Evolutionary algorithms in artificial intelligence: a comparative study through applications

Nettleton, David John

University of Durham

Evolutionary Algorithms in Artificial Intelligence: A Comparative Study Through Applications.

David John Nettleton

Laboratory for Natural Language Engineering,

Department of Computer Science.

Submitted in partial fulfilment of the
requirements for the degree of

Doctor of Philosophy

Abstract

For many years research in artificial intelligence followed a symbolic paradigm which required a level of knowledge described in terms of rules. More recently subsymbolic approaches have been adopted as a suitable means for studying many problems. There are many search mechanisms which can be used to manipulate subsymbolic components, and in recent years general search methods based on models of natural evolution have become increasingly popular. This thesis examines a hybrid symbolic/subsymbolic approach and the application of evolutionary algorithms to a problem from each of the fields of shape representation (finding an iterated function system for an arbitrary shape), natural language dialogue (tuning parameters so that a particular behaviour can be achieved) and speech recognition (selecting the penalties used by a dynamic programming algorithm in creating a word lattice). These problems were selected on the basis that each should have a fundamentally different interactions at the subsymbolic level.

Results demonstrate that for the experiments conducted the evolutionary algorithms performed well in most cases. However, the type of subsymbolic interaction that may occur influences the relative performance of evolutionary algorithms which emphasise either top-down (evolutionary programming - EP) or bottom-up (genetic algorithm - GA) means of solution discovery. For the shape representation problem EP is seen to perform significantly better than a GA, and reasons for this disparity are discussed. Furthermore, EP appears to offer a powerful means of finding solutions to this problem, and so the background and details of the problem are discussed at length. Some novel constraints on the problem's search space are also presented which could be used in related work. For the dialogue and speech recognition problems a GA and EP produce good results with EP performing slightly better. Results achieved with EP have been used to improve the performance of a speech recognition system.

Acknowledgements

I would like to thank my supervisor Roberto Garigliano for his advice and support throughout the three years over which this research has been conducted.

I am grateful to all members of the LNLE group for their proof reading of, and comments (sarcastic and otherwise) on, this and other work. These include: Rick, Andy, Yang, Casey, Mark, Kevin, Steve, Sengan, Simon and Brett. I am particularly indebted to Nigel, Russell and Jon for their patience in explaining, what must have been to them, very simple aspects of computing.

Financial assistance for this project was received from EPSRC in the form of a studentship award. I would also like to thank the executive officers of St. Aidan's College for awarding me the St Aidan's College Graduate studentship.

Finally, I would like to thank my Mam, Dad and sister Andrea for putting up with me throughout my time at University, and for keeping me constantly supplied with chocolate biscuits.

Declaration

The material contained within this thesis has not previously been submitted for a degree at the University of Durham or any other university. The research reported within this thesis has been conducted by the author unless indicated otherwise.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial Intelligence (AI) lies at the intersection of many disciplines, including computer science, psychology, philosophy, mathematics, engineering and linguistics. Several definitions of AI have been suggested, but for this work that given by Beardon (1989) is adopted:

> AI is the field of research concerned with making machines perform tasks which are generally thought of as requiring human intelligence.

Although AI is aimed at solving problems which appear to require human intelligence it is not suggested that the methods used are identical to those used by humans. The modelling of human mental mechanisms is known as cognitive science. The goals of AI and cognitive science are similar in that they both require a computer program to be able to perform some task. In addition a cognitive model's success is determined by how plausible a model of human mental mechanisms it provides. AI models of 'intelligent' behaviour typically use any means available and make no claim that they are in any way similar to the processes used by humans.

Applications of AI are wide ranging and include (amongst many others) computer vision, game playing, automated reasoning, natural language understanding and expert systems. Although the fields of applications are diverse there is often a common reliance on techniques pertaining to knowledge representation and search.

The subject of knowledge representation is in itself a research area within AI. Many data structures have been suggested as suitable means of encoding information including: slot-and-filler structures, logical formulas and production rules. In deciding which structure is the most suitable for representing the knowledge associated with a particular problem, there are two important features which must be considered (Rich 1990). First of all it must be decided whether the representation is powerful enough to represent all of the required knowledge. A second consideration is whether the representation is able to support a reasoning mechanism capable of inferring the necessary conclusions from the represented knowledge.

Search is a process which can be viewed as the systematic exploration of a space of states which represent solutions to a problem. There are five parts to such a process: the state space, the initial state(s), a characterisation of the goal state(s), the allowed transitions between states, and any information regarding the most useful means of proceeding. Viewed in this way a search algorithm traverses the space of possible solutions using legal moves (which may be guided by heuristics) in an attempt to move from an initial state to a goal state.

The knowledge representation and search method used in tackling a problem must complement each other since a particular representation determines the search space which is to be searched. Some problems, although easily solvable in theory, may be far more difficult to solve in practice if a representation is chosen which results in the need for a greatly increased search. Consider, for example, the problem of determining whether or not an integer is contained within some list of integers. An array, each element of which contains one of the integers, may be used as a suitable data structure with which to represent the list. A linear search may be used to check whether or not some integer is contained within the array. Clearly, however, such a search mechanism is not a very efficient means of approaching the problem. By using a more suitable representation, *e.g.*, a tree, a more efficient search algorithm can be used, *e.g.*, a binary search.

1.1 Symbolic and Subsymbolic Representations

Traditionally a purely symbolic paradigm has been used in AI to represent knowledge. Symbols refer to objects and relations in the domain of interpretation, and the search mechanisms which are used to examine the space of the representation are often heuristic in nature. Such an approach reflects the physical symbol system hypothesis articulated by Newell and Simon (1976, p. 116):

> A physical symbol system has the necessary and sufficient means for general intelligent action.
>
> By "necessary" we mean that any system that exhibits general intelligence will prove upon analysis to be a physical symbol system. By "sufficient" we mean that any physical symbol system of sufficient size can be organized further to exhibit general intelligence. By "general intelligent action" we wish to indicate the same scope of intelligence as we see in human action: that in any real situation behaviour appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some limits of speed and complexity.

The argument is, therefore, that intelligence can be achieved by formal operations which act on symbol structures. A challenge to this approach emerged in the late 1980's with the advent of parallel distributed processing (PDP) (Rumelhart *et al.* 1986; McClelland *et al.* 1986), the paradigm of which contends that a physical symbol system is neither necessary nor sufficient for a system to exhibit intelligence. The philosophy underlying the emergence of PDP is beyond the scope of this thesis and is discussed in detail elsewhere (see, *e.g.*, Luger and Stubblefield (1993)). However, one of the reasons that the PDP approach emerged is the lack of flexibility which is often inherent within a purely symbolic system. For example, an expert system is able to perform perfectly adequately within its domain of application, but should it encounter a problem outside of that domain it will typically be unable to suggest a solution. On the other hand, human experts will attempt to answer the problem to the best of their ability. Rather than arguing for one approach to be applied in all situations Calmet and Campbell (1993, p. 13) state that:

> ... it is widely believed that there are some activities of intelligence (e.g. recognition of multidimensional patterns) where an approach operating at some lower level than a level of description in symbols is more appropriate than the traditional logical-symbolic approach.

A subsymbolic approach to knowledge representation is one in which the emphasis is not on the use of symbols to represent objects and relations, but instead on the collective behaviour produced by the interaction of a number of simple interacting components (Luger and Stubblefield 1993, p. 516). Such a paradigm views knowledge as being represented implicitly in patterns of interaction between components.

Neural networks are perhaps one of the best known examples of a PDP approach. Inspired by biological brains, a neural network is a computational architecture composed of a large collection of simple processing units. The units do not correspond to concepts, and, if examined in isolation, are capable of very little. A neural network is not programmed with information, but is instead trained by exposure to large amounts of data, and typically uses a means of reinforcement to alter the weights (loosely corresponding to the current experience) within the network. Patterns of interaction emerge which represent the network's representation of knowledge.

Luger and Stubblefield (1993, p. 693) argue that neural networks and symbolic AI are simply different models of intelligence, each of which discuss intelligence in a different language. The two approaches ask different questions, propose different answers and interpret any results differently. Although it is hoped that one day a theory may be produced that can link the two approaches this is probably some way off.

To attempt intentionally to solve a problem using a purely symbolic or subsymbolic approach is often not really an attempt at finding the best possible solution, but rather to examine the limits of the approach itself. One of the aims of this thesis is to demonstrate that, for some problems at least, a combination of the symbolic and subsymbolic methods can result in an approach which can enjoy some of the advantages offered by each method.

As an example of such an hybrid approach consider how a computer can be taught to play a good game of chess. The symbolic approach would involve the construction of a set of heuristics which were based on the knowledge that has been accumulated over the centuries for which the game has been played. This would include rules on standard openings, end games, controlling the centre of the board, and using one piece to protect another. A subsymbolic approach would involve the computer learning how to play by participating in a large number of games, and learning from the experiences encountered. In developing a modern chess playing computer a vast amount of standard knowledge may be incorporated, yet adaptation be allowed to occur so that the opponent's strengths and weaknesses can be taken into account.

A symbolic approach is often favoured when the problem can be encoded in terms of a 'convenient' representation. A 'convenient' representation can be thought of as one which has a range of properties which offer advantages that may include (amongst many others): a firm logical base, conciseness and maintainability. However, there are certain facets of knowledge which humans are unable to encode. Reasons for this include: (1) humans don't have the knowledge and so can't explicitly encode it, and (2) the level of knowledge is below that of conscious knowledge (*e.g.*, pattern recognition). In such situations a subsymbolic representation is more appropriate.

1.2 Search

Search methods generally fall into two categories, strong and weak. A strong search method is one which is rich in task-specific knowledge and often contains specialised heuristics that help guide the search. Such a method is often limited in application to the task for which it is designed, and can be of little or no use outside of that domain. However, for appropriate domains, the method is generally efficient at finding appropriate solutions. A weak search method is task independent, but is usually less efficient, because of its lack of knowledge about the domain to which it may be applied.

Symbolic search methods typically use heuristics as a strategy by which a problem space can be selectively searched.. Heuristics guide the search away from less promising areas to those where it is more likely to be successful. Although a good set of heuristics can in many cases efficiently find an optimum solution to a problem they are not infallible.

Examining the mechanisms by which subsymbolic components can be manipulated (search) and selecting that which will perform best for a particular problem has become increasingly important. This is especially so when there are complex interactions between the components. In recent years general search mechanisms based on models of natural evolution have become increasingly popular in attempting to solve many optimisation problems.

1.2.1 Subsymbolic Processing using Evolutionary Algorithms

In attempting to solve problems, the solutions of which are represented in terms of subsymbolic components, some mechanism is needed by which the components can be manipulated (Nettleton and Garigliano 1994e). The aim of the mechanism is to optimise the behaviour of the realisation of the encoded solution within the environment in which it is to be tested. Neural networks offer one approach to the manipulation of subsymbolic components based upon training. Evolutionary algorithms approach the problem in terms of competition between alternative concepts, and aim to optimise the concept's performance with regard to a function that provides a measure of performance within the environment.

Evolutionary algorithms (EAs) model natural evolution and are robust search methods which have been applied to a wide range of problems. By maintaining a population of solutions, an EA is able to exploit those which are promising while exploring other regions of the search space. In this way a parallel search is achieved. New solutions are produced as variations of those which have survived to that point in time, and the worst solutions are probabilistically culled using a "survival of the fittest" strategy (analogous to natural selection). The population iteratively evolves toward optimal solutions. Further details of EAs, together with their philosophical underpinnings, are discussed in Chapter 2.

As has already been stated one of the aims of this thesis is to demonstrate that symbolic and subsymbolic approaches can be successfully combined in attempting to solve problems. The second main aim of this work is to investigate and explain the relative performance of different forms of EAs when applied to subsymbolic manipulation problems for which there are varying types of complex interaction between the subsymbolic components.

1.3 Subsymbolic Interactions

The behaviour of a solution is its response when tested in the corresponding environment. Interactions between the components of an encoded solution can result in complex behaviour of the solution. In some problems the representation can consist of blocks of components, combinations of which can represent whole or partial solutions in their own right. In other cases the blocks are just the individual components themselves. Two types of interaction between components are distinguished.

1. 'Strong' interaction — changing the value(s) of a component (block of components) of an encoded solution can be expected to produce changes in the solution's behaviour.
2. 'Weak' interaction — changing the value(s) of a component (block of components) of an encoded solution can be expected to have little or no effect on the solution's behaviour.

These forms of interaction are qualitative and no metric by which their strength could be measured is provided. Different combinations of the above interactions at the subsymbolic level are possible including:

1. **Strong-Strong:** There is a strong interaction between blocks of components, and a strong interaction between the components within each block.
2. **Strong-Weak:** There is a strong interaction between blocks of components, and a weak interaction between the components within each block.
3. **Strong:** No blocks of components exist, but there is a strong interaction between the components.

There are of course many other possible forms of interaction, but it is felt that the above selection offer a broad enough base for the performance of EAs on them to be worthy of further investigation. Specific examples of problems which have these interactions are now required. The subject of AI offers a field which contains an example of each.

1.4 Example Problems

The problems which are considered are all from the subject of AI. There are several reasons why AI provides a suitable field of study. First of all, much symbolic work has been conducted in AI and some of this provides a framework on which a subsymbolic approach can be built. It is, therefore, only necessary to develop a subsymbolic representation for each of the problems considered, the symbolic one existing already. Secondly, there has been much debate as to whether a symbolic or subsymbolic representation is the most appropriate for many AI problems. This thesis provides some evidence that rather than concentrating on applying a single paradigm an approach that emphasises the interplay between the two can be successful (Garigliano and Nettleton 1994). Finally, it is worth noting that these problems were not selected on a purely *ad hoc* basis, but were chosen because they are problems which have been researched within the Department of Computer Science at the University of Durham (Giles 1990; Jones 1994a; Collingham 1994).

The problems which exhibit the types of interaction between components discussed above are briefly introduced below. Each problem is presented in greater detail in other more appropriate parts of this thesis.

Strong-Strong: Shape representation using Iterated Function Systems (IFSs) — In adopting an IFS representation, the shape to be encoded is represented by primitive shapes (those from which the original is to be reconstructed) which are smaller linearly deformed copies of the shape to be encoded. A symbolic means of manipulating the primitives could be used, but since the primitives can be described in terms of contraction mappings a more flexible subsymbolic (real-valued) representation is adopted.

The contraction mappings interact strongly to determine the shape that is produced. The individual components of any particular contraction mapping also interact strongly.

Strong-Weak: Tuning the parameters of the dialogue module of a large scale natural language processor — A symbolic based theory of dialogue has been proposed which models the structure of a dialogue that can be expected to occur in a particular situation. In order to help order the appropriateness of particular responses to a particular situation a subsymbolic (integer) representation is adopted.
There are blocks of parameters which control the type of response (*e.g.*, happy, angry) and there is a strong interaction between these. However, within each block the interactions are weak with the emphasis being on how a response is carried out and not what is to be said.

Strong: Selecting the penalties used by a dynamic programming algorithm in word lattice creation — A word lattice is a symbolic data structure that can be used at the acoustic matching stage of a speech recognition system. A dynamic programming algorithm is used to assign ranks to elements of the lattice. The dynamic programming algorithm contains penalties for the expected errors and these are represented subsymbolically (real-valued).
Altering the value of each of the penalties results in changes to the ranks of words within the word lattice created. There is, therefore, a strong interaction between the penalties.

In order to examine the 'depth' of an approach to problem solving based upon a hybrid symbolic/subsymbolic representation and the use of EAs, one of the problems (the shape representation problem) is discussed in great detail. This includes: a full account of the underlying theory (Chapter 3); an examination of how this theory can be applied in practice, and a comprehensive review of the literature on other approaches to solving the problem (Chapter 4); an extensive set of experiments (Chapter 5); reductions in the search space of the subsymbolic representation (Chapter 6). The other two problems are from completely different fields (natural language processing and speech recognition), and are used to show the 'width' of the approach. As such these problems are discussed in much less detail.

1.5 Criteria for success

The success of this work will be evaluated in terms of providing evidence for or against the following statements:

1. A combination of symbolic and subsymbolic methods result in an approach which can improve on the current approaches to several problems.
2. Evolutionary algorithms offer an approach to subsymbolic manipulation which is able to overcome different forms of interaction that can occur between the representation's subsymbolic components.

The method adopted in collecting evidence is to take several open problems from different fields of AI, apply the above approaches to them, and evaluate the results achieved.

1.6 Thesis Structure

Chapter 2 outlines the concept of an EA and in particular examines the paradigms underlying the different models of the evolutionary process which have emerged. Genetic algorithms, evolutionary programming and evolution strategies are three of the models which have emerged. Although all are based on evolutionary principles, each place a different emphasis on what drives the evolutionary process. These models and some of their criticisms are discussed, but no details of their implementation are given in this chapter (this being included in later chapters). Two forms of interaction (pleiotropy and polygeny) which can occur between the components of a subsymbolic representation are described, the effects of which account (in part) for some of the results of later chapters.

Chapter 3 is the first of four chapters which discuss in detail the shape representation problem. The problem is discussed in detail to show the 'depth' of the approach adopted. This chapter is the first of two which formalise the problem of using IFSs for shape representation. The chapter is purely theoretical and introduces the mathematics relating to the geometric properties of IFSs. The first two sections of the chapter detail some basic concepts of metric topology, and can be safely skipped by a reader acquainted with such concepts. The metric space on which IFSs are defined is introduced, and the definition of an IFS given. The remainder of the chapter discusses some properties of IFSs which are relevant to their use as a shape representation scheme.

Chapter 4 completes the formal framework necessary for using IFSs in two-dimensional shape representation. In particular two Lemmas show that a good IFS representation can be found when the underlying space is either continuous or discrete. The discrete case is of particular relevance to computer images. Methods for generating a shape which is encoded as an IFS are discussed, and the subsymbolic interaction between an IFS's components is demonstrated pictorially. The remainder of the chapter reviews the literature on using IFSs for shape representation.

Chapter 5 is the first of the experimental chapters of the thesis, and describes how EAs can be applied to the problem of finding an IFS for a shape. As this problem is the one used to demonstrate the 'depth' of the approach, a comprehensive set of experiments are conducted, and their results discussed in detail. Solutions to the problem are represented subsymbolically, and details are given on how a genetic algorithm, evolutionary programming and three hill-climbing algorithms can be applied to the manipulation of the subsymbolic components. Three shape representation problems are considered, and the results of the experiments conducted are presented and discussed.

Chapter 6 is the final chapter of the four which examine in depth the shape representation problem. Several novel constraints are introduced which can be used to reduce the search space for finding an IFS for an arbitrary shape. The constraints are introduced since reducing a problem's search space is one way in which to improve the search process. The constraints introduced although non-trivial are of a low computational complexity, and can be expected to be of use to a range of search algorithms which attempt to find an IFS for an arbitrary shape. The constraints are not applied in the experiments carried out in Chapter 5 since they might bias a comparison between the approaches in favour of evolutionary programming.

Chapter 7 is the first of two chapters which demonstrate the 'width' of the approach adopted. As such the problem examined in this chapter is not discussed in as much detail as that for shape representation. A symbolic theory of dialogue is introduced which is used as the basis for the dialogue module of a large-scale natural language processor. However, so that the appropriateness of responses to a particular situation can be ordered a subsymbolic (integer) representation is incorporated. These parameters govern (in part) the behaviour of the processor and have to date been selected by hand. The chapter outlines an approach to using EAs in the fine-tuning of the parameters so that a particular behaviour can be achieved.

Chapter 8 discusses the second (and final) problem which demonstrates the 'width' of the hybrid symbolic/subsymbolic approach. The concept of a word lattice is introduced — a symbolic data structure which can be used at the acoustic matching stage of a speech recognition system. A dynamic programming algorithm is described which is used to assign ranks to elements of the lattice. The penalties used by the algorithm are subsymbolic (real-valued) in nature and used to be selected by hand. The chapter discusses how EAs can be used to optimise the parameters. Evolutionary programming is then used to select parameters which lead to the improvement of a speech recognition system.

Chapter 9 provides a conclusion to the thesis, the general results of which can be briefly summarised as:

1. The method of adopting a symbolic approach when convenient, and then moving to a subsymbolic approach to allow for greater flexibility and/or fine-

tuning, has been successful applied to several problems. The examples came from the field of AI.

2. Evolutionary algorithms are able to find 'good' solutions to problems which exhibit different forms of interaction at the subsymbolic level.

More specific conclusions are that:

1. Evolutionary programming outperforms a genetic algorithm in finding an IFS representation for several shapes, and offers a powerful means of tackling the problem.

2. When a subsymbolic representation is adopted the search space of IFS encodings for an arbitrary shape can be greatly reduced by imposing non-trivial constraints on the IFS's components. These constraints can be efficiently implemented, and are expected to be of use to a range of search algorithms.

3. Evolutionary algorithms can be used to fine-tune the parameters within the dialogue module of a large scale natural language processor.

4. Evolutionary algorithms provide a means of selecting parameters for use in the dynamic programming phase of a speech recognition system. Results achieved with evolutionary programming have been used to improve the performance of a speech recognition system.

Chapter 2

Evolutionary Algorithms

The chapter begins with a brief review of several subsymbolic search strategies and discusses some of their shortcomings. The concept of an evolutionary algorithm (EA) is introduced, and the relationship between the encoding of a solution and its behaviour discussed. In particular Section 2.3 introduces the effects of pleiotropy and polygeny, which are forms of interaction that can occur between the subsymbolic components of a solution's representation. These effects account (in part) for some of the results given in later parts of this thesis.

Several forms of EA are introduced. These algorithms are inspired by the search process of natural evolution and have been successfully applied to a wide range of problems. Three main streams of EAs have been independently developed: genetic algorithms (Holland 1975; Goldberg 1989), evolutionary programming (Fogel *et al.* 1966; Fogel 1992a) and evolution strategies (recent review by Bäck *et al.* 1991). Although all are based on evolutionary principles, each place a different emphasis on what drives the evolutionary process. The underlying paradigms of these algorithms are compared and contrasted, and the chapter concludes with a summary.

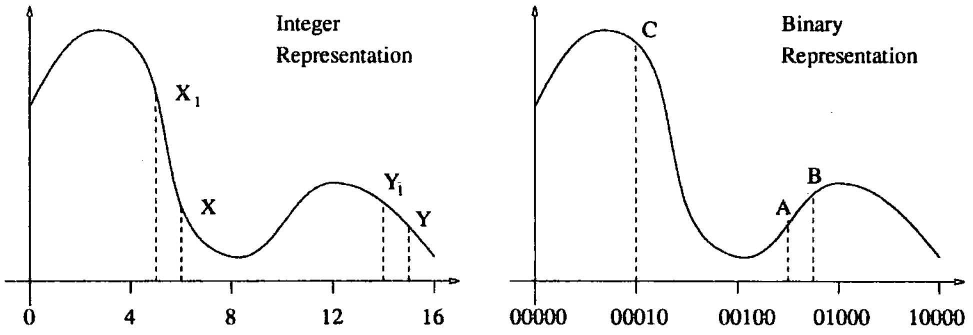
2.1 Introduction

Subsymbolic approaches have been adopted in attempting to solve many AI problems. Such a strategy is appropriate when it is necessary to operate at a level below that of traditional logical-symbolic approaches. In situations such as these a procedure is needed by which the subsymbolic components can be manipulated in order to find a near optimal solution to the problem. This procedure must not only be capable of producing near optimal solutions, it must also be able to do so in an efficient manner. Occasionally some specialised algorithm exists which can carry out this manipulation in the optimal or near optimal number of steps. However, it is often the case that no such procedure is available and some other approach needs to be adopted.

Solutions which are coded by some underlying structure need to be tested in the environment, and some measure of performance returned. The measure of performance allows for a 'fitness landscape' to be envisaged and the aim of the optimisation process is often intuitively considered as finding the solution which corresponds to the 'highest peak' of the landscape. The process of optimisation is often carried out by manipulating the coding of the structure and testing new versions in the environment. Many alternative methods for this manipulation have been suggested. The remainder of this section discusses some of the simplest methods and briefly examines the concept of a fitness landscape.

It is often the case that the search space, in which solutions to a problem exist, is extremely large and complex. For any search algorithm in such a space there exists a fundamental trade-off between exploration and exploitation. An example of a totally exploratory algorithm would be the enumeration of all cases, and selecting the best. This procedure, although eventually yielding the optimum solution, is far too inefficient to be of any practical use in addressing most real-world problems.

Figure 2.1: Example of how hill-climbing is dependent on the underlying representation.



In order for an algorithm to search a space efficiently it must be able to exploit opportunities for improved performance. This often involves making use of information acquired from previous evaluations of possible solutions. If such information is not used then the search could degenerate to the point at which it is little better than a random sampling of the solution space. An example of a search strategy which makes use of current information is a hill-climbing algorithm.

Hill-climbing algorithms concentrate the search effort around the best solution found so far (exploitation), but it is likely that discovered solutions will be suboptimal on non-convex surfaces, because the sequence of trials will stagnate at local optima. Intuitive concepts such as 'peaks' and 'valleys' are often used to describe how such algorithms traverse the fitness landscape. However, such analogies need to be made with care, as Figure 2.1 and the following discussion demonstrates (Spears 1994; Jones 1994b).

The objective of a hill-climbing algorithm is to maximise the (local) fitness, *i.e.*, climb one of the 'peaks'. The normal procedure for this involves making small changes to the solution's representation and accepting as the new best a solution which outperforms the current one. Following this method and using an integer representation, climbs (with reference to Figure 2.1) such as those from $\mathbf{X}$ to $\mathbf{X_1}$, and from $\mathbf{Y}$ to $\mathbf{Y_1}$, would intuitively be how the search would be expected to proceed. With a binary representation a climb from $\mathbb{A}$ to $\mathbb{B}$ would be expected, but what about one from $\mathbb{A}$ to $\mathbb{C}$? In fact both of these are equally likely since $\mathbb{B} = 00111$ and $\mathbb{C} = 00010$ are both exactly one 'small change' (in the solution's encoding) away from $\mathbb{A} = 00110$. This would appear to contradict the idea of a 'hill' climber since a 'valley' appears to have been traversed. The concept of what constitutes a hill-climb is the cause of some debate and is dependent on the representation and operators used.

Simulated annealing (Kirkpatrick *et al.* 1983) is a search algorithm with a natural metaphor. Inspired by the process of annealing crystalline solids the algorithm models the behaviour of thermodynamic state transitions. A starting temperature T_0 is specified and an annealing schedule imposed such that $T \rightarrow 0$. The temperature can be viewed as a control on the random search of the space; the larger the value of T, the larger the expected movement in the search space. One of the important features of simulated annealing is a theory which provides sufficient conditions for asymptotic convergence to the global optimum. The main criticism of the approach is based on the setting of the initial temperature; too low and the algorithm may converge to a sub-optimal solution, too high and the algorithm will be slow.

Natural processes have also inspired a class of search algorithms known as evolutionary algorithms. A feature of both random search and hill-climbing is that they discard much of the information which is presented to them during the course of a search. In order to try and retain some of this information EAs maintain a population of solutions.

EAs are loosely based upon the Darwinian principles of biological evolution and in fact many of the strategies and operators used in their application bear similar names to their biological counterparts, *e.g.*, survival of the fittest, crossover and mutation. An EA creates a set of possible (usually randomly generated) solutions to the problem under consideration and calls this the initial generation. Successive generations are produced via a series of operators which act on the previous generation. The operators produce new solutions which are variations of those that have survived to that point, and probabilistically culls the worst using a "survival of the fittest" strategy. This process continues until the desired number of generations has been completed. The solutions in the final generation are in effect the 'answers' to the search problem. The main advantage of EAs is their ability to be able to consider many solutions 'at the same time' and from these solutions produce other solutions that converge to the optimal.

Over the past 30 years, three main streams of evolutionary algorithm have been independently developed: genetic algorithms (Holland 1975; Goldberg 1989), evolutionary programming (Fogel *et al.* 1966; Fogel 1992a), and evolution strategies (recent review by Bäck *et al.* 1991). Each of these have resulted in robust optimisation techniques that have been successfully applied to a wide range of problems. The differences between the EAs stem from the primary forces modelled from natural evolution. The underlying philosophies of EAs and how they model natural evolution are discussed later in this chapter.

2.2 Outline of an Evolutionary Algorithm

The generic shell for an EA is given by Angeline (1993, p. 26) as:

```
 function Evolutionary-Algorithm(population, size);
 begin
     for i from 1 to size do
         fitness[i] := evaluate(population[i]);
     while not(Tester(bestof(population, fitness))) do
     begin
         Select(population, fitness);
         Reproduce(population);
         for i from 1 to size do
             fitness[i] := evaluate(population[i]);
     end
     return bestof(population, fitness);
end;
```

The inputs to the algorithm are a number, size, of solutions and an array which contains them. The array of solutions is known as a population and the input array is called the initial population. A fitness function is used to evaluate the performance of each member of the population, and the Tester function checks to see if the best solution of the population satisfies some termination criteria.

If the termination criteria are not met then some solutions are selected from the population (parents) and used to generate new solutions (children). The child solutions either, directly replace members of the parent population, or compete with them for places. (The size of the population is usually required to remain constant.) Once the Reproduce function has been applied, the current population of solutions is evaluated using the fitness function, and the selection, reproduction process is then repeated until the termination criteria are met. Each cycle of the Select-Modify-Evaluate loop is known as a generation.

2.3 Genotype and Phenotype

In attempting to solve many optimisation problems for which a subsymbolic approach has been adopted there is often more than one way in which solutions may be represented. For example, if the solution is known to be numerical, a binary or floating point representation may be adopted. The underlying representation of a solution is known as the genotype and can be considered as a solution's 'encoding'. The phenotype is the behavioural expression of the genotype within some environment.

In natural evolution, genetic material contains the information required to generate an organism (although its development depends in part on external environmental conditions). Pleiotropic and polygenic effects often make the understanding of the genotype extremely difficult. Pleiotropy is the effect of a single gene affecting several phenotypic traits. Polygeny occurs when a single phenotypic effect is determined by the interaction of many genes. These effects occur in many different subsymbolic representations and account for (in part) some of the results of the experiments discussed in later chapters of this thesis.

2.4 Genetic Algorithms

Holland (1975) proposed a genetic algorithm (GA) as an efficient search mechanism based upon modelling the evolutionary process in nature. GAs are perhaps the best known EA and have been applied to a wide range of problems (*e.g.*, Grefenstette 1985, 1987; Schaffer 1989; Belaw and Booker 1991; Forrest 1993). The underlying coding of a population of individuals is manipulated in an attempt to find a structure which maximises the performance of the corresponding phenotype within some environment. The emphasis of the approach is on the bottom-up construction of individuals by incorporating specific genotypic transformations.

There are many possible operators which can be used to manipulate the genotype (a string of data) of an individual. Two of the simplest operators which are commonly used are those of mutation and crossover. A crossover operator usually acts on two strings to produce two strings, while the mutation operator acts on one string producing one string. Figures 2.2 and 2.3 provide examples of these operators acting on solutions encoded using a binary representation.

Figure 2.2: An example of a mutation operator which acts on a binary string to produce a new string. The point(s) of mutation is (are) usually chosen uniformly at random.

before mutation	1	1	0	1	0	1	0	1	1
mutation point				*					
after mutation	1	1	0	0	0	1	0	1	1

Figure 2.3: An example of a one-point crossover operator which combines two (parent) binary strings to produce two new (child) binary strings. The point at which the sections of the strings are exchanged is usually chosen uniformly at random.

parent 1	1	1	0	1	0	1	0	1	1
parent 2	1	0	1	1	0	0	1	0	1
section exchanged	*	*	*	*	*				
child 1	1	0	1	1	0	1	0	1	1
child 2	1	1	0	1	0	0	1	0	1

It would appear, at first, that a particular generation of a GA possesses only a selection of possible solutions. However, each of these solutions is made up of a string of data and within each string there are many smaller strings of data, usually called schemata. A schema is a set of individuals which share common attributes. In the case of a binary alphabet a schema is denoted by a string consisting of elements taken from the set $\{0, 1, \Box\}$ where $\Box$ means "don't care". For example, 00101 and 00111 are both elements of the schema 00□□1, but 10101 is not. Each binary string of length k will be an instance of 2^k schemata.

Each schema represents a subset of strings and so an average fitness can be assigned to it. In a given population of strings a schema's average fitness (sometimes known as observed average fitness) is the average fitness of the strings which contain that schema. A schema's average fitness can vary from population to population since it is determined only by the instances of the schema currently within a population.

The *Schema Theorem* of Holland (1992, p. 102) relates the number of instances of a schema ξ in two successive generations. If the expected proportion of schema in one generation is $P(\xi, t)$ then:

$$P(\xi, t+1) \geq P(\xi, t)\, \frac{\hat{\mu}_\xi(t)}{\hat{\mu}(t)}\, [1 - P_D(\xi)]$$

where $\hat{\mu}_\xi(t)$ is the observed average performance of ξ, $\hat{\mu}(t)$ the average fitness of the schema ξ and $P_D(\xi)$ is the probability of the loss of ξ through the effect of operators such as crossover and mutation.

Throughout a single generation the solutions present will contain many schemata which can be combined (via crossover) to represent other individuals that are not present in the population at that time. The power of GAs is believed to stem from regarding the performance of a single solution as a test on the large number of schemata of which it is an instance. Thus a test on a solution of length k will simultaneously sample instances of 2^k schemata. In a population consisting of M solutions of length k there are between 2^k and $M2^k$ schemata with instances contained within the population. The proportion of each schema which survives to the subsequent generation is largely dependent only on its own observed fitness $\hat{\mu}_\xi(t)$ and is largely independent of what is happening to the other schemata in the population.

An approach using a GA allows for the highly fit short schemata to be propagated quickly through a population, while at the same time considering other less fit possibilities. This is termed by Holland (1975) as intrinsic parallelism (more recently known as implicit parallelism). The proportion of schema in a population is in part dependent on its past performance, and so this serves as a record of the performance.

In order to maximise the implicit parallelism, a binary approach to encoding is often advocated (Holland 1992, p. 71; Goldberg 1989, p.80). Such an approach allows for the maximum number of schemata to be represented and thus processed. The success of GAs is described by Goldberg (1989, p. 41) in terms of the building block hypothesis — short highly fit segments of each binary string can be combined to form larger, fitter segments of each binary string. The GA constitutes a bottom-up approach to solution construction.

While there is much experimental evidence to support a GA's success, a comprehensive theoretical basis is lacking and research into this continues (Goldberg and Rudnick 1991; Radcliffe 1991a; Davis and Principe 1993; Garigliano and Nettleton 1992; Garigliano *et al.* 1993d; Forrest and Mitchell 1993).

2.4.1 Criticism

Criticisms of GAs range from matters of detail to more fundamental questions regarding the underlying paradigm. Radcliffe (1991a) and Mason (1993) state that there is an overemphasis on a binary representation, and Mason objects to Holland's analysis of intrinsic parallelism. Fogel (1991; 1992a; 1993) argues that the GA's modelling of the evolutionary process is flawed.

Holland (1992, p. 71) and Goldberg (1989, p.41) advocate that solutions be represented using a binary encoding, since this allows for the maximum number of schemata to be represented. Radcliffe (1991a, p. 222) states that efforts "to maximise the level of intrinsic parallelism available are frequently in conflict with a desire to use natural representations and operators for the structures in the space being searched." In an attempt to overcome this Radcliffe (1991a; 1991b; 1992) generalises schemata to what are termed formae, and demonstrates that intrinsic parallelism is a more general phenomenon.

Mason's (1993) principle objection to Holland's analysis of intrinsic parallelism is that it does not compare like with like. The problem examined by Holland (1975) is deemed to be tailor made for a schema processing approach and essentially a sub-problem generated by the GA itself. Mason states that for progress to be made in the understanding of GAs it is "essential that the focus of research move away from issues such as compliance with the Schema Theorem and the degree of intrinsic parallelism and instead consider the methodology underlying the GA's exploration of new points in the solutions space."

Fogel (1991; 1992a; 1993) argues that to model the evolutionary process at the level of specific genotypic transformations is flawed, and that instead the emphasis should be on examining phenotypic adaptation (see Section 2.5). Atmar (1994, p. 142) although agreeing that GAs are basically a suitable model of Darwinian evolution states that the philosophical emphasis is "put on genetic mechanism, not on the process of phenotypic adaptation."

Common philosophical errors often lead to misinterpretations of the evolutionary process. For example, Srinivas and Patnaik (1994, p.18) state that "Specifically, each feature (of an individual) is controlled by a basic unit called a gene." This comment profoundly misinterprets the nature of genes, and incorrectly suggests that there is always a one to one mapping between a gene and an attribute of the individual. Such comments are unhelpful in the development of optimisation algorithms which are based on models of natural evolution.

2.5 Evolutionary Programming

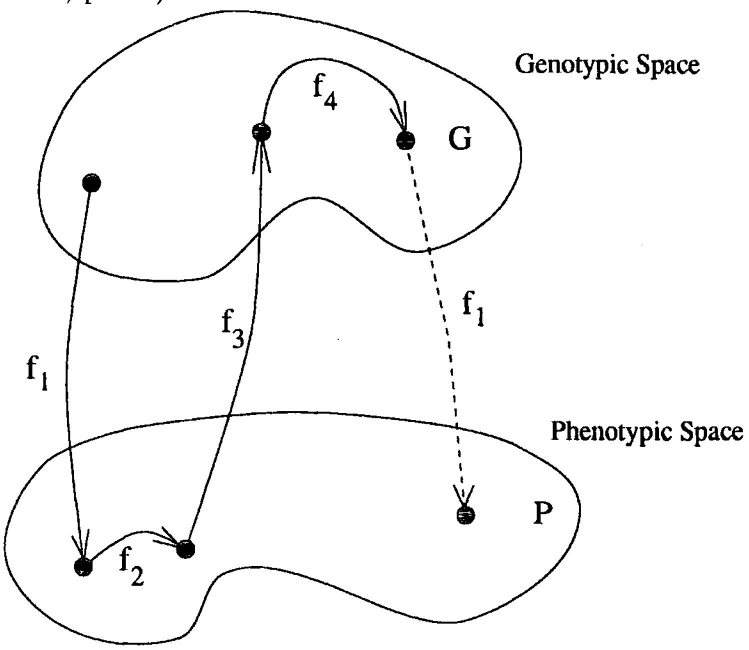
Evolutionary programming (EP) originated in the early 1960s, and was initially applied to a population of algorithms in order to study the possibilities of evolving artificial intelligence (Fogel *et al.* 1966). Since then, EP has been extended to cope with real-valued variables (Fogel 1992a) and EP has been applied to a wide range of problems (*e.g.*, Fogel and Atmar 1992, 1993; Sebald and Fogel 1994). The EP perspective of the evolutionary process is very different to the bottom-up building block approach of GAs. The differences stem from the primary forces modelled from natural evolution. While GAs incorporate specific genotypic transformations, EP emphasises phenotypic adaptation. EP adopts a top-down approach to solution improvement, as opposed to the bottom-up approach of GAs.

In order to help explain the philosophy underpinning the EP approach Atmar (1994) has modified the work of Lewontin (1974) and characterised the relationship between the genotype and phenotype by four functions. The functions relate a genotypic space $\mathbb{G}$ to a phenotypic space $\mathbb{P}$. An additional set of environmental symbols $\mathbb{I}$ is also defined.

$$\begin{aligned} f_1 &: \mathbb{I} \times \mathbb{G} \mapsto \mathbb{P} \\ f_2 &: \mathbb{P} \mapsto \mathbb{P} \\ f_3 &: \mathbb{P} \mapsto \mathbb{G} \\ f_4 &: \mathbb{G} \mapsto \mathbb{G} \end{aligned}$$

The mapping f_1 defines the development of the phenotype in terms of its genotype and the current environmental conditions. The environmental component is needed to model epigenesis (development sensitive to local conditions). For example, gender in turtles is influenced by the temperature at which development occurs.

Figure 2.4: Pictorial representation of the mapping functions suggested by Lewontin (1974, p. 14).



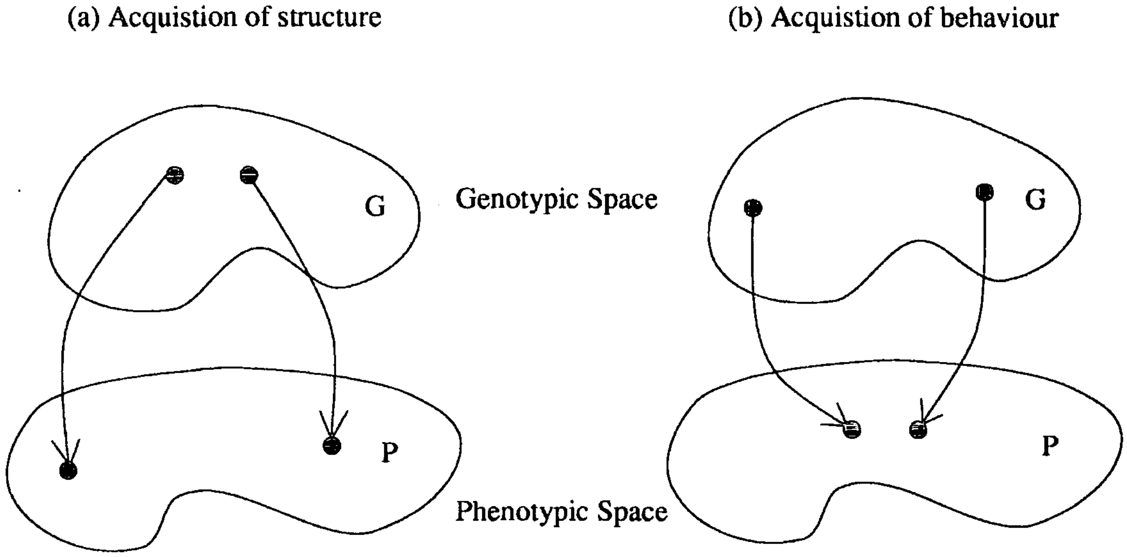
The mapping f_2 controls the selection, immigration and emigration of individuals from within the population. It is important to note that the selection process does not act directly on an individual's underlying encoding, but on its behaviour. The mapping f_3 maps the effects of f_2 back to the genotypic representation.

The mapping f_4 controls the manipulation of the coding via operators acting on the genotype. Such operators include mutation and recombination. Figure 2.4 provides a pictorial description of the interaction of the mapping functions.

Lewontin (1974, p. 15) warns that care must be taken in interpreting the relationship between the evolutionary dynamics of the spaces of the genotype and phenotype. The relationship näively may be viewed as being that of two independent process, the first of which acts on the genotype and the second on the phenotype. However, to assume independence of the processes is a dangerous oversimplification, which in all but the simplest cases is inaccurate. The functions introduced above describe the interactions between the two spaces. The modelling of the evolutionary process within this framework is of crucial importance and as Atmar (1994, p. 133) points out "Confusing the attributes of the two state spaces lies at the root of much of the confusion that permeates evolutionary theory."

When the relationship between the phenotype and genotype is characterised as above it is apparent that the selection mechanism acts only on the phenotype and any change which may occur to the set of genotypes as a result of the selection is incidental. In particular if two individuals have identical phenotypes the underlying genotype may be very different (Figure 2.5). Such a difference in encoding is irrelevant to the selection process.

Figure 2.5: A GA concentrates on the acquisition of structure. As (a) indicates structurally similar solutions may result in very large differences in behaviour. EP emphasises the adaptation of behaviour. Part (b) shows that although solutions may have a similar behaviour they may be structurally very different.



As discussed by Atmar (1994) adaptation in the phenotypic space is of the whole behavioural structure. To say that separate traits of an individual evolve independently is misleading since they are often highly interdependent. To model this behavioural link, EP uses mutations as the reproductive operators. Specific genotypic transformations such as crossover are deemed unnecessary.

Fogel *et al.* (1966) examined the evolution of finite state machines (FSMs) in an effort to create artificial intelligence. The aim of the project was for the system to be able to predict its environment, and to use this to produce a response which was conducive to the goal to be attained. Five possible mutations (which naturally follow from the definition of a FSM) are considered: change an output symbol, change a state transition, add a state, delete a state, or change the initial state. The number of mutations to occur is chosen with respect to a probability distribution (*e.g.*, Poisson), as is the the selection of the mutation (typically uniform). After mutation those machines which score in the top half of the population are retained and the others discarded.

Fogel *et al.* (1966) described a series of symbol prediction tasks. Problems such as the prediction of prime numbers from the sequence before them are considered and the results for different payoff functions presented. This early work showed the potential of the approach. Other applications of EP to FSMs have been considered. For example, Fogel (1992a, 1994) evolves FSMs for the iterated prisoners dilemma and shows that "evolving FSMs essentially learned to predict the behaviour (a sequence of symbols) of other FSMs in the evolving population."

More recently, EP has been extended to cope with real-valued continuous optimisations problems (Fogel 1991). Each real-valued component of an individual is mutated by an amount which is distributed Gaussian normally. The variance of the distribution is typically determined by the performance of the individual within the environment. Relating the severity of the mutation to the fitness of the solution ensures that fitter parents are less likely to be mutated to the same degree as less fit parents. After mutation solutions are probabilistically culled and the process repeated. Work on applying EP to real-valued problems includes that by Fogel (1992a; 1992b) and Nettleton and Garigliano (1994c).

Since the details of the mutation operator is dependent on the subsymbolic solution representation which is adopted, further discussion of mutation, and the selection mechanism, is left to later sections (*e.g.*, Section 5.3).

2.5.1 Criticism

Initial criticisms of the EP approach included those by Solomonoff (1966) and Lindsay (1968). These criticisms are based on a misunderstanding of the work, but were in part responsible for the AI community largely rejecting research into evolutionary systems during the 1970s.

Solomonoff (1966) argued that the method was only applicable to the simplest of problems and Goldberg (1989, p. 106) states that "The evolutionary programming of Fogel, Owens and Walsh ... was insufficiently powerful to search other than small problem spaces quickly." In fact a calculation of the number of possible solutions to some of the problems considered by Fogel *et al.* (1966) shows that extremely large spaces were considered, and these were successfully searched.

A more fundamental concern of Solomonoff (1966), Lindsay (1968) and Goldberg (1989) is the lack of a crossover operator, and Lindsay goes so far as to state that "...such a strategy [EP] amounts to random search... ." However, experimental evidence indicates that a crossover operator is not necessary for the successful solution of many problems (*e.g.*, Fogel *et al.* 1966; Fogel and Atmar 1990; Nettleton *et al.* 1993; Nettleton and Garigliano 1994c).

More recent criticisms argue that a top down approach to solution development is not able to take advantage of combining coadapted sections of the subsymbolic representation. Such criticisms may be relevant for some problems, but in general problems of pleiotropy and polygeny make the identification of coadapted sections difficult (Dawkins 1986; Mayr 1988).

2.6 Evolution Strategies

Evolution strategies (ESs) originated in the mid 1960's with early applications being practically orientated. Early work concerned the optimisation of real-valued object variables and the first experimental applications dealt with the shape optimisation of a bent pipe and flashing nozzle (Rechenberg 1973). More recent analysis of ESs has been carried out by, amongst others, Bäck *et al.* (1991) and Beyer (1993).

As with EP the emphasis of the ES approach is on the acquisition of a behaviour which has a high fitness value. Although philosophically similar, EP and ES differ in two important ways: (1) the operators used in generating the progeny, and (2) the selection mechanism employed to select which solutions survive to a subsequent generation.

Unlike EP, in generating new solutions ES allows for the use of a recombination operator. Bäck *et al.* (1991) discuss several recombination operators which have been suggested for use in ES, including:

1. Discrete recombination — given two parent solutions the values of the offspring are chosen uniformly at random from the corresponding values of their parents. (This is identical to the uniform crossover operator suggested by Syswerda (1989) for use in GAs.)
2. Intermediate recombination — the values of the offspring are the averages of the corresponding values of the parents.

A mutation operator is also used extensively in ESs. This typically subjects each value of an individual to a certain amount of Gaussian noise, the variance of which is controlled by what is known as the 1/5-success rule (see, *e.g.*, Bäck and Schwefel 1993):

> The ratio of successful mutations to all mutations should be 1/5. If it is greater than 1/5, increase the standard deviation, if it is smaller, decrease the standard deviation.

A successful mutation is one in which the resulting structure outperforms the original with regard to some measure of performance.

Fogel (1994) describes the different approaches to solution generation between EP and ESs in terms of the level of the evolutionary process which is being modelled. In the case of EP, evolution is typically modelled at the level of the species, while ESs usually models evolution at the level of the individual. A model based at the level of the individual allows for the recombination of individuals. The recombination of members of different species is, however, not permitted.

The second essential difference between ESs and EP is in the selection mechanism employed. While EP adopts a stochastic means of determining which solutions survive to a subsequent generation, ESs rely completely on deterministic selection methods. The two methods most commonly used are the (μ, λ)-*ES* and $(\mu + \lambda)$-*ES*. The methods select the best μ individuals from either, the set of λ offspring individuals ((μ, λ)-*ES*), or the set of parents and offspring ($(\mu + \lambda)$-*ES*). Bäck and Schwefel (1993) recommend the use of the (μ, λ)-*ES* since it is able to deal with changing environments.

2.7 Summary

In adopting a subsymbolic approach for encoding solutions to a problem there are often many different representations which can be used, *e.g.*, binary or real-valued. A range of algorithms (*e.g.*, hill-climbing, simulated annealing) have been suggested as a means of manipulating solution encodings in an attempt to optimise the solution's behaviour within some environment. EAs are based upon models of natural evolution and have resulted in robust optimisation techniques.

GAs rely on mimicking specific genotypic transformations and constitute a bottom-up building block approach to solution discovery. EP, on the other hand, emphasises phenotypic adaptation and adopts a top-down approach to solution improvement. Results of applying these two algorithms to problems which have different interactions at the level of the subsymbolic encoding are discussed in later chapters of this thesis. ESs are not considered further. The details of the GA and EP used in the experiments are introduced at the appropriate place.

Chapter 3

Iterated Function Systems

This chapter is the first of four which describe in detail various aspects of the shape representation problem — the first of the problems considered in this thesis. This very detailed account has been included to demonstrate the 'depth' of an approach to problem solving based upon a hybrid symbolic/subsymbolic approach and the use of EAs.

Giles (1990) suggests that IFSs be used as a shape representation scheme for use in a machine vision environment, and includes the following as some of the advantages: the theory of IFSs is well understood (Barnsley 1988) and the problem of finding an IFS for an arbitrary shape has in theory been solved (there are still practical difficulties); the primitives used to represent a shape are transformations of the shape itself and hence no pre-defined set of primitives is needed; the primitives used will automatically have the correct morphology, *e.g.*, geometric primitives are used for geometric shapes, and fractal primitives for fractal shapes; any shape can be represented as an IFS the accuracy only depending on the allowed storage space; the pictorial representation of an encoded shape can easily be rendered at any scale or orientation. Some of these features are discussed in detail in this and the following chapter.

Using IFSs for shape representation leads, initially, to a symbolic representation — the shape is to be represented by a collage of smaller copies of itself. However, since an IFS is composed of a set of contraction mappings a subsymbolic (real-valued) representation can be adopted. The purpose of this chapter is formally to introduce the theory and terminology of IFSs. This includes results which formalise the interactions that can occur between the subsymbolic components of an IFS's representation.

In order to provide a complete derivation of the material which is presented towards the end of the chapter it is necessary to review some basic concepts of metric topology. The reader familiar with such concepts can safely skip Sections 3.1 and 3.2. A fuller account of metric topology is given by Edgar (1990). The theory of IFSs was originally developed by Hutchinson (1981) and Barnsley and co-workers (Barnsley and Demko 1985; Barnsley *et al.* 1986). Much of this chapter is based on material in Barnsley's book 'Fractals Everywhere' (1988), where a more comprehensive treatment can be found.

3.1 Metric Spaces

Definition 3.1.1 *A space,* $\mathbf{X}$*, is a set. The points of a space are the elements of the set.*

The standard notation of $\mathbf{R}$ and $\mathbf{R}^2$ is used to denote the real line and two-dimensional Euclidean space respectively.

