An Analysis of the Genetic Algorithm and Abstract Search Space Visualisation

by

RICHARD ALAN HARRIS

A thesis submitted to the University of Plymouth in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

University of Plymouth Engineering Design Centre
Faculty of Technology

August 1996

Abstract

An Analysis of the Genetic Algorithm and Abstract Search Space Visualisation

by

RICHARD ALAN HARRIS

The Genetic Algorithm (Holland, 1975) is a powerful search technique based upon the principles of Darwinian evolution. In its simplest form the GA consists of three main operators - crossover, mutation and selection. The principal theoretical treatment of the Genetic Algorithm (GA) is provided by the Schema Theorem and building block hypothesis (Holland, 1975). The building block hypothesis describes the GA search process as the combination, sampling and recombination of fragments of solutions known as building blocks. The crossover operator is responsible for the combination of building blocks, whilst the selection operator allocates increasing numbers of samples to good building blocks. Thus the GA constructs the optimal (or near-optimal) solution from those fragments of solutions which are, in some sense, optimal.

The first part of this thesis documents the development of a technique for the isolation of building blocks from the populations of the GA. This technique is shown to extract exactly those building blocks of interest - those which are sampled most regularly by the GA. These building blocks are used to empirically investigate the validity of the building block hypothesis. It is shown that good building blocks do not combine to form significantly better solution fragments than those resulting from the addition of randomly generated building blocks to good building blocks. This results casts some doubt onto the value of the building block hypothesis as an account of the GA search process (at least for the functions used during these experiments).

The second part of this thesis describes an alternative account of the action of crossover. This account is an approximation of the geometric effect of crossover upon the population of samples maintained by the GA. It is shown that, for a simple function, this description of the crossover operator is sufficiently accurate to warrant further investigation. A pair of performance models for the GA upon this function are derived and shown to be accurate for a wide range of crossover schemes. Finally, the GA search process is described in terms of this account of the crossover operator and parallels are drawn with the search process of the simulated annealing algorithm (Kirkpatrick et al, 1983).

The third and final part of this thesis describes a technique for the visualisation of high dimensional surfaces, such as are defined by functions of many parameters. This technique is compared to the statistical technique of projection pursuit regression (Friedman & Tukey, 1974) and is shown to compare favourably both in terms of computational expense and quantitative accuracy upon a wide range of test functions. A fundamental flaw of this technique is that it may produce poor visualisations when applied to functions with a small high frequency (or order) components.

Contents

Figures

Tables

Acknowledgement

First Supervisor
Professor M.Denham

Second Supervisors
Dr. C.Ricketts
Dr. I.Parmee

With Special Thanks to
Dr. C.Ellis

The research documented by this thesis was carried out at the
University of Plymouth Engineering Design Centre
EPSRC Grant No. GR/J41765

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

This study was carried out at the University of Plymouth Engineering Design Centre under EPSRC Grant No. GR/J41765.

During this period, an undergraduate course in multivariate statistics was taken and relevant scientific seminars and conferences were regularly attended.

Publications :

An alternative description of the action of crossover. *Proceedings of ACEDC'94.* PEDC, University of Plymouth.

An empirical investigation of the Schema Theorem and building block hypothesis. *Internal Report PEDC-01-93.* PEDC, University of Plymouth

An Alternative Description of the Action of Crossover. *Internal Report PEDC-02-93.* PEDC, University of Plymouth

Signed ....

Date ..30/08/96..

1 Introduction

Traditional optimisation techniques generally operate by using local information to generate an approximation to the line of steepest ascent (or descent in the case of minimisation). Taking a short step in this direction, these technique generate a new line of steepest ascent from the new local information. After a number of iterations the technique will reach a point from which it can ascend no longer and this point must therefore, by definition, be locally optimal (better than all of it's neighbouring points). Unfortunately, there is no possible way of determining whether or not there is a better point somewhere else or that the given point is in fact the best possible - the global optimum. These techniques are therefore often referred to as hill-climbing or local optimisation techniques, since they climb the peak upon which they are situated and guarantee only to identify local optima.

The Genetic Algorithm (Holland, 1975) is an adaptive global search technique based upon the principles of Darwinian evolution. The Genetic Algorithm (GA) utilises a large number of points, all of which compete for representation in following iterations. A number of genetic operators are applied to these points, from which a new set of points are generated and in their turn must compete for representation in the same fashion. The description of the GA as a global optimisation technique, despite the fact that it is not possible in general to confirm the global optimality of any solution, stems from the search process of the GA. Unlike traditional optimisation techniques, the GA does not rely upon local information and does not follow a path of steepest ascent (or descent). As an optimisation technique the GA has enjoyed successful application in a wide range of domains, and is typically applied to those problems which defy traditional hill-climbing optimisation techniques. The relative success of the GA in such domains, together with the widely distributed based search process, are the justification for the description of the GA as a global optimisation technique.

The principal theoretic description of the GA is provided by the Schema Theorem and building block hypothesis (Holland, 1975). The building block hypothesis accounts for the success of the GA in terms of fragments of solutions known as building blocks. These building blocks represent subtle underlying relationships between the parameters of the problem at hand. It is asserted by the building block hypothesis that the GA operates through the combination, sampling and recombination of such building blocks. Poor quality building blocks are rejected, good quality building blocks prosper and the GA constructs its ultimate solution from those building blocks which are themselves optimal or near optimal.

1.1 The Genetic Algorithm as an Accountable Optimisation Technique

One of the failings of optimisation techniques in general is that they operate as black boxes. The user applies a technique within a given domain and is rewarded with a set of parameters describing the solution generated by that technique. The identification of optimal, near optimal or locally optimal solutions to a given problem is clearly useful, although this information alone does not improve the users' understanding of the problem itself or of why the given solution is better than its neighbours. It would be of great advantage, therefore, if a technique were developed which in addition to the generation of such a solution to a problem would be capable of justifying the solution upon which it converged. Such an accountable optimisation technique would not only give the user viable solutions to the problems to which it was applied, but would also increase the understanding of such problems by providing much needed justification for the optimality of their solutions.

1.1.1 Building Blocks as Justification for the Solutions Generated by the GA

As has been noted, the building blocks predicted by the building block hypothesis represent relationships between the parameters of a problem. Given the building block hypothesis, those building blocks which prosper under the GA must be representative of relationships between parameters which describe good regions of the search space, or equivalently of relationships which are innately of high utility.

In the construction of an accountable optimisation technique such relationships would clearly prove very useful. Given the solution upon which the GA finally converges, the identification of those good building blocks from which it has been constructed would yield the relationships between the parameters of the solution which account for its' optimality or near optimality. These building blocks are therefore an ideal source of information from which a justification of the generated solution may be constructed. The GA is perhaps unique amongst optimisation techniques in that the proposed mechanics of its' search process could yield a meaningful justification for the solution upon which it converges.

The potential ramifications of the identification and subsequent analysis of those building blocks which are extensively exploited by the GA during the search process inspired the initial aim of this research project - namely the development of a technique for the extraction of these highly sampled building blocks.

1.1.2 Building Blocks as a Rich Source of Information

In addition to the use of building blocks for the justification of the solutions generated by the GA, those building blocks most regularly sampled by the GA during the search process could provide a rich additional source of information concerning the problems to which the GA is applied. The relationships between parameters described by the highly sampled building blocks are indicative of underlying traits of the problem domain, some of which may have been previously unknown to the user.

This wealth of relational information could reveal a great deal about the nature of the search space. If this resource were properly exploited, the GA would transcend the goal of an accountable optimisation technique and yield an algorithm which, in addition to providing a solution and a justification of that solution, could give the user a greater understanding of the relationships between the parameters of the system under examination.

The latter phase of this research was founded upon the obvious benefits which would result from a system which was capable not only of identifying good solutions of a problem and justifying those solutions, but also of revealing possibly hitherto unknown properties of that problem. This second phase of the research was to be concerned with the effective dissemination of the information contained with the highly sampled building blocks - specifically with the development of a graphical visualisation technique which would present this information to the user in a clear and concise fashion.

1.1.3 The Use of Building Blocks to Guide Genetic Search

In addition to providing the user with additional information regarding the domain in which the GA is applied, the highly sampled building block identified by the extraction technique could be used to improve the convergence properties of the GA itself. If the GA were to utilise these building blocks for the construction of the solution upon which it converges, the early identification of potentially useful building blocks would allow their explicit propagation. Any inefficiency displayed by the GA in the exploitation of high utility building blocks could possibly be avoided through the early identification and subsequent enforced sampling of such building blocks.

A further research goal was therefore to develop a systematic methodology for the promotion of those building blocks which were, through on-line analysis, identified as potentially of high utility for the construction of an optimal or near optimal solution. The benefits of such an approach are clear - a potential reduction of the number of trials necessary for the convergence of the GA together with a potential improvement in the quality of the solution upon which the GA finally converges.

1.1.4 The Failure of the Building Block as a Source of Information

Having developed a technique for the on-line extraction and subsequent evaluation of highly sampled building blocks, the next research goal was to use these building blocks to empirically confirm the validity of the building block hypothesis. The experiments chosen for this analysis examined the effect of combining pairs of building blocks to construct new, larger, fragments of solutions. The utility of these newly constructed fragments was then compared to the utility of the building blocks from which they were constructed. If the building block hypothesis were true, those fragments constructed from good building blocks would yield a significantly greater improvement in utility than those constructed from poor building blocks. In other words the building blocks would combine - good building blocks would combine to form good solution fragments which in turn would combine to form good solutions. The assumption that building blocks combine is implicit within the building block hypothesis. If this were not so, poor quality building blocks would receive equal numbers of trials as good quality building blocks since when combined to form solutions there would no significant difference in the quality of that solution.

As will be discussed in detail in chapter 3, the results of these experiments cast some doubt upon the validity, in general, of the building block hypothesis. It was shown that the building blocks sampled most regularly by the GA during function optimisation did not combine - that the fragments of solutions created through the combination of pairs of highly sampled building blocks did not yield a significantly greater improvement in utility than did the addition of randomly generated building blocks to those highly sampled building blocks.

If there is doubt in the validity of the building block hypothesis, if there is doubt that the GA utilises the most highly sampled building blocks to construct the solution upon which it settles, then there is doubt in the quality of the information contained within these highly sampled building blocks. The potential use of such building blocks for the justification of the solutions upon which the GA converges and as a rich information source for the user is compromised by this lack of confidence in the information they contain.

1.2 The Divided Nature of This Research

As has been stated, the initial aim of this research was to develop a technique for the extraction of those building blocks sampled most highly by the GA during the search process and to exploit these building blocks in the justification of the solutions upon which the GA converges and in the dissemination to the user of the information they contain. The success of the initial phase of this research unfortunately indicated that the proposed latter phases would not prove possible.

Having shown that the building block hypothesis does not hold, in general, for the GA as a function optimiser, it was clearly desirable to develop an alternative account for the success of the GA in this role. A new research goal was therefore to identify the search mechanism utilised by the GA. With little confidence in the value of building blocks as a meaningful source of information, it was necessary that this mechanism should not rely upon the notion of building block. Furthermore, it was desired that this mechanism should also account for the success of many of the variants of the GA, for some of which the notion of building block is essentially meaningless.

Of the latter phases of the proposed research project, it was recognised that the development of an abstract search space visualisation technique could proceed without the use of building blocks. Rather than developing a technique by which the information contained within the highly sampled building blocks could be presented meaningfully to the user, this phase of the research sought to develop a technique by which the essential nature of the search space could be identified and presented clearly to the user. To this end, this technique was to involve the meaningful graphical representation of the possibly high dimensional search space.

1.3 A Brief Overview of this Thesis

There follows a short description of the content of each of the following chapters of this thesis. The goals, methodology and results of the three phases of this research are described in brief.

1.3.1 Chapter 2, The Genetic Algorithm

In its simplest form the GA consists of three major operations - those of crossover, mutation and selection. The crossover operator is often referred to as a recombination operator since it combines pairs of trial solutions to generate new trial solutions. The combination and recombination of building blocks described by the building block hypothesis relies upon the crossover operator and its effect upon the trial solutions of the population. The mutation operator results in small random changes to individual trial solutions, maintaining diversity in the population as a whole. The selection process ensures that the better solutions of the population are more likely to survive and prosper than the poorer solutions. The propagation of good building blocks is, by the building block hypothesis, a result of this operation. The crossover-selection cycle of the GA is therefore generally considered to be the most important feature of the GA search process.

These operations are described in detail within this chapter along with many of their more popular variants. The nature of these operators is further illustrated through examples and pseudo-code. The theoretical foundations of the GA provided by the Schema Theorem and building block hypothesis are discussed. Finally, a number of practical applications of the GA are described - illustrating the power of the GA as an optimisation technique.

1.3.2 Chapter 3, An Empirical Analysis of the Building Block Hypothesis

This chapter documents the initial phase of the original research plan of this thesis - the development of a technique for the on-line extraction of highly sampled building blocks. This technique, utilising a clustering algorithm to improve computational efficiency, is shown to effectively identify only those building blocks which are most regularly sampled by the GA. A number of measures of the utility of a building block are proposed and the relationships between these measures examined.

Upon completion of the building block extraction technique, the first research task was to use the building blocks identified during the application of the GA to a number of test functions to examine the validity of the building block hypothesis. As has been described above, this validation of the building block hypothesis was based upon an examination of the relative effect of combining pairs of high utility building blocks with both high utility building blocks and randomly generated building blocks. One of the proposed measures of utility is used to compare the relative improvement which results from these combinations. The results of these experiments show that combined pairs of high utility building blocks do not yield a significantly greater improvement in utility than that resulting from the combination of the high utility building blocks with the randomly generated building blocks.

The results of the research documented in this chapter cast doubt onto the validity of the building block hypothesis, and, as has been noted, resulted in the necessity of a revised research plan. Without the justification of the building block hypothesis, the use of building blocks as a justification of the solutions generated by the GA or as a source of information regarding the problems to which the GA is applied was clearly inappropriate.

1.3.3 Chapter 4, An Alternative Description of the Action of Crossover

The building block hypothesis suggests that the principal operators of the GA are those of crossover and selection, by which building blocks are combined, assigned samples according to their utility and subsequently recombined. The results of the previous chapter suggest that the building block hypothesis is not a sufficient account of the success of the GA as a function optimiser.

The research presented in this chapter describes an alternative account of the mechanics of the crossover operator. This account is based upon the geometric effect of a wide range of crossover operators upon the trial solutions maintained by the GA. Despite the fact that this geometric description of the crossover operator is an approximation, it is shown that for a large number of crossover events it is a sufficiently accurate approximation to warrant further investigation. From this description, short and long term performance models of the crossover-selection cycle of the GA are constructed for a simple quadratic function. The accuracy of these models is confirmed with experimental evidence and it is proposed that their accuracy supports the geometric description of crossover.

Having confirmed the validity of this geometric description of crossover, an alternative account of the mechanics of the GA is based upon it and from this account parallels are drawn between the GA and simulated annealing - another, similarly successful, global optimisation technique.

1.3.4 Chapter 5, Scientific Visualisation

Scientific visualisation covers a wide range of techniques for the graphical representation of data, the principal aims of which are to communicate information effectively to the user. This chapter describes a number of techniques for the visualisation of data, from the simple graph to complex iconic scatter diagrams. Particular attention is paid to the difficulty in representing high dimensional surfaces, such as are generated by scalar valued functions of many parameters. Examples of such functions are provided by parametric models of complex systems. The visualisation of the surfaces defined by such models can potentially provide a great deal of information concerning the systems under analysis, although there are few visualisation techniques capable of operating within this domain.

1.3.5 Chapter 6, A Technique for the Visualisation of High Dimensional Surfaces

This chapter describes a technique for the visualisation of high dimensional surfaces. This technique takes the form of a non-linear multivariate regression of the data generated by functions of many parameters. The form of this regression is similar to that of the statistical technique of Projection Pursuit Regression (Huber, 1985), although the techniques differ fundamentally in their construction of the regression surface. The technique described within this chapter is compared with Projection Pursuit Regression upon a wide range of illustrative test functions, an the relative merits of the two approaches are discussed.

1.3.6 Chapter 7, Conclusions

The final chapter of this thesis reiterates the conclusions of each of the above chapters. The research project as a whole is discussed and conclusions are drawn upon this research in the light of the original goals.

2 The Genetic Algorithm

The Genetic Algorithm (GA) is a heuristic search technique based upon the principles of natural selection (Holland, 1975). This thesis will treat the GA wholly as a function optimisation technique, although there are many other applications - most notably in learning classifier systems (Goldberg, 1989).

Traditional optimisation techniques are generally intended for local optimisation. Although they are guarantied to converge upon local optima (those points which are better than their neighbours), there is no satisfactory way of extending this property to global optima (those points which are better than any others). Thus for noisy or multimodal surfaces (with potentially many local optima), these techniques are unlikely to find satisfactory solutions. Some methods utilise gradient information to improve the convergence properties of the search process. When the first derivatives aren't available, they may be approximated with the first differences, although this results in greater computational expense. These techniques are likely to fail upon discontinuous surfaces, where derivatives do not exist at all points.

The GA differs from these traditional techniques on a number of counts. Firstly, the GA operates upon a large number of samples simultaneously. The GA also utilises randomness during the search process. This does not, however, imply that the GA is a random search method. The combination of large samples and carefully used randomness results in an effective global optimisation technique, capable of dealing with those domains which cause the greatest problems for traditional optimisation techniques.

2.1 Terminology

The terminology of the GA draws heavily from that of biological genetics. There follows a brief description of the most commonly used terms together with their equivalent mathematical terms.

Chromosome : Trial, sample

The chromosome consists of a series of concatenated (generally) binary numbers, each number representing a single parameter of the objective function.

e.g. $110010101101 \rightarrow (1100_2, 1010_2, 1101_2) = (12, 10, 13)$

The resulting set of integers may then be scaled to lie within a given set of bounds. The integers in the above example may be scaled onto the unit cube with a resolution of 2^{-4}.

e.g. $(12,10,13) \rightarrow 2^{-4}(12,10,13) = (0.7500, 0.6250, 0.8125)$

In order to increase the dimension or resolution of the parameter set, it is therefore necessary to increase the length of the chromosome.

Gene : Element

Each binary digit of the chromosome is known as a gene.

Allele : Value

The values taken by a given gene are known as the alleles of that gene. In the case of binary chromosomes, the genes have alleles 0 and 1.

Locus : Locus

The position of a gene within the chromosome is known as the locus of the gene. In general this is fixed and the gene may be define purely in terms of its locus. For example, the 0 allele is present in the gene at locus 3 of the above chromosome.

Population : Sample set

The current set of chromosomes is known as the population of the GA.

Fitness : Objective

The value of the objective function applied to the parameter set associated with a chromosome is known as the fitness of that chromosome. The objective function is generally known as the fitness function.

e.g. Given the fitness function $f(\mathbf{x}) = |\mathbf{x}|^2$, the fitness of the above chromosome is

$$f = 0.7500^2 + 0.6250^2 + 0.8125^2 = 1.6100$$

Genotype : Binary representation

The binary form of a particular chromosome is known as the genotype of the chromosome.

Phenotype : Parameter set

The parameter set associated with a particular chromosome is known as the phenotype of the chromosome.

2.2 The Simple Genetic Algorithm

The structure of the simple GA (Goldberg, 1989) is shown in figure 2.1. There are five basic steps - initialisation, evaluation, selection, crossover and mutation. The iterative sequence of selection, crossover, mutation and evaluation is known as a generation.

```
procedure genetic_algorithm
begin
      gen := 0;
      initialise $P_{gen}$;
      evaluate $P_{gen}$;

      while (not stopping-condition) do
      begin
            select $P_{gen+1}$ from $P_{gen}$;
            gen := gen+1;

            crossover $P_{gen}$;
            mutate $P_{gen}$;
      end
end
```

Figure 2.1 The structure of the simple GA

2.2.1 Initialisation

The initialisation of the GA is usually achieved through random sampling, each gene of each chromosome within the population being assigned a randomly chosen allele. More sophisticated techniques involve seeding the initial population with previously or algorithmically determined chromosomes.

2.2.2 Evaluation

During the evaluation phase of the GA, each chromosome is decoded into a parameter set. The parameter set is passed to the fitness function, and its fitness is stored alongside the chromosome. Some more sophisticated evaluation techniques pre-process the raw fitness of the chromosomes before moving on to the selection phase of the GA.

2.2.3 Selection

The most common method of selecting a new population from the current population is Roulette Wheel Selection (De Jong, 1975). The new population is created from randomly chosen members of the current population. For each chromosome in the new population the probability of selecting a particular chromosome from the current population is given by,

$$P_{\text{selection}}(C) = \frac{f(C)}{\sum_{X \in P_t} f(X)}$$

where $f(C)$ is the fitness of a chromosome C, and P_t is the population at generation t. Figure 2.2 shows an algorithmic implementation of this technique.

```
procedure roulette_wheel
begin
    sum := 0;
    for i := 0 to popsize-1 do
    begin
        sum := sum + fitness($C_i$);
    end

    sum := random(0, sum);

    i := 0;
    while (sum >= 0 and i < popsize) do
    begin
        sum := sum - fitness($C_i$);
        i := i+1;
    end
    i := i-1;

    roulette_wheel := $C_i$;
end
```

Figure 2.2 Roulette Wheel Selection algorithm

This process may be compared with spinning a weighted roulette wheel for each member of the new population. The proportion of the wheel assigned to each chromosome in the current population is determined by that chromosomes contribution to the total fitness of the current population.

Thus those chromosomes of above average fitness are likely to reproduce more successfully than those of below average fitness. This ensures that the population tends to improve from one generation to the next. This technique cannot be used directly with fitness functions which return negative values. However, it is relative simple to ensure that this does not occur.

2.2.4 Crossover

Crossover is generally considered to be the principal operator of the GA. Crossover acts upon pairs of randomly chosen chromosomes, exchanging information between them. A random locus is then chosen, and for all subsequent genes the alleles of the chromosomes are exchanged.

e.g. $C_1 = 110010101101 \rightarrow (0.7500, 0.6500, 0.8125)$

$C_2 = 110110010100 \rightarrow (0.8125, 0.5625, 0.2500)$

Split chromosomes at locus 7 $\begin{matrix} C_1 = 1100101 - 01101 \\ C_2 = 1101100 - 10100 \end{matrix}$

Exchange alleles $\begin{matrix} C_1 = 1100101 - 10100 \\ C_2 = 1101100 - 01101 \end{matrix}$

Giving $\begin{matrix} C_1 = 110010110100 \rightarrow (0.7500, 0.6875, 0.2500) \\ C_2 = 110110001101 \rightarrow (0.8125, 0.5000, 0.8125) \end{matrix}$

These two new chromosomes then replace the parent chromosomes in the population. The proportion of the population selected for crossover is known as the crossover rate and is generally set at 60% (De Jong, 1975).

2.2.5 Mutation

Mutation is the second operator of the GA. The mutation operator acts upon single chromosomes chosen at random from the population. A random locus is selected, and the allele value of the gene at that locus is altered.

e.g. $C = 1100101110100 \rightarrow (0.7500, 0.6875, 0.2500)$

Select locus 5 $C = 1100\underline{1}0110100$

Mutate $C = 1100\underline{0}0110100$

Giving $C = 110000110100 \rightarrow (0.7500, 0.1875, 0.2500)$

This new chromosome replaces its parent in the population. The proportion of the total number of genes in the population selected for mutation is known as the mutation rate and is generally inversely proportional to the population size (De Jong, 1975).

2.2.6 The Stopping Condition

A number of criteria may be used to halt the GA search process. For example,

- The GA executes for a pre-set number of generations.
- The maximum or average fitness of the population reaches a pre-set target.
- The population converges (all chromosomes within the population are identical).

2.3 The Fundamental Theorem of Genetic Algorithms

The behaviour of the GA is described by Holland's Schema Theorem, or Fundamental Theorem of Genetic Algorithms (Holland, 1975). The Schema Theorem describes how binary patterns known as schemata propagate from generation to generation.

2.3.1 Schemata

A schema (Holland, 1975) is a pattern defining a set of chromosomes. Schemata are defined over the ternary alphabet {0, 1, #}, where the # represents a wildcard which may match either a 0 or 1.

e.g. The schema #100# matches the set {01000, 01001, 11000, 11001}

The order of a schema is defined as the number of 1's and 0's in the pattern. The schema in the above example has order 3. This is formally written as

$$o(H) = 3$$

Thus the greater the order of a schema, the lesser the size of the set of chromosomes it defines. More accurately $|\{C{:}\, C \text{ matches } H\}| = 2^{l-o(H)}$, where l is the length of the chromosome. (Henceforth, the set of chromosomes defined by a schema and the schema itself shall be treated as the same object when this is not ambiguous or misleading).

The defining length of a schema is defined as the distance between the first and last non wild character in the pattern. The schema in the above example has defining length 2, formally written as

$$\delta(H) = 2$$

The static fitness of a schema is defined as the mean fitness of the chromosomes within the set it defines.

$$f_{\mathrm{static}}(H) = \frac{\sum_{C \in H} f(C)}{2^{l-o(H)}}$$

The dynamic fitness, or fitness, of a schema is defined as the mean fitness of the chromosomes in the current population which match it.

$$f(H) = \frac{\sum_{C \in H \cap P_t} f(C)}{\left|\{C : C \in H \cap P_t\}\right|}$$

2.3.2 The Schema Theorem

Any given chromosome must clearly be an instance of 2^l schemata, since at each locus the chromosome may match a schema in one of two ways - either with the same binary digit or with a wild card. Each chromosome may be considered as being a representative of 2^l schemata, and the behaviour of the GA may therefore be explained as sampling large numbers of increasingly fit schemata and utilising this information to guide the search effort. Since there are 3^l possible schemata of length l, and only 2^l chromosomes, it may seem that the search for fit schemata is a more complex task than the search for fit chromosomes. However, it may be shown that within a population of n chromosomes, the GA usefully processes $o(n^3)$ schemata (Holland, 1975), a property known as the implicit parallelism of the GA.

Schemata of high defining length are likely to be disrupted through the action of crossover, and those of high order through mutation. More accurately, for a schema H having $m(H, t)$ copies of itself in generation t, the expected number of copies in generation t+1 is bounded by

$$\langle m(H,t+1)\rangle \geq m(H,t)\cdot\frac{f_t(H)}{f(P_t)}\left[1-p_c\frac{\delta(H)}{l-1}-p_m o(H)\right]$$

where $f_t\,(H)$ is the dynamic fitness of H in generation t, $f(P_t)$ is the mean fitness of the chromosomes in generation t, and p_c and p_m are crossover and mutation probabilities (Holland, 1975). This is the Fundamental Theorem of Genetic Algorithms.

2.3.3 The Building Block Hypothesis

Short, low order schemata of above average fitness may therefore be expected to receive exponentially increasing numbers of trials from generation to generation. Such short, low order schemata are known as building blocks and are fundamental to this account of the success of the GA. The building block hypothesis suggests that the behaviour of the GA may be explained as the combination, sampling and recombination of highly fit building blocks - the GA progresses toward the globally optimal solution through the combination of those features of solutions which are in some sense optimal.

2.4 Alternative GA Strategies

Within the framework of the GA it is possible to construct a great many algorithms. There are a range of alternative techniques for crossover, mutation, selection and parameter representation, some of which are described below.

2.4.1 The Chromosome

There are numerous methods for encoding parameter sets as chromosomes, some of the most important of which are described here.

Gray Coding

One of the major drawbacks of a binary representation is the Hamming Cliff (Hamming, 1950). This refers to the large difference in the binary strings representing similar numbers. For example consider the following chromosomes

$$\begin{aligned} C_1 &= 0111 \rightarrow 7 \\ C_2 &= 1000 \rightarrow 8 \end{aligned}$$

Although the change in the parameter is only one unit, it requires four separate changes in the chromosome to achieve it. In order to overcome this, Gray Coding may be employed (Hollstein, 1971) - a non-linear mapping of binary numbers employed in the communications field to reduce transmission error. To increase or decrease the parameter by one unit, it is necessary to change only one bit of the binary representation, as illustrated in table 2.1.

