. (2002); Hauschild & Pelikan (2011). All the works we consider are unbiased BBO techniques and do not assume problem representations contain an inherent sequential order.

In contrast to sexual and asexual reproduction, in which implicit model building is decentralised, EDAs build a single, centralised model of a larger subpopulation. The idea is to iteratively apply selective pressure to a population to identify a subpopulation of comparatively fit individuals. This subpopulation is often described as a collection of "promising solutions", and the subsequent model as a hypothesis for the regions occupied by promising solutions in search space. By sampling this probabilistic model, the

idea is that other (hopefully fitter) solutions can be found. The model typically contains two logical components: a decomposition of population structure, and a probability distribution over this decomposition. The ability of an EDA to model the schema structure of a problem can be evaluated by its ability to recall relevant schemata when the model is sampled. This is largely dependent on the problem decomposition over which a distribution is inferred. Early EDAs assume a decomposition a priori, whereas more modern methods are able to infer a decomposition.

Algorithm 1: The Canonical Estimation of Distribution Algorithm

Initialise a population of candidate solutions
while *objective not reached* **do**
1. Use selection to identify a subpopulation of fit individuals
2. Assume or infer a structural decomposition of fit subpopulation
3. Infer a probability distribution of fit subpopulation over structural decomposition
4. Generate offspring by randomly sampling probability distribution
5. Use survivor selection to integrate offspring back into population

end

The canonical pattern of an EDA is shown in algorithm 1. In step 4, variation is performed by randomly sampling the probability distribution to generate offspring, which are integrated back into the population using survivor selection. This type of variation is often referred to as *mixing*.

The first wave of EDAs modelled problem structure at the level of individual problem variables, and thus the probabilistic models they infer are over a univariate decomposition assumed a priori. Algorithms in this category include the Univariate Marginal Distribution Algorithm (UMDA) (Mühlenbein & Paaß, 1996) and two incremental EDA variants: Population-Based Incremental Learning (PBIL) (Baluja, 1994) and the Compact Genetic Algorithm (Harik et al., 1998). The results from these works show that they are often more effective than crossover based GAs. They also have an advantage in terms of the simplicity of the model: inferring a distribution has linear time complexity and the model requires very little memory (Sastry et al., 2007).

The results show that univariate EDAs are most effective on fully-separable problem structures that match the assumed univariate decomposition. On structures containing higher order epistasis, more sophisticated models have proven to be more effective. Amongst these are algorithms that infer a joint probability distribution over a bivariate decomposition. By linking bivariate structures together they are able to sample the model through a chain of conditional probabilities. For example:

$$p(X_0) = p(X_0|X_4)p(X_4|X_1)p(X_1|X_7)p(X_7)$$

The MIMIC algorithm (De Bonet et al., 1997) uses a single linked list to model this chain, Baluja & Davies (1997) uses a tree structure and the Bivariate Marginal Distribution Algorithm (BMDA) (Pelikan & Mühlenbein, 1999) creates independent clusters of trees. As the algorithms are able to model bivariate interaction, they enjoy superior performance to univariate algorithms on problems with such interactions – such as 2d spin-glass systems. However, on problems that contain interactions of higher orders, i.e those that contain separable modularity or anything more complex, they are unable to identify and recombine schemata efficiently (Bosman & Thierens, 1999).

The Extended Compact Genetic Algorithm (ECGA) (Harik, 1999) is able to model multivariate interactions by inferring a modular decomposition of the fit subpolutation (step 2), then inferring a joint probability distribution over this decomposition (step 3). The author refers to this approach as a *Marginal Product Model*. The decomposition is able to capture interdependency within disjoint modules, allowing it to reflect the structure of problems with separable modularity. The results show that ECGA can solve certain problems of this kind in low order polynomial time, which is beyond the capability of univariate and bivariate models. The modular decomposition of ECGA is inferred through a greedy agglomerative clustering process that is guided by the MDL principle, which ensures the right balance between the size of the joint probability tables and the accuracy of the model.

By following MDL, it could be argued that ECGA, by being the most compact model, is the *best model*. However, this is subject to two limitations that are very pertinent to this thesis. Firstly, given the uncomputability of KC, this argument only applies to the class of models that make the same assumptions. In other words, other linkage learning models that assume a structure of separable modularity. Secondly, it only applies to models of the *probability distribution of the population* and not the *population itself*. This creates an important contrast to the compact SG model introduced in Chapter 3, which models the population *directly*.

The Bayesian Optimisation Algorithm (BOA) (Pelikan et al., 1999) is a multivariate EDA that is also able to capture structure containing separable modularity. In this case the algorithm captures problem decomposition in a Bayesian Network, which models variable dependencies in a directed acyclic graph. On each iteration of the algorithm, a network structure is inferred using a greedy algorithm and a scoring metric that also appeals to MDL. It then infers joint probability tables over this structure. The results show that this algorithm can also solve separable problems in low order polynomial time. In principle, Bayesian Networks can model modular interdependence through variable to variable dependency relationships and joint probability tables of the correct size. This overcomes the key limitation with ECGA, which is only able to model separable modularity. However, the practical problem with this is scalability: the size of the model grows exponentially with the number of interdependencies between variables.

To address these challenges an improved version of BOA was created that uses decision trees to compress the joint probability tables. This is called the Hierarchical Bayesian Optimisation Algorithm (hBOA) (Pelikan, 2002). hBOA has proven itself more effective that alternative EDAs and GAs on problems containing hierarchical modular interdependence (ibid.). When combined with a local hill-climbing step, it has also proven more effective that other EDAs when solving problems containing irregular, overlapping structures (see next section). Until recently, hBOA was considered the state of the art in discrete EDAs and model building algorithms more generally.

Since the development of BOA and hBOA, a class of multivariate EDAs has been introduced that uses undirected Markov Networks rather than directed Bayesian Networks to model dependency structure (Shakya & Santana, 2012). They are sometimes known as Distribution Estimation Using Markov (DEUM) algorithms (Shakya, 2006). These approaches factorise the joint probability distribution using the cliques of the network. Although performance that is similar to other multivariate PMBGAs has been reported on some tests (Shakya & Santana, 2008), the techniques are yet to establish any distinct benefits over alternative methods.

2.2.4 Hybrid Methods

Both EDAs and crossover based GAs share two key limitations. The first limitation is that the variation distribution of both methods can make efficient navigation of local landscape topologies difficult. Both are able to recombine potentially useful features of many different orders, allowing them to cross fitness valleys that defeat gradualist approaches such as bit-flip hill climbing and mutation-only variation. However, the distributions are often biased against small search steps required to reach local peaks (including the global optimum) efficiently.

Recent research has shown that hybrid methods, which combine recombinative variation with bit-flip hill climbing, can offer an effective solution to this problem. These methods have been described as *Hybrid EAs* (Pelikan, 2010). The approach benefits from the exploratory power of recombination with a repair stage that navigates local topology. The technique has been successfully applied to EDAs, showing that hBOA can solve problems with irregular, overlapping structure in polynomial time (and more efficiently than some other hybrid EDA methods) (Pelikan & Goldberg, 2003; Pelikan, 2008; Pelikan et al., 2009; Pelikan, 2010).

The same approach has been used on GAs with uniform crossover, creating a new type of unbiased BBO algorithm usually referred to as a *Hybrid GA* (previous references plus Prugel-Bennett, 2007, 2010). The results show that Hybrid GAs will often outperform vanilla GAs. Interestingly, they have been shown to solve separable problems in polynomial time, despite not being able to infer an explicit model of modular structure.

Arguably, the performance of Hybrid GAs on problems with an irregular structure is competitive with (hybrid) hBOA (see results in Pelikan, 2008), but without the overhead of explicit modelling. Given the strength of these results, a Hybrid GA is used as an implicit modelling comparison method in Chapter 5.

2.2.5 Multi Scale Search

The second limitation with vanilla EDAs and GAs is that recombination does not benefit from incremental trial and error, which allows directed mutations (of any order) to be tested one at a time. Recently, new approaches to explicit model building have emerged that overcome this limitation with powerful effect. These approaches employ centralised model-building in an iterative loop, like EDAs. However, they do not employ probabilistic models. Instead, they infer a problem decomposition and use this as a variation neighbourhood in stochastic macro hill climbing. These approaches have been described in recent work as Multi-Scale Search (MSS) (Mills et al., 2014). This work uses test problems containing modular interdependence to demonstrate that once the modular decomposition of a problem has been learned, MSS can find the optimal combination of modules in provably linear time. In comparison, EDAs take exponential time to find the correct solution using random recombination. As well as these theoretical results, the experimental work described in this section shows that on many different problems, including problems with irregular, overlapping structures, MSS methods are outperforming both vanilla and hybrid EDAs and GAs.

The Building Block Hill Climber (BBHC) (Iclanzan & Dumitrescu, 2007) and MACRO-H (Mills, 2010; Mills et al., 2014) are MSS algorithms that decompose problems into a library of schemata using a similar agglomerative clustering method. This method joins together alleles that only ever co-occur together (they have maximal mutual information). By decomposing problems in this way and applying MSS principles, they show that building block problems can be solved in linear or sub quadratic time. This clustering method proves the MSS concept. However, given the absolute nature of the joining mechanism, it is only effective on synthetic building block problems that are free of noise. A variation of the idea, which seeks to overcome this limitation using probabilistic "soft joins", is described in Mills (2010).

More recently, another MSS algorithm has been introduced that arguably represents the state of the art in model building algorithms: the Linkage Tree Genetic Algorithm (LTGA) (Thierens, 2010). This algorithm adopts the same general MSS approach, but supports a richer problem decomposition that has proved to be highly effective on a variety of different problem structures.

2.2.5.1 The Linkage Tree Genetic Algorithm

LTGA and its variants are MSS algorithms that decompose problem structure into what is known as a *Family Of Subsets* (FOS) model (Thierens & Bosman, 2011). The idea is that the linkage structure of a problem can be represented using a collection of subsets of problem dimensions. Consider the following example, which illustrates a FOS for a 10-dimensional problem:

$$F = \{\{1, 3, 10, 2\}, \{10, 3, 1\}, \{3, 10\}, \{3, 1\}, \{1, 3, 4, 8\}, \{5\}\}$$

Each subset within an FOS is unordered and describes a partition of the search space in which multivariate interactions can be observed. In theory, subsets can be disjoint, can overlap in a strictly nested way (where one subset is a subset of another), or overlap in an unnested way (where one subset intersects with, but is not a subset of, another). The LTGA algorithm produces a class of FOS that can contain a subset of these relations, and can be interpreted as a tree.

The LTGA clustering process is illustrated in figure 2.1. A population is initialised using steepest-ascent hill climbing, which applies initial selection pressure. On each iteration, a linkage tree is inferred from the population, using bottom up agglomerative clustering and an information-theoretic distance metric. Clustering starts by populating a linkage tree with individual problem loci as leaf nodes. On each iteration of clustering, the algorithm measures the distance between leaf nodes in the tree. On the first iteration, there are $\Theta(n^2)$ such distance measures, describing the information distance between pairs of loci. The "closest" subsets are merged into a new subset which is then added to the tree as a descendent of the two original subsets. In the next iteration, the original subsets are removed from the distance calculation, and the new subset is used instead. This process continues until there is only one leaf node containing a subset of length n: a state that is guaranteed within $n - 1$ steps. The result is an FOS that contains $(2n - 1)$ partitions. The length n partition is discarded, as a linkage pattern describing the whole genome does not make sense. Usually, any common ancestors of a node with 0 information distance between them are also discarded, as they are redundant.

LTGA produces a class of FOS where the subsets can be interpreted as a binary tree. It is capable of modelling modular and hierarchically modular linkage structures that are strictly nested. However, given the nature of the way subsets are clustered, it cannot capture unnested overlapping structure.

In LTGA, MSS-style variation is performed by combining the linkage structure in the tree with allele values sampled from the population. In the original algorithm, two parents are randomly selected from the population: each acts as an initial condition for macro hill climbing. The linkage tree is then enumerated, usually in a smallest first

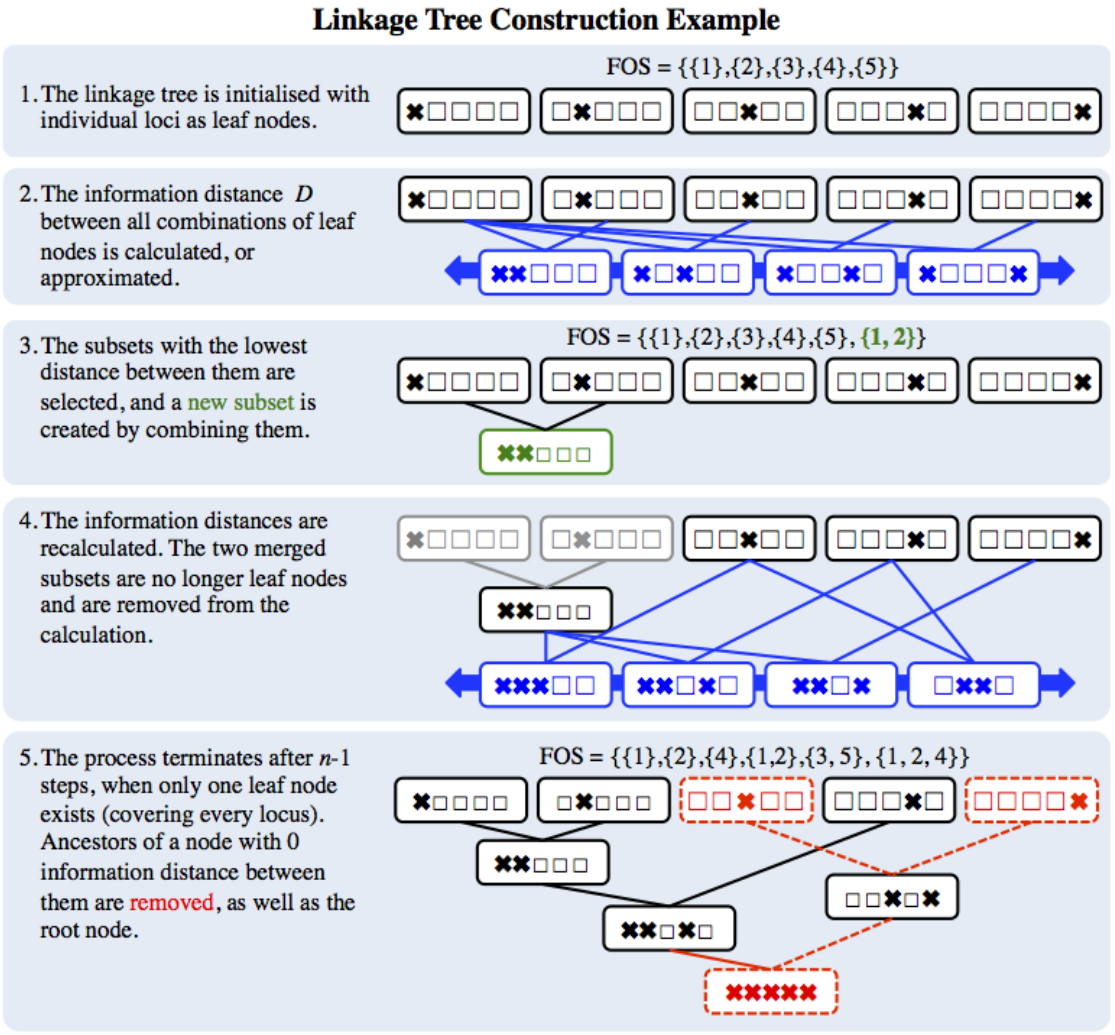

FIGURE 2.1: An illustrated example of linkage tree construction from LTGA (see text).

order (see Goldman & Tauritz, 2012). Each subset acts as a bit mask, which, when applied to each parent, produces two specific schema. The schemata are crossed over to each parent to create two mutant offspring. If either of the offspring are fitter than the fittest parent, then both parents are replaced by the offspring. This process continues until all the linkage patterns in the FOS have been enumerated. In later approaches, a Global Optimal Mixing EA (GOMEA) strategy is preferred to two-parent crossover (Thierens & Bosman, 2011; Goldman & Tauritz, 2012). In this strategy, a parent is chosen at random and cloned as an initial condition for macro hill climbing. For each enumerated subset, a random "donor" parent is selected from the population and a schema is created. This is used as a macro mutation operator which is tested on the offspring.

Two different information distance metrics have been used in the literature to create linkage trees through agglomerative clustering: the original "cluster distance" metric and a "pairwise distance" metric (see the referenced literature for specific details of

both methods). The cluster distance metric requires the joint entropy of each potential cluster to be calculated. As cluster entropy calculation scales exponentially with the size of the clusters, this can create a high computational burden. The pairwise distance metric (Pelikan et al., 2011; Thierens & Bosman, 2011) simplifies computation by using bivariate information measures only to guide the clustering process. It is essentially a heuristic approximation of the cluster distance. This is computationally very efficient – scaling reliably at $\Theta(n^2)$. Use of the pairwise metric means that the entire linkage tree is constructed using information calculated from bivariate interactions only. Perhaps surprisingly, aside from the saving in computation, the pairwise metric has proven to be more effective in overall optimisation performance than the original, reducing required population sizes and fitness evaluations (Pelikan et al., 2011; Thierens & Bosman, 2011; Goldman & Tauritz, 2012).

MSS using LTGA appears to offer a number of advantages over EDAs. In terms of modelling complexity, the linkage tree can be reliably built in $\Theta(n^2 p)$ (Pelikan et al., 2011), where n is the problem size and p is the population size. This is considerably faster than the $O(n^3 + n^2 p)$ offered by hBOA (Pelikan et al., 2000). LTGA has also been shown to have superior performance to leading EDA methods on a variety of problems. This includes ECGA on concatenated trap and nearest-neighbour NK-landscape problems (Thierens & Bosman, 2011), and hBOA on MAX-SAT problems (Sadowski et al., 2013). It has also been shown to perform efficiently on more difficult NK-landscape problems containing neutrality (Goldman & Tauritz, 2012), MAXCUT and hierarchical trap problems (Thierens & Bosman, 2012). One interesting result is that although LTGA cannot model unnested overlap, it can outperform EDAs that *can* model this structure (Sadowski et al., 2013). This is reflected upon further in Section 5.3.

Overall, we regard MSS using LTGA as the state of the art in explicit model building methods. As such, it is used as a comparative modelling method in Chapter 5.

2.3 Summary

In this chapter we have explored the nature of evolutionary problems, and shown how they can be described in terms of hierarchical schema structure. We have reviewed the field of evolutionary model building algorithms, and examined the ability of each algorithm to infer and exploit schema structures of various kinds. Two methods have been identified as the state of the art in implicit and explicit model building respectively: Hybrid GAs and LTGA. Both of these methods are capable of solving problems containing various types of complex, high-order schema structure in polynomial time. They will be used as comparison methods in Chapter 5.

The thesis now moves on to describe the Schema Grammar model, its ability to infer the explicit schema structure of evolutionary problems, and the advantages it offers over the current state of the art.

Chapter 3

Schema Grammar: A Compact Model of Problem Structure

3.1 Introduction

The research hypothesis being tested is that compact models of fit populations can reflect the compositional structure of the selective environment and have strong predictive qualities with respect to fitness. This hypothesis appeals directly to the MDL principle, which claims that the "best" explanation of a data set is the *most compact* representation of it. We follow the intuitions of the Schema Theorem, BBH and Compositional Evolution, and assume that selective environments can be decomposed into a hierarchy of semi-independent parts, which can be represented by schema structure. In the context, the challenge is in inferring a compact model that can capture schema structure without positional bias.

Typical lossless compression techniques are inadequate as their capabilities are restricted to compressing sequential structure only. Although it is common in evolutionary algorithms to represent individuals as sequences over a finite alphabet (for example, the "bit string" representation), unbiased algorithms must be able to treat the order of this representation as arbitrary (Rowe & Vose, 2011). In other words, interactions between variables will not necessarily be visible in contiguous loci, meaning problem structure is not reflected in sequential structure. Sequential compression is also limited by its inability to infer information about absolute position. In the problems we consider here, the functional context of alleles is key: it is the combination of alleles and loci that specifies a genome's semantics. However, sequential compression will only detect relative patterns and inspecting these patterns will often reveal little about the underlying structure of the problem. For both these reasons, a new approach is required that can be described as *combinatorial compression*.

In this chapter a new type of grammar-based compression is introduced that is capable of compressing a population of sample individuals by inferring a hierarchical structure from which it can be composed. We refer to it here and throughout as *Schema Grammar* (SG). SG is a lossless model, meaning that the original population can be recreated without any loss of information. SG achieves compression by identifying frequently recurring combinatorial patterns – schemata – in a population of phenotypes, capturing these in a compact G-P map, then encoding individuals in a factorised genetic representation. The SG model is inferred from a population directly using grammatical inference techniques, which are detailed in Section 3.4. This section defines an offline algorithm that adopts features of two existing sequential lossless compression algorithms, modifying them to remove positional bias.

Generative grammar is able to model a compact genetic representation using symbol substitution and recursive rewriting. Each of the symbols inferred into the grammar expands into a schema or complete phenotype by recursively rewriting a hierarchical composition of smaller schemata. There is a topological equivalence between this and the description of hierarchical schema structure introduced in Chapter 2. The power of the model is three fold. Firstly, it allows a schema structure for the selective environment to be explicitly inferred and inspected, which is illustrated in Chapter 4. Secondly, it allows model-based evolutionary search to generalise about the selective environment in terms of explicit schemata. This is demonstrated in Chapter 5. Finally, as a lossless compact model it offers a principled way of capturing the information content of the population, whilst respecting complex epistatic interdependency and decomposable problem structure. In Chapter 6, this property is used to create new quantitive measures of evolutionary information that have a strong predictive quality with respect to fitness.

3.1.1 Kolmogorov Complexity

Kolmogorov Complexity (KC), otherwise known as *algorithmic complexity*, describes the complexity of an object as the the shortest Universal Turing Machine that can generate it. KC measures the *absolute* information contained by an object. This contrasts with Shannon information, which is a probabilistic measure describing the *average* information transmitted by a random source (Grunwald & Vitanyi, 2004). KC provides the theoretical cornerstone for compression, Bayesian inference and the MDL principle (for full details see Vitányi & Li, 2000; Li & Vitányi, 2009). Essentially, it shows that the size of a lossless, compact model of an object is an estimate of its absolute information content. If the object is highly structured then it will contain redundancy, will be highly compressible and will thus have "low complexity". If an object contains no recurring structure then it will not be compressible, thus having "high complexity".

If we are to describe the sophistication of life in terms of its complexity (Bedau, 2009), then the Kolmogorov interpretation is misrepresentative. After all, it would consider a

randomly disordered object to be more "complex" than one which contains the regularity of form that characterises life. However, by applying careful interpretation the theory *can* be used to derive measures of evolutionary information that are genuinely revealing. This is demonstrated by the work in Chapter 6, which shows how the compact SG model can be used to approximate the amount of "knowledge" contained in a population about its selective environment, even if the structure of that environment contains complex epistatic interaction.

One of the key results of KC theory is that it is computationally undecidable. Essentially, this means that measuring the information content of an object is subject to how it is represented, and there can be no such thing as the *best* compact model across all possible representations. An implication of this is that if compressed size is used as an estimate of information content, it is only valid with respect to a particular representation. It also means that SG and sequential compression methods, which employ fundamentally different representations, are not *directly* comparable. However, they can be used to construct relative measures which *are* comparable, and this is the approach taken in Chapter 6.

The examples contained in this chapter are primarily intended to illustrate the symbology and function of SG and SG inference. The compression achieving by using symbol rewriting in SG is pointed out in some examples, but it is left to the work of later chapters to demonstrate the degree of compression that SG is capable of and how this relates to population structure, problem structure and KC.

3.1.2 Grammatical Evolution

It is important to draw a distinction between the techniques introduced in this thesis and other applications of generative grammar in evolutionary computation, in particular *Grammatical Evolution* (GE). GE was first introduced by Ryan et al. (1998) as a way of evolving solutions to Genetic Programming (GP) problems. Many applications and variants have since been described in the literature (see Mckay et al., 2010, for a review). The canonical approach of GE is to *predefine* a library of production rules that act as a G-P map, then evolve genotypes that are defined in terms of this map. A variety of grammars have been used for this purpose, including sequential grammars and tree grammars (ibid.). The rules of the grammar constrain phenotypes to be composed from a range of prevalidated forms, so evolutionary exploration can proceed whilst respecting these forms. In this respect GE follows the intuition of SG exactly: variation is best guided by modelling the inherent constraints of phenotype space. The key difference between GE and SG is that GE grammars are usually predetermined by the algorithm designer or evolved (O'Neill & Ryan, 2004), whereas in SG they are inferred directly from a fit population.

Although GE is usually applied to GP problems, some variations show that the technique can be used to evolve solutions to GA problems. The GAUGE algorithm (Ryan et al., 2002) uses grammar to map GP genotypes, which are defined using a sequence of integers, into a state machine that generates GA phenotypes. The model is order-independent (in phenotype space) and ensures that phenotypes are not over-specified. The authors show that the GAUGE compares favourably with Messy GAs (Goldberg et al., 1989) on some very simple problems, thus demonstrating the concept. Some success has also been reported applying GAUGE to Sudoku puzzles (Nicolau & Ryan, 2006). Given that the production rules in GAUGE do not capture any constraints of phenotype space except its size, it is hard to envisage the general advantage it may offer over any other unbiased implicit model such as uniform crossover. The meta-Grammar Genetic Algorithm (mGGA) (O'Neill & Brabazon, 2005) uses sequential grammar and a wrapping operator to map GP genotypes onto GA phenotypes. The production rules can be predefined or evolved to capture the constraints of phenotype space, although as a sequential grammar these can only be expressed in terms of alleles that are independent of functional context. Unlike GAUGE, the model is not order-independent. The authors use rules pre-defined with module configurations to evolve solutions to building block problems, although they do not offer any comparisons with non-GE methods. Given that neither of these methods constitute an unbiased model building EA nor been established as an EA performance benchmark, they will not be considered further here.

3.1.3 Related Methods in Machine Learning

Some techniques exist in machine learning that share characteristics with the combinatorial compression model implemented by SG. First amongst these are *feature extraction* techniques that attempt to reduce the dimensionality of data by finding and exploiting redundancy. The best known of these is Principal Component Analysis (PCA), in which multidimensional data spaces are remapped with a new set of dimensions (components) that are oriented such that the lowest order components explain the most variance in the data and the highest order components the least. The distribution of variance over the components can reveal information about multivariate structure in the original data. Indeed, by using PCA, multivariate correlation structure has been shown to exist in evolutionary problems (Tayarani-N & Prugel-Bennett, 2014). PCA shares the concept of *factorising* the original data with lossless compression. In compression grammars, the information structure is captured in the production rules of the grammar; in PCA the information structure is captured in the components. As well as the analytical capabilities it provides, PCA decomposition can actually be used for compression directly. By discarding components which explain the least variance, the space can be concatenated, although this is a lossy rather than lossless process. It would be an interesting piece of further work to look at the relation between SG and PCA more closely.

A related type of unsupervised learning method is *subspace clustering* (see Parsons et al., 2004, for a review). The primary objective of this type of algorithm is to categorise multi-dimensional data samples into discrete groups (ibid., Section 3.2), which can be either disjoint, overlapping or hierarchical. Clustering can be used for analysing high-dimensional order in data (as with PCA) and it can also be used to automatically label data as a precursor to supervised learning, a technique that is known as *semi-supervised learning* (Chapelle et al., 2006). The core model implemented by both SG and LTGA can be regarded as a type of subspace clustering.

Another related area is *association mining*, in particular the APRIORI algorithm (Agrawal & Srikant, 1994), and *biclustering* methods in bioinformatics (Madeira & Oliveira, 2004). These methods use *pointwise* frequencies of association to identify interactions in combinatorial data, and capture higher-order interactions by merging lower level interactions together. Pointwise relates to micro state interactions as opposed to variable interactions. For example, if a data set contained two columns named "Species" and "Status", then association mining might detect an interaction between "Panda" and "Endangered" in a subset of rows. In an evolutionary context, this is the difference between inferring schemata and linkage structure. In bi-clustering applications association mining is used to find interactions in gene expression. SG is similar in many ways to association mining methods. A common feature is that they both support the inference of overlapping, pointwise interactions of many orders. Additionally, both use pointwise frequency of association to detect information in the data set.

3.2 Context Free Grammar

In this section some relevant concepts and terminology from the field of context free grammar (CFG) will be briefly introduced. For more detailed information, a standard text on formal grammars should be consulted (e.g., Webber, 2008). The concepts in this section will be described in the context of sequential languages. In the next section the set-based symbolic representation and production rules that distinguish SG from conventional CFGs will be defined.

Formal grammars are models of how *languages* can be generated. Each language is a sequence of symbols that follows a particular structure. Each symbol in a language is drawn from a finite alphabet, each element of which is known as a *terminal symbol*. The English language for example uses an alphabet of 26 characters as well as digits and punctuation marks, and each of these is a terminal symbol. The fundamental precept of formal grammar is that language can be derived from a higher-level symbolic representation. This higher-level representation makes use of variables called *non-terminal symbols* to represent concepts or features, and *production rules* which define how the concepts interrelate with each other and produce the language. Each rule associates a

non-terminal symbol with a sequence of symbols with which it can be replaced, which can include both terminal and non-terminal symbols. Any given extract of language can be *derived* from a starting sequence of symbols by recursively applying production rules associated with those symbols. Derivation can thus be explained as the *recursive expansion* of symbols into language. The non-terminal symbols and the production rules of a grammar, model the structure of the language.

As an example, consider sequences of DNA drawn from the four nucleobases $\{G, A, T, C\}$, each of which we model as a terminal symbol. A sequence $GATCGATCATC$ can be derived from a grammar with the following production rules:

$$\begin{aligned} s &\rightarrow x_1 x_1 x_2 \\ x_1 &\rightarrow G x_2 \\ x_2 &\rightarrow ATC \end{aligned}$$

In this grammar, the non-terminal symbols are given by the set $\{s, x_1, x_2\}$. The symbol s, known as the *start symbol*, is a special case and is the root of the derivation. Each of the production rules relates a non-terminal symbol with a sequence of symbols it should be substituted with. Starting with s the sequence D can be recreated without loss of information through recursive rewriting in the following sequence of steps:

1. s
2. $x_1 x_1 x_2$
3. $G x_2 G x_2 ATC$
4. $GATCGATCATC$

This simple grammar provides an early intuition into the idea of using grammar for compression: there are 11 symbols in the original sequence and 8 symbols on the right-hand side of the grammar's production rules. The non-terminal symbols represent a hierarchy of recurring patterns in the sequence that are used to compress the representation. At the bottom level x_2 expands to ATC and x_1 extends that sequence to $GATC$. Also note that in the production rules, no two symbols co-occur together more than once. This suggests that the sequence has been completely factorised by the grammar – in other words that the information in the sequence has been captured by the production rules and it cannot be further compressed.

The CFGs produced by most lossless compression algorithms are known as *straight-line grammars* (SLGs). SLGs are more constrained than regular CFGs: every non-terminal symbol can only appear on the left-hand side of one production rule and derivation cannot loop. For example, the rule $x_1 \rightarrow x_1 x_2$ would recurse infinitely and therefore cannot appear in an SLG. As a result of this constraint derivation is finite and deterministic, which is of course necessary for lossless compression. The grammar in the example above

is an SLG. Kieffer & Yang (2000) refer to SLGs as *admissible grammars* and provide formal definitions.

3.3 Schema Grammar

SG is a type of SLG that enables inherently combinatorial data sets to be modelled grammatically. It introduces two key modifications to the CFG formalism:

1. **Language is described using *unordered sets* of symbols, as opposed to sequences**. Using this modification, both individuals and schemata can be described using an unordered set of traits.
2. **Production rules define *set merge* operations rather than *string rewrite* operations**. This modification allows individuals to be described as a recursive composition of symbols, each representing a schema. This creates a model that can explicitly represent hierarchical schema structure.

3.3.1 The Language of Problems

In SG candidate solutions are represented by an unordered collection of terminal symbols. Each terminal symbol is used to represent a discrete parameter of the search space. Let the set of terminal symbols of an SG instance be given by the finite set:

$$\Sigma = \{\alpha_1, \ldots, \alpha_j\} \tag{3.1}$$

Let the language of an SG instance be given by the powerset:

$$\mathcal{L} = \mathcal{P}(\Sigma) \tag{3.2}$$

Individual objects are represented using a sublanguage of $\mathcal{L}$: an unordered subset of terminal symbols. SG can represent and compress any population of objects that can be expressed in this way. Here, we introduce a symbolic representation for problem spaces of fixed dimensionality over a binary alphabet that are canonical in GAs. We use this representation throughout the thesis to demonstrate problem structure being inferred and exploited. In Section 7.2.1, two examples of alternative representations are presented.

Each terminal symbol is used to represent a specific combination of locus and allele, the notation for which we abbreviate to x_i^j, where i is the locus and j is the allele. For GA

problems let the terminal symbols of SG be given by:

$$\Sigma = \{x_i^j : i \in \{1, 2, \ldots, n\}, j \in \{0, 1\}\} \tag{3.3}$$

This is simply the set of all possible traits in a problem space. As an example of the representation consider a 4-dimensional GA problem. The terminal symbols are given as:

$$\Sigma = \{x_1^0, x_1^1, x_2^0, x_2^1, x_3^0, x_3^1, x_4^0, x_4^1\} \tag{3.4}$$

Using this symbology we can reinterpret the bit string individual 1100 as the set:

$$p = \{x_1^1, x_2^1, x_3^0, x_4^0\}$$

As p is order-independent is could equivalently be written as:

$$p = \{x_3^0, x_1^1, x_4^0, x_2^1\}$$

Schemata can be expressed in the same way as individuals using combinations of terminal symbols. For example, an order-3 schema on a 5-dimensional problem 0**01 can be expressed using the set:

$$s = \{x_1^0, x_4^0, x_5^1\}$$

This representation is similar to the representations used in so-called *moving locus* schemes including *messy coding* (Goldberg et al., 1989). In these representations, genes are expressed as explicit combinations of locus and allele and have no intrinsic sequential order. For example, in messy coding, genotypes and schemata are specified using variable length strings in ((locus allele)...(locus allele)) format like follows:

$$((1\ 0)(5\ 1)(4\ 0))$$

Which would specify the same schema as the previous example.

3.3.2 Set Merge Production Rules

In SG the operation of production rules is redefined to perform recursive set merging instead of string rewriting. This allows both schemata and phenotypes to be derived from a hierarchical, order independent, composition of traits. We note that set merge has been suggested by the *Minimalist Programme* of linguistics as "an indispensable operation of a recursive system ... which takes two syntactic objects A and B and forms the new object $G = \{A, B\}$" (Chomsky, 1999, p.2). This suggests the fundamental importance of merge operations in the general study of grammar and linguistics.

Let the non-terminal symbols of a SG be given by the finite set V, which includes one or more start symbols. Let each of the production rules in SG be of the form:

$$v \to p, v \in V, p \in \mathcal{P}(V \cup \Sigma) \tag{3.5}$$

Each v is a non-terminal symbol representing a specific combination of traits. The production symbol $\to$ specifies that v should be be replaced by merging in the contents of the RHS of the rule. $\mathcal{P}(V \cup \Sigma)$ is the powerset (set of all subsets) of the terminal and non-terminal symbols. This is best served by an example. Consider a grammar containing three production rules of the form:

$$\begin{aligned} g_1 &\to \{s_1\} \\ s_1 &\to \{x_6^0, x_4^0, x_3^0, s_2\} \\ s_2 &\to \{x_7^1, x_5^1, x_2^1, x_1^1\} \end{aligned}$$

In this example s_1 and s_2 are non-terminal symbols, and g_1 is a start symbol representing an individual. During expansion of g_1 each of the non-terminal symbols that appear are recursively replaced by merging in the contents of the rules associated with those non-terminal symbols. The steps in the resulting expansion are as follows:

1. $\{g_1\}$
2. $\{s_1\}$
3. $\{x_6^0, x_4^0, x_3^0, s_2\}$
4. $\{x_6^0, x_4^0, x_3^0, x_7^1, x_5^1, x_2^1, x_1^1\}$

The expansion of g_1 thus giving the phenotype 1100101. The mathematics of sets requires that unique symbols do not occur more than once. This removes the possibility of over-specification through repeating symbols, although it does not prevent that mutually exclusive alleles appearing together in a set. In other words, the representation does not formally disallow the production rule:

$$s \to \{x_0^0, x_0^1\}$$

If grammar instances are inferred using the algorithm defined in Section 3.4, mutually exclusive alleles cannot be inferred in the derivation of the population (assuming they do not appear together in the population). If symbols inferred into the grammar are used to facilitate variation (e.g., by using the methods described in Chapter 5), then it is up to the variation mechanism to ensure that mutual exclusivity is upheld.