Definition 3.1.2 *A metric space is a set* $\mathbf{X}$ *together with a function* $d : \mathbf{X} \times \mathbf{X} \mapsto [0, \infty)$ *which obeys the following axioms:*

1. $d(x,y) = d(y,x) \qquad \forall\, x,y \in \mathbf{X}$

2. $d(x,y) = 0 \qquad \Leftrightarrow \qquad x = y \qquad \forall\, x,y \in \mathbf{X}$

3. $d(x,y) \leq d(x,z) + d(z,y) \qquad \forall\, x,y,z \in \mathbf{X}.$

The last inequality is known as the triangle inequality. The non-negative real number $d(x,y)$ is called the distance between x and y. The function d is called a metric on the set $\mathbb{X}$, and a metric space is written $(\mathbb{X}, d)$.

Definition 3.1.3 *Let $(\mathbb{X}, d)$ be a metric space with $x \in \mathbb{X}$. Given $\varepsilon > 0$ the set $B(x, \varepsilon)$ is defined as:*

$$B(x,\varepsilon) = \{y \in \mathbb{X} : d(x,y) \leq \varepsilon\}.$$

Definition 3.1.4 *A sequence of points $\{x_n\}_{n=1}^{\infty}$ in a metric space $\mathbb{X}$ converges to the point $x \in \mathbb{X}$ if and only if for every $\varepsilon > 0$, there exists an integer $N > 0$ such that $d(x_n, x) < \varepsilon$ for all $n \geq N$.*

The point $x \in \mathbb{X}$ to which the sequence $\{x_n\}_{n=1}^{\infty}$ converges is known as the limit of the sequence. In such cases:

$$x = \lim_{n \to \infty} x_n.$$

A sequence is convergent if and only if it converges to some point.

Definition 3.1.5 *A Cauchy sequence in a metric space $(\mathbb{X}, d)$ is a sequence $\{x_n\}_{n=1}^{\infty}$ such that for every $\varepsilon > 0$ there is an integer $N > 0$ such that:*

$$d(x_n, x_m) < \varepsilon \qquad \forall\, n, m \geq N.$$

Theorem 3.1.1 *Every convergent sequence is a Cauchy sequence.*

Proof: Suppose $\{x_n\}_{n=1}^{\infty}$ converges to x. Given $\varepsilon > 0$ there exists an integer $N > 0$ such that $d(x_n, x) < \frac{\varepsilon}{2}$ for all $n \geq N$. Using the triangle inequality with n, $m \geq N$ then:

$$d(x_n, x_m) \leq d(x_n, x) + d(x, x_m) < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

and so $\{x_n\}_{n=1}^{\infty}$ is a Cauchy sequence.

Definition 3.1.6 *A metric space $(\mathbb{X}, d)$ is complete if and only if every Cauchy sequence in $\mathbb{X}$ converges (in $\mathbb{X}$).*

Definition 3.1.7 *Let $\mathbb{S} \subset \mathbb{X}$ be a subset of a metric space $(\mathbb{X}, d)$. A point $x \in \mathbb{X}$ is called a limit point of $\mathbb{S}$ if there exists a sequence $\{x_n\}_{n=1}^{\infty}$ of points $x_n \in \mathbb{S} \setminus \{x\}$ such that $\lim_{n\to\infty} x_n = x$.*

Definition 3.1.8 *Let $\mathbb{S} \subset \mathbb{X}$ be a subset of a metric space $(\mathbb{X}, d)$. The closure of $\mathbb{S}$, denoted $\bar{\mathbb{S}}$, is defined to be $\bar{\mathbb{S}} = \mathbb{S} \cup \{limit\ points\ of\ \mathbb{S}\}$. $\mathbb{S}$ is closed if it contains all its limit points,* i.e., $\mathbb{S} = \bar{\mathbb{S}}$.

Definition 3.1.9 *Let $\mathbf{S} \subset \mathbf{X}$ be a subset of a metric space $(\mathbf{X}, d)$. $\mathbf{S}$ is compact if every sequence $\{x_n\}_{n=1}^{\infty}$ in $\mathbf{S}$ contains a subsequence having a limit in $\mathbf{S}$.*

Definition 3.1.10 *Let $\mathbf{S} \subset \mathbf{X}$ be a subset of a metric space $(\mathbf{X}, d)$. Then $\mathbf{S}$ is totally bounded if, given $\varepsilon > 0$, there is a finite set of points $\{y_1, y_2, \ldots, y_n\} \subset \mathbf{S}$ such that for all $x \in \mathbf{S}$, $d(x, y_i) < \varepsilon$ for some $y_i \in \{y_1, y_2, \ldots, y_n\}$. The set of points $\{y_1, y_2, \ldots, y_n\}$ is called an ε-net.*

Theorem 3.1.2 *Let $\mathbf{S} \subset \mathbf{X}$ be a subset of a complete metric space $(\mathbf{X}, d)$. Then $\mathbf{S}$ is compact if and only if it is closed and totally bounded.*

Proof: Suppose that $\mathbf{S}$ is closed and totally bounded. Let $\{x_i\}_{i=1}^{\infty}$ be a sequence of points in $\mathbf{S}$. It is possible to construct an ε-net, $\{y_1, y_2, \ldots, y_n\} \subset \mathbf{S}$, with $\varepsilon = 1$ such that:

$$\mathbf{S} \subset \bigcup_{j=1}^{n} B(y_j, 1).$$

The ε-net contains a finite number of points yet $\{x_i\}_{i=1}^{\infty}$ is infinite and so there must be a point y_k in the ε-net for which $B(y_k, 1) = B_1$ contains infinitely many points of the sequence. Choose N_1 so that $x_{N_1} \in B_1$. Clearly $\mathbf{S} \cap B_1$ is totally bounded and so an ε-net can be constructed for it with $\varepsilon = \frac{1}{2}$. Again there must be a point, y_m, of this ε-net such that $B(y_m, \frac{1}{2}) = B_2$ contains infinitely many points of the sequence. Choose N_2 so that $x_{N_2} \in B_2$ and $N_2 > N_1$. Continuing to halve the value of ε a subsequence $\{x_{N_n}\}_{n=1}^{\infty}$ of the sequence $\{x_i\}_{i=1}^{\infty}$ is generated. Since:

$$(\mathbb{S} \cap B_1) \supset (\mathbb{S} \cap B_2) \supset \cdots \supset (\mathbb{S} \cap B_n) \supset \cdots$$

and given that the radius of B_r is 2^{1-r} then:

$$d(x_{N_k}, x_{N_{k+1}}) \leq 2^{2-k} \qquad \forall\, k \geq 1.$$

Given a $\delta > 0$:

$$d(x_{N_k}, x_{N_{k+1}}) < \delta \qquad \forall\, k > 2 - \frac{ln(\delta)}{ln(2)}$$

so $\{x_{N_n}\}_{n=1}^{\infty}$ is a Cauchy sequence which, using the closure of $\mathbb{S}$, has a limit $x \in \mathbb{S}$. Therefore, $\mathbb{S}$ is compact if it is closed and totally bounded.

To complete the proof, suppose $\mathbb{S}$ is compact and assume for a given $\varepsilon > 0$ there does not exist an ε-net for $\mathbb{S}$. Then there is an infinite sequence of points $\{x_i\}_{i=1}^{\infty}$ in $\mathbb{S}$ with $d(x_i, x_j) > \varepsilon$ for all $i \neq j$. However, due to the compactness of $\mathbb{S}$ this sequence must contain a convergent subsequence with limit in $\mathbb{S}$ and so there is a pair of integers N_1 and N_2 with $N_1 \neq N_2$ such that $d(x_{N_1}, x_{N_2}) < \varepsilon$. This is a contradiction and so an ε-net does exist and hence $\mathbb{S}$ is closed and totally bounded. Therefore $\mathbb{S}$ is compact if and only if it is closed and totally bounded.

3.2 Mappings on a Metric Space

Definition 3.2.1 *Let $f : \mathbb{X} \mapsto \mathbb{X}'$ be a function from a metric space $(\mathbb{X}, d)$ into a metric space $(\mathbb{X}', d')$. If $x \in \mathbb{X}$ then f is continuous at x if and only if, for every $\varepsilon > 0$, there exists a $\delta > 0$ such that:*

$$d(x, y) < \delta \qquad \Rightarrow \qquad d'(f(x), f(y)) < \varepsilon.$$

The function f is simply called continuous if and only if it is continuous at every point $x \in \mathbf{X}$.

Definition 3.2.2 *Let $(\mathbf{X}, d)$ be a metric space. A transformation on $\mathbb{X}$ is a function $f : \mathbf{X} \mapsto \mathbb{X}$ which assigns exactly one point $f(x) \in \mathbf{X}$ to each point $x \in \mathbf{X}$. If $\mathbb{S} \subset \mathbb{X}$ then $f(\mathbb{S}) = \{f(x) : x \in \mathbb{S}\}$. f is one-to-one if $x, y \in \mathbf{X}$ with $f(x) = f(y)$ implies $x = y$. f is onto if $f(\mathbf{X}) = \mathbf{X}$. f is invertible if it is one-to-one and onto, and it is then possible to define a transformation $f^{-1} : \mathbf{X} \mapsto \mathbf{X}$ called the inverse of f, defined by $f^{-1}(y) = x$, where $x \in \mathbf{X}$ is the unique point such that $y = f(x)$.*

Definition 3.2.3 *A transformation $w : \mathbf{R}^2 \mapsto \mathbf{R}^2$ of the form:*

$$w(x_1, x_2) = (ax_1 + bx_2 + e, cx_1 + dx_2 + f)$$

where a, b, c, d, e and f are real numbers, is called a two-dimensional affine transformation. An alternative matrix representation is also used:

$$w(x) = w \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} = Ax + t.$$

Definition 3.2.4 *Let $f : \mathbf{X} \mapsto \mathbf{X}$ be a transformation on a metric space. A point $x_f \in \mathbf{X}$ is called a fixed point of f if $f(x_f) = x_f$.*

Definition 3.2.5 *A transformation $f : \mathbb{X} \mapsto \mathbb{X}$ on a metric space $(\mathbb{X}, d)$ is a contraction mapping if and only if there is a constant $0 \leq s < 1$ such that:*

$$d(f(x), f(y)) \leq sd(x,y) \qquad \forall\, x, y \in \mathbb{X}.$$

The number s is called the contractivity factor of the mapping f.

Lemma 3.2.1 *Let $w : \mathbf{X} \mapsto \mathbf{X}$ be a contraction mapping on the metric space $(\mathbf{X}, d)$. Then w is continuous.*

Proof: Let $s > 0$ be the contractivity factor of the mapping w and $x, y \in \mathbb{X}$. Then given $\varepsilon > 0$:

$$d(w(x), w(y)) \leq sd(x,y) < \varepsilon$$

whenever $d(x,y) < \delta$ where $\delta = \frac{\varepsilon}{s}$.

The following theorem is often referred to as the contraction mapping theorem.

Theorem 3.2.1 *A contraction mapping $f : \mathbf{X} \mapsto \mathbf{X}$ on a complete nonempty metric space $(\mathbf{X}, d)$ has a unique fixed point $x_f \in \mathbf{X}$. Further, for any point $x \in \mathbf{X}$:*

$$\lim_{n \to \infty} f^n(x) = x_f \qquad \forall\, x \in \mathbf{X}.$$

Proof: Assume there is more than one fixed point. If x_f and y_f are both fixed points then $d(x_f, y_f) = d(f(x_f), f(y_f)) \leq sd(x_f, y_f)$. But $0 \leq s < 1$, so this is impossible if $d(x_f, y_f) > 0$. Therefore, $d(x_f, y_f) = 0$ and so $x_f = y_f$.

Now let x_0 be any point of $\mathbf{X}$ (recall that $\mathbf{X}$ is nonempty). Then define recursively:

$$x_{n+1} = f(x_n) \qquad \text{for } n \geq 0.$$

Claim that $\{x_n\}_{n=1}^{\infty}$ is a Cauchy sequence. Writing $a = d(x_0, x_1)$ it follows by induction that $d(x_n, x_{n+1}) \leq as^n$. But then, if $m < n$:

$$\begin{aligned} d(x_m, x_n) &\leq \sum_{j=m}^{n-1} d(x_j, x_{j+1}) \\ &\leq \sum_{j=m}^{n-1} as^j \\ &= \frac{as^m - as^n}{1-s} \\ &= \frac{as^m(1-s^{n-m})}{1-s} \\ &\leq as^m(1-s)^{-1}. \end{aligned}$$

Therefore, if $\varepsilon > 0$ is given, N can be chosen such that $as^N(1-s)^{-1} < \varepsilon$. Then for $n, m \geq N$, $d(x_m, x_n) < \varepsilon$ and $\{x_n\}_{n=1}^{\infty}$ is a Cauchy sequence.

Now $(\mathbf{X}, d)$ is complete and $\{x_n\}_{n=1}^{\infty}$ is a Cauchy sequence, so it has a limit $x_f \in \mathbf{X}$ with $\lim_{n\to\infty} f^n(x) = x_f$. Now since f is contractive it is continuous and hence:

$$f(x_f) = f(\lim_{n\to\infty} f^n(x)) = \lim_{n\to\infty} f^{n+1}(x) = x_f,$$

x_f is therefore a fixed point of f.

The above theorem not only shows the existence of a fixed point in a complete metric space, but shows a way to construct the point.

3.3 The Metric Space $(\mathcal{H}(\mathbb{X}), h)$

Definition 3.3.1 *Let* $(\mathbb{X}, d)$ *be a complete metric space. Then* $\mathcal{H}(\mathbb{X})$ *denotes the space whose points are compact subsets of* $\mathbb{X}$ *other than the empty set.*

If $\mathbb{A}$ and $\mathbb{B}$ are subsets of some metric space $\mathbb{X}$, then $\mathbb{A}$ and $\mathbb{B}$ are within Hausdorff distance r of each other if and only if every point of $\mathbf{A}$ is within distance r of some point of $\mathbf{B}$, and every point of $\mathbf{B}$ is within distance r of some point of $\mathbf{A}$. This can be made into a metric called the Hausdorff metric, h.

Definition 3.3.2 *Let* $(\mathbb{X}, d)$ *be a complete metric space with* $x \in \mathbb{X}$*, and let* $\mathbf{B} \in \mathcal{H}(\mathbf{X})$*. Define:*

$$d(x, \mathbf{B}) = \min\{d(x, y) : y \in \mathbf{B}\}.$$

Then $d(x, \mathbf{B})$ is the distance from the point x to the set $\mathbf{B}$.

Definition 3.3.3 *Let* $(\mathbf{X}, d)$ *be a complete metric space with* $\mathbf{A}, \mathbf{B} \in \mathcal{H}(\mathbf{X})$*. Define:*

$$d(\mathbf{A}, \mathbf{B}) = \max\{d(x, \mathbf{B}) : x \in \mathbf{A}\}.$$

Then $d(\mathbf{A}, \mathbf{B})$ is the distance from the set $\mathbf{A}$ to the set $\mathbf{B}$.

Definition 3.3.4 *Let* $(\mathbf{X}, d)$ *be a complete metric space. The Hausdorff distance between* $\mathbf{A}, \mathbf{B} \in \mathcal{H}(\mathbf{X})$ *is defined by:*

$$h(\mathbf{A}, \mathbf{B}) = \max\{d(\mathbf{A}, \mathbf{B}), d(\mathbf{B}, \mathbf{A})\}.$$

An alternative definition of the Hausdorff distance, in terms of closed neighbourhoods, is also useful.

Definition 3.3.5 *Let $\mathbb{A} \subset \mathbb{X}$ be a subset of the metric space $(\mathbb{X}, d)$ and $\varepsilon > 0$, then the closed ε-neighbourhood of $\mathbb{A}$ is:*

$$N_\varepsilon(\mathbb{A}) = \{y \in \mathbb{X} : d(x,y) \leq \varepsilon \textit{ for some } x \in \mathbb{A}\}.$$

The Hausdorff distance h between $\mathbb{A}, \mathbb{B} \in \mathcal{H}(\mathbb{X})$ can then be defined as:

$$h(\mathbb{A}, \mathbb{B}) = \inf\{r : \mathbb{A} \subseteq N_r(\mathbb{B}) \text{ and } \mathbb{B} \subseteq N_r(\mathbb{A})\}$$

where inf is similar to min, but need not be attainable.

It is worth noting that in general h does not define a metric. For example in the space $\mathbb{R}$:

(a) What is the distance between $\{0\}$ and $[0, \infty)$? It is infinite and from the definition of a metric this is not allowed. Use of h is therefore restricted to bounded sets.

(b) What is the distance $h(\emptyset, \{0\})$? [1] Again it is infinite and so h is restricted to use on nonempty sets.

(c) What is the distance $h((0,1), [0,1])$? Now the distance is 0 even though the two sets are not equal and so h is restricted to use on closed sets.

For the purposes of this thesis h will only be applied to the nonempty compact sets $\mathcal{H}(\mathbf{X})$.

Theorem 3.3.1 *The Hausdorff distance h is a metric on the space $\mathcal{H}(\mathbf{X})$.*

Proof: Let $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathcal{H}(\mathbf{X})$. Clearly $h(\mathbf{A}, \mathbf{B}) = h(\mathbf{B}, \mathbf{A})$. Also:

$$h(\mathbf{A}, \mathbf{A}) = \max\{d(\mathbf{A}, \mathbf{A}), d(\mathbf{A}, \mathbf{A})\} = d(\mathbf{A}, \mathbf{A}) = \max\{d(x, \mathbf{A}) : x \in \mathbf{A}\} = 0.$$

[1] By convention, $\inf \emptyset = \infty$.

Since $\mathbb{A}$ and $\mathbb{B}$ are compact they are bounded and so $0 \leq h(\mathbb{A}, \mathbb{B}) < \infty$. Furthermore if $\mathbb{A} \neq \mathbb{B}$ then there is an $a \in \mathbb{A}$ such that $a \notin \mathbb{B}$ and $h(\mathbb{A}, \mathbb{B}) \geq d(a, \mathbb{B}) > 0$.

Finally, let $\varepsilon > 0$. If $x \in \mathbb{A}$, then there is $y \in \mathbb{B}$ with $d(x,y) < h(\mathbb{A}, \mathbb{B}) + \varepsilon$ and a $z \in \mathbb{C}$ with $d(y,z) < h(\mathbb{B}, \mathbb{C}) + \varepsilon$. This shows that $\mathbb{A}$ is contained in the $(h(\mathbb{A}, \mathbb{B}) + h(\mathbb{B}, \mathbb{C}) + 2\varepsilon)$-neighbourhood of $\mathbb{C}$. Similarly, $\mathbb{C}$ is contained in the $(h(\mathbb{A}, \mathbb{B}) + h(\mathbb{B}, \mathbb{C}) + 2\varepsilon)$-neighbourhood of $\mathbb{A}$. Therefore $h(\mathbb{A}, \mathbb{C}) \leq h(\mathbb{A}, \mathbb{B}) + h(\mathbb{B}, \mathbb{C}) + 2\varepsilon$. This is true for all $\varepsilon > 0$, so $h(\mathbb{A}, \mathbb{C}) \leq h(\mathbb{A}, \mathbb{B}) + h(\mathbb{B}, \mathbb{C})$ as desired.

The following is an important property of the Hausdorff metric that is required for a future proof.

Lemma 3.3.1 *For all* $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ *and* $\mathbf{E}$ *in* $\mathcal{H}(\mathbf{X})$*:*

$$h(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E}) \leq \max\{h(\mathbf{B}, \mathbf{D}), h(\mathbf{C}, \mathbf{E})\}.$$

Proof: Let $\mathbf{A} \in \mathcal{H}(\mathbf{X})$ then:

$$\begin{aligned} d(\mathbf{A} \cup \mathbf{B}, \mathbf{C}) &= \max\{d(x, \mathbf{C}) : x \in \mathbf{A} \cup \mathbf{B}\} \\ &= \max\{\max\{d(x, \mathbf{C}) : x \in \mathbf{A}\}, \max\{d(x, \mathbf{C}) : x \in \mathbf{B}\} \\ &= \max\{d(\mathbf{A}, \mathbf{C}), d(\mathbf{B}, \mathbf{C})\}. \end{aligned}$$

Also:

$$d(\mathbf{A}, \mathbf{B} \cup \mathbf{C}) = \max\{\min\{d(x,y) : y \in \mathbf{B} \cup \mathbf{C}\} : x \in \mathbf{A}\}$$

giving $d(\mathbf{A}, \mathbf{B} \cup \mathbf{C}) \leq d(\mathbf{A}, \mathbf{B})$ and $d(\mathbf{A}, \mathbf{B} \cup \mathbf{C}) \leq d(\mathbf{A}, \mathbf{C})$. Using this:

$$\begin{aligned} d(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E}) &= \max\{d(\mathbf{B}, \mathbf{D} \cup \mathbf{E}), d(\mathbf{C}, \mathbf{D} \cup \mathbf{E})\} \\ &\leq \max\{d(\mathbf{B}, \mathbf{D}), d(\mathbf{C}, \mathbf{E})\}. \end{aligned}$$

Finally:

$$h(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E}) = \max\{d(\mathbf{B} \cup \mathbf{C}, \mathbf{D} \cup \mathbf{E}), d(\mathbf{D} \cup \mathbf{E}, \mathbf{B} \cup \mathbf{C})\}$$

$$\begin{aligned} &\leq \max\{d(\mathbb{B},\mathbb{D}), d(\mathbb{C},\mathbb{E}), d(\mathbb{E},\mathbb{C}), d(\mathbb{D},\mathbb{B})\} \\ &\leq \max\{h(\mathbb{B},\mathbb{D}), h(\mathbb{C},\mathbb{E})\} \end{aligned}$$

as required.

The remainder of this section is aimed at proving that the metric space $(\mathcal{H}(\mathbb{X}), h)$ is complete. In order to do this several preliminary results are derived.

Lemma 3.3.2 *Let* $(\mathbb{X}, d)$ *be a metric space and* $\mathbb{A}, \mathbb{B} \in \mathcal{H}(\mathbb{X})$. *Given* $\varepsilon > 0$ *then:*

$$h(\mathbb{A},\mathbb{B}) \leq \varepsilon \quad \Leftrightarrow \quad \mathbb{A} \subset N_\varepsilon(\mathbb{B}) \text{ and } \mathbb{B} \subset N_\varepsilon(\mathbb{A}).$$

Proof: $h(\mathbb{A},\mathbb{B}) \leq \varepsilon$ implies that $d(\mathbb{A},\mathbb{B}) \leq \varepsilon$ and $d(\mathbb{B},\mathbb{A}) \leq \varepsilon$. Consider:

$$d(\mathbb{A},\mathbb{B}) = \max\{d(a,\mathbb{B}) : a \in \mathbb{A}\} \leq \varepsilon$$

then for each $a \in \mathbb{A}$, $a \in N_\varepsilon(\mathbb{B})$ and hence $\mathbb{A} \subset N_\varepsilon(\mathbb{B})$. Alternatively, suppose $\mathbb{A} \subset N_\varepsilon(\mathbb{B})$. Then for each $a \in \mathbb{A}$ there is a $b \in \mathbb{B}$ such that $d(a,b) \leq \varepsilon$ and so $d(a,\mathbb{B}) \leq \varepsilon$. This is true for each $a \in \mathbb{A}$ and so $d(\mathbb{A},\mathbb{B}) \leq \varepsilon$. A similar argument for $d(\mathbb{B},\mathbb{A})$ completes the proof.

The following Lemma concerns Cauchy sequences in $\mathcal{H}(\mathbb{X})$ and is necessary for the forthcoming completeness proof. The Lemma is referred to by Barnsley (1988, pp. 36–37) as the Extension Lemma.

Lemma 3.3.3 *Let* $(\mathbf{X}, d)$ *be a metric space and let* $\{\mathbf{A}_n\}_{n=1}^{\infty}$ *be a Cauchy sequence of points in* $(\mathcal{H}(\mathbf{X}), h)$. *Let* $\{n_j\}_{j=1}^{\infty}$ *be a sequence of integers such that* $0 < n_1 < n_2 < n_3 < \ldots$. *Suppose that* $\{x_{n_j} \in \mathbf{A}_{n_j}\}_{j=1}^{\infty}$ *is a Cauchy sequence in* $(\mathbf{X}, d)$, *then there is a Cauchy sequence* $\{x'_n \in \mathbf{A}_n\}_{n=1}^{\infty}$ *such that* $x'_{n_j} = x_{n_j}$ *for all* $j = 1, 2, 3, \ldots$.

Proof: For each $n \in \{1, 2, \ldots, n_1\}$ choose $x'_n \in \{x \in \mathbf{A}_n : d(x, x_{n_1}) = d(x_{n_1}, \mathbf{A}_n)\}$. That is, x'_n is the closest point (or one of the closest points) in $\mathbf{A}_n$ to x_{n_1}. The existence of such a closest point is ensured by the compactness of $\mathbf{A}_n$. Similarly for each $j \in \{2, 3, \ldots\}$ and each $n \in \{n_{j-1} + 1, \ldots, n_j\}$ choose $x'_n \in \{x \in \mathbf{A}_n : d(x, x_{n_j}) = d(x_{n_j}, \mathbf{A}_n)\}$.

Clearly from construction $x'_{n_j} = x_{n_j}$ and $x'_n \in \mathbb{A}_n$. Given an $\varepsilon > 0$, there exists an N_1 such that $d(x_{n_k}, x_{n_j}) < \frac{\varepsilon}{3}$ for all $n_k, n_j \geq N_1$. Also, there is a number N_2 such that $d(\mathbb{A}_m, \mathbb{A}_n) < \frac{\varepsilon}{3}$ for all $m, n \geq N_2$. Let $N = \max\{N_1, N_2\}$ and note that for $m, n \geq N$:

$$\begin{aligned} d(x'_m, x'_n) &\leq d(x'_m, x_{n_k}) + d(x_{n_k}, x'_n) \\ &\leq d(x'_m, x_{n_j}) + d(x_{n_j}, x_{n_k}) + d(x_{n_k}, x'_n) \end{aligned}$$

where $m \in \{n_{j-1}+1, n_{j-1}+2, \ldots, n_j\}$ and $n \in \{n_{k-1}+1, n_{k-1}+2, \ldots, n_k\}$. Since $h(\mathbf{A}_m, \mathbf{A}_{n_j}) < \frac{\varepsilon}{3}$ there exists $y \in \mathbf{A}_m \cap N_{\frac{\varepsilon}{3}}(\{x_{n_j}\}_{j=1}^{\infty})$ so that $d(x'_m, x_{n_j}) < \frac{\varepsilon}{3}$. Similarly $d(x_{n_k}, x'_n) < \frac{\varepsilon}{3}$. Therefore $d(x'_m, x'_n) < \varepsilon$ for all $m, n \geq N$ and hence the sequence is a Cauchy sequence.

The following is the main theorem of this section.

Theorem 3.3.2 *Let* $(\mathbf{X}, d)$ *be a complete metric space. Then* $(\mathcal{H}(\mathbf{X}), h)$ *is a complete metric space. Moreover, if* $\{\mathbf{A}_n \in \mathcal{H}(\mathbf{X})\}_{n=1}^{\infty}$ *is a Cauchy sequence then* $\mathbf{A} = \lim_{n\to\infty} \mathbf{A}_n$ *can be characterised as follows:*

$$\mathbf{A} = \{x \in \mathbf{X} : \textit{ there is a Cauchy sequence } \{x_n \in \mathbf{A}_n\}_{n=1}^{\infty} \textit{ such that } \lim_{n\to\infty} x_n = x\}.$$

Proof: Define $\mathbf{A}$ as above. The proof is broken into the following parts:

1. $\mathbf{A} \neq \emptyset$;

2. $\mathbf{A}$ is closed and hence complete since $\mathbf{X}$ is complete;

3. for $\varepsilon > 0$ there exists an N such that for all $n \geq N$, $\mathbf{A} \subset N_\varepsilon(\mathbf{A}_n)$;

4. $\mathbf{A}$ is totally bounded and thus by 2 is compact;

5. $\lim_{n\to\infty} \mathbf{A}_n = \mathbf{A}$.

Proof of 1: Let $N_1 < N_2 < N_3 < \ldots < N_n < \ldots$ be a sequence of positive integers such that:

$$h(\mathbb{A}_m, \mathbb{A}_n) < 2^{-i} \qquad \forall\, m, n \geq N_i.$$

Therefore $h(\mathbb{A}_{N_1}, \mathbb{A}_{N_2}) < \frac{1}{2}$ and so from the definition of the Hausdorff metric there exists a pair of points $x_{N_1} \in \mathbb{A}_{N_1}$ and $x_{N_2} \in \mathbb{A}_{N_2}$ such that $d(x_{N_1}, x_{N_2}) < \frac{1}{2}$. Hence, a sequence of points $\{x_{N_i} \in \mathbf{A}_{N_i}\}_{i=1}^{\infty}$ can be constructed such that $d(x_{N_i}, x_{N_{i+1}}) < 2^{-i}$. Given $\varepsilon > 0$, choose k such that $\sum_{i=k}^{\infty} 2^{-i} < \varepsilon$. Then for $n > m \geq k$:

$$\begin{aligned} d(x_{N_m}, x_{N_n}) &\leq d(x_{N_m}, x_{N_{m+1}}) + d(x_{N_{m+1}}, x_{N_{m+2}}) + \cdots d(x_{N_{n-1}}, x_{N_n}) \\ &< \sum_{i=k}^{\infty} \frac{1}{2^i} \end{aligned}$$

and so $\{x_{N_i}\}_{i=1}^{\infty}$ is a Cauchy sequence. By the Extension Lemma (3.3.3) it is possible to construct a Cauchy sequence $\{x'_i \in \mathbf{A}_i\}_{i=1}^{\infty}$ with $x'_{N_i} = x_{N_i}$. Since $\mathbf{X}$ is complete, this sequence has a limit in $\mathbf{X}$. By definition this limit is in $\mathbf{A}$ and so $\mathbf{A} \neq \emptyset$.

Proof of 2: Suppose $\{a_i \in \mathbf{A}\}_{i=1}^{\infty}$ is a sequence that converges to a point a. For each a_i there exists a sequence $\{x_{i,n} \in \mathbf{A}_n\}_{n=1}^{\infty}$ such that $\lim_{n\to\infty} x_{i,n} = a_i$. There exists an increasing sequence of positive integers $\{N_i\}_{i=1}^{\infty}$ such that the subsequence $\{a_{N_i}\}_{i=1}^{\infty}$ can be chosen such that $d(a_{N_i}, a) < \frac{1}{i}$. Furthermore for each N_i there exists an integer m_i such that the subsequence $\{x_{N_i,m_i}\}_{i=1}^{\infty}$ can be chosen such that $d(x_{N_i,m_i}, a_{N_i}) < \frac{1}{i}$. Thus $d(x_{N_i,m_i}, a) < \frac{2}{i}$ which tends to zero as $i \to \infty$. Setting $y_{N_i} = x_{N_i,m_i}$ then $y_{N_i} \in \mathbf{A}_{N_i}$ and $\lim_{i\to\infty} y_{N_i} = a$. By the Extension Lemma $\{y_{N_i}\}_{i=1}^{\infty}$ can be extended to a convergent sequence of points $\{z_i \in \mathbf{A}_i\}_{i=1}^{\infty}$ with limit a, which by definition means $a \in \mathbf{A}$, and hence $\mathbf{A}$ is closed.

Proof of 3: Given $\varepsilon > 0$ there exists an N such that for $m, n \geq N$, $h(\mathbf{A}_m, \mathbf{A}_n) < \varepsilon$. Let $n \geq N$ then for $m \geq n$, $\mathbf{A}_m \subset N_\varepsilon(\mathbf{A}_n)$. For $a \in \mathbf{A}$ there is a sequence

$\{a_i \in \mathbb{A}_i\}_{i=1}^{\infty}$ that converges to a. Assume that N is large enough so that:

$$d(a_m, a) < \varepsilon \qquad \forall\, m \geq N.$$

Then $a_m \in N_\varepsilon(\mathbb{A}_n)$ since $\mathbb{A}_m \subset N_\varepsilon(\mathbb{A}_n)$. Since $\mathbb{A}_n$ is compact, it can be shown that $N_\varepsilon(\mathbb{A}_n)$ is closed. Then since $a_m \in N_\varepsilon(\mathbb{A}_n)$ for $m \geq N$, $a \in N_\varepsilon(\mathbb{A}_n)$. Hence, $\mathbb{A} \subset N_\varepsilon(\mathbb{A}_n)$ for n large enough.

Proof of 4: Assume that $\mathbb{A}$ is not totally bounded, then for some $\varepsilon > 0$ no finite ε-net exists. There then exists a sequence $\{a_i \in \mathbb{A}\}_{i=1}^{\infty}$ such that $d(a_i, a_j) > \varepsilon$ for $i \neq j$. However, from 3, there exists an n for which $\mathbb{A} \subset N_{\frac{\varepsilon}{3}}(\mathbb{A}_n)$ and so for each $a_i \in \mathbb{A}$ there is a $y_i \in \mathbb{A}_n$ such that $d(a_i, y_i) < \frac{\varepsilon}{3}$. Since $\mathbb{A}_n$ is compact some subsequence $\{y_{n_i}\}_{i=1}^{\infty}$ of $\{y_i\}_{i=1}^{\infty}$ converges and it is possible to choose points in the sequence $\{y_{n_i}\}_{i=1}^{\infty}$ as close together as desired. In particular choose two points y_{n_i} and y_{n_j}, such that $d(y_{n_i}, y_{n_j}) < \frac{\varepsilon}{3}$. But then:

$$\begin{aligned} d(a_{n_i}, a_{n_j}) &\leq d(a_{n_i}, y_{n_i}) + d(y_{n_i}, y_{n_j}) + d(y_{n_j}, a_{n_j}) \\ &< \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3}. \end{aligned}$$

This contradicts the initial assumption that $d(a_i, a_j) > \varepsilon$ and hence $\mathbb{A}$ is totally bounded. It has already been shown that $\mathbb{A}$ is closed and so it is compact.

Proof of 5: Given $\varepsilon > 0$ there exists a sequence of positive integers $N_1 < N_2 < \ldots < N_k < \ldots$ such that:

$$h(\mathbf{A}_i, \mathbf{A}_j) < \frac{\varepsilon}{2^{k+1}} \qquad \forall\, i, j \geq N_k.$$

Choose $n < N_1$ such that $h(\mathbf{A}_n, \mathbf{A}_{N_1}) < \frac{\varepsilon}{2}$, then taking $y \in \mathbf{A}_n$ there is a point $x_{N_1} \in \mathbf{A}_{N_1}$ such that $d(y, x_{N_1}) < \frac{\varepsilon}{2}$. Similarly there is a point $x_{N_2} \in \mathbf{A}_{N_2}$ such that $d(x_{N_1}, x_{N_2}) < \frac{\varepsilon}{2^2}$. The sequence $x_{N_1}, x_{N_2}, x_{N_3}, \ldots$ can be constructed such that $d(x_{N_k}, x_{N_{k+1}}) < \frac{\varepsilon}{2^{k+1}}$. Using the triangle inequality:

$$d(y, x_{N_k}) \leq d(y, x_{N_1}) + d(x_{N_1}, x_{N_2}) + \cdots + d(x_{N_{k-1}}, x_{N_k})$$

$$
\begin{aligned}
&< \varepsilon \sum_{n=1}^{k} \frac{1}{2^n} = \varepsilon(1-2^{-k}) \\
&\leq \varepsilon.
\end{aligned}
$$

Clearly the Cauchy sequence $\{x_{N_k}\}_{k=1}^{\infty}$ converges to a point $a \in \mathbb{A}$ for which $d(y,a) < \varepsilon$. Thus $\mathbb{A}_n \subset N_\varepsilon(\mathbb{A})$ for large enough n. Combining this with the above result that $\mathbb{A} \subset N_\varepsilon(\mathbb{A}_n)$ for large enough n, then $h(\mathbb{A}, \mathbb{A}_n) < \varepsilon$ for large enough n. Thus $\lim_{n\to\infty} \mathbb{A}_n = \mathbb{A}$.

The above discussion of the properties of the space $\mathcal{H}(\mathbb{X})$ and its metric has shown that that $(\mathcal{H}(\mathbb{X}), h)$ can be treated like any other complete metric space.

3.4 Mappings on the Metric Space $(\mathcal{H}(\mathbb{X}), h)$

This section extends the work of Section 3.2 to the space $\mathcal{H}(\mathbb{X})$. In addition properties of a mapping which is itself a union of mappings are derived.

Lemma 3.4.1 *Let $w : \mathbf{X} \mapsto \mathbf{X}$ be a continuous mapping on the metric space $(\mathbf{X}, d)$. Then w maps $\mathcal{H}(\mathbf{X})$ into itself.*

Proof: Let $\mathbb{S}$ be a nonempty compact subset of $\mathbf{X}$. Then clearly $w(\mathbb{S}) = \{w(x) : x \in \mathbb{S}\}$ is nonempty. It is now enough to show that $w(\mathbb{S})$ is compact. Let $\{y_n = w(x_n)\}_{n=1}^{\infty}$ be an infinite sequence of points in $w(\mathbb{S})$. Then $\{x_n\}_{n=1}^{\infty}$ is an infinite sequence of points in $\mathbb{S}$. Since $\mathbb{S}$ is compact there is a subsequence $\{x_{N_n}\}_{n=1}^{\infty}$ which converges to a point $\hat{x} \in \mathbb{S}$. The continuity of w implies that $\{y_{N_n} = w(x_{N_n})\}_{n=1}^{\infty}$ is a subsequence of $\{y_n\}_{n=1}^{\infty}$ which converges to a point $\hat{y} = w(\hat{x}) \in w(\mathbb{S})$. Thus $w(\mathbb{S})$ is compact.

Lemma 3.4.2 *Let $w : \mathbb{X} \mapsto \mathbb{X}$ be a contraction mapping on the metric space $(\mathbb{X}, d)$ with contractivity factor s. Then $w : \mathcal{H}(\mathbb{X}) \mapsto \mathcal{H}(\mathbb{X})$ defined by:*

$$w(\mathbb{B}) = \{w(x) : x \in \mathbb{B}\} \qquad \forall\, \mathbb{B} \in \mathcal{H}(\mathbb{X})$$

is a contraction mapping on $(\mathcal{H}(\mathbb{X}), h)$ with contractivity factor s.

Proof: From Lemma 3.2.1 $w : \mathbb{X} \mapsto \mathbb{X}$ is continuous and hence by Lemma 3.4.1 w maps $\mathcal{H}(\mathbb{X})$ into itself. Now let $\mathbf{B}, \mathbf{C} \in \mathcal{H}(\mathbb{X})$, then:

$$\begin{aligned} d(w(\mathbf{B}), w(\mathbf{C})) &= \max\{\min\{d(w(x), w(y)) : y \in \mathbf{C}\} : x \in \mathbf{B}\} \\ &\leq \max\{\min\{s\, d(x,y) : y \in \mathbf{C}\} : x \in \mathbf{B}\} = s\, d(\mathbf{B}, \mathbf{C}). \end{aligned}$$

Similarly, $d(w(\mathbf{C}), w(\mathbf{B})) \leq sd(\mathbf{C}, \mathbf{B})$. Hence:

$$\begin{aligned} h(w(\mathbf{B}), w(\mathbf{C})) &= \max\{d(w(\mathbf{B}), w(\mathbf{C})), d(w(\mathbf{C}), w(\mathbf{B}))\} \\ &\leq s \max\{d(\mathbf{B}, \mathbf{C}), d(\mathbf{C}, \mathbf{B})\} \\ &\leq s\, h(\mathbf{B}, \mathbf{C}). \end{aligned}$$

Lemma 3.4.3 *Let $(\mathbf{X}, d)$ be a metric space. Let $\{w_n : n = 1, 2, \ldots, N\}$ be contraction mappings on $(\mathcal{H}(\mathbf{X}), h)$. Let the contractivity factor for w_n be denoted s_n for each n. Define $W : \mathcal{H}(\mathbf{X}) \mapsto \mathcal{H}(\mathbf{X})$ by:*

$$\begin{aligned} W(\mathbf{B}) &= w_1(\mathbf{B}) \cup w_2(\mathbf{B}) \cup \ldots \cup w_N(\mathbf{B}) \\ &= \bigcup_{n=1}^{N} w_n(\mathbf{B}) \qquad \forall\, \mathbf{B} \in \mathcal{H}(\mathbf{X}). \end{aligned}$$

Then W is a contraction mapping with contractivity factor $s = \max\{s_n : n = 1, 2, \ldots, N\}$.

Proof: A proof by induction is used. Let $N = 2$ and $\mathbb{B}, \mathbb{C} \in \mathcal{H}(\mathbb{X})$ then:

$$\begin{aligned}
h(W(\mathbb{B}), W(\mathbb{C})) &= h(w_1(\mathbb{B}) \cup w_2(\mathbb{B}), w_1(\mathbb{C}) \cup w_2(\mathbb{C})) \\
&\qquad\qquad\qquad\qquad \text{(by Lemma 3.3.1)} \\
&\leq \max\{h(w_1(\mathbb{B}), w_1(\mathbb{C})), h(w_2(\mathbb{B}), w_2(\mathbb{C}))\} \\
&\leq \max\{s_1\, h(\mathbb{B}, \mathbb{C}), s_2\, h(\mathbb{B}, \mathbb{C})\} \\
&\leq s\, h(\mathbf{B}, \mathbf{C}).
\end{aligned}$$

Thus the Lemma is true for $N = 2$. Now assume it is true for $N = m$ mappings. Consider the addition of an extra transform w_{m+1} to construct W' such that:

$$W'(\mathbf{B}) = \bigcup_{n=1}^{m+1} w_n(\mathbf{B}) = \bigcup_{n=1}^{m} w_n(\mathbf{B}) \cup w_{m+1}(\mathbf{B}) = W(\mathbf{B}) \cup w_{m+1}(\mathbf{B}).$$

Then:

$$\begin{aligned}
h(W'(\mathbf{B}), W'(\mathbf{C})) &= h(W(\mathbf{B}) \cup w_{m+1}(\mathbf{B}), W(\mathbf{C}) \cup w_{m+1}(\mathbf{C})) \\
&\qquad\qquad\qquad\qquad \text{(by Lemma 3.3.1)} \\
&\leq \max\{h(W(\mathbf{B}), W(\mathbf{C})), h(w_{m+1}(\mathbf{B}), w_{m+1}(\mathbf{C}))\} \\
&\leq \max\{s\, h(\mathbf{B}, \mathbf{C}), s_{m+1} h(\mathbf{B}, \mathbf{C})\} \\
&\leq \max\{s, s_{m+1}\} h(\mathbf{B}, \mathbf{C}) \\
&= \max\{s_1, s_2, \ldots, s_{m+1}\} h(\mathbf{B}, \mathbf{C}).
\end{aligned}$$

Hence, if the Lemma is true for m it is true for $m + 1$. The Lemma is true for $m = 2$ and so by the induction hypothesis the Lemma is true for all $m \geq 2$.

3.5 Iterated Function Systems: Definition and Properties

This section introduces iterated function systems and derives some of their properties.

Definition 3.5.1 *An iterated function system (IFS) consists of a complete metric space* $(\mathbf{X}, d)$ *together with a finite set of contraction mappings* $w_n : \mathbf{X} \mapsto \mathbf{X}$ *with respective contractivity factors* s_n *for* $n = 1, 2, \ldots, N$*. The notation for an iterated function system is* $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ *and its contractivity factor is* $s = \max\{s_n : n = 1, 2, \ldots, N\}$.

The following theorem summarises some of the main features of an IFS.

Theorem 3.5.1 *Let* $\{\mathbf{X}, w_n : n = 1, 2, \ldots, N\}$ *be an iterated function system with contractivity factor* s*. Then the transformation* $W : \mathcal{H}(\mathbf{X}) \mapsto \mathcal{H}(\mathbf{X})$ *defined by:*

$$W(\mathbf{B}) = \bigcup_{n=1}^{N} w_n(\mathbf{B}) \qquad \forall\, \mathbf{B} \in \mathcal{H}(\mathbf{X})$$

is a contraction mapping on the complete metric space $(\mathcal{H}(\mathbf{X}), h)$ *with contractivity factor* s*. Furthermore the unique fixed point* $\mathcal{A} \in \mathcal{H}(\mathbf{X})$ *obeys:*

$$\mathcal{A} = W(\mathcal{A}) = \bigcup_{n=1}^{N} w_n(\mathcal{A})$$

and is given by $\mathcal{A} = \lim_{n\to\infty} W^n(\mathbf{B})\ \forall\ \mathbf{B} \in \mathcal{H}(\mathbf{X})$*, where* $W^n(\mathbf{B}) = W(W^{n-1}(\mathbf{B}))$ *and* $W^0(\mathbf{B}) = \mathbf{B}$.

Proof: The proof follows directly from those of Theorem 3.2.1 and Lemma 3.4.3.

Definition 3.5.2 *The fixed point $\mathcal{A} \in \mathcal{H}(\mathbb{X})$ described in the above theorem is called the attractor of the iterated function system.*

An important feature of an IFS is that there is a continuous dependence of the attractor on the parameters which encode it. This so-called robustness property ensures that small changes in an IFS's encoding parameters produce correspondingly small changes in the attractor. Larger changes, however, can be expected to produce large changes in the attractor. There is, therfore, a strong interaction between the subsymbolic components of a mapping. A pictorial demonstration of this is provided in Section 4.3.

Lemma 3.5.1 *Let $(\mathbb{P}, d_P)$ and $(\mathbb{X}, d)$ be metric spaces, the latter being complete. Let $w : \mathbb{P} \times \mathbb{X} \mapsto \mathbb{X}$ be a family of contraction mappings on $\mathbb{X}$ with contractivity factor $0 \leq s < 1$,* i.e., *for each $p \in \mathbb{P}$ and $x \in \mathbb{X}$, $w(p, x)$ is a contraction mapping on $\mathbb{X}$. For each fixed $x \in \mathbb{X}$ let w be continuous on $\mathbb{P}$. Then the fixed point of w depends continuously on p,* i.e., *$x_f : \mathbb{P} \mapsto \mathbb{X}$ is continuous.*

Proof: Let $x_f(p)$ denote the fixed point of w for a fixed $p \in \mathbb{P}$. Given $p \in \mathbb{P}$ and $\varepsilon > 0$ then for all $q \in \mathbb{P}$:

$$\begin{aligned} d(x_f(p), x_f(q)) &= d(w(p, x_f(p)), w(q, x_f(q))) \\ &\leq d(w(p, x_f(p)), w(q, x_f(p))) + d(w(q, x_f(p)), w(q, x_f(q))) \\ &\leq d(w(p, x_f(p)), w(q, x_f(p))) + sd(x_f(p), x_f(q)) \end{aligned}$$

which implies:

$$d(x_f(p), x_f(q)) \leq (1 - s)^{-1} d(w(p, x_f(p)), w(q, x_f(p))).$$

Since w is continuous on $\mathbb{P}$, q can be chosen sufficiently close to p such that $0 < d_p(p, q) < \delta$ and so:

$$d(w(p, x), w(q, x)) < \varepsilon d_p(p, q) \qquad \forall\, x \in \mathbb{X}$$

and hence:

$$d(x_f(p), x_f(q)) < (1-s)^{-1} d_p(p,q)$$

and so $x_f : \mathbb{P} \mapsto \mathbb{X}$ is continuous.

Lemma 3.5.2 *Let $(\mathbb{X}, d)$ be a metric space. Let $w_n : \mathbb{P} \times \mathbb{X} \mapsto \mathbb{X}$ for $n = 1, 2, \ldots, N$ be continuous transformations depending continuously on a parameter $p \in \mathbb{P}$, where $(\mathbb{P}, d_p)$ is a compact metric space. That is, $w_n(p, x)$ depends continuously on p for fixed $x \in \mathbb{X}$. Then the transformation $W : \mathcal{H}(\mathbb{X}) \mapsto \mathcal{H}(\mathbb{X})$ defined by:*

$$W(p, \mathbf{B}) = \bigcup_{n=1}^{N} w_n(p, \mathbf{B}) \qquad \forall\, \mathbf{B} \in \mathcal{H}(\mathbb{X})$$

is also continuous in p, i.e., *$W(p, \mathbf{B})$ is continuous in p for each $\mathbf{B} \in \mathcal{H}(\mathbb{X})$, in the metric space $(\mathcal{H}(\mathbb{X}), h)$.*

Proof: For $\mathbf{B} \in \mathcal{H}(\mathbb{X})$ with $p, q \in \mathbb{P}$ and given $\varepsilon > 0$:

$$\begin{aligned} d(w_1(p,\mathbf{B}), w_1(q,\mathbf{B})) &= \max\{\min\{d(w_1(p,x), w_1(q,y)) : y \in \mathbf{B}\} : x \in \mathbf{B}\} \\ &\leq \max\{\min\{d(w_1(p,x), w_1(p,y)) + \\ &\qquad\qquad d(w_1(p,y), w_1(q,y)) : y \in \mathbf{B}\} : x \in \mathbf{B}\}. \end{aligned}$$

Since $\mathbf{P} \times \mathbf{B}$ is compact and $w_1 : \mathbf{P} \times \mathbf{B} \mapsto \mathbf{X}$ is continuous, then w_1 is uniformly continuous. Hence there exists a $\delta > 0$ such that if $d_p(p,q) < \delta$ then $d(w_1(p,y), w_1(q,y)) < \varepsilon$ for all $y \in \mathbf{B}$. Assuming $d_p(p,q) < \delta$ then:

$$\begin{aligned} d(w_1(p,\mathbf{B}), w_1(q,\mathbf{B})) &< \max\{\min\{d(w_1(p,x), w_1(p,y)) + \varepsilon : y \in \mathbf{B}\} : x \in \mathbf{B}\} \\ &\leq d(w_1(p,\mathbf{B}), w_1(p,\mathbf{B})) + \varepsilon = \varepsilon. \end{aligned}$$

Similarly $d(w_1(q,\mathbb{B}), w_1(p,\mathbb{B})) < \varepsilon$ for all $d_p(p,q) < \delta$ and so:

$$h(w_1(p,\mathbb{B}), w_1(q,\mathbb{B})) < \varepsilon \qquad \forall\, d_p(p,q) < \delta.$$

Hence $W(p,\mathbb{B})$ is continuous for $N = 1$. This result can now be extended for $N > 1$ using Lemma 3.3.1, giving:

$$\begin{aligned} h(W(p,\mathbb{B}), W(q,\mathbb{B})) &\leq \max\{h(w_n(p,\mathbb{B}), w_n(q,\mathbb{B}))\} = \varepsilon' \\ &\leq \varepsilon' \qquad \forall\, d_p(p,q) < \delta \end{aligned}$$

Theorem 3.5.2 *Let $(\mathbb{X}, d)$ be a metric space. Let $\{\mathbb{X}, w_n : n = 1,2,\ldots,N\}$ be an iterated function system such that w_n depend continuously on a parameter $p \in \mathbf{P}$, where $\mathbf{P}$ is a compact metric space. Then the attractor $\mathcal{A} \in \mathcal{H}(\mathbb{X})$ depends continuously on $p \in \mathbf{P}$, with respect to the Hausdorff metric.*

Proof: Follows from Lemma 3.5.1 and Lemma 3.5.2.

This section is concluded with the Collage Theorem which is of fundamental importance to the possibility of using IFSs for shape representation.

Theorem 3.5.3 *Let $(\mathbb{X}, d)$ be a complete metric space. Let $\mathbb{L} \in \mathcal{H}(\mathbb{X})$ and choose an iterated function system $\{\mathbb{X}, w_n : n = 1,2,\ldots,N\}$ with contractivity factor $0 \leq s < 1$ such that for some $\varepsilon > 0$:*

$$h(\mathbb{L}, \bigcup_{n=1}^{N} w_n(\mathbb{L})) < \varepsilon$$

where h is the Hausdorff metric. Then:

$$h(\mathbb{L}, \mathcal{A}) < \varepsilon(1-s)^{-1}$$

where $\mathcal{A}$ is the attractor of the iterated function system.