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

Table 2.1 Binary and Gray code of integers 0-9

```
procedure gray_to_binary
begin
    binary[n-1] = gray[n-1];

    for i := n-2 to 0 do
    begin
        binary[i] = binary[i+1] exclusive or gray[i];
    end

    gray_to_binary := binary;
end
```

Figure 2.3 Gray code to Binary conversion algorithm

e.g. To convert the four bit Gray coded number 1011 to binary.

Gray code	1101	Most significant bit is the same.
Current Binary	----	
Giving	1---	

Gray code	1101	Exclusive or bit 2 of the Gray code
Current Binary	1---	with bit 3 of the binary.
Giving	10--	

Gray code	1101	Exclusive or bit 1 of the Gray code
Current Binary	10--	with bit 2 of the binary.
Giving	100-	

Gray code	1101	Exclusive or bit 0 of the Gray code
Current Binary	100-	with bit 1 of the binary.
Giving	1001	

Real Coded GAs

It is not necessary to restrict chromosomes to binary alphabets. One commonly used representation is real coding in which the binary genes are replaced with real numbers.

e.g. $C = (0.7501, 0.6249, 0.8125)$

This technique has two main advantages - the parameter set may include any point (up to machine precision) and there are no Hamming cliffs. The Schema Theorem does not account for the success of real coded GAs since there are effectively an infinite number of alleles for each gene.

Real coded GAs have enjoyed a degree of success in application to real world problems. In particular L. Davis (1991) suggests that using the same coding method as is used by classical optimisation techniques already employed within a given domain allows the GA to be easily hybridised with these techniques. A simple form of hybridisation includes classical optimisation techniques as additional operators within the GA. This can dramatically improve the convergence properties of the GA upon the problem at hand (Davis, 1991).

The Structured GA

The structured GA (Dasgupta & McGregor, 1991) relies upon redundancy within the chromosome to improve search in difficult domains. The chromosomes represent a tree structure from which the parameter set is generated. High level genes may act as switches, activating or deactivating lower level genes. These lower level genes may also act as switches for still lower level genes. The activated genes at all levels determine the parameter set. Since the deactivated genes are reproduced along with the active genes population diversity is maintained and rapid change in the direction of the search effort is possible.

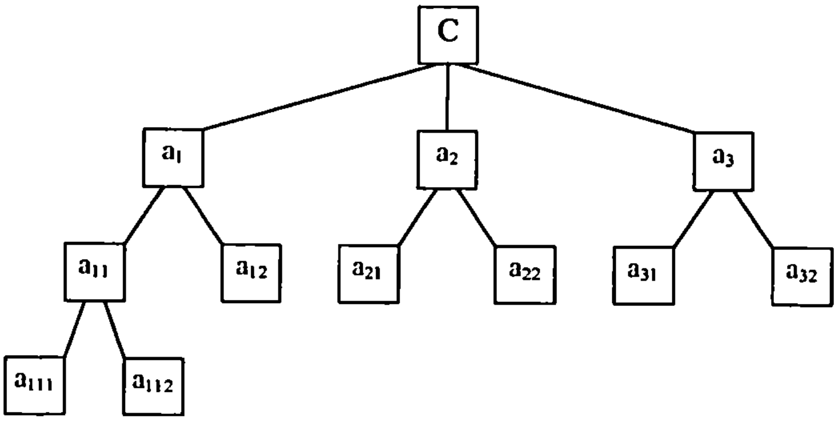


Figure 2.4 A simple structured chromosome

The chromosome in figure 2.4, for example, could be simply encoded as $C = (a_1, a_2, a_3, a_{11}, a_{12}, a_{21}, a_{22}, a_{31}, a_{32}, a_{111}, a_{112})$. The three first level genes (a_1, a_2, a_3) determine which of the second level genes are active and contribute toward the final parameter set. Similarly, the second level gene a_{11} determines which of the third level genes are active. The high level genes may determine the form of the solution, whilst the low level genes represent parameters consistent with the chosen form.

2.4.2 Initialisation

For heavily constrained optimisation problems, random sampling is highly unlikely to yield feasible solutions. In cases such as this it is often desirable to seed the initial population with previously determined feasible solutions. Population seeding does have drawbacks however, since the initial population may be dominated by a small number of comparatively fit chromosomes.

2.4.3 Evaluation

One of the principal problems in fitness evaluation is that of maintaining selection pressure - the difference in fitness between the best and worst members of the population. For example, consider the fitness function f and chromosome C,

$$f(C) = 0.95 \qquad\qquad f(P_t) = 0.475$$

Under roulette wheel selection C would be expected to receive $\frac{0.95}{0.475} = 2$ offspring. However, if the function f is transformed to f' where

$$f'(C) = f(C) + 100$$

the fitness of the chromosome and population are transformed to

$$f'(C) = 100.95 \qquad\qquad f'(P_t) = 100.475$$

The expected number of offspring of C under roulette wheel selection is now $\frac{100.95}{100.475} \approx 1$, although there is no qualitative difference between the two versions of the function. Clearly, for a robust optimisation technique, this is not a desirable property. Fortunately, there are a number of ways to combat this effect.

Windowing

The first and simplest technique is windowing (Grefenstette, 1991). This technique involves finding the worst member of the population and subtracting its fitness from the fitness of each chromosome. The worst member of the population therefore has fitness 0, with no chance of selection. This is not necessarily a positive feature, and so a minimum fitness threshold may be defined, to which all chromosomes of lower fitness are set. Windowing is the basic fitness transformation employed by GENESIS (Grefenstette, 1984).

Linear Normalisation

The second technique is linear normalisation (Davis, 1991). This involves ordering the population according to fitness and choosing a maximum fitness and fitness decrement. The best member of the population is assigned the maximum fitness, the next best the maximum fitness less one decrement, and so on. Again, a minimum threshold may be set to allow poor chromosomes some chance of reproduction.

Linear Scaling

A further technique is linear scaling (Goldberg, 1989). Firstly the desired ratio between the best and average fitnesses is chosen (equivalent to the desired number of offspring of the best chromosome). The fitness of the population is then scaled by

$$f' = af + b$$

subject to the constraint $f'(P_t) = f(P_t)$.

The constants *a* and *b* are easily calculated from the scaling factor and the best, worst and average fitnesses of the population. Under some circumstances, the resulting fitness of the worst chromosome may fall below 0 and, since the minimum fitness must always be 0, a new scaling factor must be chosen - generally the maximum scaling factor possible while maintaining positive fitness for all chromosomes.

2.4.4 Selection

There are several alternatives to roulette wheel selection, some of which are described here.

Elitism

Elitism (De Jong, 1975) guarantees that the best member of the population always survives. Before crossover, mutation and selection, a copy of the best member of the population is made. If this chromosome does not survive and is not improved upon, the copy replaces the worst (or a randomly selected) member of the next generation. Although not a selection technique in itself, elitism is often combined with a selection technique in order to guarantee that the best fitness of the population does not degrade.

Tournament Selection

Tournament selection (Brindle, 1981) involves choosing some predetermined number of chromosomes from the population at random, and placing the best of these in the next population. This process is repeated until the next population is filled. In general, two competitors are used and this is known as a binary tournament.

Stochastic Remainder Selection

Stochastic remainder selection (Booker, 1982) is a variant of roulette wheel selection which guarantees that a chromosome will receive at least the integer part of its expected number of offspring. The expected number of offspring under roulette wheel selection is calculated, and a number of copies equal to the integer part of this are placed in the next generation. The remaining fractional parts are then used with roulette wheel selection to fill the remaining places in the new population.

Steady State Reproduction

Steady state reproduction (Whitley, 1989) is a significant departure from the standard GA approach. The standard generation of selection, crossover and mutation is replaced and a pair of chromosomes are chosen from the population crossed over, mutated if some probability condition is met, and put back into the population - often replacing the worst chromosomes. Figure 2.5 illustrates the basic structure of the steady state GA.

2.4.5 Crossover

There exist a number of alternatives to the crossover operator, some of which are detailed below.

```
procedure steady_state_ga
begin
      gen := 0;
      initialise $P_{gen}$;
      evaluate $P_{gen}$;

      while (not stopping-condition) do
      begin
            i := random(0, popsize-1);
            j := random(0, popsize-1);

            copy i'th member of population to $C_0$;
            copy j'th member of population to $C_1$;

            crossover $C_0$ and $C_1$;

            if (random(0, 1) <= $p_m$) then mutate $C_0$;
            if (random(0, 1) <= $p_m$) then mutate $C_1$;

            copy $C_0$ to worst member of population;
            copy $C_1$ to second worst member of population;
      end
end
```

Figure 2.5 The structure of the steady state GA

Two Point Crossover

Two point crossover (Cavicchio, 1970) is, as the name suggests, a variant of crossover which utilises two crossover points. In this form of crossover all genes between the crossover points exchange allele values. This gives a more flexible exchange of information between the chromosomes. This can be generalised to *n* point crossover (De Jong, 1975) where *n* crossover points are chosen with genes being retained after even and exchanged after odd numbered crossover points (considering the start of the chromosome to be the 0'th crossover point).

e.g. $C_1 = 110010101101 \rightarrow (0.7500,0.6500,0.8125)$

$C_2 = 1101110010100 \rightarrow (0.8125,0.5625,0.2500)$

split chromosomes at loci 3, 7, 9
$$\begin{aligned} C_1 &= 110-0101-01-101 \\ C_2 &= 110-1100-10-100 \end{aligned}$$

Exchange alleles
$$\begin{aligned} C_1 &= 110-1100-01-100 \\ C_2 &= 110-0101-10-101 \end{aligned}$$

Giving
$$\begin{aligned} C_1 &= 110110001100 \rightarrow (0.8125,0.5000,0.7500) \\ C_2 &= 110010110101 \rightarrow (0.7500,0.6875,0.3125) \end{aligned}$$

Uniform Crossover

Uniform crossover (Syswerda, 1989) extends this notion still further. For each gene, the chromosomes retain or swap their allele values with equal probability. Uniform crossover is the limit of n point crossover as $n \rightarrow \infty$.

There exist a number of crossover techniques dedicated to real coded GAs. These techniques seek to bring the rich exchange of information possible with binary crossover to the real coded algorithms.

Average Crossover

Average crossover (Davis, 1991) is much like the standard binary form of crossover, except that at the crossover sites the parameters are averaged. This approximates the effect of binary crossover when the crossover point lies within the region of the chromosome associated with a given parameter.

e.g. $C_1 = (0.8125, 0.5000, 0.7500)$

$C_2 = (0.7500, 0.6875, 0.3125)$

Crossover at locus 2 $\begin{array}{l} C_1 = (0.8125, 0.59375, 0.3125) \\ C_2 = (0.7500, 0.59375, 0.7500) \end{array}$

Arithmetic Crossover

Arithmetic crossover (Michalewicz, 1993) bears little relationship to binary crossover, being a weighted averaging of the chromosomes.

$$\begin{array}{l} C_1 \rightarrow \alpha C_1 + (1-\alpha)C_2 \\ C_2 \rightarrow \alpha C_1 + (1-\alpha)C_1 \end{array}$$

Where α is a constant (randomly chosen) parameter of the crossover operator.

e.g. $C_1 = (0.8125, 0.5000, 0.7500)$

$C_2 = (0.7500, 0.6875, 0.3125)$

Crossover with $\alpha = 0.6$

$$\begin{array}{l} C_1 = 0.6 \cdot (0.8125, 0.5000, 0.7500) + 0.4 \cdot (0.7500, 0.6875, 0.3125) \\ C_2 = 0.6 \cdot (0.7500, 0.6875, 0.3125) + 0.4 \cdot (0.8125, 0.5000, 0.7500) \end{array}$$

Giving
$$C_1 = (0.7875, 0.5750, 0.5750)$$
$$C_2 = (0.7750, 0.6125, 0.4875)$$

2.4.6 Mutation

It is not possible to apply the binary mutation technique to real coded GAs. Therefore, mutation techniques have been developed which act upon real valued chromosomes.

Real Mutation

Real mutation (Davis, 1989) replaces the mutated gene with a randomly chosen value within the relevant bounds. Unlike binary mutation, no similarity remains between the parameter associated with the new gene and that of its parent.

Creep Mutation

Creep mutation (Davis, 1989) involves adding (or subtracting) a randomly chosen number. To preserve similarity with the parent, the random number may be chosen from a distribution which favours small numbers, such as the normal distribution.

2.5 Applications of the GA

The GA has been applied to problems from many domains, from machine learning to turbine design. A short description of some applications to design optimisation follow.

2.5.1 De Jong's Test Suite

Although not strictly an application of the GA, De Jong's suite of test functions has provided the basic testing ground for comparative analysis of the GA. De Jong sought to find a GA which was robust over a wide range of problem domains (De Jong, 1975). The suite of test function, f_1 to f_5 were chosen to encompass a variety of features such as continuity and discontinuity, smoothness and noise, and so forth.

Function	Limits	Chromosome Length (bits)	Dimension
$f_1(\mathbf{x}) = \sum_{i=1}^{3} x_i^2$	$-5.12 \le x_i < 5.12$	30	3
$f_2(\mathbf{x}) = 100\left(x_1^2 - x_2\right)^2 + (1 - x1)^2$	$-2.048 \le x_i < 2.048$	24	2
$f_3(\mathbf{x}) = \sum_{i=1}^{5} \mathrm{int}(x_i)$	$-5.12 \le x_i < 5.12$	50	5
$f_4(\mathbf{x}) = \sum_{i=0}^{30} i x_i^4 + N(0,1)$	$-1.28 \le x_i < 1.28$	240	30
$f_5(\mathbf{x}) = \left(0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} \left(x_i - a_{ij}\right)^6}\right)^{-1}$	$-65.536 \le x_i < 65.536$	34	2

Table 2.2 De Jong's test suite

The results of this study showed that a GA with a crossover rate of 60% and a mutation rate of 0.1% for a population of 100 chromosomes was the most robust over the test suite. The use of elitism was shown to improve general performance, although led to a degradation in one case (function f_5). This was attributed to elitism improving local search at the expense of global search.

Recently doubts have been raised as to the validity of this test suite for comparative analysis - GAs developed to perform well upon the test suite may be optimised for these functions alone, and thus are not robust. However, there exist no other satisfactory methods for analysing the performance of the GA and test functions, both De Jong's and others, provide the principal benchmark.

2.5.2 Two-Phase Supersonic Flow Nozzle Design

Klockgether and Schwfel (1970) applied Rechenberg's Evolution Strategy (ES) to the optimisation of a two-phase supersonic flow nozzle. The ES is a similar technique to the GA, although in its earlier forms it did not include crossover. The ES was applied by hand, using approximately 300 conically bored sections which could be placed together upon a test rig to form a flow nozzle which could then be empirically evaluated.

The resulting solution was a significant improvement upon former designs - an innovative solution consisting of two separate chambers.

2.5.3 Gas Pipeline Optimisation

Goldberg (1983) applied the GA to the optimisation of two gas pipeline models, a 10-compressor, 10-pipe steady state model and a single pipe transient control model (Wong & Larson, 1968).

In the steady state problem, the GA was used to minimise the power consumption of the pipeline, subject to pressure constraints, by changing the squared pressure difference across each compressor. In order to promote solutions lying within the constraints, Goldberg utilised a penalty function approach. If a solution violated the constraints, the fitness value of that solution would be penalised by the square of the deviation from the constraint. This was scaled so that it represented a significant fall in fitness. The GA achieved a result near to the optimum found with dynamic programming (Wong & Larson, 1968).

For the transient control problem, the GA was used to minimise the energy of compression, subject to pressure constraints, by adjusting the time dependant input flow. The constraints were handled in the same fashion as for the former problem. The input flow was discretised into 15 values and assumed to be linear between those values. Here also, the GA achieved near optimal results.

2.5.4 Pneumatic Water Engine Design

Parmee (1990) applied the GA to the optimisation of the pneumatic water engine (PWE), a renewable energy device for use in rivers. The PWE uses the upstream head to drive columns of water up and down in two cylindrical chambers, with the resulting air flow driving a turbine. In order to find the optimum chamber design the GA was applied to a mathematical model of the system. The chamber cross section was represented at 18 points, assuming linearity between these points. The solution discovered by the GA improved upon a uniform chamber by a significant 7.5% gain in average power output.

2.5.5 Structural Optimisation

Jenkins (1991) applied the GA to the minimisation of volume of an 18 bar cantilever truss, given a required load. The truss was described in terms of the co-ordinates of its nodes, and Goldberg's (1989) simple GA was used to minimise its volume.

In a similar study, Schwefel (1989) used the ES to attain a volume within 6% of the theoretical minimum. The GA slightly outperformed this, realising a volume within 5% of the minimum.

2.5.6 Digital Filter Design

Wade et al (1994) applied the GA to the design of linear phase FIR digital filters. By cascading simple filters, known as primitives, it is possible to build filters which conform to medium order frequency response specifications. A structured GA was applied to the problem of designing filters to fulfil frequency specification such as a relaxed version of the CCIR601-1 for digital television. The use of primitive filters, utilising only simple shift and add operations, yielded both a reduction in chip size and an increase in clock rate over traditionally designed filters.

Classical techniques are not practical for designing such filters, since computation time rises exponentially with the complexity of the required design. The structured GA was capable of working within this domain, generating near optimal solutions within the frequency constraints.

2.6 Simulated Annealing

The process of annealing in metallurgy involves the careful cooling of a substance to minimise the energy in the crystal lattice as it solidifies. To obtain the lowest energy state, and thus the most stable structure, it is necessary to start at a very high temperature and cool very slowly.

Simulated annealing (Kirkpatrick et al, 1983) abstracts this process to optimisation. If we consider a solution to be a state of the system, we can minimise the energy (objective function) through careful cooling.

The structure of the simulated annealing algorithm is shown in figure 2.6. The process begins with a random set of parameters - the initial state of the system. The initial temperature c_0 and transition length L_0 are chosen. At each step of the algorithm a random perturbation of the current solution is made. If this is an improvement over the current solution, it is automatically accepted and replaces the current solution. If it is not an improvement, it may be accepted under a probability condition known as the Metropolis criterion, as follows

$$P(\mathbf{x} \leftarrow \mathbf{x}') = e^{\frac{f(\mathbf{x})-f(\mathbf{x}')}{c_t}}$$

where $\mathbf{x}'$ is the current solution, $\mathbf{x}'$ the perturbed solution and c_t the current temperature. Thus at high temperatures it is likely that a poor solution will replace a good solution, and at low temperatures unlikely. Finally, new values for c_t and L_t are chosen. The method by which this choice is made is known as the cooling schedule and affects the performance of the algorithm significantly.

```
procedure simulated_annealing
begin
     initialise x, $L_0$, $c_0$;
     t := 0;

     while (not stopping-condition) do
     begin
          for l := 0 to $L_t$ do
          begin
               x' := random perturbation of x;

               if (f(x') <= f(x)) then x := x';
               else if metropolis(f(x), f(x'), $c_t$) then x := x';
          end

          t := t+1;

          calculate $c_t$ from $c_{t-1}$;
          calculate $L_t$ from $L_{t-1}$;
     end
end

procedure metropolis(f:scalar, f':scalar, c:scalar)
begin
     if (exp((f-f')/$c_t$) > random(0, 1)) then metropolis := true;
     else metropolis := false;
end
```

Figure 2.6 The structure of the simulated annealing algorithm

The behaviour of the simulated annealing algorithm has much in common with the GA. Both techniques perform well upon those domains which cause the greatest difficulty for traditional optimisation methods. The ability of simulated annealing to degrade and thus pursue alternative optima gives it similar characteristics to the GA.

3 An Empirical Analysis of the Building Block Hypothesis

Much of the theoretical framework of the GA is based upon the Schema Theorem and building block hypothesis. The building block hypothesis proposes that the GA operates through the combination, selection and recombination of building blocks. It is suggested that good building blocks combine to form fit chromosomes, enabling them to propagate under selection. There have, however, been relatively few studies of this hypothesis.

```
procedure bit_climbing_algorithm
begin
    Generate initial string C[length];
    flag := true;

    while (flag) do
    begin
        flip_list := Permutation (1,...,length);
        oldC := C;

        for i:=0 to length-1 do
        begin
            newC := C;
            j = flip_list[i];
            newC[j] = 1 - newC[j];

            if (f(newC) > f(C)) then C := newC;
        end

        flag := f(C) > f(oldC);
    end
end
```

Figure 3.1 The structure of the Bit-Climbing Algorithm

The Bit-Climbing algorithm (Davis, 1991b) is a simple hill climbing technique utilising binary strings. The algorithm flips each bit of the string in random order. After each bit is flipped the resulting string is compared to its predecessor, if it represents an improvement it replaces the original string, otherwise it is rejected. If, after all of the bits in the string have been tested, there is no improvement the algorithm halts, otherwise a new order of bits is generated and the process continues. The structure of the algorithm is illustrated in figure 3.1.

Davis observed that Bit-Climbing algorithm significantly outperformed the GA upon the functions in De Jong's test suite. From these results it was concluded that the functions in the test suite were not sufficiently difficult to require the power of high order building block combination and therefore warrant the use of the GA. Modified versions of the test functions were shown to present greater difficulty for the Bit-Climbing algorithm, and it was proposed that these functions required the combination of higher order building blocks for their solution.

The Royal Road fitness functions (Forrest & Mitchell, 1992) were proposed as a test of the building block hypothesis. The function is evaluated by comparing the binary string to a number of predetermined schemata, each of which has an associated reward. The fitness of the string is equal to the sum of the rewards of each of the building blocks it matches. Thus, the royal road functions promote the survival of these building blocks. These functions were principally used to examine the propagation of building blocks in the populations of the GA. However, the GA was compared to a selection of hill-climbing algorithms upon these functions and it was found that a variant of the Bit-Climbing algorithm, which picked a random bit to flip at each stage, outperformed the GA.

Both of these studies suggest that the manipulation of building blocks is not a necessary feature of the GA during function optimisation. However, as a test of the building block hypothesis, the evidence generated by these experiments is somewhat circumstantial. This chapter will investigate the validity of the building block hypothesis by examining directly the effect of combining pairs of good building blocks. A technique for the isolation of such building blocks is described, enabling the building block hypothesis to be tested empirically. This technique utilises cluster analysis to reveal similarities between chromosomes.

3.1 Cluster Analysis

Clusters are loosely defined as sets of data which are in some sense similar to each other. A clustering of a set of data divides the data into a number of clusters, where the members of each cluster are similar to other members of the same cluster and dissimilar to members of other clusters. A cluster analysis algorithm, or clustering algorithm, generates such a set of clusters. This set of clusters forms a partition of the data, in which each datum belongs to one, and only one, cluster. Figure 3.2 illustrates such a clustering, each datum being marked with a number to illustrate which cluster it is a member of.

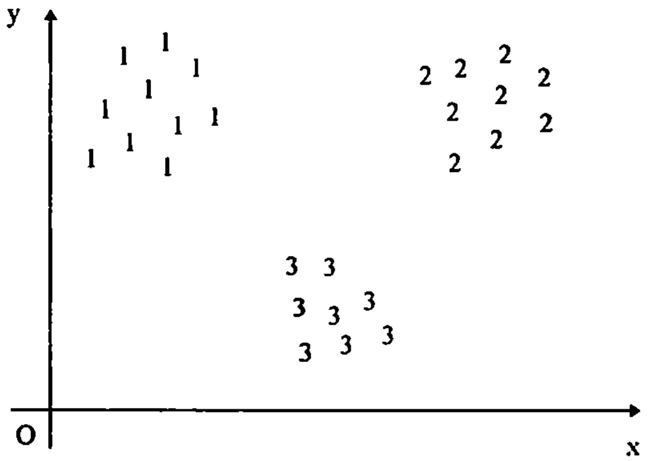


Figure 3.2 An example of a set of clusters

Clustering algorithms may be used for a number of purposed, from classification to data reduction. There are many techniques for clustering data, all of which rely upon some similarity measure (or conversely, and equivalently, distance measure) to determine whether or not a pair of data should belong to the same cluster. For example, for the clusters in figure 3.2, Euclidean distance is used as a similarity measure.

Clustering algorithms are designed to highlight the natural clusters present within a given set of data. If no such clusters are present, the results of the algorithm are likely to be misleading. Furthermore, many clustering algorithms have a predisposition to globular, or roughly spherical, clusters. Once again, misleading results are likely if the data contain non-globular clusters. Figure 3.3 illustrates the difference between these two types of cluster.

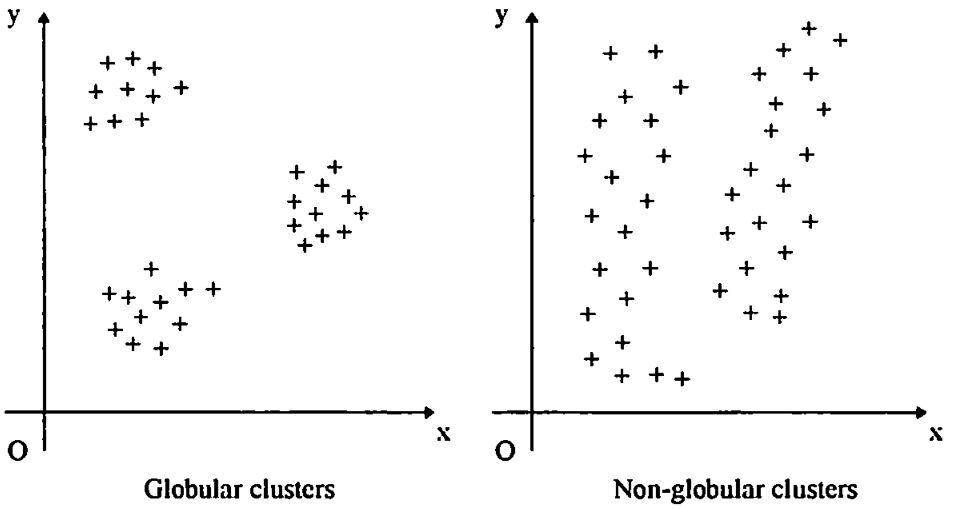


Figure 3.3 Globular and non-globular clusters

One of the great difficulties in the generation of a set of clusters is that there is no rigorous definition of cluster. Many techniques seek to minimise the difference between members of the same cluster whilst simultaneously maximising the difference between members of different clusters. In fact, this approach provides the only measure of the quality of a particular clustering - the ratio of the distance between members of the same cluster and the distance between members of different clusters. The smaller the value of this ratio, the better the quality of the clustering. Obviously, such a measure is likely to favour globular clusters.

3.1.1 The Shared Near Neighbours Algorithm

The shared Near Neighbours algorithm (Jarvis & Patrick, 1973) utilises a measure of similarity based upon the number of nearest neighbours common to a pair of data. Specifically, the pair of data are clustered together if and only if they share k_t (the threshold value) of their k nearest neighbours and are themselves members of each others *k* nearest neighbours.

```
procedure shared_nearest_neighbours
begin
    for i := 0 to n-1 do
    begin
        for j := 0 to k do
        begin
            N[i][j] := j'th nearest neighbour of datum i;
        end

        label_table[i] := i;
    end

    for i := 0 to n-1 do
    begin
        for j := i+1 to n-1 do
        begin
            A := $\{a : a \le k, \exists b \le k \text{ s.t. } N[i][a] = N[j][b]\}$ ;

            cond_1 := $\exists l, m \le k$ s.t. N[i][l] = j $\wedge$ N[j][m] = i;
            cond_2 := $\overline{A} \ge k_t$ ;

            if (cond_1 and cond_2) then
            begin
                for l := i to n-1 do
                begin
                    label_table[j]=label_table[i];
                end
            end
        end
    end
end
```

Figure 3.4 The structure of the Shared Nearest Neighbours algorithm

The structure of the Shared Nearest Neighbours algorithm is illustrated in figure 3.4. Note that $\overline{A}$ denotes the cardinal number (number of elements) of a set A.