3.3.3 Schema Grammar as a G-P Map

We can now formalise the concept of using SG as a genotype-phenotype map. In what follows, the notation $v \overset{*}{\Rightarrow} X$ is used to specify an *equivalence closure* relation, where $v \in V$ and $X \in \mathcal{P}(\Sigma)$. This simply means that the non-terminal symbol v *yields* (through repeated application of the production rules to conclusion) X, a schema or phenotype. So from the example in the previous section $g_1 \overset{*}{\Rightarrow} \{x_6^0, x_4^0, x_3^0, x_7^1, x_5^1, x_2^1, x_1^1\}$, and $s_2 \overset{*}{\Rightarrow} \{x_7^1, x_5^1, x_2^1, x_1^1\}$. By using equivalence closure we can say that non-terminal symbols represent individual genotypes and schemata. In what follows we also use notation $v \overset{+}{\Rightarrow} \beta$ to specify a transitive closure relation. This means that v can lead to the production of β. Using the same example, we can say that $g_1 \overset{+}{\Rightarrow} s_2$ is true.

First we formalise the properties of SG as an SLG. Let each non-terminal symbol in V appear once and only once as the LHS of a rule in R, and let there be no cycles in production:

$$\forall v \in V : |\{\alpha \to \beta \in R : \alpha = v\}| = 1 \tag{3.6}$$

$$\nexists v \in V : v \overset{+}{\Rightarrow} v \tag{3.7}$$

These properties mean that each non-terminal symbol deterministically yields a finite combination of terminal symbols. This allows the grammar to be used as a surjective mapping into problem space. This does not necessarily mean that the grammar is unambiguous (in the formal grammar sense): it is possible for different derivations to yield the same phenotype pattern. Whether or not this is the case depends on how the grammar is inferred. The grammars inferred by the algorithms defined in this chapter (Section 3.4) *are* unambiguous, but this might not be the case with other inference algorithms.

Let the non-terminal symbols of the grammar V contain two finite unordered sets representing genotype symbols and schema symbols respectively:

$$V = \{G, S\} \tag{3.8}$$

Where $G = \{g_1 \dots g_p\}$ is non-empty and each element of G is a start symbol. $S = \{s_1 \dots s_l\}$ may be empty. Now we can explicitly define the relationship between an instance of the grammar and a population of phenotypes. Let the *population mapping* of a grammar instance $\mathbb{G}$ be defined through the equivalence closure:

$$\{\{g_1\}, \dots, \{g_p\}\} \overset{*}{\Rightarrow} \mathbb{P} \tag{3.9}$$

Or equivalently:

$$\mathbb{G} \overset{*}{\Rightarrow} \mathbb{P} \tag{3.10}$$

Where each element of $\mathbb{P}$ is a sub-language of $\mathcal{L}(\Sigma)$ (Equation 3.2).

This means that a population of phenotypes of size p can be represented using a SG instance with p genotype symbols. Consider the following grammar for a population of two phenotypes:

$$P_1 = \{x_1^1, x_2^1, x_3^1, x_4^0, x_5^0, x_6^0, x_7^1\} \text{ (1110001)}$$
$$P_2 = \{x_1^1, x_2^1, x_3^0, x_4^1, x_5^1, x_6^0, x_7^1\} \text{ (1101101)}$$

This population can be represented compactly using the following instance of SG. The recursive expansions of each non-terminal symbol are shown in the right-hand column in binary schema format:

Production rule	$\overset{*}{\Rightarrow}$
$g_1 \rightarrow \{s_1, x_3^1, x_4^0, x_5^0\}$	1110001
$g_2 \rightarrow \{s_1, x_3^0, x_4^1, x_5^1\}$	1101101
$s_1 \rightarrow \{x_1^1, x_2^1, x_6^0, x_7^1\}$	11***01

Where g_1 and g_2 are the genotype symbols and s_1 is a schema symbol. The population can be recreated without loss of information by recursively expanding each of the genotype symbols into its own set:

$$\{\{g_1\}, \ldots, \{g_p\}\} \overset{*}{\Rightarrow} \{\{P_1\}, \{P_2\}\}$$

Even in this simple example, containing a population of just two samples, the compactness achieved by the representation can be observed. In the original population 14 symbols occurrences are used in the RHS of the grammar's rewrite rules, whereas in the compact version only 12 symbols are used. In larger populations containing significant amounts of predictable structure, this ability to factorise frequently-recurring patterns into schema symbols and reduce the overall size of the representation is significantly amplified.

3.3.4 Schema Grammar Defined

We can now formally define SG using the following tuple:

$$\mathbb{G}_{Schema} = (V, G, S, \Sigma, R), \text{ where:} \tag{3.11}$$

$V = \{G, S\}$	is a set of non-terminal symbols (variables)
G	is a finite non-empty set of genotype starting symbols that yield phenotypes
S	is a finite set of non-terminal symbols that yield schemata, which may be empty
Σ	is a finite non-empty set of terminal symbols representing order-1 hyperplanes
R	is a finite set of production rules, $R : V \to \mathcal{P}(V \cup \Sigma)$

3.3.5 Hasse Diagrams

Up to now we have illustrated the structure of SG in terms of production rules. However, a richer, more intuitive visualisation is possible using Hasse diagrams. These diagrams are used to represent the topology of a set. In our case the elements of the set are the grammar symbols $(\Sigma \cup V)$, and the topology we wish to represent is the relations between symbols, expressed in terms of hierarchical schema production.

Formally, Hasse diagrams depict the transitive reduction of a *partially ordered set* (poset) (see Simovici & Djeraba, 2014, for more complete details). Strict partial order is specified by a binary relation R. In our case we use $x \overset{+}{\Rightarrow} y$ as the relation – this is true if y is produced through the recursive expansion of x. Any two symbols x and y are said to be comparable if $x \overset{+}{\Rightarrow} y \oplus y \overset{+}{\Rightarrow} x$. Not all symbols are comparable (i.e., neither produces the other), hence *partial* order. This partial order has the following properties, which are satisfied as the grammar is an SLG:

Irreflexive $\nexists x \in (\Sigma \cup V) : x \overset{+}{\Rightarrow} x$

Antisymmetric $\nexists (x, y) \in (\Sigma \cup V) : x \overset{+}{\Rightarrow} y \wedge y \overset{+}{\Rightarrow} x$

Transitive if $x \overset{+}{\Rightarrow} y$ and $y \overset{+}{\Rightarrow} z$ then $x \overset{+}{\Rightarrow} z$

Each SG Hasse diagram is a directed acyclic graph. Symbols are represented as nodes and order as edges. We label symbols using their recursive expansion, which illustrates their meaning in phenotype space. We describe the result as the hierarchical schema structure of a grammar instance. As an example consider the following SG instance from a GA problem.

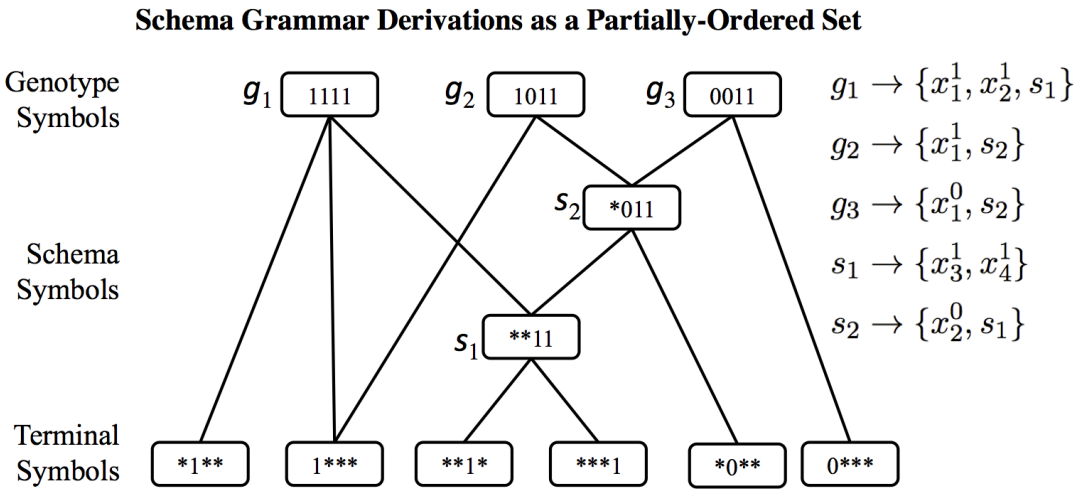

FIGURE 3.1: A Hasse diagram showing the schema structure of an example SG instance with three genotype symbols and two schema symbols.

Production rule	$\overset{*}{\Rightarrow}$
$g_1 \to \{x_1^1, x_2^1, s_1\}$	1111
$g_2 \to \{x_1^1, s_2\}$	1011
$g_3 \to \{x_1^0, s_2\}$	0011
$s_1 \to \{x_3^1, x_4^1\}$	**11
$s_2 \to \{x_2^0, s_1\}$	*011

A Hasse diagram for the schema structure of this grammar instance is shown in Figure 3.1. This diagram illustrates the topology of the grammar and the derivation of each genotype inside it. Posets and Hasse diagrams will be used in Chapter 4 to decompose evolutionary problem structure and compare it with the structure of SG.

3.3.6 Representing Canonical Forms

In this section we show by example how the canonical forms of schema structure described in Section 2.1.5 can be represented in instances of SG. Each of the examples is assumed to have been inferred from a population of above-average fitness individuals and for clarity only the schema symbols are shown. In each case the inferred structure happens to be sequentially contiguous: this is for the purposes of illustration. The examples serve to illustrate the representational capabilities of SG, rather than describe a typical problem instance.

3.3.6.1 Separable Modularity

In this instance schema symbols capture the module configurations from which the global optimum 1111 is composed:

Production rule	$\overset{*}{\Rightarrow}$
$s_1 \rightarrow \{x_1^1, x_2^1\}$	11**
$s_2 \rightarrow \{x_3^1, x_4^1\}$	**11

3.3.6.2 Modular Interdependence

In this instance schema symbols capture each of the module configurations from which the fittest candidate solutions are composed:

Production rule	$\overset{*}{\Rightarrow}$
$s_1 \rightarrow \{x_1^1, x_2^1\}$	11**
$s_2 \rightarrow \{x_3^1, x_4^1\}$	**11
$s_3 \rightarrow \{x_1^0, x_2^0\}$	00**
$s_4 \rightarrow \{x_3^0, x_4^0\}$	**00

3.3.6.3 Hierarchical Modular Interdependence

In this instance schema symbols capture the hierarchy of interdependent module configurations from which the the fittest candidate solutions are composed (only part of the hierarchy is shown for brevity):

Production rule	$\overset{*}{\Rightarrow}$
$s_1 \rightarrow \{x_1^1, x_2^1\}$	11******
$s_2 \rightarrow \{x_3^1, x_4^1\}$	**11****
$s_3 \rightarrow \{x_1^0, x_2^0\}$	00******
$s_4 \rightarrow \{x_3^0, x_4^0\}$	**00****
$s_5 \rightarrow \{s_1, s_2\}$	1111****
$s_6 \rightarrow \{s_3, s_4\}$	0000****

3.3.6.4 Overlap

In this instance schema symbols capture specific combinations of alleles that overlap in linkage space:

Production rule	$\overset{*}{\Rightarrow}$
$s_1 \rightarrow \{x_1^1, x_2^1\}$	11**
$s_2 \rightarrow \{x_2^1, x_3^1\}$	*11*
$s_3 \rightarrow \{x_3^1, x_4^1\}$	**11
$s_4 \rightarrow \{x_1^0, x_3^0\}$	0*0*
$s_5 \rightarrow \{s_2, x_4^0\}$	*110

3.4 Grammatical Inference

In this section an offline algorithm is defined that is capable of inferring instances of SG from populations of phenotypes. The time complexity of the algorithm is shown to be $O(n^2p)$, where n is the problem dimensionality and p is the size of the population. The algorithm starts by initialising the grammar with a set of genotype symbols G that directly encode the phenotype population in an uncompressed G-P mapping. As the algorithm progresses, the genotypes are factorised by creating a hierarchy of schema symbols that encode frequently recurring hyperplanes. It is important to note that the algorithm is not guaranteed to find the most compact possible model of a population. A limitation with grammatical models in general is that given an object and a way of describing it, finding the shortest description of it is NP-complete (Charikar et al., 2005). This is known as the *smallest grammar problem*.

The offline algorithm is adapted from two existing sequential compression algorithms that use grammar codes: RE-PAIR (recursive pairing), an offline algorithm from Larsson & Moffat (2000), and SEQUITUR, an online algorithm from Nevill-Manning & Witten (1997). Both algorithms compress a sequence of characters by identifying co-occurring symbols (digrams) and substituting these with non-terminal symbols. This process repeats recursively to build a hierarchy of symbols that expand into sequential motifs of various orders. In both cases, the resulting grammar has an important property: no two symbols co-occur anywhere in the grammar's production rules more than once. Essentially, the compression process identifies recurring sequential structure present in the original string of symbols and refactors it into a hierarchical symbol table, resulting in a compressed version of the original with redundant structure removed.

In the algorithm the general offline dictionary-based approach of RE-PAIR is adopted together with the rule utility constraints of SEQUITUR that collapse unnecessarily deep hierarchies of order-two production rules into larger, flatter productions. The key conceptual adaptation we introduce is inference of production rules that yield unordered sets rather than ordered sequences. This turns out to be a simple adaptation that results in grammars with analogous properties to those inferred by sequential compression. To proceed, we adopt an alternative definition of co-occurrence. In the following *sequential* CFG production rule:

$$s \rightarrow ababc$$

we would consider the co-occurrences present to be all the digrams present in the right-hand-side of the rule, i.e.:

$$\{(a,b),(b,a),(a,b),(b,c)\}$$

In contrast we define the co-occurrences in a SG production as all the order-2 subsets present in the right-hand-side of the rule. So for example the SG production rule:

$$g_1 \rightarrow \{x_1^1, x_2^0, x_3^0\}$$

contains the following order-2 subsets:

$$\{\{x_1^1, x_2^0\}, \{x_1^1, x_3^0\}, \{x_2^0, x_3^0\}\}$$

Let the co-occurrences in a production rule $\alpha \rightarrow \beta \in R$ be the set of order-2 subsets produced by the function:

$$\text{Co-occur} : \beta \rightarrow \{A \in \mathcal{P}(\beta) : |A| = 2\} \tag{3.12}$$

Where $\mathcal{P}(\beta)$ is the powerset of the RHS of the rule and $|\text{Co-occur}(\beta)| = \binom{|\beta|}{2}$. The offline algorithm works by counting all co-occurrences in a grammar's production rules to identify the most frequent, which is then substituted. This requires the following operations to be defined. Let the co-occurrences in a grammar be the multiset union of order-2 subsets across all production rules:

$$\mathbb{C} = \biguplus_{\alpha \rightarrow \beta \in R} \text{Co-occur}(\beta) \tag{3.13}$$

If we now introduce two production rules to the example:

$$g_1 \rightarrow \{x_1^1, x_2^0, x_3^0\}$$
$$g_2 \rightarrow \{x_1^1, x_2^1, x_3^0\}$$

Then:

$$\mathbb{C} = \{\{x_1^1, x_2^0\}, \{x_1^1, x_3^0\}, \{x_2^0, x_3^0\}, \{x_1^1, x_2^1\}, \{x_1^1, x_3^0\}, \{x_2^1, x_3^0\}\}$$

Let function F return the most frequent co-occurrence in a multiset X, or an empty set if no two symbols co-occur more than once:

$$m : X \rightarrow \underset{A \in X}{\text{argmax}}\, f(A) \tag{3.14}$$

$$\text{F} : X \rightarrow \begin{cases} m(X) & : \text{if } \ |m(X)| \geq 2 \\ \emptyset & : \text{otherwise} \end{cases} \tag{3.15}$$

Where $f(A)$ is the multiplicity function counting the instances of A. Using the same example then $\text{F}(\mathbb{C}) = \{x_1^1, x_3^0\}$ (2 occurrences).

The offline algorithm is defined in Algorithm 2. It begins by generating $\mathbb{C}$ from the genotype symbol rules (the only non-terminal symbols in the grammar). It then runs

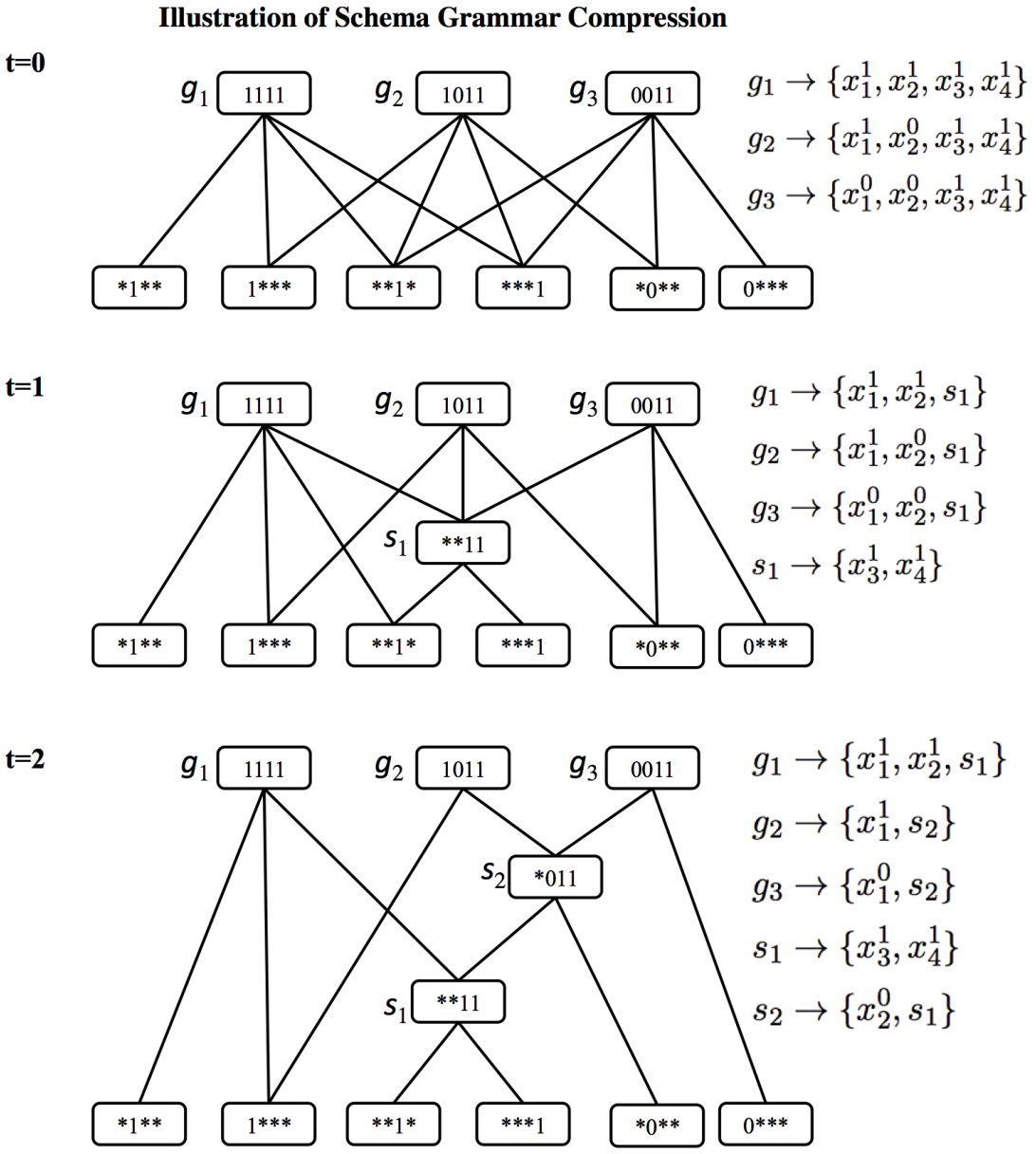

FIGURE 3.2: A sequence of Hasse diagrams showing the offline compression algorithm factorising the representation of three individuals using a hierarchy of schemata.

symbol replacement in a loop. At the beginning of each iteration the most frequently co-occurring pair of symbols $F(\mathbb{C})$ is identified. If the frequencies of two or more co-occurrences are equal then one is selected randomly. A new schema symbol is then created that yields $F(\mathbb{C})$, and each occurrence of $F(\mathbb{C})$ in the rules is replaced with the new symbol, shortening the length of each such rule by 1 symbol. Using the previous example, the first substitution would create the following rules:

$$g_1 \rightarrow \{x_2^0, s_1\}$$
$$g_2 \rightarrow \{x_2^1, s_1\}$$
$$s_1 \rightarrow \{x_1^1, x_3^0\}$$

During this replacement the multiset $\mathbb{C}$ is updated as the contents of the rules change. This substitution process repeats until no co-occurrence appears more than once anywhere in the production rules – i.e. until $F(\mathbb{C}) = \emptyset$. Larger schema symbols are formed through the recursive substitution of co-occurring symbols, creating symbol hierarchies. Figure 3.2 shows the compression process working using the poset representation of SG (Section 3.3.5).

3.4.0.5 Rule Utility

The rule utility constraint adopted from SEQUITUR and incorporated into Algorithm 2 stipulates that non-terminal symbols that are referenced only once in the right-hand-side of the production rules should be removed as they are redundant. This is best explained by an example. Consider a frequently occurring schema 1111**** (for example if the population were local optima of concatenated trap-4 functions). Without the rule utility constraint this schema would be inferred into the grammar in a hierarchy like the following:

Production rule	$\stackrel{*}{\Rightarrow}$
$g_1 \rightarrow \{s_3, x_4^1, x_6^1, x_7^1, x_8^1\}$	11111111
$g_2 \rightarrow \{s_3, x_4^0, x_6^0, x_7^0, x_8^0\}$	11110000
$s_1 \rightarrow \{x_1^1, x_2^1\}$	11******
$s_2 \rightarrow \{s_1, x_3^1\}$	111*****
$s_3 \rightarrow \{s_2, x_4^1\}$	1111****

In these production rules symbols s_1 and s_2 are only referenced once and are unnecessary in order to encode the schema 1111**** efficiently. By enforcing the rule utility constraint the production for s_1 would be back-substituted into s_2 and s_2 back into s_3, and the now-redundant symbols removed, leaving:

Production rule	$\stackrel{*}{\Rightarrow}$
$g_1 \rightarrow \{s_3, x_4^1, x_6^1, x_7^1, x_8^1\}$	11111111
$g_2 \rightarrow \{s_3, x_4^0, x_6^0, x_7^0, x_8^0\}$	11110000
$s_3 \rightarrow \{x_1^1, x_2^1, x_3^1, x_4^1\}$	1111****

A worked example of the offline algorithm including the rule utility steps is shown in Table 3.1.

Algorithm 2: Offline Compression Algorithm

input : phenotype population $\mathbb{P} = \{P_1, \ldots, P_p\}$
output: grammar instance $\mathbb{G} = \{V, G, S, \Sigma, R\}$
// initialise grammar
$\mathrm{G} = \mathrm{S} = \mathrm{R} = \Sigma = \emptyset$
// enumerate phenotypes and encode
for $P \in \mathbb{P}$ **do**
  // ensure all terminal symbols are stored
  for $\alpha \in P$ **do**
    $\Sigma = \Sigma \cup \alpha$
  end
  // encode each genotype with 1-1 mapping
  $g = \textsc{NewSymbol}()$
  $G = G \cup g$
  $R = R \cup (g \rightarrow P)$
end
// build co-occurrence multiset
$\mathbb{C} = \emptyset$
for $\alpha, \beta \in R$ **do**
  // Co-occur(β) returns all order-2 subsets in β
  $\mathbb{C} = \mathbb{C} \uplus \textsc{Co-occur}(\beta)$
end
// now implement symbol replacement loop
while $\mathrm{F}(\mathbb{C}) \neq \emptyset$ **do**
  // create a new schema symbol
  $s = \textsc{NewSymbol}()$
  $S = S \cup s$
  // add it to production rules
  $R = R \cup (s \rightarrow \mathrm{F}(\mathbb{C}))$
  // now find all rules containing F($\mathbb{C}$) (see text)
  $M = \textsc{Find}(R, \mathrm{F}(\mathbb{C}))$
  for $\alpha, \beta \in M$ **do**
    // update co-occurrence multiset from current rule
    $\mathbb{C} = \mathbb{C} \setminus \{\{A, B\} \in \textsc{Co-occur}(\beta) : A \in \mathrm{F}(\mathbb{C}) \vee B \in \mathrm{F}(\mathbb{C})\}$
    // replace co-occurring symbols with new schema symbol in rule
    $\beta = s \cup \{\beta \setminus \mathrm{F}(\mathbb{C})\}$
    // update co-occurrence multiset from updated rule
    $\mathbb{C} = \mathbb{C} \uplus \{\{A, B\} \in \textsc{Co-occur}(\beta) : s \in \{A, B\}\}$
  end
end
// now find any underutilised schema symbols (see text)
$U = \textsc{FindUnderutilised}(R, S)$
for $s, \beta_s, \beta_r \in U$ **do**
  // β_s is symbol rule and β_r is only rule where symbol is yielded
  $\beta_r = \beta_s \cup \{\beta_r \setminus s\}$
  // remove unused schema symbol
  $S = S \setminus s$
end
return $\{G \cup S, G, S, \Sigma, R\}$

Step	Action	Production Rules R
0	Initialise genotype symbols from sample phenotypes	$g_1 \rightarrow \{x_1^0, x_2^1, x_3^1, x_4^0, x_5^0\}$
		$g_2 \rightarrow \{x_1^0, x_2^1, x_3^0, x_4^0, x_5^0\}$
		$g_3 \rightarrow \{x_1^1, x_2^0, x_3^1, x_4^0, x_5^0\}$
1	Substitute $\{x_4^0, x_5^0\rangle\}$ (3 co-occurrences)	$g_1 \rightarrow \{x_1^0, x_2^1, x_3^1, s_1\}$
		$g_2 \rightarrow \{x_1^0, x_2^1, x_3^0, s_1\}$
		$g_3 \rightarrow \{x_1^1, x_2^0, x_3^1, s_1\}$
		$s_1 \rightarrow \{x_4^0, x_5^0\}$
2	Substitute $\{x_1^0, x_2^1\rangle\}$ (2 co-occurrences)	$g_1 \rightarrow \{s_2, x_3^1, s_1\}$
		$g_2 \rightarrow \{s_2, x_3^0, s_1\}$
		$g_3 \rightarrow \{x_1^1, x_2^0, x_3^1, s_1\}$
		$s_1 \rightarrow \{x_4^0, x_5^0\}$
		$s_2 \rightarrow \{x_1^0, x_2^1\}$
3	Substitute $\{s_1, s_2\}$ (2 co-occurrences)	$g_1 \rightarrow \{s_3, x_3^1\}$
		$g_2 \rightarrow \{s_3, x_3^0\}$
		$g_3 \rightarrow \{x_1^1, x_2^0, x_3^1, s_1\}$
		$s_1 \rightarrow \{x_4^0, x_5^0\}$
		$s_2 \rightarrow \{x_1^0, x_2^1\}$
		$s_3 \rightarrow \{s_1, s_2\}$
4	Back-substitute s_2 into s_3 as only referenced once, and delete s_1	$g_1 \rightarrow \{s_3, x_3^1\}$
		$g_2 \rightarrow \{s_3, x_3^0\}$
		$g_3 \rightarrow \{x_1^1, x_2^0, x_3^1, s_1\}$
		$s_1 \rightarrow \{x_4^0, x_5^0\}$
		~~$s_2 \rightarrow \{x_1^0, x_2^1\}$~~
		$s_3 \rightarrow \{s_1, x_1^0, x_2^1\}$
5	Stop (no co-occurrences with frequency > 1)	$g_1 \rightarrow \{s_3, x_3^1\}$
		$g_2 \rightarrow \{s_3, x_3^0\}$
		$g_3 \rightarrow \{x_1^1, x_2^0, x_3^1, s_1\}$
		$s_1 \rightarrow \{x_4^0, x_5^0\}$
		$s_3 \rightarrow \{s_1, x_1^0, x_2^1\}$

TABLE 3.1: A worked example of the offline compression algorithm incrementally compressing three sample individuals from a 5-dimensional binary population. Highlights in the grammar rules show changes made by each action.

3.4.0.6 Grammar Size

In this section we show that the number of schema symbols generated by the offline algorithm is linear in population size and problem dimensionality. This is intuitive: there would be no compression advantage without this property. The proof applies to all representations with finite dimensionality and not just GA problems.

Theorem 3.1. *The number of schema symbols created by the offline algorithm is upper bounded by the binary relation:*

$$|S| \leq \frac{p(n-1)}{2}$$

Where $|S|$ is the number of schema symbols, $p = |G|$ is the size of the population and n is the number of the problem dimensions.

Proof. The production rule associated with each genotype $g \in G$ will initially have cardinality $|g| \leq n$. Each schema symbol substitution replaces two symbols in a production rule with a new schema symbol (line 21 in Algorithm 2). The maximum number of substitutions is therefore $n-1$ per rule and $p(n-1)$ across the population. Given that each schema symbol replacement occurs in at least two rules (Equation 3.15), then the total number of new schema symbols is upper bounded by $p(n-1)/2$. □

3.4.0.7 Computational Complexity

In this section we show that the theoretical scalability of the offline algorithm is $O(n^2 p)$. The proof we offer relates to the construction and update of the multiset of co-occurring symbols $\mathbb{C}$ during the symbol substitution process which dominates computation and provides the lower bound to scaling.

Theorem 3.2. *The computational complexity of the offline algorithm is given by:*

$$\textsc{Offline}(n, p) = O(n^2 p)$$

Proof. For any given production rule the operation Co-occur(β) (Equation 3.12) requires each unique pair of symbols in β to be identified, which by definition requires $\frac{|\beta|(|\beta|+1)}{2}$ operations. On initiation the co-occurrence multiset $\mathbb{C}$ must be populated by calling Co-occur(β) (line 12 in Algorithm 2) for each genotype. At this stage, the length of each rule is the length of each genotype, so this requires up to $\frac{n(n+1)}{2}$ operations per genotype.

The maximum number of symbol substitutions per genotype is $n-1$ and we assume this case. On each substitution the length of the RHS of the rule $|\beta|$ is decremented by 1. The size of each rule therefore changes in an incremental series running from n to 2. For each substitution the update of $\mathbb{C}$ on line 20 requires $2(|\beta|-1)$ operations ($|\beta|-1$ for each symbol in the pair being substituted) and the update of $\mathbb{C}$ on line 22 requires $|\beta|-1$ operations. So on each substitution there are $3(|\beta|-1)$ operations. Summing these operations across the series $|\beta| = (n, n-1, \ldots, 2)$ yields:

$$3\left(1 + 2 + \cdots + (n-1)\right) \tag{3.16}$$

which is a divergent series given by the triangular number:

$$\frac{3n(n-1)}{2} \tag{3.17}$$

The number of operations per rule can thus be derived by adding the operations from the initial population of $\mathbb{C}$ with the update on substitution. This yields:

$$\begin{aligned}&\frac{n^2+n}{2}+\frac{3n^2-3n}{2}\\&=\frac{4n^2-2n}{2}\\&=n(2n-1)\end{aligned} \tag{3.18}$$

Using these operations as the dominant term and multiplying across the population gives an overall complexity $\textsc{Offline} = O(n^2p)$. □

3.5 Implementation Notes

The practical scalability of the algorithm depends on implementation details. Items that may affect scalability include the following:

1. Lookup, insert and delete on set and multiset data structures.
2. The function $\mathrm{F}(\mathbb{C})$ (Equation 3.15) which returns the most frequently co-occurring pair of symbols from $\mathbb{C}$.
3. The function $\textsc{Find}(R, \mathrm{F}(\mathbb{C}))$ which returns the production rules containing $\mathrm{F}(\mathbb{C})$.
4. The function $\textsc{FindUnderutilised}(R, S)$ which returns the schema symbols only referenced once and the rules which reference them.

The key data structure in the offline algorithm is the co-occurrence multiset $\mathbb{C}$. This is essentially a counter (or *property bag*) of all symbol co-occurrences in the grammar. A natural and computationally efficient structure for counting co-occurrences is an interaction matrix, but this would not work well here for two reasons. Firstly, the number of symbols changes during execution, so such a matrix could not be statically sized. Secondly, it would not be scalable memory-wise to dynamically size a matrix. The number of symbols scales at $O(pn)$ (Theorem 3.1). Therefore, the interaction matrix would need to scale at $O((pn)^2)$ which would quickly become infeasible. It would also be very sparse and thus inefficient with respect to memory.

A more efficient alternative is to store each co-occurring pair of symbols as distinct items in a hash map, using the value of each map entry to represent the frequency count. In this way the content of the $\mathbb{C}$ multiset can be explicitly modelled. This structure would need to be indexed using an *unordered* hash of each symbol pair (in other words $\textsc{Hash}(\{A, B\}) = \textsc{Hash}(\{B, A\})$. Bearing in mind that the function $\mathrm{F}(\mathbb{C})$ is required to return the most frequently co-occurring pair of symbols, then it is also

beneficial for the hash map to be dynamically sorted in descending order of value. By doing this the $\mathrm{F}(\mathbb{C})$ function can be implemented trivially by returning the first item in the map or $\emptyset$ if the largest co-occurrence count is 1.

In order to efficiently implement the $\textsc{Find}(R, \mathrm{F}(\mathbb{C}))$ function, which returns the production rules containing $\mathrm{F}(\mathbb{C})$, and the $\textsc{FindUnderutilised}(R, S)$ function, which returns the schema symbols only referenced once and the rules which reference them, a hash map of symbols to the rules in which they occur can be maintained. This provides a horizontal remapping of the rules, and thus is sized proportionally to pn. To find the rules in which the two symbols $\{A, B\}$ appear then the intersection operation $\textsc{Map}(A) \cap \textsc{Map}(B)$ can be performed. Using this approach the $\textsc{FindUnderutilised}(R, S)$ function can also be implemented by scanning the map for symbols where the number of referencing rules is 1.

These notes provide some insight into the best ways of implementing the algorithm, although many alternatives are possible. Investigating each of these is beyond the scope of the thesis. In the implementation used to produce the results reported here, empirical computational scalability was found to be in the region $\Theta(n^2 p)$ to $\Theta(n^3 p)$, depending on the amount of regularity present in the population.

3.6 Discussion

How can we compare the grammatical model and inference algorithm introduced by this chapter with others? In terms of SG as a compression algorithm, it is typical in the data compression literature to compare the performance of two algorithms in terms of their compression ratio. In other words, how effective they are at shrinking the size of a data stream. In the same way, one idea may be to compare compression ratios of the offline algorithm with a sequential compressional algorithm. The problem with this is that SG uses a different information model to that of sequential compression algorithms. It views the world as combinations of symbols where order is explicit, which contrasts with viewing the world as sequences of symbols where order is implicit. The compressibility of one view of the world is not directly comparable with the compressibility of another. However, it is possible to construct relative measures using the two different approaches that *are* comparable in a specific context. Chapter 6 compares the ability of SG and a sequential compression algorithm to infer problem structure. It does this quantitatively, using their compressed sizes, by constructing relative measures and exploiting recent theoretical understandings of KC in an evolutionary context.

In terms of comparing SG to other evolutionary model building algorithms, the two canonical axes of comparison are computational complexity and overall search performance. Chapter 5 deals with the latter, comparing SG against implicit and explicit reference models in global function optimisation. In terms of computational complexity,

the offline algorithm defined in Section 3.4 offers theoretical scalability of $O(n^2p)$ and empirical scalability of $\Theta(n^2p)$ to $\Theta(n^3p)$. The fairest comparison of this is with the two leading models that are capable capturing overlapping, multivariate structure: hBOA and LTGA. hBOA can theoretically rebuild its model in $O(n^3 + n^2p)$ steps (Pelikan et al., 2000), and LTGA has shown reliable scalability of $\Theta(n^2p)$ using the "pairwise distance" clustering method (Pelikan et al., 2011). Overall, then, the SG offline algorithm is generally competitive with the state of the art, but cannot match the reliable scalability of LTGA in empirical tests. Is it possible to do better?