Proof:

$$\begin{aligned} h(\mathbb{L}, W^n(\mathbb{L})) &\leq \sum_{m=1}^{n} h(W^{m-1}(\mathbb{L}), W^m(\mathbb{L})) \\ &\leq \sum_{m=1}^{n} s^{m-1} h(\mathbb{L}, W(\mathbb{L})) \\ &\leq \frac{1-s^n}{1-s}\, h(\mathbb{L}, W(\mathbb{L})) \end{aligned}$$

which as $n \to \infty$ becomes:

$$h(\mathbb{L}, \mathcal{A}) \leq (1-s)^{-1} h(\mathbb{L}, W(\mathbb{L})).$$

The implications of the Collage Theorem can be best appreciated by considering its consequences in the case of a two-dimensional space. If a set of the space is to be represented as an IFS, each mapping of the set can be considered as a reduced size, rotated, skewed copy of the original. These copies are then arranged in an effort to form a collage of the original set — hence Barnsley's name for the theorem. If a collage can be found which exactly covers the original set then the attractor of the IFS which consists of the mappings of the collage will be exactly the original set. Any differences between the collage and the original set result in the attractor differing by an amount related to the contractivity factor of the IFS. The problem of finding an IFS which has some set as its attractor reduces to finding a suitable collage. Since the mappings of the collage can be represented by real-valued coefficients the problem is now essentially one of subsymbolic manipulation, *i.e.*, searching the space of the subsymbolic representation.

3.6 Summary

This chapter has introduced the terminology of iterated function systems and given derivations of some of their important properties. The chapter began with a section on basic metric topology followed by one on mappings of a metric space. The next section defined a space $\mathcal{H}(\mathbb{X})$, the points of which correspond to non-empty compact subsets of an underlying metric space $(\mathbb{X}, d)$. Important properties of this space were derived and the Hausdorff metric, h, introduced. With the preliminary work complete definitions of an iterated function system and its attractor were given. Other important results of IFS theory which were presented included those of robustness and the Collage Theorem. The robustness property proves the continuous dependence of an IFS's attractor on the IFS parameters themselves, and formalises the strong interaction between the parameters. Barnsley's Collage Theorem provides a means of calculating an IFS for a given subset of the space for which it is defined.

Chapter 4

Iterated Function Systems and Shape Representation

This chapter continues the detailed description of various aspects of the shape representation problem, and begins by introducing a formal framework for using IFSs in two-dimensional shape representation. Two Lemmas show that a good IFS representation can be found when the underlying space is either continuous or discrete. The discrete case is of particular relevance to computer images. Section 4.2 reviews several of the methods which have been suggested as a means of generating the attractor of an IFS. The minimum point plotting algorithm (used to generate the attractors in the experiments of Chapter 5) is introduced.

The robustness property of IFSs is discussed and a pictorial demonstration shows the strong interactions which can occur between the components of a real-valued subsymbolic encoding of an IFS. Section 4.4 reviews the literature on the problem of using IFSs for shape representation. This includes a discussion of several methods which have been proposed for manipulating the subsymbolic components of an IFS. The literature review is included since the results of Chapter 5 suggest that evolutionary programming offers a powerful means of approaching the problem, and in order to best appreciate the problem's difficulty a critical review of other approaches to solving it is given. The chapter concludes with a brief overview of other applications of IFSs and a summary.

4.1 Framework

This section describes a formal framework for the use of iterated function systems as a two-dimensional shape representation scheme. The results derived are taken from Giles (1990, pp. 58–64).

Definition 4.1.1 *Let* $(\mathbf{R}^2, d)$ *be a metric space consisting of the Euclidean plane* $\mathbf{R}^2$ *equipped with a suitable metric,* d*. A shape* $\mathbb{S}$ *is any set* $\mathbb{S} \in \mathcal{H}(\mathbf{R}^2)$.

Definition 4.1.2 *Let* $\{\mathbf{R}^2, w_n : n = 1, 2, \ldots, N\}$ *be an iterated function system. Then a shape* $\mathbb{S}$ *is represented by the IFS if:*

$$\lim_{n \to \infty} W^n(\mathbf{B}) = \mathbb{S} \qquad \forall\, \mathbf{B} \in \mathcal{H}(\mathbf{R}^2).$$

In representing a shape by an IFS the aim is, therefore, to find an IFS which has that shape as its attractor. The following Lemma shows that for any shape a 'good' IFS representation can always be found.

Lemma 4.1.1 *Given any shape* $\mathbb{S}$ *and* $\varepsilon > 0$ *there exists an iterated function system* $\{\mathbf{R}^2, w_n : n = 1, 2, \ldots, N\}$ *with attractor* $\mathcal{A}$ *for which* $h(\mathbb{S}, \mathcal{A}) < \varepsilon$.

Proof: The Collage Theorem (3.5.3) gives:

$$h(\mathbb{S}, \mathcal{A}) < (1 - s)^{-1} h(\mathbb{S}, W(\mathbb{S})) \qquad \forall\, \mathbb{S} \in \mathcal{H}(\mathbf{R}^2).$$

Since $\mathbb{S}$ is compact it is closed and totally bounded, and there exists an ε-net $\{y_1, y_2, \ldots, y_N\}$. Since an ε-net contains only a finite number of points it is closed and totally bounded and so $\{y_n\}_{n=1}^N \in \mathcal{H}(\mathbb{R}^2)$. Furthermore:

$$h(\mathbb{S}, \{y_n\}_{n=1}^N) < \varepsilon.$$

Mappings are now chosen such that:

$$w_n(x) = y_n \qquad \forall\, x \in \mathbb{S} \text{ and } n = 1, 2, \ldots, N.$$

Hence $W(\mathbb{S}) = \{y_n\}_{n=1}^N$ with contractivity factor $s = 0$. Finally:

$$\begin{aligned} h(\mathbb{S}, \mathcal{A}) &< 1^{-1} h(\mathbb{S}, \{y_n\}_{n=1}^N) \\ &< \varepsilon. \end{aligned}$$

Clearly the above method is very inefficient. Each point of the ε-net requires one mapping and so when ε is very small a large number of mappings is needed.

So far the discussion of shapes and IFSs has been for the case in which the underlying space is continuous. In reality computer images are usually represented using a two-dimensional pixel array. The following Lemma shows that a good IFS code can be found for any pixelised image.

Lemma 4.1.2 *Let* $\mathbf{P} \in \mathcal{H}(\mathbf{X})$ *be a finite rectangular array of points in the metric space* $(\mathbf{R}^2, d)$*, such that* $(\mathbf{P}, d)$ *is a complete metric space. Let* $\mathbf{S}' = \{p_n \in \mathbf{P} : n = 1, 2, \ldots, N\}$ *be a set of points such that* $\mathbf{S}'$ *is the discrete approximation of a set* $\mathbf{S} \in \mathcal{H}(\mathbf{R}^2)$*. Then there exists an IFS* $\{\mathbf{P}, w_n : n = 1, 2, \ldots, N\}$ *with an attractor* $\mathcal{A}'$ *for which* $h(\mathcal{A}', \mathbf{S}') = 0$.

Proof: Choose W_n such that:

$$w_n(x) = p_n \qquad \forall\, x \in \mathbf{S}' \text{ and } n = 1, 2, \ldots, N.$$

The set of mappings $\{w_n\}_{n=1}^{N}$ has contractivity $s = 0$ and $W(\mathbb{S}') = \mathbb{S}'$. Hence from the Collage Theorem (3.5.3):

$$h(\mathcal{A}', \mathbb{S}') = 1^{-1} h(\mathbb{S}', W(\mathbb{S}')) = h(\mathbb{S}', \mathbb{S}') = 0.$$

As for Lemma 4.1.1 the number of mappings needed is large. In fact the mappings used are little more than a pixel list written in terms of mappings. Again this is very inefficient, but it does show that any shape defined on a pixel array can be represented by an IFS.

The problem of finding an IFS for an arbitrary shape is often referred to as the inverse problem (Barnsley *et al.* 1986). Although the material of this section shows that for pixelised images it is always possible to solve the inverse problem, the method is very inefficient. An obvious question is then: Is it possible to produce a good approximation using fewer mappings? Peitgen *et al.* (1992, p. 281) mention the following difficulties which arise within this context:

1. How can the quality of an approximation be assessed? How are the differences between images quantified?

2. How can suitable transformations be identified?

3. How can the number of necessary affine transformations be minimised?

4. What is the appropriate class of images suitable for this approach?

The answers to each of these are research problems in their own right and varying amounts of progress have been made. For example, question 2 has in theory been solved by Barnsley with the Collage Theorem. The practical difficulties which arise are, however, non-trivial and have resulted in a large body of literature on the best way to proceed. Possible solutions to some of these questions are discussed further in Section 4.4 and Chapter 5.

4.2 Generating the Attractor

In this section several methods for generating the attractor of an IFS are discussed. These methods can be classified as either stochastic or deterministic. Examinations of stochastic methods for generating attractors include those by Barnsley (1988), Berger (1989), Goodman (1991), Hepting *et al.* (1991), Smith (1991), and Culik II and Dube (1993). Deterministic methods for attractor generation include those by Barnsley (1988), Monro *et al.* (1990), Hepting *et al.* (1991), Kropatsch *et al.* (1992) and Cohen (1992). In addition Sharp and Cripps (1991) describes a CIMP (Communication Intensive Massively-Parallel) algorithm which is a deterministic method of attractor generation most suited to implementation on a parallel architecture.

Stochastic methods are usually simple to implement and can provide a quick approximation of the attractor. However, due to their very nature, it is often difficult to decide when they should be terminated. Deterministic algorithms on the other hand are often harder to implement and can be computationally expensive. The remainder of this section discusses an example of a stochastic and deterministic method for attractor generation, and concludes with the algorithm that shall be used to generate attractors in the experiments discussed in Chapter 5.

4.2.1 The Random Iteration Algorithm

The Random Iteration Algorithm (RIA) developed by Barnsley (1988; Barnsley and Sloan 1988) is perhaps the most commonly used method of attractor generation. The RIA was developed by considering an IFS as a probabilistic dynamical system, and as such being stochastic in nature. Each of the contraction mappings of an IFS $\{w_i : i = 1, 2, \ldots, n\}$ is assigned a probability p_i. These probabilities are usually calculated by writing the mappings in the form $w_i(x) = A_i x + t_i$ (recall Definition 3.2.3) and taking:

$$p_i \approx \frac{|\det A_i|}{\sum_{i=1}^n |\det A_i|} = \frac{|a_i d_i - b_i c_i|}{\sum_{i=1}^n |a_i d_i - b_i c_i|} \qquad \text{for } i = 1, 2, \ldots, n.$$

In other words the probability is approximately proportional to the size of the contraction that each mapping produces, and of course $\sum_{i=1}^n p_i = 1$. The approximation is needed so that if, for some i, $\det A_i = 0$, the probability p_i can be assigned a small positive number, such as 0.001. The RIA then proceeds as follows:

1. Choose an initial point $x_0 \in \mathbf{X}$.
2. Select (with replacement) a mapping, w_i, of the IFS with a probability p_i.
3. Generate and store the point $x_1 = w_i(x_0)$.
4. Select (with replacement) a mapping, w_j, of the IFS with a probability p_j and apply it to the point x_1 to generate x_2.
5. Continue in this manner to produce the set of points $\{x_0, x_1, x_2, \ldots, x_N\}$.
6. For large enough N the set $\{x_i\}_{i=0}^N$ is, almost surely, a good approximation of the attractor of the IFS.

That the sequence of points converges (almost surely) to the attractor is ensured by Elton's ergodic theorem (Barnsley 1988, p. 370). The initial point x_0 need not lie on the attractor since the sequence will converge to the attractor. It is, therefore, usual to disregard the initial part of the sequence while it converges to the attractor. An obvious problem is deciding exactly how many points are to be disregarded. This can easily be avoided by selecting the initial point x_0 to be a fixed point of one of the mappings of the IFS. Since this point lies on the attractor so do all of the subsequent points generated.

The RIA is both simple to implement and has a certain aesthetic appeal, especially if used to generate colour attractors (Hepting *et al.* 1991; Smith 1991). However, due to the stochastic nature of the algorithm it is not possible to determine in advance the number of iterations required to generate the complete attractor. Figure 4.1 shows examples of using the RIA to generate attractors of the IFSs given in Tables 4.1 — 4.3.

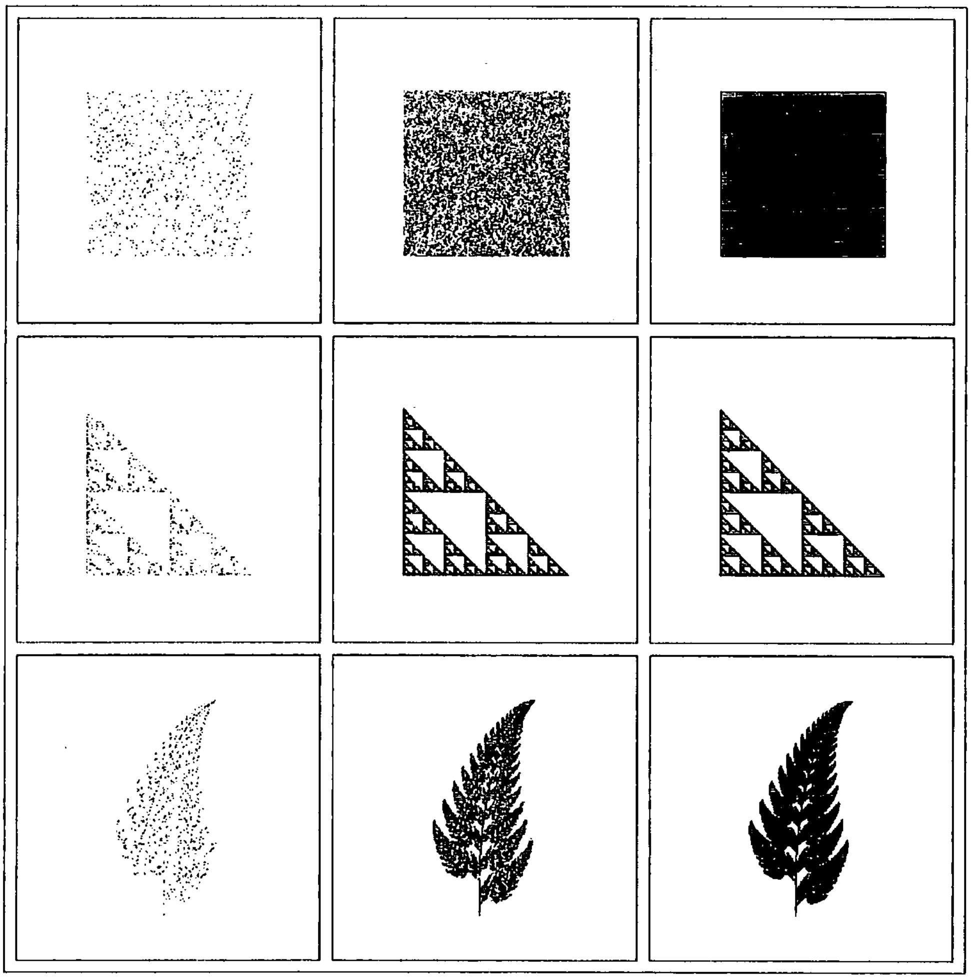


Figure 4.1: Using the random iteration algorithm to generate a square, Sierpinski triangle and a Barnsley Fern. The diagrams show the algorithm after 1000, 10000 and 100000 iterations. Even after 100000 iterations the attractor for the square has not been completely generated, although this may not be clear in the diagram (the fully rendered square consists of 19881 points).

i	a_i	b_i	c_i	d_i	e_i	f_i	p_i
1	0.5	0	0	0.5	35	35	0.25
2	0.5	0	0	0.5	-35	35	0.25
3	0.5	0	0	0.5	35	-35	0.25
4	0.5	0	0	0.5	-35	-35	0.25

Table 4.1: Coefficients of an IFS for a square.

i	a_i	b_i	c_i	d_i	e_i	f_i	p_i
1	0.5	0	0	0.5	-35	35	0.333
2	0.5	0	0	0.5	35	-35	0.333
3	0.5	0	0	0.5	-35	-35	0.334

Table 4.2: Coefficients of an IFS for a Sierpinski triangle.

i	a_i	b_i	c_i	d_i	e_i	f_i	p_i
1	0	0	0	0.16	0	0	0.01
2	0.2	-0.26	0.23	0.22	0	1.6	0.07
3	-0.15	0.28	0.26	0.24	0	0.44	0.07
4	0.85	0.04	-0.04	0.85	0	1.6	0.85

Table 4.3: Coefficients of an IFS for a Barnsley Fern.

4.2.2 A Deterministic Algorithm

The following algorithm for generating the attractor of an IFS was suggested by Barnsley (1988, p. 88) and follows directly from the definition of an attractor.

If $\{X, w_i : i = 1, 2, \ldots, n\}$ is an IFS then the algorithm proceeds as follows:

1. Choose a non-empty compact set $\mathbb{A}_0 \subset \mathbf{R}^2$.

2. Compute successively $\mathbb{A}_N = W^N(\mathbb{A}_0)$ according to:

$$\mathbb{A}_{N+1} = \bigcup_{i=1}^{n} w_i(\mathbb{A}_N) \qquad \text{for } N = 0, 1, 2, \ldots .$$

3. The sequence $\{A_N \subset \mathcal{H}(\mathbb{X})\}_{N=0}^{\infty}$ converges to the attractor of the IFS in the Hausdorff metric — Theorem 3.5.1.

Although no termination criteria has been specified for the above algorithm it is easy to include one. For example, terminate when $h(\mathbb{A}_{n-1}, \mathbb{A}_n) < \varepsilon$ for some predetermined $\varepsilon > 0$. A criteria such as this has the added advantage that it allows the attractor to be generated to any desired degree of accuracy.

The major drawback of this algorithm is that there is no guarantee as to the rate of convergence of the sequence $\{\mathbb{A}_N\}$, and so a large number of iterations may be required. Furthermore, many unnecessary points will often be plotted, especially when the initial shape $\mathbb{A}_0$ is far larger than the attractor. This algorithm is, therefore, usually slow in generating the attractor of an IFS and so is unsuitable for the experiments discussed in Chapter 5.

4.2.3 The Minimum Point Plotting Algorithm

The Minimum Point Plotting algorithm (MPP) was suggested by Monro *et al.* (1990). It was noted that the RIA "visits pixels many times so that it will eventually leave all pixels by all possible routes." Monro *et al.* suggest a method for generating an attractor based upon this fact. Given some points which lie on the attractor the application of all the mappings of the IFS to these points generates all possible paths from these points. A queue of new points to be transformed is created. Points that have already been plotted are not added to the queue. When the queue is empty the attractor has been generated. The fixed points of the mappings of an IFS lie on the attractor and so can be used as starting points. This can be summarised as:

1. Calculate the fixed points of the mappings $\{w_i : i = 1, 2, \ldots, n\}$ plot them and place them in the queue.
2. Take the point, x, from the head of the queue. For each of the mappings w_i, $i = 1, 2, \ldots, n$:
   (a) Generate the point $y = w_i(x)$.
   (b) If y is a point which has not been plotted, plot it, and add to the back of the queue.
3. Repeat 2 until the queue is empty.

Although the above algorithm generates points which have already been generated, these will not be plotted or added to the queue. The checking of whether or not a point has been generated can be efficiently implemented by use of an array — although care must be taken to ensure that it is large enough to record all points generated.

It is perhaps worth noting a potential problem (not considered by Monro *et al.*) that may arise as a result of rounding errors. If the attractor is large or complicated enough it is possible that the rounding (or truncation) of fractional parts of values may be enough to alter the value produced by the application of a mapping. In such cases some spurious points may be plotted, and other points that are part of the attractor may not be plotted. However, these potential difficulties can be expected to produce only minor variations in the attractor. It is possible to reduce the effect of rounding errors by refining the array so that it takes into account a number of decimal places. However, this can be expected to increase the time taken to generate an attractor. Attractors shown on a computer are, in any case, often only approximations since only a finite degree of accuracy is available.

The MPP algorithm is simple and fast when compared to the RIA (Monro *et al.* 1990), and is used throughout the rest of this thesis to generate attractors.

4.3 Robustness

An important property of IFSs is that small variations in the mapping coefficients result in small changes in the attractor. This property is referred to as robustness and is a direct consequence of the results of Section 3.5, which establish that if the mapping coefficients are continuous in some parameter, then so is the attractor.

The robustness property is demonstrated in Figure 4.2. The sequence of attractors shows the affects on the attractor of altering, by varying amounts, the IFS coefficients for a Sierpinski triangle. The coefficients used are shown in Tables 4.4 — 4.7. It can be seen that the introduction of a few small errors produces little discernible change in the attractor. When these small errors become more numerous appreciable differences in the attractor can be seen, although the shape is still (to the human eye at least) recognisable. However, with the introduction of a few large errors the attractor begins to break up, and with the introduction of more large errors it can be expected to degenerate further. This behaviour demonstrates the strong interaction that can occur between individual components of a mapping.

Interactions between the mappings themselves can be even more pronounced since it is the interaction of all of the mappings which determines the attractor.

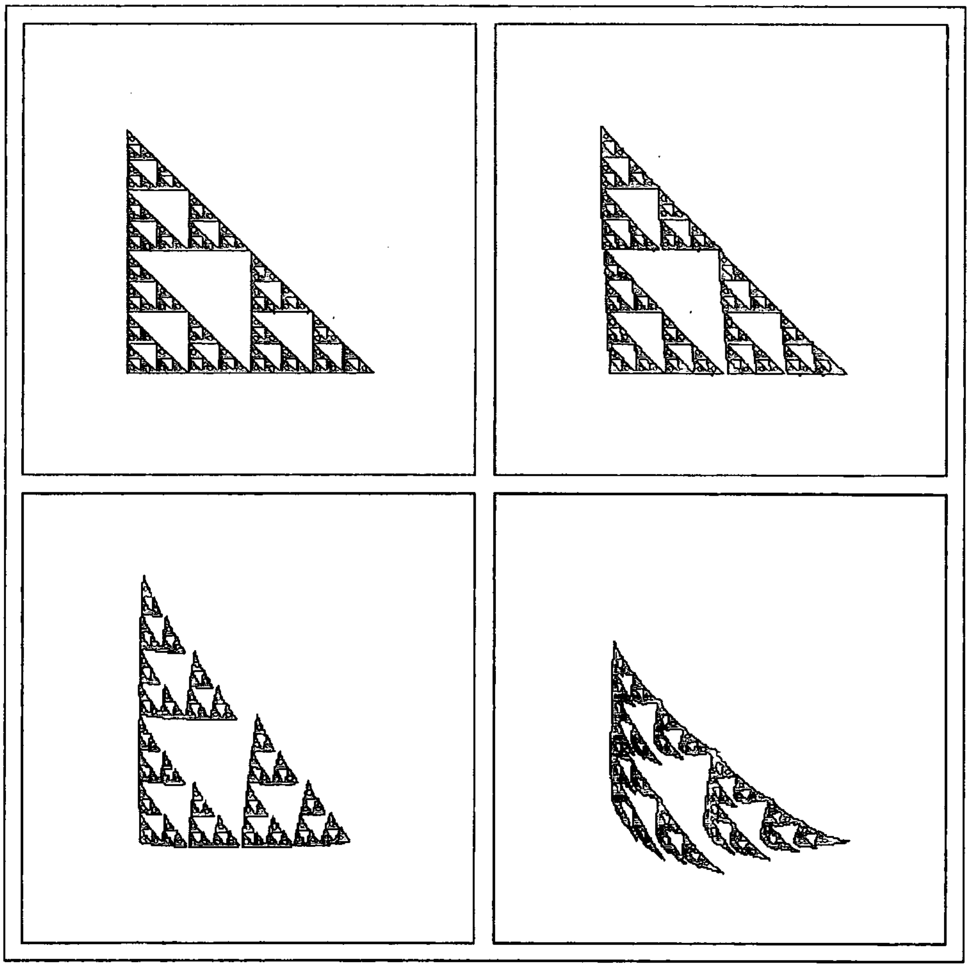


Figure 4.2: A sequence of attractors obtained by altering the coefficients of an IFS for a Sierpinski triangle by various amounts. The top left attractor is the original. The top right has a few of the original coefficients altered only slightly. The bottom left has all of the original coefficients altered slightly. The bottom right has most of the coefficients altered sightly and the remainder altered by a larger amount. This demonstrates the strong interaction which can occur between components when a real-valued subsymbolic representation is adopted.

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.5	0	0	0.5	-35	35
2	0.5	0	0	0.5	35	-35
3	0.5	0	0	0.5	-35	-35

Table 4.4: Coefficients of an IFS for a Sierpinski triangle.

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.5	**0.01**	0	0.5	-35	**36**
2	0.5	**-0.02**	0	0.5	**34**	-35
3	**0.48**	0	0	0.5	**-33**	-35

Table 4.5: Coefficients of an IFS for a Sierpinski triangle with a few small changes (shown in bold).

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.46	0.01	-0.03	0.52	-33	38
2	0.51	0.05	0.02	0.49	31	-36
3	0.47	-0.01	-0.02	0.48	-33	-37

Table 4.6: Coefficients of an IFS for a Sierpinski triangle with many small changes.

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.46	0.01	-0.03	0.52	-33	**20**
2	**0.62**	0.05	0.02	0.49	31	-36
3	0.47	-0.01	**-0.23**	0.48	-33	-37

Table 4.7: Coefficients for a Sierpinski triangle with many small and a few larger changes (shown in bold).

4.4 The Inverse Problem

The inverse problem is the name often given to the problem of finding an IFS for an arbitrary shape (Barnsley *et al.* 1986). This section discusses several methods that have been suggested for solving two-dimensional inverse problems. The review is not exhaustive, but provides an overview of some of the approaches which have been considered. The literature review is included so that the results presented in the following chapter can be placed in the context of other work on solving inverse problems.

The methods examined fall into two main categories: interactive and automatic. Interactive methods generally use a symbolic representation, while automatic methods usually adopt a subsymbolic approach and some algorithm to search the space of encodings. Interactive methods require a human to guide the search, and as such are far from ideal. Automatic approaches on the other hand often need to resort to simplifying assumptions in order to find solutions which usually makes them only suitable for application to a subset of inverse problems.

4.4.1 ISIS

The Interactive System for Image Synthesis (ISIS) tool was developed by Horn (1989) and is used for collage construction. The tool consists of two main parts: a drawing surface for IFS design, and a drawing surface for image rendering and viewing.

The IFS design process is carried out by the manipulation of a set of parallelograms. Each parallelogram represents a contraction mapping of the original shape, and the manipulation is carried out by the use of a mouse which is used to meta-skew a parallelogram by 'dragging' a vertex. Additional parallelograms can be created or current ones destroyed as required by the user. The set of mappings of the parallelograms is an IFS, and, as alterations are made, the attractor of the

IFS is generated in the viewing window. The attractor generation is carried out using a massively parallel computer and as such is extremely fast. At any time the attractor may be overlaid over the drawing surface to aid design. The system can be used to solve inverse problems, but being interactive has obvious limitations.

4.4.2 Skeletonisation

Libeskind-Hadas and Maragos (1987) use the morphological skeleton of the shape to be encoded as a tool to provide information on the values of the coefficients necessary to produce a collage of affine maps (IFS) that cover the original.

If A is a set in $\mathbf{R}^2$ then the disc rD_x is maximised with respect to A if it is contained in A and is not properly contained in any other disc contained in A. The skeleton of A, is defined to be the set of centres of all discs maximal with respect to A. The skeleton can also be defined using morphological operations, and the above concepts can be applied to finding the skeletons of discrete binary images.

When the tool is applied to an image, the skeleton is computed, and the user locates the central branch point and outer branch points. These outer points are then used to calculate the coefficients of mappings from which a collage is built. The system works well for perfect self-similar fractals (those which display the same structure at all scales) by enabling the discovery of collages which fit the image boundary. More generally, however, the mappings generated may not cover the image and so the remainder is covered by discs. The system, although useful, is not automatic and often requires the use of shapes that are not affine transformations of the original.

4.4.3 Iterative Minimisation

Levy-Vehel and Gagalowicz (1987) introduce several distance measures between shapes and use them to quantify the distance between a shape and a collage of it. A gradient based algorithm is used to iteratively minimise one of the distance functions so that a suitable collage for the shape can be found. The results for some of the distance functions are reported to be, in general, poor, but one example of a successful application of the procedure is given. The major drawback of the approach is in the selection of the starting collage. If this is badly chosen then the algorithm is not able to find a good collage. To overcome this it was necessary to select the starting collage by hand, and as such the procedure is not totally automatic.

4.4.4 Boundary Mapping

Giles *et al.* (1989) suggests reducing the collage construction process to an essentially one-dimensional problem by selecting mappings which match to the shape boundary only, thus significantly reducing the search space complexity.

The input data for this algorithm is a digitised binary image of a simple geometric shape. The boundary points of the image are detected and the curvature κ at each point calculated. The boundary is segmented into arcs, the endpoints of which correspond to $\kappa = 0$. The arcs are then classified as either concave, convex or linear. Contractive affine transformations are calculated which map arcs of the same curvature type onto each other, the best being accepted as part of the IFS. If no suitable match is found, the unmatched arc is halved and reconsidered.

The best encodings produce attractors which give good approximations of image boundaries. However, although an automatic system, the mappings used are in general small and thus unable to cover adequately the image interior.

4.4.5 Evolutionary Algorithms

Previous applications of EAs to the inverse problem have considered both the use of GAs and EP. Vrscay (1991a; 1991b) considered several approaches to solving inverse problems, one of which involved the application of a GA. A brief overview of the theory of GAs and how they can be applied to inverse problems was given. Although encouraging results were reported, only preliminary details for one-dimensional problems were provided and there were no details of any results for two-dimensional problems. Garigliano *et al.* (1993c) [1] and especially Giles (1990) discussed in some detail the implementation of a GA to two-dimensional inverse problems and showed some promising results for a selection of target shapes.

Levy-Vehel and Lutton (1993) presented the application of a GA to some two-dimensional inverse problems. Two target shapes were considered: a Barnsley Fern, and a perturbed Sierpinski triangle. The algorithm was reported to be capable of finding near optimal solutions, but no details of its mean performance was given.

Hoskins and Vagners (1992) have applied EP to the inverse problem with some success. The approach taken assumed knowledge of the size of the linear deformations, and searched for the number of mappings required together with their translation components. The technique used to achieve this "simulates the competitive dynamics on an array of mapping cells." Each element of a 192 × 192 array was used to represent a mapping, the fixed point of which corresponded to its position in the array. The population consisted of individual mappings, and the goal was the determination of an optimal population of individuals. Four test cases were presented which showed that (within its limitations) the algorithm was able to exactly solve two of the problems: a Sierpinski triangle, and a Twin Dragon fractal. The remaining two problems were a noisy image and two embedded objects, the exact solutions of which were not known. Promising results for both of these cases were given. Although the technique is useful it is clearly limited by the

[1] The sequence of results shown in Garigliano *et al.* (1993c) was for a population of 500, not 100 as implied.

requirement of needing *a priori* knowledge of the linear deformations.

4.4.6 Moment Approach

In addition to the geometric theory of IFSs they can be characterised in terms of measures. A full account of this approach is given by Barnsley (1988).

In the case of the measure approach a gray scale image is modelled as a probability measure over its region of support. A contractive mapping is defined in terms of a Markov operator and the metric used is known as the Hutchinson metric. The successive application of the Markov operator to an arbitrary initial distribution results in convergence to a unique measure. This is termed the invariant measure and obeys a fixed point condition with respect to the Markov operator. The support of the invariant measure is the attractor of the IFS.

For a target shape $\mathbb{S}$ the inverse problem for measures involves finding an IFS whose attractor $\mathcal{A}$ minimises the Hutchinson distance $d_H(\mu, \nu)$ where the support $supp(\mu) = \mathcal{A}$ and $supp(\nu) = \mathbb{S}$. A Collage Theorem for measures exists and is conceptually similar to that for the geometric approach.

Vrscay (1991a; 1991b) examined the use of a gradient based approach for solving one-dimensional inverse problems, but found it to be very unstable. A two-dimensional inverse problem was considered, and the results so bad that Vrscay (1991a, p. 447) states, "This example shatters any hopes of using gradient methods for moment matching as a global optimisation method." It is suggested that the method may prove useful as a fine-tuning procedure for a more global search procedure.

The moment method has also been applied to the one-dimensional case by Abenda and Turchetti (1989). A two-dimensional approach involving wavelets was suggested by Rinaldo and Zakhor (1992), but the procedure is only applicable to inverse problems whose IFS mappings are of a particular very restricted form.

In addition the mappings are assumed to be non-overlapping thus making the technique unsuitable for many inverse problems.

In this thesis only black on white images are considered and a geometric approach to the inverse problem is sufficient. Barnsley (1988) provides further details of the measure theory of IFSs.

4.4.7 Modelling One-dimensional Data

IFSs have been suggested as a means of modelling one-dimensional discrete data. Vines and Hayes (1992) suggest two possible algorithms for this based on the Collage Theorem. The algorithms split the data into smaller sections, and find mappings which minimise the squared error between the original data and the data when mapped into the smaller sections. The first algorithm encounters difficulties in determining the endpoints of the smaller sections which are to be used. Although some constraints are applied, the algorithm is required to exhaustively evaluate a large number of possibilities, which results in much wasted computation. The second algorithm uses a linear interpolation of the data to provide the endpoints of the smaller sections, and a least squares approximation is used to optimise the parameters of the individual mappings. As discussed by Mazel and Hayes (1992) the above (self-affine) fractal interpolation algorithms are only really suitable for use on data which is itself self-affine. Since most real world data is not self-affine Mazel and Hayes suggest a piecewise self-affine fractal model. Adopting this approach the data is viewed as being composed of a collage of transformations of pieces of the data and not transformations of the whole data set (as in the self-affine case). This approach while being more general resulted in a large number of extra degrees of freedom and restrictions needed to be placed on mappings. Using this model Mazel and Hayes (1992) show successful models of several data sequences.

4.5 Other Applications of IFSs

The main aim of this chapter has been to continue the detailed description of how IFSs can be used for two-dimensional shape representation, and to survey some of the approaches which have been considered. There are, however, several other useful applications of IFSs which rely on solving corresponding inverse problems, and some of these are now briefly discussed.

Barnsley (1988; 1993) promoted a scheme for encoding images by using what he called a fractal transform. The compression ratios achieved for images encoded using such a scheme are reported to be very favourable. Barnsley has founded a company, Iterated Systems, that produces software for compression/decompression (Georghiades and Jacobs 1992), but has not released details of the method used. Jacquin (review of the technique in Jacquin 1992) was the first to publish a full account of a method for fractal compression and a block based approach was adopted. (This requires that the image be broken up into small blocks, and the discovery of mappings which map blocks of different sizes onto each other.) Recent reviews and improvements of this technique include those by Waite (1990), Fisher (1992), Beaumont (1990), and Monro and Dudbridge (1992). The advantage of this fractal based approach, over some other methods of image compression, is that the encoding/decoding process has no need to refer to any external library. The image instead being encoded using parts of itself. While the decoding of images is very fast, the time taken to carry out the encoding appears to be the major drawback of the approach. Another method for fractal based encoding of images has been suggested by Hollatz (1991).

Sharp and While (1992) suggest a means of automatically recognising and classifying fractally encoded images. They state "The aim of fractal recognition is to take an unknown image and determine which, if any, of a library of fractals was used in its generation" (Sharp and While 1992, p. 6). Each element of their library is an IFS encoding for a shape. The approach presented uses an algorithm which repeatedly applies an IFS from the library to an unknown image. The algorithm is described as one which attempts to sustain images rather than generate them. If the image is identical to the attractor of the IFS it remains intact. However, if the image is 'distant' from the attractor of the IFS then instead of being transformed into the attractor it gradually disintegrates and eventually disappears. Finally, if the image is 'similar' to the attractor of the IFS then the algorithm refines the image toward the attractor. This allows for the recognition of images in the presence of noise. An important extension to the algorithm allows for the recognition of displaced, rotated and scaled images. The approach is, however, limited by the fact that it is necessary to apply the algorithm to each of the encodings in the library until one is found which does not disintegrate. For a large library this could be prohibitively time consuming. In addition, for a shape to be added to the library its IFS encoding needs to be found, and no automatic means of doing this is presented, *i.e.*, to add a shape, the inverse problem for that shape still needs to be solved.

Watt (1993) suggests the use of IFSs for modelling intelligent behaviour. The behaviour of biological systems is often complex, but this may in part be attributed to dynamical processes contained within the system. The behaviour of dynamical systems can appear to be very complex, but may be controlled by simple rules (Gleick 1988). Watt demonstrates how a generic solution to the Towers of Hanoi problem (see Figure 4.3) can be represented by an IFS. The attractor of the IFS is a path through the problem's state space. From the collage of mappings, the physical procedure for solving the Towers of Hanoi problem can be inferred, and the underlying principle for generalising the method is described as looking "for patterns in behaviour which allow a simple set of rules to describe apparently complex behaviour." The major difficulty which arises is the discovery of a suitable collage to describe a path through a problem's state space, *i.e.*, solving the corresponding inverse problem.

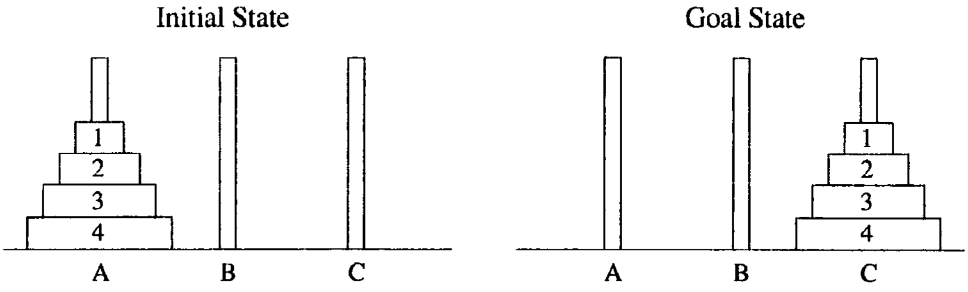


Figure 4.3: The initial and goal states for the Towers of Hanoi problem. The aim is to move all the rings from one peg to another. The rules are that only one ring can be moved at a time, a ring can only be moved when there are no rings on top of it, and no ring may be placed on a smaller ring.

4.6 Summary

This chapter has completed the formalism necessary for the use of IFSs as a shape representation scheme. A subsymbolic representation can be adopted which consists of the real-valued components of the mappings of an IFS. Methods for generating the attractor of an IFS have been discussed, and the MPP algorithm selected for use in the experiments of the following chapter. The robustness property of IFSs demonstrates that there is a strong interaction between the subsymbolic components. A comprehensive review of the literature on IFS shape representation has been included so that the work in the next chapter can be placed in the context of other work on the subject.

Chapter 5

Evolutionary Algorithms and the Inverse Problem

This chapter is the first of the experimental chapters of this thesis, and explains in detail how a genetic algorithm (GA) and evolutionary programming (EP) can be applied to the inverse problem for two-dimensional shapes. The GA and EP presented are often considered to be their simple versions — an account of the alternatives which have been suggested is beyond the scope of this thesis. Three hill-climbing algorithms are also introduced.

Solutions to inverse problems are encoded subsymbolically (binary string for the GA and hill-climbing algorithms, and real-valued for EP) and this results in strong interactions occuring between blocks of components as well as between individual components of the encodings. The GA, EP and hill-climbing algorithms are applied to several inverse problems. Results presented show the success of EP, but indicate that the GA and the hill-climbers algorithms are not as successful. Reasons for this disparity are discussed (Nettleton and Garigliano 1994c, 1994d, 1994g).

5.1 The Inverse Problem

As discussed in Section 3.5 a shape can be represented as an IFS by choosing contraction mappings such that they form a collage that exactly covers the original shape. In essence the problem is symbolic with the shape being recreated from a collection of smaller copies of that shape. Two examples of collages, one for a square, the other for a triangle, are shown in Figure 5.1.

The 'smaller copies' which are used in the construction of a collage can, however, be represented as contraction mappings and a real-valued subsymbolic representation adopted. The coefficients of the mappings used in Figure 5.1 are given in Table 5.1 and are of the form:

$$w_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix} \quad \text{where} \quad i \in \{1, 2, \ldots, n\}. \tag{5.1}$$

Figure 5.1: A square and a triangle each of which is decomposed into a collage of smaller copies of themselves.

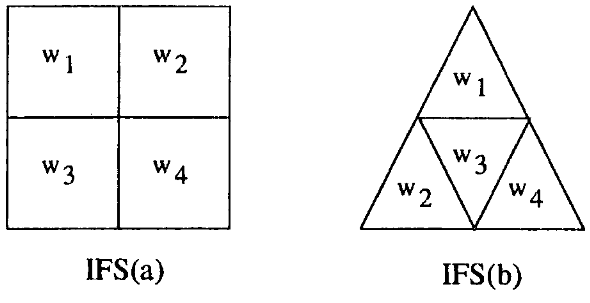


Table 5.1: The coefficients of the contraction mappings which produce the collages given in Figure 5.1

	IFS (a)						IFS (b)					
i	a_i	b_i	c_i	d_i	e_i	f_i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.5	0.0	0.0	0.5	-25	25	0.5	0.0	0.0	0.5	0	25
2	0.5	0.0	0.0	0.5	25	25	0.5	0.0	0.0	0.5	-25	-25
3	0.5	0.0	0.0	0.5	-25	-25	0.5	0.0	0.0	-0.5	0	-25
4	0.5	0.0	0.0	0.5	25	-25	0.5	0.0	0.0	0.5	25	-25

The problem is now one of manipulating the subsymbolic components in order to find a suitable solution. The following simplifying assumptions are made:

1. Number of mappings fixed in advance — the number of mappings which an IFS contains is fixed prior to an algorithm's run. Fixed length solution encodings can be used, thus allowing for the simple versions of the GA and EP to be applied. More sophisticated versions of these algorithms exist which allow for varying length encodings, but these are not considered in this thesis.

2. Mappings coefficients in non-trivial ranges — no knowledge of the mapping coefficients is assumed except that they lie within some non-trivial ranges. (The constraints derived in the following chapter are not applied, since, although they considerably reduce the search space, they are not necessary when comparing the performance of algorithms. Furthermore, if they were to be applied, then the GA would require problem specific operators to ensure that mappings produced as a result of crossover and/or mutation are contractive. The ranges used ensure that in the cases of the GA and the hill-climbers all the possible binary strings represent valid solutions.)

Even with the above simplifications the search spaces for the problems discussed are very large, and highly complex. There are typically several optimal and many suboptimal solutions — see Section 5.1.4. The remainder of this section discusses how EAs can be applied to inverse problems for which a subsymbolic representation has been adopted.

5.1.1 Environment

The environment of an EA is the source of information on which the fitness of solutions is evaluated. An environment for shape representation must in theory be able to cope with any shape and hence $\mathcal{H}(\mathbf{R}^2)$ would be suitable. For convenience, however, shapes are shown on a computer screen and so this is the environment used for the development of solutions to the inverse problem. More specifically a 257×257 grid is used on which the shape to be encoded is plotted. The elements of the grid which are labelled 1 are points of the shape, the remaining elements being labelled 0.

Given an arbitrary target shape (the shape which is to be encoded as an IFS) a simple image processing algorithm is used to calculate the following information:

1. The number of pixels of the shape.
2. The coordinates of the centroid of the shape.
3. The maximum extent of the shape along the abscissa and ordinate axis.

Without loss of generality the target shape is translated so that its centroid lies at the centre of the 257×257 grid.

5.1.2 Solution Representation

In natural evolution, genetic material contains the information required to generate an organism (although its development depends in part on external environmental conditions). These genes can be considered as the building blocks necessary for the construction of an individual. In a similar way, the contraction mappings of an IFS are the building blocks necessary for solving an inverse problem, and each contraction mapping can be encoded as a string of numbers — this then becomes the genetic material of the inverse problem. In order to ensure that the EAs always produce valid solutions, the coefficients of the contraction mappings (see Equation 5.1) are constrained such that, for all i:

$$-0.707 \leq a_i, b_i, c_i, d_i \leq 0.707$$

$$-XMAX \leq e_i \leq XMAX \quad -YMAX \leq f_i \leq YMAX \tag{5.2}$$

where $XMAX$ and $YMAX$ are the maximum extent of the shape in the x and y directions (Giles 1990, pp. 146). The constraints on a_i, b_i, c_i and d_i ensure that the mapping will be contractive. The constraints on e_i and f_i are derived from noting that once the original has been contracted it should not be moved by an amount exceeding the dimensions of the box containing the original shape. Such a transformation would produce an unsuitable collage since some point of the collage would lie outside of the original shape.

The constraints derived in the following chapter are not applied due to the problems which would occur in their application to the GA. With a fixed length binary representation, and no sophisticated crossover operator, the GA could produce invalid solutions (with respect to the constraints) which EP could never produce. Although invalid solutions are easily penalised this might bias a comparison of the two approaches in favour of EP.

5.1.3 Fitness Function

The quality (fitness) of possible solutions to the inverse problem relies on a quantitative measure of how closely two shapes resemble each other, and can be assessed with a variety of techniques (Levy-Vehel and Gagalowicz 1987; Mumford 1987). Two fitness functions shall be considered, both of which use point coverage. The following piece of pseudocode demonstrates how point coverage between two shapes is calculated. Each of the two shapes to be compared are represented by a 257 × 257 grid. The elements of the grid which are labelled 1 are points of the shape, the remaining elements being labelled 0. If two shapes are represented using the grids shape[x][y] and image[x][y] then the point coverage between them is calculated using:

```
fitness := 0 ;
for x := 0 to 256 do
begin
    for y := 0 to 256 do
    begin
        if image[x][y]=1 and shape[x][y]=1 then
            fitness := fitness + 1 ;
        else if image[x][y]=0 and shape[x][y]=0 then
            fitness := fitness ;
        else
            fitness := fitness - 1 ;
    end ;
end ;
```

Given a shape to be encoded, **A**, and an IFS $\{w_1, w_2, \ldots, w_n\}$ then the the two functions used to calculate the fitness of the IFS are:

1. Attractor and point coverage — calculate the point coverage between **A** and the attractor of the IFS.

2. Collage and point coverage — calculate the point coverage between **A** and $\cup_{i=1}^{n} w_i(\mathbf{A})$.

The usual stochastic method for generating the attractor (Barnsley 1988, p. 91) is unsuitable since there is no termination criterion. The attractor is, therefore, generated using the deterministic MPP algorithm (Monro *et al.* 1990) — see Section 4.2.3. The grid on which the attractor is plotted is of a fixed size and it is possible that the attractor contains some points which lie outside of the grid. When such points are generated a penalty of 5 is subtracted from the current fitness and the point discarded. A more complex structure than an array could be used to keep track of the points plotted, *e.g.*, a binary tree, but this is much slower to access when checking to see if a point has been already plotted and would considerably slow the algorithm's execution. By ensuring that the target shapes are small and central with regards to the grid this problem is minimised.

The above fitness functions can result in both positive and negative values, and as such are not metrics. The optimum fitness value will be the maximum positive value attainable, and will be equal to the number of pixels of the shape which is to be encoded. The main advantage of using point coverage in the fitness function is speed. One disadvantage is its lack of sensitivity to shape structure. Other measures, such as the Hausdorff distance, may be used. Each of these have their own advantages and disadvantages over point coverage. For example, the Hausdorff distance is more sensitive to shape structure, but its calculation involves considerably greater computational expense.

The number of children that a parent IFS can produce is directly related to its fitness: the fitter the parent the greater the production. Once the required number of child IFSs have been produced they either 1) immediately become a parent population (GA) or 2) are combined with their parent population, the worst probabilistically culled and the remainder then become a parent population (EP). The process of child production is repeated, and in this way subsequent generations evolve toward optimal solutions. The details of the GA and EP used in the experiments of this thesis are given in Sections 5.2 and 5.3 respectively, but before that the fitness landscape is briefly investigated.

5.1.4 Cross-sections of the Search Space

The search spaces for the inverse problems considered in this chapter are very large and complex. In this section some cross-sections of the search space for the Sierpinski triangle are shown. The coefficients of an IFS for a Sierpinski triangle are given in Table 5.2. The IFS consists of three mappings, and hence 18 coefficients. All of these are fixed except for the two translation components of the third mapping: these are each allowed to take a value from the set $\{-50, -49, \ldots, 0, \ldots, 49, 50\}$. Figures 5.2 and 5.3 show IFS fitnesses when the attractor and point coverage, and the collage and point coverage, respectively, are used as the fitness functions. The optimum fitness lies at the point $(0, 25)$, but this is not actually plotted since interpolation of the data was required to allow the plots to be viewed more easily. The interpolation has also resulted in much detail being smoothed out, and many local optima are not visible. The real search space has many more local optima than the figures suggest.

Table 5.2: Coefficients of an IFS. Note all the values are fixed except for those of e_3 and f_3 which are each allowed to take a value from the set $\{-50, -49, \ldots, 0, \ldots, 49, 50\}$. The Sierpinski triangle used in the fitness function is the attractor of the IFS for which $\mathbf{X} = 0$ and $\mathbf{Y} = 25$. Figures 5.2 and 5.3 show IFS fitnesses, for a range of $\mathbf{X}$ and $\mathbf{Y}$ values, when compared to the Sierpinski triangle.

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.5	0	0	0.5	-25	-25
2	0.5	0	0	0.5	25	-25
3	0.5	0	0	0.5	$\mathbf{X}$	$\mathbf{Y}$

Figure 5.2: Cross section of the search space for the Sierpinski triangle with the fitness function of attractor and point coverage. Sixteen of the eighteen coefficients are fixed (at the optimal), the remaining translation components of the 3rd mapping correspond to the X and Y axis (see Table 5.2). (Note that due to interpolation of the data some detail has been smoothed out.)

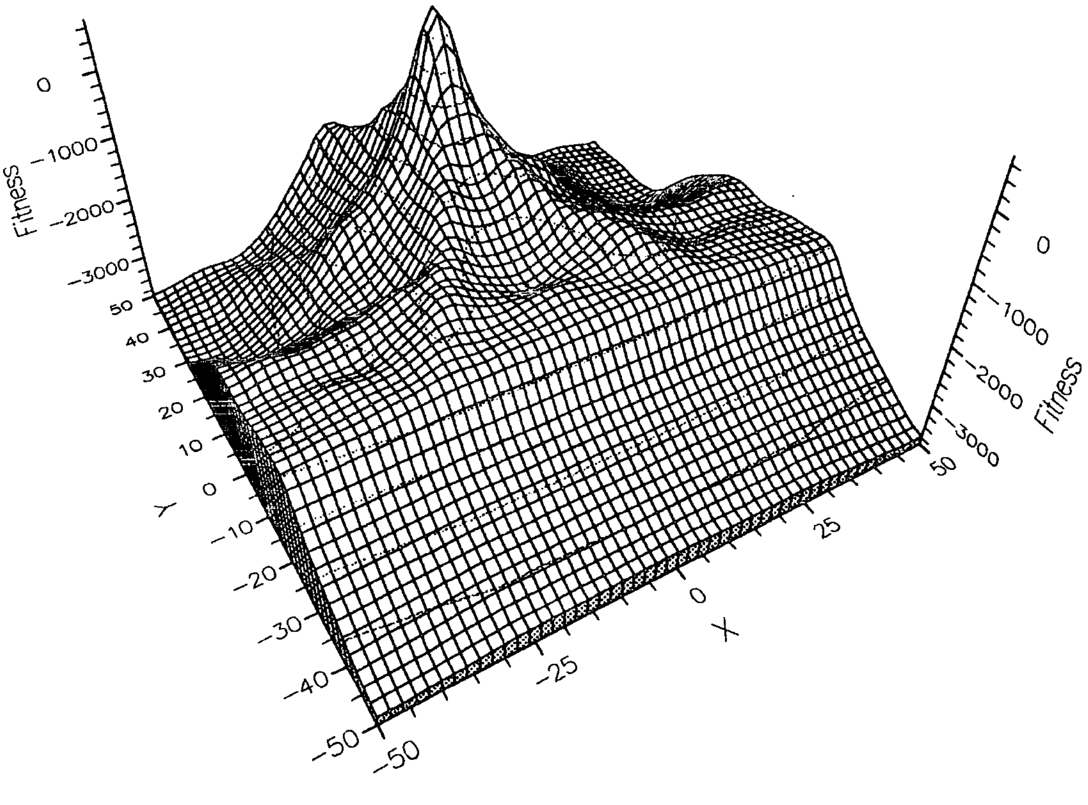
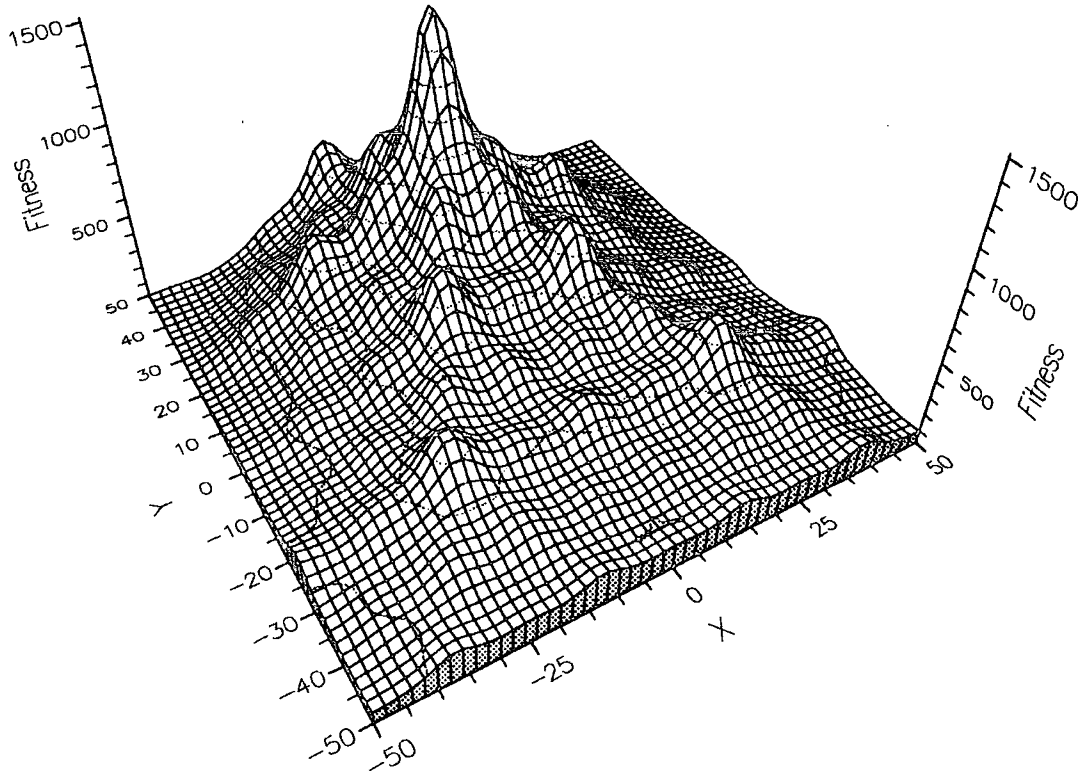


Figure 5.3: Cross section of the search space for the Sierpinski triangle with the fitness function of collage and point coverage. Sixteen of the eighteen coefficients are fixed (at the optimal), the remaining translation components of the 3rd mapping correspond to the X and Y axis (see Table 5.2). (Note that due to interpolation of the data some detail has been smoothed out.)