This technique has a number of advantages. Firstly, it is capable of generating non-globular clusters if the data is so distributed. Few techniques are capable of generating such clusters, and as noted above, can therefore yield misleading results. Furthermore, the similarity measure scales itself to the data. For example, if the data is spread out, the k_t nearest neighbours of a datum will occupy a large region of the space. Conversely, if the data is densely distributed, the k_t nearest neighbours will occupy a relatively small region of the space. If the data is sparse in some regions and dense in others, the similarity measure will effectively perform a local scaling of the data surrounding each datum being tested. Many algorithms rely upon a global distance measure, with the result that sparse data are often split into many clusters if the data is densely distributed elsewhere. This is not a desirable feature and complex scaling algorithms are often employed to avoid this. The automatic scaling provided by the nearest neighbours similarity measure neatly solves this problem.

When applied to the populations of the GA, Hamming Distance is used to construct the distance matrix. This provides a good measure of the similarity between a pair of binary chromosomes.

3.1.2 A Worked Example of the Shared Near Neighbours Algorithm

Consider the following set of chromosomes.

1:	11001000	6:	01000101
2:	10001011	7:	10001011
3:	00100011	8:	01110110
4:	01110000	9:	10100100
5:	00001010	10:	10101101

Hamming distance is used to construct a distance matrix for these chromosomes. From this, choosing k=4, k_t=3, the neighbour table is constructed. Table 3.1 shows the neighbour table for this set of chromosomes.

n'th nearest neighbour				
0th (datum)	1st	2nd	3rd	4th
1	2	5	7	4
2	7	4	1	2
3	2	5	7	3
4	8	1	3	6
5	2	7	1	3
6	1	3	4	8
7	2	5	1	3
8	4	3	6	9
9	10	1	3	4
10	9	2	7	1

Table 3.1 The neighbour table N

The final step is to initialise the label table L.

L	1	2	3	4	5	6	7	8	9	10

For the sake of brevity, only those steps which directly effect the label table will be taken.

Datum 1 and datum i within k nearest neighbours of each other: {2, 4, 5, 7}

Datum 1 and datum i share k_t of k nearest neighbours: {2, 3, 5, 7,10}

Cluster with datum 1: {2, 5, 7}

Update label table:

L	1	1	3	4	1	6	1	8	9	10

Datum 2 and datum i within k nearest neighbours of each other: {3, 5, 7}

Datum 2 and datum i share k_t of k nearest neighbours: {3, 5, 7, 10}

Cluster with datum 2: {3, 5, 7}

Update label table:

L	1	1	1	4	1	6	1	8	9	10

Datum 4 and datum i within k nearest neighbours of each other: {6, 8}

Datum 4 and datum i share k_t of k nearest neighbours: {6, 8}

Cluster with datum 4: {6, 8}

Update label table:

L	1	1	1	4	1	4	1	4	9	10

Datum 9 and datum i within k nearest neighbours of each other: {10}

Datum 9 and datum i share k_t of k nearest neighbours: {10}

Cluster with datum 9: {10}

Update label table:

L	1	1	1	4	1	4	1	4	9	9

The algorithm has thus generated a partition of three clusters

{ {1, 2, 3, 5, 7}, {4, 6, 8}, {9, 10} }

{1, 2, 3, 5, 7}	{4, 6, 8}	{9, 10}
11001000	01110000	10100100
10001011	01000101	10101101
00100011	01110110	
00001010		
10001011		

3.2 Building Block Extraction

The empirical analysis of the building block hypothesis requires the identification and extraction of the relevant building blocks, namely those which are of high utility, from the populations of the GA. Those building blocks which are of above average utility should, by the Schema Theorem, be sampled more regularly than those which are not. Therefore, the simplest and perhaps most accurate measure of the utility of a building block is the number of times it is sampled by the GA. The extraction of building blocks from the populations of the GA is therefore restricted to those which are sampled most regularly by the GA.

Given a cluster of chromosomes κ_j, let the schema H_{κ_j} be the unique schema of highest order which matches each and every member of the cluster. This schema may be defined by

$$H_{\kappa_j}[i] = \begin{cases} a & \text{if } \forall C \in \kappa_j, C[i] = a \\ \# & \text{otherwise} \end{cases}$$

where κ_j is the$j$'th cluster and C represents a chromosome.

For example, consider the clusters of chromosomes from the above worked example of the Shared Nearest Neighbours algorithm.

{1, 2, 3, 5, 7}	{4, 6, 8}	{9, 10}
11001000	01110000	10100100
10001011	01000101	10101101
00100011	01110110	
00001010		
10001011		

Tracing down a column of chromosomes reveals which genes hold a single allele value throughout a cluster, and those which do not. In the first case, the schema associated with the cluster in question takes the same value and in the second it takes a wildcard. The schemata associated with these clusters are therefore,

{1, 2, 3, 5, 7}	{4, 6, 8}	{9, 10}
###0#0##	01##0###	1010#10#

This technique therefore enables the extraction of the more commonly sampled schemata from the populations of the GA with relatively little computational expense. These schemata may then be further processed to extract from them the low defining length building blocks which are of particular interest. To this end let $\Theta(H,i,j)$ be the set of schemata defined by,

$$\Theta(H,i,j) = \{H' : H' \supseteq H, H' \text{ a schema}, H'[k] = \# \, \forall k \in (i,j)\}$$

This set therefore contains the schemata, non-wild only between genes *i* and *j*, implicitly represented by the schema *H*. For example, consider the schema 1010#10#,

$$\Theta(1010\#10\#, 0, 2) = \begin{Bmatrix} 101\#\#\#\#\#, 10\#\#\#\#\#\#, 1\#1\#\#\#\#\#, \#01\#\#\#\#\#, \\ 1\#\#\#\#\#\#\#, \#0\#\#\#\#\#\#, \#\#1\#\#\#\#\#, \#\#\#\#\#\#\#\# \end{Bmatrix}$$

From this set, any schemata which do not fit the definition of a building block are then discarded. The set then contains all of the building blocks implicit in the original schema. All that remains is to find a suitable definition of building block. The building block hypothesis defines a building block as a schema of low defining length and order. This is somewhat vague and so throughout this study a building block is defined as having a defining length of between 8% and 10% of the overall length of the chromosome and an order of at least 90% of the defining length (equating to between 7.2% and 10% of the number of bits in the chromosome, depending upon the defining length of the building block in question).

For example, let $\min o_\delta$ be the minimum order acceptable for a building block of defining length δ. For each schema *H* and each defining length δ, iterate through the loci *i*, constructing the set of schemata $\Theta(H,i,i+\delta)$, discarding those schemata of orders less than $\min o_\delta$. Figure 3.5 illustrates the building block extraction process, where l is the length of the chromosome and H_{κ_i} is the schema associated with cluster κ_i.

```
procedure building_block_extraction
begin
    generate clusters $\kappa_i$ of generation $P_t$;
    n := number of clusters;

    for i := 0 to n-1 do
    begin
        for $\delta$ := min$\delta$ to max$\delta$ do
        begin
            for j := 0 to $l-\delta$ do
            begin
                $\theta$ := $\Theta(H_{\kappa_i}, j, j+\delta)$;

                for k := 0 to $\bar{\theta}-1$ do
                begin
                    if ($o(\theta_k) \geq \min o_\delta$) then store $\theta_k$;
                end
            end
        end
    end
end
```

Figure 3.5 The structure of the building block extraction technique

It is not necessary to use the schemata associated with clusters of chromosomes to extract building blocks from the populations of the GA. The building block extraction technique can alternatively be applied directly to the chromosomes in each population of the GA. This guarantees that all of the building blocks present in the population are isolated. However, there are a great many such building blocks present in each population of the GA and therefore the extraction technique becomes excessively computationally expensive. The number of building blocks per chromosome is given by,

$$\sum_{\delta=\min\delta}^{\max\delta} \sum_{o=\min o_{\delta}}^{\delta+1} (l-\delta)\cdot{}^{\delta+1}C_o$$

since for each defining length δ there are l-δ distinct positions for the building block and for each position, defining length δ and order o of the building block there are ${}^{\delta+1}C_o$ combinations of wild and non-wild characters. The use of clustering reduces the computational expense whilst enabling the extraction of those building blocks of apparent highest utility from each population of the GA. Table 3.2 compares the mean number of samples made by the GA during a single run of building blocks extracted using the chromosomes and the schemata associated with the clusters of chromosomes. The test function f_4 is omitted since the chromosome length is too large to permit the extraction of building blocks directly from the chromosomes. Although the experiment was performed upon the building blocks extracted during a single run of the GA, the sheer number of these building blocks ensures that the results are statistically meaningful.

	Mean number of samples of building blocks		
Function	Extracted with clustering	Extracted without clustering	Not extracted with clustering
f_1	14.6	8.7	5.8
f_2	14.7	11.3	7.9
f_3	9.1	3.6	2.1
f_4	-	-	-
f_5	14.1	7.4	4.7

Table 3.2 A comparison of the number of samples of extracted building blocks

The parameters for the Shared Nearest Neighbours clustering algorithm for this experiment were k=6, k_t=4 and only those clusters with at least 3 members were considered for the building block extraction process. Clearly the mean number of samples made by the GA of those building blocks extracted with the use of clustering far exceeds the mean number of samples of those building blocks not extracted due to the use of clustering. If the number of samples of a building block is accepted as a good measure of utility, then the use of clustering is of clear benefit.

The building block extraction technique does rely to some extent upon which clustering algorithm is used. As has been discussed, the partitions generated by clustering algorithms can vary widely, and it is difficult to justify the use of one clustering algorithm over another. The convergence of the population will counteract the arbitrary nature of the clustering algorithm to some extent, although it must be noted that all results depend, to some degree, upon the selection of the Shared Nearest Neighbours clustering algorithm.

3.3 Building Block Analysis

Once extracted, the building blocks may be subjected to a number of experiments in order to test the validity of the building block hypothesis. Throughout the following experiments, the above definition of building block is used $(0.08l \leq \delta \leq 0.1l, o \geq 0.9\delta)$. As above, the Shared Nearest Neighbours algorithm is applied with parameters k=6 and k_t=4 and only those clusters with at least 3 members are considered for building block extraction.

The GA was applied to inverted versions of the functions in the De Jong test suite. It proved necessary to reduce the chromosome length for the function f_4 due to the vast number of building blocks extracted from the original 240 bit long chromosome. An alternative version of the function f'_4, was therefore defined with a chromosome length of 80 bits mapping to 10 parameters.

Function	Limits	Chromosome Length (bits)	Dimension
$f'_4(\mathbf{x}) = \sum_{i=0}^{10} i x_i^4 + N(0,1)$	$-1.28 \leq x_i < 1.28$	80	10

Of each unique building block discovered during the runs of the GA upon the test functions, a number of measures were made. Namely,

- The total number of trials made by the GA of the building block.
- The mean fitness of these trials.

- The number of unique trials made by the GA of the building block.
- The mean fitness of these unique trials.
- The mean fitness of a random sample of the building block.

The first measure, as asserted above, provides an indication of the utility of a building block to the GA. Hereafter, this measure shall therefore be referred to as the utility of the building block. The final measure, given a large enough sample of chromosomes, serves as an approximation to the static fitness of the building block (the mean fitness of every chromosome which matches the building block). Hereafter this measure shall therefore be referred to as the static fitness of the building block, and will be calculated with a random sample of 1000 chromosomes.

3.3.1 Linear Regression Analysis of Building Blocks

The static fitness of a building block provides an indication of the intrinsic fitness of that building block. If the building block represents a feature of the domain which is universally beneficial, the static fitness will be high. Conversely, if the building block represents a feature which is universally undesirable, the static fitness will be low. In general, however, the populations of the GA will not be spread uniformly about the search space. A regularly sampled building block may therefore be of only local utility with a correspondingly low static fitness. It is therefore of interest to examine the relationship between the static fitness and the utility of a building block, both to determine the suitability of static fitness as a utility measure and the extent to which the GA exploits this intrinsic fitness.

For each of the test functions three runs of the GA were performed, each to 50 generations, and the extracted building blocks collated. The relationship between the static fitness of the building blocks and their utility was examined with a linear regression. A regression of the static fitness against the utility of the building blocks was used to generate a linear model of the data, $y = ax + b$, where y is the static fitness and x the utility of the building blocks. The gradient, a, of this linear model shows the strength of the relationship between these measures. Table 3.3 summarises the results of this analysis.

Function	a	b	Pearson's r
f_1	0.0012	52.72	0.16
f_2	0.0573	3393.8	0.12
f_3	0.0043	27.44	0.36
f'_4	0.0012	120.96	0.06
f_5	0.0071	25.6	0.35

Table 3.3 Results of linear regression of static fitness against utility.

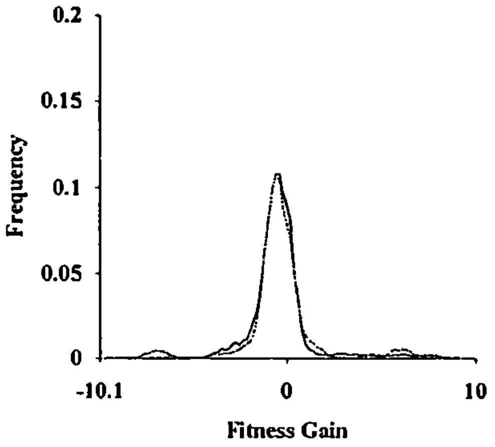
The value of Pearson's r for each of the experiments may be used to test the significance of the relationship predicted by the linear regression. For each of the test functions, the gradient a is significantly different from zero at at least the 0.1 level. This indicates that there is a definite relationship between the static fitness and the utility of the building blocks although, since the gradient is so small, this relationship is likely to be weak.

These results suggest that, for the test functions, static fitness may be used as a measure of the utility of a building block, although not a particularly good one. Furthermore, if the GA has exploited the static fitness of a building block then it must be highly sensitive to that fitness. It is likely, therefore, that static fitness has not played a role in the selection and propagation of building blocks.

3.3.2 Combinatorial Analysis of Building Blocks

The above results show that there is a definite, albeit weak, relationship between the utility and static fitness of a building block. This relationship may be exploited to enable the comparison of building blocks, some of which may not have been sampled by the GA during the optimisation process. Although static fitness is not a particularly good measure of utility, there is no other which may be used under these circumstances. For this reason, static fitness is used to examine the effect of combining pairs of high utility building blocks upon each of the test functions.

Upon each of the test functions, the GA was run for 50 generations and the 100 building blocks of highest utility were isolated. Each of these building blocks was combined with every other, except for those which were competing (whose associated sets of chromosomes were mutually exclusive). The static fitness of each resulting schema was calculated and the difference between this fitness and the greater static fitness of the pair of constituent building blocks was used as a measure of the effectiveness of that particular combination. As a control experiment, each of the 100 high utility building blocks was similarly combined with 100 randomly generated building blocks and the static fitness gain measured in the same fashion. For each of the test functions, the collated results of three runs of the GA are illustrated as histograms in figures 3.6 to 3.10. The solid lines show the results for the combination of high utility building blocks and the dashed lines those of the control experiments for each of the test functions.



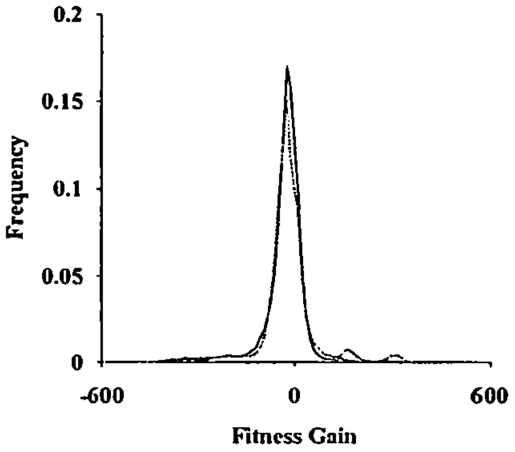
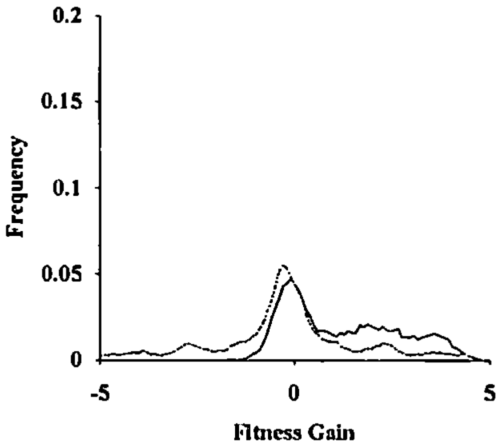


Figure 3.6 Combinatorial analysis on f_1

Figure 3.7 Combinatorial analysis on f_2



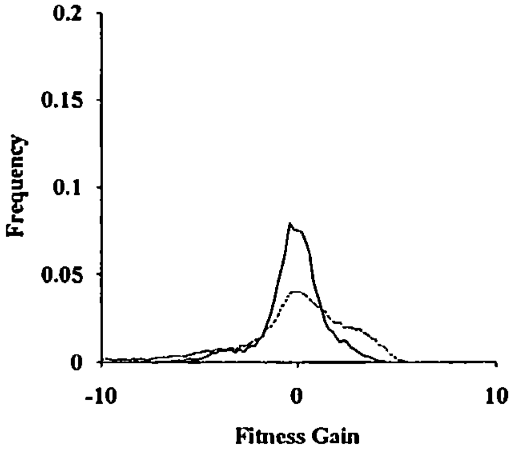


Figure 3.8 Combinatorial analysis on f_3

Figure 3.19 Combinatorial analysis on f'_4

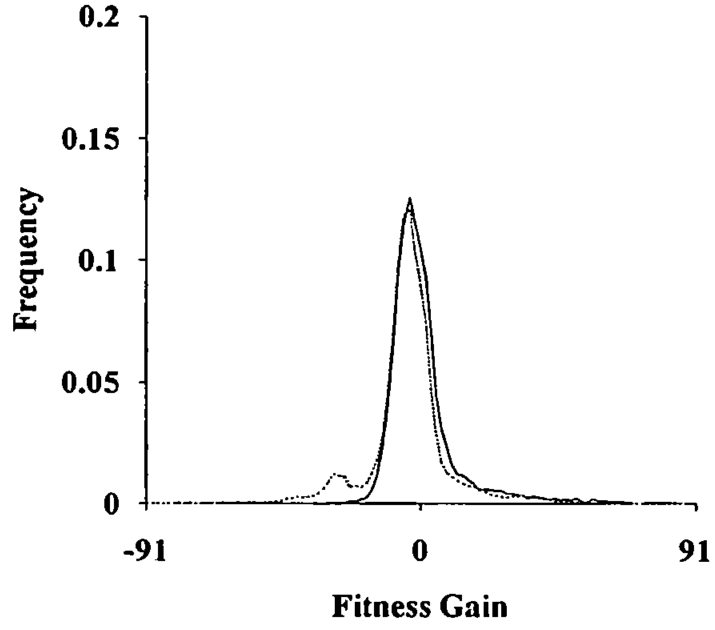


Figure 3.10 Combinatorial analysis on f_5

The distribution of fitness gain for the combined pairs of high utility building blocks does not differ greatly, in general, from that of the combination of high utility and random building blocks. The greatest differences in the distributions occur for the test functions f_3 and f'_4, although in the latter case the control experiment appears to yield a greater expected fitness gain. Except for the test function f_3, the distributions also appear to be fairly evenly distributed about zero.

Clearly, for the test functions at least, the combination of pairs of high utility building blocks does not yield a significantly different gain in fitness than that resulting from the addition of randomly generated building blocks to such high utility building blocks. It seems unlikely, therefore, that during the optimisation of the test functions the GA efficiently exploits these high utility building blocks.

3.4 Conclusions

The static building block hypothesis is an interpretation of the building block hypothesis in which high fitness is taken to mean high static fitness. This hypothesis maintains that the GA operates through the exploitation of building blocks of high static fitness, and is generally dismissed as an oversimplification of the building block hypothesis. However, the linear regression of the static fitness and utility of building blocks shows that these measures are related. The static building block hypothesis and the building block hypothesis must themselves therefore be similarly related.

The results of the combinatorial analysis of building blocks clearly shows that for the functions in the test suite, high utility building blocks do not combine to form fit schemata (any more than does the addition of random building blocks). This study therefore demonstrates that the GA cannot, in general, operate through the combination, selection and recombination of building blocks as predicted by the static building block hypothesis. The relationship between the static fitness of a building block and its utility indicates that this result may be generalised to the more general interpretation of the building block hypothesis.

The building block hypothesis implicitly assumes that the fitness of a chromosome is determined by the fitnesses of those building blocks from which it is constructed. The dynamic fitness of a building block is defined, by the Schema Theorem, in terms of the fitnesses of those chromosomes which match it in the current generation. The argument that the fitness of a chromosome depends upon the fitnesses of the building blocks it contains would therefore appear to be circular. By the building block hypothesis, the fitness of the chromosome is determined by the fitnesses of the building blocks which match it, which are themselves determined, at least in part, by the fitness of the chromosome itself. It seems unlikely, therefore, that these building blocks contain any useful information.

As an extreme example, consider a highly fit chromosome in a relatively unfit population. The building block hypothesis implies that this chromosome is fitter because the building blocks contained within it are themselves fit. However, since the rest of the population is relatively unfit, the fitnesses of these building blocks are almost entirely dependant upon this chromosome. Clearly, in this case the building block hypothesis merely asserts that the chromosome is fit because the chromosome is fit.

If a greater number of fit chromosomes are extant in the population, a similar result follows. A building block which is present within the fitter chromosomes is itself considered fit. Asserting that these chromosomes are amongst the fittest of the population because they contain this fit building block is equivalent to stating that this set of chromosomes is fit because this set of chromosomes is fit.

3.5 Further Work

This study has shown that the GA does not exploit building blocks during the optimisation of the functions in the test suite. It would be of advantage to apply this analysis to a wider class of test functions, enabling a more confident generalisation of the conclusions of this study. For example, the Royal Road functions are intended to encourage the exploitation of building blocks and would therefore provide an interesting domain in which to apply this analysis.

4 An Alternative Description of the Action of Crossover

The evidence presented in the previous chapter suggests that the GA does not utilise building blocks during function optimisation (at least for the functions in the test suite). An alternative account of the success of the GA is therefore required.

Much of the research into the behaviour of the GA has centred upon the transition from the chromosomes in the current population to the chromosomes in subsequent populations. One such approach is Markov chain analysis (Davis & Principe, 1993) which examines the probability of transition from any given state (population of chromosomes) to any other. Markov chain analysis has proved a very powerful technique for the description of the simulated annealing algorithm. Unfortunately, the GA typically has a very large state space (all possible populations), and although this analysis can provide a rigorous description of the GA, it is difficult to draw definite conclusions about its expected behaviour. An alternative approach is to study the expected transition from one population to the next upon a particular class of functions (Vose & Wright, 1994). Although such an approach yields more concrete results than those of the Markov chain analysis, these do not include the prediction of the performance of the GA.

This chapter will examine the geometric properties of the action of crossover and will thereby construct a predictive model of the crossover-selection cycle of the GA upon a simple function. This model is based upon the expected transition of the fitnesses of the chromosomes within the population rather than the chromosomes themselves. This approach simplifies the construction of the predictive model considerably.

This chapter will adopt the following notation conventions.

Notation	Meaning
X_i	Chromosome.
$\boldsymbol{x}_i$	Parameter set associated with X_i.
$X_i \times X_j, X_j \times X_i$	Offspring of chromosomes X_i and X_j under crossover.
$\mathbf{x}_i \times \mathbf{x}_j, \mathbf{x}_j \times \mathbf{x}_i$	Parameter sets associated with offspring of chromosomes X_i and X_j under crossover.

4.1 The Geometric Action of Crossover

The effect of the crossover operator upon the parameter sets associated with a pair of chromosomes closely resembles a crude rotation. The midpoint of the parameter set associated with the parent chromosomes is identical to that of the offspring chromosomes under a wide range of crossover operators. For example, consider the two following 8 bit chromosomes mapping to two dimensional space (4 bits per parameter)

$$
\begin{aligned}
&01101100 \rightarrow (6,12)\\
&10110010 \rightarrow (11,2)
\end{aligned}
$$

The unique pairs of offspring under one-point crossover and their associated parameter sets are therefore,

01101100 → (6,12)	00110010 → (3,2)	01110010 → (7,2)
10110010 → (11,2)	11101100 → (14,12)	10101100 → (10,12)
01100010 → (612)	01101010 → (6,10)	01101110 → (6,14)
10111100 → (11,12)	10110100 → (11,4)	10110000 → (11,0)

It is clear that the midpoints of all of the pairs of parameter sets are at (8.5, 7.0).

Figure 4.1 illustrates this with the pairs of parameter sets being joined by lines (the parameter sets associated with the parent chromosomes are marked with arrow heads). The similarity between crossover and rotation is highlighted with a dotted circle.

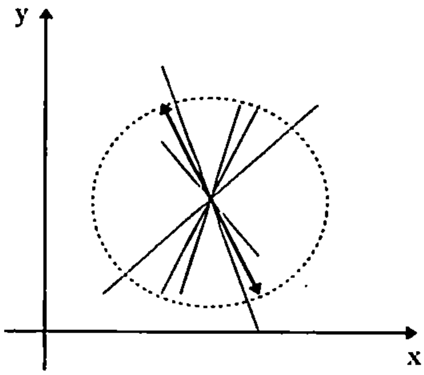


Figure 4.1 The geometric effect of crossover

Clearly the geometric effect of crossover would be better approximated by a rectangle bounded by the parent chromosomes. However, this lacks the mathematical elegance of the rotational description and has therefore not been adopted.

In fact, as the number of crossover points increase, the geometric effect of crossover is best approximated by a filled rectangle. To illustrate this, consider the uniform crossover technique (which may be considered as the limit of n point crossover as n tends to infinity), and the parent chromosomes,

$$\begin{aligned}01101100 &\rightarrow (0110,1100) \rightarrow (6,12)\\ 10110010 &\rightarrow (1011,0010) \rightarrow (11,2)\end{aligned}$$

Since the offspring under uniform crossover are produced by retaining or swapping allele values between the parents with equal probability, the effect of crossover upon the above chromosomes is identical to the effect of crossover upon the following chromosomes.

$$\begin{aligned}11111110 &\rightarrow (1111,1110) \rightarrow (15,14)\\ 00100000 &\rightarrow (0010,0000) \rightarrow (2,0)\end{aligned}$$

The offspring therefore lie within the rectangle bounded by the points (2,0) and (15,14). Not all points within this rectangle can be generated since some genes share the same allele value. For example the chromosome 11010001 cannot be an offspring of the parent chromosomes under uniform crossover (or indeed n point crossover).

This similarity between crossover and a rotation and scaling about the midpoint of the parent parameter sets in the above example does not occur by chance. It is simple to prove that for an arbitrary pair of chromosomes, they and their offspring under one point crossover must have the same midpoint in binary space. From this it follows that under certain conditions the midpoints of the pairs of parameter sets must also be equal.