At the heart of the offline algorithm is an adaptation of the RE-PAIR sequential lossless compression algorithm. This algorithm is simple and compresses data well, but online algorithms like LZ77 (Ziv & Lempel, 1977) and SEQUITUR are faster, use less memory, and are favoured in practice. In the same way as RE-PAIR was adapted for SG, it is likely that many alternative compression algorithms could also be adapted. This offers the potential for new SG inference algorithms with superior time complexity. A glimpse of this potential can be seen in section 7.2.3, where an online algorithm is outlined that has the potential to scale linearly in n. Being able to develop faster compression algorithms like this is important for the future of SG, in the field of evolutionary optimisation at least.

3.7 Summary

In this chapter we have introduced and formally defined the SG modelling framework. We have shown how SG can infer a compact model of an evolutionary population without sensitivity to positional bias. This creates a new capability that allows Toussaint's concept of "compact models as a search strategy" to be properly developed and used as the basis for unbiased, model-based BBO. As a lossless, compact model SG also unlocks a new way of estimating the algorithmic complexity of objects. The rest of this thesis explores these capabilities in depth.

Chapter 4

Inferring Compositional Schema Structure

The previous chapter showed how Schema Grammar (SG) can capture the information present in a population of samples in a grammatical representation. We now consider the *form* of this information. The intuition runs that by inferring the structural form of a representative *population*, a model-building algorithm may be able to generalise about the structural form of a *problem*. This can provide valuable insight in its own right. It can also be used to guide evolutionary search – a topic that is considered in Chapter 5. The literature reviewed in Chapter 2 describes a myriad of ways in which populations can be modelled. In this chapter we show that SG models are able to capture population structure in a way that appeals directly to the spirit of the Schema Theorem, BBH and Compositional Evolution. Specifically, that SG models can decompose populations of fit individuals into a *compositional schema structure* (CSS).

What better way to decompose a population than to infer how it could be composed? In the abstract, both BBH and Compositional Evolution describe a search algorithm that composes complete solutions by identifying and recombining partial solutions, or building blocks. The power of this algorithm lies in its potential to transform the scale and gradient of fitness landscapes, by using building blocks that reflect intrinsic problem structure as variation neighbourhoods. These theories describe building blocks that emerge from the recursive recombination and selection of fit schemata, creating a hierarchy of *stepping stones* (Forrest & Mitchell, 1993) from which fit solutions can be constructed. It is this hierarchy we refer to as compositional schema structure. To infer a CSS from a population is to infer a hypothesis about how it could be evolved by compositional search.

The approach we take is to define CSS in terms of potential compositional pathways to an individual, and a population of individuals, in a particular selective environment. This is a theoretical rather than practical construction that illustrates the space of possibilities.

We formalise the concept of CSS using the mathematical framework of partially-ordered sets. This framework builds on the existing concepts of schemata, building blocks and hyperplane competitions. A benefit of the poset formalism is that CSS can be visualised as a lattice using Hasse diagrams, which also allows it to be compared to SG structure.

In the second part of the chapter we use experiments to show cases where the hierarchical schema structure inferred by SG is not just any schema structure, but is also a CSS. In other words, the hierarchy of schemata used to compress the population is shown to be of above average fitness, with fitness monotonically increasing in the direction of composition. Moreover, the form of the schema structure is shown to be consistent with the intrinsic structure of the fitness function. This is shown on problems with modular, hierarchically modular and overlapping structures.

4.0.1 Walsh Analysis

The CSS concept we introduce in this chapter provides a means of describing problem structure and interpreting the model inferred by SG. An alternative way of decomposing and interpreting problem structure is through *Walsh Analysis* (Bethke, 1980; Goldberg, 1988). The methods are related as both infer a model of problem structure that is explicit in nature and defined over partitions of the search space. Walsh Analysis uses a type of Fourier Transform to decompose a fitness function into a nested sum of individual fitness contributions. Fitness contributions are defined for every partition of the problem space (including the null partition and individual dimensions). Their magnitude is given by the *Walsh Coefficient* of the partition (a product of the transform) and their sign by the bit pattern of the individual over that partition. More formally, the fitness of a candidate solution X is given by:

$$f(X) = \sum_{j=0}^{2^n-1} w_j \psi_j(X) \tag{4.1}$$

The Walsh Coefficient w_j specifies the average fitness effects of interactions that occur in the partition indexed by j. The position of the 1s in the binary representation of j indicate the dimensions included in the partition. For example, on a 4-bit problem j values of 1, 2, 4, and 8 (0001, 0010, 0100 and 1000) index the paritions covering single dimensions. The coefficient w_0 covers the null partition and is equal to the average fitness of all solutions in the space. The *Walsh Function* $\psi_j(x)$ specifies the direction of each fitness effect and is a function of the bit pattern in X over the partition. Walsh Functions can be computed easily (see references for details). They are orthogonal, allowing each Walsh Coefficient to be calculated using the 2^n possible solutions, x, and

their corresponding fitness values $f(x)$:

$$w_j = \frac{1}{2^n} \sum_{x=0}^{2^n-1} f(x)\psi_j(x) \tag{4.2}$$

Walsh Analysis has important limitations as a technique for decomposing problem structure. Perhaps most importantly, it is computationally unscalable. As can be seen from Equation 4.2, calculating a Walsh decomposition requires fitness to be fully observable and exhaustively evaluated. Secondly, the meaning of the Walsh Coefficients needs to be carefully interpreted. Each non-zero coefficient over the k dimensions indexed by j represents an epistatic interaction *within* that partition of at least order k. But this is not the same as detecting the presence or fitness of individual fitness-effecting schemata of order k. As an example, consider a "needle in a haystack" fitness function, where a specific point in an n-dimensional search space has a fitness of 1 and all other points have a fitness of 0. Intuitively, we would consider one relevant interaction to be present in the fitness function, of order n. However, a Walsh Analysis results in Walsh Coefficients that are *all* non-zero as the single order-n "needle" interaction is covered by every combination of dimensions in the space.

Despite these limitations, the technique can provide some useful insights into the epistatic structure of problems. It is by these means that researchers have been able to confirm the presence of high-order epistasis in biological species (Weinreich et al., 2013). When fitness is fully observable, Walsh Analysis also provides a faster way of calculating the average fitness of individual schemata, which we do later in this chapter using the method detailed by Goldberg (1988).

4.1 Deception

CSS builds upon the concept of deception, which has been developed by a number of authors (Goldberg, 1989a; Liepins & Vose, 1991; Whitley, 1991; Deb & Goldberg, 1993). Deception provides an important perspective for understanding problem difficulty from a compositional perspective. There is no universally-adopted definition of deception. However, the one we use here follows the approach, terminology and notation of Whitley (1991).

Essentially, deception describes the relationship between a high-order objective schema h (which can be a fully-formed individual) and the set of low-order schemata that contain it, which are referred to as the schemata *relevant* to h. For example, the schema ***000*** is relevant to 111000111, and the schema 0******* is relevant to the schema 000******. We will use the notation $\Re^h$ to specify the set of schemata relevant to h. We can say that deception exists in the problem if any schema $h_c \in \Re^h$ is *not* the

winner of its hyperplane competition. For example if $h = 1111$ and $h_c = 11{**}$ then if $f(h_c) \leq \max\{f(00{**}), f(01{**}), f(10{**})\}$, deception would exist between h_c and h. Usually, the global optimum is assumed as h, and thus deception exists if any relevant schema down to the level of individual alleles is not a hyperplane competition winner. The idea of deception was motivated in the literature by trying to understand the nature of problem difficulty. If some component part of the global optimum is not the fittest alternative then it follows that the search process could be led *away* from the global optimum (and hence be deceived).

If we interpret deception as being analogous to difficulty, this sets a very stringent criterion for a problem *not* being classified as difficult. Essentially, deception describes a solution that can't be composed in the simplest way – i.e., by assembling together the fittest building blocks from the lowest level upwards. But this ignores the possibility that the fitness of different features of the global optimum might still have higher-than average fitness. For example, consider a global optimum $f(1111) = 1.0$ and the average fitness of four competing schemata:

$$f(00{**}) = 0.9, f(01{**}) = 0.6, f(10{**}) = 0.57, f(11{**}) = 0.88, \bar{f} = 0.7$$

Where $\bar{f}$ specifies average fitness. Clearly the gradient between $11{**}$ and 1111 is deceptive, but categorising it in this way ignores the fact that is has a very high average fitness. In other words, a relevant schema can have a strong positive fitness signal *and* be deceptive, which is an oxymoron. If a search mechanism recognised both $00{**}$ and $11{**}$ from their fitness signal and used both for recombination, either stochastically or enumeratively, then there is a strong possibility (depending on the rest of the fitness gradient to 1111) that the global optimum can be composed. The term deceptive is thus in itself deceptive: it is really just a description of local problem difficulty that can often be overcome.

Deception perhaps makes more sense when considered at the level of individual binary alleles, which can only be hyperplane competition winners or losers. In this case a relevant allele which has lower individual fitness than its alternative, will by definition be of below average fitness and may well lead an algorithm to the wrong answer. At this level deception is intuitive. At higher orders with greater numbers of hyperplane competitors, the concept of deception is very limited in the amount of useful information it represents about a problem.

4.2 Compositional Schema Structure

CSS sits within the same conceptual framework of schema and schema fitness as deception. It relaxes the criteria deception implies for relevant schemata that may be of interest – from hyperplane competition winners, to schemata of above average fitness.

This provides a more intuitive understanding of how compositional algorithms, including the Schema Search algorithm we define in Chapter 5, are able to solve problems through decomposition and recombination. It is also more consistent with Watson's definition of relevant sub-problem solutions (see Section 2.1.5, also Watson, 2006, p.138-140).

Consider an n-dimensional problem. The total number of relevant schemata $|\Re^h|$ is equal to all combinations of individual alleles over the partition, except the schema h itself and the null case, giving:

$$|\Re^h| = 2^l - 2 \tag{4.3}$$

Given the above, if h is equal to an individual, then it is contained by $(2^n - 2)$ relevant schemata, ranging from n individual alleles to larger order-$(n-1)$ schemata. These schemata can be modelled as a transitive hierarchy that builds from the least specific lowest-order schemata, individual alleles, to the most specific highest-order schemata (the individual itself). Each of these is a competitor in a separate order-k hyperplane competition over the partition they occupy. We can use partially-ordered sets and Hasse diagrams, which were introduced in Section 3.3.5, to formalise and visualise this topology.

Let the relevant schema structure of any schema h be the partially ordered set $(\Re^h, >)$. The binary relation $>$ relates pairs of members of the set $(x, y) \in \Re^h$ and is true if the hyperplane specified by schema x contains the hyperplane specified by schema y. For example $000*** > 000*1*$ is true and $0*0*** > **0***$ is false. Not all members of the set are comparable (hence *partial* order). This is the case when neither $x > y$ nor $y > x$, for example when $x = 1*****$ and $y = *****0$.

Each poset creates an ordered compositional hierarchy from a set of relevant schemata. Its transitive reduction can be visualised using a Hasse Diagram, an example of which is illustrated in Figure 4.1. In the diagram, the joins (upward edges) and meets (downward edges) of the partial order have meaning. The joins specify composition of multiple lower order schemata into higher-order schemata, or in other words a hyperplane intersection. The meets from higher-order schemata into lower-order schemata specify hyperplane union.

By filtering and reordering the relevant schemata, we can now use the poset formalism to define CSS. We allow any relevant schema with average fitness and above to be a member of the CSS. This then includes all schemata that can lead to the composition of the individual, either absolutely (i.e., in a non-deceptive sense) or probabilistically. This relaxes the notion of problematic schemata invoked by deception, excluding only those unequivocally misleading schemata with below average fitness.

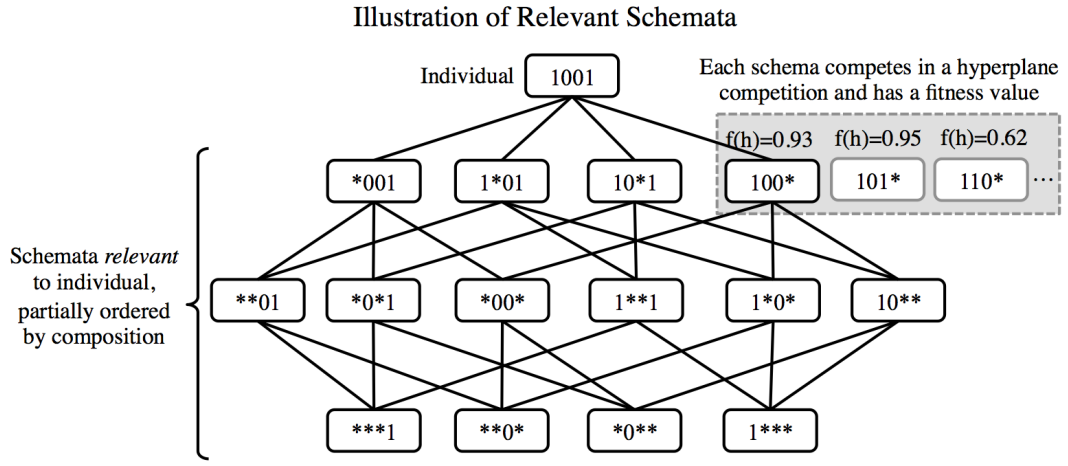

FIGURE 4.1: A Hasse diagram showing the relevant schema structure of an individual as a partially ordered set (poset). Each member of the lattice is a schema containing the individual, has an (average) fitness, and competes in a *hyperplane competition* with other schemata. The poset framework provides a natural representation for showing compositional hierarchy.

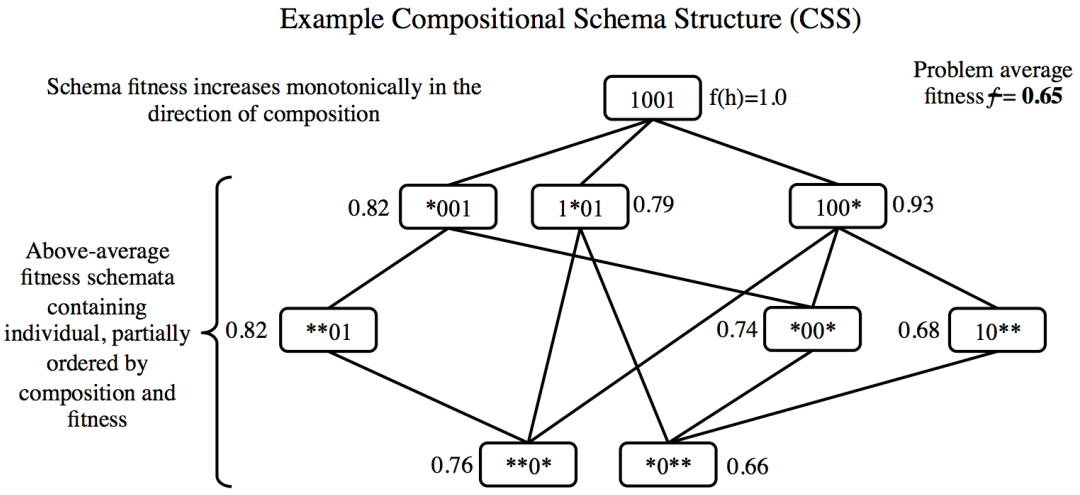

FIGURE 4.2: An illustration of the compositional schema structure (CSS) of an individual. Above-average fitness schemata containing the individual are ordered by containment and fitness. This results in a transitive hierarchy of schemata from which the individual can be composed.

A partial ordering is defined on the members of the CSS, to reflect compositions of schemata that confer a fitness advantage and excluding those that do not. As an example of the latter, consider an individual with fitness $f(1111) = 1.0$, two relevant schemata $f(1***) = 0.7$ and $f(1**1) = 0.6$ and a problem average fitness $\bar{f} = 0.5$. Our approach posits that both relevant schemata are of above-average fitness and could be selected for. Both schemata could be used to compose the individual as this represents a fitness gradient. However, there is no fitness advantage of 1**1 over 1***, and so there should be no compositional pathway between them in the CSS (in partial order terms they should be incomparable). We can formalise this with the following definition:

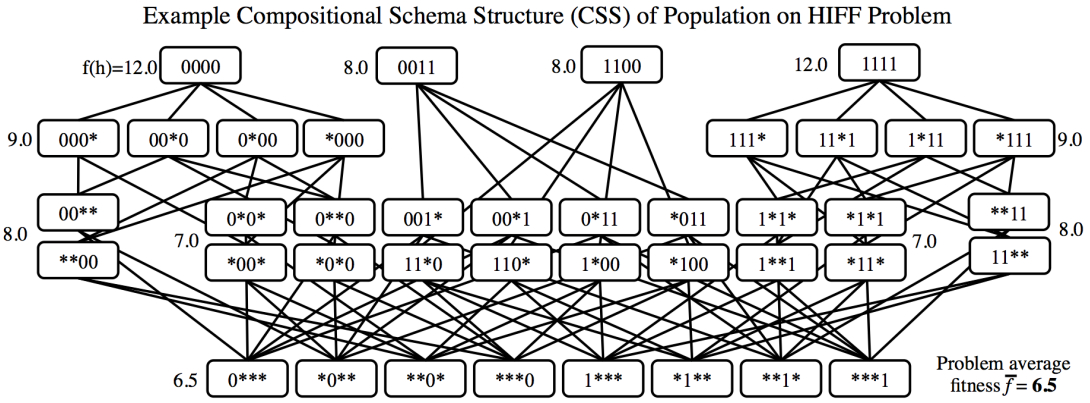

FIGURE 4.3: A Hasse diagram showing the compositional schema structure (CSS) of a population of fit individuals from a 4-bit HIFF problem. The theoretical space of compositional possibilities can be huge.

Definition 4.1. The CSS for the schema h is the partially ordered set $(\widehat{\Re}^h, \text{COMPOSES})$, where $\widehat{\Re}^h$ is the subset of h-relevant schemata that have average fitness or above:

$$\widehat{\Re}^h = \left\{ x \in \Re^h : f(x) \geq \bar{f} \right\}$$

And binary relation COMPOSES is given by:

$$x \text{ COMPOSES } y \iff (x > y) \wedge (f(x) \leq f(y))$$

In words, x COMPOSES y is true if schema x contains y (in the hyperplane sense) and it has a lower or equal schema fitness. It is named to reflect containing lower-order schema suggesting composition of the high-order schema. For example if $f(1****) = 0.6$ and $f(11*1*) = 0.65$ then $1****$ COMPOSES $11*1*$ would be true. The relation is transitive, so if $f(11*11) = 0.7$ then both $11*1*$ COMPOSES $11*11$ and $1****$ COMPOSES $11*11$ would be true. The results of this definition is a lattice of compositional pathways to a schema or individual, with fitness monotonically increasing in the direction of composition. An example CSS for an individual is shown in Figure 4.2.

The definitions provided above can be applied to any schema or individual in the search space. We can also use the same approach to define the CSS of a population of individuals. This employs the same binary relation, but over a superset of all above-average fitness schemata relevant to members in the population. Let $\mathbb{P} = \{P_1, P_2, \ldots, P_p\}$ be a population of individuals:

Definition 4.2. The CSS of the population $\mathbb{P}$ is the partially-ordered set $(\widehat{\Re}^{\mathbb{P}}, \text{COMPOSES}$, where $\widehat{\Re}^{\mathbb{P}}$ is the union of all $\mathbb{P}$-relevant schemata that have above average fitness:

$$\widehat{\Re}^{\mathbb{P}} = \bigcup_{P \in \mathbb{P}} \widehat{\Re}^P$$

In Figure 4.3 the CSS for four members of a fit population from a 4-bit HIFF problem are shown as a Hasse diagram (see Section 4.3.3 for a definition of the problem). Note that the Hasse diagram is a transitive *reduction*. For example, there is no direct join between 00** and **00 at 0000, although there is a transitive join. Although the CSS is restricted to above-average fitness individuals, and the ordering only admits those schemata with a fitness gradient between them, the figure shows that the space of theoretical compositional possibilities can be exponential in the size of the problem. A large amount of redundancy is clear, with multiple opportunities for constructing the same schema.

4.2.1 Summary

In this section we have shown that the structure of a problem can be understood as a population of schemata of above-average fitness, partially ordered by fitness composition. This provides a topology to schema structure and a way of understanding how compositional algorithms may use the structure as a fitness gradient to the fittest solutions. In the next section we show that SG models are able to infer a *compact* CSS from a population of representative individuals. In other words, they are able to infer a hypothesis about how compositional evolutionary algorithms may be able to construct fit individuals.

4.3 Inferring Compositional Schema Structure

The nature of lossless compression is such that given a sample population, a structure of some kind will be modelled. The question is whether this is the *right* structure. More specifically, does the structure used to compress the population reflect the intrinsic structure of the selective environment, and does it form a compact CSS? This section describes a series of experiments in which representative populations from a variety of selective environments are modelled using SG and the offline algorithm. Examining the SG model inferred from each population demonstrates the ability of the model to infer the structure of the selective environment.

The SG model infers a G-P map and a collection of factorised genotypes for each population. Three different techniques are then used to analyse the schema structures inferred into SG. Firstly, the derivations of the global optima in each grammar model are extracted and analysed. These derivations show the inferred building block decomposition of each optimum and allow consistency with CSS to be validated. Secondly, the complete library of schema symbols in each grammar is extracted and visualised spatially, allowing qualitative comparison with intrinsic fitness function structure. Finally, the fitness distributions of the inferred schemata are compared to the expected statistical

pattern of CSS: schemata of above average fitness, with fitness monotonically increasing with order.

Two building block test problems are examined first: modular trap functions and HIFF. For these problems the nature of the structure inferred by the grammar is straightforward to assess as the problems are contrived and their structures well studied. Both fitness functions have a modular decomposition which is reflected in the structure of the fittest individuals – and which the grammar would therefore be expected to infer. In the case of HIFF, the decomposition is hierarchical and this should be reflected in the grammar. Then, problems with less regular structures are analysed: 2D spin glasses and nearest-neighbour NK landscapes. In these problems, structure is randomised and no *specific* forms can be expected. Nevertheless, the neighbourhood structures of both problems *are* regular and spatially contiguous (in a functional sense – neighbourhood structure is still regular when the problem dimensions are shuffled). This means that where synergistic epistatic interactions occur in the fit population, although they may be irregular and overlapping, they can still be expected to occur in spatially contiguous neighbourhoods. In these cases we would expect to see regularity in the spatial distribution of inferred schemata.

For clarity of presentation the un-shuffled versions of each problem type are used in this section. The results in Chapters 5 and 6 prove that SG is not sensitive to positional bias, so this is done without loss of generality.

4.3.1 Methodology

Given the computational demands of Walsh Analysis, and to allow the results to be easily visualised, small problem instances were used: $n = 12$ in all cases except for HIFF, where 3-level $n = 8$ instances were used. These problem sizes are insignificant in practical terms. However, they allow the concept to be proven in principle and the nature of SG models to be visualised in a way that is understandable. This is sufficient for present purposes. Much larger problem sizes are used in Chapter 5 to assess the performance of SG models in global function optimisation.

The inferred grammar structure for a representative instance of each problem is captured. One instance of each problem is created (random in the case of NK-landscape and 2D spin glass) and the fittest individuals in the solution space, including the global optimum, are sampled. For the $n = 12$ problems this is set to the most fit 1% of the population, for 8 bit HIFF this is set to the top 15%. The sample population is then compressed using the offline algorithm. Each schema symbol inferred by the grammar and its transitive closure (recursive expansion) is recorded. The fitness of each of the schema symbols is then calculated using the Walsh Coefficients and the method described in Goldberg (1988). The derivation of each global optimum is also extracted and partially ordered

as per Section 3.3.5. The fitnesses of each schema symbol are superimposed onto Hasse diagrams of each derivation, allowing them to be assessed in terms of CSS. Finally, the statistical profile of inferred schemata is captured for 2d spin glass and NK landscape problems across multiple random problem instances. The fitness and order of all inferred schemata is recorded across 1000 random trials of each problem type.

To allow comparison across multiple trials and for consistency of presentation across problem types, fitness values are normalised using the following function, which is generated for each problem instance:

$$f_n(X) = \frac{f(X) - \bar{f}}{\max(f) - \bar{f}} \tag{4.4}$$

Where $f(X)$ is the original fitness function, $\bar{f}$ is average schema fitness (equal to the w_0 Walsh Coefficient) and $\max(f)$ is the fitness of the global optimum. This linear remapping returns a transformed fitness of 0 for solutions of mean fitness and 1 for the global optimum, with all other fitnesses scaled accordingly.

4.3.2 Concatenated Traps

Deceptive trap functions (Deb & Goldberg, 1993) are a commonly used mechanism for creating test problems with a modular structure and a tuneable level of difficulty. The concatenated variant of the problem creates a fitness function by composing m identical *trap functions* together and summing their individual fitnesses. The result is a structure exhibiting separable modularity (Section 2.1.5). Each function is defined over k separate input bits, creating a total problem length of $n = mk$ bits. The fitness contribution of each trap is a function of the number of the bits in the trap that are equal to 1. We use $\Psi = (\psi_1, \psi_2, \ldots, \psi_m)$ to represent the partition of an input string X into functions. The fitness of each function ψ is as follows:

$$f(\psi) = \begin{cases} f_{\text{high}} & : \text{if } u(\psi) = k \\ \left(1 - \frac{u(\psi)}{k-1}\right) f_{\text{low}} & : \text{otherwise} \end{cases} \tag{4.5}$$

Where k is the size of the function, $u(\psi)$ is the number of bits in ψ set to a value of 1, and $0 < f_{\text{low}} < f_{\text{high}}$. Unless otherwise stated, values of $f_{\text{high}} = 1.0$ and $f_{\text{low}} = 0.9$ are used. The two bit-flip local optima of each trap are when all the bits are 0s ($f(\psi) = f_{\text{low}}$) and when all the bits are 1s ($f(\psi) = f_{\text{high}}$). In all other states except the optimum there is a negative correlation between the number of 1s and the fitness contribution. This property misleads most local search mechanisms and so presents a source of problem difficulty. Finding the optimum configuration of each trap function typically requires stochasticity in initial conditions or variation to find the right basin of attraction. The

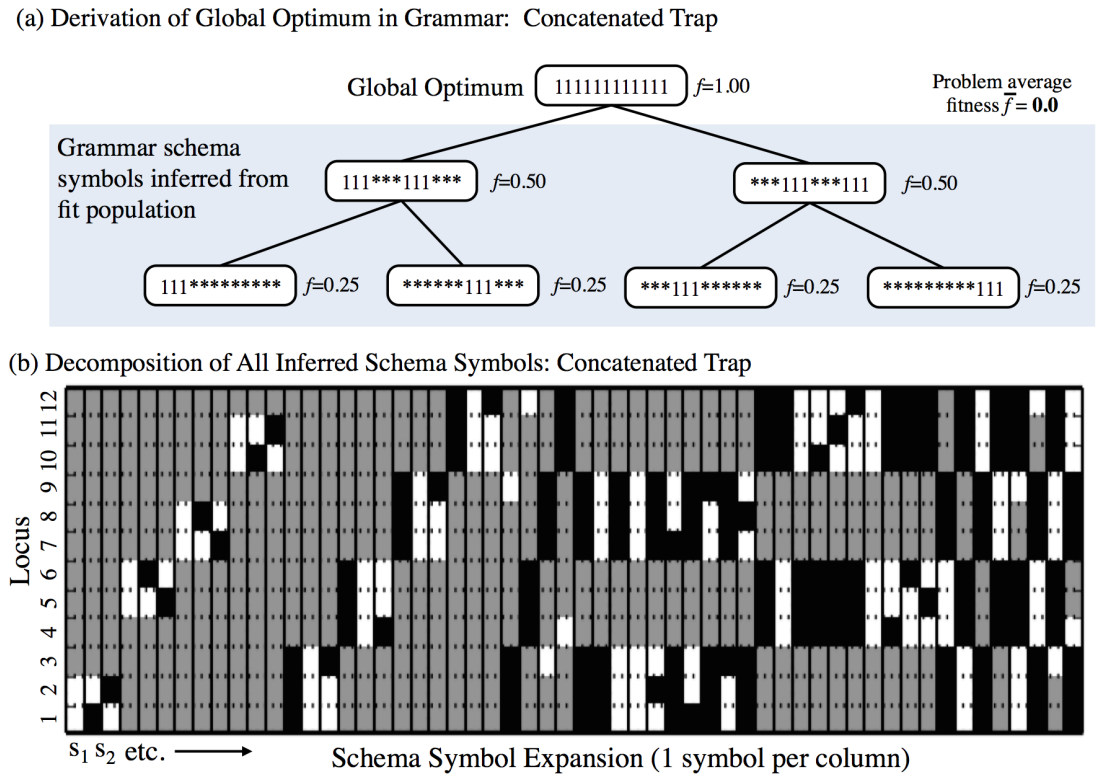

FIGURE 4.4: The structure inferred by SG using the offline algorithm and a population of fit individuals on a concatenated trap problem. (a) Derivation of the global optimum in the grammar, shown as a Hasse diagram. Each node represents a reusable building block (symbol) used by the grammar to compress the population. The inferred structure of the global optimum is a compact CSS: schemata of above average fitness partially ordered by containment and fitness. It is also consistent with the intrinsic structure of the fitness function. (b) The complete set of schema symbols inferred from the population. Each row represents a locus and each column a schema symbol. White squares represent 0s, black 1s and grey not specified. The grammar captures different combinations of local optima at multiple levels.

overall fitness value of a concatenated trap problem is a sum of the fitness contributions of each function, defined as follows:

$$f(\Psi) = \sum_{i=1...m} f(\psi_i) \tag{4.6}$$

Problem difficulty caused by the deceptive building blocks is bounded. If the variation neighbourhood of search is restricted to the two local optima of each function then the problem is separable (although this requires a compositional algorithm that can infer and exploit this neighbourhood). In CSS terms, the fittest solutions in the search space can be composed using recursive recombination of local function optima.

What would we expect an ideal compact model to look like in this experiment? The fittest individuals in the sample population are generally constructed from combinations

of module configurations, such as 111*********, that follow the intrinsic modular structure of the original problem. Each one of these module configuration appears multiple times in the population, creating redundancy and an opportunity for compression. To a lesser extent, specific higher-order combinations of these module configurations also appear multiple times, for example 111***111***. The schema structure in an ideal compact model, then, would be expected to contain a hierarchy of schema symbols. This hierarchy would include *all* of the relevant module configurations at the lowest level as well as specific combinations of these modules. By capturing modular decomposition in this way, the model would be able to represent many different combinations of fit module configuration in a compact way.

The results of the experiment on a $k = 3$ concatenated trap problem are shown in Figure 4.4. The top part of the figure shows the inferred schema structure of the global optimum, which reflects the following production rule derivation (the symbol indexes have been changed for clarity):

Production rule	$\stackrel{*}{\Rightarrow}$
$g_1 \rightarrow \{s_1, s_2\}$	111111111111
$s_1 \rightarrow \{s_3, s_4\}$	111***111***
$s_2 \rightarrow \{s_5, s_6\}$	***111***111
$s_3 \rightarrow \{x_1^1, x_2^1, x_3^1\}$	111*********
$s_4 \rightarrow \{x_7^1, x_8^1, x_9^1\}$	******111***
$s_5 \rightarrow \{x_4^1, x_5^1, x_6^1\}$	***111******
$s_6 \rightarrow \{x_{10}^1, x_{11}^1, x_{12}^1\}$	*********111

These results clearly show that the grammar has decomposed the solution using a nested hierarchy of schemata of two different orders, including a decomposition of the individual local optima. Each of these schemata are used at least twice to compress individuals in the population – they are common building blocks. The lack of order 2, 4, 5, 7, and 8 schemata is explained by the rule utility constraints of offline inference (Section 3.4.0.5). Intermediate symbols of these orders are created during bottom-up inference. However, they are only ever referenced once (by their parent) and are thus merged. This process creates a clearly-identifiable building block structure that reflects the structure of the fitness function.

The figure also shows that the derivation forms a CSS. In other words, the partial order of symbol production is consistent with a partial order of schema containment and monotonically increasing schema fitness. This gives credence to the idea that the schema structure inferred by SG is a hypothesis for how the solution could be evolved in a compositional search. It also shows that this is a *compact* CSS: a small subset of the complete space of possibilities specified by Definition 4.1, pruned for redundancy.

In Figure 4.4(b) the complete schema symbol structure of the grammar instance is expanded. In the figure, each row represents a locus and each column a schema symbol. White squares represent 0s, black 1s and grey not specified. The figure demonstrates that the grammar contains only a compact subset of all schemata relevant to the population. The number of schema symbols is upper bounded by $p(n-1)/2$ (Theorem 3.1), although in this case there are only 56 schema symbols used in the derivation of 40 individuals. This low ratio indicates a highly regular structure that SG is easily able to compress. The figures show that inferred schema structure is consistent with intrinsic modular structure of the fitness function: schemata representing combinations of module configurations with above average fitness.

The results show that the grammar can infer some schemata with below average fitness (e.g., 10******), even when all members of the population have above-average fitness. Where these appear in the derivation of certain individuals, we can say that the grammar structure is not strictly compositional. Throughout the experiments these schemata were found in the minority of cases (this is best illustrated by Figure 4.8).

4.3.3 Hierarchical If-And-Only-If (HIFF)

Hierarchical if-and-only-if (HIFF) problems are a type of hierarchically decomposable function (HDF) (Watson et al., 1998). We use them here to examine the capabilities of SG on problems containing hierarchical modular interdependence (Section 2.1.5). Each HIFF problem is defined over multiple logical levels. At the lowest level ($l = 1$) the input vector is divided into n/k non-overlapping partitions with an identical fitness function on each, as with concatenated traps. The fitness function used is an if-and-only-if (IFF) function that has a value of k^l if the input bits are either all 1s or all 0s, or 0 otherwise. We use $k = 2$ functions in all cases. This creates two optima of equal fitness for each function – 11 or 00.

At subsequent levels the IFF functions recurse, such that the first k functions on one level feed the first function on the next level. This creates a logical pyramidal structure with a single IFF function at the top level. To create the input bit for the next logical level an IFF function maps its k input bits to a single output bit that represents its overall state. The output value of each function is set to 1 if the function is in an all 1s configuration, 0 for an all 0s configuration and null ("-") otherwise. This structure creates a fitness bonus for co-ordinating module configurations. For example, composing together 11** and **11 into 1111 creates a fitness bonus. It also introduces hierarchical modular interdependency into the problem structure. This structure is best illustrated by an example – a decomposition of fitness for a $k = 2$, $l = 3$, $n = 8$ HIFF problem with the input vector 10001111 is shown in Figure 4.5.

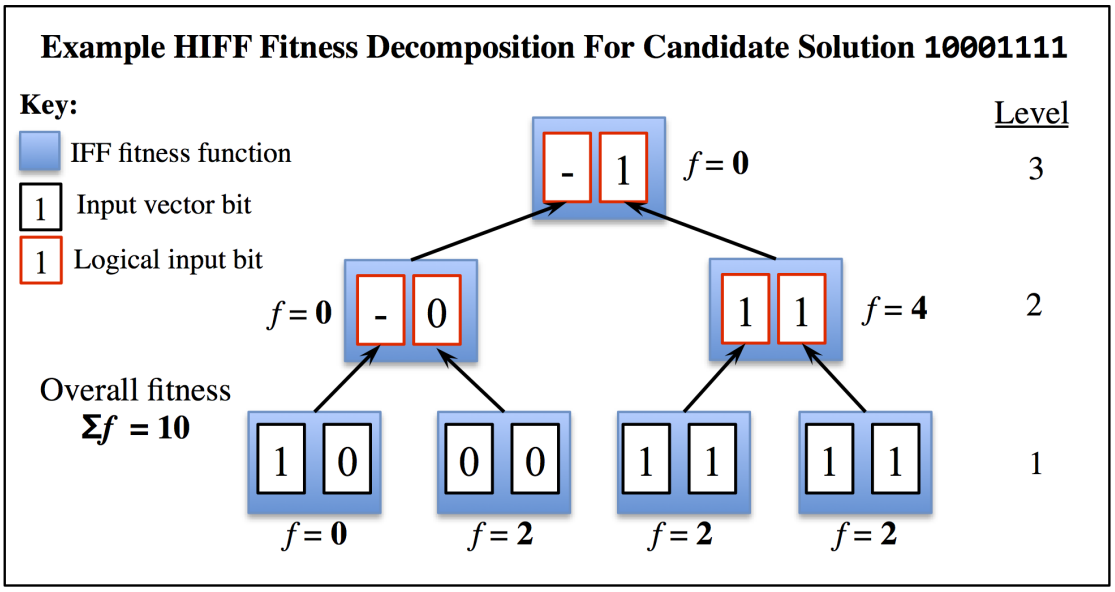

FIGURE 4.5: A fitness decomposition of the candidate solution 10001111 on a HIFF problem instance with 3 hierarchical levels. At the lowest level each fitness function is defined over a partition of the input bits. At subsequent levels each function is defined over "meaning" bits provided by the functions below.