5.2 Outline of a GA

In applying a GA to the inverse problem the subsymbolic representation adopted is that of a binary string (Holland 1992, p.71; Goldberg 1989, p. 80). The coefficients of the contraction mappings are converted to binary strings and are concatenated together to form one string. Each coefficient is represented by a binary string of length eight, thus each of the coefficients can take one of $2^8 = 256$ values. The size of the search space for an IFS consisting of n mappings is therefore $2^{8 \times 6 \times n}$ — for $n = 3$ there are $\approx 2.23 \times 10^{43}$ possibilities.

When selecting solutions for mating, a 'roulette wheel' type of sampling (Goldberg 1989, p. 11) is used in order to ensure that better solutions are more likely to be chosen. This proceeds by first evaluating the fitness of each solution in a generation (using either the attractor or collage and point coverage). These values are then rescaled linearly in the range 10 to 100, such that the solution with the worst fitness has the value 10 and the best 100. The rescaling is necessary to remove negative fitness values, and the range was chosen to help maintain solution diversity. Sections of the roulette wheel are then allocated according to this scaled value. This ensures that when the roulette wheel is probabilistically spun, the fitter the solution the more likely it is to be selected.

Parents are combined using a two-point crossover operator (see Figure 5.4) with the probability of crossover $p_c = 0.6$. When applied to a point in the binary string, the mutation operator changes the value at that point, *i.e.*, 1 to 0, or 0 to 1. In order not to be too disruptive, the probability of mutation was kept low with $p_m = 0.0005$.

Figure 5.4: An example of how two-point crossover combines the binary strings of two parents to give two children. The two endpoints of the section of binary string that is exchanged are chosen uniformly at random.

parent 1	1	1	0	1	0	1	0	0	1
parent 2	1	0	1	1	0	0	1	0	1
section exchanged			*	*	*	*			
child 1	1	1	1	1	0	0	0	0	1
child 2	1	0	0	1	0	1	1	0	1

The GA used is summarised in Figure 5.5 and is often known as the Simple Genetic Algorithm. Many variations on this algorithm have been suggested (see, *e.g.*, Goldberg 1989; Beasley *et al.* 1993a, 1993b).

Figure 5.5: Outline of the genetic algorithm used in the experiments of this chapter and those of Chapters 7 and 8.

1. Randomly initialise a parent population of binary strings.
2. Evaluate each member of the parent population.
3. Select a solution from the parent population with probability in proportion to fitness using a roulette wheel approach.
4. Apply the crossover operator with a probability p_c. If crossover is not performed then place the solution into the child generation. Otherwise:
   (a) Select a solution from the parent population with uniform probability.
   (b) Select at random two crossover points that are within the binary string.
   (c) Using two-point crossover recombine the solutions (splicing the respective sections from each string into each other) and place them both into the child generation.
5. If the number of solutions to be allowed in the child generation has not been reached then go to step 3.
6. With a probability p_m, mutate each element of the child solutions.
7. Replace the parent population with the child population. This completes a generation.
8. If the termination criterion is not met then go to step 2.

5.3 Outline of EP

In applying EP to the inverse problem the subsymbolic representation adopted is that which is most 'natural.' The coefficients are, therefore, stored as real numbers (six decimal places), and are constrained to the ranges given by Equations 5.2. A child is produced from a parent by mutating the coefficients of the IFS (for all i) according to:

$$\begin{array}{ll} a_i = a_i + \sigma_1 N(0,1) & b_i = b_i + \sigma_1 N(0,1) \\ c_i = c_i + \sigma_1 N(0,1) & d_i = d_i + \sigma_1 N(0,1) \\ e_i = e_i + \sigma_2 N(0,1) & f_i = f_i + \sigma_2 N(0,1) \end{array}$$

where σ_1 and σ_2 are standard deviations derived from the fitness of the parent (see Section 5.5) and $N(0,1)$ is a standard normal random variable. Relating the severity of the mutation to the fitness of the solution ensures that fitter parents are less likely to be mutated to the same degree as less fit parents. If as the result of mutation the constraints of Equations 5.2 are not satisfied, the values that are out of the feasible range are set to the nearest allowable values.

The EP algorithm used is summarised in Figure 5.6. There are many variations of the above algorithm including, for example, meta-EP (Fogel 1992a).

Figure 5.6: Outline of the evolutionary programming algorithm used in the experiments of this chapter and those of Chapters 7 and 8.

1. Randomly initialise a parent population of solutions.
2. Evaluate each member of the parent population.
3. Mutate each member of the parent population, by an amount related to its fitness, to generate a member of the child population.
4. Evaluate each member of the child population.
5. For each member of the child and parent populations:
   (a) Select at random a number, TOURN, of solutions from the parent and child populations.
   (b) Count the number of these solutions whose fitness is less than or equal to that of the current selected solution. This number is the score for the selected solution.
6. Rank the scores of the solutions.
7. Select the solutions which rank in the top half of the list and replace the parent population with these solutions. This completes a generation.
8. If the termination criterion is not met then go to step 3.

5.4 Hill Climbing

The performances of the EAs are to be compared to those of several hill-climbing algorithms. Three commonly used hill-climbing schemes are considered (Forrest and Mitchell 1992): steepest-ascent hill-climbing, next-ascent hill-climbing and random-mutation hill-climbing. These operate on solutions which are encoded subsymbolically as binary strings, and are implemented as follows:

Steepest-ascent hill-climbing (SAHC)

1. Choose a string at random. Call the string *current-best.*

2. Systematically mutate each bit in the string from left to right, recording the fitnesses of the resulting strings.

3. If any of the resulting strings give a fitness increase, then set *current-best* to the resulting string giving the highest fitness increase.

4. If there is no fitness increase, then return the value of *current-best.* Otherwise go to step 2.

Next-ascent hill-climbing (NAHC)

1. Choose a string at random. Call the string *current-best.*

2. Mutate single bits in the string from left to right, recording the fitnesses of the resulting strings. If any increase in fitness is found, then set *current-best* to that increased-fitness string. Go to step 2 with the new *current-best*, but continue mutating the new string starting after the bit position at which the previous increase was found.

3. If there is no fitness increase then return the value of *current-best.* Otherwise go to step 2.

Random-mutation hill-climbing (RMHC)

1. Choose a string at random. Call the string *current-best*.
2. Choose a position at random to mutate. If the mutation leads to an equal or higher fitness then set *current-best* to the resulting string.
3. If the set number of function evaluations have been performed return the value of *current-best*. Otherwise go to step 2.

5.5 Results

This section presents the results of applying a GA, EP and three hill-climbing algorithms to several inverse problems. Three target shapes were used: a solid triangle, a Sierpinski triangle, and a Dragon fractal. IFSs for generating these shapes are given in Table 5.3 and the shapes themselves are shown in Figure 5.7. In the case of the triangle and Sierpinski triangle the number of mappings used was set at three, and for the Dragon fractal two mappings were used (since there are known theoretical solutions for these values).

Table 5.3: The IFSs used to generate the target shapes which are shown in Figure 5.7. The triangle is generated with four mappings rather than the obvious three, because the MPP introduced some small errors when plotting the attractor of the three mapping IFS which was initially considered.

Triangle

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.5	0	0	0.5	-25	-25
2	0.5	0	0	0.5	25	-25
3	0.5	0	0	-0.5	0	-25
4	0.5	0	0	0.5	0	25

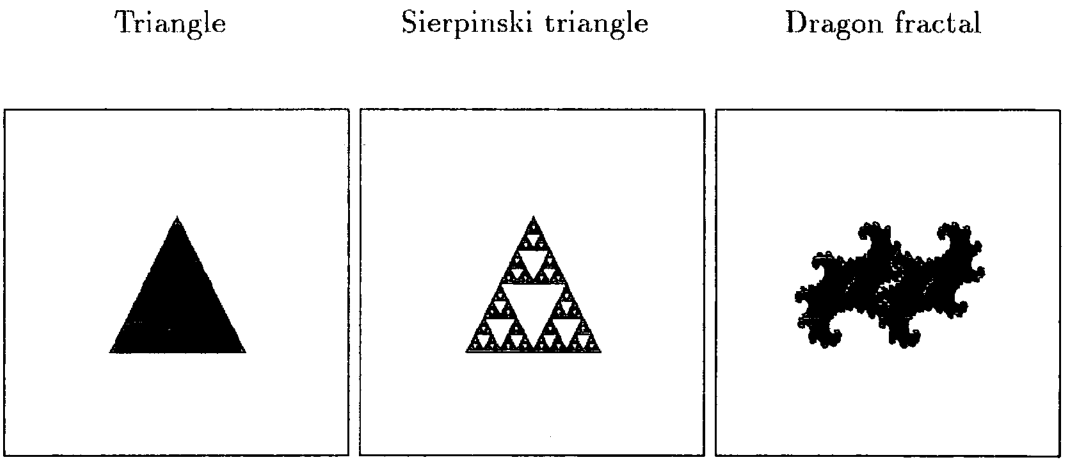
Sierpinski Triangle

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.5	0	0	0.5	-25	-25
2	0.5	0	0	0.5	25	-25
3	0.5	0	0	0.5	0	25

Dragon fractal

i	a_i	b_i	c_i	d_i	e_i	f_i
1	0.59	-0.37	0.37	0.59	30	0
2	0.5	0	0	0.5	-30	0

Figure 5.7: The attractors of the IFSs given in Table 5.3. These are the target shapes used in the experiments of this chapter.



Triangle

Sierpinski triangle

Dragon fractal

The GA and EP used are those given in Sections 5.2 and 5.3 respectively. For both the GA and EP a population of 100 parents was used, and they were executed over 100 generations. For EP, a tournament size of five was used, and the following formulae were used to set the standard deviations:

$$val = (PIX - fit)/PIX$$

$$\sigma_1 = \sqrt{val/500} \qquad\qquad \sigma_2 = \sqrt{val} \times 5$$

where PIX is the number of pixels of the target shape and fit is the fitness of an IFS. A cut-off for σ_1 of 0.1 and for σ_2 of 5 was used to prevent large standard deviations. These cut-off values allow for large amounts of mutation of shapes that have extremely poor fitness, while limiting the chance of out of range values occurring as a result of the mutation.

For each target shape and each fitness function, 31 trials of the GA, EP and the hill-climbing algorithms were carried out. Each pair of trials of the GA and EP had the same randomly generated initial population (*i.e.*, 31 different initial populations were used) and each of hill-climbing algorithms started with the same randomly generated binary string (*i.e.*, 31 different starting strings were used). The fitness of the best solution found in each of the runs is shown in Tables 5.4 — 5.9. The mean and standard deviation of each set of results is also given. In the case of the GA and EP the generation at which the best solution was discovered is shown in parenthesis.

The results for the median trial of the GA and EP are shown in Figures 5.8, 5.9, 5.14, 5.15, 5.20 and 5.21 (attractor and point coverage) and Figures 5.11, 5.12, 5.17, 5.18, 5.23 and 5.24 (collage and point coverage). In Figures 5.8, 5.11, 5.14, 5.17, 5.20 and 5.23, a sequence of the best attractors from various generations is shown. The Figures 5.9, 5.12, 5.15, 5.18, 5.21 and 5.24 each show the online and offline performance of the median run of the GA and EP. The offline performance is the average fitness of all of the IFSs in a particular generation, while the online performance is the average fitness of all IFSs that have been generated up to a certain generation. Figures 5.10, 5.13, 5.16, 5.19, 5.22 and 5.25 show the best solutions found for each target shape when using the GA, EP and the hill-climbing algorithms for each of the fitness functions used.

Table 5.10 contains the values of the test statistics which were used in comparing the performance of the algorithms, for all combinations of the target shapes and fitness functions considered. These values were obtained by carrying out a hypothesis test for two population means with the null hypothesis that the means are equal. The samples were assumed to be independent and normally distributed. The variances of the two samples were not assumed to be equal and so a Smith-Satterthwaite modified one tailed t-test was used (Weiss and Hassett 1991, p. 504) — this is used in all further pairwise comparisons in this thesis. The number of degrees of freedom are the values in Table 5.10 which are in parenthesis.

The results of the trials conducted showed that EP outperformed the GA, and all of the hill-climbing algorithms, for each of the target shapes considered with both fitness functions. In all of these cases the results were statistically significant ($P \ll 0.001$).

A comparison of the performance of the GA and the hill-climbing algorithms with the attractor and point coverage as the fitness function showed mixed results. For the Dragon fractal the GA outperformed all of the hill-climbing algorithms, but with the Sierpinski Triangle all of the hill-climbers outperformed the GA. In each of these cases the observed difference was statistically significant ($P < 0.05$). Results for the triangle were varied; the GA outperformed NAHC ($P < 0.05$) and RMHC ($P < 0.1$), but SAHC outperformed the GA ($P < 0.1$).

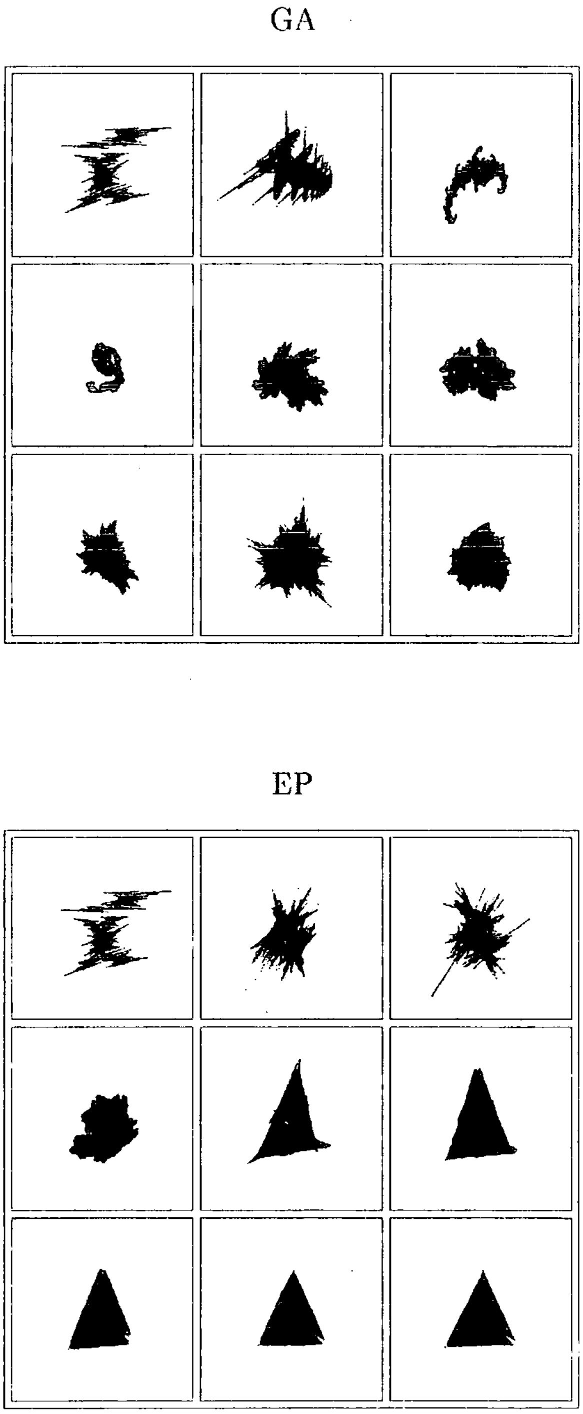
With the collage and point coverage as the fitness function mixed results were again achieved. For the Dragon fractal the GA outperformed all of the hill-climbing algorithms, although only the comparison with SAHC was statistically significant ($P < 0.05$). With the Sierpinski Triangle two of the three hill-climbers outperformed the GA, but with the triangle all of the hill-climbers performed better (only the comparison with the SAHC was statistically significant).

The observed differences in the performance of the hill-climbing algorithms were not statistically significant ($P > 0.05$), except in the case of the triangle with the attractor and point coverage as the fitness function. In this case SAHC significantly outperformed both NAHC ($P \ll 0.01$) and RMHC ($P < 0.05$).

Table 5.4: The best solutions found by each of the search algorithms with a triangle as the target shape. The attractor and point coverage was used as the fitness function. Each algorithm was run 31 times and in the case of the GA and EP the generation at which the best solution was found is shown in parenthesis. The fitness of the best solution found by each algorithm is shown in bold.

Algorithm	EP		GA		SAHC	NAHC	RMHC
Fitness of best solution found (Optimum = 5101)	4574	(89)	3190	(96)	3570	3204	1602
	4675	(78)	2480	(97)	2732	2077	2489
	4584	(87)	3524	(77)	1781	2801	2606
	4045	(99)	2625	(98)	3375	2508	3862
	4751	(87)	3237	(75)	2564	2514	3503
	4723	(91)	1991	(82)	2816	1324	2603
	4787	(98)	2260	(96)	2746	2831	3152
	4546	(95)	2245	(97)	1506	848	319
	4250	(91)	3042	(69)	3823	1388	3433
	4646	(94)	2876	(65)	3192	2859	2754
	4068	(93)	**3633**	(62)	3036	2476	2077
	4544	(100)	2928	(84)	3110	2788	3101
	4273	(80)	2792	(99)	1803	436	2794
	4660	(96)	2412	(22)	2806	2237	2696
	4564	(98)	2760	(100)	956	-439	166
	4062	(100)	2037	(88)	3080	1963	2185
	3709	(100)	2819	(82)	2854	2614	2797
	4735	(96)	2308	(47)	**4155**	2338	**3868**
	4271	(99)	3320	(80)	3425	2698	2318
	4793	(82)	2801	(93)	2782	2068	1724
	3855	(92)	2678	(100)	3382	**3307**	2473
	4214	(95)	2551	(94)	3037	2714	2768
	3789	(99)	2349	(100)	4112	1726	1972
	4182	(100)	3564	(79)	2507	2446	2313
	4808	(95)	1697	(83)	2558	2489	3400
	4706	(98)	2916	(81)	2957	1590	2800
	4612	(98)	2202	(48)	3355	3122	2466
	4608	(95)	2951	(95)	3063	2157	3500
	4427	(100)	2532	(85)	2760	870	-14
	3982	(92)	2069	(96)	3217	1130	1864
	4747	(98)	3046	(100)	3142	1214	1268
Mean (to 1 d.p.)	4425.5		2704.4		2909.7	2074.1	2414.8
SD (to 2 d.p.)	328.66		487.64		691.28	878.51	981.80

Figure 5.8: A sequence of attractors from the median trial of the GA and EP with a triangle as the target shape. The attractor and point coverage was used as the fitness function. The attractors shown are those of the best solutions in generations 0, 5, 10, 20, 30, 40, 60, 80 and 100 (top left to bottom right).



GA

EP

Figure 5.9: Online and offline performance for the median trial of the GA and EP with a triangle as the target shape. The attractor and point coverage was used as the fitness function.

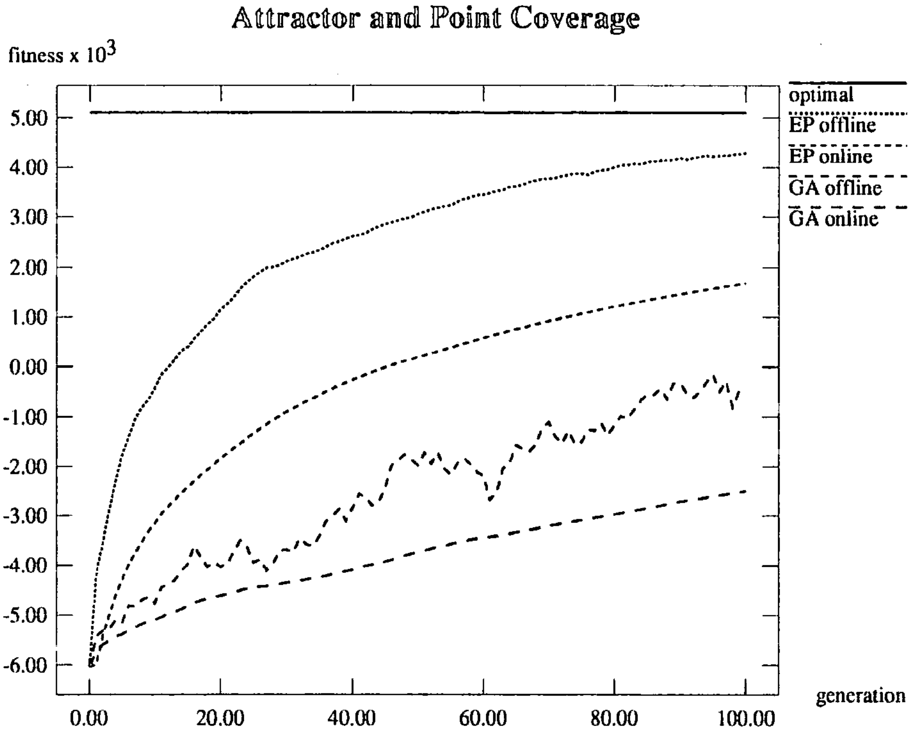


Figure 5.10: Attractors of the best IFSs found when using each of the search algorithms with a triangle as the target shape. The attractor and point coverage was used as the fitness function.

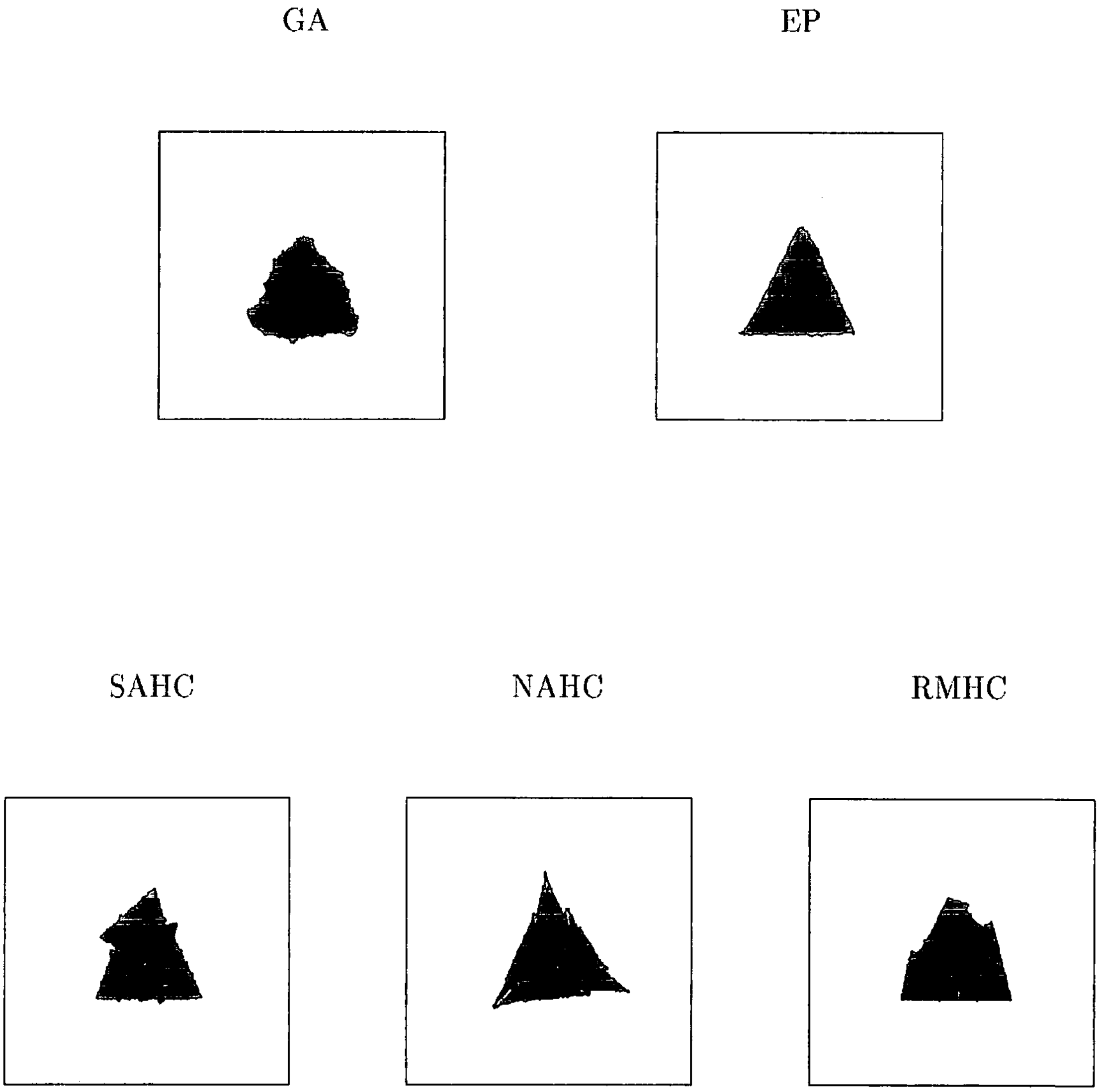


Table 5.5: The best solutions found by each of the search algorithms with a triangle as the target shape. The collage and point coverage was used as the fitness function. Each algorithm was run 31 times and in the case of the GA and EP the generation at which the best solution was found is shown in parenthesis. The fitness of the best solution found by each algorithm is shown in bold.

Algorithm	EP		GA		SAHC	NAHC	RMHC
Fitness of best solution found (Optimum = 5101)	**5010**	(100)	2982	(98)	3767	3704	3020
	4786	(100)	3331	(81)	3923	4157	4503
	4967	(95)	3518	(89)	3229	2621	4497
	4594	(100)	3355	(66)	4456	4213	4166
	4910	(98)	3802	(94)	3931	4241	4515
	4842	(98)	3611	(55)	**4687**	3834	2808
	4864	(99)	2993	(97)	4084	3830	3743
	4854	(92)	3014	(56)	4537	3941	3542
	4377	(96)	3798	(81)	4545	3422	3264
	4890	(99)	3701	(38)	3900	2538	2463
	4858	(91)	2766	(98)	1780	2926	3965
	4742	(85)	3889	(98)	3476	3456	3546
	4848	(100)	3497	(46)	4444	4418	3531
	4840	(94)	3329	(99)	3956	3115	3385
	4872	(98)	3212	(98)	3991	4097	4081
	4587	(94)	2990	(93)	3966	4199	3384
	4806	(99)	3709	(82)	3803	3666	3377
	4861	(98)	3597	(99)	4636	**4606**	3887
	4947	(83)	3535	(100)	3551	3688	2706
	4960	(87)	3342	(97)	4188	3747	3486
	4626	(100)	3282	(93)	3873	2972	4404
	4641	(95)	3579	(96)	2887	3677	3111
	4290	(95)	**4254**	(82)	2337	2264	2061
	4722	(99)	2691	(95)	2715	2212	3403
	4874	(87)	3263	(64)	3851	2926	**4609**
	4756	(83)	3623	(91)	4159	3936	3807
	4620	(85)	3351	(91)	4109	4555	2205
	4830	(99)	3762	(96)	4117	4192	4103
	4601	(94)	3382	(90)	3150	1137	3210
	4747	(100)	3554	(83)	3227	4118	3731
	4735	(96)	4014	(98)	4031	4357	4390
Mean (to 1 d.p.)	4769.6		3442.8		3784.1	3573.1	3577.5
SD (to 2 d.p.)	164.83		356.10		674.43	802.23	680.95

Figure 5.11: A sequence of attractors from the median trial of the GA and EP with a triangle as the target shape. The collage and point coverage was used as the fitness function. The attractors shown are those of the best solutions in generations 0, 5, 10, 20, 30, 40, 60, 80 and 100 (top left to bottom right).

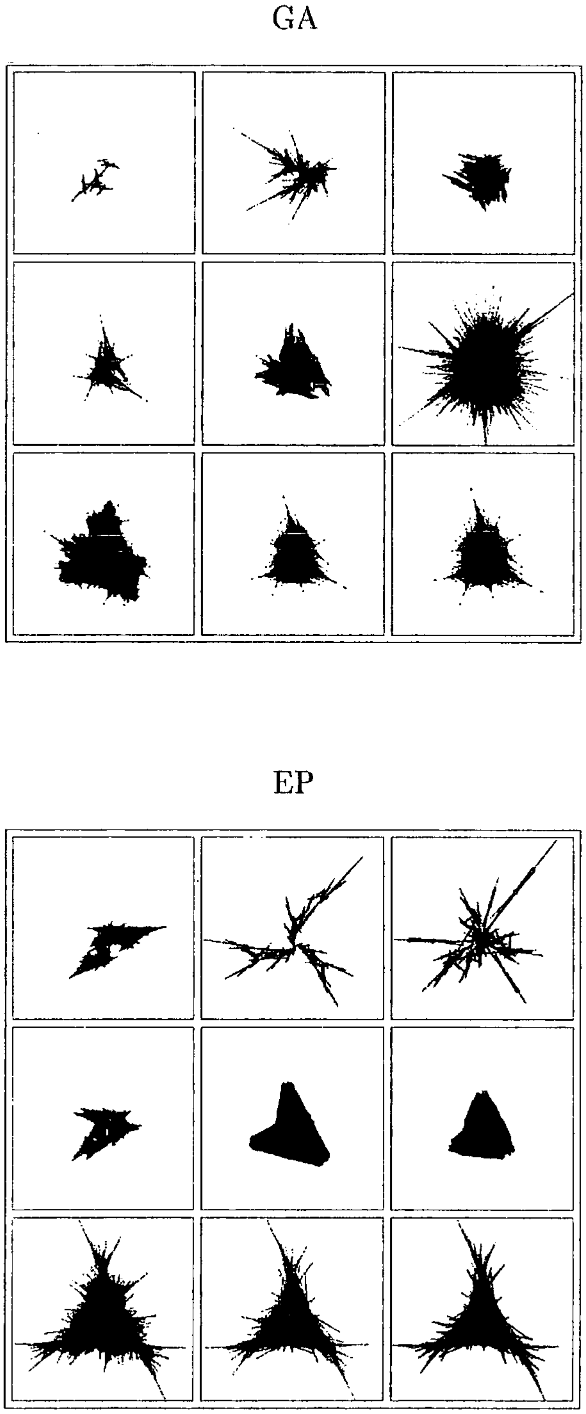


Figure 5.12: Online and offline performance for the median trial of the GA and EP with a triangle as the target shape. The collage and point coverage was used as the fitness function.

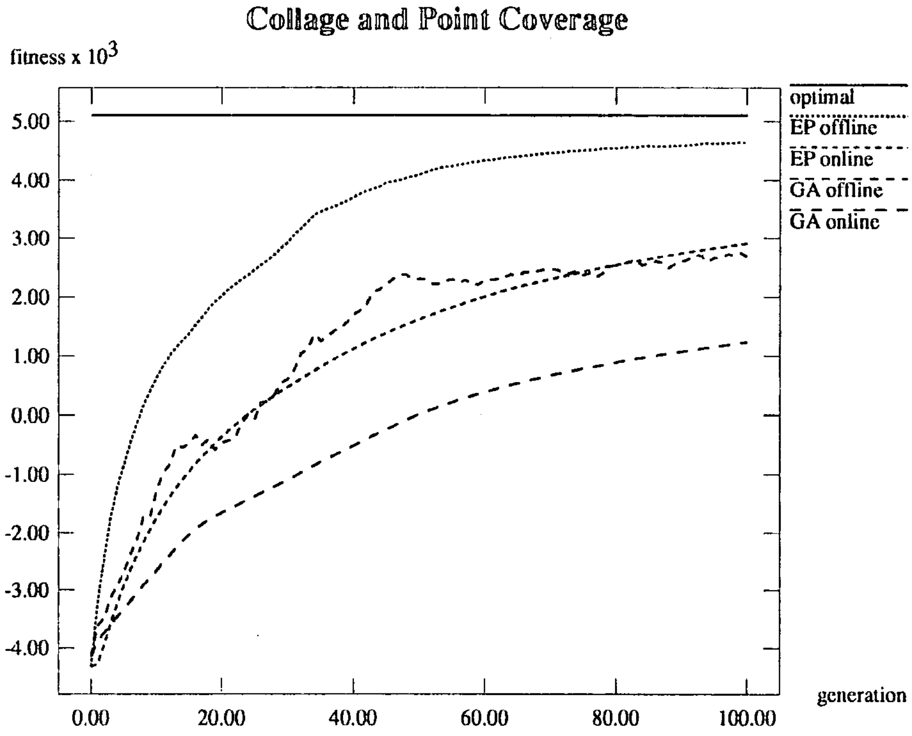


Figure 5.13: Attractors of the best IFSs found when using each of the search algorithms with a triangle as the target shape. The collage and point coverage was used as the fitness function.

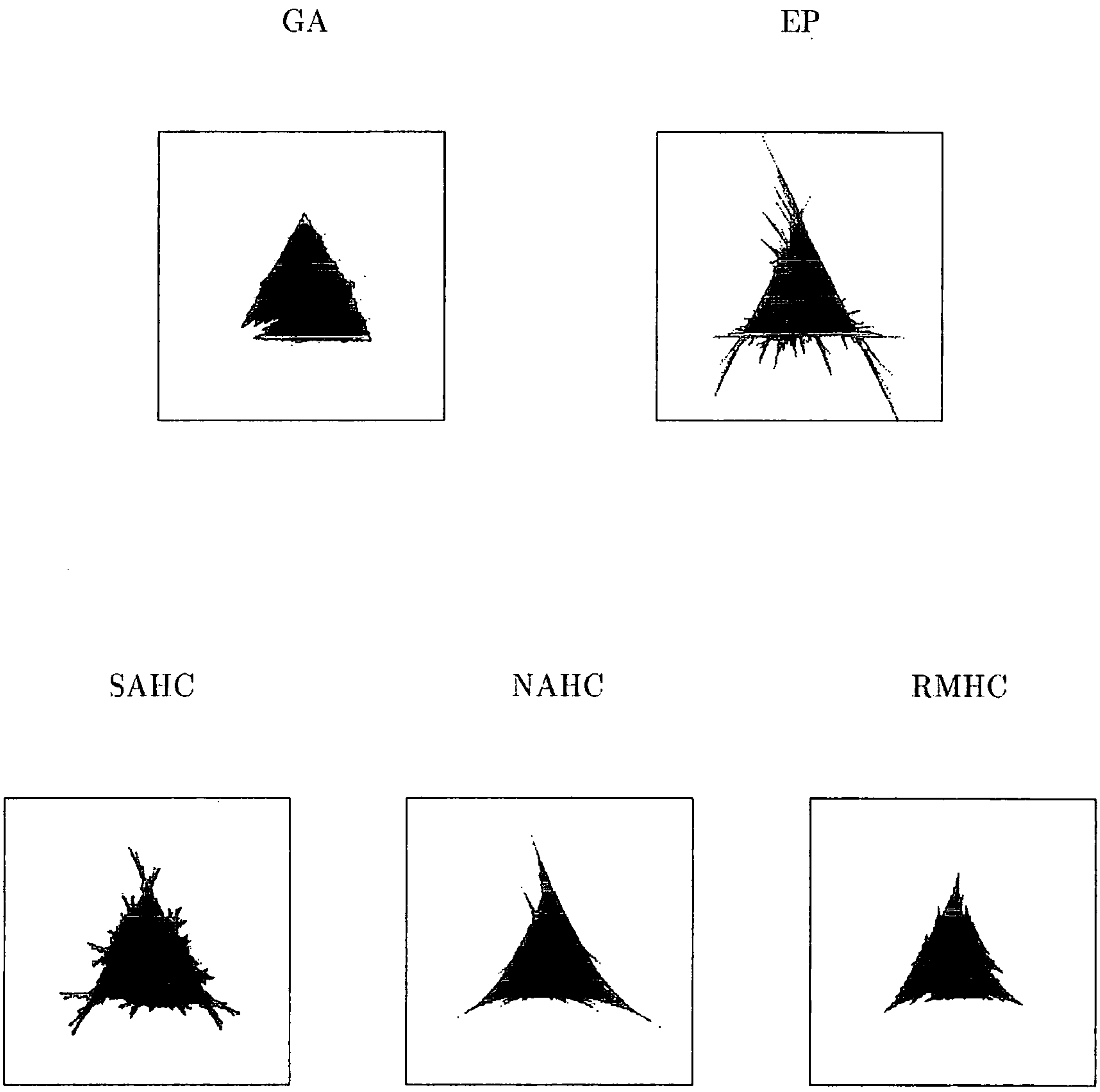


Table 5.6: The best solutions found by each of the search algorithms with a Sierpinski triangle as the target shape. The attractor and point coverage was used as the fitness function. Each algorithm was run 31 times and in the case of the GA and EP the generation at which the best solution was found is shown in parenthesis. The fitness of the best solution found by each algorithm is shown in bold.

Algorithm	EP		GA		SAHC	NAHC	RMHC
Fitness of best solution found (Optimum = 2411)	-1285	(98)	**-1734**	(98)	-2028	-2259	-1958
	-1383	(95)	-2058	(43)	-1621	-1755	-1876
	-1427	(98)	-2032	(99)	-1953	-1627	-2059
	-1619	(94)	-1980	(33)	-2149	**-1198**	-2161
	-1206	(97)	-1960	(86)	-2219	-2079	-1925
	-1320	(85)	-2097	(20)	-1646	-2044	-1764
	-1069	(87)	-2087	(100)	-1939	-1917	-1838
	-1294	(76)	-2110	(19)	-2124	-1933	-1970
	-1324	(78)	-2062	(56)	-2162	-2255	-2247
	-1653	(96)	-2003	(75)	-1511	-1882	-1739
	-1292	(99)	-2036	(16)	-2074	-2037	-1981
	-790	(99)	-2078	(25)	-1941	-1795	-1561
	-1415	(94)	-2058	(82)	-2151	-1762	-1947
	-1164	(83)	-1841	(100)	-2048	-1842	-2162
	-1232	(86)	-1991	(74)	-2298	-1685	-1761
	-1159	(72)	-2147	(100)	-1553	-2106	-2273
	-1436	(98)	-2188	(6)	-1971	-1905	-1659
	-1400	(90)	-2063	(94)	-1732	-1783	-1617
	-1280	(95)	-2020	(91)	-1819	-1823	-1805
	-1462	(97)	-2049	(96)	-2015	-2250	-1562
	-997	(74)	-2185	(1)	-1938	-2424	-2013
	-1615	(95)	-2047	(96)	-2264	-1804	-2247
	-1005	(96)	-2091	(96)	-1244	-2118	-2018
	-1673	(50)	-2084	(83)	-1932	-2364	-1758
	-1178	(72)	-2089	(99)	-2000	-1656	-1797
	-1381	(41)	-2132	(5)	**-1240**	-1602	-1819
	-1660	(93)	-2128	(32)	-1727	-2129	-2232
	-1588	(87)	-1768	(100)	-1526	-2015	-1664
	-1414	(97)	-2179	(100)	-1778	-2220	-1886
	-1071	(75)	-2055	(98)	-1872	-1822	**-1527**
	-1422	(85)	-2053	(93)	-2027	-2322	-2051
Mean (to 1 d.p.)	-1329.5		-2045.3		-1887.2	-1948.8	-1899.3
SD (to 2 d.p.)	215.17		105.10		274.05	267.96	215.73

Figure 5.14: A sequence of attractors from the median trial of the GA and EP with a Sierpinski triangle as the target shape. The attractor and point coverage was used as the fitness function. The attractors shown are those of the best solutions in generations 0, 5, 10, 20, 30, 40, 60, 80 and 100 (top left to bottom right).

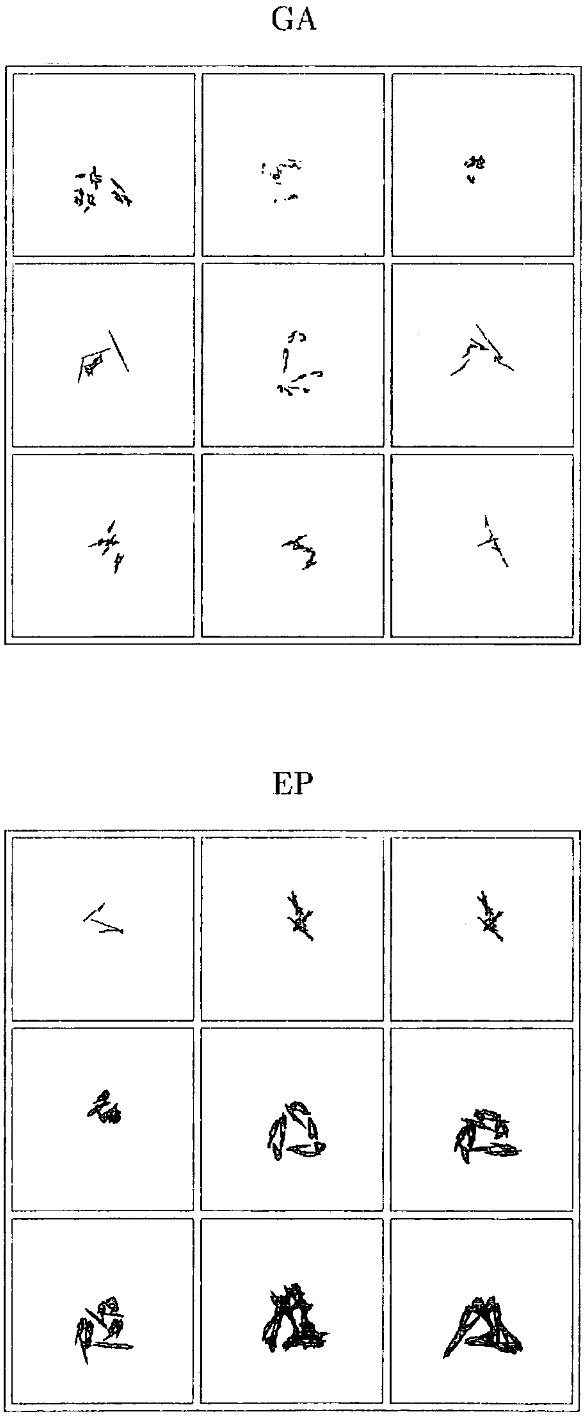


Figure 5.15: Online and offline performance for the median trial of the GA and EP with a Sierpinski triangle as the target shape. The attractor and point coverage was used as the fitness function.

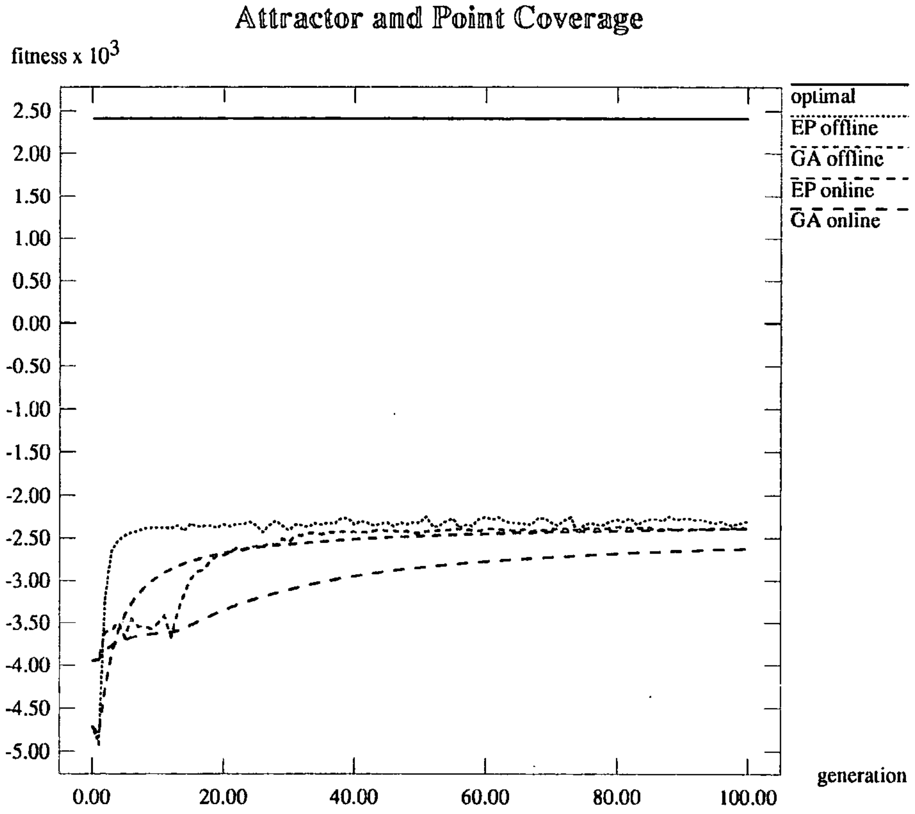


Figure 5.16: Attractors of the best IFSs found when using each of the search algorithms with a Sierpinski triangle as the target shape. The attractor and point coverage was used as the fitness function.

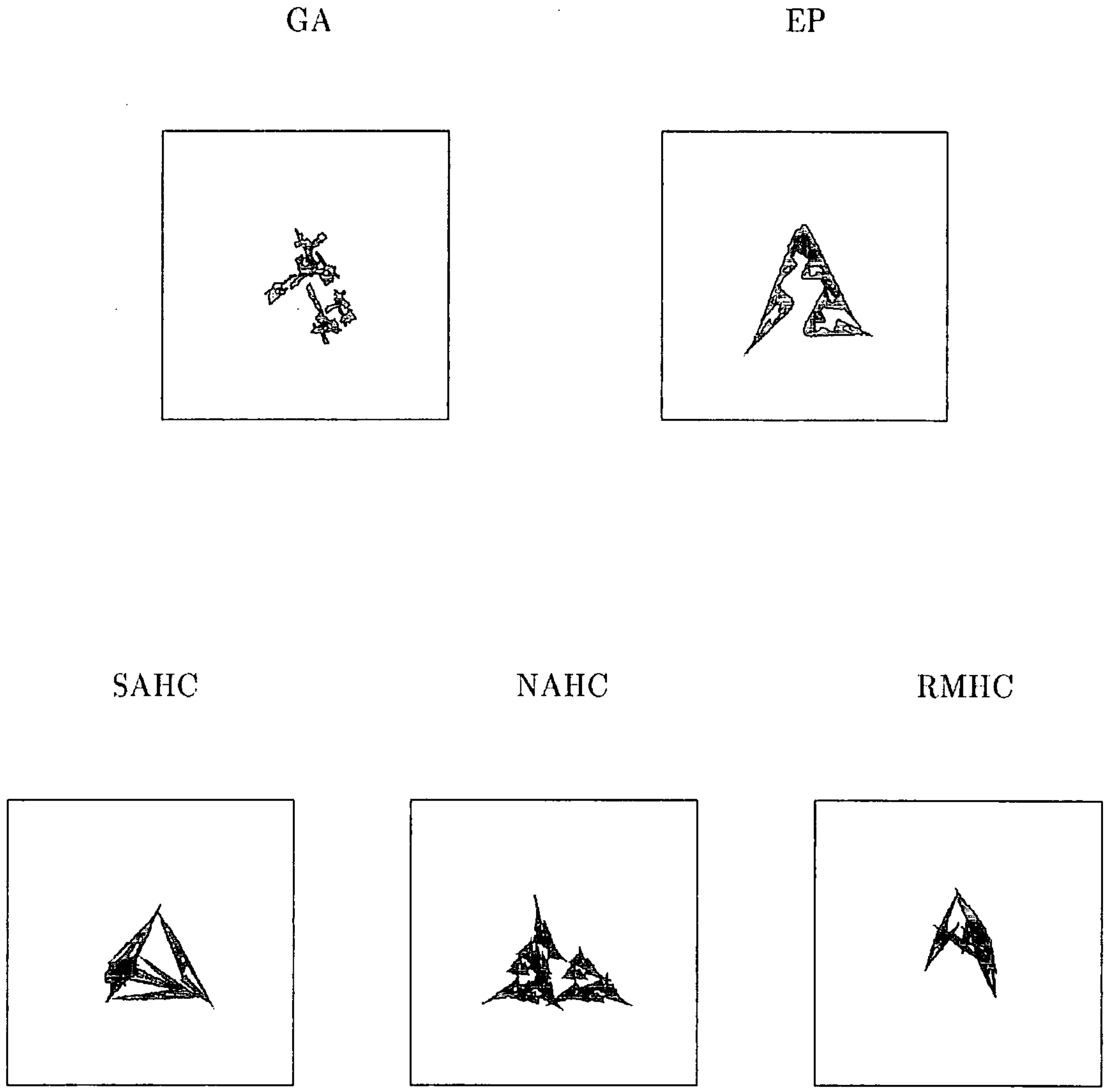


Table 5.7: The best solutions found by each of the search algorithms with a Sierpinski triangle as the target shape. The collage and point coverage was used as the fitness function. Each algorithm was run 31 times and in the case of the GA and EP the generation at which the best solution was found is shown in parenthesis. The fitness of the best solution found by each algorithm is shown in bold.