THEOREM 4.1: *The midpoints of two chromosomes X_1 and X_2 before and after one point crossover are identical.*

$$X_1 + X_2 \equiv (X_1 \times X_2) + (X_2 \times X_1) \tag{4.1}$$

PROOF: *Consider the chromosomes X_1 and X_2,*

$$\begin{aligned} X_1 &= a_1, b_1, \ldots, p_1, q_1, \ldots, y_1, z_1 \\ X_2 &= a_2, b_2, \ldots, p_2, q_2, \ldots, y_2, z_2 \end{aligned}$$

$$\Rightarrow X_1 + X_2 = a_1 + a_2, b_1 + b_2, \ldots, p_1 + p_2, q_1 + q_2, \ldots, y_1 + y_2, z_1 + z_2$$

Without loss of generality, assume that the crossover point lies between genes p and q.

$$\begin{aligned} X_1 \times X_2 &= a_1, b_1, \ldots, p_1, q_2, \ldots, y_2, z_2 \\ X_2 \times X_1 &= a_2, b_2, \ldots, p_2, q_1, \ldots, y_1, z_1 \end{aligned}$$

$$\begin{aligned} \Rightarrow (X_1 \times X_2) + (X_2 \times X_1) &= a_1 + a_2, b_1 + b_2, \ldots, p_1 + p_2, q_2 + q_1, \ldots, y_2 + y_1, z_2 + z_1 \\ &= X_1 + X_2 \end{aligned}$$ □

REMARK: *This theorem applies equally to chromosomes defined over binary, n-ary and real alphabets since no assumption of alphabet has been made.*

COROLLARY: *The midpoints of a pair of chromosomes before and after n-point, uniform and average crossover are identical.*

COROLLARY: *This theorem holds for chromosomes representing n-dimensional binary numbers.*

COROLLARY: *Providing that the mapping from chromosome to parameter set is linear, this theorem holds for the parameter sets associated with the chromosomes.*

The simple mapping from chromosome to parameter is essentially a change of base and is therefore a linear mapping. However, Gray coding is an example of a non-linear mapping from the binary chromosome to the parameter space. The theorem does not therefore hold for crossover applied to Gray coded chromosomes. Henceforth, it is assumed that the mapping from chromosome to parameter set is linear.

Since the midpoint of the parameter sets associated with both the parent and offspring chromosomes are equal, the action of crossover upon a pair of chromosomes may be considered to be a rotation and scaling about the midpoint of their associated parameter sets. Using matrix notation,

$$\begin{aligned} \mathbf{x}_1 \times \mathbf{x}_2 &= s \cdot R\left(\mathbf{x}_1 - \bar{\mathbf{x}}\right) + \bar{\mathbf{x}} \\ \mathbf{x}_2 \times \mathbf{x}_1 &= s \cdot R\left(\mathbf{x}_2 - \bar{\mathbf{x}}\right) + \bar{\mathbf{x}} \end{aligned} \qquad (4.2)$$

where s is a scalar, R is a rotation and $\bar{\mathbf{x}}$ is the midpoint of the parent chromosomes.

Note that the values of s and R are dependant upon both the parent chromosomes and the crossover site(s).

4.2 A Performance Model for the GA

The construction of a performance model of the GA requires a further simplification to the rotational description of crossover. It shall be assumed that the action of crossover is well approximated, on average, by a pure rotation (equivalently, that the mean and variance of s in equation 4.2 are approximately equal to 1 and 0 respectively for a large number of crossover events). Although this is a strong assumption, it will be shown that it is not an invalid one.

Finally, the performance model will deal only with the crossover-selection cycle of the GA. The mutation operator is therefore omitted from the GA for the purposes of this study.

This description of the crossover operator will be used to predict the performance of the GA upon the simple function

$$f(\mathbf{x}) = |\mathbf{x}|^2, \mathbf{x} \in [0,1)^2 \qquad (4.3)$$

4.2.1 The Average Effect of Crossover

The first step in the construction of a predictive model for the GA is to calculate the expected effect of a crossover event upon the fitness of a pair of chromosomes.

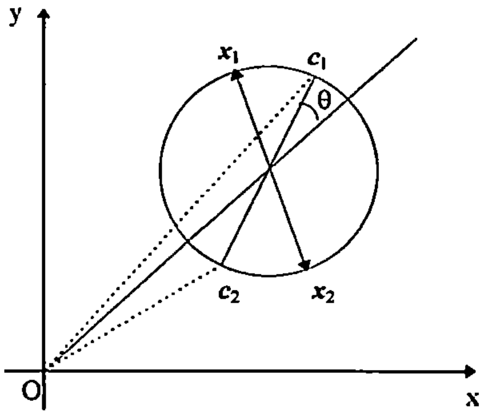


Figure 4.2 The average geometric effect of crossover

THEOREM 4.2: *Given the parent parameter sets x_1 and x_2, the expected fitnesses of the offspring parameter sets c_1 and c_2 are given by*

$$\begin{aligned}\langle f(\mathbf{c}_1)\rangle &= \frac{f(\mathbf{x}_1)+f(\mathbf{x}_2)}{2}+\frac{|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|}{\pi}\\ \langle f(\mathbf{c}_2)\rangle &= \frac{f(\mathbf{x}_1)+f(\mathbf{x}_2)}{2}-\frac{|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|}{\pi}\end{aligned} \tag{4.4}$$

PROOF: *From figure 4.2, applying the cosine rule*

$$\begin{aligned}|\mathbf{c}_1|^2 &= \frac{1}{4}|\mathbf{x}_1+\mathbf{x}_2|^2+\frac{1}{4}|\mathbf{x}_1-\mathbf{x}_2|^2+\frac{1}{2}|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|\cos\theta\\ &= \frac{1}{2}|\mathbf{x}_1|^2+\frac{1}{2}|\mathbf{x}_2|^2+\frac{1}{2}|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|\cos\theta\end{aligned} \tag{4.5}$$

Without loss of generality, choose $|\mathbf{x}_1|\ge|\mathbf{x}_2|, |\mathbf{c}_1|\ge|\mathbf{c}_2|, -\frac{\pi}{2}\le\theta\le\frac{\pi}{2}$

$$\Rightarrow \left\langle |\mathbf{c}_1|^2 \right\rangle = \frac{|\mathbf{x}_1|^2 + |\mathbf{x}_2|^2}{2} + \frac{1}{2\pi}|\mathbf{x}_1 + \mathbf{x}_2||\mathbf{x}_1 - \mathbf{x}_2| \int_{-\pi/2}^{\pi/2} \cos\theta d\theta$$

$$\left\langle f(\mathbf{c}_1) \right\rangle = \frac{f(\mathbf{x}_1) + f(\mathbf{x}_2)}{2} + \frac{|\mathbf{x}_1 + \mathbf{x}_2||\mathbf{x}_1 - \mathbf{x}_2|}{\pi}$$

Similarly for $\left\langle f(\mathbf{c}_2) \right\rangle$. □

This result may be illustrated with the chromosomes used to demonstrate the rotational effect of crossover - namely

$$\begin{aligned} X_1 &= 01101100 \\ X_2 &= 10110010 \end{aligned}$$

The parameter sets associated with these chromosomes may be scaled onto the unit square by dividing by 2^4. The fitness of the chromosomes is easily calculated from the parameter sets.

$$\begin{aligned} X_1 &= 01101100 \rightarrow \frac{1}{16}(6,12) = (0.375, 0.75) & \qquad f(\mathbf{x}_1) &= |\mathbf{x}_1|^2 = 0.7031 \\ X_2 &= 10110010 \rightarrow \frac{1}{16}(11,2) = (0.6875, 0.125) & \qquad f(\mathbf{x}_2) &= |\mathbf{x}_2|^2 = 0.4883 \end{aligned}$$

The fitnesses of all possible pairs of offspring under one point crossover are calculated in the same fashion. Table 4.1 catalogues the results, distinguishing between the greater and lesser fitnesses of each pair of offspring.

Finally, the expected fitnesses of the offspring may be calculated by theorem 4.2.

$$\begin{aligned}\langle f(\mathbf{c}_1)\rangle &= \frac{f(\mathbf{x}_1)+f(\mathbf{x}_2)}{2}+\frac{|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|}{\pi}\\ &= \frac{0.7031+0.4883}{2}+\frac{|(1.0625,0.875)||(-0.3125,0.625)|}{\pi}\\ &= 0.5957+\frac{1.3764\times 0.6988}{\pi}\\ &= 0.5957+0.3062=0.9019\end{aligned}$$

Similarly for c_2

$$\langle f(\mathbf{c}_2)\rangle = 0.5957-0.3062=0.2895$$

The expected fitnesses of the offspring given by theorem 4.2 agree well with the averages of the observed fitnesses of the offspring. Clearly for this example, the approximation of crossover as a pure rotation is justified.

Crossover Location	$f(\mathbf{c}_1)$ - greater fitness	$f(\mathbf{c}_2)$ - lesser fitness
1	1.3281	0.0508
2	0.9531	0.2070
3	0.9531	0.2070
4	1.0352	0.1563
5	0.5352	0.5313
6	0.9063	0.4727
7	0.7031	0.4883
8	0.7031	0.4883
Mean Fitness	0.8897	0.3252

Table 4.1 The fitnesses of pairs of offspring chromosomes

To test the theorem more thoroughly, this experiment was performed with 10,000 pairs of randomly generated 30 bit chromosomes (15 bits per parameter). The pairs of chromosomes were crossed over at each possible site, and the fitnesses of the resulting offspring were recorded. For each pair of chromosomes, the expected fitnesses of the offspring by theorem 4.2 were calculated. Finally, the error of the approximation for each pair of chromosomes was calculated.

Table 4.2 compares the mean expected fitnesses and the mean observed fitnesses for these 10,000 chromosome pairs. The RMS. error between the expected and observed fitnesses of the offspring of the 10,000 chromosome pairs is also presented here.

Offspring	Mean Observed Fitness	Mean Expected Fitness	RMS. Error
c_1 - greater fitness	0.9011	0.9036	0.0599
c_2 - lesser fitness	0.4232	0.4205	0.0584

Table 4.2 The mean expected and observed fitnesses of offspring

The results of this experiment confirm that the rotational description of crossover is, at least on average, accurate. This description will therefore be confidently used to provide the basis for a predictive performance model of the crossover-selection cycle of the GA.

4.2.2 A First Model

Having derived the average effect of crossover upon a pair of chromosomes, it is now necessary to combine this with the effect of roulette wheel selection upon the population.

Before the construction of the model, it is necessary to introduce some additional notation. Namely

$$\mu_r(P_t) = \frac{\sum_{\mathbf{x} \in P_t} f(\mathbf{x})^r}{N}$$

For integer r, this is equivalent to the r'th moment (about 0) of the fitnesses of the chromosomes in population P_t (the population at generation t). However, this notation will be also be used for real values of r.

LEMMA 4.1: *Under roulette wheel selection the expected mean fitness of population* $P_{t+1}, \mu_1(P_{t+1})$ *is given by*

$$\langle \mu_1(P_{t+1}) \rangle = \frac{\mu_2(P'_t)}{\mu_1(P'_t)} \qquad (4.6)$$

where P'_t *is the population immediately prior to selection in generation t.*

PROOF: *Under roulette wheel selection the expected number of copies of* x *in generation* t+1, $\langle m(\mathbf{x},t+1)\rangle$ *is given by*

$$\langle m(\mathbf{x},t+1)\rangle = \frac{f(\mathbf{x})}{\mu_1(P'_t)} \tag{4.7}$$

$$\Rightarrow \langle \mu_1(P_{t+1})\rangle = \frac{\sum\limits_{\mathbf{x}\in P'_t} \langle m(\mathbf{x},t+1)\rangle \cdot f(\mathbf{x})}{N} = \frac{\sum\limits_{\mathbf{x}\in P'_t} f(\mathbf{x})^2}{N\mu_1(P'_t)} = \frac{\mu_2(P'_t)}{\mu_1(P'_t)} \qquad \square$$

Clearly, it is therefore necessary to find the effect of crossover upon the squares of the fitnesses of a pair of chromosomes. This is achieved in a similar fashion to the proof of theorem 4.2.

LEMMA 4.2: *The expected squared fitnesses of the offspring parameter sets are given by*

$$\begin{aligned} \langle f(\mathbf{c}_1)^2\rangle &= \frac{(f(\mathbf{x}_1)+f(\mathbf{x}_2))^2}{4} + \frac{(f(\mathbf{x}_1)+f(\mathbf{x}_2))|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|}{\pi} + \frac{|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2}{8} \\ \langle f(\mathbf{c}_2)^2\rangle &= \frac{(f(\mathbf{x}_1)+f(\mathbf{x}_2))^2}{4} - \frac{(f(\mathbf{x}_1)+f(\mathbf{x}_2))|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|}{\pi} + \frac{|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2}{8} \end{aligned} \tag{4.8}$$

PROOF: *Taking equation 4.5 and squaring both sides*

$$f(\mathbf{c}_1)^2 = \frac{(|\mathbf{x}_1|^2+|\mathbf{x}_2|^2)^2}{4} + \frac{(|\mathbf{x}_1|^2+|\mathbf{x}_2|^2)|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|\cos\theta}{2} + \frac{|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2\cos^2\theta}{4} \tag{4.9}$$

Without loss of generality choose $|\mathbf{x}_1|\ge|\mathbf{x}_2|, |\mathbf{c}_1|\ge|\mathbf{c}_2|, -\frac{\pi}{2}\le\theta\le\frac{\pi}{2}$

$$\Rightarrow\left\langle f(\mathbf{c}_1)^2\right\rangle=\frac{\left(|\mathbf{x}_1|^2+|\mathbf{x}_2|^2\right)^2}{4}+\frac{1}{2\pi}\left(|\mathbf{x}_1|^2+|\mathbf{x}_2|^2\right)|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|\int\limits_{-\pi/2}^{\pi/2}\cos\theta\, d\theta$$

$$+\frac{1}{4\pi}|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2\int\limits_{-\pi/2}^{\pi/2}\cos^2\theta\, d\theta$$

$$=\frac{\left(f(\mathbf{x}_1)+f(\mathbf{x}_2)\right)^2}{4}+\frac{\left(f(\mathbf{x}_1)+f(\mathbf{x}_2)\right)|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|}{\pi}+\frac{|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2}{8}$$

Similarly for $\left\langle f(\mathbf{c}_2)^2\right\rangle$ □

It will be shown later that all of the $|\mathbf{x}_1\pm\mathbf{x}_2|$ terms cancel, except for the $\frac{|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2}{8}$ term in equations 4.8. It is therefore necessary to determine the expected value of this term.

LEMMA 4.3: $2\mu_1(P_t)^2-2\mu_2(P_t)\le\left\langle|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2\right\rangle\le 2\mu_2(P_t)+8\mu_{1.5}(P_t)\mu_{0.5}(P_t)+6\mu_1(P_t)^2$

PROOF: *N.B.* $|\mathbf{a}\pm\mathbf{b}|\le|\mathbf{a}|+|\mathbf{b}|, |\mathbf{a}||\mathbf{b}|\ge\mathbf{a}\cdot\mathbf{b}$

$$\begin{aligned}
\left\langle |\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2 \right\rangle &\geq \left\langle \left((\mathbf{x}_1+\mathbf{x}_2)\cdot(\mathbf{x}_1-\mathbf{x}_2)\right)^2 \right\rangle \\
&= \left\langle \left(|\mathbf{x}_1|^2-|\mathbf{x}_2|^2\right)^2 \right\rangle \\
&= \left\langle \left(f(\mathbf{x}_1)-f(\mathbf{x}_2)\right)^2 \right\rangle \\
&= \left\langle f(\mathbf{x}_1)^2-2f(\mathbf{x}_1)f(\mathbf{x}_2)+f(\mathbf{x}_2)^2 \right\rangle \\
&= \left\langle f(\mathbf{x}_1)^2 \right\rangle - 2\langle f(\mathbf{x}_1)\rangle\langle f(\mathbf{x}_2)\rangle + \left\langle f(\mathbf{x}_2)^2 \right\rangle \quad \begin{pmatrix}\text{Since } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are}\\ \text{independant samples}\end{pmatrix} \\
&= 2\mu_1(P_t)^2-2\mu_2(P_t)
\end{aligned}$$

$$\begin{aligned}
\left\langle |\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2 \right\rangle &\leq \left\langle \left(\left(|\mathbf{x}_1|+|\mathbf{x}_2|\right)^2\right)^2 \right\rangle \\
&= \left\langle |\mathbf{x}_1|^4+4|\mathbf{x}_1|^3|\mathbf{x}_2|+6|\mathbf{x}_1|^2|\mathbf{x}_2|^2+4|\mathbf{x}_1||\mathbf{x}_2|^3+|\mathbf{x}_2|^4 \right\rangle \\
&= \left\langle |\mathbf{x}_1|^4 \right\rangle + 4\left\langle |\mathbf{x}_1|^3 \right\rangle\langle |\mathbf{x}_2| \rangle + 6\left\langle |\mathbf{x}_1|^2 \right\rangle\left\langle |\mathbf{x}_2|^2 \right\rangle + 4\langle |\mathbf{x}_1| \rangle\left\langle |\mathbf{x}_2|^3 \right\rangle + \left\langle |\mathbf{x}_2|^4 \right\rangle \\
&= 2\mu_2(P_t)+8\mu_{1.5}(P_t)\mu_{0.5}(P_t)+6\mu_1(P_t)^2
\end{aligned}$$

□

Having taken the necessary preliminary steps, it is now possible to build the predictive model.

THEOREM 4.3: *The expected mean fitness of population* $P_{t+1}, \langle \mu_1(P_{t+1}) \rangle$ *is bounded by*

$$\left(1+\frac{p_c}{4}\right)\frac{\mu_2(P_t)}{\mu_1(P_t)}-\frac{p_c}{4}\mu_1(P_t) \leq \langle \mu_1(P_{t+1}) \rangle \leq \left(1+\frac{3p_c}{4}\right)\frac{\mu_2(P_t)}{\mu_1(P_t)}+\frac{p_c}{4}\mu_1(P_t)+p_c\frac{\mu_{0.5}(P_t)\mu_{1.5}(P_t)}{\mu_1(P_t)}$$

PROOF: *From theorem 4.2 and lemma 4.2*

$$\left(\langle f(\mathbf{c}_1)\rangle+\langle f(\mathbf{c}_2)\rangle\right)-\left(f(\mathbf{x}_1)+f(\mathbf{x}_2)\right)=0 \qquad (4.10)$$

$$\left(\left\langle f(\mathbf{c}_1)^2\right\rangle+\left\langle f(\mathbf{c}_2)^2\right\rangle\right)-\left(f(\mathbf{x}_1)^2+f(\mathbf{x}_2)^2\right)=\frac{|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2}{4}-\frac{\left(f(\mathbf{x}_1)-f(\mathbf{x}_2)\right)^2}{2} \qquad (4.11)$$

It is assumed that the population of the GA is sufficiently large for the average effect of crossover upon the fitness of the chromosomes to be well approximated by the effect of crossover upon the average fitness of the chromosomes. It is asserted that for sufficiently large populations, this is a good approximation.

Hence from equations 4.10 and 4.11, it is trivial to show that

$$\left\langle \mu_1(P'_t)\right\rangle=\mu_1(P_t) \qquad (4.12)$$

$$\begin{aligned}\left\langle \mu_2(P'_t)\right\rangle &= \mu_2(P_t)+\frac{p_c}{2}\left(\frac{\left\langle|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2\right\rangle}{4}-\frac{\left\langle\left(f(\mathbf{x}_1)-f(\mathbf{x}_2)\right)^2\right\rangle}{2}\right).\\ &= \mu_2(P_t)+\frac{p_c}{8}\left\langle|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2\right\rangle-\frac{p_c}{2}\left(\mu_1(P_t)^2-\mu_2(P_t)\right) \qquad (4.13)\end{aligned}$$

from the proof of lemma 4.3

From lemma 4.1

$$\begin{aligned}\left\langle \mu_1(P_{t+1})\right\rangle &= \frac{\mu_2(P'_t)}{\mu_1(P'_t)}\\ &= \frac{\mu_2(P_t)}{\mu_1(P_t)}+\frac{p_c}{8\mu_1(P_t)}\left\langle|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2\right\rangle-\frac{p_c}{2\mu_1(P_t)}\left(\mu_1(P_t)^2-\mu_2(P_t)\right) \qquad (4.14)\end{aligned}$$

from equations 4.12 and 4.13

Finally, from lemma 4.3 and equation 4.14

$$\langle\mu_1(P_{t+1})\rangle \ge \frac{\mu_2(P_t)}{\mu_1(P_t)} - \frac{p_c}{4\mu_1(P_t)}\left(\mu_1(P_t)^2 - \mu_2(P_t)\right)$$

$$\begin{aligned}\langle\mu_1(P_{t+1})\rangle \le {} & \frac{\mu_2(P_t)}{\mu_1(P_t)} - \frac{p_c}{2\mu_1(P_t)}\left(\mu_1(P_t)^2 - \mu_2(P_t)\right) \\ & + \frac{p_c}{8\mu_1(P_t)}\left(2\mu_2(P_t) + 8\mu_{1.5}(P_t)\mu_{0.5}(P_t) + 6\mu_1(P_t)^2\right)\end{aligned}$$

Simplifying yields

$$\left(1+\frac{p_c}{4}\right)\frac{\mu_2(P_t)}{\mu_1(P_t)} - \frac{p_c}{4}\mu_1(P_t) \le \langle\mu_1(P_{t+1})\rangle \le \left(1+\frac{3p_c}{4}\right)\frac{\mu_2(P_t)}{\mu_1(P_t)} + \frac{p_c}{4}\mu_1(P_t) + p_c\frac{\mu_{0.5}(P_t)\mu_{1.5}(P_t)}{\mu_1(P_t)} \quad \square$$

To test this theorem, the GA was repeatedly applied to the function $f(\mathbf{x}) = |\mathbf{x}|^2$, with a variety of crossover schemes and rates. Throughout the experiment, the mutation operator was omitted from the GA, reflecting its omission from the model under examination. Given a sufficiently large population and number of trials, the means of the fitness measures $\mu_{0.5}, \mu_1, \mu_{1.5}$ and μ_2 at each generation were assumed to accurately approximate their expected values. Therefore, a GA with a population size of 128 and a chromosome length of 30 (15 bits per parameter) was applied 256 times to the function f for each of the crossover schemes and rates. The theorem was then applied to these data at each generation in order to predict the upper and lower bounds upon the expected mean fitnesses of the following generation. Figures 4.3 - 4.11 illustrate the results of this experiment - the dashed lines show the predicted bounds, and the solid lines the average mean fitness of the population.

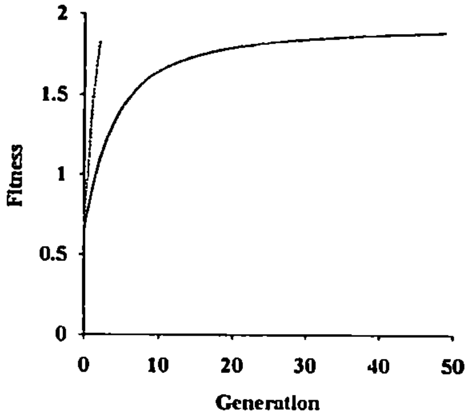


Figure 4.3 1 point crossover, $p_c = 0.4$

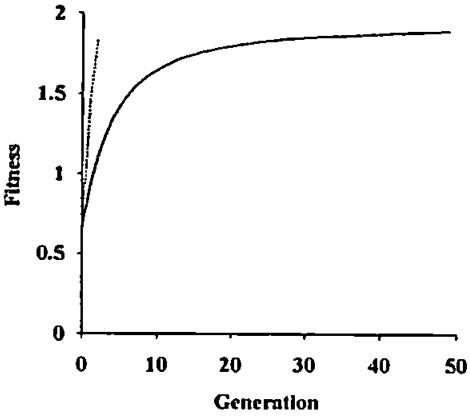


Figure 4.4 2 point crossover, $p_c = 0.4$

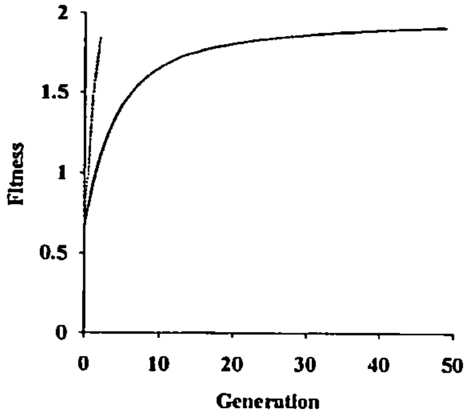


Figure 4.5 Uniform crossover, $p_c = 0.4$

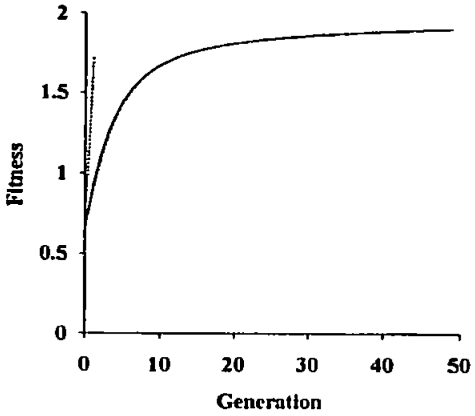


Figure 4.6 1 point crossover, $p_c = 0.6$

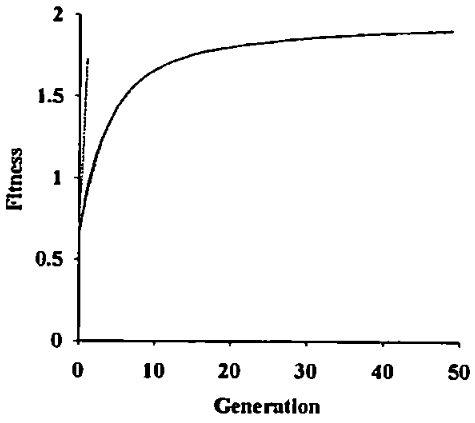
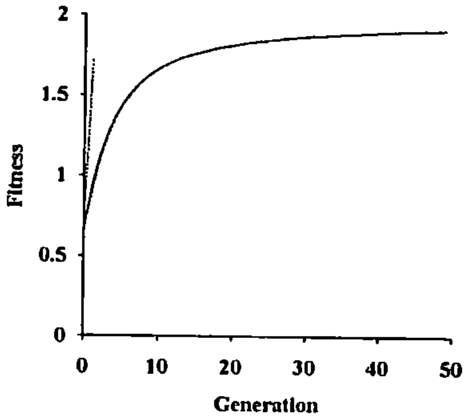


Figure 4.7 2 point crossover, $p_c = 0.6$



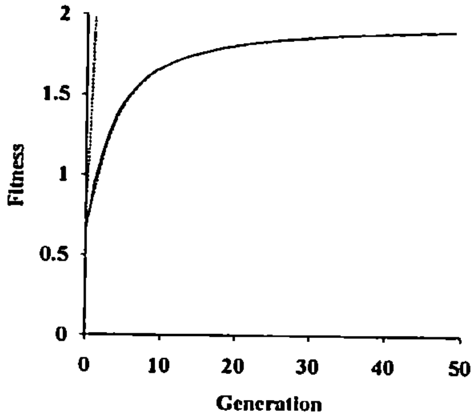
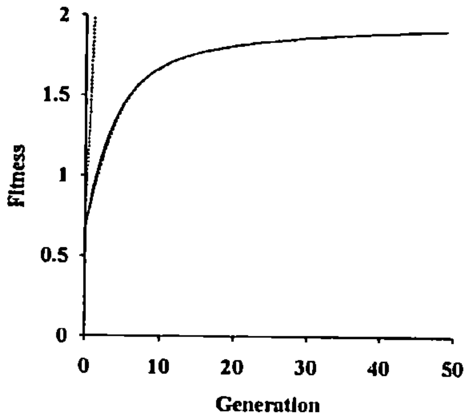


Figure 4.8 Uniform crossover, $p_c = 0.6$ **Figure 4.9** 1 point crossover, $p_c = 0.8$



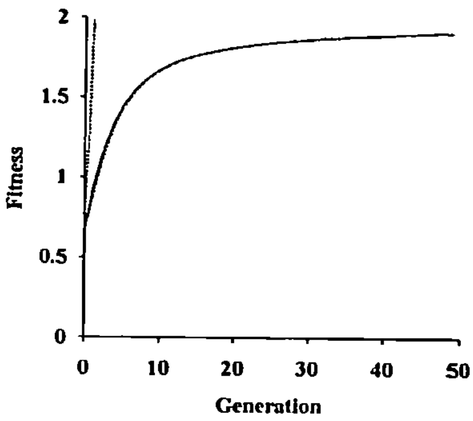


Figure 4.10 2 point crossover, $p_c = 0.8$ **Figure 4.11** Uniform crossover, $p_c = 0.8$

Although it is not clear from the graphs, the lower predicted bound occasionally exceeds the average of the mean fitnesses of the population during the latter generations of the experiments. The error is however very small, typically of the order 0.0004. It is likely that the error results from the assumptions made about the geometric nature of crossover for the purposes of the model (that crossover may be approximated by a pure rotation for example). Table 4.3 shows an extract of the raw data from the experiment with 2 point crossover and a crossover rate of 0.6. The error is calculated by subtracting the predicted lower bound from the observed average mean fitness of the population at each generation. When this value is negative, the lower bound is in error, having exceeded the observed correct value.