HIFF problems are deceptive in the formal schema sense as although the composite features of each global optimum have average or above average schema fitness, they are not unambiguous hyperplane competition winners. To overcome this compositional ambiguity requires the modular composition of the problem to be learned and variation to be facilitated at multiple levels of scale.

The results of the experiment for the 8-bit HIFF problem are shown in Figure 4.6. The results show that, as with concatenated traps, SG infers a compact schema structure that is consistent with the hierarchically modular structure of the fitness function. In the top half of the figure, the derivation of one global optimum is shown. The derivation is composed of a compact two level hierarchy that includes the fittest module configurations at each level. The fitnesses and ordering of the schemata show that the derivation is consistent with CSS. In the bottom half of the figure, the rest of the schemata are shown to be generally consistent with the hierarchically modular structure of the fitness function. As with concatenated traps, alternative configurations are captured for each module.

4.3.4 2D Spin Glasses

Ising spin glass systems are simple models of physical systems that exhibit disorder. Each system is defined in terms of n identical nodes or *spins* that can be in one of two symmetrical states, 1 or -1 (mapped to 1 or 0 in binary representations), and weighted edges that connect some or all of the nodes. Each weighted edge w_{ij} specifies a constraint

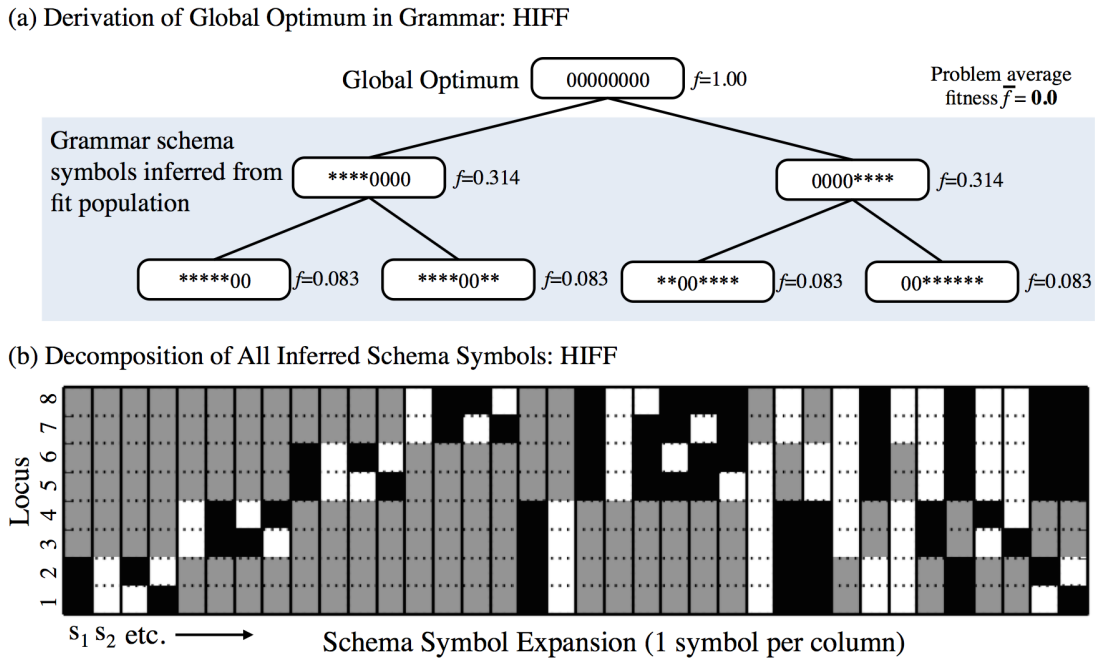

FIGURE 4.6: The structure inferred by SG using the offline algorithm and a population of fit individuals on a 3-level $n = 8$ HIFF problem. The figures show that SG is able to infer all three scales of hierarchical organisation in the population for both global optima.

between the pair of nodes $\{x_i, x_j\}$ it connects: if it is positive, nodes with correlated (matching) states are preferred; if it is negative, nodes with anti-correlated states are preferred. The magnitude of the weight specifies the magnitude of the constraint. These constraints can be in conflict, creating *frustration* in the system that is measured in terms of energy. The objective in optimising a spin glass is to find the *ground state(s)* in which energy (and thus frustration) is at its minimum. There are at least two global optima for each problem instance due to symmetry: the complement of each ground state will have the same energy. Frustration is analogous to epistasis and is a source of problem difficulty.

The local energy of each pairwise constraint is calculated by negating the product of the node states and the edge weight. For example, if $x_i = -1$, $x_j = -1$ and $w_{ij} = -1$ this gives $-x_i x_j w_{ij} = 1$. Note that the sign of the two node states is irrelevant, only whether they are correlated or anti-correlated. For example $x_i = 1$, $x_j = 1$ gives the same result as the previous case. The total energy of the system is given by:

$$E(X) = -\sum_{ij}^{n} w_{ij} x_i x_j \tag{4.7}$$

We calculate fitness by reversing the sign of the energy function to have consistency with the other test problems used, and present optimisation as a maximisation rather than minimisation task:

$$f(X) = -E(X) \tag{4.8}$$

Nodes on the 2D spin glasses we use here are arranged as a wrapped square lattice with each node connected to its 4 spatially adjacent nodes. We use edge weights randomly chosen from $\{+1, -1\}$. This subset of problem instances are known as $2D \pm J$ Spin Glasses (Saul & Kardar, 1994). We use them here as the ground states for random instances that can be solved in polynomial time, enabling them to be used as optimisation benchmarks in evolutionary computation literature (Pelikan & Goldberg, 2003; Goldman & Punch, 2014). 2D spin glasses are randomised problems that do not contain predictable explicit structure. However, where structure emerges this is expected in spatially contiguous regions of the problem space.

The results of the experiment for the 2D spin glass problem are shown in Figure 4.7. In the upper panel, one of the two complementary global optima and its grammar derivation are shown using a spatial representation of the problem. The 1D GA representation (shown for each schema in the bottom half of the figure) maps to the spatial representation in a left-to-right, top-to-bottom order. For example, locus 1 maps to cell (1, 1), locus 4 maps to cell (1, 4) and locus 5 maps to cell (2, 1). Edges extending at the boundaries indicate toroidal wrapping. For example, the four connected neighbours of cell (1, 1) are cells (1, 2), (2, 1), (1, 4) and (3, 1). The figure shows that the optimum is compressed using a compact CSS, with all schemata of above-average fitness and fitness monotonically increasing in the direction of composition. A spatial order can also be seen, with contiguous spatial interaction agglomerating up the hierachy, to cover neighbourhoods of increasing size.

A spatial pattern is also present in the lower part of the figure, with most schemata occupying contiguous regions of the (2D) space. What can also be seen is a certain a degree of redundancy caused by the symmetric nature of spin glasses. In most cases, schemata are inferred in complementary pairs (for example *00********* and *11*********). This is the result of inference detecting absolute, rather than relative, correlations between variables.

An interesting feature of the results is that although SG can model overlapping structure that is not strictly nested (for example 11* and *11), only a small amount of this can be seen in the schemata (it is better observed in the next section). Instead, an implicit linkage structure appears to emerge that is overlapping in a nested, hierarchical sense. Given the amount of overlap intrinsic in the energy function, this is perhaps unexpected. One possible reason is that these problems may contain a *critical backbone* of problem dimensions that form the basin of attraction of the fittest solutions (Prugel-Bennett, 2007), and decomposition of the fittest solutions may be shaped around this backbone.

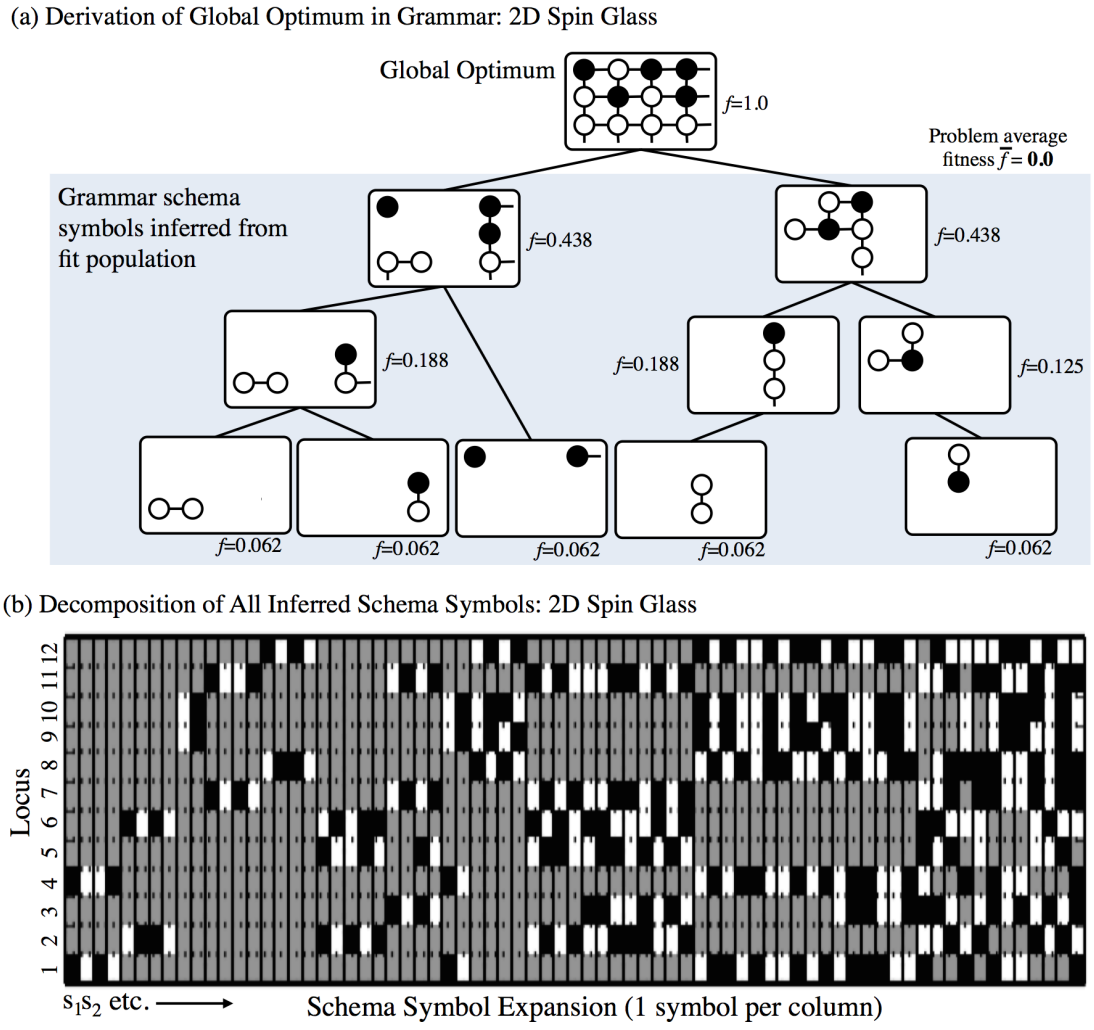

FIGURE 4.7: The structure inferred by SG using the offline algorithm and a population of fit individuals on a 2D spin glass problem where $n = 12$. (a) The phenotype expansions of each symbol are rendered on the spin glass lattice and show the inferred hierarchical spatial structure of a global optima. Edges extending at the boundaries indicate toroidal wrapping. (b) Complementary spin configurations for each of the two global optima are successfully inferred. White and black cells indicate + and − spin states respectively.

An additional experiment run on the spin glass problem captures the distribution of schema fitnesses over 1000 random problem instances. This provides an additional perspective for analysing inferred schema structure. The results of this experiment are shown in Figure 4.8(a). The distributions are arranged by order of schema, and they show a statistical pattern that is consistent with the presence of CSS: schemata of above-average fitness with fitness monotonically increasing with schema order.

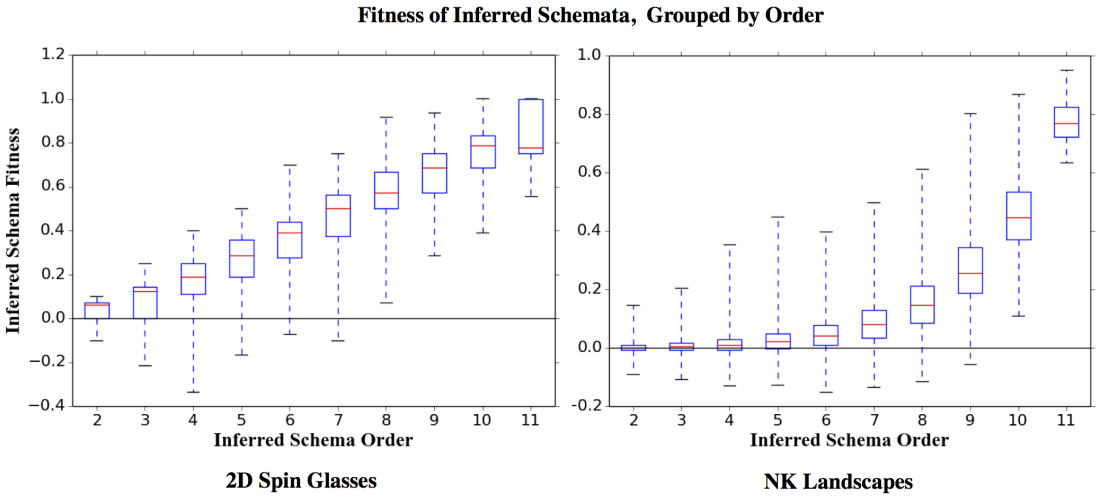

FIGURE 4.8: The distribution of schema symbols inferred by SG on 1000 random instances of 2D spin glass and NK-landscape problems. The distribution of schema fitnesses is grouped by order of schema. The figure shows that SG captures above-average fitness schemata with fitness monotonically increasing in the direction of hierarchical composition. The box plot whiskers extend to the minima and maxima.

4.3.5 Nearest Neighbour NK-Landscapes

NK-Landscapes are a well known test problem containing randomised epistatic interactions of various orders (Kauffman, 1993). The amount of epistasis in each problem is tuned using parameter $k = \{0, 1, \ldots, n-1\}$, which specifies the number of "neighbours" of each bit. The fitness contribution of each bit x_i is taken from a lookup table f_i, which maps each of the 2^{k+1} unique states of the bit and its neighbours to a randomised value in the range 0 to 1. In fitness landscape terms k controls the *ruggedness* of the landscape, and allows the difficulty of the problem to be tuned. When $k = 0$ each bit has no neighbours and the function is fully separable. As k increases the amount of epistasis in the problem increases, creating an increasing number of local optima.

The nearest neighbour variant of the problem was used, in which the k neighbours of each bit are taken from its immediate neighbours in the problem representation. Where the neighbourhood extends beyond the end of the individual then it is wrapped. The nearest-neighbour structure provides more intrinsic regularity than selecting neighbours randomly, as each bit participates in the same number of epistatic interactions. In the unshuffled version of the problem used here, neighbours are arranged contiguously, therefore some sequential regularity can also be present.

The overall fitness F of a n-bit bitstring X with state $\{x_1, x_2, ..., x_n\}$ is given by:

$$F(X) = \frac{1}{n} \sum_{i=1}^{n} f_i(x_i, \Omega(i)) \tag{4.9}$$

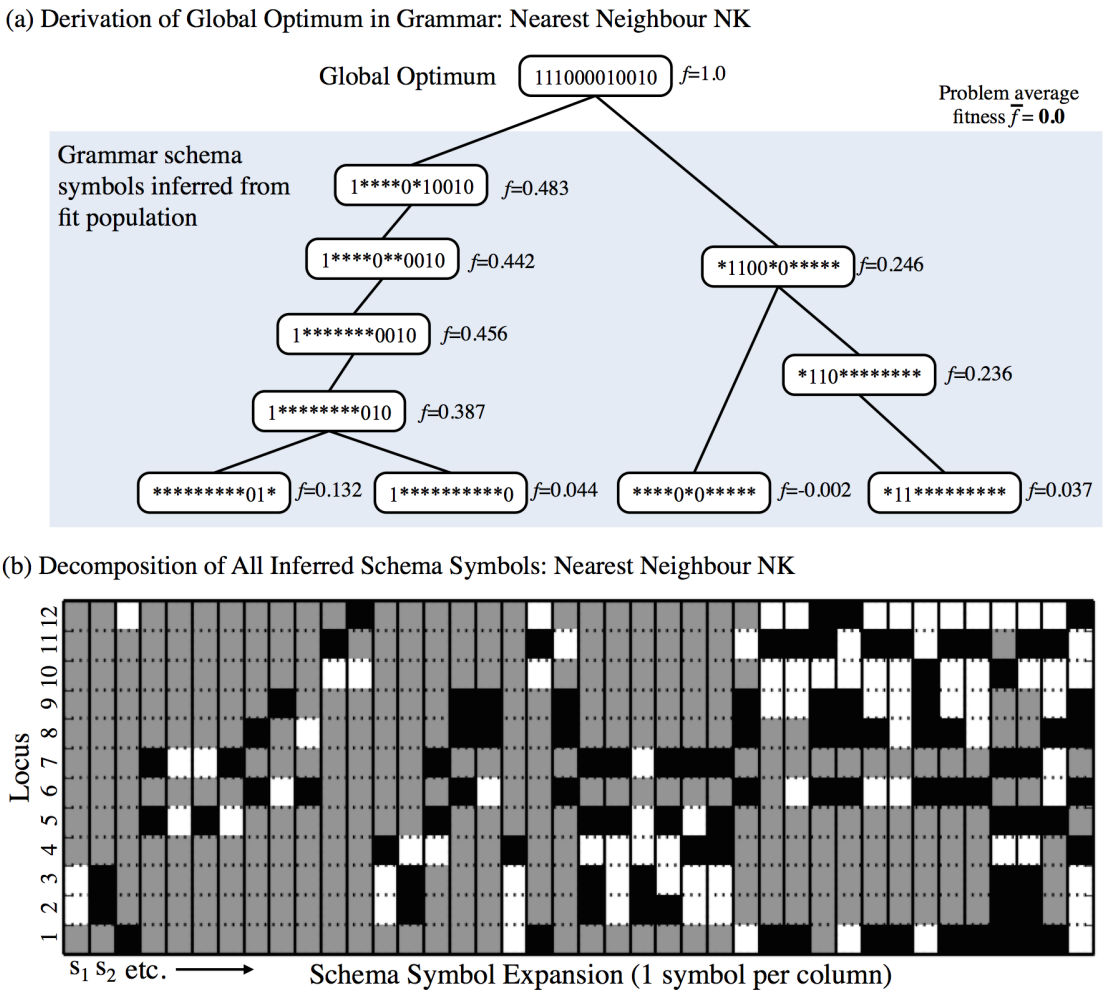

FIGURE 4.9: The structure inferred by SG using the offline algorithm and a population of fit individuals on a nearest-neighbour NK landscape problem where $n = 12$ and $k = 3$. The inferred schemata demonstrate SGs capability to capture irregular overlapping structure.

Where $\Omega(i)$ is the state of the k neighbours of bit i. Nearest-neighbour NK landscapes are randomised problems and their specific structure is unpredictable. However, given that their epistatic neighbourhoods are formed in a regular, contiguous, overlapping pattern, then structure can be expected to emerge in spatially-contiguous regions.

The results of the first experiment for the nearest-neighbour NK-landscape problem are shown in Figure 4.9. The derivation of the global optimum is shown in the top part of the figure. In this case there is a deeper, more complex hierarchy. Note that some of the larger schemata appear to only cover one smaller schema – this is because the joins with terminal symbols are not shown in the diagram for brevity. Both branches of schemata used in the derivation occupy particular spatial neighbourhoods, with these becoming more specific in lower orders. This follows the neighbourhood structure of the fitness function. The derivation is largely consistent with a compact CSS, although there are two anomalies. Firstly, the schema ***0*0***** is inferred by the grammar, which is

of below average fitness. Secondly, fitness does not monotonically increase between the schemata 1*******0010 and 1****0**0010.

Figure 4.9(b) shows the full set of schema symbols inferred by SG. This is a compact subset of all relevant schemata of the population – 40 building blocks reused by 40 individuals. The schemata are generally formed in contiguous regions of locus space, which is consistent with the spatial structure of the fitness function. As with spin glasses, inferred schemata suggest an implicit linkage structure or critical backbone in the problem. In this case though, un-nested, overlapping schemata are more prevalent (for example, *001******** overlaps with ***01*1*****, *****1*1**** with *****0**1***), showing that SG is capable of capturing these features. The results of the second experiment, capturing the distribution of schema fitnesses over 1000 random problem instances, are shown in Figure 4.8(b). Again, they show a statistical pattern that is consistent with the presence of CSS: schemata of above-average fitness with fitness monotonically increasing with schema order.

4.4 Discussion

The CSS framework presented in Section 4.2 provides a topology to schema structure that appeals to the nature of compositional search algorithms. Each CSS is relevant to a selective context and can be used to describe both individuals or populations of individuals. It provides an additional perspective for thinking about problem difficulty. In contrast to deception, which has a focus on compositional problems, it highlights compositional opportunity. Figure 4.3 shows that the space of these opportunities can be very large.

It is interesting to consider whether the compositional order of relevant schemata generally follows the order of fitness. In other words, to what extent is it naturally the case that smaller order schemata have a lower average fitness than the higher order schemata they contain. If this is true, the implication is that in many problems a natural compositional schema structure is likely to exist in a fit population. This structure would provide a natural gradient toward the fittest solutions for compositional search algorithms that were able to incrementally infer and then exploit schema structure. By definition schemata contain exponentially fewer points in search space as their order increases. Variance in schema average fitnesses can therefore be expected to increase as order increases[1] and be at its highest at the level of individual solutions. Therefore, to the extent that a problem is decomposable, it is logical that higher order relevant schemata are generally higher fitness than the lower order relevant schemata. The experimental results shown in Figure 4.8 provide support for this theory,

[1]Note that this concept of variance refers to the distribution of schema fitnesses and not the fitness of solutions each schema contains. The latter is known as schema variance (Goldberg & Rudnick, 1991), and this is generally (but not always – (ibid.)) expected to *decrease* in higher orders.

The results in Section 4.3 demonstrate the SG model inferring schema structure from a variety of problems. This schema structure is shown to be compact, consistent with the structure of the fitness function, and in most cases exhibits a compositional fitness gradient from low to high order. This provides strong evidence for the claim that SG can infer a schema structure that is consistent with the compositional structure of the selective environment. Each model is a hypothesis for how a population could be evolved by compositional evolutionary search. In Chapter 5 this information is used to generalise about the selective environment, by using inferred schema structure as an exploration distribution.

It is perhaps not remarkable the SG can learn the decomposition of simple building block problems. There are numerous examples of multivariate model building algorithms that can do something similar, and both hBOA (Pelikan, 2002) and LTGA (Thierens, 2010) can do so using a hierarchical model. Both support the modelling of the nested hierarchies that are intrinsic to these problems. In problems with a less regular structure, hBOA can also capture overlapping structures that are not strictly nested. However, there are two key reasons why the results here are novel and significant. Firstly, SG is the first model that is able to decompose an explicit, hierarchical *schema* structure from a population and describe each individual in the population in terms of this structure. In some ways this closes the loop between model-building, Schema Theorem and BBH. Secondly, it proves that compact models can effectively capture non-sequential structure on combinatorial optimisation problems. Together, these contributions create significant new research opportunities for evolutionary model building.

It is important to note that the schema structures used in this chapter were on un-shuffled problems as this made the results easier to visualise. When positional bias is removed from the representation using the random shuffling, then the same functional results are observed. SG inference is driven by the combinatorial co-occurrence of symbols, and is order independent.

4.5 Summary

In this chapter a new framework was introduced for describing the schema structure of decomposable problems in terms of partially ordered sets. Compositional Schema Structure (CSS) allows schema structure to be understood and visualised as a hierarchy of above-average fitness schemata, with fitness monotonically increasing with schema order. CSS captures the idea of schemata acting as a fitness gradient for compositional model-building algorithms. Experimental results demonstrated that SG models are able to successfully infer CSS from populations of fit individuals in a variety of selective environments, supporting the claims of this thesis. Chapter 5 explores the potential of this structure for directing evolutionary variation.

Chapter 5

Compact Models as a Search Strategy

Chapter 4 illustrated the form of schema structure captured by SG models and showed cases where this form was consistent with the compositional structure of the selective environment. This chapter investigates the qualities of SG schema structure with respect to evolutionary variation, and how it can be successfully applied to practical function optimisation.

Evolutionary populations only reveal partial information about the selective environment from which they are sampled. However, if regularity exists in the problem space then there is a possibility to *generalise* this information. In the current context, to generalise means to use information about the selective environment to generate offspring that are both high quality with respect to the objective function *and* novel. Arguably, the performance of evolutionary systems attempting to minimise or maximise objective functions are ultimately driven by these abilities. The typical interpretation of fitness in evolutionary computation is the state of the population with respect to the objective function. Although this use of fitness can measure progress toward an objective in absolute terms, it does not capture the ability to generalise. This has lead researchers to propose the concept of *effective fitness* (Stephens & Vargas, 2000; Toussaint & von Seelen, 2007). Effective fitness is a more holistic measure of adaptedness, which considers not only the (objective function) fitness of individuals but the number of fit offspring they are able to produce. For an evolutionary system to be effectively fit, the variation neighbourhood used to evolve the population must respect the constraints of phenotype space. Not only does this protect against disastrous mutations that violate key dependencies, it also allows problem decomposition to be exploited, by recombining semi-independent partial solutions together. Effective fitness has been suggested as an explanation for the evolution of effective representations (Nordin & Banzhaf, 1995; Stephens & Vargas, 2000; Toussaint & von Seelen, 2007). In this thesis, where we infer

representations directly, it provides a useful conceptual framework for thinking about the fitness of those representations.

The first half of the chapter examines the qualities of the SG model with respect to effective fitness and its ability to generalise. It shows that by inferring a language of variation expressed in terms of hierarchical schema structure, SG is able to guide evolutionary search in an explicitly directed way. Experiments are used to analyse the variation neighbourhoods created by SG models, in a variety of selective environments. The quantity and fitness of novel mutant offspring produced by SG-guided variation is compared to unguided recombinative variation using uniform crossover. These results show that on the tested problems, uniform crossover produces only a fraction of the fit novel offspring produced by SG-facilitated variation. SG is also compared to a state of the art model-guided variation method, LTGA global mixing, and is shown to have comparable or superior performance in all cases.

In the second part of the chapter the model is applied to global function optimisation, establishing SG as an instance of a *Compression EA* (Toussaint, 2006). The search algorithm follows the general framework of Multi-Scale Search (MSS) (see also Section 2.2.5). Essentially, MSS describes evolutionary algorithms that use iterated model-guided hill-climbing as a variation mechanism. MSS exploits models that are able to specify an explicit variation neighbourhood for multi-scale hill climbing. This approach benefits from the efficiency of systematic trial-and-error search, combined with the opportunity for large scale recombination. Indeed, recent theoretical work has proved that MSS can radically reduce the complexity of additive decomposable problems (Mills et al., 2014). Arguably, MSS-style search is emerging as the state of the art in model-building EAs. It can be seen in a series of recent works that demonstrate superior performance to standard EDAs on a variety of problems. This includes work utilising LTGA global mixing (Thierens & Bosman, 2011; Goldman & Tauritz, 2012; Goldman & Punch, 2014), BBHC (Iclanzan & Dumitrescu, 2007), MACRO (Mills, 2010) and compact grammatical models related to this thesis (Cox & Watson, 2014a,b).

The SG search algorithm implements MSS by adopting inferred schema structure as a variation neighbourhood. The performance of the algorithm is evaluated on a range of NP-hard problems containing modular, hierarchical modular and randomised overlapping structures. Two algorithms that are comparable in a MSS framework are also tested: LTGA with global mixing and a hybrid GA. The latter is not strictly an MSS search. However, it is closely related as it combines crossover-based variation and hill climbing in an iterated population-based search (Section 2.2.5 contains more details on hybrid algorithms). The results in this section show that on the tested problems, SG significantly outperforms the hybrid GA in terms of time complexity, and is comparable to the state-of-the-art method. This establishes SG as a credible alternative to leading

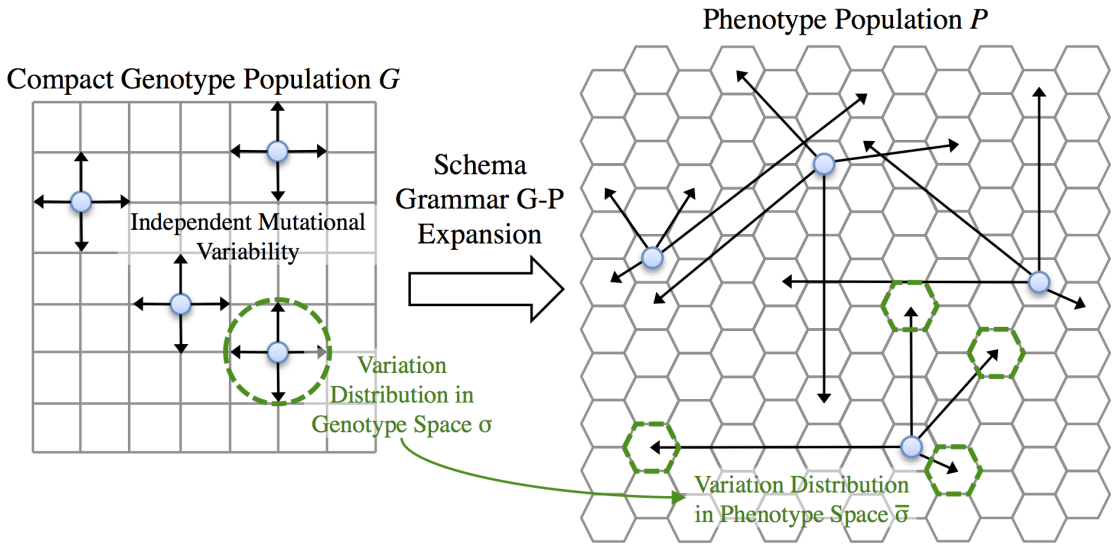

FIGURE 5.1: An illustration of the SG exploration distribution. A population compressed by SG is defined in a factorised symbolic representation. SG expands each genotype into the "language" of phenotype space, which follows its own structure. Independent mutational variability in genotype space corresponds to directed mutation variability in phenotype space.

model-building optimisation algorithms. It also demonstrates for the first time the potential of compact models for unbiased BBO, and provides additional support for the claim that SG can capture the information structure of an evolutionary population.

5.1 Inferring a Language of Variation

In what follows we adopt the general notation and concepts introduced by Toussaint (Toussaint, 2006; Toussaint & von Seelen, 2007). This work provides a useful framework for describing variation in the context of compact models. The variation neighbourhood of a population at any point in time can be expressed as a probability distribution over phenotype space, P, which we denote using $\bar{\sigma}(p)$. In any given evolutionary generation, $\bar{\sigma}(p)$ gives the probability that variation will produce a mutant offspring $p \in P$. Ultimately, we will characterise *effective* fitness using the *objective function* fitness of offspring sampled from this distribution.

In an important sense, evolutionary algorithms differentiate themselves by the way they construct $\bar{\sigma}$. The approach typically adopted by model-building algorithms is to infer a model of fit phenotypes in the current generation, then use this model to generate offspring phenotypes in the next generation. This approach addresses phenotype space directly, obviating the need for a genotype to phenotype (G-P) map[1]. These "direct EDAs" can be described as performing "complex adaptation on an arbitrary representation". An alternative approach is to perform "simple adaptation on a suitable

[1]Strictly, it implies an injective 1:1 mapping.

representation". If the dependencies of phenotype space can be captured in a G-P map, and genotypes factorised using this map, then independent mutational variability in genotype space can correspond to directed mutation variability in phenotype space. Toussaint describes this idea as an "indirect EDA", and provides theoretical results that suggest compact models are optimal in this context. The indirect approach can be formalised by expressing the variation in terms of mutant offspring in genotype space. Let $\sigma(g)$ be the exploration distribution over genotype space G, and $\phi : g \to p$ be a surjective G-P mapping. Phenotypic variation can now be described in terms of genotypic variation:

$$\bar{\sigma}(p) = \sum_{g \in [p]} \sigma(g) \tag{5.1}$$

Where $[p] = \phi^{-1}(p)$ is the *neutral set* of all genotypes from which p can be derived. Figure 5.1 illustrates this idea in the context of SG. The left of the figure shows a compact SG instance inferred from a population of phenotypes, which are shown to the right of the figure. The "language" of phenotypic structure is factorised into the schema symbol structure of the grammar (the G-P map), leaving an information-less encoding in genotype space. Random variation in genotype space is mapped by the grammar onto directed variation in phenotype space.

A practical question remains regarding how to actually invoke random mutation in factorised genotype space. The approach we take here is to randomly select a schema symbol from the grammar and use it as a mutation operator. This set of symbols typically includes smaller building blocks which often form the base of the grammar's symbol hierarchy, and larger composite building blocks which appear in the top-level compressed representation. We create mutant offspring by reproducing a genotype symbol randomly selected from the population and expanding it into a fully-formed phenotype. Then, a mutation schema symbol is selected from the grammar's symbol table at random and expanded into the same phenotype, overwriting any existing alleles. As an example, consider a parent genotype $g \overset{*}{\Rightarrow} 11110000$ and a mutation schema symbol $s \overset{*}{\Rightarrow} ****1111$. The generated offspring g' would yield $g' \overset{*}{\Rightarrow} 11111111$. This is similar to the idea of over-specification in the Messy GA (Goldberg et al., 1989). Intuitively this kind of variation has the opportunity to produce novel offspring that would be *well compressed* by the grammar, which, as shown in Section 6.2, is analogous to the grammar classifying the offspring as fit.

The concept of effective fitness is hard to define and interpret across evolutionary time, although it is certainly possible (Stephens & Vargas, 2000). Here, we choose to examine effective fitness in a single evolutionary generation, allowing the characteristics of each exploration distribution to be easily visualised and intuitively understood. We use populations of fit individuals as parents, the majority of which are locally optimal

with respect to the bit-flip fitness landscape. This particular stage of evolution is examined as it arguably presents the most challenging time for evolutionary variation: when simple micro-mutation has exhausted its possibilities and more radical recombination is required. Performance across evolutionary time is assessed separately, by measuring function evaluations in global function optimisation tasks. These experiments and results are described in Section 5.2.

5.1.1 Methods

In this section we describe an experiment to evaluate the effective fitness of SG and its ability to generalise. Two recombinative variation methods are used as comparisons: uniform crossover and LTGA global mixing (see also Section 2.2.5.1). Both of these methods are not sensitive to positional bias, so can be seen to occupy the same category of BBO. Both can also be used in MSS-style search, which allows them to be compared with SG in Section 5.2. Uniform crossover is chosen as it represents the traditional approach to unbiased GA recombination. Both random single-bit mutation and a combination of uniform crossover and mutation were also tested, however neither of these methods performed better than uniform crossover at this stage of evolution. LTGA global mixing is chosen as it arguably represents the state of the art in evolutionary model building, and has proven to be effective at modelling and solving the classes of problems tested here (Thierens, 2010; Pelikan et al., 2011; Thierens & Bosman, 2011, 2013; Goldman & Punch, 2014).