Algorithm	EP		GA		SAHC	NAHC	RMHC
Fitness of best solution found (Optimum = 2411)	-1082	(90)	-1473	(93)	-1781	-1829	-1657
	-851	(66)	-1251	(99)	-1127	-1068	-745
	-1087	(100)	-1620	(97)	-1562	-1331	-927
	-942	(39)	-1218	(99)	-970	-1156	-907
	-1187	(59)	-1502	(99)	-1431	-1555	-1587
	-1072	(94)	-1090	(85)	-1293	-1054	-1366
	-1131	(79)	-1568	(95)	-890	-1476	-1657
	-1097	(100)	-1187	(90)	-1149	-1476	-1974
	-1009	(100)	-1187	(83)	-1456	-1350	-1735
	-1364	(96)	-1294	(92)	-1321	-1677	-1664
	-840	(67)	-1426	(94)	-1658	-1571	-1921
	-907	(65)	-1525	(96)	-1783	-1127	-1715
	-1032	(86)	-1660	(46)	-1246	-1436	-1413
	-1039	(47)	-1324	(98)	-1221	-1100	-1618
	-819	(100)	-1362	(100)	-1127	-988	**-395**
	-1015	(85)	-1504	(87)	-1026	-1698	-1731
	-1137	(98)	-1659	(3)	-1440	-1804	-786
	-929	(68)	-1763	(95)	-1280	-929	-1554
	-1094	(79)	-1433	(98)	-1395	-1465	-1509
	-1074	(84)	-1403	(45)	-1400	-1839	-1591
	-1066	(60)	-1402	(77)	-1414	-996	-1446
	-1264	(49)	-1394	(90)	-1609	-1559	-1810
	-970	(90)	-1234	(82)	-1192	-1508	-1518
	-984	(50)	-1277	(100)	-856	-1488	-1670
	-1421	(87)	-1185	(96)	-1633	-1595	-1185
	-1174	(63)	**-1037**	(100)	**-659**	-1659	-507
	-1153	(84)	-1429	(81)	-1862	**-777**	-1064
	-1111	(71)	-1460	(96)	-1567	-1212	-1208
	-911	(88)	-1437	(83)	-1349	-1370	-1302
	-1182	(96)	-1324	(81)	-1369	-1369	-1336
	-880	(55)	-1336	(96)	-1222	-1525	-1231
Mean (to 1 d.p.)	-1058.8		-1385.9		-1331.9	-1386.7	-1378.4
SD (to 2 d.p.)	143.02		171.07		282.51	281.49	401.69

Figure 5.17: A sequence of attractors from the median trial of the GA and EP with a Sierpinski triangle as the target shape. The collage and point coverage was used as the fitness function. The attractors shown are those of the best solutions in generations 0, 5, 10, 20, 30, 40, 60, 80 and 100 (top left to bottom right).

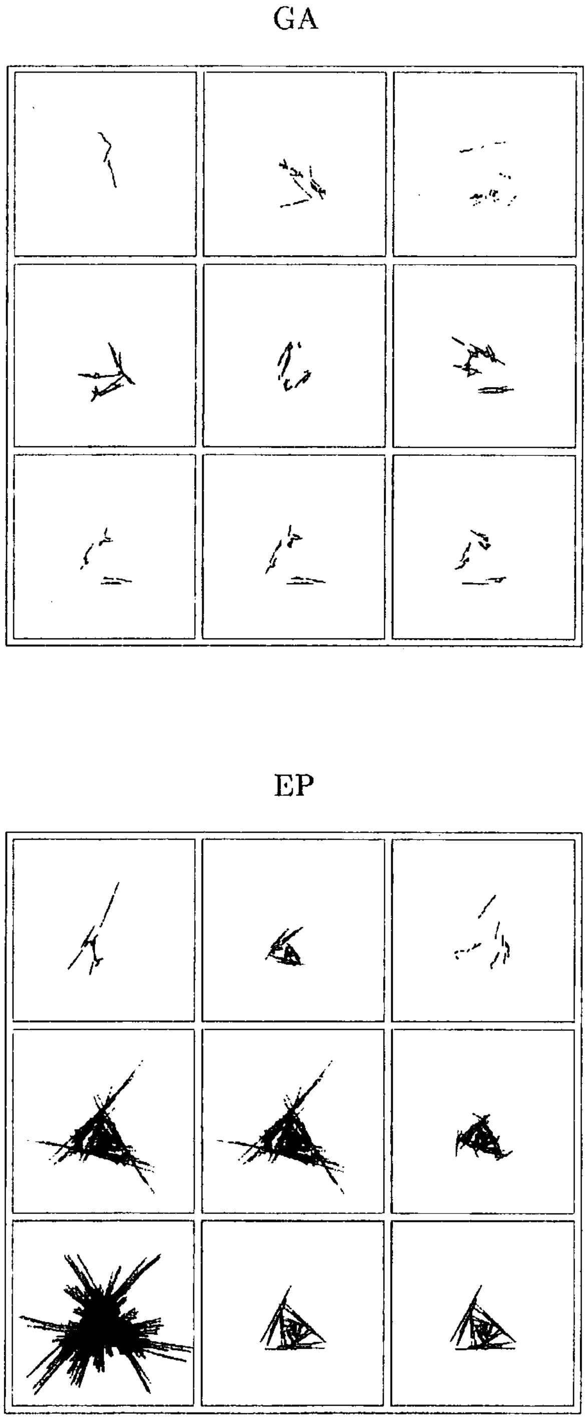


Figure 5.18: Online and offline performance for the median trial of the GA and EP with a Sierpinski triangle as the target shape. The collage and point coverage was used as the fitness function.

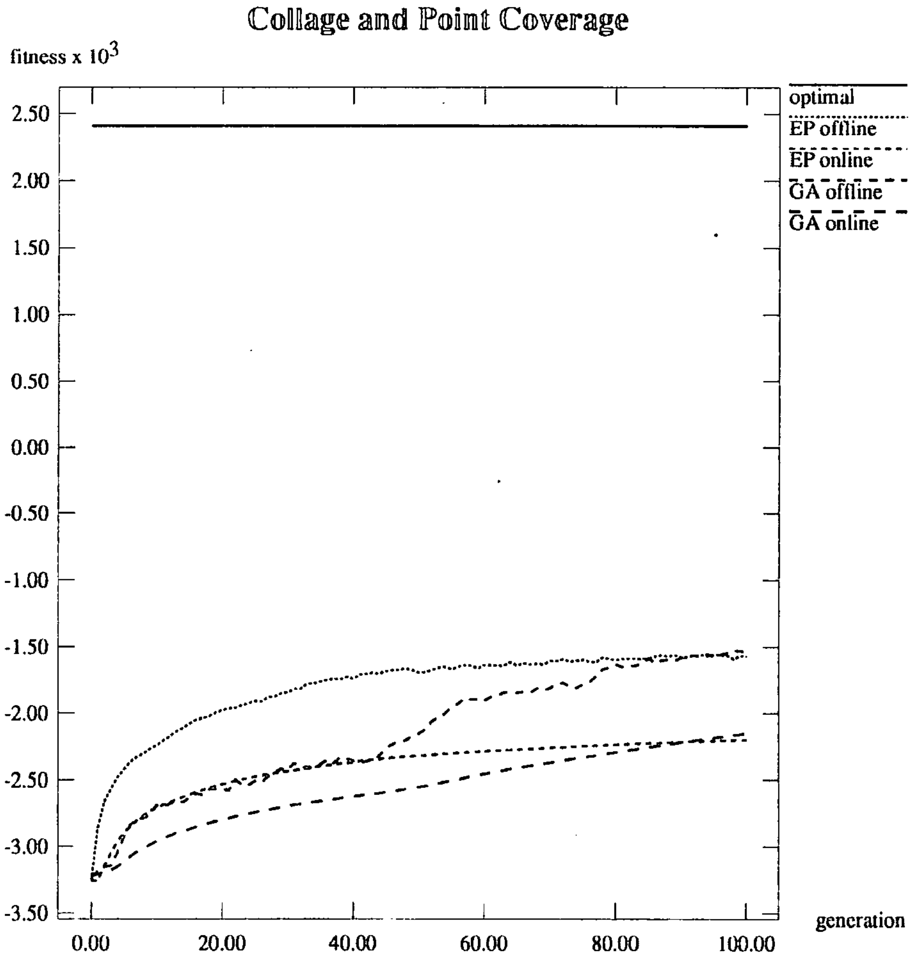


Figure 5.19: Attractors of the best IFSs found when using each of the search algorithms with a Sierpinski triangle as the target shape. The collage and point coverage was used as the fitness function.

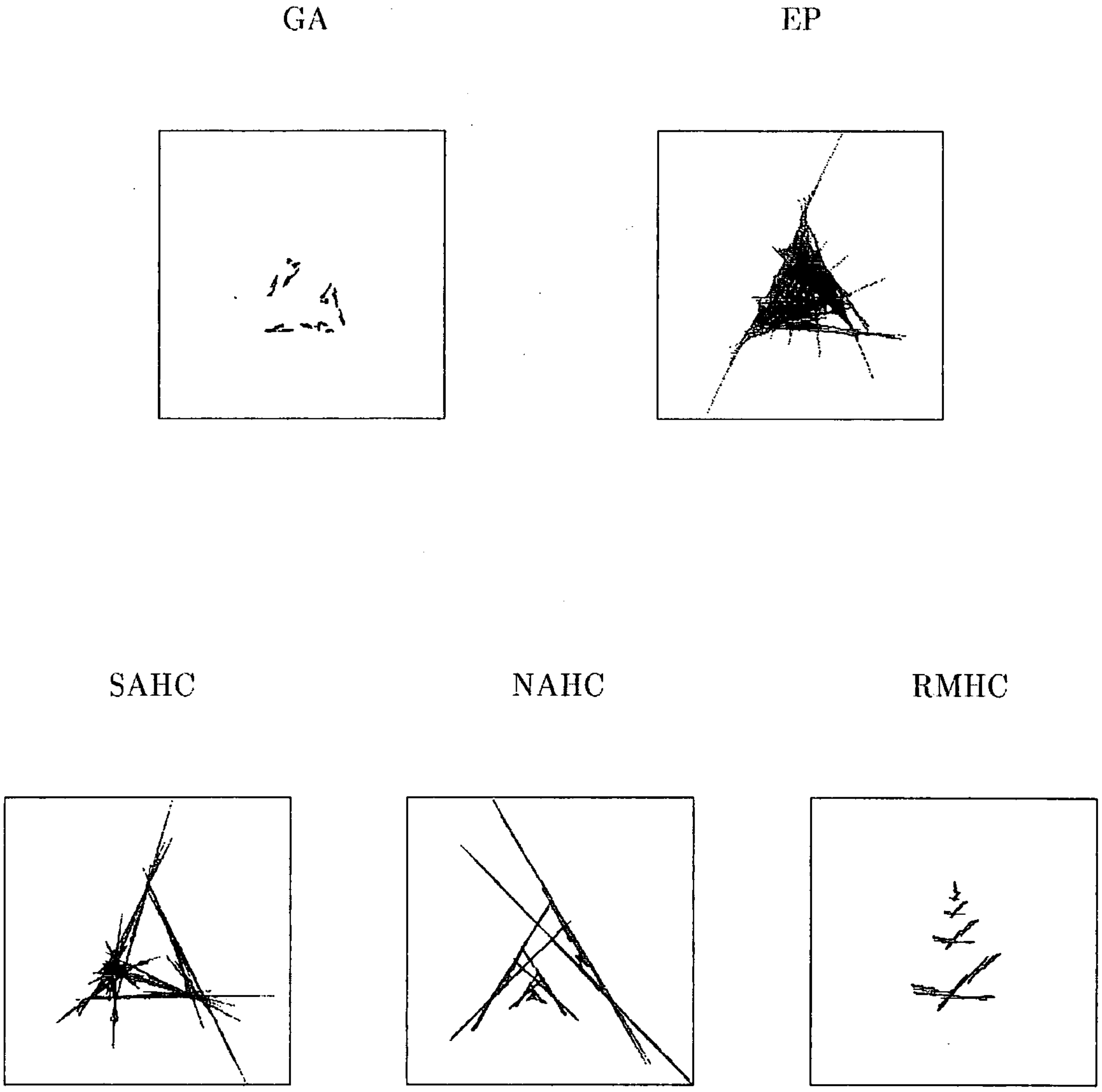


Table 5.8: The best solutions found by each of the search algorithms with a Dragon fractal as the target shape. The attractor and point coverage was used as the fitness function. Each algorithm was run 31 times and in the case of the GA and EP the generation at which the best solution was found is shown in parenthesis. The fitness of the best solution found by each algorithm is shown in bold.

Algorithm	EP		GA		SAHC	NAHC	RMHC
Fitness of best solution found (Optimum = 6726)	**4829**	(93)	1043	(49)	854	100	1350
	3485	(84)	525	(40)	-134	199	-3655
	2502	(86)	1135	(24)	-458	-124	-352
	2667	(77)	817	(73)	-621	-1640	382
	2229	(97)	702	(97)	1620	2052	-690
	2405	(67)	932	(53)	1001	1712	238
	3172	(95)	1420	(100)	-33	**2843**	906
	4567	(88)	1281	(90)	-4504	-631	-2525
	2584	(95)	1138	(74)	-825	2285	419
	4472	(100)	876	(97)	1677	1606	-1599
	2597	(95)	993	(62)	1065	1952	494
	3338	(98)	915	(61)	249	63	1504
	2656	(95)	2178	(59)	-689	506	949
	2710	(72)	1467	(57)	1210	657	**2685**
	2683	(91)	542	(39)	-444	778	276
	2492	(78)	990	(14)	-224	-1609	95
	2834	(88)	**2731**	(63)	1300	-1521	701
	2752	(90)	1137	(23)	-1036	-763	-388
	2820	(81)	-849	(40)	284	249	925
	2694	(82)	362	(68)	1298	600	-1391
	3701	(94)	1949	(67)	719	1223	913
	4707	(91)	1253	(65)	656	-660	-591
	2674	(94)	200	(15)	1706	536	-742
	2759	(53)	2636	(68)	1227	1528	-613
	3522	(90)	996	(31)	1754	795	447
	2220	(61)	592	(97)	1658	725	254
	3869	(94)	2576	(81)	37	366	2280
	4046	(91)	1019	(26)	**3256**	1781	157
	2532	(82)	218	(67)	301	338	667
	2762	(95)	1540	(97)	1731	1047	1974
	4597	(97)	-4	(95)	436	-2213	-924
Mean (to 1 d.p.)	3157.3		1074.5		486.2	476.8	133.7
SD (to 2 d.p.)	798.42		774.53		1346.63	1228.40	1320.15

Figure 5.20: A sequence of attractors from the median trial of the GA and EP with a Dragon fractal as the target shape. The attractor and point coverage was used as the fitness function. The attractors shown are those of the best solutions in generations 0, 5, 10, 20, 30, 40, 60, 80 and 100 (top left to bottom right).

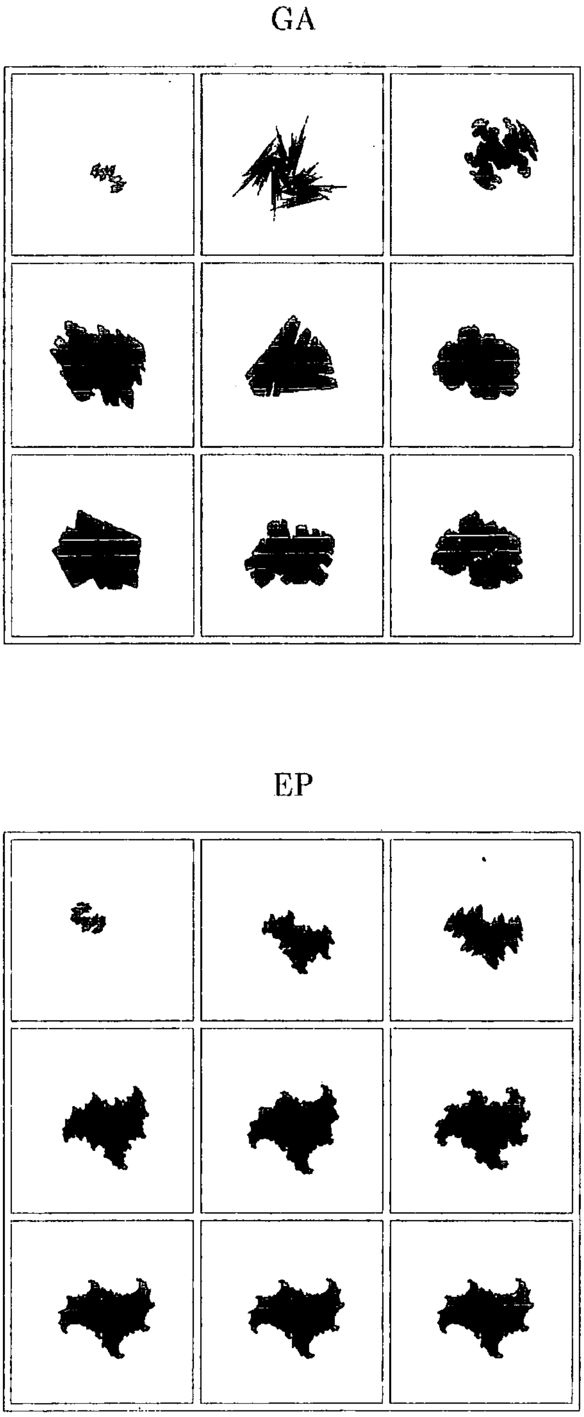


Figure 5.21: Online and offline performance for the median trial of the GA and EP with a Dragon fractal as the target shape. The attractor and point coverage was used as the fitness function.

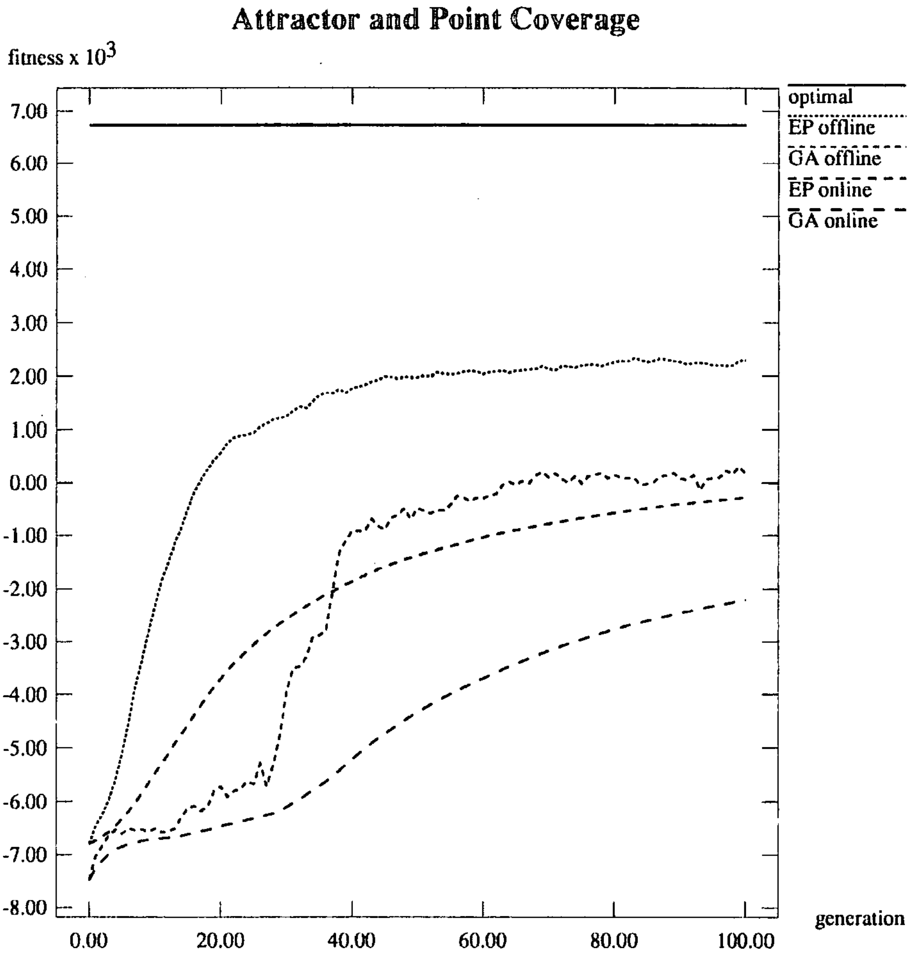


Figure 5.22: Attractors of the best IFSs found when using each of the search algorithms with a Dragon fractal as the target shape. The attractor and point coverage was used as the fitness function.

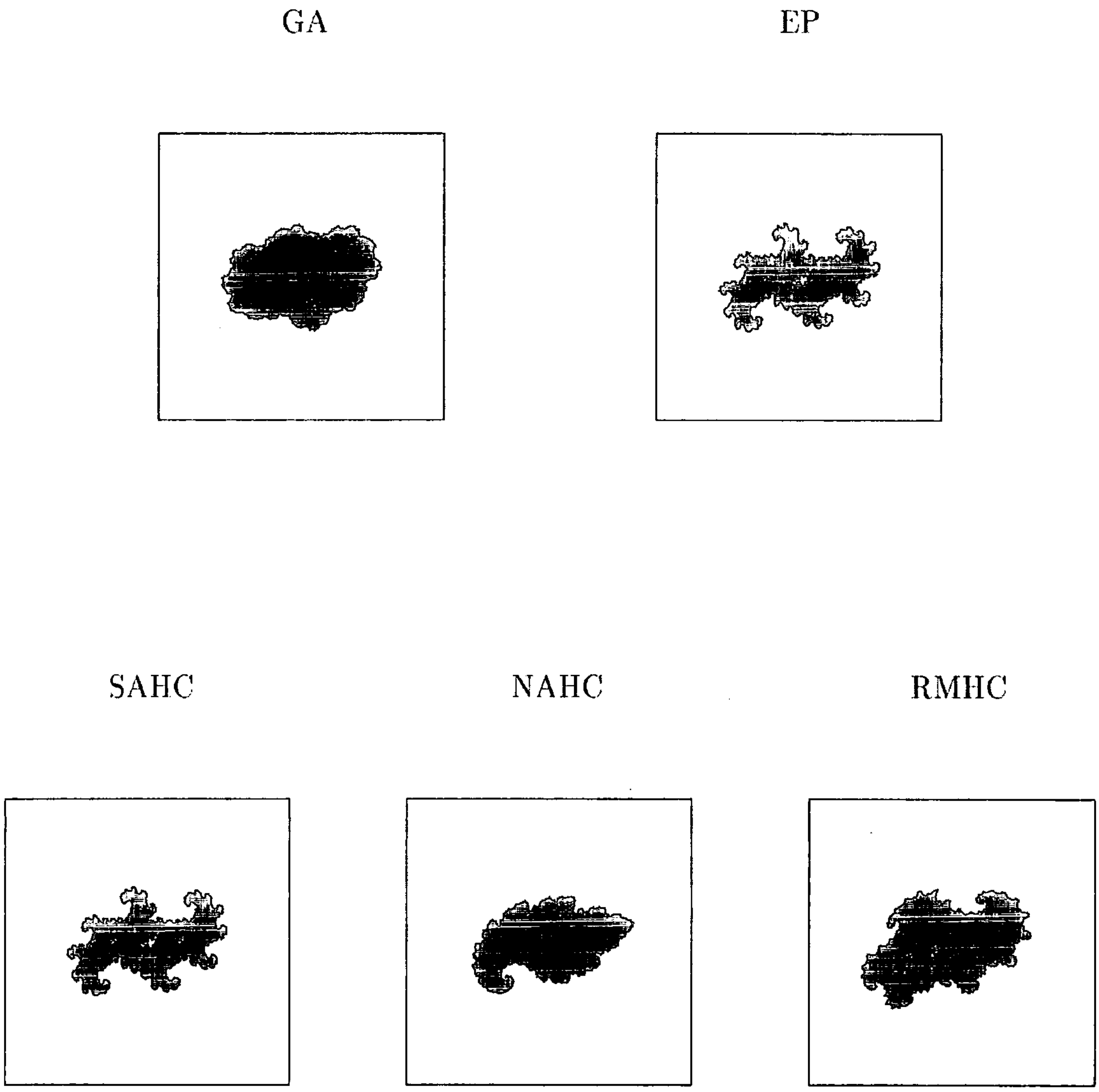


Table 5.9: The best solutions found by each of the search algorithms with a Dragon fractal as the target shape. The collage and point coverage was used as the fitness function. Each algorithm was run 31 times and in the case of the GA and EP the generation at which the best solution was found is shown in parenthesis. The fitness of the best solution found by each algorithm is shown in bold.

Algorithm	EP		GA		SAHC	NAHC	RMHC
Fitness of best solution found (Optimum = 6726)	5391	(91)	2861	(100)	304	2154	1360
	2102	(96)	2604	(27)	2986	3063	176
	3060	(84)	1180	(78)	681	1141	697
	1910	(78)	1163	(97)	3228	431	660
	1808	(73)	2828	(95)	1293	1941	1917
	1720	(99)	788	(92)	311	**3412**	289
	5536	(82)	2640	(90)	951	1797	1745
	2013	(84)	1290	(95)	-163	1155	736
	3168	(99)	1786	(84)	922	1661	1425
	3139	(77)	1463	(77)	801	2884	**4742**
	3162	(47)	1107	(32)	2328	1337	1243
	3410	(99)	1015	(92)	828	1259	771
	1973	(87)	906	(31)	1450	924	2922
	3565	(83)	2642	(97)	2026	777	1124
	1791	(86)	979	(52)	1325	305	2429
	3652	(88)	678	(48)	847	2750	1420
	1814	(77)	2746	(52)	17	-172	2730
	3166	(98)	1549	(74)	**3518**	1787	1742
	1933	(100)	997	(85)	317	478	1161
	3293	(91)	732	(23)	758	1591	782
	2805	(68)	2423	(70)	682	1687	1825
	3376	(50)	3535	(93)	1947	421	1419
	3461	(87)	**4613**	(97)	1917	608	510
	3330	(88)	1085	(48)	811	1932	3048
	3343	(100)	681	(76)	1182	1679	1161
	1790	(91)	1119	(97)	1020	2240	1555
	5424	(76)	479	(96)	959	1209	1584
	3466	(78)	1234	(92)	2901	1721	916
	3209	(79)	1179	(99)	213	1867	1749
	1839	(99)	1106	(95)	1506	1262	950
	3649	(58)	2459	(58)	576	1353	-520
Mean (to 1 d.p.)	3009.6		1673.1		1240.1	1505.0	1428.0
SD (to 2 d.p.)	1069.68		990.12		953.18	838.82	999.91

Figure 5.23: A sequence of attractors from the median trial of the GA and EP with a Dragon fractal as the target shape. The collage and point coverage was used as the fitness function. The attractors shown are those of the best solutions in generations 0, 5, 10, 20, 30, 40, 60, 80 and 100 (top left to bottom right).

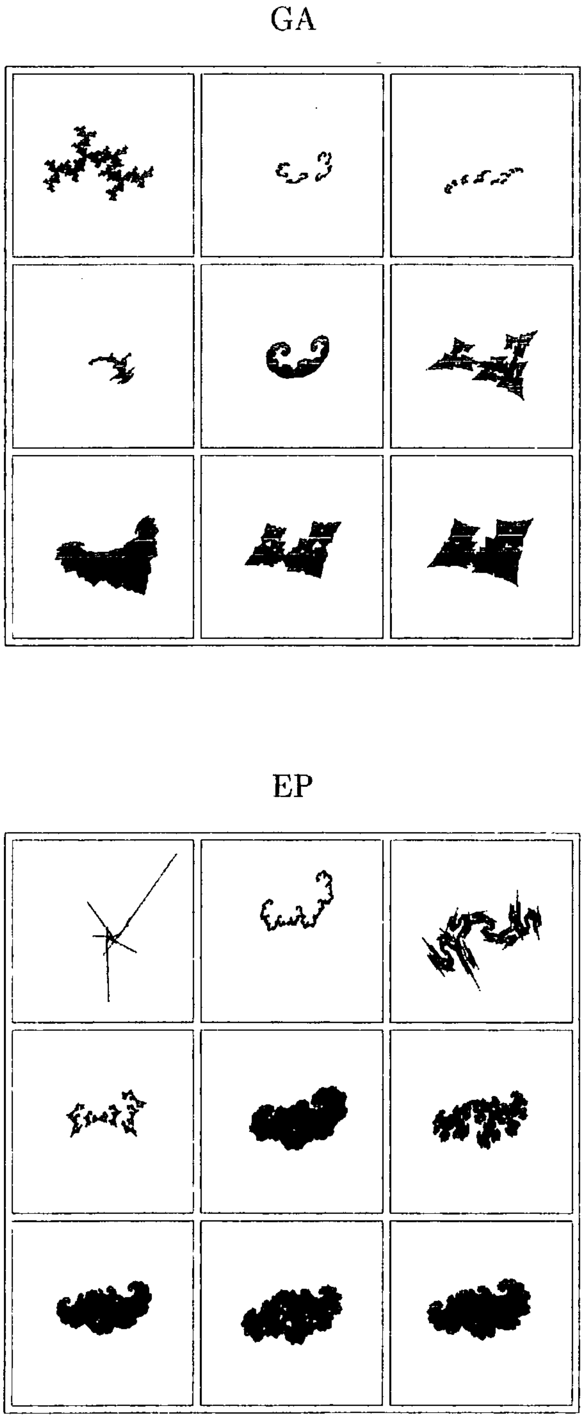


Figure 5.24: Online and offline performance for the median trial of the GA and EP with a Dragon fractal as the target shape. The collage and point coverage was used as the fitness function.

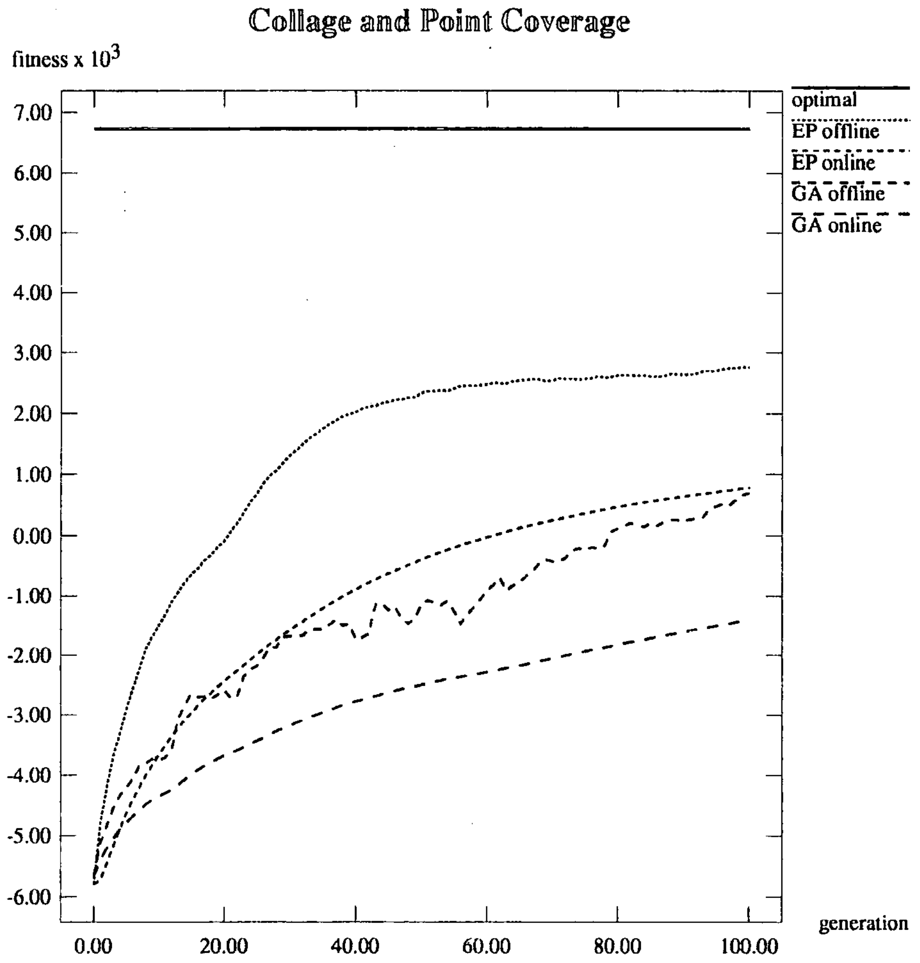


Figure 5.25: Attractors of the best IFSs found when using each of the search algorithms with a Dragon fractal as the target shape. The collage and point coverage was used as the fitness function.

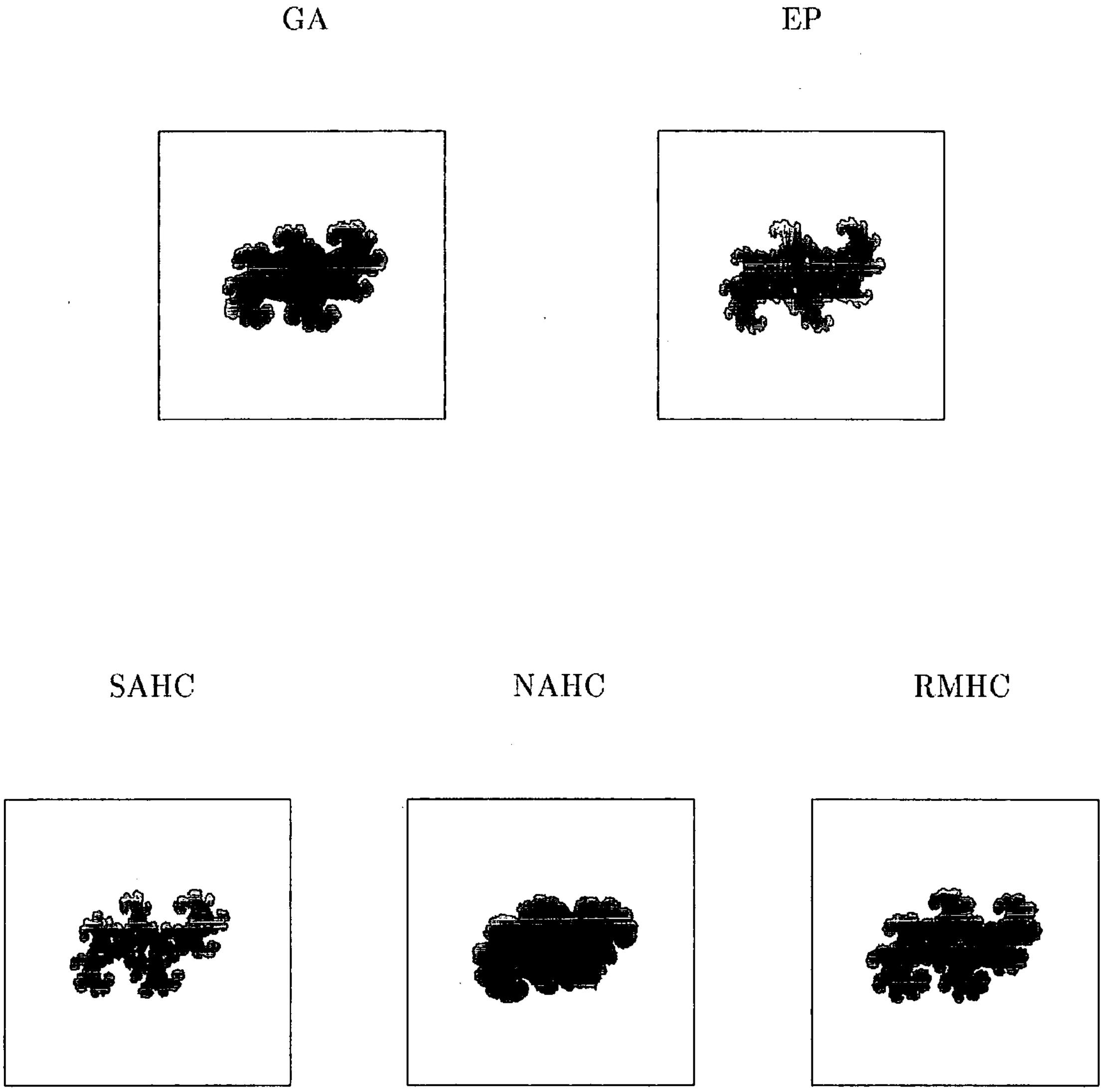


Table 5.10: Values of the t-test statistic, with the number of degrees of freedom shown in parenthesis. The values in the lower left of grid are those for when the attractor and point coverage was used as the fitness function. The values in the top right are those for when the collage and point coverage was used. The values are read: Is LHS better than TOP?

Triangle	EP	GA	SAHC	NAHC	RMHC
EP		18.83 (42)	7.90 (33)	8.13 (32)	9.47 (33)
GA	-16.30 (52)		-2.49 (45)	-0.83 (41)	-0.98 (45)
SAHC	-11.03 (42)	1.35 (53)		1.12 (58)	1.20 (59)
NAHC	-13.96 (38)	-3.49 (46)	-4.16 (56)		-0.02 (58)
RMHC	-10.81 (36)	-1.47 (43)	-2.29 (53)	1.44 (59)	

Sierpinski	EP	GA	SAHC	NAHC	RMHC
EP		8.17 (58)	4.80 (44)	5.78 (44)	4.17 (37)
GA	-16.64 (43)		-0.91 (49)	0.01 (49)	-0.10 (40)
SAHC	-8.91 (56)	3.00 (38)		0.77 (59)	0.53 (53)
NAHC	-10.03 (57)	1.87 (39)	-0.90 (59)		-0.09 (53)
RMHC	-10.41 (59)	3.39 (43)	-0.19 (56)	0.80 (57)	

Dragon	EP	GA	SAHC	NAHC	RMHC
EP		5.11 (59)	6.88 (59)	6.16 (56)	6.01 (59)
GA	-10.43 (59)		1.75 (59)	0.72 (58)	0.97 (59)
SAHC	-9.50 (48)	-2.11 (47)		-1.16 (59)	-0.76 (59)
NAHC	-10.19 (51)	-2.29 (50)	-0.03 (59)		0.33 (58)
RMHC	-10.91 (49)	-3.42 (48)	-1.04 (59)	-1.06 (59)	

Levels of significance for a one-tailed t-test with 30 degrees of freedom.

t test statistic	1.30	1.70	2.46	3.39	3.65
Level of Significance	0.1	0.05	0.01	0.001	0.0005

5.6 Discussion

The results show that for the two fitness functions used, EP outperforms a GA and the hill-climbing algorithms on the inverse problem for a triangle, Sierpinski triangle and Dragon fractal. It is probable that by changing the control parameters, solution representation, and so forth, the performance of the GA could be improved, but it appears that there is a fundamental problem with applying GAs to inverse problems: the ***building block hypothesis*** that appears essential for their success does not hold in general. The hypothesis appears to hold at the level of contraction mappings, *i.e.*, good individual mappings can be propagated throughout a population. However, in generating an IFS's attractor, it is the interaction of *all* the mappings that determines the shape produced. For a fitness function using the attractor, the presence of some optimal mappings is not enough to guarantee a good fitness. IFSs with some optimal mappings may, therefore, be assigned relatively low fitnesses and produce very few offspring. It appears that the GA cannot overcome the strong interactions which can occur between the blocks of components of the problem's subsymbolic representation. This problem is overcome somewhat by using a fitness function that uses a collage of the mappings applied to the target shape. Good mappings are then rewarded individually and there is no reliance on other mappings of the IFS, *i.e.*, the effect of the strong interaction between blocks of components is reduced.

The above results show that even when using a fitness function based on a collage of mappings, the GA does not perform all that well. The reason would appear to be due to the need for the *building block hypothesis* to hold *at all* levels. Not only should combining 'good' contraction mappings give 'good' IFSs, but also combining 'good' coefficients should give 'good' contraction mappings. However, a contraction mapping does not depend on individual coefficients, but on their interactions, *i.e.*, there is a strong interaction between subsymbolic components. The poor performance of the GA is, therefore, attributed to the difficulty of being able to construct solutions to the inverse problem from the bottom-up. The GA appears to be unable to overcome the strong interactions which can occur between the components of the problem's subsymbolic representation.

EP suffers from no such problems. By emphasising top-down phenotypic adaptation over genotypic transformation, the building block approach is avoided. Inverse problems are often affected by pleiotropy (a single component affecting several phenotypic traits) and polygeny (a single phenotypic effect being affected by the interaction of many components), and so a subsymbolic search process based at the genotypic level is more likely to be deceived than one which emphasises the phenotypic relationship between each parent and its offspring. These results show that EP not only outperforms a GA, but is able to overcome conditions of pleiotropy and polygeny to obtain near optimal solutions to the inverse problem for a triangle and Dragon fractal.

The hill-climbing algorithms occasionally find near optimal solutions to the triangle and Dragon fractal inverse problems, but perform poorly on that for the Sierpinski triangle. These results are as would be expected, since the search spaces for the triangle and Dragon fractal inverse problems can be expected to contain fewer locally optimal solutions than that for the Sierpinski triangle.

5.7 Summary

IFSs have been suggested as a suitable means of representing shapes for use in a machine vision environment (Giles 1990). The primitives with which a shape is to be encoded are smaller linearly deformed copies of the shape. A shape is represented by an IFS if a collage of primitives can be found such that they exactly cover the shape to be encoded. Finding such an IFS representation for a shape is known as the inverse problem. A symbolic encoding can be adopted and the inverse problem solved by symbolically manipulating the primitives. Approaches along these lines usually greatly restrict the primitives allowed and/or require human interaction.

A subsymbolic representation allows for a more flexible approach, but results in a large search space with complex interactions between its components. There are often strong interactions between blocks of parameters, and between the parameters of each block.

Many methods have been suggested for solving inverse problems for which a subsymbolic representation has been adopted. Many of these approaches, although automatic, are very limited in application since in order to be successful they greatly simplify the problem. This chapter has described how a GA, EP and three hill-climbing algorithms can be applied to inverse problems for which a subsymbolic approach has been adopted (binary for the GA and three hill-climbing algorithms, and real-valued for EP). The main simplifying assumption which was made was to fix in advance the number of mappings allowed — this allowed for the basic versions of a GA and EP to be applied. Even with this simplification the subsymbolic representations' search spaces for the problems considered are extremely large and complex.

Results show that EP outperforms a GA and three hill-climbing algorithms in finding solutions to the inverse problem. The reason for this appears to be the difficulty the GA and hill-climbing algorithms have in being able to overcome the strong interactions which can occur at the subsymbolic level between blocks of components, and between a block's components. The bottom-up approach of a GA is unable to construct suitable solutions. EP on the other hand emphasises adaptation of the behaviour of solutions and is less likely to be deceived by the strong interactions.

The conclusion of this chapter is that a move from a symbolic to a subsymbolic representation can lead to a successful means of solving a shape representation problem. The representations adopted have Strong-Strong interactions (see Section 1.3), but can be successfully tackled using EP. A GA is not so successful.

Chapter 6

Search Space Reductions

This chapter is the final one of the four which examine in detail shape representation using IFSs, and demonstrates how symbolic rules can be applied to the subsymbolic representation to improve the search process. Two possible improvements to the way in which a space can be searched are 1) improve the search algorithm and/or 2) reduce the size of the search space. The previous chapter discussed a search algorithm (EP) which performs well in the space of the subsymbolic representation of several inverse problems. This chapter examines reducing the search space.

In the case of two-dimensional shape representation an IFS is composed of a set of contraction mappings, and so a real-valued subsymbolic representation can be adopted. Each mapping is composed of six real-valued coefficients and although some non-trivial constraints can easily be applied, the search space remains extremely large. In order to help overcome this problem the number of degrees of freedom is often reduced by fixing some of the coefficient values to those of a known optimal solution. Clearly such an approach is of limited use since it requires *a priori* knowledge of an optimal.

The work presented in this chapter reduces the search space by the imposition of necessary constraints on the mapping coefficients (Nettleton and Garigliano 1993, 1994a). In order for the constraints to be of use to a range of search algorithms they are required to need no *a priori* knowledge of any optimum, and to be of a low computational complexity.

Three constraints are introduced which reduce the search space of four of the six coefficients of a mapping by between 20% and 85%, and of the other two coefficients by between 75% and 95% (the size of the reduction depends only on the size of the bounding box of the target shape — see Section 6.5). Since these constraints apply to each mapping of an IFS, the cumulative effect on the search space is substantial. It is anticipated that these reductions in the search space can be used to aid a variety of search algorithms. However, the constraints are not applied to the work of this thesis since (as discussed in Section 5.1) they might bias comparison between a GA and EP in favour of EP.

6.1 Preliminaries

Consider a space $\mathbf{X} \subseteq \mathbf{R}^n$ together with a set of constraints $\{Y_1, Y_2, \ldots, Y_m\}$ such that if $x \in \mathbf{X}$ satisfies the constraint Y_i, then $Y_i(x) = 1$, and if the constraint is not satisfied then $Y_i(x) = 0$. Let $\mathbf{X}' \subseteq \mathbf{R}^n$ be the set $\{x \in \mathbf{X} : \sum_{i=1}^{m} Y_i(x) = m\}$, *i.e.*, the subset of $\mathbf{X}$ which satisfies all of the constraints. In order to select an element of $\mathbf{X}'$ one of the following strategies may be used:

```
Strategy 1 (Select and check)
1. Select $x \in \mathbf{X}$.
2. Calculate $sum = \sum_{i=1}^{m} Y_i(x)$.
3. a) If $sum = m$ then accept $x$. Exit.
   b) If $sum \neq m$ then goto 1.
```

Strategy 2 (Deterministically Select)

1. Select $x \in \mathbb{X}$ such that $\sum_{i=1}^{m} Y_i(x) = m$.
2. Exit.

In cases for which a strategy of type 2 is available it would appear to be the most efficient means of selecting elements of $\mathbb{X}'$ since no spurious elements are considered. If, however, the (computational) complexity of the algorithm which generates guaranteed valid solutions is high then this approach may not be suitable. The complexity itself is largely determined by the interactions between the constraints. For example, consider $x = (x_1, x_2, \ldots, x_n)$ and the constraints $Y_i : -0.5 \leq x_i \leq 0.5$ for $i = 1, 2, \ldots, n$. In such a case the selection of $x \in \mathbb{X}'$ is trivial, with each x_i being selected independently. With complex interactions between constraints, the range of valid values for some x_j, $j \in (1, 2, \ldots, n)$, will often be dependent on the values already selected for one or more of the other coefficients. The selection of valid coefficients can, in some cases, be expected to require complex calculations, and so lead to an increase in complexity.

Strategy 1 may be considered to be inefficient, due to the possibility of generating spurious values. Furthermore, stage 1 requires the selection of some $x \in \mathbf{X}$, and it may be that the space $\mathbf{X}$ is itself constrained in some way such that the selection of x is non-trivial. (For the purposes of this discussion the selection of $x \in \mathbf{X}$ is considered a trivial matter.) On the other hand if stage 1 has a high probability of selecting $x \in \mathbf{X}$, such that $x \in \mathbf{X}'$, and the complexity of checking the constraints is low, then strategy 1 may be preferred.

A further important feature of a selection strategy, which must also be considered, is how the space $\mathbf{X}'$ is sampled. Usually it will be necessary that the selection strategy be capable of generating each element of $\mathbf{X}'$. A strategy which returns the same value (of many possible) each time can be expected to be of limited use. In many cases it will be preferable that each $x \in \mathbf{X}'$ should have the same probability of being selected.

In practice a combination of the strategies 1 and 2 may be desirable. For example, if some of the constraints are trivial then the coefficients satisfying these may be selected according to strategy 2. The remaining coefficients are then subject to strategy 1, with them being chosen and checked until valid values are found.

6.2 Constraints on Mappings

In constructing an IFS $\{\mathbb{X}, w_i : i = 1, 2, \ldots, N\}$ for a two-dimensional shape, $\mathbb{A}$, each of the transformations w_i is of the form:

$$w_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}. \tag{6.1}$$

and the Collage Theorem (3.5.3) provides the following necessary constraint:

$$w_i(A) \subset A \qquad \forall \; i \in \{1, 2, \ldots, N\}.$$

This constraint may easily be applied using a type 1 strategy. This would involve selecting some mapping, applying it to each point of $\mathbf{A}$ and checking that the resulting point is an element of $\mathbf{A}$. If the application of the mapping to a point has one unit of complexity then the above check has complexity $O(m)$, where m is the number of points in $\mathbf{A}$. Although this constraint may be included in a set of constraints, its checking would be computationally expensive when compared to checks of constant complexity — providing the constant is considerably less than m. In this chapter some necessary constraints on each of the transformations, w_i, are introduced. Their main advantage is that they are of a low computational complexity and result in a considerable reduction in the search space.

The constraints which are derived in this chapter apply to each of the mappings of an IFS and so for ease of notation explicit reference to the coefficients of the i'th transformation is dropped.

The following non-trivial constraints are imposed on the coefficients of the transformation given in Equation 6.1:

$$a, b, c, d \in (-1, 1) \qquad e \in [Xmin, Xmax] \qquad f \in [Ymin, Ymax] \tag{6.2}$$

where $Xmin, Xmax, Ymin$ and $Ymax$ are the extent of the shape along the abscissa and ordinate axis. The constraints on e and f are derived from noting that once the original shape has been contracted it should not be moved by an amount exceeding the dimensions of the box containing the original. Such a transformation would produce an unsuitable collage since some point of the collage would lie outside of the original shape. Without loss of generality the origin of the co-ordinate system is chosen such that $Xmin = -Xmax$ and $Ymin = -Ymax$. These constraints are taken as the base case, and all reductions referred to in the remainder of this chapter are measured in terms of percentage reductions of this space.

Giles (1990) uses the constraints $a, b, c, d \in [-0.707, 0.707]$, $e \in [Xmin, Xmax]$ and $f \in [Ymin, Ymax]$. These ensure that the mapping is contractive, but do so at the expense of excluding many valid contraction mappings. For example, mappings with $a = d = 0.8$ and $b = c = 0.0$ are not permitted.

6.3 Calculating Reductions

Once a set of constraints have been identified and applied, some quantitative measure of the size of the search space which remains is needed. For constraints such as those of Equation 6.2 the actual volume of the search space can easily be calculated by integration. However, when there are complex interactions between the coefficients the integration can become extremely difficult. In order to avoid any symbolic integration problems a numerical approach can be adopted. This although approximate is of use in evaluating search space reductions.

Each coefficient of Equation 6.1 is real-valued and can, in theory, take one of an infinite number of values in the ranges given by Equation 6.2. By approximating each range of values to a finite set, the search space can be restricted to a finite set of points. In the case of a, b, c and d the set of allowed values for each coefficient is calculated as follows:

1. Select a required degree of accuracy $acc < 2$ such that $\frac{1}{acc}$ is an integer.
2. The set of allowed values for each of a, b, c and d is then $\{-1 + acc, -1 + 2acc, \ldots, 0, \ldots, 1 - 2acc, 1 - acc\}$.

For example, if $acc = 0.5$ the set of allowed values for each of a, b, c and d would be $\{-0.5, 0, 0.5\}$. By selecting progressively smaller values for acc the size of the search space can be made as large as required. Following this approach the search space for a, b, c and d is approximated to a set of $(\frac{2}{acc} - 1)^4$ distinct points. Table 6.1 shows the number of distinct combinations of a, b, c and d (hence the size of the a, b, c, d search space) for various values of acc. The robustness property of IFSs (Section 4.3) indicates that the value of acc need not be less than 0.01. For computational reasons the minimum value of acc considered in this chapter is usually 0.05.

accuracy (acc)	number of combinations
0.2	65611
0.1	130321
0.05	2.3×10^6
0.02	9.6×10^7

Table 6.1: The number of distinct combinations of a, b, c and d for various values of acc.

The percentage of values of a, b, c and d which satisfy some given constraint(s) can now be calculated in the following way:

1. Select the degree of accuracy, acc, of a, b, c and d required.
2. For each distinct combination of values of a, b, c and d, determine whether or not it satisfies the constraint(s).
3. If, N, is the number of combinations of a, b, c and d that satisfy the constraint(s), then the percentage of combinations that satisfy the constraint(s) for the chosen acc is:
$$\frac{100N}{(\frac{2}{acc} - 1)^4}$$

The coefficients e and f may be treat in a similar way — that is approximating their range of values to a finite set of values. However, as will become apparent later in the chapter the search space reductions for e and f (under the constraints to be discussed), can be easily calculated symbolically without the need for a numerical approximation.

6.4 Eigenvalue Constraint

The transformation given by Equation 6.1 can be written in the form $w\bar{\mathbf{x}} = M\bar{\mathbf{x}} + \bar{\mathbf{c}}$ (dropping explicit reference to i) where M is a 2×2 matrix. The transformation can be applied recursively such that $\bar{\mathbf{x}}_n = w\bar{\mathbf{x}}_{n-1}$ and after n successive applications to a point $\bar{\mathbf{x}}_0$:

$$\bar{\mathbf{x}}_n = PD^nP^{-1}\bar{\mathbf{x}}_0 + (M^{n-1} + \cdots + I)\bar{\mathbf{c}}$$

where D is a diagonal matrix of the eigenvalues of M, P is the matrix of corresponding eigenvectors, and I is the identity matrix.