Generation	Predicted Lower Bound	Average Mean Fitness of Population	Error
20	**1.804314**	**1.803810**	**-0.000504**
21	**1.811279**	**1.810554**	**-0.000725**
22	1.817560	1.817681	0.000121
23	**1.824244**	**1.823644**	**-0.0006**
24	**1.829776**	**1.828595**	**-0.001181**
25	**1.834352**	**1.834245**	**-0.000107**
26	1.839537	1.840134	0.000597
27	1.844952	1.844977	0.000025
28	**1.849516**	**1.849229**	**-0.000287**
29	1.853590	1.853821	0.000231
30	1.857854	1.858160	0.000306
31	1.861979	1.862340	0.000361
32	**1.866008**	**1.865078**	**-0.00093**
33	1.868640	1.869059	0.000419
34	**1.872371**	**1.872002**	**-0.000369**
35	1.875194	1.875473	0.000279

Table 4.3 An example of the error of the lower bound (2 point crossover, $p_c = 0.6$)

During this portion of the experiment, half of the predicted values for the lower bound exceeded the observed value for the mean fitness of the population. Since the magnitude of the error is typically very small, it may be said that the error is qualitative rather than quantitative - the predicted lower bound is a fair approximation to a lower bound. In order to avoid this error, it is necessary that the geometric model of crossover be improved. An obvious starting point would be to use a rectangle bounded by the parent chromosomes rather than a rotation (as noted above).

In fact, the magnitude of the difference between the lower bound and the average mean fitness of the population is small at every generation (whether positive or negative). The predicted lower bound therefore provides a good prediction of the expected mean fitness of the population.

4.2.3 An Extended Model

A major failing of the model described by theorem 4.3 is that it is only capable of predicting the expected bounds of the mean fitness of the population one generation in advance. It is desirable that a model be constructed which is capable of predicting the bounds of the performance of the GA upon the function $f(\mathbf{x})=|\mathbf{x}|^2, \mathbf{x}\in[0,1)^2$ a number of generations in advance.

The next stage of this research therefore, was to construct such a model. The inaccuracy of the upper bound of theorem 4.3 together with the presence of the $\mu_{0.5}$ and $\mu_{1.5}$ terms prompted the omission of an upper bound from the extended model.

Before the construction of the extended model it is, as for the first model, necessary to take some preliminary steps.

LEMMA 4.4: *Given a symmetric polynomial of two parameters*

$$p(x,y) = p(y,x) = \sum_{i=0}^{N} a_i x^i y^{N-i} \tag{4.15}$$

where $a_i \equiv a_{N-i}$ *then the expected value of* $p(\max(x,y),\min(x,y))$ *for independent samples x, y of a population P is given by*

$$\langle p(\max(x,y),\min(x,y))\rangle = \sum_{i=0}^{N} a_i \mu_i(P)\mu_{N-i}(P) \tag{4.16}$$

PROOF: *Trivially*

$$\begin{aligned} &\max(x,y)^a \min(x,y)^b + \max(x,y)^b \min(x,y)^a \equiv x^a y^b + x^b y^a \\ &\max(x,y)^a \min(x,y)^a = x^a y^a \end{aligned} \tag{4.17}$$

It follows that

$$\begin{aligned} \langle p(\max(x,y),\min(x,y))\rangle &= \left\langle \sum_{i=0}^{N} a_i \max(x,y)^i \min(x,y)^{N-i} \right\rangle \\ &= \left\langle \sum_{i=0}^{N} a_i x^i y^{N-i} \right\rangle = \sum_{i=0}^{N} a_i \langle x^i y^{N-i} \rangle \\ &= \langle p(x,y) \rangle \end{aligned} \tag{4.18}$$

Finally, since x and y are independent samples of P

$$\begin{aligned}\left\langle x^a y^b \right\rangle &= \left\langle x^a \right\rangle\left\langle y^b \right\rangle \\ &\equiv \mu_a(P)\mu_b(P) \qquad \text{by definition}\end{aligned} \tag{4.19}$$

and therefore

$$\left\langle p\left(\max(x,y), \min(x,y)\right)\right\rangle = \sum_{i=0}^{N} a_i \mu_i(P)\mu_{N-i}(P) \qquad \square$$

LEMMA 4.5: *Given the parent parameter sets* $\mathbf{x}_1$ *and* $\mathbf{x}_2$, *the expected sum of the n-th powers of the fitnesses of the offspring parameter sets* $\mathbf{c}_1$ *and* $\mathbf{c}_2$ *is given by*

$$\begin{aligned}\left\langle f(\mathbf{c}_1)^n + f(\mathbf{c}_2)^n \right\rangle &= \left\langle f(\mathbf{c}_1)^n \right\rangle + \left\langle f(\mathbf{c}_2)^n \right\rangle \\ &= \frac{1}{2^{n-1}} \sum_{i=0}^{[n/2]} {}^nC_{2i} \prod_{j=1}^{i} \frac{2j-1}{2j} \left(f(\mathbf{x}_1) + f(\mathbf{x}_2)\right)^{n-2i} \left(|\mathbf{x}_1 + \mathbf{x}_2||\mathbf{x}_1 - \mathbf{x}_2|\right)^{2i}\end{aligned} \tag{4.20}$$

where [x] is the integer part of x.

PROOF: *From equation 4.5 (θ defined by figure 4.2)*

$$\begin{aligned} f(\mathbf{c}_1) &= \frac{f(\mathbf{x}_1) + f(\mathbf{x}_2)}{2} + \frac{|\mathbf{x}_1 + \mathbf{x}_2||\mathbf{x}_1 - \mathbf{x}_2|}{2}\cos\theta = A + B\cos\theta \\ f(\mathbf{c}_2) &= \frac{f(\mathbf{x}_1) + f(\mathbf{x}_2)}{2} - \frac{|\mathbf{x}_1 + \mathbf{x}_2||\mathbf{x}_1 - \mathbf{x}_2|}{2}\cos\theta = A - B\cos\theta \end{aligned}$$

Therefore

$$\begin{aligned}
\left\langle f(\mathrm{c}_1)^n\right\rangle+\left\langle f(\mathrm{c}_2)^n\right\rangle &= \left\langle (A+B\cos\theta)^n\right\rangle+\left\langle (A-B\cos\theta)^n\right\rangle \\
&= \left\langle \sum_{i=0}^{n} {}^nC_i A^{n-i} B^i \cos^i\theta \right\rangle + \left\langle \sum_{i=0}^{n} {}^nC_i A^{n-i} (-B)^i \cos^i\theta \right\rangle \\
&\qquad \text{from the binomial expansion} \\
&= \left\langle 2\sum_{i=0}^{[n/2]} {}^nC_{2i} A^{n-2i} B^{2i} \cos^{2i}\theta \right\rangle \quad \text{since terms with odd } i \text{ cancel} \\
&= 2\sum_{i=0}^{[n/2]} {}^nC_{2i} A^{n-2i} B^{2i} \left\langle \cos^{2i}\theta \right\rangle \quad \text{since } A,B \text{ are given}
\end{aligned} \tag{4.21}$$

As in the proof of theorem 4.2, without loss of generality choose $|\mathrm{x}_1| \ge |\mathrm{x}_2|, |\mathrm{c}_1| \ge |\mathrm{c}_2|, -\frac{\pi}{2} \le \theta \le \frac{\pi}{2}$

Thus

$$\begin{aligned}
\left\langle \cos^m\theta \right\rangle &= \frac{1}{\pi}\int_{-\pi/2}^{\pi/2} \cos^m\theta \, \mathrm{d}\theta \\
&= \frac{1}{\pi}\left(\left[\frac{\cos^{m-1}\theta \sin\theta}{m} \right]_{-\pi/2}^{\pi/2} + \frac{m-1}{m}\int_{-\pi/2}^{\pi/2} \cos^{m-2}\theta \, \mathrm{d}\theta \right) \\
&= \frac{1}{\pi}\left(\frac{m-1}{m}\int_{-\pi/2}^{\pi/2} \cos^{m-2}\theta \, \mathrm{d}\theta \right) \quad \text{since } \cos\left(\pm\frac{\pi}{2}\right)=0 \\
&= \begin{cases} \displaystyle\prod_{j=1}^{m/2} \frac{2j-1}{2j} & \text{for even } m \\ \displaystyle\frac{2}{\pi}\prod_{j=1}^{m/2} \frac{2j-1}{2j} & \text{for odd } m \end{cases}
\end{aligned} \tag{4.22}$$

And so

$$\left\langle \cos^{2i}\theta \right\rangle = \prod_{j=1}^{i} \frac{2j-1}{2j} \tag{4.23}$$

Finally, from equations 4.21 and 4.23

$$\left\langle f(\mathbf{c}_1)^n \right\rangle + \left\langle f(\mathbf{c}_2)^n \right\rangle = 2\sum_{i=0}^{\lfloor n/2 \rfloor} {}^nC_{2i} \prod_{j=1}^{i} \frac{2j-1}{2j} A^{n-2i} B^{2i} \tag{4.24}$$

Substitute A and B back into the equation and the result follows. □

These results provide the basis for the construction of the extended model. Before progressing however, it is necessary to calculate the expected moments of fitness of the first, randomly generated population.

LEMMA 4.6: *Given a random sample P of the function* $f(\mathbf{x}) = |\mathbf{x}|^2, \mathbf{x} \in [0,1)^2$, *then the expected k'th moment of fitness of the sample is given by*

$$\langle \mu_k(P) \rangle \approx \sum_{i=0}^{k} \frac{1}{(2k-2i+1)(2i+1)} {}^kC_i \tag{4.25}$$

PROOF: *Given that* $\mathbf{x} = (x_0, x_1)$

$$\begin{aligned}
\langle \mu_k(P) \rangle &= \left\langle \frac{1}{N} \sum_{\mathbf{x} \in P} f(\mathbf{x})^k \right\rangle = \left\langle \frac{1}{N} \sum_{\mathbf{x} \in P} \left(x_0^{\,2} + x_1^{\,2}\right)^k \right\rangle \\
&\approx \int_0^1 \int_0^1 \left(x_0^{\,2} + x_1^{\,2}\right)^k \mathrm{d}x_0 \mathrm{d}x_1 \\
&= \int_0^1 \int_0^1 \sum_{i=0}^{k} {}^kC_i x_0^{\,2k-2i} x_1^{\,2i} \mathrm{d}x_0 \mathrm{d}x_1 \\
&= \int_0^1 \left[\sum_{i=0}^{k} {}^kC_i x_0^{\,2k-2i} \frac{x_1^{\,2i+1}}{2i+1} \right]_0^1 \mathrm{d}x_0
\end{aligned}$$

$$\langle \mu_k(P)\rangle \approx \int_0^1 \sum_{i=0}^{k} {}^kC_i x_0^{2k-2i} \frac{1}{2i+1} \mathrm{d}x_0$$

$$= \left[\sum_{i=0}^{k} {}^kC_i \frac{x_0^{2k-2i}}{2k-2i+1}\frac{1}{2i+1}\right]_0^1$$

$$= \sum_{i=0}^{k} {}^kC_i \frac{1}{2k-2i+1}\frac{1}{2i+1} \qquad \square$$

To test this lemma, the first 51 moments of fitness were calculated for a number of randomly generated populations. The averages of each of these moments were then compared to the expected values predicted by the lemma. As above, a GA with a population size of 128 and a chromosome length of 30 (15 bits per parameter) was applied 256 times to the function f. The results of this experiment are given in table 4.4.

k	$\mu_k(P)$	$\langle\mu_k(P)\rangle$
0	1.000000	1.000000
1	0.662790	0.666667
2	0.617940	0.622222
3	0.681418	0.685714
4	0.839071	0.843175
5	1.120159	1.123617
10	9.496082	9.463041
20	2.48×10^{3}	2.50×10^{3}
30	1.12×10^{6}	1.16×10^{6}
40	6.36×10^{8}	6.71×10^{8}
50	4.15×10^{11}	4.42×10^{11}

Table 4.4 The average and expected moments of the first population

Clearly, the value of the expected moment of fitness predicted by lemma 4.4 grows progressively less accurate as the order of the moment increases. This may, in part, be due to rounding error, although it is more likely to be a result of the limited (finite) size of the population and length of the chromosome. Figure 4.12 shows the proportional error of the first 51 expected moments of fitness, $\frac{\langle \mu_k(P) \rangle}{\mu_k(P)}$.

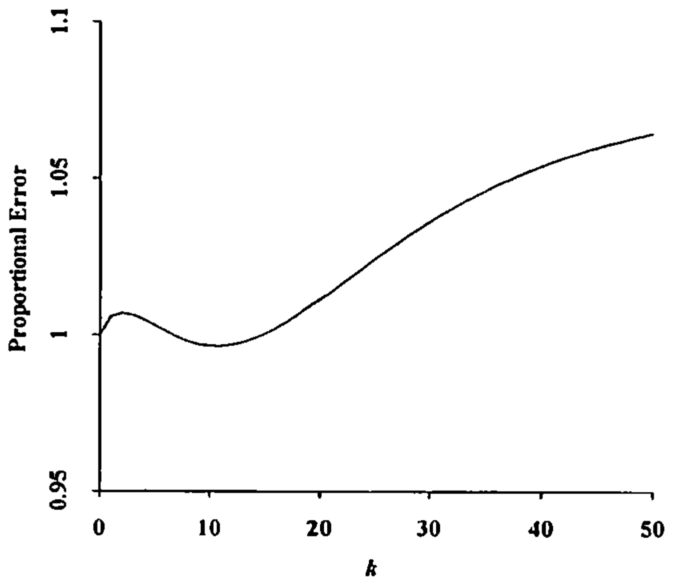


Figure 4.12 The proportional error of the expected moments of fitness, $\frac{\langle \mu_k(P) \rangle}{\mu_k(P)}$

Having now taken the necessary preliminary steps, it is possible to derive the extended performance model of the GA.

THEOREM 4.4: *The expected k'th moment of fitness of population* $P_{t+1}, \langle \mu_k(P_{t+1}) \rangle$ *is bounded below by*

$$\langle \mu_k(P_{t+1}) \rangle \geq (1-p_c)\frac{\mu_{k+1}(P_t)}{\mu_1(P_t)} + \frac{p_c}{2^{k+1}\mu_1(P_t)}\left(\sum_{i=0}^{\lfloor (k+1)/2 \rfloor} \sum_{r=0}^{k+1} \sum_{s=\max(r-2i,0)}^{\min(k+1-2i,r)} (-1)^{r-s} \prod_{j=1}^{i} \frac{2j-1}{2j} {}^{k+1}C_{2i} \, {}^{k+1-2i}C_s \, {}^{2i}C_{r-s} \mu_r(P_t) \mu_{k+1-r}(P_t) \right)$$

(4.26)

PROOF: *Following the proof of lemma 4.1 it is trivial to show that the expected k'th moment of fitness of the population* P_{t+1} *is given by*

$$\langle \mu_k(P_{t+1}) \rangle = \frac{\mu_{k+1}(P'_t)}{\mu_1(P'_t)} \qquad (4.27)$$

where P'_t *is the population of generation t immediately prior to selection.*

Once again, it is asserted that for a large population, the average effect of crossover upon the fitness of the population is well approximated by the effect of crossover upon the average fitness of the population.

Hence trivially

$$\langle \mu_{k+1}(P'_t) \rangle = \mu_{k+1}(P_t) + \frac{p_c}{2}\left\langle \left\langle f(\mathbf{c}_1)^{k+1} + f(\mathbf{c}_2)^{k+1} \right\rangle - \left(f(\mathbf{x}_1)^{k+1} + f(\mathbf{x}_2)^{k+1} \right) \right\rangle \qquad (4.28)$$

From lemma 4.5 equation 4.28 yields

$$\begin{aligned}\left\langle \mu_{k+1}(P'_t)\right\rangle = \mu_{k+1}(P_t) & \\ +\frac{p_c}{2^{k+1}}\left(\sum_{i=0}^{\lfloor (k+1)/2\rfloor} {}^{k+1}C_{2i}\prod_{j=1}^{i}\frac{2j-1}{2j}\big(f(\mathbf{x}_1)+f(\mathbf{x}_2)\big)^{k+1-2i}\big(|\mathbf{x}_1+\mathbf{x}_2||\mathbf{x}_1-\mathbf{x}_2|\big)^{2i}\right) & \\ -\frac{p_c}{2}\left\langle f(\mathbf{x}_1)^{k+1}+f(\mathbf{x}_2)^{k+1}\right\rangle &\end{aligned} \tag{4.29}$$

Furthermore, since $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *are independent samples of* P_t

$$\left\langle f(\mathbf{x}_1)^{k+1}+f(\mathbf{x}_2)^{k+1}\right\rangle = 2\mu_{k+1}(P_t) \tag{4.30}$$

and from the proof of lemma 4.3

$$|\mathbf{x}_1+\mathbf{x}_2|^2|\mathbf{x}_1-\mathbf{x}_2|^2 \geq \big(f(\mathbf{x}_1)-f(\mathbf{x}_2)\big)^2 \tag{4.31}$$

Combining equations 4.29, 4.30 and 4.31

$$\begin{aligned}\left\langle \mu_{k+1}(P'_t)\right\rangle \geq (1-p_c)\mu_{k+1}(P_t) & \\ +\frac{p_c}{2^{k+1}}\left(\sum_{i=0}^{\lfloor (k+1)/2\rfloor} {}^{k+1}C_{2i}\prod_{j=1}^{i}\frac{2j-1}{2j}\big(f(\mathbf{x}_1)+f(\mathbf{x}_2)\big)^{k+1-2i}\big(f(\mathbf{x}_1)-f(\mathbf{x}_2)\big)^{2i}\right) &\end{aligned} \tag{4.32}$$

Now consider the symmetric polynomial

$$\begin{aligned}p(a,b) &= (a+b)^{k+1-2i}(a-b)^{2i} \\ &= \left(\sum_{u=0}^{k+1-2i} {}^{k+1-2i}C_u a^{k+1-2i-u}b^u\right)\left(\sum_{v=0}^{2i} {}^{2i}C_v a^{2i-v}(-b)^v\right)\end{aligned}$$

Note that the a^r terms occur when

$$\begin{aligned} a^r &= a^{k+1-2i-u} a^{2i-v} \\ r &= k+1-2i-u+2i-v = k+1-u-v \end{aligned}$$

These terms are therefore given by

$$\begin{aligned} {}^{k+1-2i}C_u \, {}^{2i}C_v a^r b^u (-b)^v &= (-1)^v \, {}^{k+1-2i}C_u \, {}^{2i}C_v a^r b^{u+v} \\ &= (-1)^v \, {}^{k+1-2i}C_u \, {}^{2i}C_v a^r b^{k+1-r} \end{aligned}$$

where $u+v = k+1-r$
$u \in [0, k+1-2i]$
$v \in [0, 2i]$

However, p is symmetric and so the final $a^r b^{k+1-r}$ coefficient equals the $a^{k+1-r} b^r$ coefficient.

$$\sum_{\substack{u,v \\ u+v=k+1-r}} (-1)^v \, {}^{k+1-2i}C_u \, {}^{2i}C_v \equiv \sum_{\substack{u,v \\ u+v=r}} (-1)^v \, {}^{k+1-2i}C_u \, {}^{2i}C_v$$

where $u \in [0, k+1-2i]$
$v \in [0, 2i]$

The a^r terms therefore summate to

$$\sum_{u}(-1)^{r-u}\ {}^{k+1-2i}C_u\ {}^{2i}C_{r-u}a^r b^{k+1-r}$$

where $u \in [0, k+1-2i]$
$r-u \in [0, 2i]$

or, equivalently

$$\sum_{s=\max(r-2i,0)}^{\min(k+1-2i,r)}(-1)^{r-s}\ {}^{k+1-2i}C_s\ {}^{2i}C_{r-s}a^r b^{k+1-r}$$

Therefore

$$p(a,b)=\sum_{r=0}^{k+1}\sum_{s=\max(r-2i,0)}^{\min(k+1-2i,r)}(-1)^{r-s}\ {}^{k+1-2i}C_s\ {}^{2i}C_{r-s}a^r b^{k+1-r} \tag{4.33}$$

From lemma 4.4, given that a and b are the fitnesses of independent samples of P_t

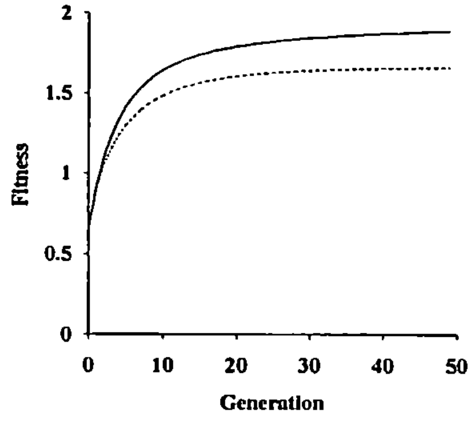
$$\langle p(\max(a,b),\min(a,b))\rangle=\sum_{r=0}^{k+1}\sum_{s=\max(r-2i,0)}^{\min(k+1-2i,r)}(-1)^{r-s}\ {}^{k+1-2i}C_s\ {}^{2i}C_{r-s}\mu_r(P_t)\mu_{k+1-r}(P_t) \tag{4.34}$$

Finally, substituting equation 4.34 into equation 4.32 with $a=f(\mathbf{x}_1), b=f(\mathbf{x}_2)$ *yields*

$$\langle\mu_{k+1}(P'_t)\rangle \ge (1-p_c)\mu_{k+1}(P_t)+\frac{p_c}{2^{k+1}}\left(\sum_{i=0}^{\lfloor(k+1)/2\rfloor}\ {}^{k+1}C_{2i}\prod_{j=1}^{i}\frac{2j-1}{2j}\sum_{r=0}^{k+1}\sum_{s=\max(r-2i,0)}^{\min(k+1-2i,r)}(-1)^{r-s}\ {}^{k+1-2i}C_s\ {}^{2i}C_{r-s}\mu_r(P_t)\mu_{k+1-r}(P_t)\right)$$

Rearranging and substituting this and equation 4.12 into equation 4.27 yields the desired result. □

In order to test this theorem, the first 51 expected moments of the fitness, calculated by lemma 4.6, were used to iteratively calculate the expected lower bound upon the mean fitness of the first 50 generations. The expected mean fitness of each generation was approximated by applying the GA repeatedly to the function, $f(\mathbf{x}) = |\mathbf{x}|^2, \mathbf{x} \in [0,1)^2$. As with the testing of theorem 4.3, the GA in question had a population size of 128, a chromosome length of 30 bits and was applied 256 times to the function f, for each of the crossover schemes and rates. The results of this experiment are illustrated graphically in figures 4.13 - 4.21. The dashed lines represent the predicted lower bound, the solid line the approximated expected mean fitness of each generation of the GA.



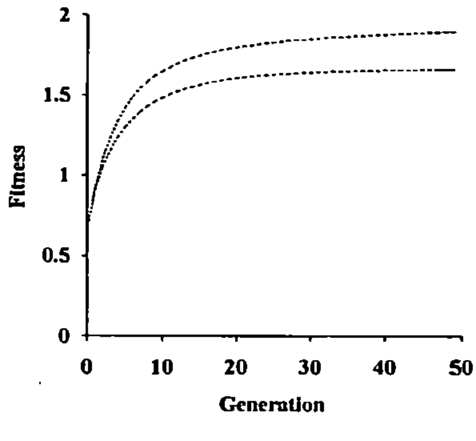
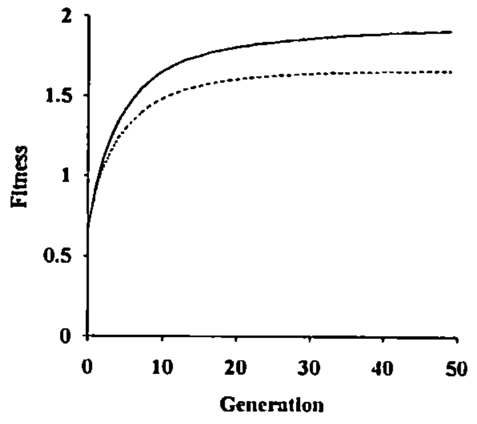


Figure 4.13 1 point crossover, $p_c = 0.4$ **Figure 4.14** 2 point crossover, $p_c = 0.4$



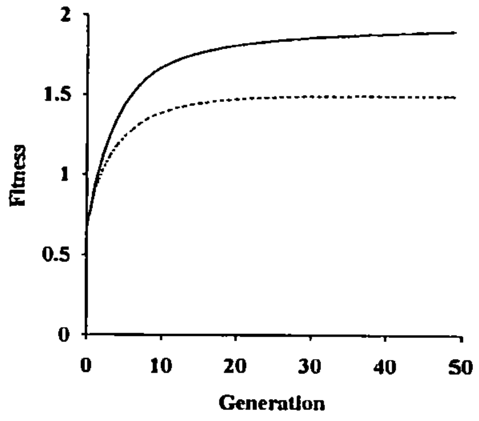


Figure 4.15 Uniform crossover, $p_c = 0.4$ **Figure 4.16** 1 point crossover, $p_c = 0.6$

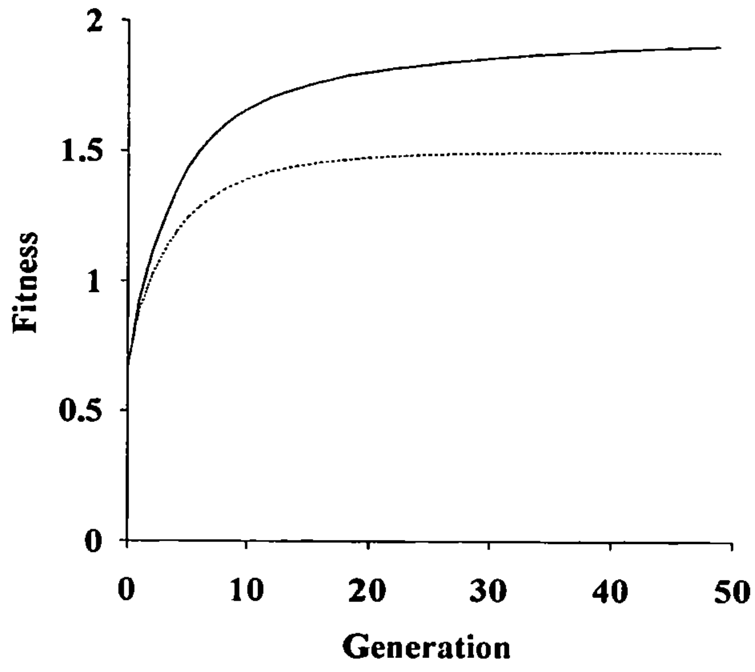
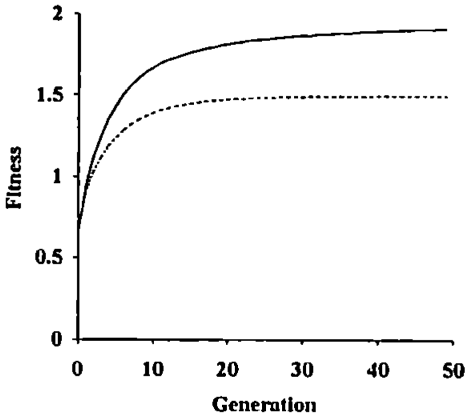


Figure 4.17 2 point crossover, $p_c = 0.6$



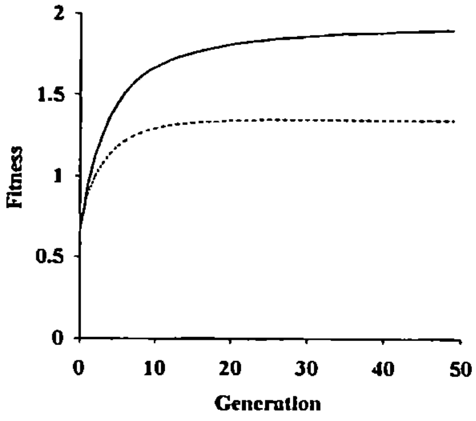
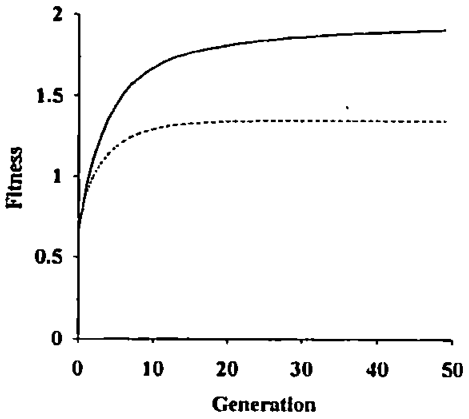


Figure 4.18 Uniform crossover, $p_c = 0.6$ **Figure 4.19** 1 point crossover, $p_c = 0.8$



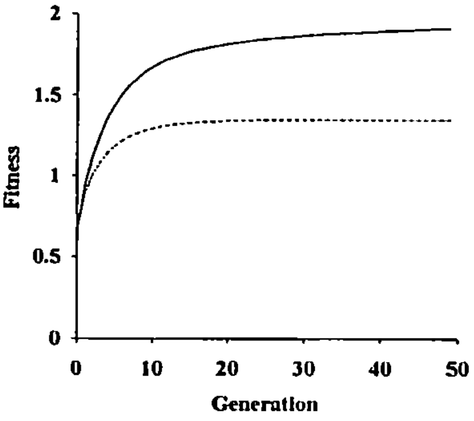


Figure 4.20 2 point crossover, $p_c = 0.8$ **Figure 4.21** Uniform crossover, $p_c = 0.8$

Clearly the lower bound predicted by the extended model is far less accurate than that predicted by the simple model. This is not surprising, since the extended model predicts the performance of the GA a great deal further in advance.