In the experiment 500 shuffled instances of four test problems classes are generated: $n = 64$ $k = 3$ modular traps, $n = 64$ 6-level HIFF problems, $n = 64$ 8×8 2d spin-glasses and $n = 64$ $k = 3$ NK-fitness landscapes. Each of these problems is described and defined in Chapter 4. On each problem instance a sample population of 100 unique parents with above-average fitness is created using stochastic bit flip hill climbing. Each parent is initialised to a random bit string. A random permutation of the n loci is then iterated and for each locus the allele value is flipped. If the flip creates a strict fitness improvement then it is kept, otherwise it is discarded. Any parents already represented in the population are discarded and the process is repeated. 1000 mutant offspring are then generated for each method using the processes described in the paragraphs that follow. Offspring overlapping with the parent population are removed, as are any offspring with a fitness lower than the lowest fitness parent. This leaves a population of offspring that is both novel and fit. The size of this population is recorded for each method, as well as the distribution of offspring fitnesses. Results are aggregated across the 500 instances for each of the problem classes.

Each sample population is compressed using SG and the offline algorithm. 1000 offspring are created by cloning randomly-chosen members of the parent population, then expanding a randomly chosen grammar schema symbol into the individual, overwriting

any existing allele values. Duplicates and offspring already represented by the sample population are discarded, leaving novel offspring only. The fitness values of the mutants are then calculated and recorded.

For the uniform crossover neighbourhood, to generate each offspring two parents are chosen at random from the sample population. To create the offspring, the first parent is cloned and alleles from the second parent crossed over with a probability of 0.5. As before, any duplicates and mutants already in the sample population are discarded.

To create each LTGA variation neighbourhood the population is modelled using pairwise distance clustering (see Section 2.2.5.1). This creates a tree of $2n - 2$ subsets for each problem instance. The order 1 subsets are discarded, as order-1 mutations are used to generate the sample population. To create each mutation a parent and a donor individual are selected at random from the sample population, as well as a random subset from the linkage tree. The parent individual is cloned, then crossover performed from the donor to the mutant using the mask specified by the subset. This operation is equivalent to a single step of the "global mixing" crossover strategy. Essentially, LTGA creates a mutation operator dynamically by combining a linkage pattern from the FOS with an allele pattern randomly sampled from the population. The maximum number of unique mutations is bounded by $2p(n-1)$, where p is the size of the population and n is the problem, which is of the same order as SG (Theorem 3.1). As before, any duplicates and offspring already represented by the sample population are discarded.

To allow comparison across multiple trials and for consistency of presentation across problem types, fitness values are normalised using the following function, which is generated for each sample population:

$$f'(p) = \frac{f(p) - \min(F_{pop})}{\max(F_{pop}) - \min(F_{pop})} \tag{5.2}$$

Where $f(p)$ is the original objective function fitness and F_{pop} represents the fitnesses of the sample population. Scaling in this way means offspring with fitness equal to the least fit member of the sample population have a fitness of 0, and offspring that are fitter than any member of the sample population have a fitness > 1.

5.1.2 Results

The results for each of the problem types are shown in Figures 5.2, 5.3, 5.4 and 5.5. In summary, the results show that random variation on the compact SG representation was able to consistently generate far more and far fitter novel phenotypes than crossover-based recombination. In many cases SG was able to combine decomposed problem features of many orders to create novel offspring, whereas uniform crossover

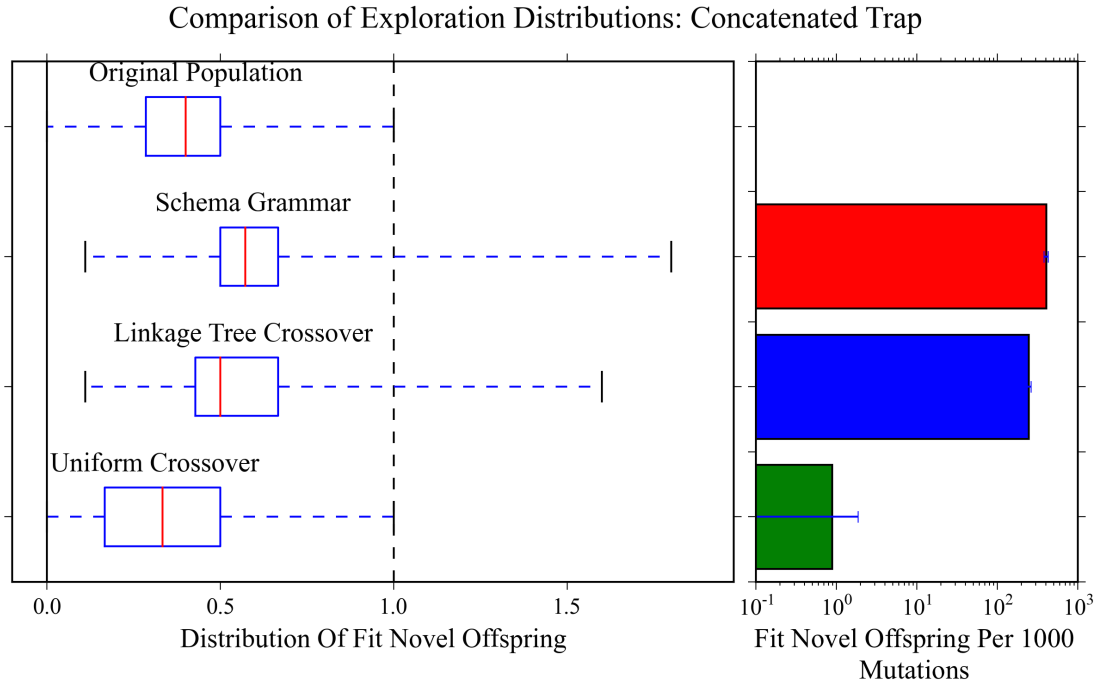

FIGURE 5.2: Comparison of exploration distributions for the modular trap function, measured over 500 random trials. (left) The fitness distribution of novel offspring generated by each method. Fitness was scaled such that 0 represents the least fit parent and 1 represents the most fit parent. The whiskers on each box plot extend to the minima and maxima. (right) The number of novel fit mutants generated by each method, shown on a log scale. In the experiment SG was able to combine building block configurations to generate fit offspring, significantly outperforming unguided crossover. Unguided uniform crossover typically destroyed the modular configuration of the parents and on average produced only 1 fit offspring per instance. In comparison to LTGA, SG created the most viable offspring and had the highest median and maximum offspring fitness.

was restricted to "hopeful monsters" that often violated the constraints of phenotype space. This was particularly noticeable on the concatenated trap problem, where compositional schema structure follows a strict modular pattern. In this case uniform crossover performed poorly as there was a very high chance of modular configurations being corrupted, creating on average less than one fit offspring in 1000 attempts. In contrast, SG was able to combine inferred building block configurations to create large numbers of novel offspring, some of which were significantly fitter than any member of the parent population.

The comparative advantage of the SG variation distribution over uniform crossover held across all of the problems, including spin glasses and NK landscapes, which have an irregular, randomised structure. If we accept that fit offspring in this experiment (offspring with an objective function fitness greater than or equal to the least fit member of the parent population) are analogous to *viable* offspring, then the results show the dramatic impact that representations can have on the viability of random recombination. This is summarised in Table 5.1.

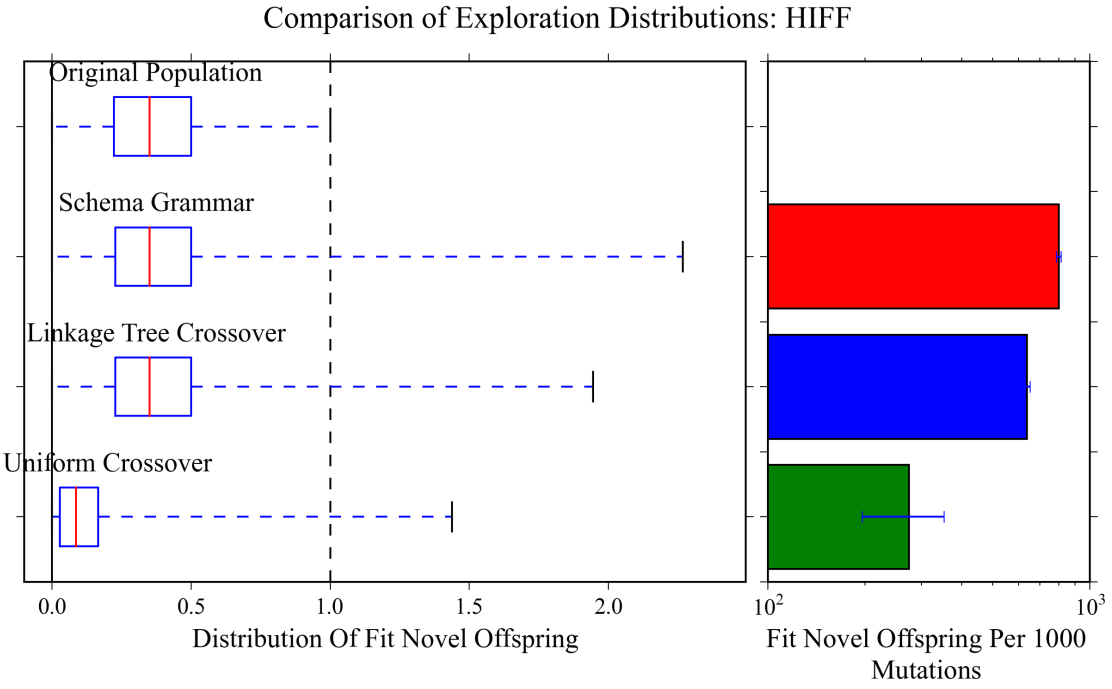

FIGURE 5.3: Comparison of exploration distributions for the HIFF function. In the experiment, SG was able to recombine the problem's building block structure to find fit new regions of the search space, performing significantly better than the unguided method. The chances of uniform crossover destroying modular configurations were less than the deceptive traps case, and the method was able to generate some offspring that were fitter than the fittest parent. SG offspring had a similar interquartile range to LTGA, although created more fit offspring and the offspring with the highest fitness.

Problem	Proportion of Viable Mutant Offspring	
	Uniform Crossover	SG Macro Mutation
Concatenated Traps	0.89%	40.03%
HIFF	26.28%	81.07%
2D Spin Glasses	19.42%	80.29%
Nearest-Neighbour NK	20.93%	79.44%

TABLE 5.1: The viability of mutant offspring generated using random recombination (uniform crossover), compared to random variation using the compact SG representation.

One possible objection to the comparison between the effective fitness of SG and uniform crossover, is that the latter is sometimes only effective when followed with a bit-flip hill-climbing phase, allow a mixture of recombination and repair. This is the idea behind hybrid GAs, which have been shown to be effective in a number of search tasks (Prugel-Bennett, 2007; Pelikan, 2010; Prugel-Bennett, 2010). The exploration distribution of uniform crossover on its own will not capture the potential benefit of the hybrid variation method. Although this is the case, it is also true that SG-facilitated macro mutation combined with bit-flip micro mutation might also produce better results than SG on its own. In Section 5.2, the search algorithms incorporate bit-flip moves into all of the tested methods, allowing a comparison between hybrid GAs, SG and LTGA over evolutionary

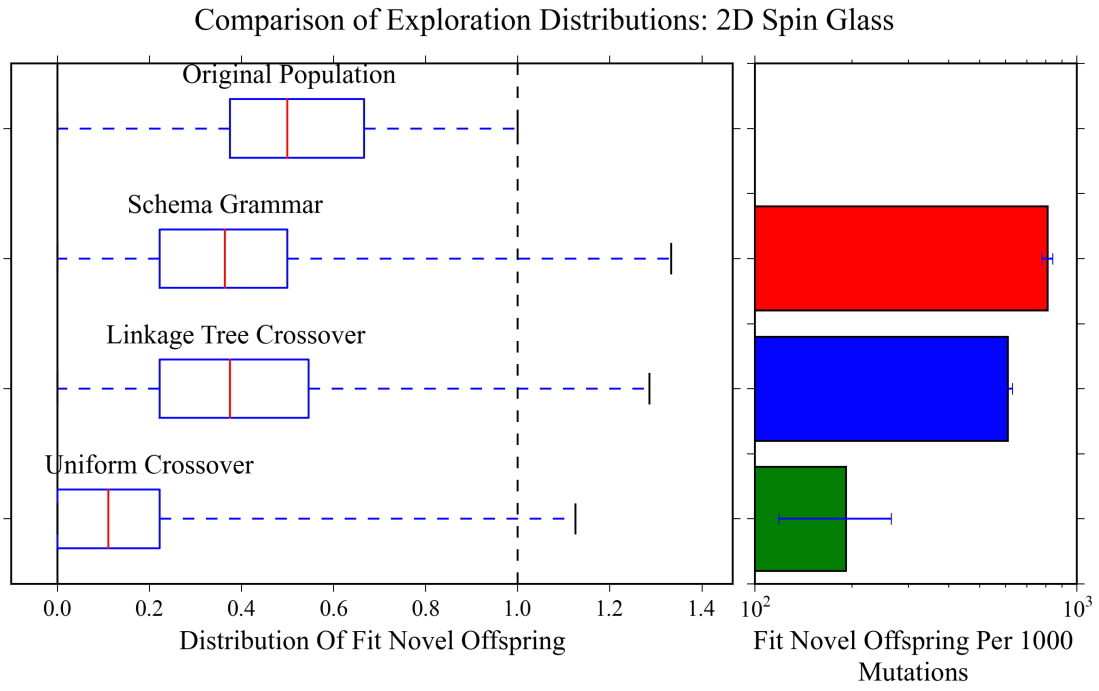

FIGURE 5.4: Comparison of exploration distributions for 2D spin glass problems. In this case problem structure was randomised and irregular. Nonetheless, SG was able to produce far more and far fitter offspring than unguided recombination. The fitness distributions of SG and LTGA offspring were similar, with both creating offspring fitter than the fittest member of the original population. LTGA had a slighter fitter interquartile range, although SG created the most fit offspring overall.

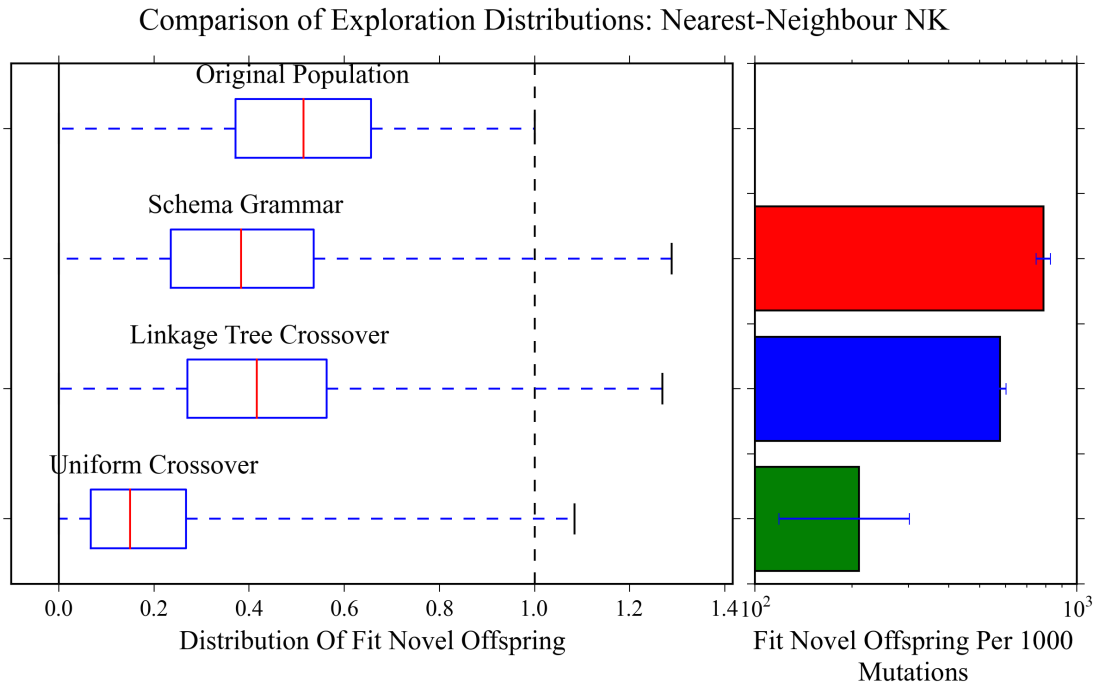

FIGURE 5.5: Comparison of exploration distributions neighbourhoods for nearest-neighbour NK landscape problems. The results on this randomised problem were very similar to the spin glass case, with both model-building methods offering a clear advantage over unguided uniform crossover.

time. The results are shown to be consistent with the comparative result here, confirming that the variation neighbourhood offered by the compact SG representation offers a far more powerful exploration distribution than unguided variation.

The results show that the performance of SG was comparable if not superior to the state of the art model-building method, LTGA. The offspring of both methods had fitness in a similar interquartile range, although SG consistently produced the fittest individual offspring. The size of the SG exploration distribution, in terms of the number of viable offspring it was able to generate, was also significantly larger than LTGA in all cases. The most likely reason for this is that LTGA does not support the modelling of overlapping subsets that are not strictly nested. This means that any fit schemata that occur within these subsets cannot be inferred or exploited by LTGA, limiting the size of the effective neighbourhood. The results suggest that this did not negatively effect the fitness of the neighbourhoods that LTGA *was* able to exploit, which were significantly larger than uniform crossover in all cases. However, in some instances it may have implications for search, and represents a comparative disadvantage in exploratory power.

5.2 Global Function Optimisation

In this section the compact SG model is incorporated into an evolutionary algorithm and applied to global function optimisation tasks. This provides an empirical test of the effective fitness of SG-facilitated variation, and a comparative assessment of practical problem-solving performance. The algorithm is inspired by the concept of Compression EAs proposed by Toussaint (2006). The idea of Compression EAs is to replace the approach of traditional EDAs – complex model-building and sampling in phenotype space – with simple model-building and sampling in a compact, factorised genotype space. Toussaint demonstrated the concept in his work, but until now the idea has been limited by the constraints of sequential compression. Compressing phenotypes with sequential methods can only capture information that is observable in contiguous patterns on sequential representations. It cannot capture functional context, and it cannot capture combinatorial patterns that are not observable sequentially. This renders the idea ineffective for unbiased BBO. As we have seen, SG overcomes these constraints and can compress data with no positional bias. This advance enables Compression EAs to be re-examined as a contemporary optimisation method.

We adapt the general solving framework of Compression EAs to accommodate recent advances in evolutionary model-building. Instead of building and sampling a simple probabilistic model over compressed genotype space, essentially performing recombination using multiple random macro mutations, we use an MSS-based search method. This uses the SG symbol structure as an indirect variation neighbourhood in a stochastic hill-climbing process. As discussed in the introduction to this chapter, this benefits from

the efficiency of systematic trial-and-error search, combined with the opportunity for large scale recombination. It also enables a meaningful comparison between SG and the state of the art in model building, LTGA global mixing, which follows the same basic approach.

A comparison with unguided variation in a contemporary context is also possible. Recent research has shown that hybrid methods than combine recombination and bit-flip hill climbing are often more effective than recombination alone. This approach benefits from the exploratory power of radical recombination combined with local optimisation and repair. It has been proven effective using uniform crossover (Prugel-Bennett, 2007, 2010) and a variety of EDA-based recombination methods (Pelikan & Goldberg, 2003; Pelikan, 2008; Pelikan et al., 2009; Pelikan, 2010). The term *hybrid evolutionary algorithms* has been suggested to represent the idea (Pelikan, 2010). Combining implicit or explicit modelling with hill-climbing appears closely related to MSS (Mills et al., 2014). For the purposes of this thesis, the commonality between the methods allows a meaningful and contemporary comparison between SG and uniform crossover.

The experiments described in this section are performed on the previously introduced test problems. Performance is measured in terms of function evaluations until the global optimum of each problem is found. This is evaluated across a range of problem sizes, allowing time complexity to be estimated.

5.2.1 Search Framework

The overall search framework is shown in Figure 5.6. The framework is based around the functional overlap between MSS, LTGA global mixing and Hybrid EA methods. The framework defines an iterative, population based evolutionary search process. Variation is performed using one of two different approaches. In the Hybrid GA approach offspring are created by replicating and recombining parents, then repairing the offspring with a local search step. In the MSS approach offspring are created by modelling the population and performing stochastic multi-scale hill-climbing, using the parent population as initial conditions. Offspring are integrated back into the population using a replacement strategy (described in Section 5.2.2). This search process continues until a global optimum is found or no more improvements are made in a single evolutionary generation. The selective environment is manifested in two ways. Firstly, both bit-flip and multi-scale hill-climbing incorporate selection by accepting mutations that improve the objective function fitness of the individual (strictly in the bit-flip case). Secondly, the replacement process implements survivor selection using both objective function fitness and niching.

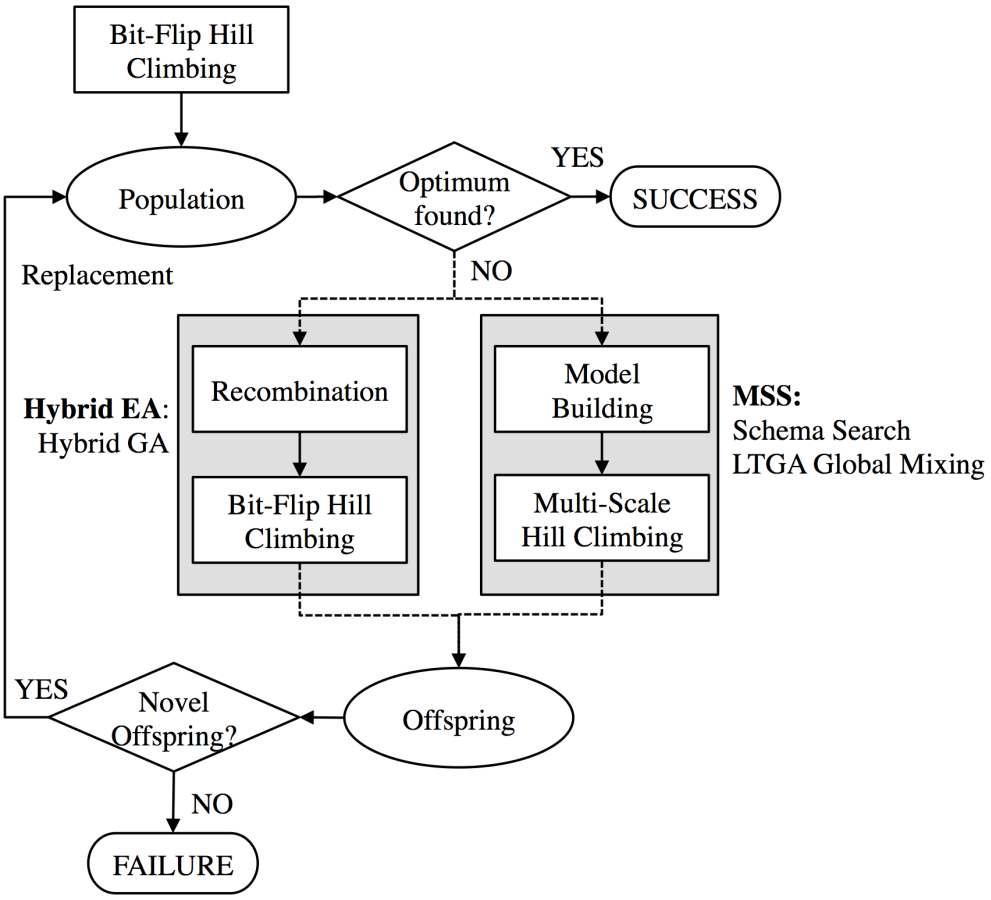

FIGURE 5.6: The search framework used for global function optimisation. The framework allows both Hybrid EAs and MSS algorithms to be compared in a unified evolutionary search process. See text for details.

Each of the components of the framework was developed using a combination of best practice from recent literature and experimentation. The pseudocode for the overall algorithm incorporating SG compression, which we denote using the term *Schema Search*, is shown in Algorithm 3. Schema Search uses the offline compression algorithm defined in Algorithm 2 for inferring the compact representation. The implementations of both Schema Search and LTGA global mixing have some key features that are worth highlighting. In both cases the multi-scale hill climbing step allows both bit flip mutations and larger macro mutations. For SG this is enabled by including both terminal symbols and schema symbols in the variation neighbourhood; for LTGA it is enabled by including the single locus subsets in the linkage tree. So, bit-flip mutations are attempted first, followed by any order-2 mutations and so forth. The multi-scale hill climb also enumerates schemata / subsets in reverse size order. This was found to produce marginally better performance on some problems and is consistent with recent approaches to LTGA (Goldman & Tauritz, 2012; Goldman & Punch, 2014). The niching strategy used for both methods is to ensure that the population only contains unique individuals. This stops premature convergence and was found to produce better performance than alternative strategies, including restricted tournament replacement (Pelikan & Goldberg, 2001). It

is also consistent with recent work that adopts the same strategy for LTGA (Goldman & Punch, 2014).

5.2.2 Methods

A stochastic "greedy ascent" hill climb is adopted for bit-flip hill climbing. When being used to initialise the population, random bit strings are used as initial conditions. When used as a repair step in the Hybrid GA, recombined offspring are used as initial conditions. A random permutation of the n loci is iterated and for each locus the allele value is flipped. If the flip creates a strict fitness improvement, it is kept and the process is restarted. Otherwise, it is discarded. This continues until no more improvements are made. This process generates a local optimum in $O(n \log n)$ steps (Mühlenbein, 1992).

On each iteration of Schema Search the population is modelled using the offline algorithm. Then, offspring are generated by performing *indirect* mutational variation over the population. The union of grammar schema symbols and terminal symbols create a dictionary of mutation symbols. This encompasses both single-bit mutations and directed macro mutations at many different levels of scale. The multi-scale hill-climbing step is run for each member of the parent population. For each parent an offspring is initialised by replicating the parent phenotype. This is more efficient than replicating and then expanding the inferred genotype symbol. Then, a random selection of m symbols is drawn from the mutation dictionary. In early experiments it was found that $m = O(n)$ produced the best results; we therefore set $m = 2n - 2$ to create consistency with LTGA global mixing. The mutation symbols are applied in reverse order of size. If a mutation produces equal or improved objective function fitness it is kept, otherwise it is reversed. If a mutation has no effect on the phenotype then a fitness evaluation is not performed. Each new mutant created is added to a population of offspring. The replacement strategy uses truncation selection combined with niching. A set of unique individuals is formed from the union of offspring and the original population, discarding any duplicates. The top p individuals are selected and the remainder are discarded. If no new offspring are included in the selected population then search terminates in a failure.

LTGA is performed in a very similar way to Schema Search, with the only differences in the variation step. On each iteration the population is modelled into a linkage tree using the pairwise distance clustering (see Section 2.2.5.1). This creates a tree of $2n - 2$ subsets for each problem instance. Then, offspring are generated using *direct* mutational variation. For each offspring a random permutation of the $2n - 2$ subsets is enumerated in size order. For each subset a random donor phenotype is selected from the population. The allele values specified by the subset are sampled from the donor and then applied to the offspring as a mutation in the same way as Schema Search. This process is equivalent to "smallest first" global mixing (Goldman & Tauritz, 2012).

Algorithm 3: Pseudocode for the Schema Search Algorithm

input : population size p
input : objective function $f : \{0,1\}^n \to \mathbb{R}$
input : global optimum fitness f_{MAX}
input : number of multi-scale hill-climbing steps m
output: global optimum, or $\emptyset$ if not found

```
ℙ = ∅ // initialise population
while |ℙ| < p do
    // use random local search to seed the population
    ℙ = ℙ ∪ {HillClimb(f)}
end
// main search loop
while max_{P∈ℙ} f(P) < f_MAX do
    // infer a compact representation for the population
    // V = non-terminal symbols, G = factorised genotypes
    // S = schema symbols, Σ = terminals, R = production rules
    {V, G, S, Σ, R} = Offline(ℙ) // offline SG compression, see Algorithm 2
    ℚ = ∅ // initialise set of offspring
    // initialise the set of all possible mutations: the union of
       terminal (bit-flip) and schema symbols
    Λ = S ∪ Σ
    // stochastic multi-scale hill-climbing
    for P ∈ ℙ do
        // initialise mutant from phenotype
        // we don't need to re-expand from genotypes
        O = P
        // randomly select neighbourhood of m symbols
        ℕ = RandomSelection(Λ, m)
        // enumerate neighbourhood smallest-first
        for λ ∈ SortByOrder(ℕ) do
            // recursively expand symbol into offspring using rules
            M = ExpandInto(O, λ, R)
            if M ≠ O then
                if f(M) ≥ f(O) then
                    O = M
                end
            end
        end
        // add unique offspring
        ℚ = ℚ ∪ {O}
    end
    // truncation replacement: select p fittest unique individuals
    ℙ_NEW = Fittest(ℙ ∪ ℚ, p)
    // exit if no new offspring
    if ℙ_NEW ∩ ℙ = ℙ then
        return ∅
    end
    ℙ = ℙ_NEW
end
return argmax_{P∈ℙ} f(P)
```

The Hybrid GA uses uniform crossover for recombination. In each generation p offspring are created. For each offspring two parents are selected at random from the population. The first is reproduced and the allele values of the second parent are crossed over with a probability of $1/2$. A repair step is then run using bit-flip hill climbing. A small performance improvement can be obtained by limiting the number of function evaluations in this repair step. Therefore, we limit the repair to $2n - 2$ evaluations to create consistency across all 3 variation methods. The best replacement strategy found was a truncation selection scheme. The offspring are pooled with the parents and the most fit p individuals are retained. Note that this is different from the replacement strategy used by Schema Search and LTGA, as duplicates are allowed. For the problems tested, this is found to produce better performance. It is also found to produce better performance than both Boltzmann selection (Prugel-Bennett, 2007, 2010) and restricted tournament replacement (Pelikan & Goldberg, 2001).

Each problem type is tested for a range of different problem sizes. Populations are statically sized using bisection for each solving method, requiring 19 out of 20 runs to find the global optimum in under 10^6 function evaluations. The Hybrid GA is not able to satisfy these criteria for some larger problem sizes. Therefore, experimental results for these configurations are not recorded.

400 instances of each problem type and size (random in the case of spin glasses and NK landscapes) are generated. Global optimisation is then performed using Schema Search and the two comparison methods and the results recorded.

5.2.3 Results

The results for each of the different problem classes will be presented and discussed in turn. A full summary of the results, including population sizes, fitness evaluations and estimated scalability for each method is provided in Table 5.2.

The results for concatenated traps are shown in Figure 5.7. Given the results earlier in this chapter examining the exploration distribution of SG, and the results in the previous chapter showing the hierarchical schema structure it can infer, it is no surprise that it can solve this additively decomposable problem so effectively. By using an MSS style search, once the problem structure is decomposed then the problem can typically be solved trivially in a small number of attempts. Across all the problem sizes, the average number of generations used by Schema Search to solve the problem was 1.64, and 2.26 for LTGA. The better absolute performance of LTGA on this problem is predominately explained by a consistently lower population requirement. The nature of the lossless compression algorithm is such that each module configuration (e.g., ***111***) must be present in at least two separate contexts for it to be inferred as a schema symbol

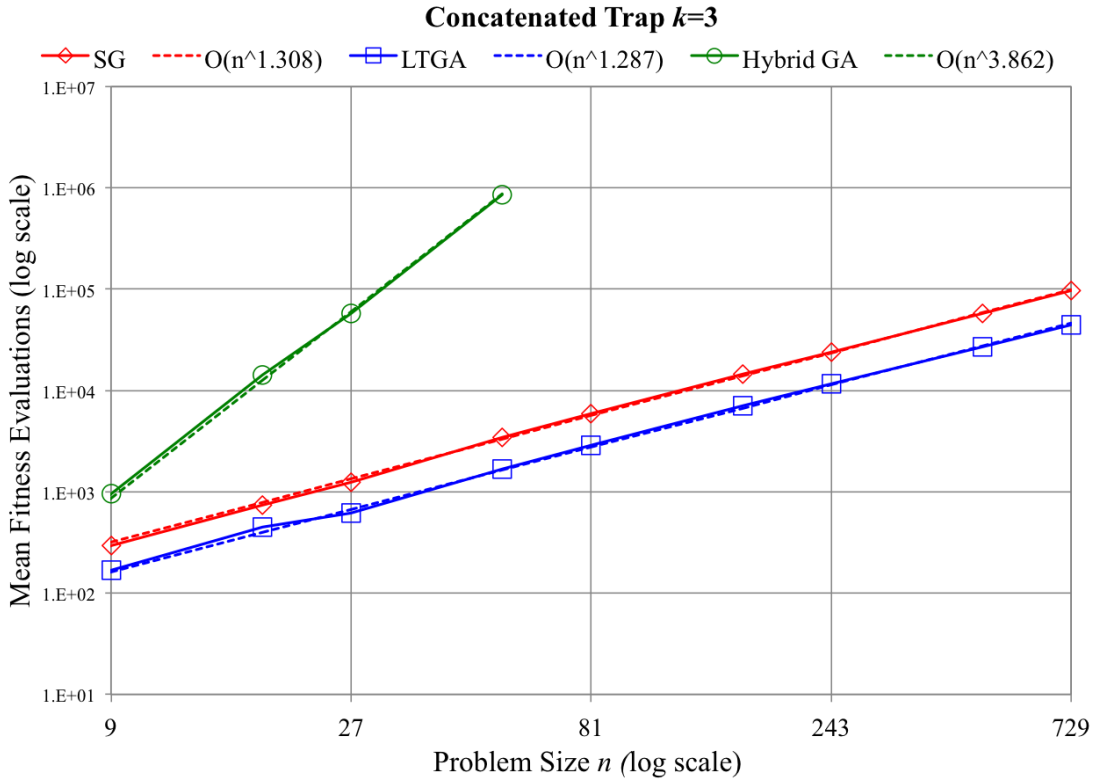

FIGURE 5.7: Global function optimisation results for the concatenated trap problem, shown on a log-log scale. Mean fitness evaluations are shown for Schema Search, LTGA global mixing and the Hybrid GA. A regression line is fitted for each method. The results show both SG and LTGA achieved similar sub-quadratic scaling in n. In comparison, the polynomial scaling in the uniform crossover method was much higher order. Experiments were terminated above $n = 54$ as the method could not reliably succeed in under 10^6 evaluations.

(otherwise it would not offer any value for compression). In contrast, LTGA only models linkage structure, which is present in much smaller populations. In order for each module configuration to be sampled only one instance needs to be present in the population. The Hybrid GA using uniform crossover appeared to achieve polynomial scaling without any positional bias in the representation, although it was significantly outperformed by the two explicit model-building methods. As expected, building blocks were frequently corrupted by recombination. However, the repair step restored them to fully formed module configurations. This provided a kind of "two-scale search", combining large neighbourhood variation through crossover and single bit neighbourhood variation through hill climbing. In the problem sizes tested, the average number of generations required by the Hybrid GA to solve the problem was 18.33.