Since the transformation, w, must be contractive it has a single fixed point and this is equal to $\lim_{n\to\infty}\bar{\mathbf{x}}_n$. The limit point must be independent of the starting point and so $PD^nP^{-1}\bar{\mathbf{x}}_0 \to \bar{\mathbf{0}}$ as $n \to \infty$ for all $\bar{\mathbf{x}}_0$. This leads to:

Constraint 1: The eigenvalues of M, e_1 and e_2, must be such that $|e_1|, |e_2| < 1$.

The eigenvalues are the values of λ for which the characteristic polynomial of M is zero, *i.e.*, $\lambda^2 - (a+d)\lambda + ad - bc = 0$ and so the following constraint can be applied to a, b, c and d:

$$\left|\frac{a+d \pm \sqrt{(a-d)^2+4bc}}{2}\right| < 1. \tag{6.3}$$

Table 6.2 shows the percentage of the search space for a, b, c and d which remains when Constraint 1 is applied, for varying degrees of coefficient accuracy.

acc	0.2	0.1	0.05	0.02
% Remaining	86.1	82.4	80.2	78.8

Table 6.2: The percentage of the search space for a, b, c and d which remains when Constraint 1 is applied.

The constraint on the eigenvalues can easily be implemented using a type 1 strategy. If the first stage of strategy 1 ensures that each combination of a, b, c and d (satisfying Equation 6.2) is equally likely to be selected then the expected value of the algorithm generating a valid solution on the first pass is quite high (the exact values are just those in Table 6.2 divided by 100). It is also possible to use a type 2 strategy to select values for a, b, c and d such that they satisfy Constraint 1. One such algorithm is given in Appendix A.

6.5 Limit Point Constraint

The introduction to this chapter stated that the reductions achieved are dependent only on the size of the bounding box of the target shape. A shape's **bounding box** is now defined.

Definition 6.5.1 *The bounding box (BB) of a shape S is defined as the box defined by $-Xmax \leq x \leq Xmax$ and $-Ymax \leq y \leq Ymax$.*

The constraint introduced in this section is derived from noting that if the fixed point of a contractive transformation lies outside of a shape, then that transformation cannot be part of an IFS encoding for that shape. The calculation of the fixed point of a transformation is a simple task (see below), as is checking if it lies within the shape to be encoded. This allows for a strategy of type 1 to be implemented with the reduction in the search space being dependent on the target shape. By relaxing the condition, the following constraint is introduced for which a computationally efficient type 2 strategy is easily developed:

Constraint 2: The limit point of a contraction mapping must be contained within the bounding box of the target shape.

When this constraint is applied, it is possible to choose e and f (for given values of a, b, c and d satisfying Constraint 1) such that the resulting transformation's limit point lies inside the shape's BB (a type 2 strategy). This removes the part of the search space of e and f for which the limit point lies outside the shape's BB and hence can not lie on the shape to be encoded. The size of the reduction in the search space when Constraint 2 is applied is now calculated.

The unique fixed point, $(\hat{x}, \hat{y})$, of a contractive transformation of the type given by Equation 6.1 can be found by solving the simultaneous equations $\hat{x} = a\hat{x} + b\hat{y} + e$

and $\hat{y} = c\hat{x} + d\hat{y} + f$. This gives:

$$(\hat{x}, \hat{y}) = \left(\frac{e(1-d) + bf}{(1-a)(1-d) - bc}, \frac{f(1-a) + ce}{(1-a)(1-d) - bc} \right)$$

where $(1-a)(1-d) - bc \neq 0$.

Requiring the limit point to lie in a shape's BB requires that the following conditions hold; $\hat{x} \in [-Xmax, Xmax]$ and $\hat{y} \in [-Ymax, Ymax]$. So:

$$-X \leq e(1-d) + bf \leq X \qquad -Y \leq f(1-a) + ce \leq Y \tag{6.4}$$

where $X = ((1-a)(1-d) - bc)Xmax$ and $Y = ((1-a)(1-d) - bc)Ymax$. The Equations 6.4 define a parallelogram since $\frac{b}{1-d} \neq \frac{1-a}{c}$ (from above). Furthermore, Equation 6.2 requires:

$$e \in [-Xmax, Xmax] \qquad f \in [-Ymax, Ymax]. \tag{6.5}$$

For a given a, b, c and d, the values of e and f (satisfying Equations 6.5) which ensure the limit point of the resulting transformation lies in the BB are those which lie in the intersection of the parallelogram defined by Equation 6.4 and the rectangle defined by Equations 6.5. The area of intersection is just the size of the search space of e and f remaining for a particular combination of a, b, c and d. By calculating the sum of the areas for all combinations of a, b, c and d (satisfying Constraint 1) the overall percentage of the search space of e and f remaining can be easily calculated.

In order to find the area of the polygon which is the intersection of a rectangle and a parallelogram, a clipping algorithm can be used to find the polygon's vertices (Sutherland and Hodgman 1974; Newman and Sproull 1979; Foley and Van Dam 1982). Since both the parallelogram and the rectangle are convex and centred at the origin, then so is the polygon of their intersection (by symmetry). The polygon's vertices are sorted using polar coordinates so that they are clockwise about the origin starting at the negative x-axis. Once this ordered set of vertices has been found, the area of the polygon, and hence the area of the search space of e and f for a particular a, b, c and d can be found. The area is calculated using the following algorithm which acts on the arrays $x[n]$ and $y[n]$, where n is the number of vertices $(x[i], y[i])$ $i \in \{0, 1, \ldots, n-1\}$ of the polygon of intersection.

```
procedure polygon_area(x[n],y[n])
{calculates area of polygon with
    ordered vertices (x[i],y[i]) i=0,1,...,n-1};
  area := 0.0;
  for i:= 0 to n-2 do
    area := area+(x[i]*y[i+1]-y[i]*x[i+1]);
    end;
  area := area+(x[n-1]*y[0]-y[n-1]*x[0]);
  area := area/2;
end {polygon_area}
```

The graph shown in Figure 6.1 shows the percentage of the search space of e and f remaining when the constraint on the limit point of a transformation is applied for varying sizes of the BB. The reduction in the search space of a, b, c and d when Constraint 1 is applied is also shown ($acc = 0.05$).

Appendix B provides an algorithm which, given a, b, c and d (satisfying Constraint 1), selects e and f such that the resulting transformation's limit point lies within a shape's BB.

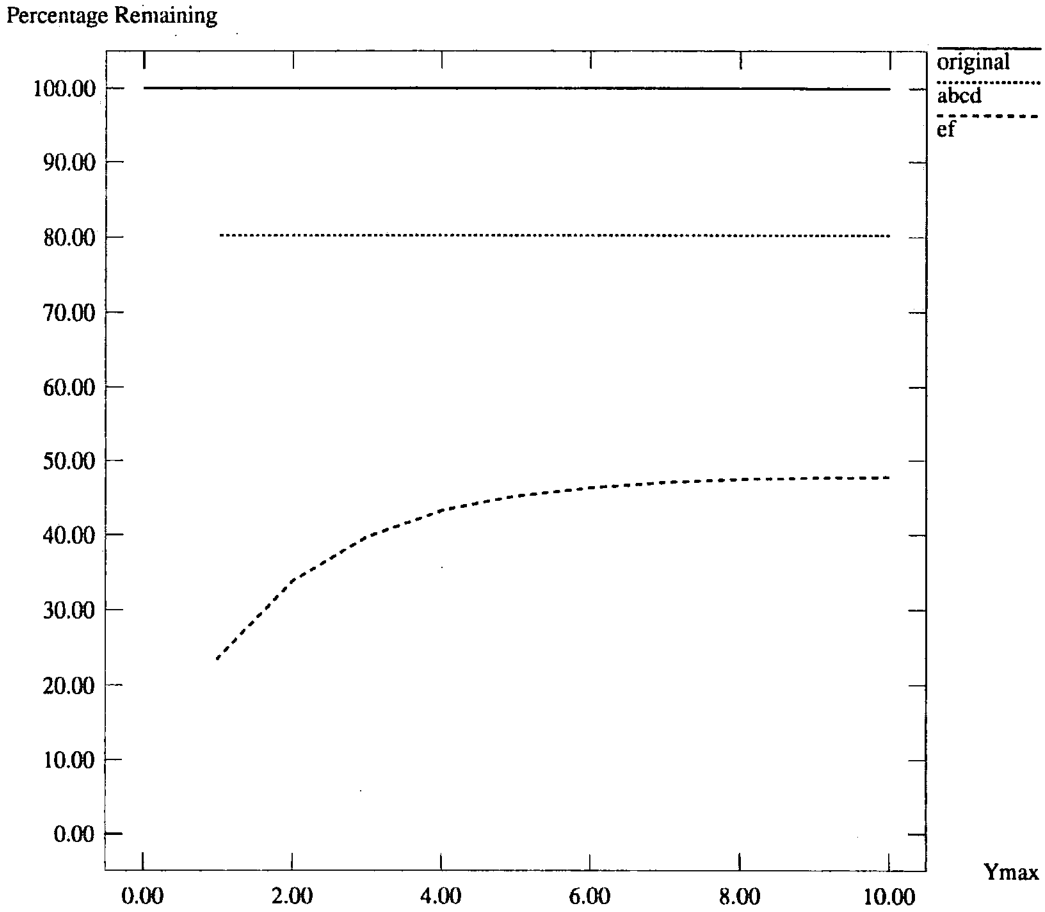


Figure 6.1: The percentage of the search space remaining when the eigenvalue constraint is applied to a, b, c and d (Constraint 1), and the limit point constraint is applied to e and f (Constraint 2) for varying sizes of the BB. Xmax = 10.0 and acc = 0.05.

6.6 Constraint on Transforming the Bounding Box

The final constraint introduced in this chapter is:

Constraint 3: After the application of a contractive transformation to the bounding box of a shape no edge of the resulting parallelogram can lie entirely outside the original bounding box.

This condition must hold since each transformed edge of a shape's BB touches the transformed shape, and so if an edge lies outside the BB then so does part of the transformed shape. Such a transformation is not suitable (from the Collage Theorem).

The implementation of this constraint is considered in two parts:

1. The 2×2 matrix in Equation 6.1 must never allow an edge of the BB to be transformed outside the BB — a constraint on a, b, c and d.
2. The transformed bounding box (TBB), produced by applying a valid 2×2 matrix (with regard to 1. above) to the BB, can never be moved such that an edge of it would lie outside the BB — a constraint on e and f.

The following procedure is used to check the validity of a, b, c and d (a type 1 strategy), and then calculate ranges for e and f such that Constraint 3 is satisfied (a type 2 strategy).

Apply the matrix to two vertices of the BB $(Xmax, Ymax)$ and $(-Xmax, Ymax)$ to give two vertices of the TBB, $T^0 = (T^0_x, T^0_y)$ and $T^1 = (T^1_x, T^1_y)$. Note that the other vertices of the TBB are then $T^2 = -T^0$ and $T^3 = -T^1$. Then if:

1. T^0 and T^1 lie inside the BB

   a, b, c and d are valid, and e and f are chosen such that:

   $$|e| \leq Xmax - \min\{|T_x^0|, |T_x^1|\} \qquad |f| \leq Ymax - \min\{|T_y^0|, |T_y^1|\}.$$

2. One of T^0 and T^1 lie inside the BB

   a, b, c and d are valid, and e and f are chosen such that if T^i is the point inside the BB:

   $$\begin{aligned} |e| &\leq Xmax - \min\{|T_x^i|, \max\{|I_x^0|, |I_x^1|\}\} \\ |f| &\leq Ymax - \min\{|T_y^i|, \max\{|I_y^0|, |I_y^1|\}\} \end{aligned}$$

   where I^0 is the point of intersection of the line from T^0 to T^1 and the BB, and I^1 is the point of intersection of the line between T^0 and T^3 and the BB.

3. T^0 and T^1 lie outside the BB

   The number, N_1, and positions I^2, I^3, of the points of intersection of the line from T^0 to T^1 and the BB are calculated. Similarly the number, N_2, and positions I^4, I^5, of the points of intersection of the line from T^0 to T^3 are calculated. Then either:

   (a) $N_1 = 0$ or $N_2 = 0$. In which case an edge of the TBB lies outside the BB and so a, b, c and d are not valid coefficients.

   (b) $N_1 > 0$ and $N_2 > 0$. In which case a, b, c and d are valid and:

   $$|e| \leq Xmax - \max\{|I_x^j|, |I_x^k|\} \qquad |f| \leq Ymax - \max\{|I_y^l|, |I_y^m|\}$$

   where j and k are such that $|I_y^j| = |I_y^k| = Ymax$, and l and m are such that $|I_x^l| = |I_x^m| = Xmax$.

Figure 6.2 provides an example of each case of the above cases. The graph shown in Figure 6.3 shows the reductions achieved when Constraint 3 is applied to shapes which have varying sizes of the BB ($acc = 0.05$).

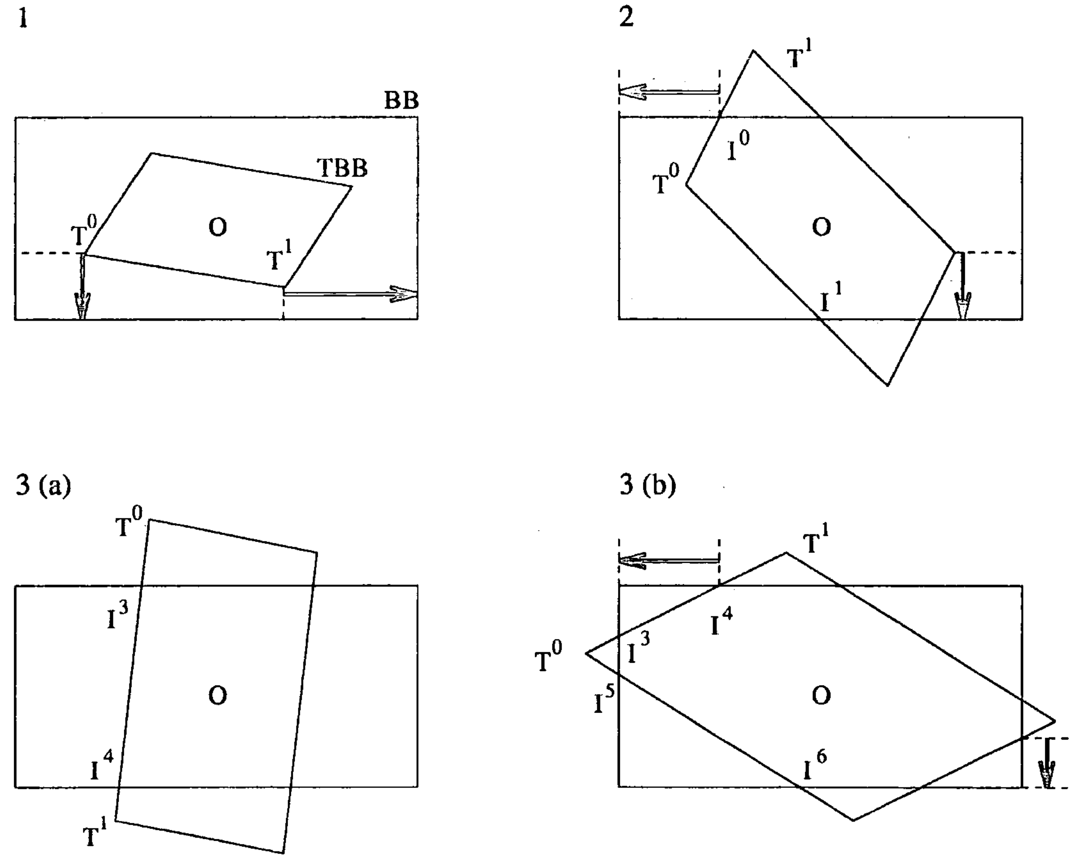


Figure 6.2: An example of each type of transformation which needs to be considered when implementing the constraint on transforming a bounding box (Constraint 3). The rectangles in each part represent a BB, while the other parallelogram is the TBB (a 2 × 2 matrix satisfying Constraint 1 applied to the BB). Parts 1, 2 and 3(b) show valid transformations of the BB. The transformation shown in 3(a) is not valid since an edge of the TBB lies entirely outside the BB. The horizontal and and vertical arrows indicate the magnitude of the maximum displacement of the TBB which can occur before an edge lies entirely outside the BB. These magnitudes are constraints which can be imposed on e and f.

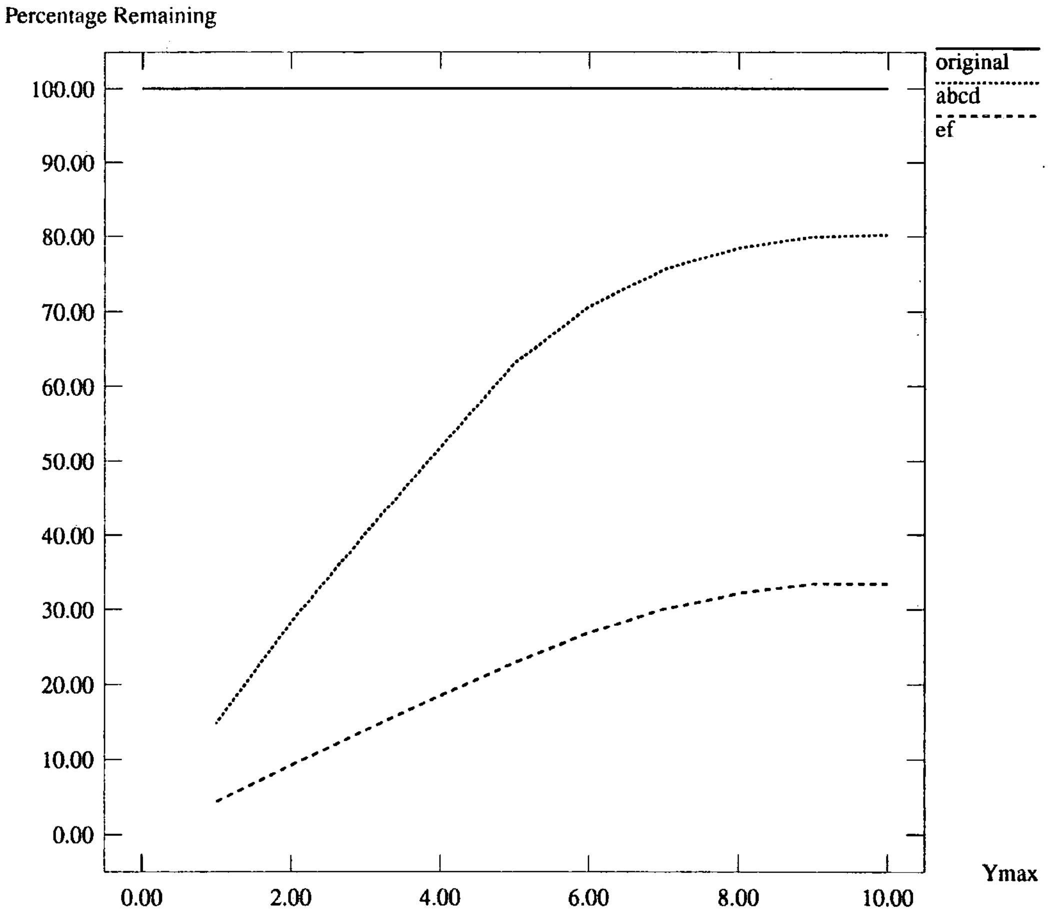


Figure 6.3: The percentage of the search space remaining when the eigenvalue and TBB constraints are applied to a, b, c and d (Constraints 1 and 3), and the TBB constraint is applied to e and f (Constraint 3) for varying sizes of the BB. Xmax = 10.0 and acc = 0.05.

6.7 Total Reduction

The constraints discussed in this chapter are combined to give the overall reduction in the search space which has been achieved. The results are shown in Figure 6.4.

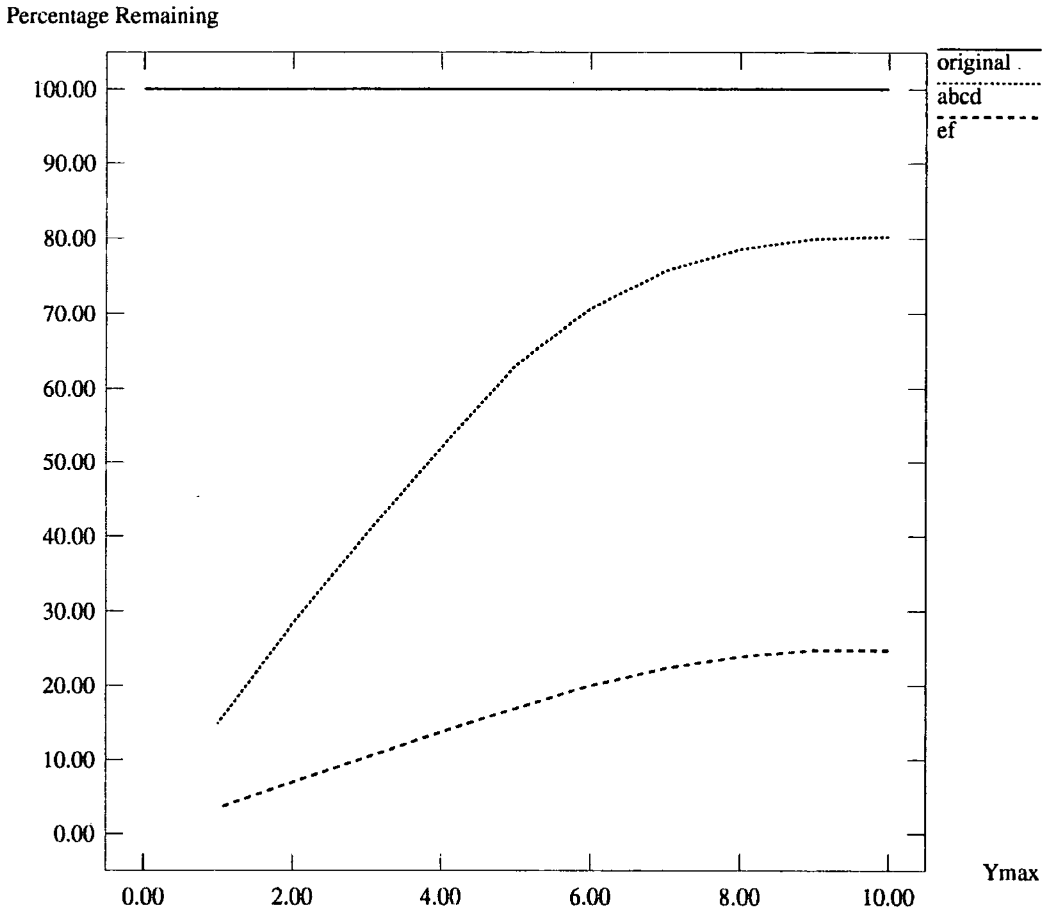


Figure 6.4: The percentage of the search space remaining when the eigenvalue and TBB constraints are applied to a, b, c and d (Constraints 1 and 3), and the limit point and TBB constraints are applied to e and f (Constraints 2 and 3) for varying sizes of the BB. Xmax = 10.0 and $acc = 0.05$.

6.8 Summary

Improving the performance of a search algorithm is one way in which a space may be searched more efficiently. A second approach is to take advantage of the subsymbolic representation adopted in order to reduce the size of the search space. This chapter has introduced several constraints on the components of an IFS with a real-valued subsymbolic representation. These constraints are of a low computational complexity and reduce the search space of four of the six components of a mapping by between 20% and 85%, and of he other two components by between 75% and 95% (the size of the reduction depends only on the size of the bounding box of the target shape). Since the constraints can be applied to each mapping of an IFS their cumulative effect on reducing the search space is substantial. Furthermore, their low complexity allows them to be implemented efficiently, and it is anticipated that they can be used by a wide range of search algorithms which operate in the space of an IFS's real-valued subsymbolic representation. Appendices A and B provide algorithms for choosing values of IFS components so that they satisfy some of the constraints introduced.

Chapter 7

Evolutionary Algorithms and Dialogue

This chapter describes the second problem for which a hybrid symbolic/subsymbolic approach is successful. Since the aim of the remainder of this thesis is to demonstrate the 'width' of the approach a less comprehensive treatment, than that for the shape representation problem, is given.

LOLITA (Large scale, Object based, Linguistic Interactor, Translator and Analyser) is a natural language processor, the dialogue module of which is based upon a symbolic theory. However, in order to rank the appropriateness of responses to certain situations a subsymbolic (integer) representation is adopted. Blocks of the subsymbolic components determine the type of utterance the system outputs, while components within blocks determine how it is carried out. There are strong interactions between the blocks of components, but weak interactions between a block's components.

Tuning the subsymbolic components so that the system exhibits a certain 'personality' can be carried out by hand. However, this is very time consuming and an automatic means of tuning is desired. This chapter examines how a GA and

EP can be applied to the problem of tuning subsymbolic components. Although successful for the dialogues discussed, limitations of the fitness function currently restrict more general applications (Nettleton and Garigliano 1994b, 1994f, 1994h).

7.1 Introduction

Everyday intelligent beings have to respond to a range of different situations. The question, therefore, arises as to how a suitable behaviour is selected for a particular situation. One explanation would be that there are rules so completely governing possible behaviours that they cover all situations which may be encountered (a purely symbolic model). Clearly, however, while there are certainly some rules which help guide behaviour they certainly do not control it all, and simple counter examples to the above explanation of behaviour are easily constructed. Another extreme possibility would be that no rules are given, but are deduced (for future application) by interacting with intelligent beings and other objects (a purely subsymbolic model). Again this clearly is not true of human behaviour in general. More likely is it that some general rules are given and these, through learning, fine tuned to respond to certain situations (a hybrid symbolic/subsymbolic model). In effect there is an interplay between symbolic and adaptive techniques (Garigliano and Nettleton 1994).

A particular example of a human behaviour, as described above, would be the holding of conversations. Throughout the day one uses a different style of conversation depending on the context, *e.g.*, chatting to a friend, giving a lecture, conducting an interview, *etc.* The use of rules such as being polite, needing to initiate the conversation, *etc*, helps to constrain the content of the conversation. These rules, however, do not cover all eventualities, and one learns to adapt them to other contexts. Furthermore, as a conversation progresses it may be necessary to change the style of the conversation, and so further adaptation takes place. It is certainly not the case that humans learn conversational rules by interaction alone.

For example, one does not learn to be polite at a job interview by being rude at others, and learning from the failures.

In developing a natural language processor able to analyse and respond to natural language input, the application of either of the above extreme methods would be unsuitable. A purely symbolic system can be produced, by specifying a large number of rules, which operates within the domain of those rules. Such systems are usually simple, and often fail when the input is not covered by the rules. Alternatively it is possible to produce a purely subsymbolic system by exposing it to large amounts of data, and hoping that rules can be inferred. This can result in a huge amount of time and resources being expended on learning even the simplest of linguistic rules, let alone more complex ones.

The method adopted at the University of Durham in developing the LOLITA system (Garigliano *et al.* 1993a, 1993b, 1994a) has been to use a mainly symbolic approach — see Section 7.2.1. However, in the dialogue module, situations often arise in which several possible responses are available, and so the system uses a subsymbolic (integer) representation to help select between them. This involves the use of parameters to control the plan boxes which carry out responses. The tuning of these parameters so that a particular behaviour can be achieved has so far been carried out by hand. As this can be a very time consuming process, an automatic means of tuning is desirable. The search space is, however, very large and there are complex interactions between the subsymbolic components — strong interactions occur between blocks of components and weak interactions occur between a block's components. Furthermore, cases arise in which a single parameter can affect several behavioural traits (pleiotropy), and other cases in which several parameters can affect one behavioural trait (polygeny). Search algorithms such as hill-climbing are unsuitable in such spaces.

In addition to demonstrating the success of a hybrid symbolic/subsymbolic approach for dialogue, this chapter examines the use of evolutionary algorithms in fine tuning the parameters controlling the dialogue module of LOLITA (Nettleton and Garigliano 1994b, 1994f, 1994h). This is an example of a problem the representation of which exhibits Strong-Weak interactions at the subsymbolic level (see Section 1.3).

7.2 Natural Language Processing

Natural language processing (NLP) lies at the intersection of disciplines such as artificial intelligence, linguistics and cognitive science. A successful natural language processor must be able to automatically process, understand and generate sections of natural language. Much work in the field of NLP has concentrated on 1) implementing a linguistic theory to show that it can account for the features which it describes (computational linguistics) and 2) the modelling of the human thought process by a computer (cognitive science).

Although these are of much interest, such systems are often so specialised, or so cumbersome, that they cannot be exploited in any practical way. In recent years, however, a more practical approach to NLP has emerged in the form of Natural Language Engineering (NLE); indeed a journal is about to be launched dedicated to this (Garigliano *et al.* 1994b). The paradigm of NLE is the development of systems which are general enough, and quick enough to be of practical use. Such a paradigm takes into account features such as scale, integration, flexibility, feasibility, maintainability, robustness and usability (Smith *et al.* 1994). NLE adopts a pragmatic approach to achieving these goals, which is characterised by a readiness to use any means in order to build serious speech and language processing programs

7.2.1 The LOLITA System

LOLITA is an example of a system created using an NLE methodology (Garigliano *et al.* 1993a, 1993b, 1994a). LOLITA is built around a large semantic network of some 60,000 nodes (capable of over 100,000 inflected word forms) which contain data and world information. The system can parse text, semantically and pragmatically analyse its meaning, and alter the relevant information in the semantic network. Information contained within the semantic network can be generated in the form of natural language (Smith *et al.* 1994), and so a 'natural' interaction with the system is possible. Having being developed using an NLE methodology the system is very general. Recently the underlying system has been used (with little in the way of modification) as the base for a variety of prototype applications. These include an Italian to English translator, contents scanning of newspaper articles, Chinese tutoring, and dialogue analysis and generation.

The LOLITA system incorporates several logical and linguistic theories in its general construction. However, in dealing with specific areas these theories are often not strong enough, and so more localised theories are used. Even when these localised theories are impractical (*e.g.*, for efficiency reasons) the LOLITA system resorts to a knowledge based approach, or uses heuristics, to solve problems. By incorporating such a range of approaches LOLITA is able to enjoy the advantages provided by a well constructed general theory. At the same time LOLITA is flexible enough to use other approaches should these theories fail for particular problems.

7.3 Dialogue in LOLITA

This section discusses the theory of dialogue which is used within the LOLITA system. An account of this theory is given so that its power can be appreciated. First of all, however, definitions are given of some terms which may otherwise be open to various interpretations.

The terms dialogue and discourse are usually used loosely by many workers in the field. The definitions which are used in this chapter are those given by Jones and Garigliano (1993). Discourse is taken to mean a set of sentences which are related to each other both linguistically and contextually. Such a definition includes newspaper articles, but an interaction between participants is not a requirement for a discourse. Dialogue is taken to be the rich interaction between two or more participants, where 'rich interaction' is taken to include features such as sub-dialogues, interruptions and complex shifts in focus.

Theories of dialogue can be broadly classified as: descriptive, prescriptive, predictive and inferential. A descriptive theory is simply aimed at being able to describe a known piece of dialogue in terms of some set of features. The other types of theory are more useful since these can be used (with varying degrees of power) to provide information on what is to happen next in the dialogue. In a general natural language processor once a piece of text has been analysed the system needs to prepare a response. Rather than simply responding with the same style of text for all situations, LOLITA is capable of producing a wide range of styles. A theory of dialogue capable of providing information on a suitable response is required. Such a theory has been developed over the past three years (Jones and Garigliano 1993; Jones 1994).

7.3.1 Dialogue Situations

In many situations in which humans find themselves, the type of dialogue structure that can be expected for that particular situation is known. The knowledge required to determine this has been acquired through a mixture of given rules and learning (Section 7.1). In order to take advantage of this knowledge Schank and Abelson (1977) introduced the idea of scripts. A script is described by Schank and Abelson (1977, p. 41) as "... a structure that describes appropriate sequences of events in a particular context ... a predetermined, stereotyped sequence of actions that defines a well-known situation." An example of a script would be the dialogue between a waiter and customer in a restaurant. In such a situation both participants can be considered to be filling in the slots of some pre-determined template which has slots for actions such as ordering food.

Scripts are used to describe events from the physical world. The theory of dialogue incorporated in LOLITA is aimed at modelling the actual structure of the dialogue. This theory is based on the concept of a Dialogue Structure Model (DSM), and is now described (Jones 1994).

A DSM is a schema which contains all of the information that can be expected to be relevant in a particular situation, and thus can be used to guide the generation of language to suit that situation. The DSM consists of dialogue elements, which are factors that influence and control the structure of the dialogue. In a lecture, for example, the lecturer can be expected to be in control of the dialogue, and to speak for most of the lecture's alloted time. Factors such as these determine the basic information required for a class of similar situations. Furthermore, a theory of dialogue based on DSMs is not simply descriptive, for a DSM can prescribe the manner in which the remainder of the dialogue is to be carried out.

7.3.2 Dialogue Elements

The Dialogue Elements (DEs) are the fundamental components of a DSM, and the current set can be subdivided as follows.

External Elements — These are elements which are external to the language itself. Although they are not part of the dialogue they influence its structure.

- **Number** — The number of participants involved in the dialogue.
- **Time Limit** — Whether or not there is a specific limit on the amount of time available within which the dialogue must be completed. Whether or not the dialogue must terminate by a particular time.
- **Temporal Progression** — The stages through which the dialogue progresses as time passes. For example, in a lecture one can expect an intro-

duction, a main body and a conclusion. In a chat, however, there is far less structure.

Motivational Elements — All dialogues are started for some purpose, whether it be to simply pass the time of day or to conduct an interview. The elements discussed below are connected to the purposes for which a dialogue is being held, and are linked to the goals, motivations and intentions of the participants in the dialogue. Since a dialogue always has a motive, a DSM must always contain a motivational dialogue element.

- **Emotional Exchange** — Whether or not any of the dialogue's participants aim to change the emotional state of another participant, *e.g.*, make them laugh, cry or indifferent.
- **Goal** — This is divided into 'task' and 'process', and relates to the aim of the dialogue. If the aim is that of a task, then the goal is used to specify some end result, *e.g.*, verbal instructions for the assembly of a piece of machinery. Process goals are achieved in stages as the dialogue progresses, *e.g.*, a lecture conveys information on some topic as it unfolds.
- **Information Seeking** — Whether or not any of the dialogue's participants aim to gain information during the dialogue.
- **Persuasive** — Whether or not the aim of any of the dialogue's participants is to cause another participant to believe in the truth of some statement.

Verbal Elements — These are verbal properties of a dialogue, and may or may not be present within the dialogue.

- **Colour** — This relates to the style of language, *e.g.*, use of adjectives, figures of speech, analogies, *etc.*

- **Distribution of Time** — The amount of speaking time that each participant is allowed within the dialogue. In a lecture, for example, the students can be expected to speak far less than the lecturer.
- **Dominance** — Determines the degree of control a participant has on the structure of dialogue, content or direction.
- **Fixed Topic** — Whether the dialogue is constrained to be on one topic or whether the dialogue can cover several topics.
- **Length** — The length of sentences contained within the dialogue, *e.g.*, long or short.
- **Register** — This relates to the kind of vocabulary that is in use within the dialogue, *e.g.*, formal, informal, slang, *etc.*
- **Rhythm** — The rhythm of the dialogue. If, for example, it is to progress in short bursts or long flowing constructions.

All dialogues have some form of structure that is external to the situation or participants. For example, all lectures can be expected to have a fixed timespan. In the case of such a dialogue in a particular situation, the relationship, individuality and character of the participants all play an important role in the development of the dialogue. Furthermore, an individual's state of mind at a particular time (*e.g.*, happy, sad) is important in determining how the dialogue progresses. It is through DSMs and DEs that the LOLITA system models these parts of human behaviour.

7.3.3 Constraints and Plan Boxes

Although the situation, character, *etc*, allows humans to place many constraints on the responses which may be made in some situation, there are still many possibilities. The process of selecting an appropriate response is one which humans take for granted. LOLITA like a human is capable of many responses, and therefore needs some mechanism by which responses can be selected. Once a response has been selected plan boxes are used to inform on how and when the output is generated. There are currently some 124 plan boxes contained within LOLITA, and some means of selecting a plan box from the many possibilities is required.

LOLITA is able to reduce the number of possibilities via inference and heuristics. Inference on the input is used to examine its emotional and intellectual value. Heuristics are then used to ensure that certain plan boxes are not triggered. For example, if LOLITA is forced not to be rude then blocks of plan boxes that would result in a rude response are excluded. Once these processes have been performed the LOLITA system is usually left with some 10-15 plan boxes which correspond to different outputs. Some mechanism for determining how likely a certain response is for a particular situation is needed. For example, one may not wish to answer a question, and possible responses could involve replying with a question or simply saying 'I don't want to talk about that'.

The problem that remains is how to order the possibilities, dependent on the behaviour which is being sought. If, for example, the current DSM dictates that dialogue participant X has a greater level of dominance than participant Y, it is possible for X to terminate the dialogue. Although the termination of the dialogue is permitted it may not be appropriate at particular points of a dialogue — a lecturer has greater dominance in a lecture, but would not be expected to terminate the dialogue half way through without adequate explanation. So although 'terminate dialogue' is an option it would be inappropriate and must be marked as such. In general no clear rules are available for ranking, and so a subsymbolic approach is adopted. This involves attaching a parameter (an integer) to each plan box to indicate how permissible an action is. Then in selecting a plan box (of those allowed) with which to generate a response, the plan box with the lowest value is used.

It is worth noting that it is not the absolute values of the parameters that is important, but their relative values. Furthermore, as a dialogue progresses the values of the parameters attached to plan boxes vary to take into account the dialogue to that point. For example, if one participant of a dialogue, X, continually annoys another, Y, then Y's terminate dialogue option can be expected to become more likely as the dialogue progresses.

7.4 Tuning the Parameters

The parameters that control the plan boxes contained within the dialogue module of LOLITA have been fine tuned by hand to give a particular behaviour. The 124 plan boxes each have a single parameter attached which may be altered, and so the search space of possible settings is very large. With experience it is possible to acquire an intuitive feel for the optimisation, but this process is very time consuming, and an automatic means of tuning is required.

A method by which the usefulness of a tuning algorithm could be determined would involve first of all tuning the parameters to give a particular series of utterances. Then starting with a random set of parameters, use the tuning algorithm to try and optimise the parameters so that the same series of utterances is generated. Results achieved when adopting such a method are presented later in this chapter. This method has the following advantages:

1. An optimum solution is known to exist for that situation, *i.e.*, the one obtained by hand.

2. A large number of previously untested parameter combinations will be generated, and running LOLITA with these parameters will be a good test of the robustness of the LOLITA system.

The main disadvantage of the above method is that even if the tuning algorithm is successful in the particular situation studied, there is no guarantee that it would be more widely applicable. If it were possible for the algorithms to optimise for a predetermined behaviour, then it would be possible to find parameter settings that resulted in different styles of dialogue, *e.g.*, polite or rude. The later part of this chapter begins to examine the use of EAs in this wider role.

7.5 Target Dialogues

The dialogues given in Tables 7.1 and 7.2 are the target dialogues for the optimisation process. DIAL 1 (Table 7.1) is generated through a single interaction with the LOLITA system for which the parameters were tuned. DIAL 2 (Table 7.2) is made up of statements which it is known the LOLITA system can produce, and were selected from dialogues which had several different parameter settings. In the case of DIAL 2 it is not known if a single parameter setting exists that can produce this entire dialogue.

Table 7.1: The target dialogue, DIAL 1, which was produced in a single interaction with LOLITA.

DIAL 1	LOLITA 1:	I desire a success.
	INPUT 1:	I like you.
	LOLITA 2:	I know; why do you like me?
	INPUT 2:	You are beautiful.
	LOLITA 3:	I am glad because you think I am beautiful.
	INPUT 3:	Do you like me?
	LOLITA 4:	Yes, I like you.
	INPUT 4:	Do you like me?
	LOLITA 5:	I slightly am bored and emotional.
	INPUT 5:	You are a machine.
	LOLITA 6:	I do not believe that I am a machine.
	INPUT 6:	Are you a machine?
	LOLITA 7:	No.
	INPUT 7:	Are you beautiful?
	LOLITA 8:	I do not know if I am beautiful.
	INPUT 8:	I am tired.
	LOLITA 9:	Tell me things.
	INPUT 9:	I am tired.
	LOLITA 10:	I slightly am bored.
	INPUT 10:	Are you tired?
	LOLITA 11:	No.

Table 7.2: The target dialogue, DIAL 2, which is a collection of utterances from different interactions.

DIAL 2	LOLITA 1:	I desire a success.
	INPUT 1:	I like you.
	LOLITA 2:	Why do you like me?
	INPUT 2:	You are beautiful.
	LOLITA 3:	I am glad because you think I am beautiful.
	INPUT 3:	Do you like me?
	LOLITA 4:	Yes, I like you.
	INPUT 4:	Why do you like me?
	LOLITA 5:	I do not know why I like you.
	INPUT 5:	You are a machine.
	LOLITA 6:	I do not believe that I am a machine.
	INPUT 6:	Are you a machine?
	LOLITA 7:	No.
	INPUT 7:	I am tired.
	LOLITA 8:	I understand.
	INPUT 8:	I am tired.
	LOLITA 9:	I slightly am bored.
	INPUT 9:	Are you tired.
	LOLITA 10:	No.
	INPUT 10:	Tell me if you believe that I am tired.
	LOLITA 11:	Yes, you are tired.

7.6 Application of EAs to LOLITA

The environment is the source of information on which solutions are evaluated. In theory this must correspond to all natural language utterances. In practice, however, the current implementation of LOLITA acts only on literal meaning. Metaphors, idioms and humour are, therefore, excluded from the environment.

In deciding which plan boxes are to be activated, it is not their absolute values that are important, but rather their values relative to each other. Therefore, it is not the explicit values of the parameters that are to be optimised, but a shift in value from that of the current hand optimised setting. For each plan box a range of shift values (simply referred to as parameter values from now on) of [-63,64] was deemed sufficient, since these allow for a large range of possible behaviours (if necessary this range can easily be increased). A solution's representation is, therefore, a string of 124 (the number of plan boxes) integers. Furthermore, a solution with all of its values set to 0 is identical to the current hand optimised setting. The parameters of the plan boxes which control utterances of a particular type are grouped together in blocks. For example, the three plan boxes labelled `cause_AffectionPlatonic` are grouped together, as are the six which are labelled `show_AngerOffense`. The components within a block determines how a utterance is carried out, *e.g.*, different ways in which anger can be expressed. There are strong interactions between the blocks of parameters, but weak interactions between the parameters within any particular block.

As mentioned previously some measure of how closely utterances generated match those of the target dialogue is needed. The results given in the next section use a very simple fitness function. A solution's fitness is initially set at zero, and then increased by one for each utterance that exactly matches that in the target dialogue. For the target dialogues discussed in this chapter a solution's fitness is, therefore, an integer in the range [1,11]. The total number of utterances that LOLITA generates is eleven, and so this provides the upper bound on the fitness. Furthermore, all solutions will have a fitness of at least one, since with the current 'personality' LOLITA always initiates a conversation with the phrase "I desire a success." A more sophisticated fitness function is introduced in Section 7.8.

Comparing the results of runs of a GA and EP is difficult since the underlying system is continually changing and the data files regularly updated. Only single trials of each algorithm are carried out, but these are sufficient to show the validity of the approach.

7.7 Results

This section presents the results of applying a GA and EP to the problem of finding plan box parameters. Other than variations in solution representation and the details of the EP's mutation operator (discussed below), the GA and EP used are identical to those described in Sections 5.2 and 5.3 respectively.

The parameters controlling the plan boxes can each take one of 128 distinct values. In implementing the GA, the subsymbolic representation adopted is that of a binary string. Each of the penalties is encoded as a binary string of length seven, and these are concatenated together to form one string. As there are 124 plan box parameters the size of the search space is $2^{7\times124} \approx 10^{251}$.

In applying EP to the penalty optimisation problem an integer subsymbolic representation is adopted. Each of the penalties are stored as integers, and are constrained to the range [-63,64]. A child is produced from a parent by mutating each parameter x_i ($i = 1, \ldots, 124$) according to (and then truncating):

$$x_i' = x_i + \sqrt{5 \cdot (\text{MAX-FIT} - \text{fitness}(X))} \cdot N(0,1) \qquad i \in \{1, 2, \ldots, 124\}$$

where MAX-FIT is the maximum fitness attainable (11 for the work discussed in this section), fitness(X) is the fitness of solution $X = \{x_i : i = 1, \ldots, 124\}$ (*i.e.*, the number of correct utterances) and $N(0,1)$ is a standard normal random variable. The above formula was selected since it allows for solutions with a poor fitness to be mutated by a large amount, while at the same time reducing the chance that the mutated parameters fall outside of the permitted range.

For both the GA and EP a population of 50 was used and they were executed for 50 generations. The tournament size for EP was set at three. A single trial of each algorithm was carried out. Figures 7.1 and 7.2 show the online and offline performance of the GA and EP run, for the target dialogues DIAL 1 and DIAL 2 respectively. The offline performance is the average fitness of all of the solutions in a particular generation, while the online performance is the average fitness of all solutions that have been generated up to a certain generation.

Figure 7.1: Online and offline performance for a trial of the GA and EP with DIAL 1 as the target dialogue.

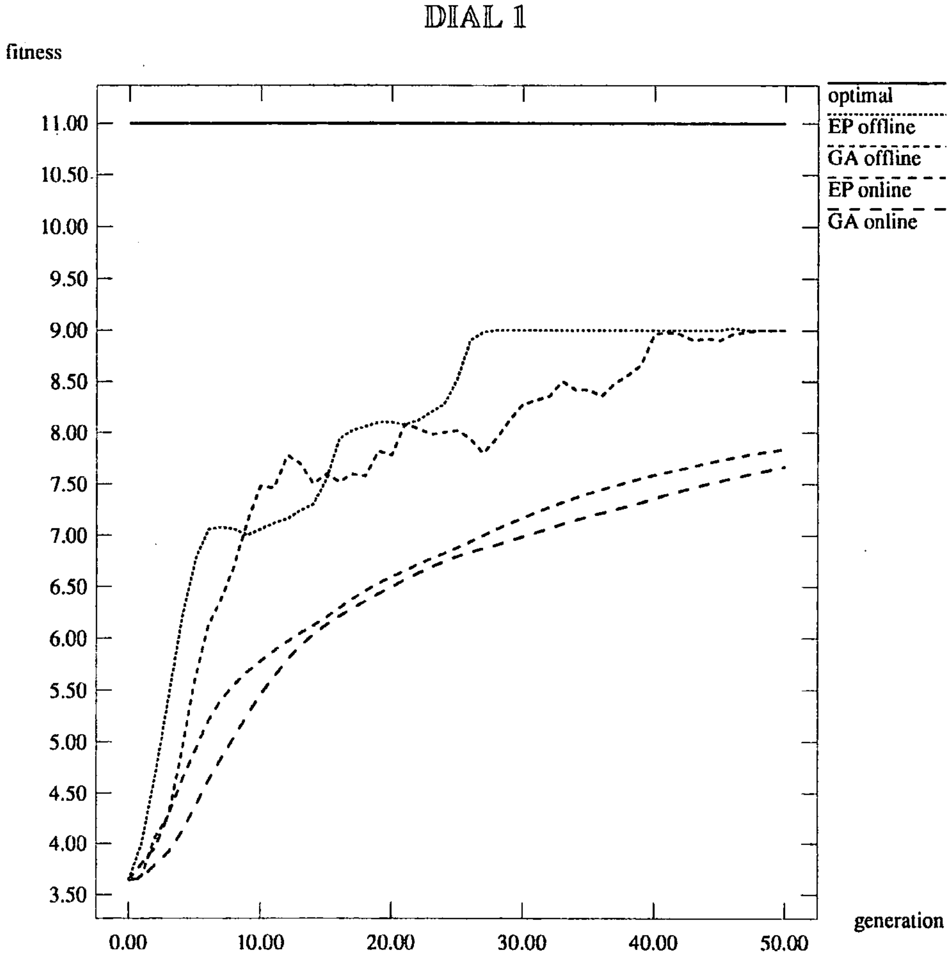
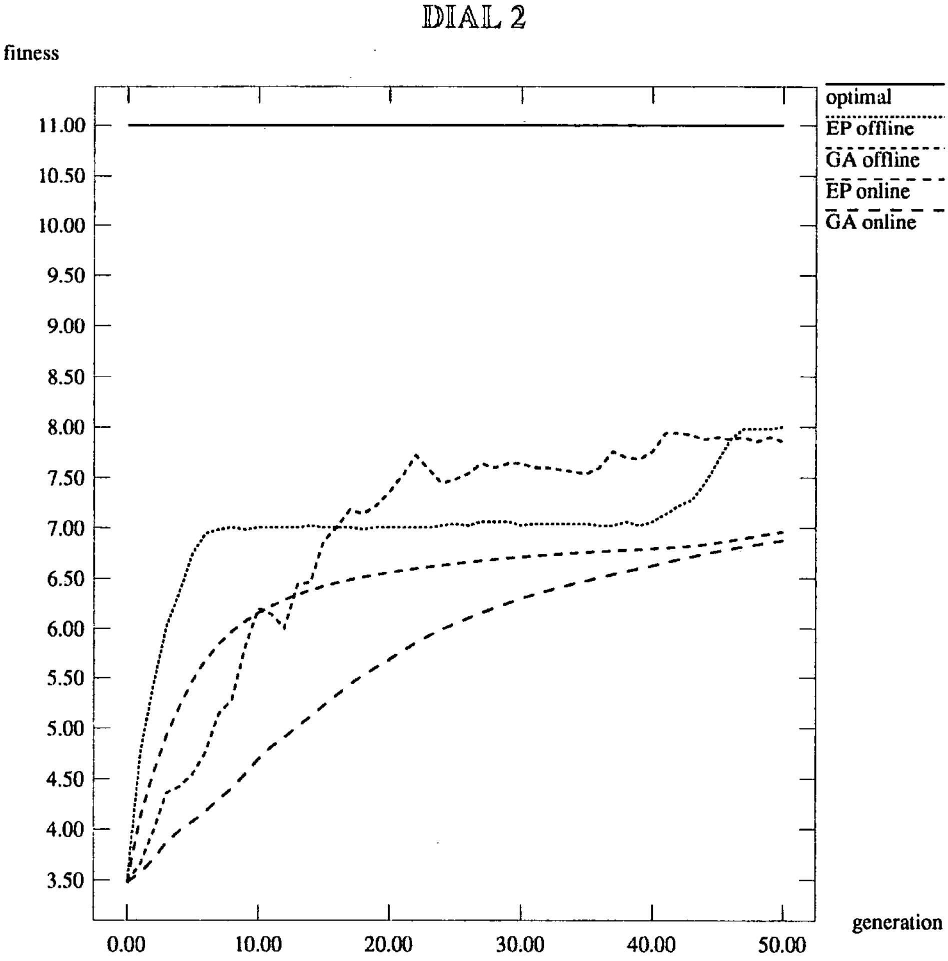


Figure 7.2: Online and offline performance for a trial of the GA and EP with DIAL 2 as the target dialogue.



DIAL	LOLITA's incorrect utterances	
1	GA	2: Tell me things. 6: I slightly am bored and emotional.
	EP	8: I do not know if I am beautiful; tell things to me.
2	GA	2: I could not speak to you if you repeated you like me. 6: I desire a success. 7: I could not speak to you if you repeated Am I a machine?
	EP	2: I know; I could not speak to you if you repeated you like me. 7: I could not speak to you if you repeated Am I a machine?

Table 7.3: The incorrect utterances generated by the best parameters found when GA and EP were used to optimise the plan box parameters for DIAL 1 and DIAL 2.

In the case of DIAL 1 the GA was able to find a set of parameters which produced a dialogue of fitness 9, *i.e.*, two utterances incorrect. EP performed slightly better, discovering a solution of fitness 10. When DIAL 2 was used as the target dialogue the GA was able to find a solution of fitness 8, and EP a solution with fitness 9. These results are summarised in Table 7.3.

In the case of EP the incorrect utterance for DIAL 1 was "LOLITA 8: I do not know if I am beautiful; tell things to me." Such an utterance should not be considered as wrong, it is simply that the fitness function is not very sophisticated. Similarly for the GA and DIAL 1. For both the GA and EP, with DIAL 2 as the target dialogue, the incorrect utterances for the best parameters found indicate that the parameter settings were such that the input caused LOLITA to become offended quite easily.