Much of the difference between the predictive model and the expected mean fitness of the population at each generation is accounted for by cumulative error. Figure 4.22 illustrates this by comparing the gradients of the two curves for 2 point crossover with a crossover rate of 0.6 (q.v. figure 4.16). As before, the dashed line represents the gradient of the predicted lower bound and the solid line that of the expected mean fitness. The remaining error is likely to be the result of the growing discrepancy between the predicted and observed moments of fitness and, as before, the inaccuracy of the geometric description of crossover.

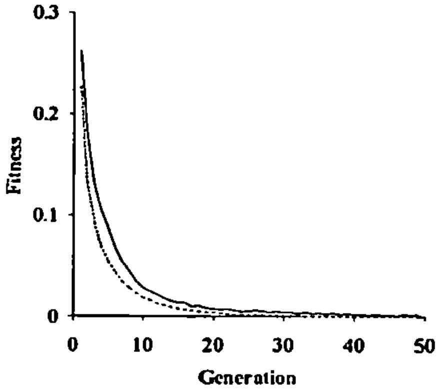


Figure 4.22 A comparison of gradient (2 point crossover, $p_c = 0.6$)

4.3 Conclusions

Although this analysis does not provide a rigorous description of the dynamics of the GA, it does achieve a good description of the performance of the GA for the specific function $f(\mathbf{x}) = |\mathbf{x}|^2, \mathbf{x} \in [0,1)$, both in the short term and in the long term. Interesting results in themselves, the predictive models also provide validation of the assumption that crossover can be described as a rotation in the parameter space. It is interesting to note that this description does not rely upon the notions of schema or building block, and unlike schema or building block based descriptions is valid for a wide range of chromosomal representations.

Furthermore, if crossover is held to act as a rotation in the parameter space, it is possible to draw parallels between the operation of the GA and that of simulated annealing. The technique of simulated annealing, as described in Chapter 2, operates by making a progressively smaller series of random perturbations to the current parameter set. At each step, the newly generated parameter set is accepted or rejected under a progressively more strict probabilistic criterion, the Metropolis criterion. In comparison, the GA begins with a diverse population of randomly generated parameter sets (defined by the chromosomes of the initial population). The effect of crossover upon these diverse parameter sets is to make large scale changes which are then selected or rejected by some fitness based selection technique (for example, Roulette Wheel Selection). As the population begins to converge upon the fitter regions of the search space, the parameter sets become more clustered, and so the effect of crossover upon pairs of parameter sets is less marked. Furthermore, the fitness of the population as a whole rises, and so poor solutions are far more likely to be rejected. The GA may therefore be described as initially performing a coarse search, allowing large scale changes to the population and allowing poor solutions to survive. As the generations advance, the search becomes progressively less coarse and the chance of survival for poor solutions reduces. The parallels between this behaviour and that of simulated annealing are clear, and go some way to describing the observed similarity in the behaviour of the two techniques.

Although the rotational description of crossover is almost certainly not an ideal account of the mechanics of the GA, it does show that it is possible to approach the problem of describing the GA from an alternative perspective.

4.4 Further Work

There are clearly a number of improvements to be made to the description of crossover, and to the predictive models. As has been noted, the distribution of offspring under crossover more closely resembles a rectangle than it does a circle, as illustrated in figure 4.23. It is likely that a model built upon a rectangular distribution of offspring would yield a more accurate description of the performance of the GA. Some work has been undertaken in this direction, although no such model was constructed. One of the major problems with this description is that it becomes very difficult to describe the effect of crossover in terms of vector algebra. When using a rotational description of crossover such a description is natural, and it is not necessary to resort to a Cartesian description of the problem. This significantly reduces the difficulty in constructing the performance model of the GA.

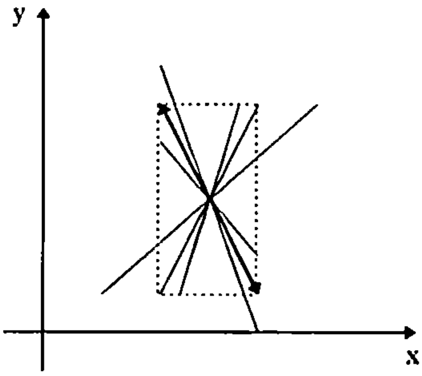


Figure 4.23 The geometric effect of crossover

It would be of great use to apply the models to a more general class of functions. For example, consider a performance model for the GA applied to the quadratic function,

$$f(\mathbf{x}) = a\mathbf{x}^{\mathrm{T}}\mathbf{x} + \mathbf{b}^{\mathrm{T}}\mathbf{x} + c$$

for arbitrary scalars a and c, and vector $\mathbf{b}$. Such a function may be used to approximate small regions of a surface. If the GA is applied to an arbitrary function, a model built for a quadratic function may still be applicable provided that the neighbourhood of each pair of chromosomes selected for crossover is well approximated by such a quadratic. It would then be possible to use the simple model as a predictive tool for a GA applied to any (locally quadratic) function optimisation problem during the latter stages of the optimisation process - once the population has converged sufficiently for the neighbourhood of each pair of chromosomes to be well approximated by such a quadratic.

Figure 4.24 shows the lower bound of the simple predictive model, represented by the dashed line, and the expected mean fitness at each generation of the GA applied to function f_5 of De Jong's test suite, represented by the solid line (as before, the expected mean fitness of the populations of the GA was calculated by applying the GA 256 times to the function, with a population size of 128 and a chromosome length of 30 bits). The predicted lower bound clearly exceeds the expected mean fitness over most of the 50 generations. Nevertheless, the error is not so great as to render the predictive model completely useless, and at least illustrates that a model built for a wider class of quadratic functions may be of use in predicting the performance of the GA upon an arbitrary function.

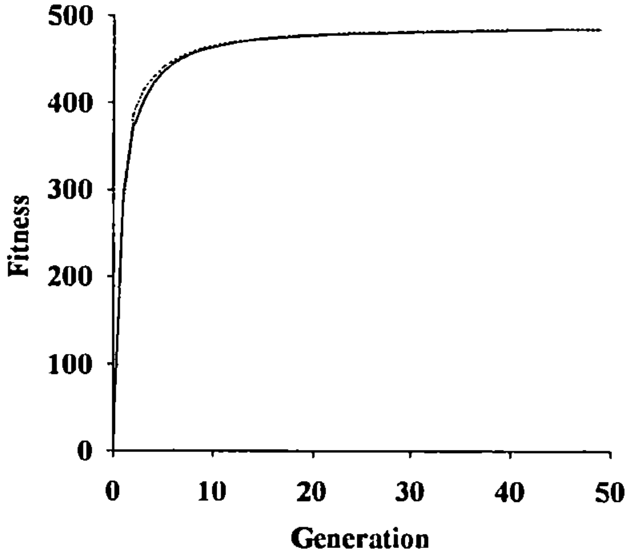


Figure 4.24 The simple model applied to f_5, (2 point crossover, $p_c = 0.6$)

If the model is to be used as a predictive tool for the user of the GA rather than for the vindication of a particular account of the operation of the GA, it will prove necessary to include the mutation operator. This has not been attempted during this study, although it would appear to be a fairly simple addition to the model. One possible difficulty could arise from the fact that mutation is generally applied to the binary chromosome rather than to the real parameter set. The effect of binary mutation upon the parameter set is difficult to describe simply in terms of the parameter set itself.

This study has not sought to provide a convergence theorem for the GA, despite the desirability of such a theorem. The extended model can be used to predict the convergence of a mutation free GA upon the function $f(\mathbf{x}) = |\mathbf{x}|^2$, simply through examination of the second moment of fitness at each generation. However, the model only predicts the lower bound upon the expected moments of fitness of each generation, and in its current form does not generalise well to a wide range of functions. It would be of interest, therefore, to investigate the possibility of using the geometric description of crossover to derive a general convergence theorem for the GA.

On the practical side, the geometric description of binary crossover suggests the use of rotation as a crossover operator for real coded GAs. A preliminary investigation of such an operator revealed a significant improvement in performance over simple one-point crossover for real coded GAs. However, before any conclusions may be drawn, a more detailed study is necessary, encompassing the many alternative approaches to crossover for the real coded GA (such as arithmetic crossover, for example).

It was suggested above that the effect of crossover during the early generations of the GA is to perform a coarse grained search of the domain under examination. If this is indeed the case, it would be of benefit to choose the initial population so as to promote maximal diversity. This suggests the use of quasi-random sequences to seed the initial population. Such sequences are often used for monte-carlo techniques since they guarantee a near uniform distribution, unlike purely random numbers.

5 Scientific Visualisation

The aim of scientific visualisation is to provide a clear and (at least qualitatively) accurate graphical representation of a system (Tufte, 1983, Earnshaw & Wiseman, 1992). This system may be dynamic, as for computational fluid dynamics, or static, as for function representation. In all cases the richness and flexibility of the visual field is exploited to provide greater insight into the system under consideration.

The principles of scientific visualisation have been applied in many fields of study, some of which are detailed below.

Medicine

The data gathered from sophisticated medical techniques such as MRI are generally analysed through volumetric rendering (Drebin et al, 1988). MRI, in particular, results in a vast quantity of data describing the tissue density of a patient throughout the body. This data can be transformed into a three dimensional image (projected onto the plane) of the patients skeleton or brain for example. The image may be rotated or otherwise transformed to reveal its structure - a clear improvement over the rigidly two dimensional x-ray. This technique allows the rapid diagnosis of certain conditions without recourse to invasive surgery.

Molecular Modelling

Molecules may be modelled and graphically represented by the computer. These models may be manipulated and combined to give a great deal of insight into the nature of the molecule. This process is a powerful tool for the chemist, simplifying the design process for new drugs.

Computational Fluid Dynamics

The field of computational fluid dynamics uses the computer to model the behaviour of fluids in motion. The complex behaviour of such fluids is difficult to understand without the use of visualisation, and many techniques have been developed for this field.

This thesis will concentrate upon the visualisation of the high dimensional surfaces described by functions of many parameters.

5.1 The Definition of Function

Given two sets of objects *X* and *Y*, a function may be defined as a mapping from *X* to *Y* under which to each member of *X* a unique member of *Y* is assigned. This is often written as $f : X \mapsto Y$ or $y = f(x)$ where $y \in Y, x \in X$ and f is the function. More rigorously, a function *f* may be defined as the set of order pairs where,

$$
\begin{aligned}
& f = \{(x,y) : x \in X, y \in Y\} \\
& \forall x \in X, \exists y \in Y \text{ s.t. } (x,y) \in f \\
& (x,y_0) \in f \cap (x,y_1) \in f \Rightarrow y_0 = y_1
\end{aligned}
$$

Under this definition, $y = f(x)$ is equivalent to $(x,y) \in f$.

The set X is known as the domain of f, and the set Y the codomain of f. Where $y = f(x)$, x is known as the argument or parameter set of f and y as the image of x under f or objective. Similarly, the set $f(X)$ is known as the image of X under f and is defined by,

$$f(X) = \{y \in Y : \exists x \in X \text{ s.t. } y = f(x)\}$$

For example the function $f(x) = x^2$ has the domain and codomain R. The image of R under f is $[0,\infty)$. The set of ordered pairs associated with this function is given by,

$$f = \{(x, x^2) : x \in \mathrm{R}\}$$

If the domain and codomain of a function are both one dimensional, the function may be represented by a graph where the order pairs are plotted as points upon a pair of Cartesian axes. This is the simplest and most familiar form of function visualisation.

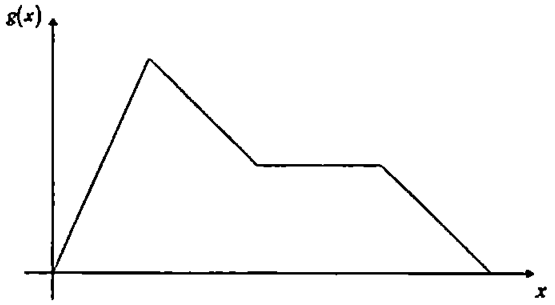


Figure 5.1 A Graph of the function $g = \{(0,0),(1,2),(2,1),(3,1),(4,0)\}$

It is important to distinguish between those functions with a multidimensional domain and one dimensional codomain and those with a multidimensional codomain and one dimensional domain. Both classes of functions may be described as being multidimensional, although the former are significantly more difficult to visualise than the latter.

For example, consider the function $f(t) = \{t, 1-t\}$. This may be represented exactly by two graphs, one of the function $f_0(t) = t$ and one of $f_1(t) = 1-t$. In general, an n dimensional vector valued function of one parameter, $f(t) = \mathbf{x}(t)$, may be represented exactly by the n functions $x_i(t)$. This is clearly not possible for a scalar valued function of many parameters.

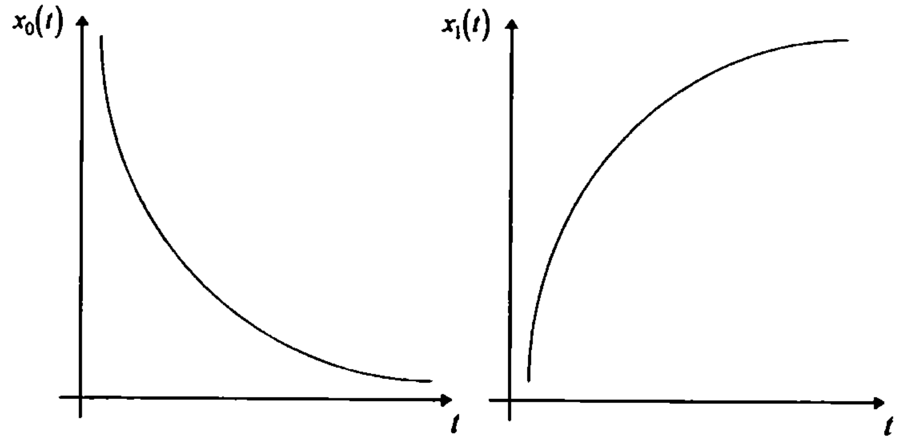


Figure 5.2 A vector valued function of time, $f(t) = \mathbf{x}(t)$

It is sufficient to consider only those techniques for the visualisation of scalar valued functions of many parameters. When presented with a vector valued function, any such technique may be used by constructing a table of graphical representations, as above.

5.2 The Limits of the Graph

The graphical representation of a function is generally constrained to lie in the plane (the printed page or the VDU) and cannot exceed three spatial dimensions and one temporal - the extent of the physical universe as perceived by the human mind. Clearly, one dimension is always required for the objective, although in some cases this can be provided through the careful use of colour or annotation.

As the number of parameters increases, it becomes more and more difficult to use the simple graph to visualise a function. A description of the problems inherent in this type of visualisation of such functions follows.

5.2.1 Functions of One Parameter

Functions of a single parameter fit nicely onto the plane, there is one dimension available for the parameter and one for the objective. The graph is an extremely effective tool for the rapid communication of vast amounts of information concerning the relationship between the parameter and the objective.

5.2.2 Functions of Two Parameters

There exist a number of techniques for constructing graphs of functions of two parameters. Firstly, a physical model of the surface may be constructed. Existing in three spatial dimensions, this technique guarantees a perfect qualitative representation of the function. In general, however, it is undesirable to construct a physical model of a surface since it is a relatively expensive and time consuming process.

Secondly, the surface may be projected onto the plane. This technique is attractive since the projection of solids onto the plain is a familiar process - the eye, for example, projects the three dimensional world onto the two dimensional retina. The human mind is capable of interpreting perspective cues to gain a great deal of information from this type of representation. However, the projection of the space onto the plane invariably leads to loss of information (through occlusion or ambiguity, for example).

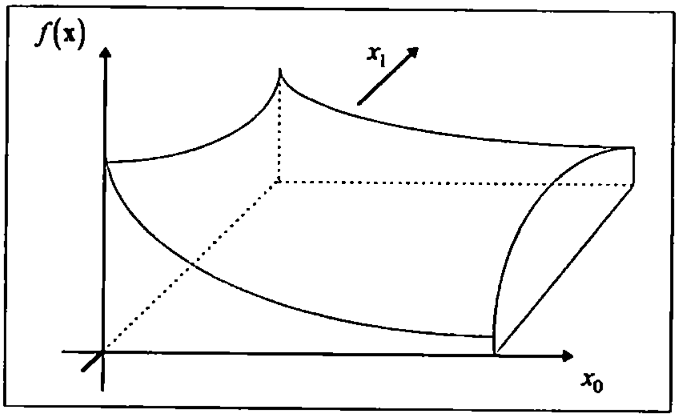


Figure 5.3 The Projection of a 2D Function onto the Plane

Finally, the function may be represented by curves of constant objective, also known as contours. The value of the objective for a given contour may be illustrated through colour or annotation. This avoids the loss of information which may result from occlusion or ambiguity, although the contours must be chosen carefully so that important features are not missed. This technique is also appealing, having obvious parallels with geographic maps.

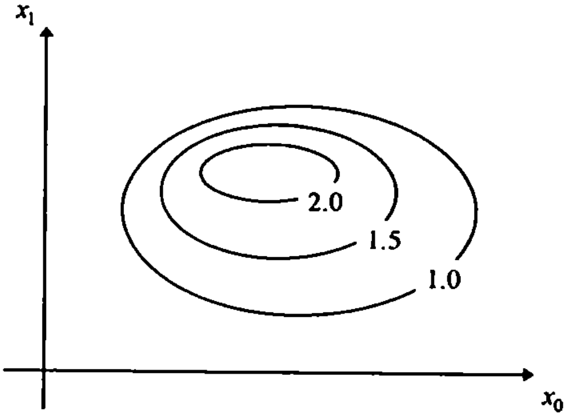


Figure 5.4 A Contour Diagram of a 2D Function

5.2.3 Functions of Three Parameters

The visualisation of functions of three parameters presents a major problem. For example, consider a function which describes the temperature of a cube of metal at every point. Simply plotting the points in space with colour providing the objective is not satisfactory since the internal structure of the cube is concealed.

Time may be used to provide an extra dimension for the visualisation of such functions. Techniques for the visualisation of two dimensional surfaces may be animated as the third parameter varies. Although this does solve the problem, it can lead to confusion when the final parameter is not timelike.

Isosurfaces are the three dimensional equivalent of contours, resulting in surfaces rather than lines. Once again, internal structure can be obscured, although this can be obviated to some extent through animation or transparency. However, as before, the simplicity and comprehensibility of the graph may be compromised.

5.2.4 Functions of Four or More Parameters

The visualisation of functions with more than three parameters cannot be achieved through the use of simple graphing techniques - there simply aren't enough dimensions available. The development of more sophisticated forms of graphical representation is therefore necessary.

5.3 Visualisation Techniques

There follows a brief description of a number of techniques for the visualisation of multidimensional surfaces and data.

5.3.1 The Graph

The use of the graph for the visualisation of one, two and to some extent three dimensional surfaces has already been described. Higher dimensional functions are often visualised by making graphs of one or two dimensional slices through the surface. This may be done by either fixing all but one or two of the parameters at a constant value, or by forcing the (generally linear) dependence of the parameter set upon one or two temporary variables.

Unfortunately, this technique may result in ambiguity and severe loss of information. For example, when applied to the quadratic function $f(x_0, x_1) = x_0 \cdot x_1$, this technique results in a pair of linear graphs. Such a result is clearly misleading and is therefore an extremely undesirable feature of this technique. Taking several slices through the surface may reduce the possibility of misrepresentation, although it can be difficult to combine the information from the many graphs.

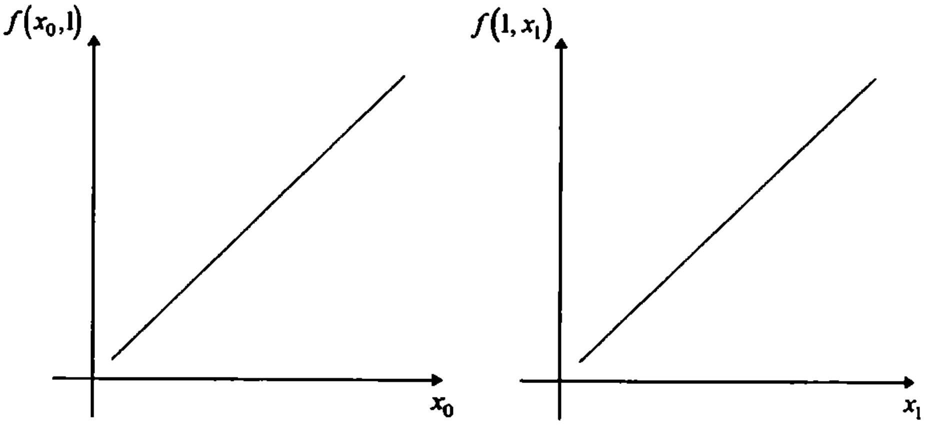


Figure 5.5 Slices through the 2D function $f(x_0, x_1) = x_0 \cdot x_1$

5.3.2 Projection onto the Plane

In a generalisation of the method used to project a two dimensional surface onto the plane, it is possible to project surfaces of any number of dimensions onto the plane. The familiar image of the hypercube is in fact a central projection of the four dimensional object onto the plane page. The problems arising from the projection of three dimensional objects onto the plane are exaggerated when projecting objects of more than three dimensions onto the plane. These problems are further aggravated by the inherent difficulty in interpreting high dimensional objects - the space they inhabit is beyond human sensory experience. For example, the six cubic faces of the hypercube are severely distorted by the four dimensional perspective and do not appear cubic at all. The results of this technique can be highly misleading and it does not therefore lend itself well to the visualisation of high dimensional surfaces.

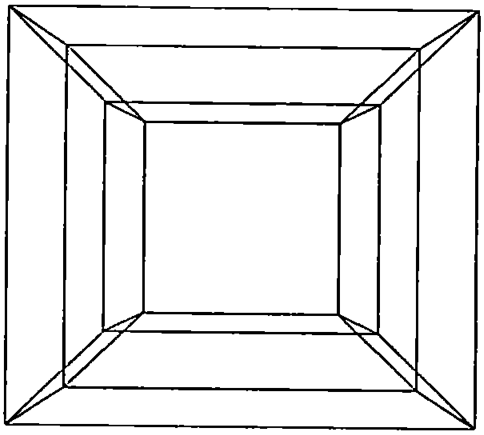


Figure 5.6 The central projection of the hypercube

5.3.3 The Graph of Graphs

The graph of graphs (Tufte, 1983) is an elegant technique for the visualisation of four (and to a lesser extent, higher) dimensional surfaces. A set of outer axes determine the constant values taken by the first pair of parameters for a number of slices through the surface. The graphs of these slices are used as if they were the points of a scatter plot. A desirable feature of this technique is that it requires only those skills which are required to read an ordinary graph. Unfortunately, it is difficult to trace relationships between parameters upon different sets of axes, a feature which can compromise the usefulness of this technique.

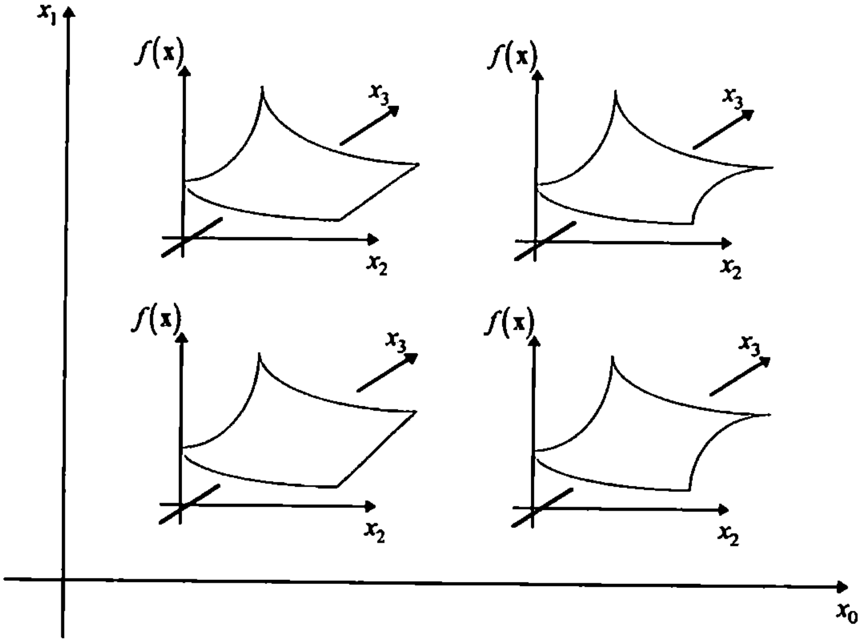


Figure 5.7 The graph of graphs

Higher dimensional surfaces may be represented by increasing the number of nested sets of axes. However, interpretation becomes difficult beyond two sets of axes and thus this technique is best applied to four dimensional surfaces. This is, nevertheless, a significant improvement over simpler visualisation techniques.