Figure 5.8 gives the results for the HIFF problem. The comparative results in this case were similar to concatenated traps, suggesting that both modelling methods were able to exploit knowledge of problem structure to solve the problem efficiently. An analysis of the average number of generations taken to solve the problem shows this was approximately logarithmic in n for both modelling methods. This is consistent with the

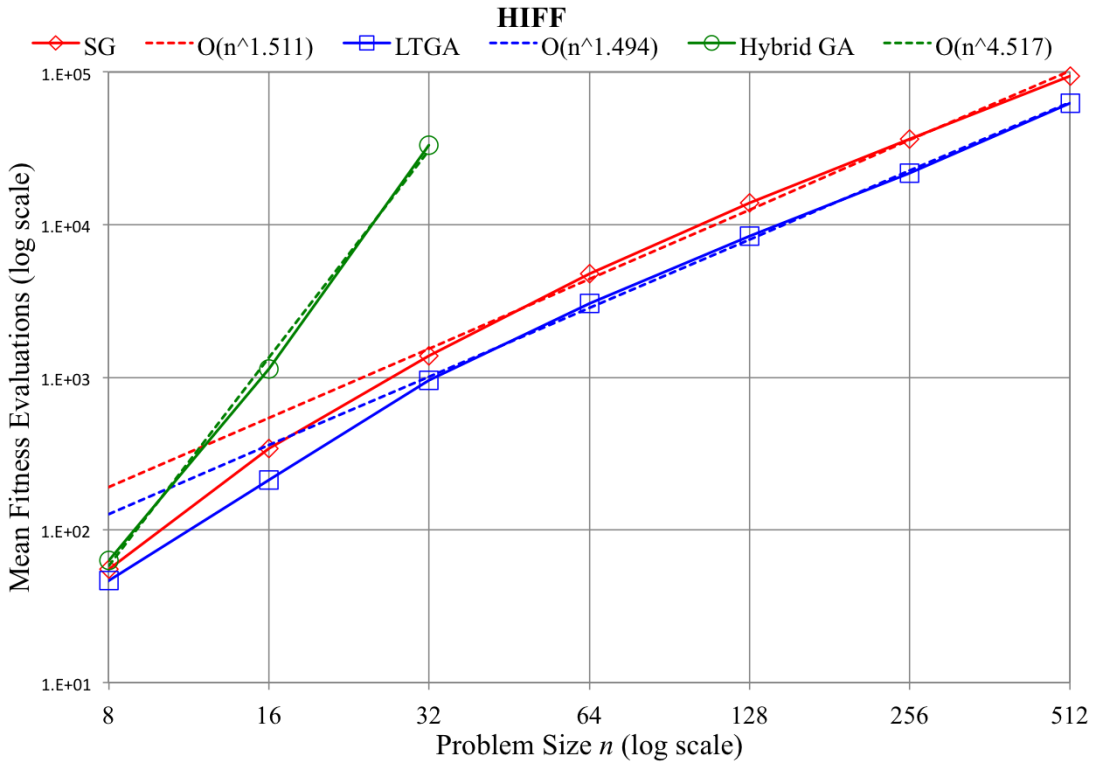

FIGURE 5.8: Global function optimisation results for the HIFF problem, shown on a log-log scale. SG and LTGA achieved very similar sub-quadratic scaling in n, whilst scaling of the Hybrid GA was much higher order, with experiments terminated above $n = 32$.

intuition that the hierarchical levels in the problem need to be solved recursively, with the variation neighbourhood size doubling on each recursion. Once again, the Hybrid GA appeared to achieve polynomial scaling, however it was significantly outperformed by the two other methods.

The results for the 2D spin glass problem are shown in Figure 5.9. Problem instances in this class contain overlapping, randomised structure, which is expected to emerge in spatially contiguous neighbourhoods. The results show that both modelling methods were able to achieve efficient, low-order polynomial scaling that was significantly superior to the Hybrid GA. They also suggest some advantage to the SG model over LTGA, with slightly better scaling in larger problem sizes. This result might be expected from the fact that the SG model is able to infer un-nested overlapping structure, whereas LTGA can only model strictly nested overlap. Performance of the Hybrid GA was once again a higher-order polynomial in n, and experiments were terminated above $n = 100$.

The results for the nearest-neighbour NK landscape are shown in Figure 5.10. One significant difference on this problem was that the Hybrid GA was not able to achieve polynomial scaling, with an exponential rise in fitness evaluations seen above $n = 30$. This is likely to be a reflection of the large amount of epistasis in the problem, compared to spin glasses, and the inability for uniform crossover to implicitly model the correct

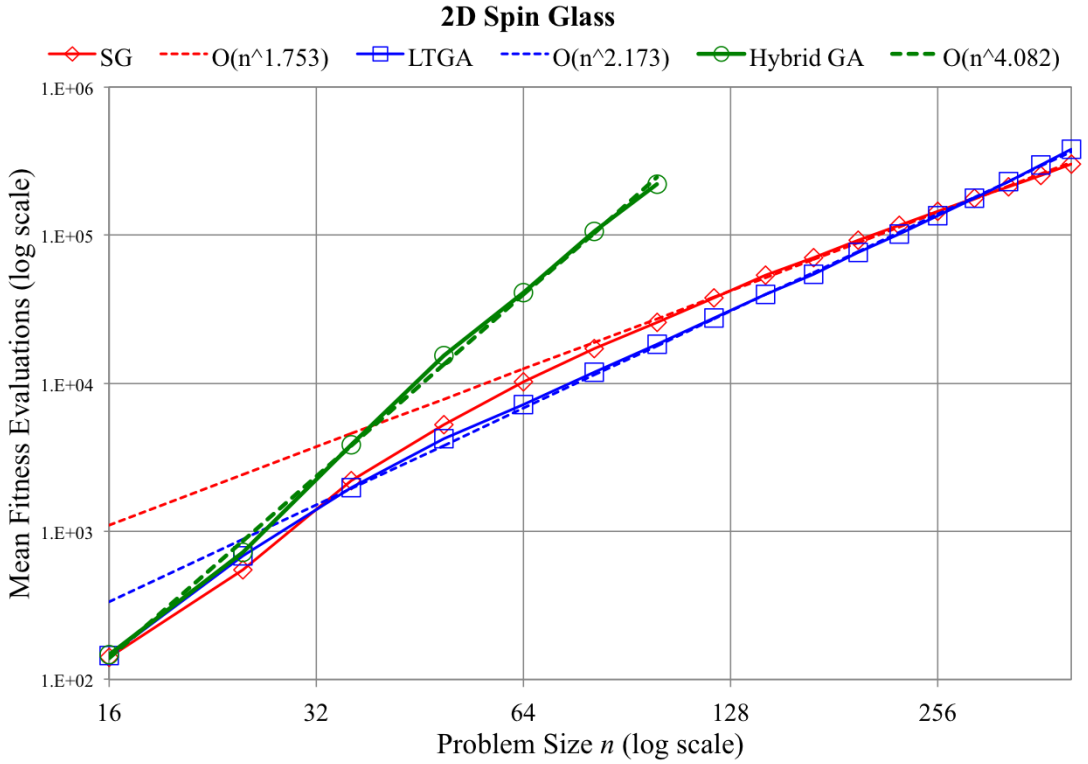

FIGURE 5.9: Global function optimisation results for the 2D spin glass problem. Problem instances in this class contain overlapping, randomised structure. SG and LTGA achieved similar low-order polynomial scaling. SG appeared to have the advantage overall, with better scalability in larger problem sizes. The Hybrid GA also achieved polynomial scaling, however this was much higher order and experiments were terminated above $n = 100$.

structures. Table 5.2 shows that on this problem, the absolute performance of SG was superior to LTGA and required significantly smaller population sizes. The approximate scaling of SG was also slightly better. Combined with the results on the 2D spin glasses, this starts to suggest that as problem complexity increases then the more flexible schema structure offered by SG may offer an advantage over LTGA. Proving whether or not this is the case is beyond the scope of this thesis, as we only intend to show that the compact SG model offers a credible alternative to the state of the art. Nevertheless, an important direction for future work is identifying the problem classes where SG offers a clear advantage over the alternatives and vice versa.

5.3 Discussion

The results in Section 5.1 demonstrate that by inferring the compositional schema structure from fit populations, SG models are able to generalise about the selective environment, creating an exploration distribution that not only respects the dependencies of phenotype space, but facilitates large scale compositional variation without resorting to "hopeful monsters". The exploration distribution is implemented indirectly, using

Problem Class	n	Population Size			Mean Fitness Evaluations (s.d.)		
		SG	LTGA	HyGA	SG	LTGA	Hybrid GA
Concatenated Traps $k = 3$	9	7	6	34	295 (193)	168 (99)	956 (846)
	18	17	13	59	746 (188)	449 (91)	14,318 (3,760)
	27	20	14	85	1,236 (324)	616 (71)	57,320 (7,529)
	54	27	18	212	3,456 (452)	1,674 (137)	863,032 (55,232)
	81	30	20		5,877 (669)	2,901 (185)	
	162	37	24		14,514 (1,161)	7,107 (234)	
	243	40	26		23,894 (2,027)	11,644 (417)	
	486	47	30		57,286 (4,765)	27,141 (1,106)	
	729	50	32		96,624 (12,425)	44,158 (2,496)	
	Estimated scalability				$\Theta(n^{1.308})$	$\Theta(n^{1.287})$	$\Theta(n^{3.862})$
HIFF	8	6	6	6	55 (35)	47 (27)	64 (57)
	16	12	9	21	340 (133)	214 (56)	1,137 (732)
	32	16	16	105	1,376 (366)	952 (149)	33,309 (10,199)
	64	21	21		4,783 (837)	3,040 (276)	
	128	25	26		13,884 (1,936)	8,387 (388)	
	256	27	32		36,280 (4,251)	21,908 (609)	
	512	30	44		93,966 (8,386)	62,950 (1,123)	
	Estimated scalability				$\Theta(n^{1.511})$	$\Theta(n^{1.494})$	$\Theta(n^{4.517})$
2D Spin Glass	16	8	11	8	140 (108)	145 (127)	146 (136)
	25	11	16	12	547 (376)	687 (482)	720 (588)
	36	20	19	24	2,206 (1,003)	1,970 (886)	3,839 (2,807)
	49	27	21	40	5,278 (1,616)	4,218 (943)	15,452 (8,075)
	64	34	24	58	10,189 (2,541)	7,180 (1,217)	41,121 (18,514)
	81	40	28	79	17,173 (3,111)	11,883 (1,490)	106,634 (40,180)
	100	45	32	104	25,868 (3,730)	18,355 (2,208)	221,003 (63,857)
	121	50	37		37,585 (5,500)	27,714 (2,662)	
	144	54	42		53,483 (8,668)	39,954 (3,597)	
	169	58	47		70,145 (9,616)	54,360 (4,071)	
	196	62	54		92,058 (11,198)	76,423 (5,889)	
	225	65	61		115,766 (15,193)	101,626 (7,306)	
	256	69	69		144,906 (16,878)	136,196 (9,108)	
	289	72	77		178,288 (20,529)	178,259 (11,056)	
	324	74	87		211,370 (21,039)	231,428 (13,255)	
	361	77	97		253,659 (25,467)	296,109 (14,402)	
	400	80	109		300,364 (27,293)	378,502 (16,121)	
	Estimated scalability				$\Theta(n^{1.753})$	$\Theta(n^{2.173})$	$\Theta(n^{4.082})$
NK Landscape $k = 5$	10	20	20	22	323 (303)	278 (259)	339 (354)
	15	36	41	29	1,505 (1,531)	1,530 (1,628)	1,377 (1,276)
	20	57	65	61	4,903 (3,998)	6,108 (4,566)	5,719 (4,468)
	25	75	91	66	11,970 (7,482)	12,786 (8,046)	13,763 (8,758)
	30	101	127	80	21,661 (11,381)	23,719 (11,898)	29,198 (13,972)
	35	109	150	712	30,040 (15,921)	37,541 (15,460)	229,406 (130,996)
	40	118	178		40,297 (17,375)	52,360 (19,546)	
	45	131	191		53,757 (20,501)	68,102 (21,763)	
	50	153	208		75,992 (29,840)	84,371 (23,122)	
	55	163	253		91,956 (32,505)	116,587 (32,407)	
	60	190	268		116,327 (35,009)	137,208 (42,916)	
	65	208	290		142,865 (42,147)	161,899 (30,464)	
	70	224	316		171,490 (46,441)	198,003 (40,129)	
	Estimated scalability				$\Theta(n^{3.134})$	$\Theta(n^{3.259})$	$\Theta(2^{0.351n})$

TABLE 5.2: Results of global optimisation experiments (see text).

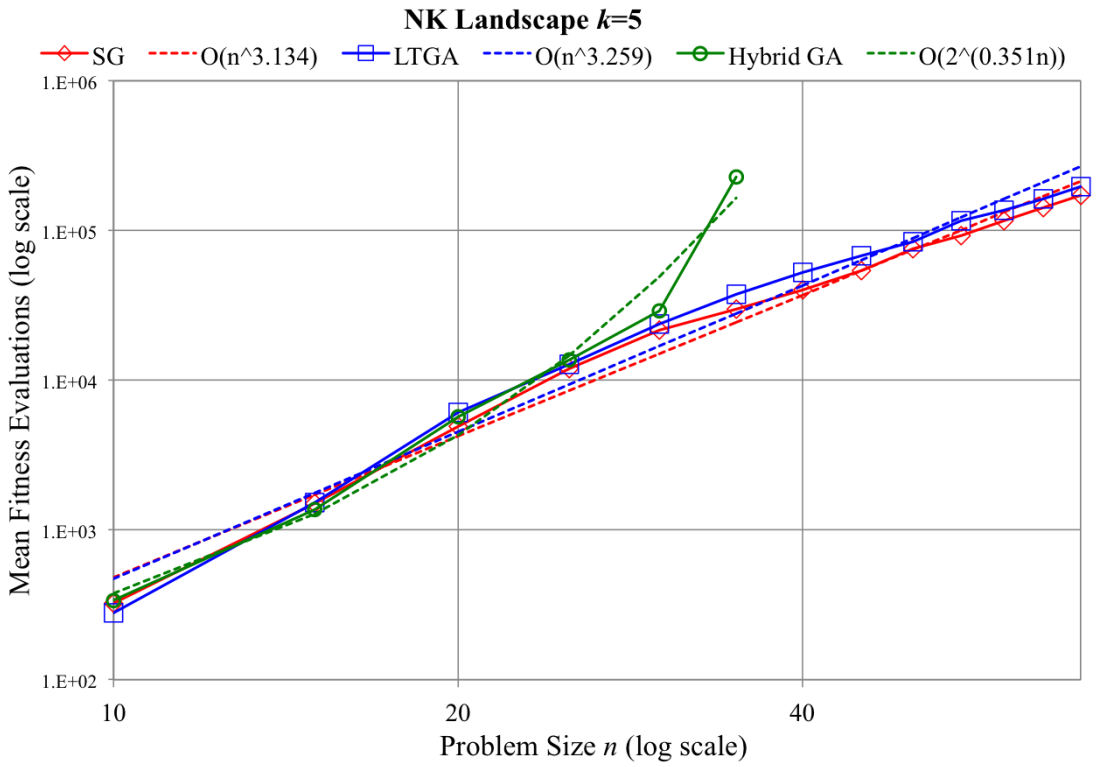

FIGURE 5.10: Global function optimisation results for the nearest-neighbour NK-Landscape problem. This is another example of a problem class with an overlapping, randomised structure. In this case, the Hybrid GA scaled exponentially in the problem size, whereas SG and LTGA achieved a similar low-order polynomial scaling. A slight advantage is suggested in the use of SG.

"simple variation on a suitable representation", with the structure of the problem inferred into the hierarchical schema structure of SG. This representation provides a map between independent mutational variation in genotype space and directed variation in phenotype space. These directions have a powerful impact on effective fitness, allowing large number of fit and novel offspring to be generated, even when the parent population are at local peaks on the fitness landscape. The results suggest a significant advantage of indirect model-guided variation over unguided recombination.

In Section 5.2 the performance of SG in global function optimisation experiments is shown to match the state of the art, LTGA. Given the similarity in performance between SG and LTGA on the problems tested in Chapter 5, an obvious question arises: is SG just LTGA in disguise, and if so, why use SG instead of LTGA? Both create hierarchical models using agglomerative clustering, and both facilitate explicitly directed variation using schemata (although in the LTGA case alleles are sampled from the population and are extrinsic to the model). As we have seen, both can create a powerful exploration distribution that enables evolutionary search performance far superior to unguided methods. LTGA cannot model complex overlapping structures. However, we are yet to see examples where this creates a significant disadvantage when compared with SG, or in fact any other contemporary model-building algorithm. It is also simple and in

practice often builds its model faster than SG, offering reliable $\Theta(n^2)$ performance. Perhaps, for the problems we have examined here, SG does not distinguish itself as the clear method of choice. However, it is the potential of the SG model for other problems and applications, on top of its outright performance here, that affords its significance. This is illustrated in the next chapter, where SG is used to create to quantitative measures of evolutionary information that break new ground for evolutionary model building.

5.4 Summary

In this chapter the effective fitness of SG was tested over evolutionary time using global function optimisation experiments. The results demonstrate that a Compression EA, implemented using a compact SG model, is able to exploit inferred problem structure to solve important classes of test problem in low order polynomial time. They show a significant benefit of the compact model and the exploration distribution it offers over traditional (unbiased) GA crossover. Moreover, they show that this approach can match state of the art performance. This establishes SG as a first class alternative to existing model-building algorithms, and demonstrates for the first time the potential of compact models for unbiased BBO.

Chapter 6

Quantifying Information in Evolutionary Populations

Having examined the use of SG for search, we can now consider additional benefits it provides over alternative model-building algorithms. This chapter takes advantage of the information-theoretic properties of SG as a lossless compact model. In particular, it exploits a relationship between lossless compression and Kolmogorov Complexity (KC) that makes it possible to approximate the information content of data by measuring the size of its compressed representation in SG. This approach not only enables quantitative evidence supporting the thesis, it also allows information measures to be created that are useful in their own right. The advantage of using SG in this context is that it is able to detect information even when it contains complex epistatic structure. This sets it apart from both sequential compression, which can only detect sequential interactions, and some alternative measures of evolutionary information, which do not detect interactions at all.

The first half of this chapter builds upon recent theoretical work, which shows how the entropy of evolutionary populations relates to knowledge about their selective environments (Adami & Cerf, 2000). Using this theory it is possible to create measures that estimate how much knowledge about evolutionary problems different population models contain. Three different models are compared in this way: SG using offline compression; a lossless sequential compression model; and an evolutionary complexity model from the literature. A number of problems are used to test the models, including problems with a simple univariate structure, a symmetric bivariate structure, a modular structure, and a complex overlapping structure. Un-shuffled versions of the problems are tested, in which there is an inherent sequential order to the problem representation, as well as shuffled versions, where this positional bias is removed.

The results show that SG can detect information content across all the problems tested, whilst on some of the problems sequential compression and evolutionary complexity

models fail. In every case, SG is able to detect more information about the problem than the alternatives. These results prove that SG is able to capture information about complex selective environments from a population, including those containing overlapping, high-order and both sequential and non-sequential epistatic interaction.

The second half of the chapter shows that the quantity of information detected by SG has a strong predictive quality with respect to fitness. Instances of the model are trained using representative populations of fit individuals in different selective environments. Then, the fitnesses of previously-unseen test individuals are predicted by measuring how well they fit the model. This is performed by parsing the individuals using the grammar's production rules, which infers a genotype. The size of the parsed genotype provides a quantitative estimate of "distance" between the individual and the model. This method is compared against a distance measure based on PCA, the *Mahalabonis Distance* (MD), which is also able to deal with data sources with interacting variables.

The results in this section demonstrate that on all tested problem classes there is a correlation between an unseen individual's actual fitness and the fitness predicted by the SG model. Moreover, for each of the problem classes tested, there is a greater correlation using the SG model than the PCA-based approach. These results provide further evidence that SG models are able to infer the structure of the selective environment. They also suggest the potential of SG as a more general classification algorithm for combinatorial data sets – an idea explored further in the chapter's conclusions.

6.1 Approximating Population Knowledge

A number of recent works have shown that it is possible, using KC theory, to construct measures analogous to Shannon mutual information for models containing absolute information (Bennett et al., 1998; Li et al., 2004; Cilibrasi & Vitányi, 2005). These measures relate to the information shared between *objects*, and contrast with classical measures of mutual information, which relate to the average information shared between random sources of data. Using $K(X)$ to specify the complexity of an object (the shortest program that can generate it), the *mutual information* between two objects A and B is given by:

$$\begin{aligned} I(A:B) &= I(B:A) \\ &= K(A) - K(A|B) \\ &= K(B) - K(B|A) \end{aligned} \tag{6.1}$$

Which holds within a logarithmic error term (Bennett et al., 1998). In words, the information shared between the objects is equal to the smallest possible representation

of one of the objects *without* using knowledge of the other, minus the smallest possible representation of the same object *with* knowledge of the other. The non-computability of KC tells us that any estimate of this measure is subjective. Therefore, the clearest interpretation of any such estimate is that it represents the mutual information between *particular models* of A and B.

We use work by Adami & Cerf (2000) to relate this measure to evolutionary populations and their selective environment. The authors show that the mutual information between an individual s and a selective environment e can be written in terms of equation 6.1:

$$I(s:e) = K(s) - K(s|e) \tag{6.2}$$

The authors refer to this as the *physical complexity* of the individual s. Here $K(s)$ is the *unconditional complexity* of s – its model complexity in the *absence* of selection, and $K(s|e)$ is the *conditional complexity* of s – its model complexity in the *presence* of selection[1]. The variable s is an abstract concept, representing an individual that is not instantiated in any particular environment. Individuals physically sampled from a population will be subject to the selective pressure of the environment from which they were taken. Therefore, estimating their complexity gives an estimate of $K(s|e)$. The authors show that $K(s)$ is equal to the *expected* complexity of an individual – in other words, the average complexity of all possible instantiations of an individual in a given space.

We can generalise this approach to the population modelling case that we are interested in here. We consider an abstract population $\mathbb{P}$ of phenotypes, not taken from any particular selective environment, and a physical sample population $\mathbb{P}_e$ taken from a selective environment e. Following the approach described above, we can define the mutual information between a population and its environment as:

$$I(\mathbb{P}:e) = K(\mathbb{P}) - K(\mathbb{P}_e) \tag{6.3}$$

Where $K(\mathbb{P})$ is the expected model complexity of a population instance and $K(\mathbb{P}_e)$ is the complexity of a sample population taken from a selective environment e. We can turn $I(\mathbb{P}:e)$ into a relative measure that is comparable between models by normalising it. We refer to this measure as the *population model information content* (PMIC):

$$\begin{aligned} \mathrm{PMIC}(\mathbb{P}, e) &= \frac{I(\mathbb{P}:e)}{K(\mathbb{P})} \\ &= 1 - \frac{K(\mathbb{P}_e)}{K(\mathbb{P})} \end{aligned} \tag{6.4}$$

[1]The original authors use $K(s)$ and $K_0(s)$ to differentiate between the complexity and unconditional complexity of s.

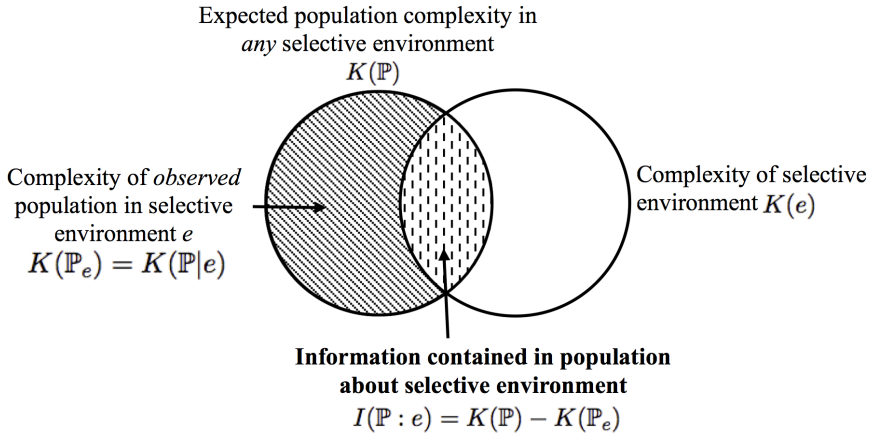

FIGURE 6.1: A Venn diagram illustration of the Population Model Information Content (PMIC). The amount of information contained in a population about an evolutionary problem can be estimated by measuring the difference between the expected and observed information the population contains.

Where $0 \leq \text{PMIC}(\mathbb{P}, e) \leq 1$. A PMIC value of 0 indicates that a model detects no mutual information between a population and a selective environment, and a PMIC value of 1 indicates that a model detects complete overlap. This measure is analogous to the concepts of *grammar redundancy* (Ebeling & Jiménez-Montaño, 1980), which is defined using the length of a grammar's production rules instead of K, and Shannon redundancy, which is defined using entropy. Using this measure, we can quantitatively compare the ability of models to infer the structure of various problems from populations of fit individuals. The PMIC concept is illustrated in Figure 6.1.

To make this measure practical, $K(\mathbb{P})$ and $K(\mathbb{P}_e)$ need to be approximated for each model considered. We now show how to do this for a sequential compression benchmark, SG, and a benchmark based on univariate entropy.

6.1.1 Sequential Compression Models

Work by Cilibrasi & Vitányi (2005); Li et al. (2004) has shown that the length of a compression code for sequence X, $C(X)$, can be used to estimate $K(X)$. Substituting into equation 6.4 gives the following:

$$\text{PMIC}(\mathbb{P}, e) \approx 1 - \frac{C(\mathbb{P}_e)}{C(\mathbb{P})} \tag{6.5}$$

For any given sample population $\mathbb{P}_e$, $C(\mathbb{P}_e)$ is simply the length of its losslessly compressed representation in bits. $C(\mathbb{P})$ is equivalent to the average compressed length of all possible instantiations of the population. This can be estimated by generating a large number of random populations of the same size as the sample population, compressing

them and taking the average compression length. In the experiments described below this procedure was performed using 10,000 random population instances.

We use the DEFLATE algorithm as a sequential compression benchmark. This algorithm combines LZ77 compression (Ziv & Lempel, 1977) with Huffman coding (Huffman, 1952) and was configured for maximum compression. To compress each population, bit string representations of each individual were concatenated into a single source sequence.

6.1.2 Schema Grammar Instances

SG instances inferred using lossless compression are compact information models, but are not complete entropy codings in themselves. Therefore, a different approach to estimating complexity is required. Fortunately, it is possible to calculate the *unnormalised* entropy of compression grammars (Kieffer & Yang, 2000), giving an idealised estimate of $C(\mathbb{P})$ and $C(\mathbb{P}_e)$. The unnormalised entropy calculation is based on straight-line grammars with sequential production rules. However, this does not effect the calculation of entropy as this is a permutation-independent calculation based on symbol frequency only.

Grammar entropy is calculated over the rule hierarchy of the grammar rather than the language yielded by the grammar. The calculation regards the production of each symbol as an independent event. However, as interactions in the original data are factorised by lossless compression into non-terminal symbols, the information they represent is captured by the entropy calculation.

Following the approach of Kieffer & Yang (2000) the RHS of all production rules are first aggregated into a single collection. Let ρ_G be the multiset union of the right hand side of each production rule in R, with the first instance of each non-terminal symbol removed:

$$\rho_G = \left\{ \biguplus_{\alpha \to \beta \in R} \beta \right\} \setminus V \tag{6.6}$$

ρ_G is a multiset, meaning that the contents of the set are unordered and each item may appear more than once. As an example, the grammar instance described in Section 3.3.5 gives:

$$\rho_G = \{x_1^1, x_2^1, x_1^1, x_1^0, s_2, x_3^1, x_4^1, x_2^0, s_1\}$$

We use the multiplicity function $f(\alpha, \rho_G)$ (shortened to $f(\alpha)$ for brevity) (Blizard, 1988) to indicate the frequency of occurrence of α in ρ_G, so in the previous example $f(x_1^1) = 2$. Given this, we can now define unnormalised entropy.

Definition 6.1. The unnormalised entropy of an SG instance $\mathbb{G}$ is given by:

$$\begin{aligned} H(\mathbb{G}) &= \sum_{\alpha \in \rho_G} \log\left(\frac{|\rho_G|}{f(\alpha)}\right) \\ &= |\rho_G| \log |\rho_G| - \sum_{\alpha \in \rho_G} \log f(\alpha) \end{aligned}$$

So using the same example, and taking logarithms in base 2:

$$\begin{aligned} H(\mathbb{G}) &= 9 \log 9 - f(x_1^1) \log f(x_1^1) - f(x_2^1) \log f(x_2^1) - f(x_1^0) \log f(x_1^0) \\ &\quad - f(s_2) \log f(s_2) - f(x_3^1) \log f(x_3^1) - f(x_4^1) \log f(x_4^1) - f(s_1) \log f(s_1) \\ &= 9 \log 9 - 2 \log 2 - 1 \log 1 - 1 \log 1 - 1 \log 1 - 1 \log 1 - 1 \log 1 \\ &\quad - 1 \log 1 - 1 \log 1 \\ &= 26.53 \text{ bits} \end{aligned} \tag{6.7}$$

We can now rewrite equation 6.4 using the grammar entropy.

$$\text{PMIC}(\mathbb{P}, e) \approx 1 - \frac{H(\mathbb{G}_e)}{H(\mathbb{G})} \tag{6.8}$$

Where $\mathbb{G}$ and $\mathbb{G}_e$ are SG instances inferred from $\mathbb{P}$ and $\mathbb{P}_e$ respectively. For each sample population, $\mathbb{G}_e$ is generated by losslessly compressing $\mathbb{P}_e$ using the offline algorithm, from which $H(\mathbb{G}_e)$ is then calculated. $H(\mathbb{G})$ is estimated by generating a large number of random populations of the same size as the sample population, compressing them with SG and taking the average entropy. As before, for the experiments described below, this procedure was performed using 10,000 random population instances.

6.1.3 Per-Site Entropy Models

One possible information model for a population is to consider each dimension as an independent variable. Summing the entropy of each dimension then provides a way of estimating the regularity of traits across a population. Per-site entropy can then also be used to estimate mutual information between populations and selective environments in the way that has been described. This was the approach used to first demonstrate the concept of physical complexity (Adami & Cerf, 2000; Adami et al., 2000; Adami, 2002). The key limitation of this idea is that multivariate mutual information cannot be detected by the model: it is univariate. Nevertheless, it provides a useful benchmark for comparing against the compression-based approaches. In what follows, we assume that each individual in the population is of fixed dimensionality n, with each trait defined over a fixed alphabet Σ. PMIC can be estimated as follows:

$$\text{PMIC}(\mathbb{P}, e) \approx 1 - \frac{H(\mathbb{P}_e)}{H(\mathbb{P})} \tag{6.9}$$

Where $H(X)$ is the per-site average entropy summed over the n sites of X:

$$H(X) = -\sum_{i=1}^{n} \sum_{\alpha \in \Sigma} p(X_i = \alpha) \log p(X_i = \alpha) \tag{6.10}$$

In this case, the value $H(\mathbb{P}_e)$ is simply the per-site entropy of the sample population, calculated as above. $H(\mathbb{P})$ is equivalent to the average entropy of all possible instantiations of the population. One possible approach to quantifying this (and that taken in Adami et al's work) is to use the entropy of an infinitely sized population of all possible individuals, which we define as $H_{\text{MAX}}(X)$. Let $N = |\Sigma|^n$ be the number of all possible individuals:

$$\begin{aligned} H_{\text{MAX}}(X) &= \sum_{i=1}^{N} N^{-1} \log N^{-1} \\ &= \log N \\ &= n \text{ (taking logarithms to base } |\Sigma|) \end{aligned} \tag{6.11}$$

The problem with this approach is that it uses a different modelling assumption to that used to generate $H(\mathbb{P}_e)$: specifically, that the population is infinite. This is erroneous as a finite random population is *expected* to have a small amount of accidental regularity in it. To create a better estimate, we follow a similar approach to that used in the compression models: 10,000 random population instances were created and the entropy of each population was calculated and then averaged. For the population sizes used in the experiments this returned an entropy of 0.9928 bits per bit, which contrasts with the rate of 1 bit per bit in an infinite population, reflecting the anticipated accidental regularity.

6.1.4 Methodology

All experiments used a fixed population size of 100 individuals and binary problems with a dimensionality fixed to $n = 50$. Test problems were selected to measure the information detection capabilities of each model across a variety of different structure types and problem difficulties. This allowed unconditional complexity estimates for each model to be generated in advance using 10,000 random population instances.

For each problem instance, a sample population of unique individuals with above-average fitness was generated using stochastic bit flip hill climbing. Each individual was initialised to a random bit string. A random permutation of the n loci was then generated

and for each locus the allele value was flipped. If the flip created a strict fitness improvement then it was kept, otherwise it was discarded. Any generated individuals already in the population were discarded and the process was repeated. On the unimodal and bimodal test problems that were used (OneMax, OneMax*, TwoMax and TwoMax* – see below) hill-climbing was terminated after $n/2$ steps to prevent convergence on a single global optimum, whilst still generating above-average fitness individuals.

Once sample populations were generated then the conditional complexity of each population $K(\mathbb{P}_e)$ was estimated for each model using the approaches defined above, and PMIC calculated. Results were averaged over 400 random instances of each test problem type. The resulting figures represent the average amount of information detected by each model about each problem type.

6.1.5 Results

6.1.5.1 Simple Univariate Problems

Two fully separable problems with no epistasis and a simple univariate structure were selected: OneMax and a variant with shuffled allele values we define as OneMax*. The OneMax problem fitness is given as a function of the number of ones in the candidate solution X:

$$\textsc{OneMax}(X) = \sum_{i=1}^{n} X_i \tag{6.12}$$

The results of the experiment for OneMax are show in Table 6.1. In this case, the selective environment causes individuals of above average fitness to have alleles with a higher probability of being 1 than 0. This appeals directly to the per-site entropy model, and the results show that it was able to detect information as expected. The sequential compression model works by detecting and exploiting information that is observable as sequential allelic structure. Although the problem contains no such structure intrinsically, the univariate structure that *is* present implicitly favours sequential patterns of 1s. The results show that the model was able to detect information in this way, although the signal was not as strong as the entropy model case.

In the case of SG compression, the model works by detecting and exploiting interactions between alleles across the population (either sequential or non-sequential). Once again, although the problem contains no such structure intrinsically, it implicitly favours interactions between alleles of many different orders and these were detected by the model. In fact, this provided the strongest information signal of the three different approaches. These results also demonstrate that a uniquely multivariate model can be sufficient to detect information from a problem with a uniquely univariate structure.

Problem	PMIC Information (Standard. Dev.)		
	Sequential Compression	Schema Grammar	Univariate Entropy
OneMax	0.095 (0.009)	0.246 (0.009)	0.175 (0.008)
OneMax*	0.004 (0.007)	0.246 (0.009)	0.175 (0.008)
TwoMax	0.079 (0.009)	0.082 (0.02)	0.000 (0.003)
TwoMax*	0.003 (0.007)	0.084 (0.02)	0.000 (0.003)
M-TwoMax	0.425 (0.011)	0.796 (0.008)	-0.001 (0.003)
SM-TwoMax	0.015 (0.011)	0.797 (0.008)	-0.001 (0.003)

Table 6.1: Average PMIC information measure for each modelling method (see text).

In the OneMax* function the all-1s optimum is replaced with a single, random, global optimum X^*. Fitness is calculated using the hamming distance between the candidate solution and the global optimum. This creates a problem with an identical bit-flip fitness landscape to the OneMax problem, which is convex and trivial to navigate, but with no implicit order to allele values:

$$\text{OneMax}^*(X) = n - \sum_{i=1}^{n} |X_i^* - X_i| \tag{6.13}$$

The results show that in this case, the information signal detected by both SG and the per-site entropy model was identical to the OneMax case. This is expected, as both models are unbiased with respect to specific allele values and the problems are combinatorially identical. However, the sequential compression algorithm struggled to detect any information. The difference in this case is that although sequential patterns still exist between alleles in the population, they do not all look the same. Each such pattern thus occurs far less frequently, and the algorithm is not able to generalise in the same as OneMax.

6.1.5.2 Simple Epistatic Problems

The level of complexity was raised by introducing two problems with a simple epistatic structure. The TwoMax function is a bi-modal extension of OneMax with global optima at the all-0s and all-1s configuration. It is defined as follows:

$$\text{TwoMax}(X) = \max\left\{\sum_{i=1}^{n} X_i, n - \sum_{i=1}^{n} X_i\right\} \tag{6.14}$$

In this selective environment, the distribution of 1s and 0s in each above-average fitness individual is bimodal rather than unimodal. The bit-flip fitness landscape of TwoMax is convex and trivial to solve, as with the OneMax problems. However, there is no

univariate structure in the problem. Instead, structure is defined by pairwise epistatic interactions between each of the variables. The results in Table 6.1 show that the per-site entropy failed to detect any information. This demonstrates a case where a uniquely univariate model is not sufficient to detect information about a uniquely multivariate problem structure. The sequential compression model was able to detect information, and only slightly less than the OneMax case. As before, SG detected the most information, although considerably less than the OneMax case, and only slightly more than sequential compression.

As with OneMax, a version of TwoMax was created that removes the regularity of allele values in the global optima. In this problem, the optima are defined using a random configuration X^*:

$$\text{TwoMax}^*(X) = \max\left\{\sum_{i=1}^{n}|X_i^* - X_i|, n - \sum_{i=1}^{n}|X_i^* - X_i|\right\} \tag{6.15}$$

As with OneMax*, the results show that removing the regularity in allelic values made it difficult for sequential compression to detect any information, although this did not have a detrimental effect on SG.