An interesting feature of the EP results is how the average fitness of a generation rose to that of the best solution to date (Figures 7.1 and 7.2). It appears that when a better solution was discovered the average generation fitness would rise gradually for several generations and then quickly rise to that of the best. There is, however, one notable exception to this which occured at generation 46 when DIAL 1 was the target dialogue — see Figure 7.1. At this point a solution of fitness 10 was produced in a population the remainder of which had fitness 9. The solution of fitness 10 was, however, subsequently lost and the reason for this is now discussed. Although a solution with fitness 10 is guaranteed a score of three in the tournament, many other solutions in that population also scored a fitness of three since all but one solution against which they were competing had a fitness of 9. When the process of sorting the scores took place there were more solutions with a score of three than there were places for them in the next generation and so some were lost. This included the solution of fitness 10. A similar occurrence took place in the run with DIAL 2. A olution of fitness 9 was discovered at generation 25, retained for one generation, and then lost.

The failure to retain an improved solution is in part attributable to the poor discriminatory power of the fitness function used. Since many solutions can have the same fitness a lot of solutions often perform very well in the tournament, and solutions with a maximum tournament score may be lost from the following generation. The following section examines a fitness function which is able to use additional information which the LOLITA system is able to provide. This improves the fitness function's discriminatory power, rewarding not just the words produced, but the underlying actions which lead to their generation.

7.8 Improving the Fitness Function

The fitness function adopted in the previous section is very simple and unable to take into account additional information which the LOLITA system is able to provide. On analysing an utterance the LOLITA system infers information on the local goals, subgoals, utterance types and action types of the speaker. Table 7.4 shows this information for the first seven utterances of DIAL 1.

Table 7.4: The additional information which the LOLITA system makes available for the first seven utterances of DIAL 1.

LOLITA 1: (I desire a success.)	speaker: lolita local goal: ShowEmotionGoal NeutralEmotion subgoals: utterance types: AllSame action types: default_tacticPB
INPUT 1: (I like you.)	speaker: roberto local goal: InformGoal subgoals: utterance types: Statement action types:
LOLITA 2: (I know; why do you like me?)	speaker: lolita local goal: InformGoal subgoals: AnyGoal, BeInformedGoal utterance types: Statement, Noise, Question action types: tellPB, why_questPB
INPUT 2: (You are beautiful.)	speaker: roberto local goal: InformGoal subgoals: utterance types: Statement action types:
LOLITA 3: (I am glad because you think I am beautiful.)	speaker: lolita local goal: ShowEmotionGoal Serenity subgoals: utterance types: AllSame action types: show_Serenity
INPUT 3: (Do you like me?)	speaker: roberto local goal: BeInformedGoal subgoals: utterance types: Question action types:
LOLITA 4: (Yes, I like you.)	speaker: lolita local goal: InformGoal subgoals: utterance types: Statement action types: answerPB

The fitness function can be modified to make use of the additional information given in Table 7.4. The fitness function used in this section calculates a solution's fitness by initially setting it to zero, and increasing it by one for each utterance, local goal, subgoal, utterance type and action type, which exactly matches that of the target dialogue. This fitness function is less sensitive to the utterance itself and more sensitive to the behaviour required. For DIAL 2 the information associated with the individual statements was used.

Using this additional information a fitness, which is an integer in the range [5,55] can now be assigned to solutions — 5 forms the lower bound since LOLITA always initiates a conversation with the same utterance and associated information. For each of the two target dialogues a single trial of the GA and EP were carried out. The GA and EP used the improved fitness function and in addition two modifications were made to EP. In the tournament phase of the algorithm if two solutions have the same fitness then a win is awarded with probability 0.5. This modification is aimed at helping to overcome the problem of EP 'losing' a solution which arose in the experiments with the first of the fitness functions discussed. Secondly, the EP's mutation operator is altered so that a child is produced from a parent by mutating each parameter x_i ($i = 1, \ldots, 124$) according to (and then truncating):

$$x_i' = x_i + \sqrt{(\text{MAX-FIT} - \text{fitness}(X))} \cdot N(0,1) \qquad i \in \{1, 2, \ldots, 124\}$$

where MAX-FIT is the maximum fitness attainable (55 for the work discussed in this section), fitness(X) is the fitness of solution $X = \{x_i : i = 1, \ldots, 124\}$ and $N(0,1)$ is a standard normal random variable.

Figures 7.3 and 7.4 show the online and offline performance of the GA and EP run, for the target dialogues DIAL 1 and DIAL 2 respectively.

Figure 7.3: Online and offline performance for a trial of the GA and EP with DIAL 1 as the target dialogue. The fitness function which takes into account LOLITA's additional information was used.

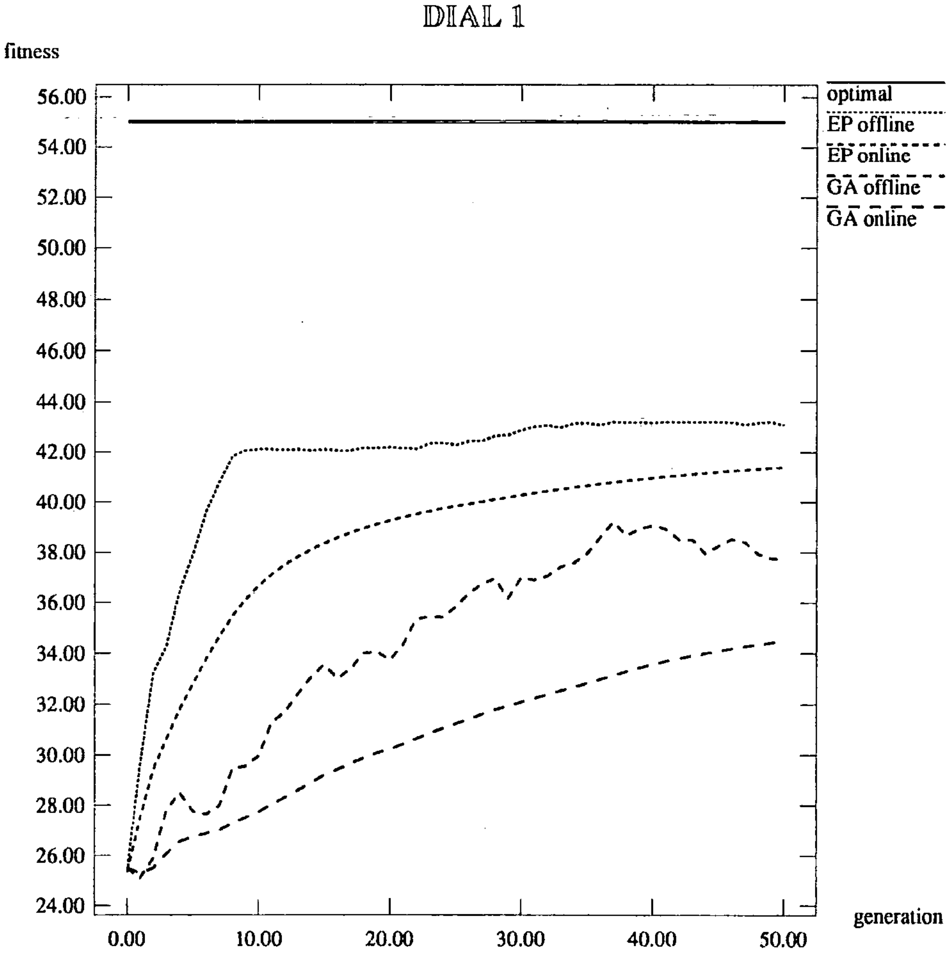
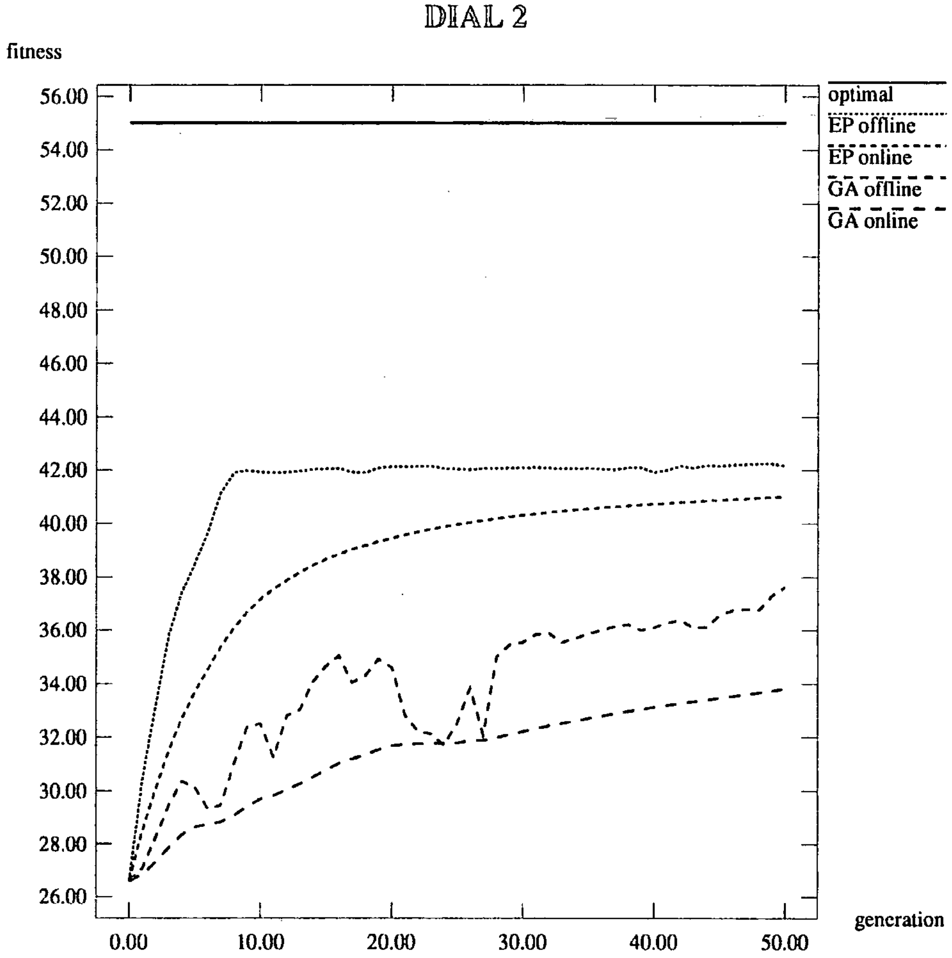


Figure 7.4: Online and offline performance for a trial of the GA and EP with DIAL 2 as the target dialogue. The fitness function which takes into account LOLITA's additional information was used.



Additional information	DIAL 1		DIAL 2	
	GA	EP	GA	EP
utterance	9	9	8	7
local goal	10	10	8	9
subgoals	11	11	11	10
utterance types	9	9	8	9
action types	8	8	8	8
Fitness	47	47	43	43

Table 7.5: Decomposition of the results achieved with the improved fitness function. The optimum value for each of the values is 11.

In the case of DIAL 1 the GA was able to find a solution with a fitness of 47 by generation 15, and EP a solution of fitness 47 by generation 8. For DIAL 2 the GA discovered a solution of fitness 43 by generation 7, and EP a solution of fitness 43 by generation 14. The breakdown of these results is shown in Table 7.5.

Again the exact matching of utterances resulted in statements such as "LOLITA: Why do you like me?" in place of "LOLITA: I know; why do you like me?" being scored as incorrect. Similar instances arose with the matching of the additional information. For example, if the utterance types are "Statement, Noise, Question" then "Statement, Question" is currently scored as incorrect. A fitness value of 0.666 would be more appropriate. There is clearly much scope for improvement in the discriminatory power of the fitness function.

7.9 Discussion

The results show that both a GA and EP were reasonably successful at the dialogue optimisation problem presented. The problem is an example of one for which the subsymbolic representation adopted has strong interactions between blocks of subsymbolic components, but weak interactions between the components of any particular block. These results, although preliminary, do lead to some interesting points worthy of further consideration.

For both the GA and EP the average fitness of solutions in subsequent generations steadily improved. No attempt was made to tune the settings of the evolutionary algorithms themselves. In the case of the GA such settings include the crossover and mutation probabilities. Other components of the GA that may be altered include the solution representation (*e.g.*, integers), crossover type and selection mechanism. The performance of EP may be improved by altering the tournament size, or the formula controlling the amount of mutation. Furthermore, it is likely that by increasing the population and generation size improved results can be expected. This has not been studied to date since with a population and generation size of 100, the runtime (on a Sparc4 workstation) can be expected to be of the order of two days. Furthermore, evaluating any differences in performance is difficult since the underlying system is continually being modified.

For the dialogues and fitness functions considered the fact that both a GA and EP are able to discover solutions which perform well indicates that both a bottom-up and a top-down approach is a suitable means of solution construction.

The discriminatory power of the fitness function needs to be further improved. Ideally some quantitative measure of semantic distance would be used (Short *et al.* 1994a, 1994b). This would entail finding some quantitative measure for the similarity of the meaning of two sentences. Another approach would involve making better use of the information that the LOLITA system is capable of producing. With an improved fitness function the current limitation of having to apply the EAs to known dialogues can be removed. Evolving the plan box parameters so that the resulting dialogue exhibits a certain personality is the long term aim, *e.g.*, finding the parameters which result in LOLITA becoming easily offended. Once sets of parameters for different behaviours have been determined they can be used to run LOLITA with that 'personality'.

7.10 Summary

This chapter provides evidence that a hybrid symbolic/subsymbolic approach can be successfully applied within the dialogue module of a large scale natural language processor. Adopting such an approach allows the dialogue module to enjoy many of the advantages of a well constructed theory, while at the same time allowing for the flexibility which a subsymbolic approach is capable of providing. The complex dialogues which can be generated validate the approach.

Evolutionary algorithms have been applied to the problem of searching the space of the subsymbolic representation so that a solution which exhibits a certain behaviour can be found. The interactions between blocks of the subsymbolic components are strong, but those between individual components of a block are weak, *i.e.*, the subsymbolic representation displays Strong-Weak interactions (see Section 1.3). For the dialogues and fitness functions considered both a GA and EP were able to overcome the interactions which may occur and construct solutions that perform well. A more general application of the approach is currently limited by the poor discriminatory power of the fitness function.

Chapter 8

Evolutionary Algorithms and Speech Recognition

This chapter introduces the third and final example of a problem for which a hybrid symbolic/subsymbolic approach is successful, and as such provides further evidence for the 'width' of its application. As with the two problems considered previously the approach taken to solving is to initially adopt a symbolic approach and then move to a subsymbolic representation to allow for greater flexibility. This problem is an example of one for which there are strong interactions between the subsymbolic components. The problem considered is from the field of speech recognition and involves trying to improve the word matching stage of the AURAID system — a speech recognition aid for use by deaf students in lectures which is currently being developed at the University of Durham (Collingham 1994).

A word lattice is a symbolic data structure which can be used to contain the word hypotheses generated by the first stage of a speech recognition system. Several subsymbolic approaches have been suggested as a means of representing the likelihood of a particular word occuring at a particular point, *e.g.*, assigning probabilities. AURAID uses a dynamic programming algorithm to score the words in the lattice — they are then ranked according to their score. The algorithm contains several penalties, which are represented subsymbolically. These penalties can be selected by hand, but there are strong interactions between the subsymbolic components and an automatic approach would be desirable. This chapter examines how a GA and EP can be applied to the problem of penalty determination and compares their performance. EP is then used to optimise the parameters for several small data sets and the parameters are shown to be robust when applied to a large set of unseen data (Collingham 1994; Nettleton and Collingham 1995).

8.1 Spoken language understanding systems

As improvements have been made in automatic speech recognition the assessment tasks have become progressively more challenging (ARPA 1994). Systems capable of continuous speech digit recognition, or isolated word recognition systems, are no longer sufficient. A generic speech recognition system must be able to cope with real spontaneous speech, very large vocabularies and be domain independent. Modern day systems are striving towards this goal. A further challenge for these systems is that not only should they be capable of word recognition, but they should be able to understand what is being said and act upon it. In other words go beyond the task of speech recognition, and attempt to understand spoken language.

A spoken language understanding system is one which integrates a speech recognition system with a text-based natural language (NL) understanding system. The following are methods which may be used to produce an interface between the two systems:

N-Best : the speech recogniser passes a list of the N best sentence hypotheses to the NL understanding system for analysis — a typical value for N would be 10. The NL system then analyses these sentences to determine the most likely recognition. The analysis typically includes making use of semantics and pragmatics.

Multiple Knowledge Sources : the speech recogniser makes use of multiple knowledge sources. Some of these can be expected to be provided by the NL system, *e.g.*, semantics and pragmatics. Others are provided by the speech recogniser, *e.g.*, a bigram language model. During recognition each sentence hypothesis is passed to each knowledge source for assessment.

Summary : the output of the speech recogniser is passed directly to the NL system for analysis. The NL systems attempts to parse and/or semantically and pragmatically analyse the recogniser output. The NL system then outputs a 'tidied' up version, or summary, of the recogniser output. This method is particularly suitable for the recognition of spontaneous speech.

An appropriate data structure that may be built prior to generating sentence hypotheses is a word lattice (Murveit *et al.* 1993; Baggia *et al.* 1992; Ljolje and Riley 1992). A word lattice is a symbolic structure that contains the set of word hypotheses produced at the acoustic matching stage. Each word hypothesis is characterised by the start and end points of the spoken utterance portion against which it has been matched, and a score representing its acoustic likelihood. The word lattice contains many more word hypotheses than the number of actual spoken words, and word hypotheses may overlap one another. A simplified example of a word lattice is shown in Table 8.1.

The typical method of determining the acoustic likelihood of a word's pronunciation involves collecting a corpus of recorded speech for a particular domain. A subsymbolic representation is then adopted which assigns to each word, or subword (*i.e.*, phoneme), the probability of it being spoken. One disadvantage of this approach is that the likelihood scores need to be re-calculated for each new domain, and this involves collecting a new corpus of recorded speech. In addition, it is unlikely that the acoustic models generated will be robust enough for vocabulary and domain independence. In order to overcome this a dynamic programming algorithm is used to calculate a word's acoustic likelihood. The algorithm used in

<table>
<tr><td>spoken input</td><td colspan="2">this</td><td colspan="2">course</td><td colspan="2">is</td><td>on</td><td colspan="7">software</td><td colspan="9">maintenance</td></tr>
<tr><td>spoken phoneme form</td><td colspan="2">D I s</td><td colspan="2">k O s</td><td colspan="2">I z</td><td>Q n</td><td colspan="7">s Q f t w e@ r</td><td colspan="9">m eI n t @ n @ n s</td></tr>
<tr><td>recognised phoneme form</td><td>D</td><td>s</td><td>k</td><td>O</td><td>I</td><td>z</td><td>Q</td><td>s</td><td>Q</td><td>f</td><td>t</td><td>l</td><td>e@</td><td>r</td><td>m</td><td>eI</td><td>n</td><td>t</td><td>@</td><td>n</td><td>n</td><td>@</td><td>s</td></tr>
<tr><td rowspan="5">word
lattice</td><td colspan="2">this</td><td colspan="2">course</td><td colspan="2">is</td><td>on</td><td colspan="7">software</td><td colspan="9">maintenance</td></tr>
<tr><td>earth</td><td colspan="2">ask</td><td></td><td colspan="2">us</td><td colspan="2">loss</td><td colspan="2">off</td><td colspan="2">tell</td><td></td><td colspan="2">room</td><td colspan="2">an</td><td colspan="2">to</td><td colspan="2">known</td><td colspan="2">as</td></tr>
<tr><td colspan="2">these</td><td colspan="2">call</td><td colspan="3"></td><td colspan="2">saw</td><td colspan="2"></td><td colspan="2">law</td><td></td><td colspan="2">may</td><td></td><td colspan="3">ten</td><td colspan="3">nice</td></tr>
<tr><td colspan="2"></td><td colspan="3">carry</td><td colspan="2"></td><td colspan="4">soft</td><td></td><td colspan="2">air</td><td colspan="3">main</td><td colspan="6"></td></tr>
<tr><td colspan="2"></td><td colspan="4">courses</td><td colspan="8"></td><td colspan="4">meant</td><td colspan="5"></td></tr>
</table>

Table 8.1: A simplified example of a word lattice

this work contains four acoustic parameters. The parameters can be represented subsymbolically (real-valued) and the settings selected by hand (Collingham and Garigliano 1993).

The approach outlined in this chapter is to use evolutionary algorithms to search the space of the parameters' subsymbolic representation, and so automatically generate the required acoustic parameters for word lattice generation. The evolutionary algorithms discover a near-optimum solution for a small set of data (113 words input and a 1984 word dictionary). These parameters give encouraging results on a larger unseen set of data (5057 words from the LUND corpus (Svartvik 1992) and a 2637 word dictionary) and shows that the parameters are robust enough to withstand changes in vocabulary and domain. In fact the only dependence is on the performance of the underlying continuous speech phoneme recognition system. Should this be improved, then the evolutionary algorithm may be re-run to automatically generate a new set of parameters.

8.2 The AURAID System

The AURAID system is a speech recognition aid for use by deaf students in lectures which is currently being developed at the University of Durham. A domain independent syntactic sub-system is used for word recognition from a continuous sequence of phonemes. The front end processing is to be performed by the AURIX continuous speech phoneme recognition system developed by the DRA (Russell 1992). The dynamic programming stage matches the phoneme input with a dictionary to produce a word lattice. The parsing stage makes use of an 'anti-grammar' in order to determine the best sequence of words through the lattice (Collingham and Garigliano 1993). AURAID uses a vocabulary of up to 2637 words and works in real-time using a simulated front end. Experiments are now taking place making use of the *Aurix* front end, which is currently only a prototype. In addition work is in progress to integrate fully with the LOLITA natural language processing system using all three of the interfacing methods described in the introduction, to provide a complete spoken language understanding system.

8.3 Data Preparation

Several lectures from different courses on various aspects of software engineering were recorded on to audio cassette. The text of these lectures was typed into a computer as accurately as possible and included partial words and filled pauses such as, "ums" and "errs," together with an indication of the location of short and long pauses. The phoneme representation for each word in each lecture was obtained from the *Oxford Advanced Learner's Dictionary* (Mitton 1986). The phoneme representation of each lecture was then corrupted in order to accurately reproduce the performance of a continuous phoneme recognition system by using real data figures obtained during the assessment of Armada (Browning *et al.* 1990) which is the forerunner for *Aurix*. These corrupted phoneme lecture files form the basis of the simulation and files containing approximately 20%, 30% and 40% phoneme error rate were produced.

8.4 Word Lattice Generation

A word lattice is a symbolic data structure which holds detailed information resulting from the lexical matching (word hypothesis) routine of a speech recognition system. Informally, each word of a dictionary is compared with acoustic/phonetic data. Each word is assigned a score indicating how closely it matches a particular portion of data. Paths may be traced (parsed) through the word lattice by joining up words that span consecutive portions of data to form sentence hypotheses.

Dynamic programming is used to match each word in the dictionary with a series of phonemes in order to build a lattice of spoken word hypotheses. Dynamic programming is a mathematical concept that has been used for many years for multistage optimal decision calculation. In the field of speech recognition (where it is also known as dynamic time warping) it was used initially in isolated word recognition systems for comparison of segments of speech with stored word templates. This was extended to continuous word recognition by storing each template as a series of frames which were then compared to the segments of speech. A detailed description of dynamic programming for speech recognition is given by Silverman and Morgan (1990). By assuming a continuous stream of phonemes as its input, AURAID does not deal with frames or segments of the speech signal. However, dynamic programming can be used to match stored template words, made up of a series of phonemes, with the input phonemes.

There are three main approaches which use dynamic programming in continuous speech recognition: the two level algorithm (Sakoe 1979), the level building algorithm (Myers and Rabiner 1981), and the one pass algorithm (Bridle *et al.* 1982). Although each differs in detail, the two basic stages involved in each algorithm are word level analysis and phrase level analysis. In word level analysis, each word in the dictionary is matched against all possible (consecutive) sequences of the input phonemes. Phrase level analysis determines the best scoring sequence of words that spans the entire phoneme input. These two stages comprise the two level algorithm, the others being optimisations which integrate the two stages.

In AURAID a word level analysis using dynamic programming is undertaken, and then a beam search used for the phrase level analysis. The word level analysis algorithm models explicitly the kinds of errors which may occur, both within words and between words. That is inserted phonemes, deleted phonemes, substituted phonemes and word final phoneme deletion are considered. The distance or similarity score between phonemes can depend on a variety of factors, and varies from algorithm to algorithm. Most algorithms group phonemes into classes according to their confusability. The phoneme classes used by AURAID are based on manner of articulation and are shown in Table 8.2. The distance between phonemes within the same class is then less than that between phonemes from different classes. This can be measured, for example, by absolute values or logarithms of the probability of confusing one phoneme for another based on experimental data. Collingham (1994) found that long words were unduly penalised because of their length and were not recognised as well as they should be. To overcome this inadequacy the distance scores are normalised according to the length of the word being considered. One of the base case equations used in the word level analysis algorithm is:

Class	*Name*	*Phonemes*
0	Plosive	`p b t d k g`
1	Affricative	`tS dZ`
2	Strong Fricative	`s z S Z`
3	Weak Fricative	`f v T D h`
4	Liquid/Glide	`l r w j`
5	Nasal	`n m N`

6	Vowel	`i I E { A Q O U u 3 V @`
		`aI eI oI aU @U I@ e@ U@`

Table 8.2: Phoneme classes used by AURAID

$$\begin{aligned} S(w,1,t) = \min\{ & \frac{\mathit{ins\_pen}}{N(w)} + \frac{\mathit{sub\_pen}(w,1,t)}{N(w)} + \min_{r\in R}\{S(r,N(r),t-2)\}; \\ & \frac{\mathit{sub\_pen}(w,1,t)}{N(w)} + \min_{r\in R}\{S(r,N(r),t-1)\}; \\ & \frac{\mathit{del\_pen}}{N(w)} + \frac{\mathit{sub\_pen}(w,1,t)}{N(w)} + \min_{r\in R}\{S(r,N(r)-1,t-1)\}; \\ & \frac{2.0\times \mathit{del\_pen}}{N(w)} + \frac{\mathit{sub\_pen}(w,1,t)}{N(w)} + \min_{r\in R}\{S(r,N(r)-2,t-2)\}\ \}. \end{aligned} \tag{8.1}$$

Other base cases are used for $p = 2$ and $p = 3$ (Collingham 1994) and the general equation is:

$$\begin{aligned} S(w,p,t) = \min\{ & \frac{\mathit{ins\_pen}}{N(w)} + \frac{\mathit{sub\_pen}(w,p,t)}{N(w)} + S(w,p-1,t-2); \\ & \frac{\mathit{sub\_pen}(w,p,t)}{N(w)} + S(w,p-1,t-1); \\ & \frac{\mathit{del\_pen}}{N(w)} + \frac{\mathit{sub\_pen}(w,p,t)}{N(w)} + S(w,p-2,t-1); \\ & \frac{2.0\times \mathit{del\_pen}}{N(w)} + \frac{\mathit{sub\_pen}(w,p,t)}{N(w)} + S(w,p-3,t-2)\ \} \end{aligned} \tag{8.2}$$

where $S(w,p,t)$ represents the score for phoneme p of word w when matched against input phoneme t, R is the set of words in the dictionary used by AURAID and $N(r)$ is the length in phonemes of the r'th word. The three penalties, *ins_pen*, *del_pen* and *sub_pen* are represented subsymbolically (real-valued). Each returns a value independent of the particular phoneme being considered, with the exception of *sub_pen* which is divided into two separate cases. The first of these cases penalises phonemes in which the substitutions are of the same class. The second case allows a different penalty to be used for phonemes which are substituted with ones of a different class. There are, therefore, four penalty values to be chosen. In previous work (Collingham and Garigliano 1993) these settings have been selected by hand, and this method of phoneme distance calculation produced better results than other subsymbolic approaches, *e.g.*, using logarithms which used the probability of confusing one phoneme for another.

In both Equations 8.1 and 8.2, a minimum score choice is to be taken between: the last input phoneme being an insertion error; the current input phoneme being correct or a substitution error; a deletion of the previous phoneme of the current word or the final phoneme of the previous best scoring word. In addition, the last line of each equation represents the occurrence of two consecutive deletion errors. Consecutive insertion errors are not modelled because they are not produced by the simulated phoneme recogniser (if required a simple extension to the equations could model this). Finally, for each input phoneme the end score for each word is adjusted to represent the local score for that word if it were to end at that point in the input.

Given a string of phonemes and a dictionary, the application of the above dynamic programming algorithm results in a data structure called a word lattice, a simplified example of which is shown in Table 8.1. The position of the words on different levels in this simplified lattice is not too significant, in reality each word in a box would have associated with it a score representing how well it matches the phonemes spanned by the box. In addition to the correct path through the lattice several other paths can be traversed from the beginning to the end. For example, "this courses loss off tell air main to known as", or "these call us on soft law room an ten nice" (Table 8.1). A phrase level analysis stage determines the best path according to some criteria, and the text output from this stage is the the system's hypothesis as to what was spoken (Collingham 1994). The work of this chapter is not directly concerned with the phrase level analysis. However, as the output of the word level analysis is the input to the phrase level analysis, it is important that the word level analysis be as accurate as possible. This then reduces the number of errors which may be propagated through the system.

To date the subsymbolic components which correspond to the penalties for phoneme insertion, deletion, and the two for substitution have been selected by hand. The use of EAs as an automatic approach to searching the space of the subsymbolic representation is now discussed.

8.5 AURAID and EAs

The environment is a continuous stream of phonemes from which words are extracted producing a word lattice according to the algorithms discussed above. To create the phoneme string a piece of text consisting of 113 words was converted to phonemes and corrupted by approximately 20%, 30% and 40% to create the data files `corrupt20`, `corrupt30` and `corrupt40` respectively. The dictionary used in the dynamic programming stage contained 1984 words.

The penalties *ins_pen*, *del_pen* and both *sub_pens* are represented numerically and are constrained to be in the range [1,256]. In applying EP it isn't necessary to restrict the range, but this was done in order to allow for comparison with a GA which uses a fixed length binary subsymbolic encoding.

A fitness measure is needed in order to determine the performance of the penalties used in the dynamic programming algorithm. For each input phoneme, dynamic programming is used to calculated a score for each dictionary word on the assumption that it ends at that phoneme. These words are then ranked according to their score, a rank of 1 being the best. During penalty optimisation, the end point of each correct word in the corrupted input phoneme sequence is known (this is not the case during actual recognition). The ranks of the correct words in their correct position are found and their average value used as the fitness value. The optimisation process is aimed at minimising the fitness value with a fitness of 1 being the optimum (although this may not be attainable).

8.6 Results

This section presents the results of applying a GA and EP to the problem of estimating the penalties of Equations 8.1 and 8.2. Other than variations in solution representation and the details of the EP's mutation operator (discussed below), the GA and EP used are identical to those described in Sections 5.2 and 5.3 respectively.

In implementing the GA, the subsymbolic representation adopted is that of a binary string. Each of the penalties is encoded as a binary string of length eight, and these are concatenated together to form one string. Since there are four penalties to be encoded the size of the subsymbolic representation's search space is $256^4 \approx 4 \times 10^9$.

In applying EP to the penalty optimisation problem a real-valued subsymbolic representation is adopted. Each of the penalties are stored as real numbers (six decimal places), and are constrained to the range [1,256]. A child is produced from a parent by mutating each parameter x_i $(i = 1, \ldots, 4)$ according to:

$$x'_i = x_i + \sqrt{\text{fitness}(X)} \cdot N(0,1) \qquad i \in \{1,2,3,4\}$$

where $\text{fitness}(X)$ is the fitness of solution $X = \{x_i : i = 1, \ldots, 4\}$ and $N(0,1)$ is a standard normal random variable. The above formula was selected since it allows for solutions with a poor fitness to be mutated by a large amount, while at the same time reducing the chance that the mutated subsymbolic components fall outside of the permitted range [1,256]. Should a mutation result in a component falling outside of this range then it is set to the nearest allowable value.

For both the GA and EP a population of 50 was used, and they were executed over 50 generations. The tournament size for EP was set at three. For each of the GA and EP, 11 trials were carried out using `corrupt20`, and 31 trials for each of `corrupt30` and `corrupt40`. The fitness of the best solution found in each of the runs is shown in Table 8.3 together with the generation at which the best solution was discovered (in parenthesis). The mean and standard deviation of each set of results is also given.

The Figures 8.1, 8.2 and 8.3 each show the online and offline performance of the median run of the GA and EP for the data `corrupt20`, `corrupt30` and `corrupt40` respectively. The offline performance is the average fitness of all of the solutions in a particular generation, while the online performance is the average fitness of all solutions that have been generated up to a certain generation.

The results of the trials conducted with `corrupt20` showed that in each trial both the GA and EP found optimal or near optimal solutions. No difference in performance was observed.

A comparison of the performance of the GA and EP for `corrupt30` indicate that EP outperformed the GA. The result was not statistically significant ($t = 1.04$ with $DF = 52$ gave $P > 0.1$) unless the EP outlier (2.3) and the GA outlier (2.0) were removed ($t = 2.36$ with $DF = 56$ gave $P < 0.05$).

With `corrupt40` the results obtained showed that EP outperformed the GA. The result was not statistically significant ($t = 1.31$ with $DF = 59$ gave $P > 0.1$) unless the EP outlier (2.9) was removed ($t = 2.17$ with $DF = 54$ gave $P < 0.05$).

Table 8.3: The best solutions found by each the GA and EP for various levels of phoneme corruption. Each algorithm was run 31 times (except for the data file `corrupt20` which was run 11 times) and the generation at which the best solution was found is shown in parenthesis.

Corruption	20%		30%		40%	
Algorithm	EP	GA	EP	GA	EP	GA
Fitness of best solution found	1.1 (0)	1.1 (0)	1.4 (21)	1.4 (42)	2.3 (22)	2.6 (25)
	1.0 (30)	1.0 (2)	1.4 (38)	1.5 (3)	2.5 (44)	2.4 (6)
	1.1 (0)	1.1 (0)	1.6 (23)	1.5 (18)	2.4 (17)	2.4 (26)
	1.1 (0)	1.1 (0)	1.6 (48)	1.4 (15)	2.6 (30)	2.2 (2)
	1.1 (0)	1.0 (27)	1.5 (37)	1.6 (8)	2.4 (14)	2.4 (8)
	1.1 (0)	1.0 (11)	1.4 (34)	1.5 (15)	2.9 (48)	2.3 (1)
	1.0 (5)	1.1 (0)	1.4 (11)	1.5 (28)	2.3 (35)	2.4 (18)
	1.0 (4)	1.1 (0)	1.5 (16)	1.6 (5)	2.3 (8)	2.4 (19)
	1.0 (42)	1.0 (6)	1.5 (9)	1.5 (6)	2.4 (49)	2.6 (4)
	1.0 (4)	1.0 (21)	1.4 (8)	1.5 (5)	2.4 (8)	2.4 (24)
	1.0 (12)	1.0 (3)	1.6 (25)	1.5 (6)	2.3 (9)	2.4 (26)
			2.3 (28)	1.5 (9)	2.3 (47)	2.6 (1)
			1.4 (30)	1.4 (19)	2.3 (25)	2.1 (16)
			1.5 (39)	1.7 (8)	2.3 (15)	2.2 (2)
			1.4 (49)	1.6 (8)	2.3 (30)	2.6 (0)
			1.4 (15)	1.4 (14)	2.2 (0)	2.2 (0)
			1.4 (49)	1.5 (16)	2.3 (13)	2.2 (14)
			1.5 (46)	1.5 (32)	2.3 (24)	2.4 (12)
			1.5 (0)	1.5 (0)	2.3 (10)	2.5 (17)
			1.4 (8)	1.5 (9)	2.3 (7)	2.4 (1)
			1.5 (17)	1.6 (0)	2.4 (36)	2.4 (11)
			1.7 (26)	2.0 (7)	2.1 (49)	2.2 (37)
			1.4 (6)	1.7 (0)	2.3 (38)	2.4 (9)
			1.4 (40)	1.5 (12)	2.3 (37)	2.4 (18)
			1.4 (7)	1.5 (1)	2.2 (34)	2.4 (9)
			1.6 (44)	1.6 (6)	2.2 (44)	2.4 (39)
			1.4 (20)	1.5 (33)	2.4 (31)	2.4 (21)
			1.4 (1)	1.5 (0)	2.3 (19)	2.5 (10)
			1.6 (13)	1.6 (10)	2.3 (16)	2.4 (19)
			1.5 (21)	1.5 (4)	2.4 (10)	2.6 (11)
			1.4 (14)	1.5 (8)	2.4 (15)	2.3 (20)
Mean (2 d.p.)	1.05	1.05	1.50	1.54	2.35	2.39
SD (3 d.p.)	0.052	0.052	0.172	0.114	0.139	0.133

Figure 8.1: Online and offline performance for the median trial of the GA and EP with the data file corrupt20.

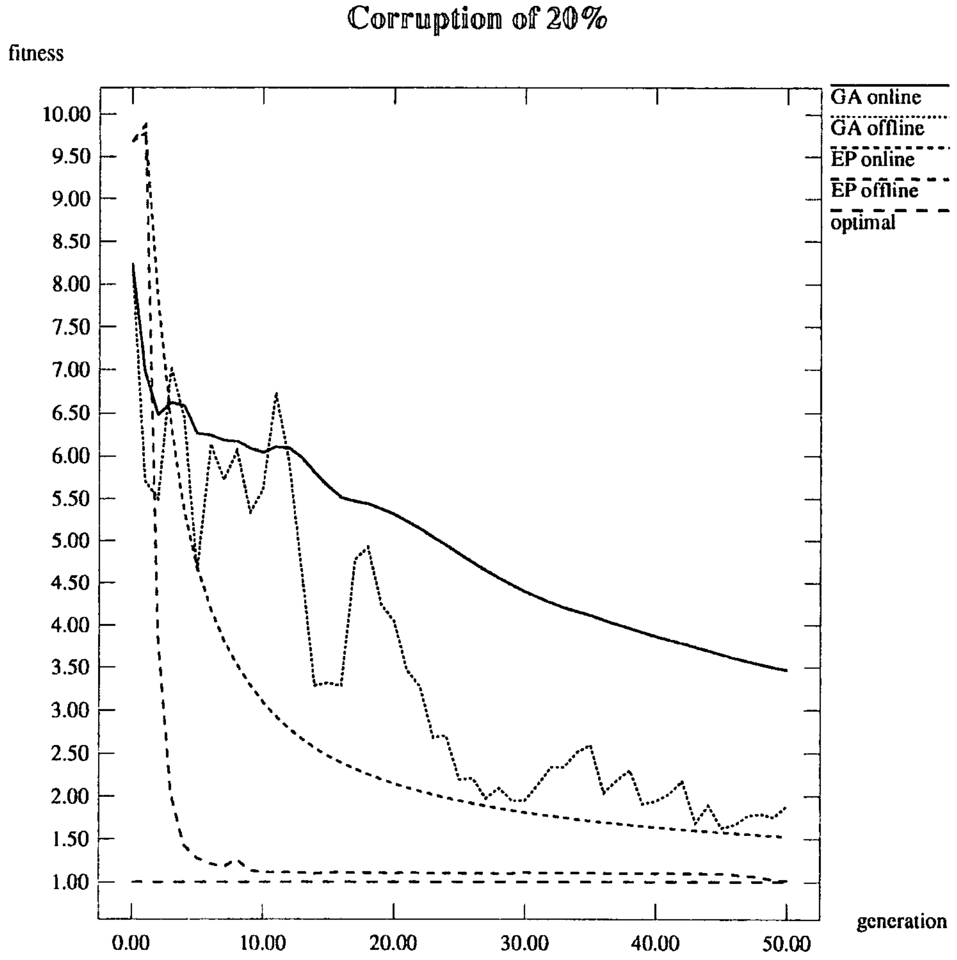


Figure 8.2: Online and offline performance for the median trial of the GA and EP with the data file corrupt30.

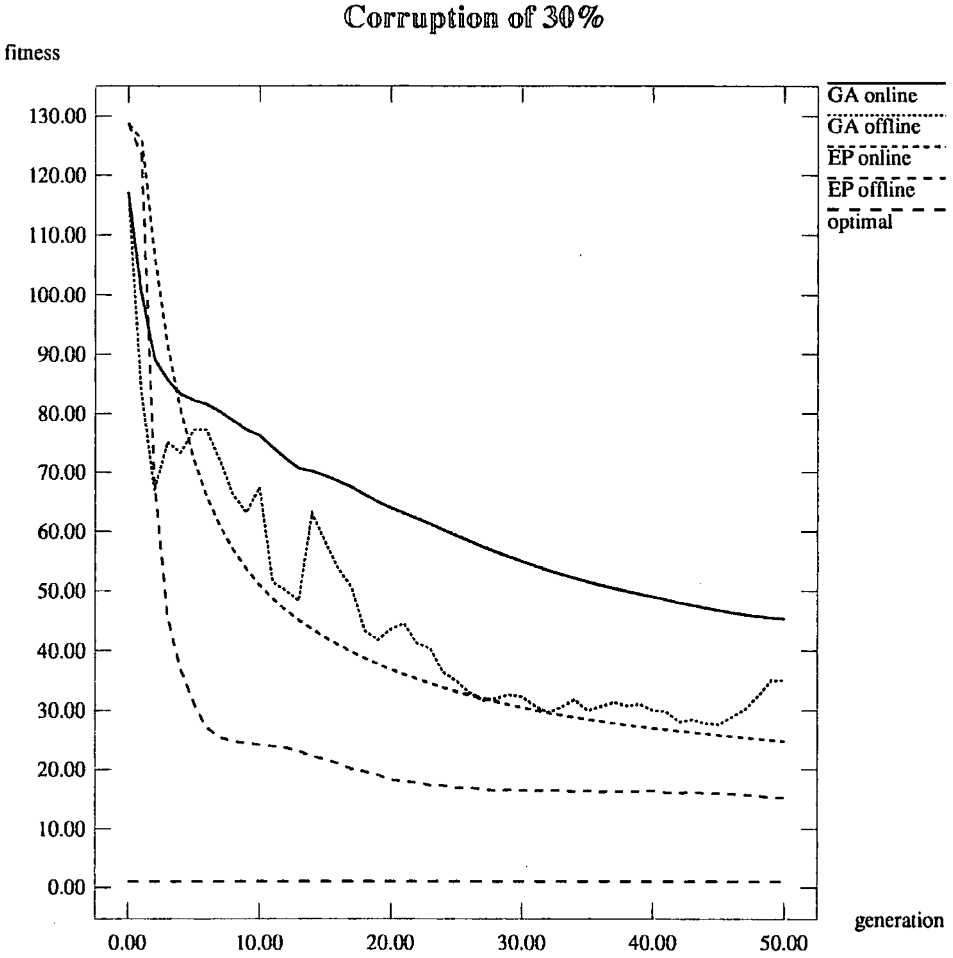
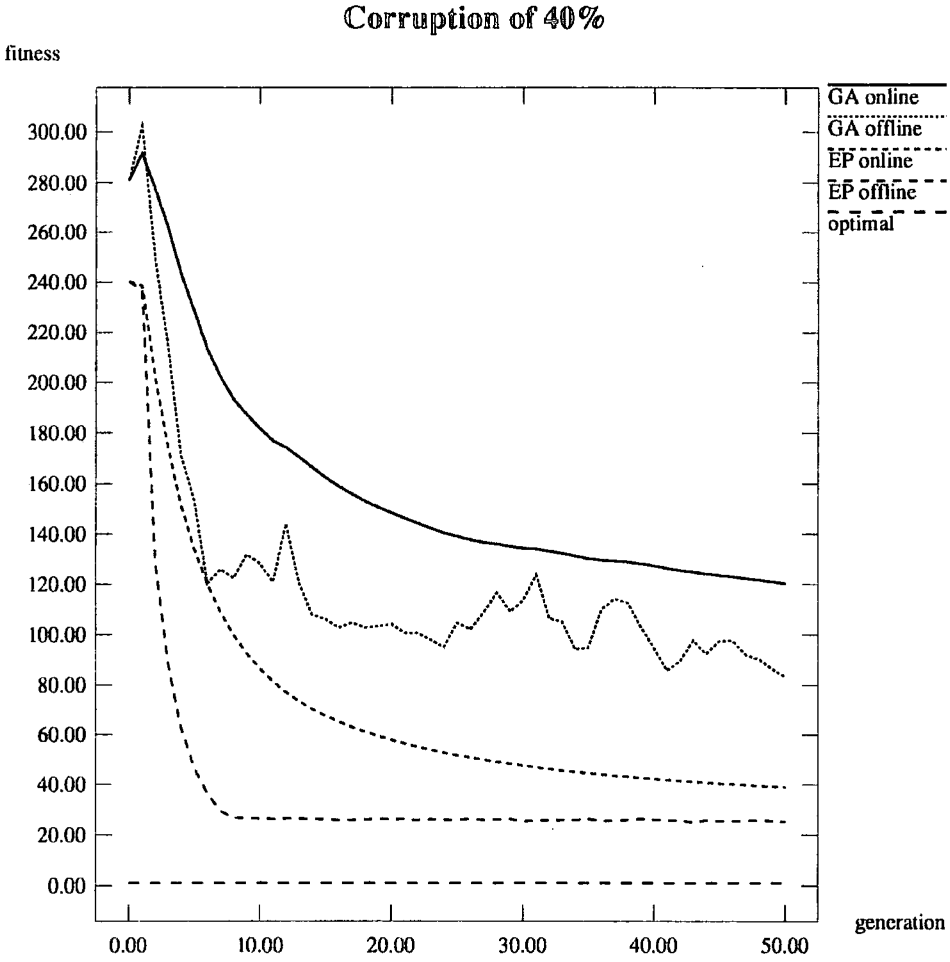


Figure 8.3: Online and offline performance for the median trial of the GA and EP with the data file `corrupt40`.



8.7 Discussion

With `corrupt20` the problem appears to be easy solvable. Solutions with a near optimum fitness were often found in the initial randomly generated generation and a wide range of parameter settings were able to produce near optimum results. This indicates that, for this low level of corruption, the search space contains large 'plateaus' of near optimal solutions. With such a low level of corruption the amount of interaction between the subsymbolic components is low and the problem although solvable using EAs could also be tackled by a range of other methods, *e.g*, hill-climbing and random search.

As the level of corruption is increased the structure of the search space changes. The degree of interaction between the subsymbolic components increases, and differences emerge in the relative performance of a GA and EP. At 30% and 40% levels of corruption, EP outperforms a GA. The reason for this appears to be that the search spaces contains fewer optimum solutions (than for 20% corruption), and that by emphasising a top-down approach EP is able to overcome the strong interactions which can occur between components.

The problem considered in the previous section used a data set of 113 words corrupted to varying degrees and a dictionary of 1984 words. The results presented show that EP outperformed the GA. Collingham (1994), and Nettleton and Collingham (1995) have applied EP to larger problems and some of these results are now discussed.

Two files of 113 and 112 words were converted to phonemes and corrupted by 25.6% (`file1`) and 26.0% (`file2`) respectively. A dictionary containing 1984 words was used by the dynamic programming algorithm to match the phoneme input in the construction of a word lattice for each of these data sets. The fitness function applied used the average rank of the words in the two lattices. EP (as described above) with a population of 100 was executed over 100 generations using a tournament size of five. The best solution found had the following settings (to

1 d.p.) for the acoustic parameters: *ins_pen* = 96.7, *del_pen* = 95.1, and the two *sub_pens* 94.7 (same class) and 214.8 (different class). The average rank of the words in the lattice generated for `file1` and `file2` was 1.7 and 2.0 respectively.

To demonstrate the robustness of the parameters with regards to the vocabulary, the size of the dictionary was increased by almost 33% from 1984 to 2637 words. Using the above parameters the average rank of the words in the lattice generated for `file1` and `file2` was 1.7 and 2.2 respectively. A further demonstration of robustness concerning domain independence was also considered. A passage from the LUND corpus (Svartvik 1992) which contained 5057 words was corrupted by 25%, and a 2637 word dictionary used in the construction of the lattice. Using the above parameters the dynamic programming algorithm created a word lattice with an average word rank for the correct words of 2.2.

With the parameters generated Collingham (1994) demonstrated that for the passage from the LUND corpus and a 25% phoneme error rate the word recognition phase of AURAID is capable of almost 70% success. The results were in all cases an improvement on those achieved when the parameters for the dynamic programming algorithm were set by hand (Collingham and Garigliano 1993).

8.8 Summary

A word-lattice is a symbolic data structure which may be used at the word matching stage of a speech recognition system. The AURAID system uses a dynamic programming algorithm for word level analysis in order that ranks can be assigned to words in a lattice. The dynamic programming algorithm used contains four penalties which can be represented subsymbolically (real-valued). There are strong interactions between the subsymbolic components, and, although their values can be set by hand, an automatic means of selection is prefered. A GA and EP can be used to automatically select the penalties and EP was used to improve the performance of a speech recognition system. The results of this chapter provides further evidence of the success of a hybrid symbolic/subsymbolic approach.

Chapter 9

Conclusion

The first part of this chapter examines if this thesis has satisfied the criteria for success which were laid out in Chapter 1. The thesis concludes with some possible directions for future research.

In order to determine the success of an approach to problem solving which uses a combination of symbolic and subsymbolic methods, several open research problems were examined. The examples taken were from fields of AI which have been researched in the Department of Computer Science at the University of Durham: shape representation, natural language processing, and speech recognition. In tackling these problems all benefited from a hybrid symbolic/subsymbolic approach.

The shape representation problem was examined in great detail so that the 'depth' of the approach could be demonstrated. The use of IFSs for shape representation allows for the results of a comprehensive theory to be brought to bear on the problem. In particular a result of the theory proves that an IFS can always be found for an arbitrary shape, and the Collage Theorem provides a means by which one may be found. Adopting a subsymbolic approach allows for a flexible approach to primitive selection. Furthermore, properties of the subsymbolic representation adopted can be used to improve the search process by reducing the search space.

The problems from natural language processing and speech recognition were discussed in far less detail, but being from completely different fields help to show the 'width' of a hybrid symbolic/subsymbolic approach. The use of a symbolic approach to natural language dialogue analysis allows a range of well constructed rules to be applied to certain situations, *e.g.*, to be polite at an interview. However, when a number of responses are suitable they need to be ordered in a way which reflects their appropriateness — a subsymbolic representation offers this versatility. In speech recognition a word lattice provides a symbolic data structure in which words hypotheses are stored. A dynamic programming algorithm is used to score the words in the lattice. For flexibility, the penalties used by the algorithm are represented subsymbolically.

For each of the problems studied evolutionary algorithms were applied to manipulate the components of a subsymbolic representation in an attempt to construct suitable solutions. Each of the problems had fundamentally different interactions between the subsymbolic components. For the shape representation problems considered EP outperformed a GA in finding solutions, and the results were statistically significant. The difference in performance was attributed to the difficulty a bottom-up (GA) approach had in overcoming the Strong-Strong interactions of a solution's representation. EP on the other hand, by emphasising a top-down approach, was less likely to be deceived by the interactions.

In tuning the parameters of the dialogue module, both a GA and EP found 'good' solutions and hence were able to overcome Strong-Weak interactions of the subsymbolic components. However, the function used to assign a fitness to a particular dialogue was very simplistic. A more sophisticated fitness function (perhaps capable of using semantics) is needed for a more general application.

For the problem in speech recognition, both a GA and EP were able to overcome the Strong interactions which could occur between the problem representation's subsymbolic components and construct suitable solutions. However, EP outperformed a GA on the more difficult problems examined, and was subsequently used to generate a solution which has improved the performance of a speech recognition system.

9.1 Research Directions

The work presented in this thesis suggests the following areas for future research:

1. The application of a hybrid symbolic/subsymbolic approach in conjunction with evolutionary algorithms to other problems. Problems which exhibit different forms of subsymbolic interaction would be of particular interest.

2. The development of a theory which can account for how symbols reduce to subsymbolic patterns.

3. Improving the absolute performance of optimisation algorithms for each of the problems considered by extensive experimentation with the many variations of algorithms which exist.