5.3.4 Chernoff Faces

Chernoff faces (Chernoff, 1973) are used to represent high dimensional scatter data. Each point is represented as a face, the shape and position of which is determined uniquely from the parameter set. The two most important parameters determine the position of the face on the graph. The remaining parameters determine features of the face, such as curvature of smile and radius of eyes. The reliance upon the facial recognition abilities of the viewer allows the quick and easy recognition of clusters and outliers in very high dimensional data sets. Although it is not a technique applicable to the visualisation of high dimensional surfaces, it is a simple and effective tool for the interpretation of very high dimensional data.

5.3.5 Multiple Linear Regression

Although not strictly a visualisation technique, multiple linear regression (Harris, 1975) may be used for the visualisation of multidimensional surfaces. Multiple linear regression involve the fitting of a linear model to multidimensional scatter data, and may be applied directly to the surface data generated by a function of many parameters. The linear approximation of the n dimensional surface resulting from a multiple linear regression is of the form,

$$f(\mathbf{x}) \approx \sum_{i=0}^{n} a_i x_i + b$$

The surface may therefore be visualised by the n linear graphs,

$$f_i(x_i) = a_i x_i + \frac{b}{n}$$

The objective value for a given parameter set is approximated by the sum of the objective values of each graph at the relevant parameter values.

$$f(\mathbf{x}) \approx \sum_{i=0}^{n} f_i(x_i)$$

If the function in question happens to be linear, this technique results in a quantitatively accurate representation of the surface, regardless of the number of parameters. This is a highly desirable trait for a visualisation technique, although many of the functions of interest are unlikely to be linear.

5.3.6 Generalised Additive Models

The generalised additive model (Hastie & Tibshirani, 1990) extends the notion of multiple linear regression to non-linear data. As with multiple linear regression the data may be provided by a function, thus allowing this technique to be used for function visualisation. The function is approximated by the sum of a series of one dimensional, possibly non-linear, functions

$$f(\mathbf{x}) = \alpha + \sum_{i=0}^{n} f_i(x_i) + \varepsilon$$

where ε are the errors, with $E(\varepsilon) = 0$ and $\text{var}(\varepsilon) = \sigma^2$.

These n functions are generally calculated iteratively, by removing the contribution the remaining functions make to each sample before applying a smooth to the data along the relevant axis.

$$g_i = f - \alpha - \sum_{j \neq i} f_j$$
$$f_i = \text{smooth of } g_i \text{ against } x_i$$

```
procedure generalised_additive_model
begin
      for i := 0 to n do
      begin
            initialise $f_i$;
      end

      alpha := average f(x)
      flag := 1;

      while (flag = 1) do
      begin
            flag := 0;

            for i := 0 to n do
            begin
                  g := f - alpha;
                  oldf := f;

                  for j := 0 to n do
                  begin
                        if (not (j = i)) then g := g - $f_j$;
                  end

                  $f_i$ := g smoothed against $x_i$;

                  if (not ($f_i$ = oldf)) then flag := 1;
            end
      end
end
```

Figure 5.8 Generalised Additive Model algorithm

The smooth is a statistical technique which involves smoothing a response (objective) against a predictor (parameter). This may be as simple as taking the mean value of the response at each value of the predictor, although more sophisticated techniques are generally used.

Figure 5.8 illustrates this process. In some cases (for example, polynomial approximation or linear splines), the univariate functions can be calculated directly using curve fitting techniques.

This approach has all of the advantages of multiple linear regression - it is capable of dealing with high dimensional data, and results in a series of easily interpreted one dimensional graphs. Furthermore, the errors resulting from the approximation are made explicit, allowing the user to determine how much faith should be placed in the results. Unlike multiple linear regression, this technique makes no assumption of linearity and is therefore less likely to mislead.

5.3.7 Projection Pursuit Regression

Projection pursuit regression (Friedman & Tukey, 1974) is similar to the generalised additive model. In the case of projection pursuit regression however, a model of the form

$$f(\mathbf{x}) \approx \sum_i f_i(\mathbf{a}_i \cdot \mathbf{x})$$

is fitted to the data. The vectors $\mathbf{a}_i$ project the data onto a new basis, hence the name projection pursuit regression. The advantage of choosing an optimal projection of the data is clear - the original axes are unlikely to be those for which the additive model is most accurate. The error of the regression is therefore likely to be smaller with this technique.

The model is fitted with a similar technique to that used for generalised additive models, although the introduction of a new set of basis vectors requires a cycle of optimisation rather than smoothing. This optimisation cycle will seek to minimise the error of the regression surface at each iteration. As for the other forms of regression documented here, this technique may be applied to the data generated by a function and the resulting graphs may then be used to visualise this function.

For example, consider the two dimensional function

$$f(\mathbf{x}) = x_0 \times x_1$$

Clearly, this function does not fit the generalised additive model, although it may be rewritten as

$$f(\mathbf{x}) = \frac{(x_0 + x_1)^2}{2} - \frac{(x_0 - x_1)^2}{2}$$

and may therefore fitted exactly by projection pursuit regression. The graphs of the two univariate functions

$$f_0(x) = \frac{1}{2}x^2$$
$$f_1(x) = -\frac{1}{2}x^2$$

may be used with the new bases

$$x'_0 = x_0 + x_1$$
$$x'_1 = x_0 - x_1$$

to exactly represent the original two dimensional function f.

6 A Technique for the Visualisation of High Dimensional Surfaces

This chapter will describe a technique for the visualisation of high dimensional functions. This technique, hereafter known as variable separation, generates a model similar to that of projection pursuit regression,

$$f(\mathbf{x}) = \sum_{i=0}^{N-1} f_i(\mathbf{b}_i \cdot \mathbf{x}) + \varepsilon(\mathbf{x}) \tag{6.1}$$

Unlike projection pursuit regression, the number of univariate functions in this model is limited to the dimension of the function under investigation, N. For the purpose of visualisation it is sensible to impose this restriction, since although an arbitrary degree of accuracy may be achieved by increasing the number of these functions, the legibility of the visualisation decreases significantly.

6.1 The Identification of a Good Projection

Unlike the scatter data usually analysed with statistical techniques such as the generalised additive model and projection pursuit regression, the data generated by a function have meaningful derivatives. This gradient information may be utilised to identify a good projection of the surface data generated by the function, given by the set of basis vectors $\mathbf{b}_i$.

6.1.1 The Use of Second Partial Derivatives

Consider once again the model to which the function is to be fitted, given by equation 6.1. Trivially, the error term ε represents the residual of the data which, after the change of basis, still depends upon more than one variable. If this were not the case then that portion of the error which was dependant upon only one variable would be absorbed by the relevant functions. The second partial derivatives of the model with respect to unique pairs of the bases b_i are therefore dependant only upon ε.

$$\frac{\partial^2 f(\mathbf{x})}{\partial b_i \partial b_j} \equiv \frac{\partial^2 \varepsilon(\mathbf{x})}{\partial b_i \partial b_j}, \quad i \neq j \tag{6.2}$$

Assuming that the function can be exactly represented by the model, the optimal projection for the surface data is trivially the one in which all of these second partial derivatives vanish. However, in most cases, the model will not exactly fit the original function and the best that may be achieved is the minimisation of the magnitude of these terms. For the purposes of the variable separation visualisation technique, the magnitude of a function is defined as the integral of its square over the space,

$$|f(\mathbf{x})| = \int_{x_0^{\min}}^{x_0^{\max}} \cdots \int_{x_{N-1}^{\min}}^{x_{N-1}^{\max}} f^2(\mathbf{x})\,\mathrm{d}x_0 \ldots \mathrm{d}x_{N-1} \tag{6.3}$$

In minimising the magnitude of each of the second partial derivatives with respect to the pairs of unique basis vectors, this approach implicitly assumes that this is equivalent to minimising the error term itself. This is not, in fact, the case and there are two extremes which illustrate this.

The first of these are those components of the function with large magnitude and very low frequency (or order). For example consider the univariate function f and its second derivative f'' over the range $[0,1]$.

$$f(x) = 10^{6}\cos\left(10^{-6}x\right)$$
$$f''(x) = -10^{-6}\cos\left(10^{-6}x\right)$$

Computing the magnitude of these gives

$$\left|f(x)\right| = \int_{0}^{1} 10^{6}\cos^{2}\left(10^{-6}x\right)\mathrm{d}x \approx 0.5\times 10^{6}$$
$$\left|f''(x)\right| = \int_{0}^{1} -10^{-6}\cos^{2}\left(10^{-6}x\right)\mathrm{d}x \approx -0.5\times 10^{-6}$$

Thus, although the magnitude of the second derivative is very small, the magnitude of the function itself is very large. Fortunately, such low frequency components of the function are well approximated by linear functions and will therefore be well approximated by the model regardless of the basis.

At the other extreme are those components of the function with small magnitude but very high frequency (or order). For example consider the univariate function g and its second derivative g'' over the range $[0,1]$,

$$g(x) = 10^{-6}\cos\left(10^{6}x\right)$$
$$g''(x) = -10^{6}\cos\left(10^{6}x\right)$$

Once again, computing the magnitude of these gives

$$|g(x)| = \int_0^1 10^{-6}\cos^2\left(10^6 x\right)\mathrm{d}x \approx 0.5\times 10^{-6}$$
$$|g''(x)| = \int_0^1 -10^6\cos^2\left(10^6 x\right)\mathrm{d}x \approx -0.5\times 10^6$$

In this case, the magnitude of the function is small, although the magnitude of the second derivative is very large. In reducing the magnitude of this second derivative, this approach will generate a projection which is aligned to remove what may very well be a small component of the function, possibly at the expense of removing larger components of the function. It is unlikely, therefore, that this approach will be effective for functions with high frequency or order components.

6.1.2 The Minimisation of the Second Partial Derivatives

Assuming that this approach is valid for the function under investigation, it is necessary to develop a technique by which minimises all of the second partial derivatives with respect to unique pairs of basis vectors. Returning to equation 6.2, the derivative information with respect to the optimal basis is not available in advance and so it is necessary to transform this identity so that the derivatives are in terms of the original basis. By the chain rule,

$$\frac{\partial^2 f(\mathbf{x})}{\partial b_p \partial b_q} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\frac{\partial b_i^0}{\partial b_p}\frac{\partial b_j^0}{\partial b_q}\frac{\partial^2 f(\mathbf{x})}{\partial b_i^0 \partial b_j^0} \tag{6.4}$$

where the b_i^0 are the original bases of the data. Consider the matrix **B** which transforms the data from the original basis b^0 to the optimal basis b. Then,

$$\frac{\partial b_i^0}{\partial b_j} \equiv \mathbf{B}_{ij}^{-1} \qquad (6.5)$$

Setting the matrix **D** to equal $\mathbf{B}^{-1}$, equation 6.4 becomes,

$$\frac{\partial^2 f(\mathbf{x})}{\partial b_i \partial b_j} \equiv \left(\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D}\right)_{ij} \qquad (6.6)$$

where **H** is the Hessian matrix of second partial derivatives of f, $\mathbf{H}_{ij} = \dfrac{\partial^2 f(\mathbf{x})}{\partial b_i^0 \partial b_j^0}$.

Identifying a basis under which the magnitudes of each of the second partial derivatives with respect to unique pairs of basis vectors are minimised is therefore equivalent to finding a scalar valued matrix **D** which, as much as is possible, diagonalises **H**,

$$\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D} \approx \Lambda(\mathbf{x}) \qquad (6.7)$$

where $\Lambda(\mathbf{x})$ is a diagonal function valued matrix, and both **D** and $\Lambda(\mathbf{x})$ are unknown. If **H** were real valued, equation 6.7 is solved trivially by a matrix **D** whose rows are the eigenvectors of **H** (since the eigenvectors of a symmetric real valued matrix are orthogonal). There are many techniques for the diagonalisation of symmetric real valued matrices which are generally used for the identification of eigenvectors. These techniques may be easily adapted for symmetric function valued matrices and may therefore be used for the identification of the optimal set of basis vectors. The technique described here is perhaps the simplest of these.

6.1.3 Matrix Diagonalisation

Given a symmetric real valued matrix H, firstly note that

$$\left(\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D}\right)_{pq} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\mathbf{D}_{ip}\mathbf{D}_{jq}\mathbf{H}_{ij} \tag{6.8}$$

Let the elements of **D** be given by

$$\mathbf{D}_{ij} = \begin{cases} 1 & i = j \\ \lambda & i = a, j = b \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

for some $a \neq b$. Equation 6.8 yields

$$\left(\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D}\right)_{pq} = \begin{cases} \lambda^2\mathbf{H}_{aa} + 2\lambda\mathbf{H}_{ab} + \mathbf{H}_{bb} & p = b, q = b \\ \lambda\mathbf{H}_{aq} + \mathbf{H}_{bq} & p = b, q \neq b \\ \lambda\mathbf{H}_{pa} + \mathbf{H}_{pb} & p \neq b, q = b \\ \mathbf{H}_{pq} & \text{otherwise} \end{cases} \tag{6.10}$$

Note that

$$\begin{aligned} \lambda = -\frac{\mathbf{H}_{ab}}{\mathbf{H}_{aa}} \quad &\Rightarrow \quad \left(\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D}\right)_{ab} = 0 \\ \lambda = 1 \quad &\Rightarrow \quad \left(\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D}\right)_{bb} = \mathbf{H}_{aa} + 2\mathbf{H}_{ab} + \mathbf{H}_{bb} \end{aligned} \tag{6.11}$$

These results form the basis for a technique for the diagonalisation of symmetric real valued matrices. The structure of the technique is illustrated in figure 6.1. The diagonalising matrix **D** is initialised to the identity. Iterating from top left to bottom right, the first of equations 6.11 is used to zero each of the off diagonal elements of **H**. Iterating through the matrix in this order guarantees that each step does not undo the work done by previous steps. If the element on the leading diagonal of the current column is zero, the second of the equations is first used to add to it a non-zero element of **H** from a column to the right (again to preserve the work already done). At each step the matrix **D'** represents the transition, and both **H** and **D** must be updated,

$$\begin{aligned} &\mathbf{H} \mapsto \mathbf{D}'^{\mathrm{T}}\,\mathbf{H}\mathbf{D}' \\ &\mathbf{D} \mapsto \mathbf{D}\mathbf{D}' \end{aligned} \tag{6.12}$$

Note that at each step **D** is postmultiplied by **D'** since if $\mathbf{D}_i$ represents the transition matrix of the *i*'th step, the transformation of **H** is given by

$$\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D} = \mathbf{D}_i^{\mathrm{T}}\mathbf{D}_{i-1}^{\mathrm{T}}\ldots\mathbf{D}_1^{\mathrm{T}}\mathbf{D}_0^{\mathrm{T}}\mathbf{H}\mathbf{D}_0\mathbf{D}_1\ldots\mathbf{D}_{i-1}\mathbf{D}_i \tag{6.13}$$

```
procedure diagonalise_matrix
begin
    D := I;

    for i := 0 to $N$-1 do
    begin
        if ($H_{ii}$ = 0) then do
        begin
            for j := i+1 to $N$-1 do
            begin
                if (not ($H_{ij}$ = 0)) then do
                begin
                    D' := I;
                    $D'_{ji}$ := 1;

                    D := D D';
                    H := $D'^{T}$ H D';

                    j := $N$;
                end
            end
        end

        for j := i+1 to $N$-1 do
        begin
            if (not ($H_{ij}$ = 0)) then do
            begin
                D' := I;
                $D'_{ij}$ := -$H_{ij}$ / $H_{ii}$;

                D := D D';
                H := $D'^{T}$ H D';
            end
        end
    end
end
```

Figure 6.1 The structure of the matrix diagonalisation technique

As an example consider the symmetric real valued matrix **H** given by

$$\mathbf{H}=\begin{pmatrix}1 & 2 & 3\\ 2 & 4 & 5\\ 3 & 5 & 6\end{pmatrix}$$

Following the matrix diagonalisation algorithm yields the following steps,

$$\mathbf{H}\mapsto\begin{pmatrix}1 & 0 & 0\\ -2 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 2 & 3\\ 2 & 4 & 5\\ 3 & 5 & 6\end{pmatrix}\begin{pmatrix}1 & -2 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & 0 & 3\\ 0 & 0 & -1\\ 3 & -1 & 6\end{pmatrix}$$

$$\mathbf{D}\mapsto\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & -2 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & -2 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$\mathbf{H}\mapsto\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -3 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 3\\ 0 & 0 & -1\\ 3 & -1 & 6\end{pmatrix}\begin{pmatrix}1 & 0 & -3\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & 0 & 0\\ 0 & 0 & -1\\ 0 & -1 & 3\end{pmatrix}$$

$$\mathbf{D}\mapsto\begin{pmatrix}1 & -2 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & -3\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}1 & -2 & -3\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$\mathbf{H}\mapsto\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 1\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 0 & 0 & -1\\ 0 & -1 & 3\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 1 & 1\end{pmatrix}=\begin{pmatrix}1 & 0 & 0\\ 0 & -5 & -4\\ 0 & -4 & -3\end{pmatrix}$$

$$\mathbf{D}\mapsto\begin{pmatrix}1 & -2 & -3\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 1 & 1\end{pmatrix}=\begin{pmatrix}1 & -5 & -3\\ 0 & 1 & 0\\ 0 & 1 & 1\end{pmatrix}$$

$$\mathbf{H} \mapsto \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.8 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -5 & -4 \\ 0 & -4 & -3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -0.8 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & 0.2 \end{pmatrix}$$

$$\mathbf{D} \mapsto \begin{pmatrix} 1 & -5 & -3 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -0.8 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -5 & 1 \\ 0 & 1 & -0.8 \\ 0 & 1 & 0.2 \end{pmatrix}$$

Finally, this may be checked by calculating $\mathbf{D}^T\mathbf{HD}$.

$$\begin{aligned} \mathbf{D}^T\mathbf{HD} &= \begin{pmatrix} 1 & 0 & 0 \\ -5 & 1 & 1 \\ 1 & -0.8 & 0.2 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & -5 & 1 \\ 0 & 1 & -0.8 \\ 0 & 1 & 0.2 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ -5 & 1 & 1 \\ 1 & -0.8 & 0.2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 2 & -1 & -0.2 \\ 3 & -4 & 0.2 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & 0.2 \end{pmatrix} \end{aligned}$$

6.1.4 Diagonalising the Hessian

To adapt this technique for use with symmetric function valued matrices such as the Hessian matrix of second partial derivatives it is necessary only to change the choice of λ in equation 6.9. Specifically, it is necessary to choose λ such that the magnitude of $\lambda \mathbf{H}_{aa} + \mathbf{H}_{ab}$ is minimised.

$$
\begin{aligned}
|\lambda \mathbf{H}_{aa} + \mathbf{H}_{ab}| &= \int_V |\lambda \mathbf{H}_{aa} + \mathbf{H}_{ab}|^2 \mathrm{d}V \\
&= \int_V \left(\lambda^2 \mathbf{H}_{aa}^2 + 2\lambda \mathbf{H}_{aa}\mathbf{H}_{ab} + \mathbf{H}_{ab}^2\right)\mathrm{d}V
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial |\lambda \mathbf{H}_{aa} + \mathbf{H}_{ab}|}{\partial \lambda} &= 2\lambda \int_V \mathbf{H}_{aa}^2 \mathrm{d}V + 2\int_V \mathbf{H}_{aa}\mathbf{H}_{ab}\mathrm{d}V \\
&\equiv 0 \quad \text{iff} \quad \lambda = -\frac{\int_V \mathbf{H}_{aa}\mathbf{H}_{ab}\mathrm{d}V}{\int_V \mathbf{H}_{aa}^2 \mathrm{d}V}
\end{aligned}
\tag{6.14}
$$

For example, consider the two dimensional function

$$f(\mathbf{x}) = x_0^3 + 3x_0^2 x_1 + 3x_0 x_1^2 + 2x_1^3$$

The Hessian of this function is therefore

$$\mathbf{H} = \begin{pmatrix} 6x_0 + 6x_1 & 6x_0 + 6x_1 \\ 6x_0 + 6x_1 & 6x_0 + 12x_1 \end{pmatrix}$$

Clearly $\mathbf{H}_{01} \equiv \mathbf{H}_{00}$ and therefore $\frac{\int_V \mathbf{H}_{aa}\mathbf{H}_{ab}\mathrm{d}V}{\int_V \mathbf{H}_{aa}^2 \mathrm{d}V} = 1$. $\mathbf{H}_{01}$ is therefore minimised by setting

$\mathbf{D}_{01} = -1$.

$$
\begin{aligned}
\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D} &= \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 6x_0 + 6x_1 & 6x_0 + 6x_1 \\ 6x_0 + 6x_1 & 6x_0 + 12x_1 \end{pmatrix}\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 6x_0 + 6x_1 & 0 \\ 6x_0 + 6x_1 & 6x_1 \end{pmatrix} \\
&= \begin{pmatrix} 6x_0 + 6x_1 & 0 \\ 0 & 6x_1 \end{pmatrix}
\end{aligned}
$$

In practise, the algebraic forms of the second derivatives are not available, and the function valued Hessian matrix must therefore be approximated by a vector valued matrix of the values of the second derivatives at a finite number of sample points. Once again, the value of λ in equation 6.9 must be modified so as to minimise $\lambda \mathbf{H}_{aa} + \mathbf{H}_{ab}$ with vector valued $\mathbf{H}_{ij}$. In the same fashion as for equation 6.14 the value for λ may be calculated to yield

$$\lambda = -\frac{\mathbf{H}_{aa} \cdot \mathbf{H}_{ab}}{\left|\mathbf{H}_{aa}\right|^2} \tag{6.15}$$

Consider once again the two dimensional function

$$f(\mathbf{x}) = x_0^3 + 3x_0^2 x_1 + 3x_0 x_1^2 + 2x_1^3$$

$$\mathbf{H} = \begin{pmatrix} 6x_0 + 6x_1 & 6x_0 + 6x_1 \\ 6x_0 + 6x_1 & 6x_0 + 12x_1 \end{pmatrix}$$

Given the set of samples

(0, 0)	(0, 0.5)	(0, 1)
(0.5, 0)	(0.5, 0.5)	(0.5, 1)
(1, 0)	(1, 0.5)	(1, 1)

the Hessian may be represented by

$$\mathbf{H} = \begin{pmatrix} (0,3,6,3,6,9,6,9,12) & (0,3,6,3,6,9,6,9,12) \\ (0,3,6,3,6,9,6,9,12) & (0,6,12,3,9,15,6,12,18) \end{pmatrix}$$

As before $\mathbf{H}_{00} \equiv \mathbf{H}_{01}$ and therefore $\mathbf{D}_{01} = -1$.

$$\begin{aligned}
\mathbf{D}^{\mathrm{T}}\mathbf{H}\mathbf{D} &= \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} (0,3,6,3,6,9,6,9,12) & (0,3,6,3,6,9,6,9,12) \\ (0,3,6,3,6,9,6,9,12) & (0,6,12,3,9,15,6,12,18) \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} (0,3,6,3,6,9,6,9,12) & (0,0,0,0,0,0,0,0,0) \\ (0,3,6,3,6,9,6,9,12) & (0,3,6,0,3,6,0,3,6) \end{pmatrix} \\
&= \begin{pmatrix} (0,3,6,3,6,9,6,9,12) & (0,0,0,0,0,0,0,0,0) \\ (0,0,0,0,0,0,0,0,0) & (0,3,6,0,3,6,0,3,6) \end{pmatrix}
\end{aligned}$$

Given the second derivatives at a finite number of points, it is therefore possible to find an approximation to the matrix **D** which diagonalises the Hessian. Provided that there are no high frequency (or order) components of the function, the inverse of this matrix should provide a good basis, although not in general optimal, upon which the model is to be fitted to the surface. This derivative information may be obtained in a number of ways, two of which are described below.

6.1.5 Obtaining the Second Partial Derivatives

The simplest method by which derivative information may be generated is to approximate them with finite differences. By definition, the partial derivative of a function with respect to a basis b at a point $\mathbf{x}$ is given by

$$\left.\frac{\partial f}{\partial b}\right|_{\mathbf{x}} = \lim_{\delta \to 0} \frac{f(\mathbf{x} + \delta \mathbf{b}) - f(\mathbf{x})}{\delta} \tag{6.16}$$

where **b** is the basis vector of the basis *b*.

The partial derivative may be approximated by choosing a small but non-zero value for δ. For example given the two dimensional function

$$f(\mathbf{x}) = x_0^2 + x_1^2$$

then by choosing $\delta = 0.01$

$$\begin{aligned}\left.\frac{\partial f}{\partial x_0}\right|_{(1,1)} &\approx \frac{f((1,1) + 0.01(1,0)) - f((1,1))}{0.01} \\ &= \frac{\left(1.01^2 + 1^2\right) - \left(1^2 + 1^2\right)}{0.01} = \frac{0.0201}{0.01} \\ &= 2.01\end{aligned}$$

which closely approximates the correct value of 2.0. The second partial derivatives may be calculated in a similar fashion. One of the advantages of using second finite differences to approximate the second partial derivatives is that by Shannon's Sampling Theorem, components of the functions with a frequency greater than $0.5 \times \delta^{-1}$ are removed. This goes some way to removing the problems which such components are likely to create for this technique.

A second, more computationally expensive, technique is to approximate the neighbourhood of each point by a quadratic function. The second partial derivatives may then be calculated with the coefficients of the quadratic. For example, let

$$p_2(\mathbf{x}) = \sum_i \sum_j a_{ij} x_i x_j + \sum_i b_i x_i + c$$

be the quadratic equation which approximates the surface in the neighbourhood of the point $\mathbf{x}$. The second partial derivatives of the function f may then be approximated by

$$\left.\frac{\partial f}{\partial x_i \partial x_j}\right|_{\mathbf{x}} \approx a_{ij} + a_{ji} \qquad (6.17)$$

Rather than fitting the quadratic to the points within some arbitrary neighbourhood of $\mathbf{x}$, it is not difficult to fit the quadratic to the entire sample of the surface but with the emphasis upon points close to $\mathbf{x}$. This has the added advantage of effectively smoothing high frequency components out of the data, removing one of the cases for which the technique is likely to fail.

6.2 Fitting the Model to the Surface

Having found a good projection of the surface data, the next step is to fit the model to the data using this projection. This may be achieved using a simple curve fitting technique, adapted so that it fits a curve in the form of the model.

6.2.1 A Simple Curve Fitting Technique

Given a function f of which a finite number of samples are known, consider the approximating function f' given by

$$f'(\mathbf{x}) = \sum_i a_i f'_i(\mathbf{x}) \tag{6.18}$$

where the f_i are known and the a_i unknown. To fit this function to the data generated by f it is necessary to minimise the difference between them throughout the sample.

$$\text{minimise } \sum_i \left(f'(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2 = \sum_i \sum_j \left(a_j f'_j(\mathbf{x}_i) - f(\mathbf{x}_i)\right)^2 \tag{6.19}$$

Setting the partial derivatives of this expression with respect to the a_j to zero yields the simultaneous linear equations in a_i

$$\sum_i \sum_j a_j f'_j(\mathbf{x}_i) f'_k(\mathbf{x}_i) = \sum_i f(\mathbf{x}_i) f'_k(\mathbf{x}_i) \tag{6.20}$$

the solution of which are the coefficients which yield the best approximation of the function (since equation 6.19 is quadratic with respect to the a_i , there is only one stationary point - the minimum).