6.1.5.3 Simple Modular Problems

A problem with a simple modularly separable structure was created by concatenating TwoMax functions together, each of length $k = 5$, into an additive fitness function:

$$\text{M-TwoMax}(X) = \sum_{j=1}^{m}\text{TwoMax}(X_{1+(j-1)k}, \ldots, X_{jk}) \tag{6.16}$$

The problem structure contains $m = 10$ modules, each with two locally-optimal configurations (all-1s and all-0s), and 2^m global optima. In this case, the members within each module all interact with one another but there is no interaction between modules. As before, the bit-flip fitness landscape is convex, and each member of the sample population was located on a unique global optimum. The results in Table 6.1 show that once again the univariate entropy model failed to detect any information in the population. On the other hand, both compression models detected strong information signals, with SG returning the largest value at 0.796. The strength of performance of both models is explained by the fact that problem structure was decomposable into $2m$ module configurations. This structure essentially transforms the size of the search space, creating a huge amount of redundancy in the original problem dimensions. This redundancy is observable without noise in both the sequential and combinatorial structure of the sample population.

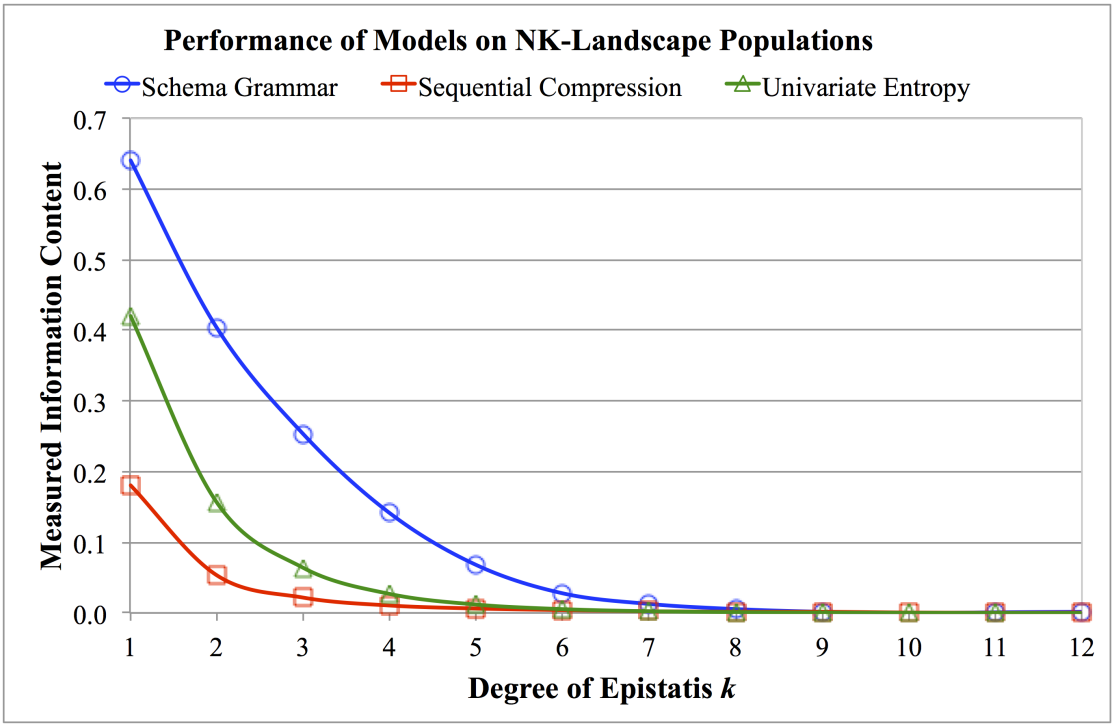

FIGURE 6.2: Information detection rates on nearest-neighbour NK landscape problems with varying epistasis k.

A shuffled version of M-TWOMAX was also tested. In this variation a random bijective re-mapping of problem dimensions, $\sigma(i)$, is applied to the M-TWOMAX fitness function:

$$\text{SM-TWOMAX}(X) = \sum_{j=1}^{m} \text{TWOMAX}(X_{\sigma((1+(j-1)k)}, \ldots, X_{\sigma(jk)}) \tag{6.17}$$

This re-mapping does not effect the underlying epistatic structure or difficulty of the problem, but it essentially removes its observability in a sequential representation. The results show that as SG observes combinatorial and not sequential interaction, it was not effected by the change. On the other hand, the sequential compression model, which relies on the presence of sequential order, was able to detect less than $1/27^{\text{th}}$ of the information it was previously able to.

6.1.5.4 Problems Containing Higher-Order Epistasis

The models were tested on NK-landscape problems, which were defined in Section 4.3.5. Figure 6.2 illustrates the results for the different models across a range of k values. They show that the univariate model detected a strong information signal from the population for low k, with a non-negligible signal up to $k = 5$. This demonstrates a case where a uniquely univariate model is able to detect information from a uniquely multivariate problem structure, in contrast to TWOMAX, and suggests that simple univariate EDA's

can offer some benefit on problems with a complex epistatic structure. Sequential compression was also able to detect a clear information signal for low k, but this was not as strong as the other methods. Detected information tailed off quicker than the other methods as complexity was raised, returning a non-negligible signal up to $k = 4$. The SG compression model had the best comparative performance, with a very strong information signal for low k and a slower tail-off than the other methods, returning a non-negligible signal up to $k = 7$.

6.2 Classifying Fit Individuals

By using the same framework of information and complexity theory, it is possible to test quantitatively the predictive quality of the SG model with respect to fitness. In this section a continuous measure is defined that calculates the *information distance* between previously unseen test individuals and SG models of fit training populations. The distance is estimated by parsing each test individual using the production rules of SG. This infers a compact genotype representation of the individual using the CSS of the grammar as a G-P map. The size of the inferred genotype reflects how well a model instance *recognises* a previously unseen individual. In this case then, compressibility is synonymous with classification. The results in this section demonstrate that for the test problems examined, using SG parsing to classify fitness performs as well as an established PCA-based method from the machine learning literature.

6.2.1 Information Distance

By using the Kolgorov Complexity analogues of Shannon information, a number of authors have created information distance measures for objects using absolute information. The original formulation of information distance between two objects x and y was proposed by Bennett et al. (1998) as the following:

$$E(x, y) = \max\{K(x|y), K(y|x)\} \tag{6.18}$$

If the complexity of x and y is identical, this measure is simply equal to the amount of absolute information in each object that is not shared. The reason for using the maximum of $K(x|y)$ and $K(y|x)$ is in order to create a measure that is symmetric between x and y. The idea of information distance has since been developed further, and a relative version of the measure called the *Normalised Information Distance* (NID) proposed (Li et al., 2004; Cilibrasi & Vitányi, 2005). In this variant, information distance is rescaled relative to the absolute size of the objects x and y:

$$\text{NID}(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \tag{6.19}$$

Assuming that at least one of $K(x)$ and $K(y)$ is non-zero, then $0 \leq NID(x, y) \leq 1.0$. The authors argue that scaling the measure in this way allows comparison between pairs of objects of different sizes, which is similar to the approach taken with PMIC in the previous section. Again, the use of the max terms is in order to create a measure that is symmetric between x and y. These works also show that NID can be approximated using the size of the compressed versions of the objects as a substitute for their complexity. Reworking Equation 6.19 so it can be expressed in terms of the individually and jointly compressed versions of the objects gives the *Normalised Compression Distance* (NCD):

$$\begin{aligned} \text{NCD}(x, y) &= \frac{C(x, y) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \\ &\approx NID(x, y) \end{aligned} \tag{6.20}$$

The intuition here is that mutual information between two objects will lead to a space saving when they are compressed together. In this section, we wish to measure the information distance between a population under selective pressure, $\mathbb{P}_e$, and a previously unseen test individual t. In this case it is likely that $K(\mathbb{P}_e) \gg K(t)$. This means that if a NID based measure is adopted then the $K(\mathbb{P}_e|t)$ and $K(\mathbb{P}_e)$ terms that would appear using Equation 6.19 would dominate the $K(t|\mathbb{P}_e)$ and $K(t)$ terms respectively, rendering the scale of the measure difficult to interpret for the use case.

For our purposes here it is not necessary to make the distance measure either symmetric or relative, allowing us to create a simpler variant of the concept that we refer to as the *Distance To Population* (DTP). DTP is illustrated in Figure 6.3, and is given as follows:

$$\begin{aligned} \text{DTP}(t, \mathbb{P}_e) &= K(t|\mathbb{P}_e) \\ &= K(t, \mathbb{P}_e) - K(\mathbb{P}_e) \end{aligned} \tag{6.21}$$

In words, this says that the information distance between an individual and a population is equal to the amount of absolute information contained by the individual, minus the information it shares with the population.

6.2.2 Classification By Parsing

One approach to estimating DTP is to infer a SG instance from the sample population $\mathbb{P}_e$ on its own, then infer another instance using the population together with test individual.

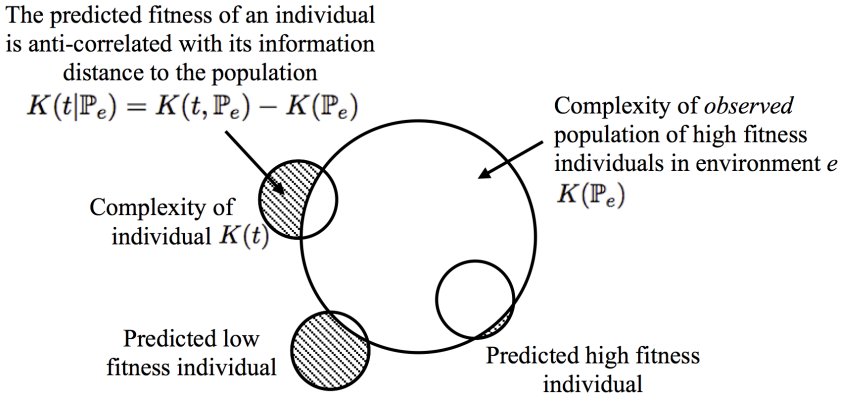

FIGURE 6.3: A Venn diagram illustration of the information distance of previously unseen test individuals to a fit population. By interpreting the fit population as a proxy for the selective environment, the fitness of each individual can be predicted using the distance measure.

Step	Parse Rule State	$\overset{*}{\Rightarrow}$	Schema Symbol	$\overset{*}{\Rightarrow}$
1	$g_t \to \{x_1^1, x_2^1, x_3^1, x_4^1, x_5^0, x_6^0, x_7^0, x_8^0\}$	11110000		
2	$g_t \to \{s_1, x_3^1, x_4^1, x_5^0, x_6^0, x_7^0, x_8^0\}$	11110000	$s_1 \to \{x_1^1, x_2^1\}$	11******
3	$g_t \to \{s_1, s_2, x_5^0, x_6^0, x_7^0, x_8^0\}$	11110000	$s_2 \to \{x_3^1, x_4^1\}$	**11****
4	$g_t \to \{s_1, s_2, x_5^0, x_6^0, x_7^0, x_8^0\}$	11110000	$s_3 \to \{x_1^0, x_2^0\}$	00******
5	$g_t \to \{s_1, s_2, x_5^0, x_6^0, x_7^0, x_8^0\}$	11110000	$s_4 \to \{x_3^0, x_4^0\}$	**00****
6	$g_t \to \{s_1, s_2, x_5^0, x_6^0, x_7^0, x_8^0\}$	11110000	$s_5 \to \{x_5^1, x_6^1\}$	****11**
7	$g_t \to \{s_1, s_2, x_5^0, x_6^0, x_7^0, x_8^0\}$	11110000	$s_6 \to \{x_7^1, x_8^1\}$	******11
8	$g_t \to \{s_1, s_2, s_7, x_7^0, x_8^0\}$	11110000	$s_7 \to \{x_5^0, x_6^0\}$	****00**
9	$g_t \to \{s_1, s_2, s_7, s_8\}$	11110000	$s_8 \to \{x_7^0, x_8^0\}$	******00
10	$g_t \to \{s_9, s_7, s_8\}$	11110000	$s_9 \to \{s_1, s_2\}$	1111****
11	$g_t \to \{s_9, s_7, s_8\}$	11110000	$s_{10} \to \{s_3, s_4\}$	0000****
12	$g_t \to \{s_9, s_7, s_8\}$	11110000	$s_{11} \to \{s_5, s_6\}$	****1111
13	$g_t \to \{s_9, s_{12}\}$	11110000	$s_{12} \to \{s_7, s_8\}$	****0000

TABLE 6.2: A step-by-step illustration of parsing the test phenotype 11110000. The parse process walks through the schema symbol table. If the schema symbol rule is a subset of the parse rule then the schema symbol is substituted. The result in the example is the factorised genotype $g_t \to \{s_9, s_{12}\}$

Measuring the unnormalised entropy of each of these SG models will provide estimates of $K(\mathbb{P}_e)$ and $K(t,\mathbb{P}_e)$ respectively. However, this is computationally expensive, as the second procedure has to be performed for each test individual.

Another approach is to infer an SG instance from the sample population $\mathbb{P}_e$ on its own. Then, the individual t can be *parsed* using the production rules of the SG instance. Parsing infers a factorised genotype for the individual, $g_s \overset{*}{\Rightarrow} t$, using the compositional schema structure of the grammar as a G-P map. Essentially, the smaller the parsed genotype then the closer the distance from the individual to the population. In the following description of parsing, we assume that the SG instance was inferred using the offline algorithm (Section 3.4). SG instances inferred by the offline algorithm have schema symbols created in order of frequency of symbol co-occurrence. This allows a

simple parsing algorithm to be defined that has a time complexity linear in the number of schema symbols in the grammar. Additionally, it ensures that the resulting derivations are unambiguous and consistent with the genotype derivations already in the grammar. In practice, alternative inference and parsing algorithms could be used, but the approach taken here is sufficient for current purposes.

To enable parsing we assume the set of schema symbols in each grammar, $S = \{s_1, s_2, \ldots, s_n\}$, is ordered in sequence of creation by the inference algorithm. Parsing proceeds by creating a new production rule for the test phenotype, which we specify as $g_t \rightarrow \delta$. For example, and assuming a binary problem representation, if the test phenotype $t = 10101$ then $\delta = \{x_1^1, x_2^0, x_3^1, x_4^0, x_5^1\}$. Then the production rule for each schema symbol in S is evaluated in sequence. Here we use $\alpha \rightarrow \beta$ to specify each evaluated production rule. If β is a subset of δ then these symbols are removed from δ and replaced with α. For example, if the evaluated rule was $s_1 \rightarrow \{x_1^1, x_2^0, x_3^1\}$, then the resulting replacement would give $g_t \rightarrow \{s_1, x_4^0, x_5^1\}$. This process continues until all schema symbols have been evaluated. The process is defined in Algorithm 4. An example parse that illustrates the process is shown in Table 6.2.

Algorithm 4: Simple parsing algorithm for SG instances inferred using the offline algorithm.

input : test phenotype to parse t
input : SG instance $\mathbb{G} = \{V, G, S, \Sigma, R\}$
output: parsed production rule δ for inferred genotype g_t, where $g_t \rightarrow \delta$ and $g_t \overset{*}{\Rightarrow} t$

```
// Initialise parse rule
δ = t
// Enumerate schema symbol rules in order of creation
for s ∈ 𝕊 do
    // Retrieve the production rule associated with s
    β = R[s]
    // Is the production rule associated with s a subset of δ?
    if β ⊆ δ then
        // Substitute the symbols with the schema symbol s
        δ = {δ \ β} ∪ {s}
    end
end
return δ
```

Once parsing is complete, DTP can be estimated using the size of the new parsed rule, calculated in terms of unnormalised entropy. Given $\mathbb{G}_e$ as the SG model of the original sample population, where $\mathbb{G}_e \overset{*}{\Rightarrow} \mathbb{P}_e$, let $\mathbb{G}_e^{+t}$ be the same model with the parsed genotype added:

$$\mathbb{G}_e^{+t} \overset{*}{\Rightarrow} \{\mathbb{P}_e, t\} \tag{6.22}$$

This allows DTP to be estimated as follows:

$$\mathrm{DTP}(t, \mathbb{P}_e) \approx H(\mathbb{G}_e^{+t}) - H(\mathbb{G}_e) \tag{6.23}$$

In practical terms, this can be calculated in linear time for each test individual, by copying and incrementing the symbol frequencies used to calculate $H(\mathbb{G}_e)$.

6.2.3 Mahalanobis Distance

In order to compare the predictive quality of SG, a PCA-based benchmark was used. *Mahalanobis Distance* (MD) (Mahalanobis, 1936) is a continuous measure of the distance between a single point and a collection of samples. It is calculated using the Euclidian distance between the point and the centre of the samples in transformed principal component space. By transforming the space using the covariance of the sample data, the measure is able to support interactions in the data set. Additionally, it is insensitive to sequential ordering. For these reasons it is a good comparison benchmark for the SG DTP measure. MD is commonly used in the machine learning literature for classification and clustering (for a good example see Xiang et al., 2008).

In the following, we assume a binary problem representation. We treat each test individual s as a vector of length n and each sample population of fit individuals $\mathbb{P}_e$ a matrix of dimensionality $n \times p$, where p is the size of the population. Using $\overline{\mathbb{P}_e}$ to represent the mean of $\mathbb{P}_e$ and $\textsc{Covar}(\mathbb{P}_e)$ the covariance, the MD $D(s, \mathbb{P}_e)$ is given by the following:

$$D(s, \mathbb{P}_e) = \sqrt{(s - \overline{\mathbb{P}_e})^T \textsc{Covar}(\mathbb{P}_e)^{-1} (s - \overline{\mathbb{P}_e})} \tag{6.24}$$

This measure is unitless and scale invariant.

6.2.4 Methodology

For each problem instance a population of 100 unique above-average fitness individuals was generated using the methods described in the previous section, with one exception. The covariance calculation used in MD can degenerate if the sample population is highly discontinuous. A good example of this occurring is when the sample population contains bit-flip local optima of the M-TwoMax problem. In this circumstance, it is not possible to invert the covariance matrix and thus to calculate the MD comparison measure. To avoid this, the local search step for M-TwoMax was terminated after $n/2$ steps. The fit sample populations were then modelled by inferring an instance of SG using the offline algorithm, and their mean and co-variances calculated for the MD.

For each problem instance test, individuals were generated across a wide distribution of fitness values. Bit-flip hill climbing and descent searches were run alternately, with

each mutant on the adaptive path added to the test population if unique. Any test individuals already present in the fit sample population were discarded. This process continued until 100 previously unseen test individuals were generated. For each test individual the DTP distance measure was estimated using SG parsing, as well as the MD distance measure and the actual fitness.

Experiments were run using OneMax, TwoMax, M-TwoMax and $k = 3$ nearest-neighbour NK-landscapes problems as selective environments. The allele and locus shuffled versions of the problems were not tested as neither distance measure is sensitive to these permutations. Results were aggregated over 1000 separate problem instances of each type. To enable aggregation and comparison across problem instances, the fitness values of each test individual were normalised using the distribution of the training population fitness:

$$f_n(t) = \frac{f(t) - \bar{f}}{f_\sigma} \tag{6.25}$$

Where $f(t)$ was the fitness of the test individual according to the original fitness function, $\bar{f}$ was the average fitness of the fit sample population $\mathbb{P}_e$, and f_σ was the standard deviation of the fit sample population.

6.2.5 Results

The results of the experiments are shown in Figure 6.4. For each problem class the fitnesses of the test individuals are plotted against their distance to each model. Linear R^2 correlation values are shown in each case, which gives an indication of how well fitness is classified by the distance measures.

Both distance measures are shown to have a significant negative correlation with fitness on the problems tested. This indicates that both are able to detect information containing univariate, bivariate, modular, and higher-order overlapping structure. In the case of SG, this is consistent with the previous experiment. In the case of MD, it confirms the ability of PCA to detect complex interactions even when the sample sizes (populations) are relatively small.

In all cases, the correlation values of SG were more significant than the MD case. This suggests that it offers a stronger predictive model of the selective environments over the range of fitnesses tested. The difference in performance is most apparent as the structure of the selective environment becomes more complex. In the case of NK-landscapes, which contain overlapping, higher order structure, the MD model can be seen deteriorating as the fitness of the test individual moves further away from the training population distribution. This illustrates the limitation of the bivariate PCA model in the presence of more complex interactions.

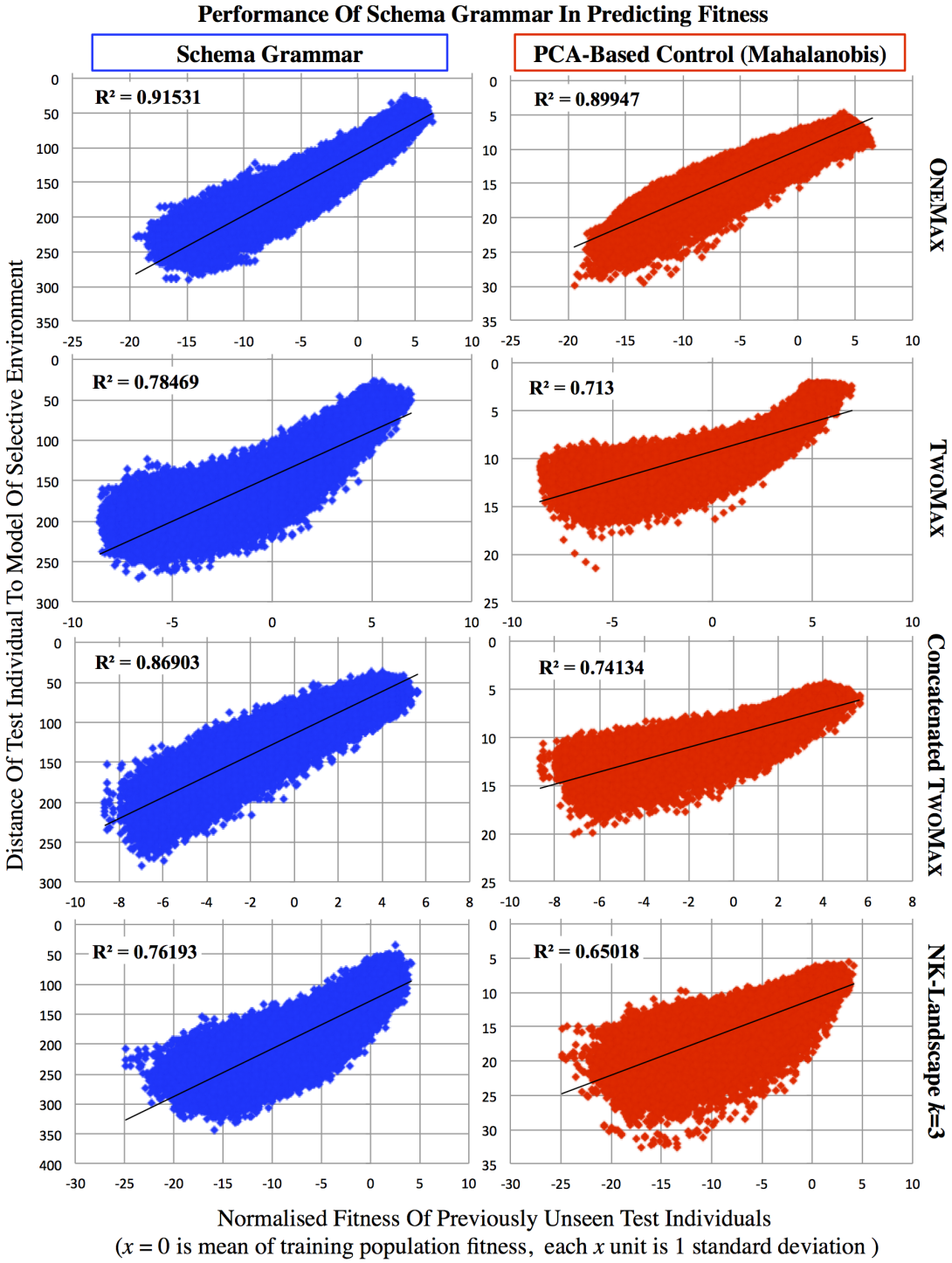

FIGURE 6.4: **Classifying fitness using Schema Grammar**. Training populations containing individuals of above-average fitness were selected and modelled using SG and a PCA-based control. The fitness of previously unseen test individuals was predicted by measuring their distances to each model. In the SG case, this was performed by parsing the test individual using the grammar model and measuring the unnormalised entropy delta. Distances are in bits in the SG case, and unitless in the MD case.

The results show the ability of SG to generalise about fitness, including when the test individual fitnesses fall outside the range seen in the training population. In the upper ranges where test individuals are fitter than any member of the training population, an accurate prediction signifies test individuals sharing more information with the training population (as a whole) than any of its members. At the lower end, where individuals have a fitness below that of any member of the training population, the model predicts fitness as a function of information that *isn't* shared with the population.

6.3 Discussion

The results in this chapter Section 6.1 provide quantitative support, grounded in information and complexity theory, for the claim that the SG model is able to infer the structure of the selective environment. More generally, the work creates an important link between KC and evolutionary model building, allowing model-guided evolutionary search to be understood in information and complexity-theoretic terms.

The work also contributes new measures of evolutionary information that are sensitive to complex epistatic interaction. It demonstrates the limitations of both sequential compression and the "physical complexity" model (Adami, 2002) for detecting information when it contains complex structure, and that these limitations are overcome by the SG model. To the author's knowledge, a general purpose measure of absolute information that is sensitive to both functional context and epistatic interaction has not yet appeared in the literature. Therefore, this presents a potentially exciting avenue of further research.

The results for NK landscapes also provide useful insights into the nature and difficulty of this class of problem. They show, for the SG model at least, when epistasis might become so complex that there is no observable information in the population about its selective environment. The relationship between epistasis and PMIC information content seems to reflect the canonical relationship between epistasis and ruggedness, although the latter is a description based on bit-flip fitness landscapes. The work presented here suggests it may be possible to create general measures of problem difficulty that are grounded in information and complexity theory and avoid the need to presuppose a variation neighbourhood. This also presents a potentially exciting avenue of further research.

In Section 6.2 the ability of SG to classify fitness was demonstrated. This has provided quantitative evidence that the structure inferred by the SG model has a strong predictive quality with respect to fitness. It also shows that SG can be used to create a useful information distance measure in its own right, and can outperform a well established distance metric in certain cases. What makes this more impressive is the fact that SG

can be used for a variety of purposes and is not just a distance measure, whereas this is not the case for MD, or in fact most other alternatives.

Overall, the work represents an interesting new direction for the theory and practice of evolution. It also has the potential to be applied outside of the field as a more generic machine learning mechanism. This is taken up further in Chapter 7.

6.4 Summary

This work breaks new ground by using evolutionary model building to create quantitative measures of evolutionary information. These measures have been used to provide strong quantitative support for the hypothesis that compact models of fit populations can reflect the compositional structure of the selective environment and have strong predictive qualities with respect to fitness. Beyond this, being able to measure absolute information whilst respecting multivariate interaction is an important contribution in its own right. Developing these ideas further is an important direction for future research.

Chapter 7

Conclusions

7.1 Thesis Summary

This work aimed to further the theory and practice of evolutionary search. It has introduced a new type of compact evolutionary model, Schema Grammar, which overcomes the limitations of positional bias that have hampered the development of compact models. The work has proved the hypothesis that compact models of fit populations can reflect the compositional structure of the selective environment and have strong predictive qualities with respect to fitness. This has established compact models in the state of the art in evolutionary model building. Beyond this, it has shown that the foundational idea of the Schema Theorem, BBH and Compositional Evolution – efficiently constructing good solutions to difficult problems by recursively finding and recombining fit schemata together – can be explicitly implemented using SG.

For the first time, it has been shown how it is possible to infer an explicit model of a problem's schema structure with no a priori assumptions. It has also been shown how evolution can use schema structure to guide variation in an explicitly directed way. By combining the new model with MSS methods, an evolutionary search algorithm has been created that is competitive with the state of the art. Moreover, by exploiting the information-theoretic qualities of the model, a new method for quantifying the information content of evolutionary populations has been created. This capability sets SG apart from alternative methods.

We can now revisit our original claims and reflect on the evidence that we have provided for each, as well as the implications.

- **SG can infer a compact model of a sample population without sensitivity to positional bias.**

Canonical compression methods are only capable of detecting structure that is visible in contiguous sequences, rendering them unsuitable as modelling methods in unbiased BBO. This has inhibited further research into compact models. By introducing a lossless compression model that overcomes positional bias, and demonstrating its unique qualities with respect to alternatives, a new paradigm for unbiased evolutionary modelling has been established. Moreover, the potential of compact models has been established and further research in this area has been enabled.

- **By modelling a representative population of fit individuals, SG can infer a schema structure that is consistent with the compositional structure of the selective environment.**

  Since the Schema Theorem and BBH, researchers in the field of evolutionary computation have been inspired by the notion that evolution may be able to learn and then exploit the schema structure of a problem. However, until now, this idea has not been adequately demonstrated. The work in Chapter 4 shows that the SG model is able to capture the compositional schema structure of the underlying problem from a population of fit individuals. This contributes important new knowledge and practical techniques to the pursuit of these foundational ideas.

- **By using SG to infer an exploration distribution that reflects the constraints of phenotype space, global function optimisation performance can be achieved that is comparable with the state of the art in evolutionary algorithms.**

  The results in Chapter 5 establish SG as a state of the art model-building algorithm. But they also have a wider significance. Firstly, they are a powerful demonstration of Toussaint's ideas of "compact models as a search strategy", and "simple variation on a suitable representation" as an alternative to "complex adaptation on arbitrary representations". This creates some tantalising prospects regarding our understanding of evolutionary development and the emergence of complex Gene Regulation Networks (GRN). These prospects are discussed further in Section 7.2.4. Secondly, they demonstrate the potential of using schema structure to guide variation in an explicitly directed way. This potential was always assumed by proponents of the Schema Theorem and BBH, but could not be adequately demonstrated. Finally, they add further support to the emergence of MSS as the pre-eminent approach to evolutionary model-building.

- **By exploiting the information-theoretic properties of SG as a compact model, the information content of evolutionary populations and individuals can be quantified with more accuracy than alternative methods.**

  Luminaries in evolutionary thinking have often described the idea of evolution "learning" about its environment. In the words of Deutsch (2011) "genes embody

knowledge about their niche", and according to Wilson (2001), organisms have "encoded the predictable occurrence of nature's storms in the letters of their genes."[1] This idea is also supported by the behaviour of model-building algorithms, which can often be observed employing relevant knowledge to generalise about fitness. However, until now it has been difficult to measure the *quantity* of this knowledge in all but the simplest problems. The work in Chapter 6 shows that the SG model can be used to measure evolutionary information in a way that respects the complex epistatic nature of problems. This not only breaks new ground for evolutionary model building, it is a novel and significant contribution in its own right.

7.2 Research Directions

7.2.1 Beyond Bit String Representations

In the field of evolutionary model building, comparison of any two models is usually based on performance in global function optimisation. It is usually also subjective. Variables include the class of problem used, algorithm configuration, experimental methodology and how time complexity is measured (absolute time, function evaluations or hill-climbing steps). To compare a new algorithm to an existing one, one approach is to use problems and a set of parameters that suit the new algorithm, creating a new problem class if necessary. This has the benefit of being able to clearly illustrate something that the new algorithm can do that the others cannot. The downside is that you are competing on your own (implicitly favoured) terms. The alternative, which is the approach taken by this thesis, is to adopt the parameters favoured by the original literature. This has the potential benefit of being able to show competitive performance against a benchmark on its own terms. The downside is that it is sometimes not clear what the new algorithm can do that the others cannot.

The results in Chapter 5 illustrate this dichotomy. The SG methods use a binary GA "bit-string" representation and compete with the state of the art on a subset of problem classes that the compared methods have already reported results for. The SG Schema Search algorithm does well; offering similar, and in cases, arguably superior performance to LTGA on a variety of problems. This establishes SG as a credible alternative to the state of the art, but in themselves the performance results do not afford it a special significance.

Beyond the qualities of SG being able to infer explicit schema structure and quantify evolutionary information, how *could* further research demonstrate the special significance

[1] These eloquent descriptions of information in evolutionary systems were originally collected by Adami (2002).

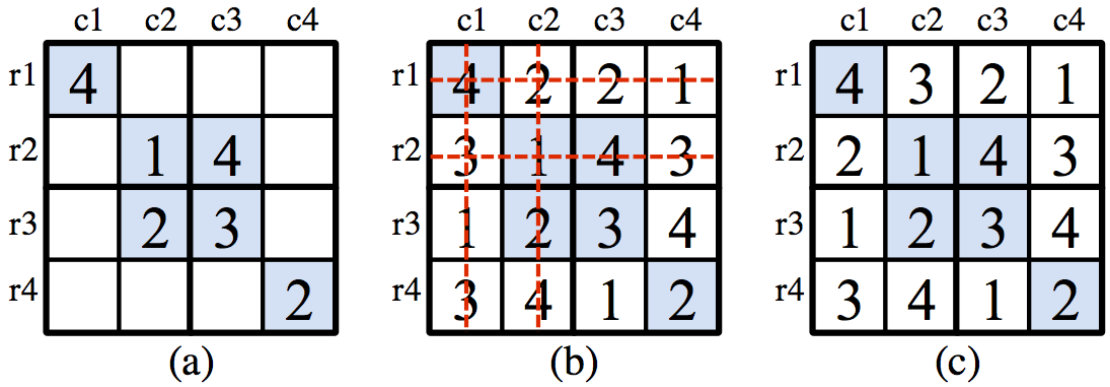

FIGURE 7.1: An example order-2 Sudoku puzzle. The objective is for each row, column and 2x2 sub puzzle to contain the digits 1-4. (a) Puzzles are defined by a subset of digits that are fixed into place. (b) A candidate phenotype: 4 constraints are violated reducing fitness. (c) The global optimum with no constraints violated.

of SG for global function optimisation? One of the advantages of SG is its flexibility to model all kinds of problem "languages". In principle, any individual that can be specified as a set of discrete symbols can be modelled and compressed using the grammar. There are also no constraints on the length or the subset of symbols used in the encoding of each individual. Compression is generic, and will deal with whatever is there. The fact that the model is lossless also means that inference of individual symbols cannot over-generalise. If, for example, a non-terminal symbol is inferred that yields a 10 symbol schema, then there are at least two samples where those 10 symbols appear together. In this context, binary GA "bit-string" representations are only a small subset of the space of possibilities. We now consider some examples that illustrate the alternative problem representations that the model is capable of dealing with. These examples suggest a variety of interesting directions for future work.

7.2.1.1 Sudoku

Sudoku is a popular NP-complete puzzle played on a square grid (Takayuki & Takahiro, 2003). The order of the puzzle is given by s and is usually 3, although in the example here, we use $s = 2$ for brevity. The sudoku puzzle grid is composed of s^2 rows and s^2 columns, and divided into square sub-grids each of which containing s rows and s columns. An example $s = 2$ puzzle is shown in Figure 7.1. Each box on the grid must contain a digit from the alphabet $D = \{1, 2, \ldots, s^2\}$ – in the example this is equal to the digits 1 to 4. The objective of the game is to complete the grid so that the contents of every row, column and sub grid are equal to D, which requires the digits to form a Latin square in which no digit appears more than once in any row or column. Each problem instance is defined by a subset of cells containing immutable digit values. Solving the puzzle requires digits to be assigned to the remaining cells without violating

the constraints. Problems are set so that only one such solution exists. An example instance is shown in Figure 7.1(a).

It is straightforward to model sudoku phenotypes using the SG set representation. We can define the terminal symbols of the grammar as the set of all possible digit and cell combinations, each of the form $\langle \textsc{Row}, \textsc{Column}, \textsc{Digit} \rangle$. For example the tuple $\langle 1, 2, 4 \rangle$ represents a digit 4 at row 1 column 2. If we define the immutable cells of each puzzle as F, then the terminal symbols are given by:

$$\Sigma = \{\langle r, c, d \rangle : r, c, d \in D\} \setminus F \tag{7.1}$$

This scheme allows each phenotype to be specified as an unordered set of $s^4 - |F|$ symbols. The global optimum of the example puzzle shown in Figure 7.1(c) can then be specified as a phenotype $\{\langle 1, 2, 3 \rangle, \langle 1, 3, 2 \rangle, \cdots, \langle 4, 2, 4 \rangle, \langle 4, 3, 1 \rangle\}$. It is straightforward to see how schemata can be modelled using the set presentation also. Consider an interaction between the 3 free digits on the bottom row of the example. This could be represented using the schema $\{\langle 4, 1, 3 \rangle, \langle 4, 2, 4 \rangle, \langle 4, 3, 1 \rangle\}$.