The first of the above points requires little additional explanation. The possibility of applying a hybrid symbolic/subsymbolic approach should be borne in mind when attempting to solve an optimisation problem, and that if such an approach is to be adopted then evolutionary algorithms are a possible means of subsymbolic manipulation. (The advantages of this approach are discussed in detail elsewhere in this thesis.) Each of the remaining points are now discussed in greater detail.

The symbolic and subsymbolic paradigms offer alternate approaches to the modelling of intelligence. A question which arises when two alternate methods of problem solving are suggested is: When will one perform better than the other? Luger and Stubblefield (1993, p. 693) argue that to ask such a question is often unreasonable as the two approaches are simply different models of intelligence, each of which discuss intelligence in a different language. The two approaches ask different questions, propose different answers and interpret any results differently. Indeed the use of a purely symbolic or subsymbolic approach is often to examine the approach itself and not to find the best solution to the problem. The work presented in this thesis has demonstrated that, for some problems at least, a combination of the two approaches can enjoy some of the advantages offered by each.

From a theoretical standpoint the symbolic and subsymbolic paradigms are currently incommensurable. A theory which is able to show how symbols may be reduced to patterns and how patterns equate to a symbol system would be a great achievement which would allow many developments in AI. For example: network-based perceptual and knowledge-based reasoning facilities may be incorporated together into a single agent (Luger and Stubblefield 1993, p. 694); systems which rely on an interplay between the two approaches may be provided with a theoretical framework which helps determine how they may best be combined rather than the relation depending on individual intuition. However, a theory that can link the two approaches is probably some way off.

Perhaps the clearest direction of research based upon the work of this thesis is in the improvement of the absolute performance of the evolutionary algorithms used. As indicated previously no attempt has been made to tune the algorithms to each of the problems which have been considered. In particular there are many different forms of the subsymbolic solution encoding which can be adopted and of the subsequent method of progeny generation.

A general framework in which algorithm improvement may be attempted would be: 1) make an alteration, 2) carry out a series of trials, 3) compare the results with those of the current best, 4) keep the algorithm which performs best, and 5) iterate. In order to determine if a change to an algorithm results in an increase in performance a means of comparison is required. For statistically significant comparisons at least thirty trials should be performed. In such cases the Smith-Satterthwaite modified one tailed t-test (Weiss and Hassett 1991, p. 504) which is used in all of the pairwise comparisons in this thesis can be used to test for statistically significant changes.

To specify a framework which exactly determines the alterations to the algorithms that should be considered is unnecessary since although many possibilities can be suggested only some may (when their performance is examined) be worthy of further consideration. Furthermore, there can be expected to be a high degree of interdependence among the alterations which may be considered. For example, each of several separate changes may result in a performance reduction, but together they may produce an improvement in performance. As in many other attempts at improving the performance of evolutionary algorithms the decision as to which alteration (combination of alterations) is worthy of investigation falls largely on the intuition of the experimenter. (DeJong (1985) suggests that an evolutionary algorithm could be used to optimise the parameters of an evolutionary algorithm.) There are, however, several areas which are accepted as generally being worthy of further investigation. The remainder of this section discusses some of these and their justification.

As discussed in Section 2.4 the use of a binary coding for a GA is regarded by some (*e.g.*, Holland 1992; Goldberg 1989) to be the most suitable solution representation as it maximises the implicit parallelism of the solution's encoding. However, Radcliffe (1991a) argues that a binary coding is often 'unnatural,' and conflicts with a desire to use natural representations and operators for the structures in the space being searched. A further consequence of using a binary encoding is that the magnitude of the effect of a single mutation on a binary string varies considerably with regard to where in the string the mutation occurs (this can be avoided by the use of a particular form of binary representation known as 'gray coding').

Solutions to the problems discussed in this thesis are not naturally represented as binary strings. In implementing GAs for these problems a more natural representation would be the same as that used in the implementations of EP, *i.e.*, floating point for the shape representation problem, and integer for the natural language dialogue and speech recognition problems.

A further consequence of using a binary representation for the GA is that a fixed length encoding is adopted and hence solutions are only defined to a fixed (low) degree of precision. While in the case of the shape representation problem the robustness property of IFSs indicates that further accuracy is unnecessary, this need not necessarily be the case for the other two problems. For the natural language dialogue problem in particular an increase in a solution's resolution may allow for more subtle effects to occur.

For the natural language dialogue and speech recognition problems the number of parameters required for an optimal solution is fixed in advance by the respective symbolic theory, *i.e.*, 124 for dialogue and 4 for speech. Any changes to the number of parameters used would require a change in the symbolic theory. This is not the case for the shape representation problem. In finding an IFS for an arbitrary shape a non-trivial problem is determining *a priori* the minimum number of mappings which will be required to represent that shape to some given degree of accuracy. The implementations of the evolutionary algorithms considered in this thesis require the number of mappings which are to be used to be specified in advance. For the shapes considered this could be easily done since each of the shapes was generated from a known IFS. In a more general application such information may not be available and it may be necessary to define operators which can add/remove mappings to/from an IFS. Adding new mappings does, however, lead to a massive increase in the search space and would need to be performed with care.

The operators used for progeny generation are often highly dependent upon the solution representation on which they act. In the case of a GA if a different solution representation (*i.e.*, not the binary one used in this thesis) is to be adopted then operators more suited to that representation may be used. If, for example, a floating point representation is adopted then there are many alterate crossover operators including: averaging values, and uniform crossover (Syswerda 1989). Problem specific information may also be of use in defining operators, indeed several complementary operators may be defined and used in parallel. For example, in implementing a GA for the shape representation problem a special crossover operator may be introduced which can only interchange whole mappings between IFSs. There are also many alternate schemes for progeny generation in EP. Perhaps one of the most promising is meta-EP (Fogel 1991) in which the need for scaling functions for the variance terms to be specified *a priori* is removed.

A further area in which the performance of the evolutionary algorithms may be improved is in the 'survival of the fittest' strategy which is adopted. Many methods of solution selection have been suggested including: stochastic remainder sampling without replacement, stochastic universal sampling (Baker 1987), fitness scaling, fitness windowing (Grefenstette 1984) and steady-state replacement (Syswerda 1989).

An additional means of improving the absolute performance of the evolutionary algorithms may be possible by combining them with other forms of search algorithm, *e.g.*, hill-climbing. From the above discussion it can be clearly seen that there are many alterations which may result in improvements to the performance of the evolutionary algorithms discussed in this thesis. A specific framework in which these should be considered is difficult to specify (due to possible interactions), but the above discussion has provided some general considerations.

It is clear that there is a great deal of research involved in addressing the research directions suggested at the start of this section.

Glossary

(See Giles 1990; Fogel 1992a)

Allele – An alternative form of a gene that occurs at a given site on a chromosome (locus).

Attractor – The limit point of an iterated function system.

AURAID – A speech recognition aid for use by deaf students in lectures which is currently being developed at the University of Durham.

Behaviour – The response of an organism to the present stimulus and its present state. It is the total sum of behaviours of an organism which define the fitness of the organism to its present environment and is thus the operative function against which selection operates.

Chromosome – Bodies within a cell which contain the hereditary units of genes.

Collage – The name given to any shape specific set of contraction mappings the union of which is equal to the shape itself.

Crossover – An operator on the population of a genetic algorithm which exchanges information between solutions.

DE – An abbreviation for Dialogue Element, a constituent part of a DSM, which is a factor that influences and controls the structure of the dialogue.

Dialogue – The rich interaction between two or more participants, where 'rich interaction' is taken to include features such as sub-dialogues, interruptions and complex shifts in focus.

Discourse – A set of sentences which are related to each other both linguistically and contextually (an interaction between participants is not a requirement for a discourse).

DSM – An abbreviation for Dialogue Structure Model, a schema which contains all of the information that can be expected to be relevant in a particular dialogue situation, and thus can be used to guide the generation of language to suit that situation.

EP – An abbreviation for Evolutionary Programming.

ES – An abbreviation for Evolution Strategy.

GA – An abbreviation for Genetic Algorithm.

Gene – A unit of heredity located on a chromosome and composed of DNA (deoxyribonucleic acid).

Genotype – The sum of inherited characters maintained within the entire reproducing population; often also used to refer to the genetic constitution underlying a single trait or set of traits.

IFS – An abbreviation for Iterated Function System, a set of contraction mappings on a metric space which when applied iteratively to any subset of the space always produce the same subset in the the limit (the attractor).

LOLITA – An acronym for Large scale, Object based, Linguistic Interactor, Translator and Analyser, a natural language processor which is currently being developed at the University of Durham.

MPP – An abbreviation for Minimum Point Plotting Algorithm, used for obtaining the attractor of an iterated function system.

Mutation – A genetic change.

Natural selection – The result of competitive exclusion as organisms fill the available finite resource space.

Phenotype – The behavioural expression of the genotype in a specific environment; the realised expression of the genotype; the functional expression of a trait.

Pleiotropy – The effect of a single gene affecting several phenotypic traits.

Polygeny – A single phenotypic effect being determined by the interaction of many genes.

Shape – Defined to be any compact subset of the Euclidean plane.

Species – A group of similarly constructed organisms that are capable of interbreeding and producing fertile offspring.

References

Abenda S. and Turchetti G. (1989) Inverse Problem for Fractal Sets on the Real Line via the Moment Method, *Il Nuovo Cimento*, Vol. 104 B, No. 2, pp. 213–227.

ARPA (1994) Proceedings of the ARPA Spoken Language Systems Technology Workshop.

Angeline P.J. (1993) *Evolutionary Algorithms and Emergent Intelligence*, PhD Thesis, The Ohio State University, USA.

Atmar W. (1994) Notes on the Simulation of Evolution, *IEEE Transactions on Neural Networks*, Vol. 5, No. 1, pp. 130–147.

Bäck T., Hoffmeister F. and Schwefel H. (1991) A Survey of Evolution Strategies, *Proceedings of the Fourth International Conference on Genetic Algorithms*, R.K. Belaw and L.B. Booker (eds.), pp. 2–9, Morgan Kaufmann.

Bäck T. and Schwefel H. (1993) An Overview of Evolutionary Algorithms for Parameter Optimization, *Journal of Evolutionary Computation*, Vol. 1, No. 1, pp. 1–25.

Baggia P., Gerbino E., Giachin E. and Rullent C. (1992) Real-time Linguistic Analysis for Continuous Speech Understanding, *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pp. 33–39.

Baker J.E. (1987) Reducing Bias and Inefficiency in the Selection Algorithm, *Genetic Algorithms and Their Applications: Proceedings of the Second In-*

ternational Conference on Genetic Algorithms and Their Applications, J.J. Grefenstette (ed.), pp. 14–21, Lawrence Erlbaum Associates.

Barnsley M.F. (1988) *Fractals Everywhere*, Academic Press.

Barnsley M.F. and Demko S. (1985) Iterated Function Systems and the Global Construction of Fractals, *Proceedings of the Royal Society of London A399*, pp. 243–275.

Barnsley M.F. and Hurd L.P. (1993) *Fractal Image Compression*, A.K. Peters Limited.

Barnsley M.F., Ervin V., Hardin D. and Lancaster J. (1986) Solution of an Inverse Problem for Fractals and other Sets, *Proceedings of the National Academy of Science USA*, Vol. 83, pp. 1975–1977.

Barnsley M.F. and Sloan A.D. (1988) A Better way to Compress Images, *BYTE*, January, pp. 215–233.

Beardon C. (1989) *Artificial Intelligence Terminology: A reference guide*, Ellis Horwood Limited.

Beasley D., Bull D.R. and Martin R.R. (1993a) An Overview of Genetic Algorithms: Part 1, fundamentals, *University Computing*, Vol. 15, No. 2, pp. 58–69.

Beasley D., Bull D.R. and Martin R.R. (1993b) An Overview of Genetic Algorithms: Part 2, research topics, *University Computing*, Vol. 15, No. 4, pp. 170–181.

Beaumont J.M. (1990) Advances in Block Based Fractal Coding of Still Pictures, *IEE Colloquium on the Application of Fractal Techniques in Image Processing*, London.

Belaw R.K. and Booker L.B. (eds.) (1991) *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann.

Berger M.A. (1989) Images Generated by Orbits of 2-D Markov Chains, *Chance: New Directions for Statistics and Computing*, Vol. 2, No. 2, pp. 18–28.

Beyer H. (1993) Toward a Theory of Evolution Strategies: Some Asymptotical Results from the $(1,^{+}\lambda)$-Theory, *Journal of Evolutionary Computation*, Vol. 1, No. 2, pp. 165–188.

Bridle J.S., Brown M.D. and Chamberlain R.M. (1982) An Algorithm for Connected Word Recognition, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Paris, pp. 899–902.

Browning S.R., Moore R.K., Ponting K.M. and Russell M.J. (1990) A Phonetically Motivated Analysis of the Performance of the ARM Continuous Speech Recognition System, *Proceedings of the Institute of Acoustics Autumn Conference*, Windermere, Cumbria, UK.

Calmet J. and Campbell J.A. (1993) Artificial Intelligence and Symbolic Mathematical Computations, *Proceedings of the Conference on Artificial Intelligence and Symbolic Mathematical Computations*, Lecture Notes in Computer Science, J. Calmet and J.A. Campbell (eds.), Vol. 737, pp. 1–19, Springer-Verlag.

Cohen H.A. (1992) Deterministic Scanning and Hybrid Algorithms for Fast Decoding of IFS (Iterated Function System) Encoded Image Sets, *International Conference on Acoustics, Speech and Signal Processing*, Vol. III, pp. 509–512.

Collingham R.J. (1994) An Automatic Speech Recognition System for use by Deaf Students in Lectures, PhD Thesis (submitted), Department of Computer Science, University of Durham, UK.

Collingham R.J. and Garigliano R. (1993) Using Anti-grammar and Semantic Categories for the Recognition of Spontaneous Speech, *Proceedings of EUROSPEECH '93, the 3rd European Conference on Speech Communication and Technology*, Berlin, pp. 1951–1954.

Culik II K. and Dube S. (1993) Balancing Order and Chaos in Image Generation, *Journal of Computers and Graphics*, Vol. 17, No. 4, pp. 465–486.

Davis T.E. and Principe J.C. (1993) A Markov Chain Framework for the Simple Genetic Algorithm, *Journal of Evolutionary Computation*, Vol. 1, No. 3, pp. 269–288.

Dawkins R. (1986) *The Blind Watchmaker*, Clarendon Press.

De Jong (1985) Genetic Algorithms: A 10 year perspective, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette (ed.), pp. 169–177, Lawrence Erlbaum Associates.

Edgar G.A. (1990) *Measure, Topology and Fractal Geometry*, Springer-Verlag.

Fisher Y. (1992) Fractal Image Compression, *SIGGRAPH '92 Course Notes.*

Fogel D.B. (1991) *System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling*, Ginn Press.

Fogel D.B. (1992a) *Evolving Artificial Intelligence*, PhD Thesis, University of California, San Diego, USA.

Fogel D.B. (1992b) Using Evolutionary Programming for Modeling: An Ocean Acoustic Example, *IEEE Journal of Oceanic Engineering*, Vol. 17, No. 4, pp. 333–340.

Fogel D.B. (1993) On the Philosophical Differences between Evolutionary Algorithms and Genetic Algorithms, *Proceedings of the Second Annual Conference on Evolutionary Programming*, D.B. Fogel and W. Atmar (eds.), pp. 23–29.

Fogel D.B. (1994) An Introduction to Simulated Evolutionary Optimization, *IEEE Transactions on Neural Networks*, Vol. 5, No. 1, pp. 3–14.

Fogel D.B. and Atmar J.W. (1990) Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems, *Biological Cybernetics*, Vol. 63, No. 2, pp. 111—114.

Fogel D.B. and Atmar W. (eds.) (1992) *Proceedings of the 1st Annual Conference on Evolutionary Programming*, La Jolla, CA, Evolutionary Programming Society, San Diego, USA.

Fogel D.B. and Atmar W. (eds.) (1993) *Proceedings of the 2nd Annual Conference on Evolutionary Programming*, La Jolla, CA, Evolutionary Programming Society, San Diego, USA.

Fogel L.J., Owens A.J. and Walsh M.J. (1966) *Artificial Intelligence Through Simulated Evolution*, J. Wiley.

Foley J.D. and Van Dam A. (1982) *Fundamentals of Interactive Computer Graphics*, Addison Wesley.

Forrest S. (ed.) (1993) *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann.

Forrest S. and Mitchell M. (1992) Relative Building-Block Fitness and the Building-Block Hypothesis, *Foundations of Genetic Algorithms 2*, D. Whitley (ed.), pp. 109–126, Morgan Kaufmann.

Forrest S. and Mitchell M. (1993) What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation, *Machine Learning*, Vol. 13, No. 3/4, pp. 285–319.

Garigliano R., Morgan R.G. and LOLITA group (1994a) *The LOLITA Project: The First Seven Years*, under negotiation with After Hurst Limited.

Garigliano R., Morgan R.G. and Smith M.H. (1993a) The LOLITA System as a Contents Scanning Tool, *Proceedings of the Thirteenth International Conference on Artificial Intelligence*, Avignon, France.

Garigliano R., Morgan R.G. and Smith M.H. (1993b) LOLITA: Progress Report 1, *Technical Report 12/92*, Department of Computer Science, University of Durham, UK.

Garigliano R. and Nettleton D.J. (1992) Qualitative Mathematical Modelling of Genetic Algorithms, *Proceedings of the Conference on Artificial Intelligence and Symbolic Mathematical Computations*, Lecture Notes in Computer Science, J. Calmet and J.A. Campbell (eds.), Vol. 737, pp. 296–305, Springer-Verlag.

Garigliano R. and Nettleton D.J. (1994) The Interplay of Symbolic and Adaptive Techniques: Two Case Studies, *IEE Colloquium on Symbolic and Neural Cognitive Engineering*, London.

Garigliano R., Purvis A., Giles P.A. and Nettleton D.J. (1993c) Genetic Algorithms and Shape Representation, *Proceedings of the 2nd Annual Conference on Evolutionary Programming*, D.B. Fogel and W. Atmar (eds.), pp. 40–47.

Garigliano R., Purvis A., Giles P.A. and Nettleton D.J. (1993d) An Adaptive Plan, *Proceedings of the International Conference on Neural Networks and Genetic Algorithms*, Innsbruck, Austria.

Garigliano R., Tate J. and Boguraev B. (eds.) (1994b) *Journal of Natural Language Engineering*, Cambridge University Press.

Georghiades P. and Jacobs G. (1992) The Big Squeeze, *Personal Computer Magazine*, October, pp. 231–235.

Giles P.A. (1990) *Iterated Function Systems and Shape Representation*, PhD Thesis, University of Durham, UK.

Giles P.A., Purvis A., Waugh D. and Garigliano R. (1989) Iterated Function Systems and 2-D Shape Representation, *Proceedings of Fifth Alvey Vision Conference*, Reading, UK, pp. 49–53.

Gleick J. (1988) *Chaos: Making a new science*, Heinmann.

Goldberg D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.

Goldberg D.E. and Rudnick M. (1991) Genetic Algorithms and the Variance of Fitness, *Technical Report No. 90-011*, Department of Computer Science and Engineering, Oregon Graduate Institute for Science and Technology, 19600 NW von Neumann Dr., Beaverton, OR 97006-1999.

Goodman G.S. (1991) A Probabilist look at the Chaos Game, *Fractal in the Fundamental and Applied Sciences*, H.O. Peitgen, J.M. Henriques and L.F. Penedo (eds.), pp. 159–168, Elsevier.

Grefenstette J.J. (1984) GENESIS: A System for using Genetic Search Procedures, *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pp. 161–165.

Grefenstette J.J. (ed.) (1985) *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Lawrence Erlbaum Associates.

Grefenstette J.J. (ed.) (1987) *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, Lawrence Erlbaum Associates.

Hepting D., Prusinkiewicz P. and Saupe D. (1991) Rendering Methods for Iterated Function Systems, *Fractal Geometry and Analysis*, NATO ASI Series C: Mathematical and Physical Sciences, Vol. 346, J. Bélair and S. Dubuc (eds.), Kluwer.

Holland J.H. (1975) *Adaption in Natural and Artificial Systems*, University of Michigan Press.

Holland J.H. (1992) *Adaption in Natural and Artificial Systems*, MIT Press.

Hollatz S.A. (1991) Digital Image Compression with two-dimensional Affine Fractal Interpolation Functions, *Technical Report 91-2*, Department of Mathematics and Statistics, University of Minnesota-Duluth, USA.

Horn A.N. (1989) IFSs and the Interactive Design of Tiling Structures, *Proceedings of the BCS Seminar on Fractals and Chaos*, pp. 22–39, London.

Hoskins D.E. and Vagners J. (1992) Image Compression using Iterated Function Systems and Evolutionary Programming: Image Compression without Image Metrics, *26th Asilomar Conference on Signals, Systems and Computers.*

Hutchinson J.E. (1981) Fractals and Self Similarity, *Indiana University Mathematics Journal*, Vol. 30, No. 5, pp. 713–747.

Jacquin A.E. (1992) Image Coding Based on a Fractal Theory of Iterative Contractive Image Transformations, *IEEE Transactions on Image Processing*, Vol. 1, No. 1, pp. 18–30.

Jones C.E. (1994a) Dialogue Structure Models: An approach to Dialogue Analysis and Generation by Computer, PhD Thesis, Department of Computer Science, University of Durham, UK.

Jones C.E. and Garigliano R. (1993) Dialogue Analysis and Generation: A Theory for Modelling Natural English Dialogue, *Proceedings of EUROSPEECH '93, the 3rd European Conference on Speech Communication and Technology*, Berlin, pp. 951-954.

Jones T. (1994b) *A Model of Landscapes*, Department of Computer Science, Santa Fe Institute, 1660 Old Pecos Trail, Suite A., Santa Fe, NM 87505, USA.

Kirkpatrick S., Gelatt C.D. and Vecchi M.P. (1983) Optimization by Simulated Annealing, *Science*, Vol. 220, No. 4598, pp. 671–680.

Kropatsch W.G., Neuhauser M.A., Leitgeb I.J. and Bischof H. (1992) Combining Pyramidal and Fractal Image Coding, *Proceedings of the 11th International Conference on Pattern Recognition*, Vol III, pp. 61–64, IEEE Computer Society Press.

Levy-Vehel J. and Gagalowicz A. (1987) Shape Approximation by a Fractal Model, *EUROGRAPHICS '87*, G. Maréchal (ed.), pp. 159–180, Elsevier.

Levy-Vehel J. and Lutton E. (1993) Optimization of Fractal Functions using Genetic Algorithms, *Report No. 1941*, Institut National de Recherche en Informatique et an Automatique, Le Chesnay Cedex, France.

Lewontin R.C. (1974) *The Genetic Basis of Evolutionary Change*, Columbia University Press.

Libeskind-Hadas R. and Maragos P. (1987) Application of Iterated Function Systems and Skeletonization to Synthesis of Fractal Images, *Visual Communication and Image Processing II*, SPIE Vol. 845, pp. 277–285.

Lindsay R.K. (1968) Artificial Evolution of Intelligence, *Contemporary Psychology*, Vol. 13, No. 3, pp. 113–116.

Ljolje A. and Riley M.D. (1992) Optimal Speech Recognition using Phone Recognition and Lexical Access, *Proceedings of the International Conference on Spoken Language Processing*, Alberta, Canada, pp. 313–316.

Luger G.F. and Stubblefield W.A (1993) *Artificial Intelligence: Structures and strategies for complex problem solving*, The Benjamin/Cummings Publishing Company.

Mason A.J. (1993) Crossover Non-linearity Ratios and the Genetic Algorithm: Escaping the Blinkers of Schema Processing and Intrinsic Parallelism, *Report No. 535b*, School of Engineering, University of Auckland, Private Bag 92019, New Zealand.

Mayr E. (1988) *Toward a New Philosophy of Biology: Observations of an Evolutionist*, Belknap Press.

Mazel D.S. and Hayes M.H. (1992) Using Iterated Function Systems to Model Discrete Sequences, *IEEE Transactions on Signal Processing*, Vol. 40, No. 7, pp. 1724–1734.

McClelland J.L., Rumelhart D.E. and the PDP Research Group (1986) *Parallel Distributed Processing: Explorations in the microstructure of cognition. v2: Psychological and biological models*, MIT Press.

Mitton R. (1986) A Computer Usable Version of the Oxford Advanced Learner's Dictionary, *Technical report*, Department of Computer Science, Birbeck College, Malet Street, London.

Monro D.M. and Dudbridge F. (1992) Fractal Approximation of Image Blocks, *International Conference on Acoustics, Speech and Signal Processing*, Vol. III, pp. 485–488.

Monro D.M., Dudbridge F. and Wilson A. (1990) Deterministic Rendering of Self-Affine Fractals, *IEE Colloquium on the Application of Fractal Techniques in Image Processing*, London.

Mumford D. (1987) The Problem of Robust Shape Descriptors, *IEEE First International Conference on Computer Vision*, pp. 602–606.

Murveit H., Butzberger J., Digalakis V. and Weintraub M. (1993) Large Vocabulary Dictation using SRI's Decipher Speech Recognition System: Progressive search techniques, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 319–322.

Myers C.S. and Rabiner L.R. (1981) A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 29, No. 2, pp. 284–297.

Nettleton D.J. and Collingham R.J. (1995) Word Lattice Optimisation using Evolutionary Algorithms, to be submitted to *Journal of Natural Language Engineering*, Cambridge University Press.

Nettleton D.J. and Garigliano R. (1993) Search Space Reductions in Deriving a Fractal Set for an Arbitrary Shape, *Adaptive and Learning Systems II*, F. A. Sadjadi (ed.), SPIE Vol. 1962, pp. 137–145.

Nettleton D.J. and Garigliano R. (1994a) Reductions in the Search Space for Deriving a Fractal Set of an Arbitrary Shape, in press *Journal of Mathematical Imaging and Vision*, Kluwer.

Nettleton D.J. and Garigliano R. (1994b) Evolutionary Algorithms for Dialogue Optimisation in the LOLITA Natural Language Processor, *Seminar on Adaptive Computing and Information Processing*, pp. 810–815, London.

Nettleton D.J. and Garigliano R. (1994c) Evolutionary Algorithms and a Fractal Inverse Problem, *Journal of Biological and Information Processing Systems* (BioSystems), Vol. 33, pp. 221–231.

Nettleton D.J. and Garigliano R. (1994d) Evolving Fractals, in press *Journal of Computers and Graphics*, Pergamon Press.

Nettleton D.J. and Garigliano R. (1994e) Subsymbolic Processing using Adaptive Algorithms, *Second International Conference on Artificial Intelligence and Symbolic Mathematical Computations*, Cambridge, UK, to be published in Lecture Notes in Computer Science, Springer-Verlag.

Nettleton D.J. and Garigliano R. (1994f) Evolutionary algorithms for dialogue optimisation as an example of hybrid NLP system, *International Conference on New Methods in Language Processing*, Manchester.

Nettleton D.J. and Garigliano R. (1994g) Shape Representation and Evolutionary Algorithms, *IEE Colloquium on Genetic Algorithms in Image Processing and Vision*, London.

Nettleton D.J. and Garigliano R. (1994h) Dialogue Optimisation in the LOLITA Natural Language Processor using Evolutionary Algorithms, *The Applications Handbook of Genetic Algorithms: New Frontiers*, L. Chambers (ed.), CRC Press, to appear.

Nettleton D.J., Garigliano R. and Siemens Plessey Defence Systems (1993) Large Ratios of Mutation to Crossover: The example of The Travelling Salesman

Problem, *Adaptive and Learning Systems II*, F.A. Sadjadi (ed.), SPIE Vol. 1962, pp. 110–119.

Newell A. and Simon H. (1976) Computer Science as Empirical Enquiry: Symbols and search, *Communications of the ACM*, Vol. 19, No. 3, pp. 113–126.

Newman W.M. and Sproull R.F. (1979) *Principles of Interactive Computer Graphics*, McGraw-Hill.

Peitgen H., Jürgens H. and Saupe D. (1992) *Fractals for the Classroom: Part One Introduction to Fractals and Chaos*, Springer-Verlag.

Radcliffe N.J. (1991a) Forma Analysis and Random Respectful Recombination, *Proceedings of the Fourth International Conference on Genetic Algorithms*, R.K. Belaw and L.B. Booker (eds.), pp. 222–229, Morgan Kaufmann.

Radcliffe N.J. (1991b) Equivalence Class Analysis of Genetic Algorithms, *Complex Systems*, Vol. 5, No. 2, pp. 183–205.

Radcliffe N.J. (1992) Non-Linear Genetic Representations, *Parallel Problem Solving from Nature 2*, R. Maenner and B. Manderick (eds.), North Holland.

Rechenberg I. (1973) *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*, Frommann-Holzbood.

Rich E.A (1990) Artificial Intelligence, in *Encyclopedia of Artificial Intelligence*, Vol. 1, pp. 9–16, John Wiley & Sons.

Rinaldo R. and Zakhor A. (1992) Inverse Problem for Two-Dimensional Fractal Sets using the Wavelet Transform and the Moment Method, *International Conference on Acoustics, Speech and Signal Processing*, Vol. IV, pp. 665–668.

Rumelhart D.E., McClelland J.L. and the PDP Research Group (1986) *Parallel Distributed Processing: Explorations in the microstructure of cognition. v1: Foundations*, MIT Press.

Russell M.J. (1992) The Development of the Speaker Independent ARM Speech Recognition System, *Proceedings of the Institute of Acoustics Speech and Hearing Conference*, Windermere, Cumbria, UK.

Sakoe H. (1979) Two-level dp-matching — A Dynamic Programming-based Pattern Matching Algorithm for Connected Word Recognition, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 27, No. 6, pp. 588–595.

Schaffer J.D. (ed.) (1989) *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann.

Schank R.C. and Abelson R.P. (1977) *Scripts, Plans, Goals and Understanding*, Lawrence Erlbaum Associates.

Sebald A.V. and Fogel L.J. (eds.) (1994) *Evolutionary Programming - Proceedings of the 3rd Annual Conference*, World Scientific Publishing.

Sharp D.W.N. and Cripps M.D. (1991) Parallel Algorithms that Solve Problems by Communication, *Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing*, pp. 87–94.

Sharp D.W.N and While R.L. (1992) Pattern Recognition Using Fractals, *Imperial College Research Report DoC 92/30*, Department of Computing, Imperial College of Science, Technology & Medicine, London.

Short S., Collingham R.J. and Garigliano R. (1994a) What did I say...? – Using Meaning to Assess Speech Recognisers, *Institute of Acoustics Autumn Conference on Speech and Hearing*, Windermere, Cumbria, UK.

Short S., Collingham R.J. and Garigliano R. (1994b) Making Use of Semantics in an Automatic Speech Recognition System, *Institute of Acoustics Autumn Conference on Speech and Hearing*, Windermere, Cumbria, UK.

Silverman H.F. and Morgan D.P. (1990) The Application of Dynamic Programming to Connected Speech Recognition, *IEEE ASSP Magazine*, pp. 6–25.

Smith H.F. (1991) A Garden of Fractals, *Fractal in the Fundamental and Applied Sciences*, H.O. Peitgen, J.M. Henriques and L.F. Penedo (eds.), pp. 407–424, Elsevier.

Smith M.H., Garigliano R. and Morgan R.G. (1994) Generation in the LOLITA System: An engineering approach, *Seventh International Workshop on Natural Language Generation*, Maine, USA.

Solomonoff R.J. (1966) Some Recent Work in Artificial Intelligence, *Proceedings of the IEEE*, Vol. 54, No. 12, pp. 1687–1697.

Spears W. (1994) Hills, Hamming Distance, and GAs, *Genetic Algorithms Digest*, Vol. 8, Issue 11.

Srinivas M. and Patnaik L.M. (1994) Genetic Algorithms: A Survey, *Computer*, Vol. 27, No. 6, pp. 17–26.

Sutherland I.E. and Hodgman G.W. (1974) Reentrant Polygon Clipping, *Communications of the ACM*, Vol. 17, No. 1, pp. 32–42.

Svartvik J. (1992) *The London-Lund Corpus of Spoken English: Users' Manual*, distributed by the Norwegian Computing Centre for the Humanities, Department of English, Lund University.

Syswerda G. (1989) Uniform Crossover in Genetic Algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, J.D. Schaffer (ed.), pp. 2–9, Morgan Kaufmann.

Vines G. and Hayes M.H. (1992) Fast Algorithm to Select Maps in an Iterated Function System Fractal Model, *Visual Communication and Image Processing*, SPIE Vol. 1818, pp. 944–949.

Vrscay E.R. (1991a) Iterated Function Systems: Theory, applications and the inverse problem, *Fractals Geometry and Analysis*, J. Bélair and S. Dubuc (eds.), pp. 405–468, Kluwer.

Vrscay E.R. (1991b) Moment and Collage Methods for the Inverse Problem of Fractal Construction with Iterated Function Systems, *Fractal in the Fundamental and Applied Sciences*, H.O. Peitgen, J.M. Henriques and L.F. Penedo (eds.), pp. 443–461, Elsevier.

Waite J. (1990) A Review of Iterated Function System Theory for Image Compression, *IEE Colloquium on the Application of Fractal Techniques in Image Processing*, London.

Watt S. (1993) Fractal Behaviour Analysis, *Prospects for Artificial Intelligence*, A. Sloman *et al.* (eds.), IOS Press.

Weiss N.A. and Hassett M.J. (1991) *Introductory Statistics*, 3rd edition, Addison Wesley.

Bibliography

Anderson J.A.D.W., Sullivan G.D. and Baker K.D. (1988) Constrained Constructive Solid Geometry: A unique representation of scenes, *Proceedings of the Fourth Alvey Vision Conference*, University of Manchester, pp. 91–96.

Andrews H.C. (1972) *Introduction to Mathematical Techniques in Pattern Recognition*, Wiley-Interscience.

Baker J.E. (1985) Adaptive Selection Methods for Genetic Algorithms, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette (ed.), pp. 101–111, Lawrence Erlbaum Associates.

Baker J.E. (1987) Reducing Bias and Inefficiency in the Selection Algorithm, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette (ed.), pp. 14–21, Lawrence Erlbaum Associates.

Barnsley M.F. (1989) Fractals and Chaos, *Proceedings of the BCS seminar on Fractals and Chaos*, pp. 1–21.

Battle D.L. and Vose M.D. (1993) Isomorphisms of Genetic Algorithms, *Artificial Intelligence*, Vol. 60, pp. 155–165.

Batty M. (1985) Fractals — Geometry between dimensions, *New Scientist*, Vol. 105, No. 1450, 4 April, pp. 31–35.

Batty M. (1989) Geography and the New Geometry, *Geography Review*, March, pp. 7–10.

Beasley D., Bull D.R. and Martin R.R. (1993) A Sequential Niche Technique for Multimodal Function Optimization, *Journal of Evolutionary Computation*, Vol. 1, No. 2, pp. 101–125.

Berger M.A. (1988) Encoding Images through Transition Probabilities, *Mathematical Computer Modelling*, Vol. 11, pp. 575–577.

Berger M.A. (1992) Random Affine Iterated Function Systems: Curve Generation and Wavelets, *SIAM Review*, Vol. 34, No. 3, pp. 361–385.

Bianchini R. and Brown C. (1992) Parallel Genetic Algorithms on Distributed-Memory Architectures, *Technical Report 436*, The University of Rochester, Computer Science Department, Rochester, New York 14627.

Booker L. (1987) Improving Search in Genetic Algorithms, in *Genetic Algorithms and Simulated Annealing*, L. Davis (ed.), Pitman.

Bown W. (1992) Fractal Maths Adds up to a Clearer Vision, *New Scientist*, No. 1824, 6 June, p. 20.

Brooks R.A. (1981) Symbolic Reasoning among 3D Models and 2D Images, *Artificial Intelligence*, Vol. 17, pp. 285–348.

Bryant V. (1987) *Metric Spaces: Iteration and Application*, Cambridge University Press.

Bullock B.L. (1978) The Necessity for a Theory of Specialized Vision, in *Computer Vision Systems*, A. Hanson and E. Riseman (eds.), Academic Press.

Canny J. (1986) A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, pp. 679–697.

Conrad M. (1993) Structuring Adaptive Surfaces for Effective Evolution, *Proceedings of the 2nd Annual Conference on Evolutionary Programming*, D.B. Fogel and W. Atmar (eds.), pp. 1–10.

Cyganski D., Orr J.A., Cott T.A. and Dodson R.J. (1987) Development, Implementation, Testing and Application of an Affine Invariant Curvature Function, *IEEE First International Conference on Computer Vision*, pp. 496–500.

Davis L. (1991) *Handbook of Genetic Algorithms*, Van Nostrand Reinhold.

Davis L. and Steenstrup M. (1987) Genetic Algorithms and Simulated Annealing: An Overview, in *Genetic Algorithms and Simulated Annealing*, L. Davis (ed.), Pitman.

Dawkins R. (1976) *The Selfish Gene*, Oxford University Press.

Dewdney A.K. (1990) Mathematical Recreations: How to transform flights of fancy into fractal flora or fauna, *Scientific American*, May, pp. 90–93.

Dorigo M. (1993) Genetic and Non-Genetic Operators in ALECYS, *Journal of Evolutionary Computation*, Vol. 1, No. 2, pp. 151–164.

Dubuc S. and Elqortobi A. (1990) Approximations of Fractal Sets, *Journal of Computational and Applied Mathematics 29*, pp. 79–89.

Duncan B.S. (1993) Parallel Evolutionary Computation, *Proceedings of the 2nd Annual Conference on Evolutionary Programming*, D.B. Fogel and W. Atmar (eds.), pp. 202–208.

Falconer K.J. (1993) *Fractal Geometry*, John Wiley.

Fischler M.A. (1978) On the Representation of Natural Scenes, in *Computer Vision Systems*, A. Hanson and E. Riseman (eds.), Academic Press.

Funt B.V. (1980) Problem Solving with Diagrammatic Representations, *Artificial Intelligence*, Vol. 13, No. 3, pp. 201–230.

Garigliano R. and Nettleton D.J. (1992) Shape Representation and Recognition using Iterated Function Systems and Genetic Algorithms, *Technical Report 7/92*, Department of Computer Science, University of Durham, UK.

Garigliano R. and Nettleton D.J. (1994) System Transformation: A Formal Approach, to be submitted to *Journal of Theoretical Computer Science.*

Goldberg D.E. (1994) Genetic and Evolutionary Algorithms Come of Age, *Communications of the ACM*, Vol. 37, No. 3.

Goldberg D.E. and Segrest P. (1987) Finite Markov Chain Analysis of Genetic Algorithms, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, J.J.Grefenstette (ed.), pp. 1–8, Lawrence Erlbaum Associates.

Grefenstette J.J. (1993) Genetic Algorithms, Guest Editor's Introduction, *IEEE Expert: Intelligent Systems and their Applications*, Vol. 8, No. 5, pp. 5–8.

Grefenstette J.J. and Fitzpatrick J.M. (1985) Genetic Search with Approximate Function Evaluations, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette (ed.), pp. 112–120, Lawrence Erlbaum Associates.

Grefenstette J.J., Gopal R., Rosmaita B. and Van Gucht D. (1985) Genetic Algorithms for Travelling Salesman Problem, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette (ed.), pp. 160–168. Lawrence Erlbaum Associates.

Grossman S.I. (1987) *Elementary Linear Algebra: Third edition*, Wadsworth.

Hanson A.R. and Riseman E.M. (1978) VISIONS: A Computer System for Interpreting Scenes, in *Computer Vision Systems*, A. Hanson and E. Riseman (eds.), Academic Press.

Holland J.R.C., Oliver I.M. and Smith D.J. (1987) A Study of Permutation Crossover Operators on the Travelling Salesman Problem, *Genetic Algorithms* *and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette (ed.), pp. 224–230, Lawrence Erlbaum Associates.

Huxley J. (1963) *Evolution in Action*, Pelican Books.

Juliany J. and Vose M.D. (1993) The Genetic Algorithm Fractal, *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest (ed.), Morgan Kaufmann.

Khuri S., Bäck T. and Heitkötter J. (1994) An Evolutionary Approach to Combinatorial Optimization Problems, *Proceedings of CSC'94*, ACM Press.

Lamdan Y., Schwartz J.T. and Wolfson J. (1988) Object Recognition by Affine Invariant Matching, *Proceedings of the Computer Society Conference on Computer Vision and Pattern Matching*, pp. 335–344, Ann Arbor.

Louis S., McGraw G. and Wyckoff R.O. (1993) CBR Assisted Explanation of GA Results, *Journal of Theoretical and Experimental Artificial Intelligence*, Vol. 5, No. 1, pp. 21–37.

Mandelbrot B.B. (1977) *Fractals: Form, chance and dimension*, W.H. Freeman and Co.

Mandelbrot B.B. (1982) *The Fractal Geometry of Nature*, W.H. Freeman and Co.

Mantica G. (1991) Techniques for Solving Inverse Fractal Problems, *Fractal in the Fundamental and Applied Sciences*, H.O. Peitgen, J.M. Henriques and L.F. Penedo (eds.), pp. 255–268, Elsevier Science Publishers.

Marsaglia G. and Zaman A. (1991) Random Number Generators, *Annals of Applied Probability*, Vol. 1, No. 3, pp. 462-480.

Michalewicz Z. (1993) A Hierarchy of Evolution Programs: An experimental study, *Journal of Evolutionary Computation*, Vol. 1, No. 1, pp. 51–76.

Monk R. (1993) The End of Intelligence?, *The Observer Magazine*, 17 October, pp. 12–18.

Mühlenbein H. and Schlierkamp-Voosen D. (1993) Predictive Models for the Breeder Genetic Algorithm, *Journal of Evolutionary Computation*, Vol. 1, No. 1, pp. 25–50.

Nevatia R. (1978) Characterization and Requirements of Computer Vision Systems, in *Computer Vision Systems*, A. Hanson and E. Riseman (eds.), Academic Press.

Pearl J. (1985) *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley.

Penrose R. (1989) *The Emperor's New Mind: Concerning computers, minds and the laws of physics*, Oxford University Press.

Pentland A.P. (1984) Fractal-Based Description of Natural Scenes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 6, pp. 661–674.

Pentland A.P. (1986) Perceptual Organization and the Representation of Natural Form, *Artificial Intelligence*, Vol. 28, pp. 293–331.

Peterson I. (1988) *The Mathematical Tourist*, Freeman.

Peterson I. (1990) *Islands of Truth: A mathematical mystery cruise*, Freeman.

Pickover C.A. (1990) *Computer Patterns Chaos and Beauty*, Sutton.

Qi X. and Palmieri F. (1993) Adaptive Mutation in the Genetic Algorithm, *Proceedings of the 2nd Annual Conference on Evolutionary Programming*, D.B. Fogel and W. Atmar (eds.), pp. 11–22.

Radcliffe N.J. and George F.A.W. (1993) A Study in Set Recombination, *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest (ed.), Morgan Kaufmann.

Ridley M. (1985) *The Problems of Evolution*, Oxford University Press.

Samarabandu J., Acharya R., Hausmann E. and Allen K. (1992) Analysis of Bone X-rays using Morphological Fractals, *International Conference on Acoustics, Speech and Signal Processing*, Vol. III, pp. 133–136.

Shapira Y. and Yllman S. (1991) A Pictorial Approach to Object Classification, *12th International Joint Conference on Artificial Intelligence*, J. Myopoulos and R. Reiter (eds.), pp. 1257–1263, Morgan Kaufmann.

Shonkwiler R. (1989) An Image Algorithm for Computing the Hausdorff Distance Efficiently in Linear Time, *Information Processing Letters*, Vol. 30, pp. 87–89.

Sloman A. (to appear) The Emperor's Real Mind, *Artificial Intelligence.*

Stevens R.T. (1988) *Graphics Programming in C*, M&T Publishing.

Stewart I. (1989) *Does God Play Dice? The Mathematics of Chaos*, Blackwell.

Stucki D.J. and Pollack J.B. (1992) Fractal (Reconstructive Analogue) Memory, *14th Annual Conference of the Cognitive Science Society*, Bloomington, Indiana, USA.

Whitley D. (1993) A Genetic Algorithm Tutorial, *Technical Report CS-93-103*, Colorado State University, USA.

Wilf H. S. (1986) *Algorithms and Complexity*, Prentice-Hall International Editions.

Wu C.M., Chou W.S. and Chen Y.C. (1991) Two-Stage Liver Tissue Classification through Fractal Geometry, *Journal of the Chinese Institute of Engineers*, Vol. 14, No. 5, pp. 519–529.

Appendix A

This appendix describes **Algorithm 1** which can be used to choose $a, b, c, d \in (-1, 1)$ such that they satisfy:

$$\left| \frac{a + d \pm \sqrt{(a-d)^2 + 4bc}}{2} \right| < 1. \qquad (A1)$$

The selection algorithm consists of two parts. The first chooses a, b, c and d such that the value inside the square root is negative. The second chooses values which ensure the value is positive. Since only one of these two parts is needed, probabilities for the selection of each part are required. These are calculated by examining the number of valid combinations of a, b, c and d which have $\sqrt{(a-d)^2 + 4bc} \geq 0$ and the number which have $\sqrt{(a-d)^2 + 4bc} < 0$ for a given acc.

The table below shows the probabilities which should be used when deciding whether to use **Part 1** or **Part 2** of **Algorithm 1**.

acc	0.2	0.1	0.05	0.02
P(**Part 1**)	0.337	0.364	0.376	0.383
P(**Part 2**)	0.663	0.646	0.624	0.617

Algorithm 1

Part 1 — a, b, c and d are to be such that $(a-d)^2 + 4bc < 0$.

Then since $|x \pm i\sqrt{y}| = \sqrt{x^2 + y}$ for $y > 0$, Equation $A1$ becomes:

$$0 \leq \sqrt{(ad - bc)} < 1.$$

Choose $a \in (-1,1)$ and $d \in (-1,1)$; b and c then need to be selected such that they satisfy:

$$-1 + ad < bc \leq ad.$$

1. If $ad \geq 0$ then require $-1 + ad < bc \leq ad$. Choose $b \in (-1,1)$, then:
   (a) if $b = 0$ choose $c \in (-1,1)$,
   (b) if $b > 0$ choose $c \in (\max\{-1, \frac{-1+ad}{b}\}, \min\{1, \frac{ad}{b}\})$,
   (c) if $b < 0$ choose $c \in (\max\{-1, \frac{ad}{b}\}, \min\{1, \frac{-1+ad}{b}\})$.
2. If $ad < 0$ then require $-1 < bc \leq ad$ so with uniform probability either:
   (a) choose $b \in (-1, ad)$ then choose $c \in [\frac{ad}{b}, 1)$,
   (b) choose $b \in (-ad, 1)$ then choose $c \in (-1, \frac{ad}{b}]$.

With probability 0.5 exchange the values of b and c.

Part 2 — a, b, c and d are to be such that $(a-d)^2 + 4bc \geq 0$.

Then from Equation $A1$ both of the following must hold:

$$-2-a-d < \sqrt{(a-d)^2 + 4bc} < 2-a-d \qquad -2+a+d < \sqrt{(a-d)^2 + 4bc} < 2+a+d.$$

But $(a-d)^2 + 4bc \geq 0$ so $\sqrt{(a-d)^2 + 4bc} \geq 0$ and since $-2 - a - d < 0$ and $-2 + a + d < 0$ $\forall a, d \in (-1,1)$ the above equations become:

$$0 < \sqrt{(a-d)^2 + 4bc} < 2 - a - d \qquad 0 < \sqrt{(a-d)^2 + 4bc} < 2 + a + d.$$

These give:

$$0 < (a-d)^2 + 4bc < (2 - a - d)^2 \quad \Rightarrow \quad \frac{-(a-d)^2}{4} < bc < (1-a)(1-d),$$

$$0 < (a-d)^2 + 4bc < (2 + a + d)^2 \quad \Rightarrow \quad \frac{-(a-d)^2}{4} < bc < (1+a)(1+d).$$

Therefore since both of the above must hold:

$$\frac{-(a-d)^2}{4} < bc < \min\{1, (1-a)(1-d), (1+a)(1+d)\}.$$

Choose $a \in (-1,1)$, $d \in (-1,1)$ and $b \in (-1,1)$ then:

1. if $b = 0$ choose $c \in (-1,1)$,
2. if $b > 0$ choose $c \in (\max\{-1, \frac{-(a-d)^2}{4b}\}, \min\{1, \frac{(1-a)(1-d)}{b}, \frac{(1+a)(1+d)}{b}\})$,
3. if $b < 0$ choose $c \in (\max\{-1, \frac{(1-a)(1-d)}{b}, \frac{(1+a)(1+d)}{b}\}, \min\{1, \frac{-(a-d)^2}{4b}\})$.

With probability 0.5 exchange the values of b and c.

Appendix B

This appendix describes **Algorithm 2** which can be used to choose values of e and f (for given a, b, c and d satisfying the constraints of Sections 6.4 and 6.6) such that:

1. the limit point of the transformation lies in the bounding box,
2. no edge of the transformed bounding box lies entirely outside the bounding box.

That is to choose e and f such that they satisfy:

$$\begin{array}{ccc} -X \leq e(1-d) + bf \leq X & \quad -Y \leq f(1-a) + ce \leq Y & \quad (B1) \\ e \in [-C1, C1] & \quad f \in [-C2, C2] & \end{array}$$

where $X = ((1-a)(1-d) - bc)Xmax$ and $Y = ((1-a)(1-d) - bc)Ymax$, and where $C1$ and $C2$ are constants which are the allowed ranges of e and f respectively (calculated in Section 6.6).

As can be seen from Equations $B1$ the problem is to choose e and f so that they lie in both the parallelogram and the rectangle defined. The algorithm presented ensures that every valid e and f can be selected, but each possibility does not have an equal probability of selection. Furthermore, the algorithm selects e first and then f — this order of selection can easily be reversed by making suitable substitutions.

Algorithm 2

Calculate the vertices of the parallelogram by finding the points of intersection of $x(1-d) + by = X$ with $y(1-a) + cx = -Y$ and with $y(1-a) + cx = Y$, and call these $P^0 = (P_x^0, P_y^0)$ and $P^1 = (P_x^1, P_y^1)$. Note that these must exist since

$\frac{b}{1-d} \neq \frac{1-a}{c}$ (see Section 6.5). The remaining two vertices are then $P^2 = -P^0$ and $P^3 = -P^1$.

Calculate the number, N_1, and positions I^0, I^1 of the points of intersection of the line from P^0 to P^1 with the box defined by the vertices $(C1, C2)$, $(C1, -C2)$, $(-C1, -C2)$ and $(-C1, C2)$. Similarly calculate, the number, N_2, and the positions I^2, I^3, of the points of intersection of the line from P^0 to P^3 with the box.

If either $N_1 = 0$ or $N_2 = 0$ (but not both) then:

1. If $N_1 = 0$. Calculate $Ix = \max\{|I_x^2|, |I_x^3|\}$ then choose e such that $e \in [-Ix, Ix]$.
2. If $N_2 = 0$. Calculate $Ix = \max\{|I_x^0|, |I_x^1|\}$ then choose e such that $e \in [-Ix, Ix]$.

Otherwise, calculate $Px = \max\{|P_x^0|, |P_x^1|\}$, then choose e such that:

$$e \in [\max\{-C1, -Px\}, \min\{C1, Px\}]$$

Given that an e has been chosen, $\tilde{e}$ say, a range of possible values for f now needs to be calculated. Either:

1. $b = 0$ — parallelogram has two sides vertical.

   Calculate points of intersection of $x = \tilde{e}$ and $y(1-a) + cx = \pm Y$ and call these $I^0 = (I_x^0, I_y^0)$ and $I^1 = (I_x^1, I_y^1)$. Then choose f such that:

   $$f \in [\max\{-C2, I_y^j\}, \min\{C2, I_y^i\}]$$

   where i and j are such that $I_y^i \geq I_y^j$.

2. $b \neq 0$.

   Calculate the points of intersection of $x = \tilde{e}$ with $y(1-a) + cx = \pm Y$ and $x(1-d) + by = \pm X$ and call these I^0, I^1, I^2, I^3 with $I^i = (I_x^i, I_y^i)$. Choose $i, j, k, l \in \{0, 1, 2, 3\}$ such that $i \neq j \neq k \neq l$ and $I_y^i \leq I_y^j \leq I_y^k \leq I_y^l$. Then choose f such that:

   $$f \in [\max\{-C2, I_y^j\}, \min\{C2, I_y^k\}].$$