The SG representation for Sudoku would have to be combined with a variation operator in order for the Schema Search process to be viable. One simple approach is to initialise the game board with each sub-grid randomly populated with the digits 1 to s^2. Each variation would be defined as a subset of terminal symbols, representing a set of digits being placed in specific locations. In order for the game grid to be left consistent upon each mutation, the operator could overwrite any existing digits and use the displaced digits to repair each sub-grid.

7.2.1.2 TSP

Travelling Salesman Problem phenotypes are usually represented as permutations. In a symmetrical TSP problem with $n = 5$ cities, a phenotype might be specified as the sequence $(3, 1, 2, 5, 4)$ which contains an implicit link from city 4 to city 3 to complete the tour. One way of encoding TSP phenotypes is to represent the tour as a combination of edges. Using this scheme, each terminal symbol would be defined as an unordered pair of cities of the form $\{\textsc{CityA}, \textsc{CityB}\}$. A full set of terminal symbols can then be defined using the power set of all subsets of all cities $\mathcal{P}(\{1 \ldots n\})$:

$$\Sigma = \{\alpha : \alpha \in \mathcal{P}(\{1 \ldots n\}) \wedge |\alpha| = 2\} \tag{7.2}$$

For a map with n cities the cardinality of Σ would be equal to $\binom{n}{2}$. Using this representation the example phenotype could be encoded as:

$$\{\{3,1\},\{1,2\},\{2,5\},\{5,4\},\{4,3\}\}$$

This representation is not degenerate and is capable of capturing structure in non-contiguous parts of the tour. For example, consider the two schemata $S_1 = \{\{2,5\},\{1,2\}\}$ and $S_2 = \{\{3,1\},\{4,5\}\}$. In these examples S_1 captures the sequential pattern of cities $(1,2,5)$ or $(5,2,1)$ and S_2 captures a non-sequential pattern of cities: an edge between cities 4 and 5 co-occurring with an edge between cities 3 and 1. Using this symbology, each variation would be defined as a subset of one or more edges yielded by a grammar symbol. Given the way that the symbols are inferred into the grammar, each such variation would be guaranteed to be self-consistent. In other words, each variation would constitute a valid partial tour (not necessarily a contiguous sub-tour). In order for Schema Search to work, the variation operator would have to repair tours as variations (of many orders) are applied by schema search. This is standard in TSP meta-heuristics.

7.2.1.3 Variable Length Representations

SG inference does not require phenotypes of a fixed length. As long as individuals can be represented using sets of terminal symbols, they can be compressed into an instance of SG. This raises the possibility that SG could be used to solve a *class* of problems rather than just an instance. Consider a TSP map containing 100 locations, and a class of TSP problems that includes a problem instance for every subset of the 100 locations. Rather than infer an SG instance for each of these subsets, it would be possible to build a model across the entire problem class using a population of fit tours of various lengths. SG inference would create a grammar representing the language of route-finding on this map.

Using such a model, it would be possible to generalise about good routes to previously unseen problem instances. Schema search could quickly generate a suggested route using schema symbols that yield locations in the problem. This is very similar to the mental model that is used by couriers every day. Given a subset of previously unseen locations, a courier can quickly construct a good route around the locations using knowledge built up over time and many different lengths of route. The same principle could be used on other problems.

7.2.2 New Applications of Evolutionary Information Measures

In Chapter 6 the SG model was used to create measures of population model information content (PMIC) and information distance between an individual and a population (DTP). These measures were used to provide quantitative support for the overarching

hypothesis that compact models of fit populations can reflect the compositional structure of the selective environment and have strong predictive qualities with respect to fitness. Beyond this, other useful applications exist. One example is to use the PMIC measure for tracking the progress of an evolutionary algorithm. As evolutionary adaptation progresses, the PMIC measure would be expected to rise. This would offer an alternative to tracking the distribution of population fitnesses and would work even if fitness was not directly observable.

Another possible application of PMIC is as a measure of problem difficulty. Existing measures such as autocorrelation (Weinberger, 1990) and fitness-distance correlation (Jones & Forrest, 1995) regard difficulty in terms of the shape of the bit-flip fitness landscape. The limitation with this approach is that there are many contexts, some of which are described in this thesis, where a well-adapted evolutionary search process will use a more exotic variation neighbourhood. This renders difficulty measures predicated on a rigid view of variation as invalid. As an alternative, fit individuals (not necessarily bit-flip local optima) could be selected from a landscape and compressed with SG to create a PMIC measure. This would give a measure of difficulty based on the similarity between fit individuals, but derived from an effective variation distribution for evolutionary search that can include complex interaction.

The DTP measure could also be used for tracking evolutionary progress by measuring the distance from a population to an objective (e.g., a global optimum). Another possible application is a generative mechanism for EDA-style search. In the Schema Search algorithm, the grammar's symbols are used as a library of neighbourhood operators in systematic trial and error mutation. An alternative approach to variation is to generate new offspring by "sampling" the SG model and then subjecting them to survivor selection. This is the approach taken by most PMBGAs including BOA. In the SG case, a process akin to "reverse parsing" could be used to perform this sampling. The idea is to generate random new offspring that would be well compressed by the model – in other words, they would have short parses and a short information distance to the population. A probability distribution for sampling schema symbols that reflects compressibility could be created using the unnormalised entropy of each symbol occurrence and the size of the schemata they yield. This could also be parameterised so that the average information distance of each new offspring fell within a desired distribution, allowing the balance of exploration and exploitation to be tuned.

The theory underlying PMIC and DTP could be used to create other useful measures. One example is to create a measure of mutual information between two populations of potentially different sizes, A and B. SG compression could be combined with simple complexity-theoretic algebra to estimate absolute mutual information: $K(A) + K(B) - K(AB)$. It could also be used to create a relative measure by dividing this through by $K(AB)$.

7.2.3 New Inference Methods

In Section 3.4 an offline inference algorithm for SG was defined. This algorithm is just one possible algorithm for inferring instances of SG. The offline algorithm is based on the RE-PAIR sequential lossless compression algorithm (Larsson & Moffat, 2000). This algorithm is simple and compresses data well, but online algorithms like LZ77 (Ziv & Lempel, 1977) and SEQUITUR (Nevill-Manning & Witten, 1997) are faster, use less memory, and are favoured in practice. In the same way as RE-PAIR was adapted for SG, it is likely that many alternative compression algorithms could also be adapted. As an example, consider the following outline sketch for an online compression algorithm.

When some time is spent examining what SG compression really does, the realisation is that it essentially works by identifying and factorising the overlap between phenotypes. Consider the following two binary individuals:

$$P_1 = 110011$$
$$P_2 = 000000$$

Compression of this population with the offline algorithm yields the following grammar:

Production rule	$\stackrel{*}{\Rightarrow}$
$g_1 \rightarrow \{x_1^1, x_2^1, x_5^1, x_6^1\}$	110011
$g_2 \rightarrow \{x_1^0, x_2^0, x_5^0, x_6^0\}$	000000
$s_1 \rightarrow \{x_3^0, x_4^0\}$	**00**

s_1 encodes the overlap between the two phenotypes. In set-theoretic terms this overlap is just the intersection between P_1 and P_2:

$$s_1 \rightarrow P_1 \cap P_2$$

The benefit of inferring s_1 in this way is that the operation is $O(n)$. In contrast, offline compression will use the frequency of co-occurring symbols in the population, which is generated in $O(n^2)$. It turns out that it is possible to infer a complete SG instance solely using intersection rather than frequency of co-occurrence. The algorithm is online, meaning it builds the SG model iteratively one sample at a time. It is very similar to the parsing operation defined in Algorithm 4.

When a sample phenotype is added to the grammar, a new non-terminal genotype symbol is created with a production rule that yields the phenotype directly. Then, the production rules of the grammar are iterated in order. At each step the set intersection between the new rule and the compared rule is taken. If the intersection is equal to the compared rule the relevant symbols are replaced in the new rule and iteration continues. This is equivalent to a single step of parsing. If the intersection is a subset of the

compared rule and larger than a single symbol, then a compression benefit exists in creating a new schema symbol. A symbol and rule are created which yield the intersecting symbols. The rule is then inserted into the rule table before the current location. The new non-terminal symbol is substituted into both the new rule and the compared rule and iteration continues. As an example, consider the following 4 phenotypes, which will be compressed in order:

$$P_1 = 111000$$
$$P_2 = 111111$$
$$P_3 = 000000$$
$$P_4 = 111101$$

At step 1 the grammar is empty, so P_1 is added to the grammar as a new genotype symbol:

Step 1

Production rule	$\stackrel{*}{\Rightarrow}$
$g_1 \rightarrow \{x_1^1, x_2^1, x_3^1, x_4^0, x_5^0, x_6^0\}$	111000

At step 2 a new rule for P_2 is created: $g_2 \rightarrow \{x_1^1, x_2^1, x_3^1, x_4^1, x_5^1, x_6^1\}$. Then, the existing rules in the grammar are iterated. In this case, there is a single rule for g_1. The intersection of g_1 and g_2 is taken. This yields:

$$g_1 \cap g_2 = \{x_1^1, x_2^1, x_3^1\}$$

As the intersection is a subset of the production rule for g_1 and greater than one symbol in size, a new schema symbol is created and inserted into the table. The new schema symbol is then substituted into both rules. Iteration is then finished, and the new rule is added to the end of the table. This gives the following:

Step 2

Production rule	$\stackrel{*}{\Rightarrow}$
$s_1 \rightarrow \{x_1^1, x_2^1, x_3^1\}$	111***
$g_1 \rightarrow \{s_1, x_4^0, x_5^0, x_6^0\}$	111000
$g_2 \rightarrow \{s_1, x_4^1, x_5^1, x_6^1\}$	111111

At step 3 P_3 is compressed. There is no intersection with s_1, but there is with g_1. Following the same process as above, yields the following grammar:

Step 3

Production rule	$\stackrel{*}{\Rightarrow}$
$s_1 \rightarrow \{x_1^1, x_2^1, x_3^1\}$	111***
$s_2 \rightarrow \{x_1^1, x_2^1, x_3^1\}$	***000
$g_1 \rightarrow \{s_1, s_2\}$	111000
$g_2 \rightarrow \{s_1, x_4^1, x_5^1, x_6^1\}$	111111
$g_3 \rightarrow \{s_2, x_1^0, x_2^0, x_3^0\}$	000000

At the final step P_4 is compressed by the algorithm. During iteration there is an intersection between g_4 and s_1. In this case the intersection is simply equal to s_1, therefore the relevant symbols in g_4 are replaced by s_1. This gives $g_4 \rightarrow \{s_1, x_4^1, x_5^0, x_6^1\}$. Iteration then continues. There is also an intersection with g_2, creating a new schema symbol. The final version of the grammar is as follows:

Step 4

Production rule	$\stackrel{*}{\Rightarrow}$
$s_1 \rightarrow \{x_1^1, x_2^1, x_3^1\}$	111***
$s_2 \rightarrow \{x_1^1, x_2^1, x_3^1\}$	***000
$g_1 \rightarrow \{s_1, s_2\}$	111000
$s_3 \rightarrow \{s_1, x_4^1, x_6^1\}$	1111*1
$g_2 \rightarrow \{s_3, x_5^1\}$	111111
$g_3 \rightarrow \{s_2, x_1^0, x_2^0, x_3^0\}$	000000
$g_4 \rightarrow \{s_3, x_5^0\}$	111101

The online algorithm sketched above performs each intersection operation in $O(n)$. When encoding a complete population, each time a sample is added to the grammar, the existing encoded population is scanned, the size of which grows on each iteration. This operation is triangular in nature, giving scalability of $O(p^2)$ where p is the size of the population. Overall then, we can estimate the overall time complexity of model building as $O(np^2)$. If confirmed, this would mean that model building is *linear* in problem size yet able to capture multivariate interactions of many orders. This would be something of a first for evolutionary model building. Therefore, developing this and associated algorithms is an important direction for ongoing work.

7.2.4 Theoretical Biology

In this thesis, we have considered models over a population of co-existing individuals at a single point in evolutionary time. Although these models can be used to guide evolutionary search to great effect, on first consideration they seem to offer very little in terms of biological insight. After all, how could a population-wide model be realised in the biological domain? One alternative perspective is to consider the phylogenetic

history of a single species as a population from which a model may emerge. The idea here is that evolution could learn the constraints of the environment by observing phenotypic patterns that have been selected for over evolutionary time, then encode these into developmental processes.

Research has recently been published that shows this idea in action, using GRN models that coevolve with an asexual species over evolutionary time (Watson et al., 2014). The results show that evolution can develop a "memory" of fit phenotypic patterns that are then "recalled" during the development of an offspring, demonstrating a cycle of implicit model-building and model-guided variation. Furthermore, evolutionary adaptation of the GRN is shown to be Hebbian in nature (Hebb, 1949) and analogous to the type of learning found in neural networks. This work suggests a deep relationship between evolvability and machine learning. It also shows that our scientific understanding of evolutionary model building can inform our scientific understanding of evolvability.

This then relates back to Toussaint's concept of compact models as a search strategy (Toussaint, 2006) and the idea that developmental systems may evolve to form a compact encoding of phenotype space. In this thesis we have demonstrated the potential of this idea persuasively in the context of population modelling and optimisation. A significant contribution could be made by showing how the same ideas can be applied to studies of evolvability. There is plenty to build on in this regard. This not only includes the work of Toussaint, but the study of L-Systems (Lindenmayer, 1968; Prusinkiewicz et al., 1990), which suggests a deep relationship between grammatical encoding and biological development.

7.2.5 Machine Learning

Many of the ideas realised by this thesis could be applied outside the domain of evolutionary computation. SG finds patterns in populations of objects that are defined in terms of sets of symbols. It is straightforward to see how this framework could model data sets in machine learning tasks. Consider a table of data, with each row defined over a set of discrete variables. In the same way that we considered an individual to be a set of distinct alleles, we can also consider a row to be a set of distinct attributes. We can compress the table with SG in the same way as we compress an evolutionary population. The resulting grammar could reveal the schema structure of the data set in the same way as SG can reveal the schema structure of the selective environment, creating an alternative to existing association mining methods.

The theory and algorithms described in Chapter 6 would also apply to more generic machine learning representations. So, just as we could measure the absolute mutual information between two evolutionary populations using SG, we could also measure the similarity between two data sets. And just as grammatical parsing can be used to predict

the fitness of a previously unseen individual, it could also be used to classify a test set in a machine learning task. Consider an image divided into a set of discrete pixels. We can create a set of terminal symbols by quantising each pixel, and describe each sample using a subset of these. Inferring an SG instance over a population of images representing a concept will infer a graphical language for this concept. Then, a test set of images could be classified using SG parsing and DTP of each image.

Similar opportunities have occurred for sequential compression, with applications in clustering (e.g. Christen et al., 2004; Cilibrasi & Vitányi, 2005), classification (e.g. Ferragina et al., 2007; Alvarado et al., 2012) and a host of works in bioinformatics (Giancarlo et al., 2009). All of these advances have exploited the relationship between compression and KC, but have been subject to the constraints of sequential compression. If the concept of combinatorial compression introduced by SG can be developed sufficiently, then it might be possible to apply compact models to a host of new machine learning applications.

Bibliography

Adami, C. (2002). What is complexity? *BioEssays*, *24*, 1085–1094.

Adami, C., & Cerf, N. J. (2000). Physical complexity of symbolic sequences. *Physica D: Nonlinear Phenomena*, *137*(1), 62–69.

Adami, C., Ofria, C., & Collier, T. C. (2000). Evolution of biological complexity. *Proceedings of the National Academy of Sciences*, *97*(9), 4463–4468.

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, & C. Zaniolo (Eds.) *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, (pp. 487–499). Morgan Kaufmann.

Alvarado, A. S., Lakshminarayan, C., & Principe, J. C. (2012). Time-based compression and classification of heartbeats. *IEEE Transactions on Biomedical Engineering*, *59*(6), 1641–1648.

Baluja, S. (1994). Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. CMU-CS-94-163, Carnegie Mellon University, Department of Computer Science.

Baluja, S., & Davies, S. (1997). Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Tech. Rep. No. CMU-CS-97-107, Carnegie Mellon University, Department of Computer Science.

Bedau, M. A. (2009). The evolution of complexity. In A. Barberousse, M. Morange, & T. Pradeu (Eds.) *Mapping the Future of Biology*, (pp. 111–130). Springer Netherlands.

Bennett, C. H., Gács, P., Li, M., Vitányi, P. M., & Zurek, W. H. (1998). Information distance. *IEEE Transactions on Information Theory*, *44*(4), 1407–1423.

Bertsekas, D. P. (2003). *Convex analysis and optimization*. Belmont: Athena Scientific.

Bethke, A. D. (1980). *Genetic algorithms as function optimizers*. Ph.D. thesis, University of Michigan.

Beyer, H.-G. (1997). An alternative explanation for the manner in which genetic algorithms operate. *Biosystems*, *41*(1), 1–15.

Blizard, W. D. (1988). Multiset theory. *Notre Dame Journal of Formal Logic*, *30*(1), 36–66.

Bosman, P. A. N., & Thierens, D. (1999). Linkage information processing in distribution estimation algorithms. In W. Banzhaf, A. E. Daida, J abd Eiben, M. H. Garzon, V. Honavar, M. Jakiela, & R. E. Smith (Eds.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, (pp. 60–67). Morgan Kaufmann.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. MIT Press.

Charikar, M., Lehman, E., & Liu, D. (2005). The smallest grammar problem. *IEEE Transactions on Information Theory*, *51*(7), 2554–2576.

Chomsky, N. (1999). Derivation by phase. MIT, Department of Linguistics.

Christen, M., Ott, T., & Stoop, R. (2004). Spike train clustering using a Lempel-Ziv distance measure. In *Proceedings of the International Symposium on Nonlinear Theory and its Applications*, (pp. 379–382).

Cilibrasi, R., & Vitányi, P. M. (2005). Clustering by compression. *IEEE Transactions on Information Theory*, *51*(4), 1523–1545.

Cox, C. R., & Watson, R. A. (2014a). Inferring and exploiting problem structure with schema grammar. In T. Bartz-Beielstein, J. Branke, B. Filipič, & J. Smith (Eds.) *Parallel Problem Solving from Nature PPSN XIII*, (pp. 404–413). Springer International Publishing.

Cox, C. R., & Watson, R. A. (2014b). Solving building block problems using generative grammar. In C. Igel (Ed.) *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, (pp. 341–348). ACM.

De Bonet, J. S., Isbell, C. L., & Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. In M. Mozer, M. I. Jordan, & T. Petsche (Eds.) *Advances in Neural Information Processing Systems*, (pp. 424–430). MIT Press.

de Jong, E. D., Thierens, D., & Watson, R. A. (2004). Defining modularity, hierarchy, and repetition. In K. Deb, R. Poli, W. Banzhaf, H. Beyer, E. K. Burke, P. J. Darwen, D. Dasgupta, D. Floreano, J. A. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, & A. M. Tyrrell (Eds.) *Genetic and Evolutionary Computation—GECCO 2004*, (pp. 2–6). Springer Berlin Heidelberg.

Deb, K., & Goldberg, D. E. (1993). Analyzing deception in trap functions. In L. D. Whitley (Ed.) *Foundations of Genetic Algorithms*, vol. 2, (pp. 98–108). Morgan Kaufmann.

Deutsch, D. (2011). *The fabric of reality*. Penguin UK.

Droste, S., Jansen, T., & Wegener, I. (2002). Optimization with randomized search heuristics—the (A) NFL theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, *287*(1), 131–144.

Ebeling, W., & Jiménez-Montaño, M. A. (1980). On grammars, complexity, and information measures of biological macromolecules. *Mathematical Biosciences*, *52*(1), 53–71.

Eshelman, L. J., Caruana, R. A., & Schaffer, J. D. (1989). Biases in the crossover landscape. In J. D. Schaffer (Ed.) *Proceedings of the Third International Conference on Genetic Algorithms*, (pp. 10–19). Morgan Kaufmann.

Ferragina, P., Giancarlo, R., Greco, V., Manzini, G., & Valiente, G. (2007). Compression-based classification of biological sequences and structures via the universal similarity metric: Experimental assessment. *BMC Bioinformatics*, *8*(1), 252.

Forrest, S., & Mitchell, M. (1993). Relative building-block fitness and the building-block hypothesis. In L. D. Whitley (Ed.) *Proceedings of the Second Workshop on Foundations of Genetic Algorithms*, (pp. 109–126). Morgan Kaufmann.

Giancarlo, R., Scaturro, D., & Utro, F. (2009). Textual data compression in computational biology: A synopsis. *Bioinformatics*, *25*(13), 1575–1586.

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal deceptive problem. In L. Davis (Ed.) *Genetic Algorithms and Simulated Annealing*, (p. 216). Morgan Kaufmann.

Goldberg, D. E. (1988). Genetic algorithms and Walsh functions: Part I, a gentle introduction. Tech. Rep. TCGA Report No. 88006, Tuscaloosa, AL: Department of Engineering Mechanics, University of Alabama.

Goldberg, D. E. (1989a). Genetic algorithms and Walsh functions: Part II, deception and its analysis. *Complex Systems*, *3*(2), 153–171.

Goldberg, D. E. (1989b). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional.

Goldberg, D. E. (1991). Genetic algorithms as a computational theory of conceptual design. *Applications of Artificial Intelligence in Engineering VI*, (pp. 3–16).

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, *3*(5), 493–530.

Goldberg, D. E., & Rudnick, M. (1991). Genetic algorithms and the variance of fitness. *Complex Systems*, *5*(3), 265–278.

Goldman, B. W., & Punch, W. F. (2014). Parameter-less population pyramid. In C. Igel (Ed.) *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, (pp. 785–792). ACM.

Goldman, B. W., & Tauritz, D. R. (2012). Linkage tree genetic algorithms: Variants and analysis. In T. Soule (Ed.) *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Conference*, (pp. 625–632). ACM.

Grunwald, P., & Vitanyi, P. (2004). Shannon information and Kolmogorov complexity. *arXiv.org*.

Grünwald, P. D. (2007). *The minimum description length principle*. MIT Press.

Harik, G. R. (1999). Linkage learning via probabilistic modeling in the ECGA. Tech. Rep. IlliGAL Report No. 99010, Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1998). The compact genetic algorithm. In *Proceedings of the International Conference on Evolutionary Computation (ICEC'98)*, (pp. 523–528). IEEE Press.

Harik, G. R., Lobo, F. G., & Sastry, K. (2006). Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ecga). In M. Pelikan, K. Sastry, & E. Cantú-Paz (Eds.) *Scalable optimization via probabilistic modeling*, (pp. 39–61). Springer.

Hauschild, M., & Pelikan, M. (2011). An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, *1*(3), 111–128.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological approach*. John Wiley & Sons.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. MIT Press.

Holland, J. H. (2000). Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, *8*(4), 373–391.

Huffman, D. A. (1952). A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, *40*(9), 1098–1101.

Iclanzan, D., & Dumitrescu, D. (2007). Overcoming hierarchical difficulty by hill-climbing the building block structure. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, (pp. 1256–1263). ACM.

Igel, C., & Toussaint, M. (2003). On classes of functions for which no free lunch results hold. *Information Processing Letters*, *86*(6), 317–321.

Jones, T., & Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, (pp. 184–192). Morgan Kaufmann.

Kauffman, S. A. (1993). *The origins of order*. Self-organization and Selection in Evolution. Oxford University Press.

Kieffer, J. C., & Yang, E. (2000). Grammar-based codes: A new class of universal lossless source codes. *IEEE Transactions on Information Theory*, *46*(3), 737–754.

Kolmogorov, A. N. (1968). Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, (pp. 157–168).

Larsson, N. J., & Moffat, A. (2000). Off-line dictionary-based compression. *Proceedings of the IEEE*, *88*(11), 1722–1732.

Lehre, P. K., & Witt, C. (2012). Black-box search by unbiased variation. *Algorithmica*, *64*(4), 623–642.

Li, M., Chen, X., Li, X., Ma, B., & Vitányi, P. M. (2004). The similarity metric. *IEEE Transactions on Information Theory*, *50*(12), 3250–3264.

Li, M., & Vitányi, P. M. B. (2009). *An introduction to Kolmogorov complexity and its applications*. Springer Science & Business Media.

Liepins, G. E., & Vose, M. D. (1991). Deceptiveness and genetic algorithm dynamics. In G. J. E. Rawlins (Ed.) *Foundations of Genetic Algorithms*, vol. 1, (pp. 36–52). Morgan Kaufmann.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development: I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, (pp. 280–299).

Madeira, S. C., & Oliveira, A. L. (2004). Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *1*(1), 24–45.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, *2*, 49–55.

Mckay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y., & O'Neill, M. (2010). Grammar-based genetic programming: A survey. *Genetic Programming and Evolvable Machines*, *11*(3-4), 365–396.

Mills, R. (2010). *How micro-evolution can guide macro-evolution: Multi-scale search via evolved modular variation*. Ph.D. thesis, University of Southampton.

Mills, R., Jansen, T., & Watson, R. A. (2014). Transforming evolutionary search into higher-level evolutionary search by capturing problem structure. *IEEE Transactions on Evolutionary Computation*, *18*(5), 628–642.

Mitchell, M., Holland, J. H., & Forrest, S. (1993). When will a genetic algorithm outperform hill climbing? In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.) *Advances in Neural Information Processing Systems 6*, (pp. 51–58). Morgan Kaufmann.

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., & Troyansky, L. (1999). Determining computational complexity from characteristic 'phase transitions'. *Nature*, *400*(6740), 133–137.

Mühlenbein, H. (1992). How genetic algorithms really work: I. mutation and hillclimbing. In B. Manderick (Ed.) *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, (pp. 15–26). Elsevier Science Inc.

Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions: I. binary parameters. In W. Ebeling, I. Rechenberg, H. P. Schwefel, & H. M. Voigt (Eds.) *Parallel Problem Solving from Nature PPSN IV*, (pp. 178–187). Springer Berlin Heidelberg.

Nevill-Manning, C. G., & Witten, I. H. (1997). Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, *7*, 67–82.

Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, *103*(23), 8577–8582.

Nicolau, M., & Ryan, C. (2006). Solving sudoku with the gAuGE system. In *Genetic Programming*, (pp. 213–224). Springer Berlin Heidelberg.

Nordin, P., & Banzhaf, W. (1995). Complexity compression and evolution. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, (pp. 310–317). Morgan Kaufmann.

O'Neill, M., & Brabazon, A. (2005). mGGA: The meta-grammar genetic algorithm. *Genetic Programming*, (pp. 311–320).

O'Neill, M., & Ryan, C. (2004). Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code. In *Genetic Programming*, (pp. 138–149). Springer Berlin Heidelberg.

Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: A review. *ACM SIGKDD Explorations Newsletter*, *6*(1), 90–105.

Pelikan, M. (2002). *Bayesian optimization algorithm: From single level to hierarchy*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Pelikan, M. (2008). Analysis of estimation of distribution algorithms and genetic algorithms on NK landscapes. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, (pp. 1033–1040). ACM.

Pelikan, M. (2010). NK landscapes, problem difficulty, and hybrid evolutionary algorithms. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, (pp. 665–672). ACM.

Pelikan, M., & Goldberg, D. E. (2000). Hierarchical problem solving by the bayesian optimization algorithm. Tech. Rep. IlliGAL Report No. 2000002, Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pelikan, M., & Goldberg, D. E. (2001). Escaping hierarchical traps with competent genetic algorithms. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H. M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, & E. Burke (Eds.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, (pp. 511–518). Morgan Kaufmann.

Pelikan, M., & Goldberg, D. E. (2003). Hierarchical BOA solves Ising spin glasses and MAXSAT. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Lawrence, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. Potter, A. Schultz, N. Jonoska, K. Dowsland, & J. Miller (Eds.) *Genetic and Evolutionary Computation—GECCO 2003*, (pp. 1271–1282). Springer Berlin Heidelberg.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The bayesian optimization algorithm. In *Genetic and Evolutionary Computation Conference*, (pp. 525–532). Morgan Kaufmann.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (2000). Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary Computation*, *8*(3), 311–340.

Pelikan, M., Goldberg, D. E., & Lobo, F. G. (2002). A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, *21*(1), 5–20.

Pelikan, M., & Hauschild, M. W. (2012). Distance-based bias in model-directed optimization of additively decomposable problems. In T. Soule (Ed.) *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Conference*, (pp. 273–280). ACM.

Pelikan, M., Hauschild, M. W., & Thierens, D. (2011). Pairwise and problem-specific distance metrics in the linkage tree genetic algorithm. In N. Krasnogor (Ed.) *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, (pp. 1005–1012). ACM.

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, & P. K. Chawdry (Eds.) *Advances in Soft Computing*, (pp. 521–535). Springer London.

Pelikan, M., Sastry, K., Goldberg, D. E., Butz, M. V., & Hauschild, M. (2009). Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, (pp. 851–858). ACM.

Prugel-Bennett, A. (2007). Finding critical backbone structures with genetic algorithms. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, (pp. 1343–1348). ACM.

Prugel-Bennett, A. (2010). Benefits of a population: five mechanisms that advantage population-based algorithms. *IEEE Transactions on Evolutionary Computation*, *14*(4), 500–517.

Prusinkiewicz, P., Lindenmayer, A., & Hanan, J. (1990). *The algorithmic beauty of plants*. Springer Verlag.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*(5), 465–471.

Rowe, J. E., & Vose, M. D. (2011). Unbiased black box search algorithms. In N. Krasnogor (Ed.) *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, (pp. 2035–2042). ACM.

Ryan, C., Collins, J. J., & Neill, M. O. (1998). Grammatical evolution: Evolving programs for an arbitrary language. *Genetic Programming*, (pp. 83–96).

Ryan, C., Nicolau, M., & O'Neill, M. (2002). Genetic algorithms using grammatical evolution. In *Genetic Programming*, (pp. 278–287). Springer Berlin Heidelberg.

Sadowski, K. L., Bosman, P. A., & Thierens, D. (2013). On the usefulness of linkage processing for solving MAX-SAT. In N. Krasnogor (Ed.) *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, (pp. 853–860). ACM.

Sastry, K., Goldberg, D. E., & Llora, X. (2007). Towards billion-bit optimization via a parallel estimation of distribution algorithm. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, (pp. 577–584). ACM.

Saul, L., & Kardar, M. (1994). The 2D± J Ising spin glass: Exact partition functions in polynomial time. *Nuclear Physics B*, *432*(3), 641–667.

Schlosser, G., & Wagner, G. P. (2004). *Modularity in development and evolution*. University of Chicago Press.

Schumacher, C., Vose, M. D., & Whitley, L. D. (2001). The no free lunch and problem description length. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H. M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, & E. Burke (Eds.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, (pp. 565–570). Morgan Kaufmann.

Shakya, S. (2006). *DEUM: A framework for an estimation of distribution algorithm based on Markov random fields*. Ph.D. thesis, The Robert Gordon University, Aberdeen.

Shakya, S., & Santana, R. (2008). An EDA based on local Markov property and Gibbs sampling. In M. Keijzer (Ed.) *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, (pp. 475–476). ACM.

Shakya, S., & Santana, R. (2012). A review of estimation of distribution algorithms and Markov networks. In S. Shakya, & R. Santana (Eds.) *Markov Networks in Evolutionary Computation*, (pp. 21–37). Springer Berlin Heidelberg.

Simon, H. A. (1962). The architecture of complexity. *Proceedings of the American Philosophical Society*, *106*(6), 467–482.

Simovici, D. A., & Djeraba, C. (2014). *Mathematical tools for data mining*. Set Theory, Partial Orders, Combinatorics. Springer Science & Business Media.

Spears, W. M., & De Jong, K. A. (1990). An analysis of multi-point crossover. In G. J. E. Rawlins (Ed.) *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, (pp. 301–315). Morgan Kaufmann.

Stephens, C. R., & Vargas, J. M. (2000). Effective fitness as an alternative paradigm for evolutionary computation I: General formalism. *Genetic Programming and Evolvable Machines*, *1*(4), 363–378.

Takayuki, Y., & Takahiro, S. (2003). Complexity and completeness of finding another solution and its application to puzzles. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *86*(5), 1052–1060.

Tayarani-N, M. H., & Prugel-Bennett, A. (2014). On the landscape of combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, *18*(3), 420–434.

Thierens, D. (1995). *Analysis and design of genetic algorithms*. Ph.D. thesis, Katholieke Universiteit Leuven.

Thierens, D. (1999). Scalability problems of simple genetic algorithms. *Evolutionary Computation*, *7*(4), 331–352.

Thierens, D. (2010). The linkage tree genetic algorithm. In C. Cotta, J. Kolodziej, R. Günter, & R. Schaefer (Eds.) *Parallel Problem Solving from Nature PPSN XI*, (pp. 264–273). Springer Berlin Heidelberg.

Thierens, D., & Bosman, P. (2012). Predetermined versus learned linkage models. In T. Soule (Ed.) *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference*, (pp. 289–296). ACM.

Thierens, D., & Bosman, P. A. (2013). Hierarchical problem solving with the linkage tree genetic algorithm. In C. Blum (Ed.) *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, (pp. 877–884). ACM.

Thierens, D., & Bosman, P. A. N. (2011). Optimal mixing evolutionary algorithms. In N. Krasnogor (Ed.) *Proceedings of the Thirteenth Annual Conference on Genetic and Evolutionary Computation*, (pp. 617–624). ACM.

Toussaint, M. (2006). Compact representations as a search strategy: Compression EDAs. *Theoretical Computer Science*, *361*(1), 57–71.

Toussaint, M., & von Seelen, W. (2007). Complex adaptation and system structure. *Biosystems*, *90*(3), 769–782.

Vitányi, P. M., & Li, M. (2000). Minimum description length induction, Bayesianism, and Kolmogorov complexity. *IEEE Transactions on Information Theory*, *46*(2), 446–464.

Watson, R. A. (2002). *Compositional evolution: Interdisciplinary investigations in evolvability, modularity, and symbiosis*. Ph.D. thesis, Brandeis University, MA. USA.

Watson, R. A. (2006). *Compositional evolution: The impact of sex, symbiosis and modularity on the gradualist framework of evolution*. MIT Press.

Watson, R. A., Hornby, G. S., & Pollack, J. B. (1998). Modeling building-block interdependency. In T. Bäck, A. E. Eiben, M. Schoenauer, & H. Schwefel (Eds.) *Parallel Problem Solving from Nature PPSN V*, (pp. 97–106). Springer Berlin Heidelberg.

Watson, R. A., & Pollack, J. B. (2000). Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In K. Deb, E. Lutton, J. J. Merelo, R. Günter, M. Schoenauer, H.-P. Schwefel, & X. Yao (Eds.) *Parallel Problem Solving from Nature PPSN VI*, (pp. 425–434). Springer Berlin Heidelberg.

Watson, R. A., & Pollack, J. B. (2005). Modular interdependency in complex dynamical systems. *Artificial Life*, *11*, 445–458.

Watson, R. A., Weinreich, M., Wagner, P. G., Pavlicev, M., Weinreich, D. M., & Mills, R. (2014). The evolution of phenotypic correlations and 'developmental memory'. *Evolution*, *68*(4), 1124–1138.

Webber, A. B. (2008). *Formal language: A practical introduction*. Franklin Beedle & Associates.

Weinberger, E. D. (1990). Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, *63*(5), 1–12.

Weinreich, D. M., Lan, Y., Wylie, C. S., & Heckendorn, R. B. (2013). Should evolutionary geneticists worry about higher-order epistasis? *Current Opinion in Genetics & Development*, *23*(6), 700–707.

Weinreich, D. M., Watson, R. A., & Chao, L. (2005). Sign epistasis and genetic constraint on evolutionary trajectories. *Evolution*, *59*(6), 1165–1174.

Whitley, L. D. (1991). Fundamental principles of deception in genetic search. In G. J. E. Rawlins (Ed.) *Foundations of Genetic Algorithms*, vol. 1, (pp. 221–241). Morgan Kaufmann.

Wilson, E. O. (2001). *The diversity of life*. Penguin UK.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Xiang, S., Nie, F., & Zhang, C. (2008). Learning a Mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, *41*(12), 3600–3612.

Yu, T. L., Goldberg, D. E., Sastry, K., Lima, C. F., & Pelikan, M. (2009). Dependency structure matrix, genetic algorithms, and effective recombination. *Evolutionary Computation*, *17*(4), 595–626.

Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, *23*(3), 337–343.