Note that with a little modification, this technique may be used for the locally weighted quadratic fit mentioned in section 6.1.5. To fit a quadratic to the data, weighted to the neighbourhood of a point **x**, equation 6.20 becomes

$$\sum_i \sum_j w_{\mathbf{x}}(\mathbf{x}_i) a_j f'_j(\mathbf{x}_i) f'_k(\mathbf{x}_i) = \sum_i w_{\mathbf{x}}(\mathbf{x}_i) f(\mathbf{x}_i) f'_k(\mathbf{x}_i) \tag{6.21}$$

where $w_{\mathbf{x}}(\mathbf{x}_i)$ is the weight for the point $\mathbf{x}_i$ with respect to the point of interest $\mathbf{x}$. This has the effect of making the curve fitting process more sensitive to the errors at those points near to the point of interest. One possible form of this weight function could be

$$w_{\mathbf{x}}(\mathbf{x}_i) = \frac{1}{\omega|\mathbf{x} - \mathbf{x}_i| + 1} \tag{6.22}$$

where ω is a constant determining the severity of the weighting.

6.2.2 Generating the Model with the Simple Curve Fitting Technique

Consider once again the model which is to be fitted to the surface data, given by equation 6.1,

$$f(\mathbf{x}) = \sum_{i=0}^{N-1} f_i(\mathbf{b}_i \cdot \mathbf{x}) + \varepsilon(\mathbf{x})$$

Before the curve fitting technique may be used to fit this model, it is necessary to decide upon the set of basis functions g_j which will be used to describe the univariate functions f_i. Each of the univariate functions may then be described by,

$$f_i = \sum_j a_{ij} g_j \tag{6.23}$$

and given M such basis functions, equation 6.1, ignoring the error term ε, becomes

$$f(\mathbf{x}) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} a_{ij} g_j(\mathbf{b}_i \cdot \mathbf{x}) \tag{6.24}$$

This is similar in structure to equation 6.18, the equation governing the curve fitting algorithm. Substituting equation 6.24 for equation 6.18 yields the simultaneous linear equations,

$$\sum_i \sum_{j=0}^{N-1} \sum_{k=0}^{M-1} a_{jk} g_k(\mathbf{b}_j \cdot \mathbf{x}_i) g_q(\mathbf{b}_p \cdot \mathbf{x}_i) = \sum_i f(\mathbf{x}_i) g_q(\mathbf{b}_p \cdot \mathbf{x}_i) \tag{6.25}$$

The solution of which yields the best fit of the model to the surface given the basis vectors $\mathbf{b}_i$.

For the variable separation technique it was decided that the best set of basis functions were linear splines. These have a number of advantages over polynomial or sinusoidal basis functions, primarily that they are applicable over a much wider range of domains (including discontinuous functions) and that they do not suffer from the effect of ringing - high frequency components of the curve fit causing wild fluctuations at the boundaries of the domain. Furthermore, in most cases, when the model is finally presented graphically to the user, most graphing utilities approximate the curve with linear splines and so they seem to be a natural choice.

The functional form of a linear spline is simple and is shown graphically in figure 6.2.

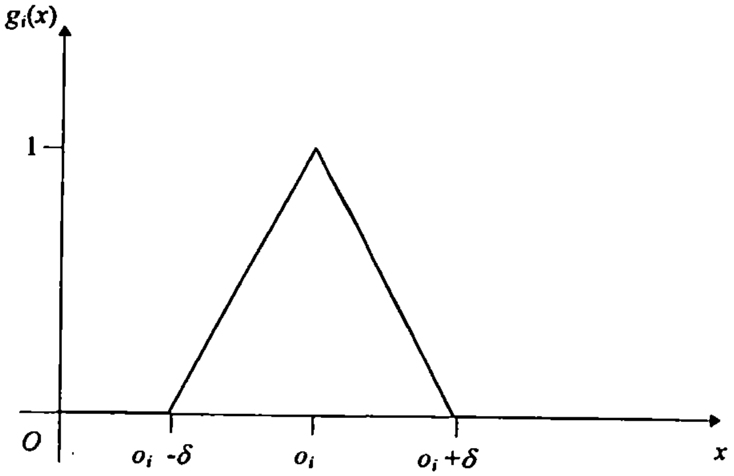


Figure 6.2 The graphical form of a univariate linear spline

The algebraic form of the linear spline is given by,

$$g_i(x) = \begin{cases} 1.0 - \dfrac{o_i - x}{\delta} & o_i - \delta \le x \le o_i \\ 1.0 - \dfrac{x - o_i}{\delta} & o_i \le x \le o_i + \delta \\ 0.0 & \text{otherwise} \end{cases} \qquad (6.25)$$

where o_i and δ depend upon the range of the parameter x and the number of splines being used to approximate the function. Assuming that the parameter has been scaled onto the unit interval $[0,1]$, these parameters are given by,

$$o_i = \frac{i}{M-1}$$
$$\delta = \frac{1}{M-1} \tag{6.26}$$

where M is the number of linear splines used to approximate each univariate function.

In effect, the set of linear splines produce a linear interpolation between the points o_i. To illustrate this, consider a pair of linear splines g_0 and g_1. With only two splines the parameters o_i and δ are given by

$$o_0 = 0, \quad o_1 = 1, \quad \delta = 1$$

and so, noting that the range of x is assumed to be scaled to the unit interval [0,1], the splines g_i are given by

$$g_0(x) = 1.0 - x$$
$$g_1(x) = x$$

Consider a function approximation of the form of equation 6.23 using these splines,

$$f = \sum_{i=0}^{1} a_i g_i$$

and therefore

$$f(x) = a_0(1.0 - x) + a_1(x)$$

As expected, this function describes a line from the point (0, a_0) to (1, a_1).

6.3 The Variable Separation Algorithm

Having described the individual tasks necessary for the variable separation visualisation technique it is now necessary to combine these into an algorithm for the visualisation of high dimensional surface data.

6.3.1 Diagonalising the Hessian in Practise

As has been noted, in practise it may not be possible to completely diagonalise the Hessian. One of the simplest functions for which this is true is given by,

$$f(\mathbf{x}) = x_0^2 x_1$$

The Hessian of this function is

$$\mathbf{H} = \begin{pmatrix} 2x_1 & 2x_0x_1 \\ 2x_0x_1 & 0 \end{pmatrix}$$

and clearly $\mathbf{H}_{00}$ and $\mathbf{H}_{01}$ are independent functions. In such cases it is unlikely that the simple matrix diagonalisation technique will identify the optimal basis in a single pass, having been originally developed for application to real valued matrices. An obvious improvement is to iterate the technique a number of times. This gives the technique greater flexibility since, after the initial pass, the leading diagonal of the matrix will have changed. The transformed elements of the leading diagonal may, in later passes, prove useful for the reduction of the magnitude of some of the off diagonal elements.

An additional change which may be made is to allow the diagonalisation technique to remove only a proportion of each off diagonal element during a given iteration. This may be achieved by changing the choice of λ in equation 6.15,

$$\lambda = -\alpha \frac{\mathbf{H}_{aa} \cdot \mathbf{H}_{ab}}{|\mathbf{H}_{aa}|^2} \tag{6.27}$$

where $\alpha \in [0,1]$ is a constant. This has the effect of transforming the iterated diagonalisation of the sampled Hessian matrix into a form of optimisation, taking only a small step in the right direction at each stage.

6.3.2 Fitting the Model in Practise

Once the basis is generated it is necessary to fit the model to the surface. As has already been explained, it was decided that linear splines would provide the most flexible set of basis functions for the fitting procedure. For the sake of simplicity, the sample data is scaled onto the unit interval [0,1] along each of the new basis vectors before fitting the model. Once the model has been fitted it is necessary to reverse this before displaying the data so that it retains some quantitative value.

Having generated the simultaneous linear equations which describe the best fit of the model for the chosen basis, it is necessary solve these equations. There are many techniques for the solution of such systems of equations, although the one selected for the variable separation technique is perhaps one of the least elegant.

Consider once again equation 6.20,

$$\sum_i \sum_j a_j f'_j(\mathbf{x}_i) f'_k(\mathbf{x}_i) = \sum_i f(\mathbf{x}_i) f'_k(\mathbf{x}_i)$$

In matrix notation, this may be represented by

$$\mathbf{Ma} = \mathbf{b}$$

where **M** is a matrix and **a** and **b** are vectors, **M** and **b** are known and

$$\begin{aligned} \mathbf{M}_{ij} &= \sum_k f'_i(\mathbf{x}_k) f'_j(\mathbf{x}_k) \\ \mathbf{a}_i &= a_i \\ \mathbf{b}_i &= \sum_k f(\mathbf{x}_k) f'_i(\mathbf{x}_k) \end{aligned}$$

The solution of equation 6.27 is given by

$$\mathbf{a} = \mathbf{M}^{-1}\mathbf{b}$$

A matrix inversion technique is therefore required to generate $\mathbf{M}^{-1}$, from which the solution to the simultaneous equations may be trivially generated. Singular Value Decomposition (SVD) was used to invert the matrix **M**. This technique has a number of advantages and is itself the justification for the use of the matrix inversion approach. SVD is capable of generating an RMS. best inverse to singular and ill-conditioned matrices, where in the first instance no inverse exists and in the second many techniques will fail. Alternative approaches to the solution of simultaneous linear equations generally rest upon matrix inversion, although they do not all explicitly use matrix manipulation, and are subject to the same problems. When curve fitting, there is no guarantee that the matrix **M** will not be ill-conditioned or indeed singular.

I Generate sampled Hessian matrix
    Finite differences
    Weighted quadratic approximation
II Diagonalise Hessian
    Iterated matrix diagonalisation
    Optimisation-like iterated matrix diagonalisation
III Invert diagonalising matrix to generate basis vectors
IV Transform and scale data onto new basis
V Use curve fitting to fit model to data
VI Rescale data to preserve quantitative information
VII Calculate Error

Figure 6.3 The structure of the variable separation visualisation technique

6.3.3 The Completed Algorithm

Finally, the various stages must be combined into a single algorithm. Each stage is, more or less, independent and the full description in pseudo code would be somewhat lengthy. The structure of the variable separation algorithm is therefore presented in figure 6.3 as a number of steps (noting possible alternative approaches).

The final step is perhaps one of the most important features of using regression-like techniques for visualisation. The user is provided with a measure of confidence in the results of the visualisation algorithm, which is essential if the visualisation is to communicate properly with the user. The error measure used for the variable separation visualisation technique is defined by,

$$E = \frac{\left(\sum_i \left(f(\mathbf{x}_i) - \sum_{j=0}^{N-1} f_j(\mathbf{b}_j \cdot \mathbf{x}_i) \right)^2 \right)^{\frac{1}{2}}}{\max f(\mathbf{x}) - \min f(\mathbf{x})} \qquad (6.28)$$

the RMS. error of the model divided by the range of the original function. The division by the range of the function allows the error to be presented as a proportion rather than as a raw number. This simplifies the comparison of different visualisations and is generally simpler for the user to understand.

6.4 A Comparative Study of Variable Separation and Projection Pursuit Regression

In order to test the effectiveness of the variable separation technique, it was compared upon a number of test problems to the technique of projection pursuit regression. During this study the PPR algorithm was encoded using the polytope (or simplex) optimisation algorithm for the iterative optimisation of the basis vectors. Furthermore, the smoothing cycle of the PPR algorithm was provided by the curve fitting technique described above. It must be noted that this encoding is not necessarily the most computationally efficient.

The visualisation techniques were each applied 32 times to each test function and the time taken, average error and best visualisation were recorded. The elapsed time for 32 runs of each technique was measured in real time rather than CPU time and so gives only a rough comparative measure (The experiments were performed upon a Sparc Station GX) and the errors of the visualisations were measured with a random sample of 256 points. Both techniques operated upon a further random sample of 256 points, the variable separation technique taking 16 samples of the Hessian matrix. Where finite differences were used to approximate the derivatives they were calculated with a step length, δ, of 0.01, and where locally weighted quadratic approximation was used, the weighting function was

$$w_{\mathbf{x}}(\mathbf{x}_i) = \frac{1}{50|\mathbf{x} - \mathbf{x}_i| + 1}$$

Both techniques approximated the functions of the model with 16 evenly spaced linear splines. The PPR algorithm was allowed up to 32 iterative cycles and the variable separation algorithm 32 iterations of the matrix diagonalisation stage.

For simplicity of presentation, all variables of the test functions were, when necessary, scaled onto the unit interval [0,1]. When applying the visualisation technique in practise, this process is automated, with the final visualisation being rescaled onto the ranges of the original parameters, allowing the user to view the visualisation in terms of the original variables. This was not thought necessary for the purposes of comparative testing.

6.4.1 An Initial Test Suite

The initial suite of test function developed for the comparative analysis of the two techniques was chosen to illustrate the expected behaviour of the variable separation technique, both in success and in failure. Table 6.1 shows the algebraic form and gives a short description of each test function.

The first function provides an example of a separable function,

$$f_1(\mathbf{x}) = x_0 x_1 \equiv \frac{1}{4}(x_0 + x_1)^2 - \frac{1}{4}(x_0 - x_1)^2$$

Functions f_2 and f_3 were chosen that they might fool the visualisation techniques since, from sample data, they appear very similar to f_1. The former is in fact separable

$$f_2(\mathbf{x}) = 0.01\left(x_0^2 + x_1^2\right) + x_0 x_1 \equiv 25(0.02x_0 + x_1)^2 - 24.99x_1^2$$

although the latter is not.

Function	Description
$f_1(\mathbf{x}) = x_0 x_1$	A simple separable function.
$f_2(\mathbf{x}) = 0.01\left(x_0^2 + x_1^2\right) + x_0 x_1$	A separable function which appears from samples to be, but for a small deviation, identical to f_1
$f_3(\mathbf{x}) = 0.01\left(x_0^3 + x_1^3\right) + x_0 x_1$	A non-separable function which appears from samples to be, but for a small deviation, identical to f_1
$f_4(\mathbf{x}) = x_0^2 x_1$	A simple non-separable function
$f_5(\mathbf{x}) = \dfrac{x_0}{x_1 + 1}$	A simple non-separable function
$f_6(\mathbf{x}) = 0.01\cos(100x_0x_1) + x_0x_1$	A non-separable function which appears from samples to be, but for a small high frequency deviation, identical to f_1
$f_7(\mathbf{x}) = 0.001\cos(1000x_0x_1) + x_0x_1$	A non-separable function which appears from samples to be, but for a small very high frequency deviation, identical to f_1
$f_8(\mathbf{x}) = 0.01(x_0x_1)^{100} + x_0x_1$	A non-separable function which appears from samples to be, but for a small high order deviation, identical to f_1
$f_9(\mathbf{x}) = 0.001(x_0x_1)^{1000} + x_0x_1$	A non-separable function which appears from samples to be, but for a small very high order deviation, identical to f_1

Table 6.1 The initial test suite

Functions f_4 and f_5 were chosen since they are perhaps two of the simplest possible non-separable functions. The final four functions were chosen for their high frequency (or order) components, a feature which is expected to cause problems for the variable separation visualisation technique. Note that the range of each function is the unit square, $\mathbf{x} \in [0,1]^2$.

6.4.2 Results for the Initial Test Suite

The average errors and elapsed times for the experiments upon each of the test functions are given in table 6.2. Table 6.3 shows the unit basis vectors identified by the best visualisation from each experiment for each of the test functions.

	PPR		Variable Separation using finite differences		Variable Separation using quadratic approximation	
Function	Time Taken (mins)	Mean Error (%)	Time Taken (mins)	Mean Error (%)	Time Taken (mins)	Mean Error (%)
f_1	127	0.72	1	0.34	4	0.64
f_2	115	0.62	2	0.76	5	0.73
f_3	142	0.41	1	0.14	4	0.50
f_4	299	1.17	1	0.72	5	0.74
f_5	312	2.03	2	0.64	4	0.52
f_6	141	1.59	4	5.68	4	0.86
f_7	132	1.10	5	1.60	5	0.61
f_8	124	1.02	4	1.44	4	4.96
f_9	120	0.26	5	0.35	4	0.69

Table 6.2 Results of experiments upon the initial test suite

	PPR		Variable Separation using finite differences		Variable Separation using quadratic approximation	
Function	Basis	Error (%)	Bases	Error (%)	Bases	Error (%)
f_1	(0.72,0.69) (-0.72,0.70)	0.05	(0.71,0.71) (-0.71,0.71)	0.05	(0.71,0.71) (-0.71,0.71)	0.05
f_2	(0.69,0.72) (-0.69,0.72)	0.05	(0.02,1.00) (0.00,1.00)	0.66	(0.02,1.00) (0.00,1.00)	0.62
f_3	(0.72,0.69) (-0.72,0.69)	0.06	(0.17,0.99) (-0.14,0.99)	0.11	(0.04,1.00) (-0.01,1.00)	0.42
f_4	(0.95,0.31) (-0.91,0.41)	0.30	(0.99,0.16) (-0.98,0.17)	0.13	(0.92,0.38) (-0.84,0.54)	0.41
f_5	(0.98,-0.18) (-0.79,0.62)	0.62	(0.32,0.95) (-0.27,0.96)	0.05	(0.32,0.95) (-0.26,0.97)	0.05
f_6	(0.73,0.68) (-0.73,0.69)	0.75	(0.74,0.67) (-0.99,0.15)	2.98	(0.27,-0.96) (0.27,0.96)	0.74
f_7	(0.72,0.70) (-0.72,0.70)	0.09	(0.68,-0.74) (0.69,0.73)	0.12	(0.02,-1.00) (0.01,1.00)	0.51
f_8	(0.71,0.70) (-0.71,0.70)	0.05	(0.71,0.71) (-0.71,0.71)	0.05	(0.71,0.71) (-0.71,0.71)	0.05
f_9	(0.69,0.72) (-0.69,0.72)	0.05	(0.71,0.71) (-0.71,0.71)	0.05	(0.71,0.71) (-0.71,0.71)	0.05

Table 6.3 Best visualisations of the functions in the initial test suite

The variable separation technique clearly outperforms the PPR technique in terms of computational expense. The results are a little less conclusive when examining the average and best errors of the visualisations however. It is interesting to note that for f_2, both variants of the variable separation technique identified the correct basis, although the best error of the PPR is, in fact, the lesser. As predicted, the variable separation technique utilising finite differences fails for function f_6 where there are high frequency components of the order of δ^1 (δ being the step length used to calculate the finite differences).

Suprisingly, when using weighted quadratic approximation to generate the second partial derivatives the variable separation technique performs poorly, on average, upon function f_8 although not upon function f_9. This may be due to the fact that the very high order component of f_9 is approximately zero over most of the range [0,1] and that the high order component of f_8 is poorly approximated by a quadratic. Excepting these cases, the variable separation technique appears to compare favourably with PPR upon the functions in the initial test suite. For most of the functions in this test suite, the use of finite differences for the approximation of derivative information appears to be the more robust.

6.4.3 A Second Test Suite

The De Jong test suite for function optimisation (q.v. table 2.2) was selected to provide a second, less contrived, test suite for the variable separation and PPR visualisation techniques. Unfortunately, the computational inefficiency of PPR removed the possibility of using the 30 dimensional function f_4 from De Jong's test suite. The two variants of the variable separation technique and PPR were therefore compared upon the remaining four functions. Note that the parameters of the De Jong test functions have been scaled onto the unit interval [0,1].

6.4.4 Results for the Second Test Suite

As for the results for the initial test suite, the average errors and elapsed times for the experiments upon each of the test functions from the second test suite are given in table 6.4. The unit basis vectors and errors of the best visualisations for each of the functions are given in table 6.5.

	PPR		Variable Separation using finite differences		Variable Separation using quadratic approximation	
Function	Time Taken (mins)	Mean Error (%)	Time Taken (mins)	Mean Error (%)	Time Taken (mins)	Mean Error (%)
De Jong's f_1	319	0.14	11	0.10	38	0.10
De Jong's f_2	216	9.68	5	3.47	13	7.98
De Jong's f_3	327	5.65	39	2.24	182	3.07
De Jong's f_5	72	17.65	4	17.98	10	17.56

Table 6.4 Results of experiments upon the second test suite

	PPR		Variable Separation using finite differences		Variable Separation using quadratic approximation	
Function	Basis	Error (%)	Basis	Error (%)	Basis	Error (%)
De Jong's f_1	(1.00,0.01,−0.03) (−0.01,1.00,0.01) (0.03,−0.01,1.00)	0.09	(1.00,0.00,0.00) (0.00,1.00,0.00) (0.00,0.00,1.00)	0.08	(1.00,0.00,0.00) (0.00,1.00,0.00) (0.00,0.00,1.00)	0.09
De Jong's f_2	(0.96,0.27) (−0.96,0.29)	2.30	(0.97,−0.26) (0.95,0.31)	2.32	(0.95,0.30) (−0.84,0.54)	3.89
De Jong's f_3	(1.00,0.00,−0.03,0.03,0.03) (0.02,0.99,0.15,0.00,0.06) (0.13,0.30,0.62,−0.15,0.70) (−0.04,−0.04,0.25,0.95,0.19) (−0.13,0.02,0.01,−0.04,0.99)	2.38	(1.00,0.00,0.00,0.00,0.00) (0.00,1.00,0.00,0.00,0.00) (0.00,0.00,1.00,0.00,0.00) (0.00,0.00,0.00,1.00,000) (0.00,0.00,0.00,0.00,1.00)	1.89	(0.96,0.12,−0.20,0.12,0.00) (0.22,0.92,0.01,0.26,−0.19) (0.09,0.12,0.95,0.17,−0.20) (0.04,−0.12,0.18,0.89,−0.40) (0.00,0.02,0.26,−0.19,0.95)	1.67
De Jong's f_5	(1.00,0.03) (−0.06,1.00)	14.17	(0.00,1.00) (1.00,0.00)	15.17	(1.00,0.02) (0.03,1.00)	14.45

Table 6.5 Best visualisations of the functions in the second test suite

The first and third of the De Jong test functions are, in their initial forms, separable functions and it is interesting to note that the PPR algorithm has not retained the original bases for either of these functions. The variable separation technique, when utilising quadratic approximation for the generation of second partial derivatives, has also failed to retain the original basis for the third of the De Jong test functions. The third test function is a step function, having a number of discontinuities. It is likely, therefore, that the failure of the variable separation technique upon this function occurs due to the high order components introduced by such discontinuities. Such discontinuities may also be described in terms of high frequency components and it may therefore appear surprising that, when using finite differences, the variable separation technique succeeds in identifying the correct basis. However, the function in question is not, in fact, cyclic and it is likely that this accounts for the success of the variable separation technique in this case.

The remaining pair of functions are not separable, and all three techniques identified similar sets of basis vectors. As for the initial test suite, the variable separation technique compares favourably with the PPR technique in terms of the mean errors of the visualisations and very well in terms of computational expense. Furthermore, upon these test functions, the use of finite differences for the approximation of the second partial derivatives appears to be more effective, both in terms of mean errors and computational expense, than the use of quadratic approximation.

6.4.5 A Final Test Suite

The final test suite is comprised of functions which have some practical application, as opposed to those functions in the previous test suites which were constructed purely for the purposes of comparative analysis - the first for the visualisation techniques, the second for the GA. The algebraic forms of these test functions and their origins are given in table 6.6. As for the above test functions, the parameters range over the unit interval, [0,1].

Function	Origin
$f_{10}(\mathbf{x}) = -2\log_{10}\left\{\frac{2.51}{(99{,}999{,}700x_1 + 3000)\sqrt{0.023x_0 + 0.002}} + \frac{x_2}{3.7(19x_3 + 1)}\right\} - \frac{1}{\sqrt{0.023x_0 + 0.002}}$	Colebrook and White's formula for the friction factor in turbulent pipe flow
$f_{11}(\mathbf{x}) = \min(x_0 + x_1 + x_2 + x_3, x_0 + x_5 + x_2 + x_4, x_3 + x_5 + x_1 + x_4)$	The length of the shortest path for the four city non-geometric travelling salesman problem
$f_{12}(\mathbf{x}) = \int_0^1 \left(\frac{\lvert\mathbf{x}\rvert^2 + \left(\sum_{i=0}^{5}(i+1)x_i y^i\right)^2}{\lvert\mathbf{x}\rvert^2 - \lvert\mathbf{x}\rvert\sum_{i=0}^{5} x_i y^{i+1}} \right) dy$	The brachistochrone problem applied to a family of normalised sixth order polynomials

Table 6.6 The final test suite

The first function of the final test suite is derived from Colebrook and White's formula for the friction factor in turbulent pipe flow,

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left\{\frac{2.51}{RE\sqrt{f}} + \frac{K}{3.7D}\right\} \tag{6.29}$$

where f is the friction factor, RE Reynold's number, K a coefficient of surface roughness, D the diameter of the pipe and

$$\begin{aligned} &0.002 \le f \le 0.025 \\ &3000 \le RE \le 100{,}000{,}000 \\ &0 \le K \le 1 \\ &1 \le D \le 20 \end{aligned}$$

are reasonable ranges for these four parameters. Setting

$$\begin{aligned} x_0 &= \frac{f - 0.002}{0.023} \\ x_1 &= \frac{RE - 3000}{99{,}999{,}700} \\ x_2 &= K \\ x_3 &= \frac{D - 1}{19} \end{aligned}$$

and taking the left hand side away from both sides of the equation yields the first of the final test functions - a normalised version of Colebrook and Whites formula

$$f_{10}(x) = -2\log_{10}\left\{\frac{2.51}{(99{,}999{,}700x_1 + 3000)\sqrt{0.023x_0 + 0.002}} + \frac{x_2}{3.7(19x_3 + 1)}\right\} - \frac{1}{\sqrt{0.023x_0 + 0.002}} \qquad (6.30)$$

the roots of which yield the solutions to Colebrook and White's formula.

The second of the functions in the final test suite gives the length of the shortest tour of the four city non-geometric travelling salesman problem (TSP), the structure of which is illustrated in figure 6.4.

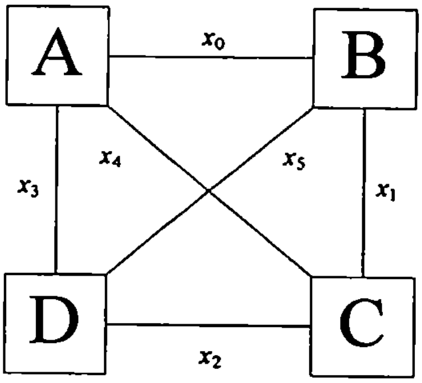


Figure 6.4 The four city non-geometric TSP

The parameters x_0 to x_5 represent the lengths of the paths between the cities A-D, next to which they are placed in figure 6.10. By non-geometric, it is meant that the lengths of the paths are not necessarily consistent with a geometric interpretation of figure 6.10 - for example parameters x_0 to x_3 may be non-zero whilst parameters x_4 and x_5 are zero. Each tour through the four cities must visit each city once and only once, except for the first city in which the tour must also end. For the four city non-geometric TSP there are but three unique paths, namely ABCDA, ABDCA and ADBCA. All other tours are isometric to one of these three. The lengths of these tours are given by

$$
\begin{aligned}
length(ABCDA) &= x_0 + x_1 + x_2 + x_3 \\
length(ABDCA) &= x_0 + x_5 + x_2 + x_4 \\
length(ADBCA) &= x_3 + x_5 + x_1 + x_4
\end{aligned}
$$

and for each set of parameters the minimum of these lengths provides the second function of the final test suite,

$$f_{11}(\mathbf{x}) = \min(x_0 + x_1 + x_2 + x_3, x_0 + x_5 + x_2 + x_4, x_3 + x_5 + x_1 + x_4) \qquad (6.31)$$

The lengths the paths between the cities are assumed to lie within the interval [0,1].

The last function in this final test suite is based upon the brachistochrone problem proposed by John Bernoulli in 1696. The brachistochrone is a curve joining two points in the plane $\mathbf{x}_0$ and $\mathbf{x}_1$, such that a bead placed upon this curve will, under gravity, traverse its length in minimal time. This problem is not soluble by calculus and led to the development of the calculus of variations (Pars, 1962) by which it was solved. The problem is illustrated graphically in figure 6.5.

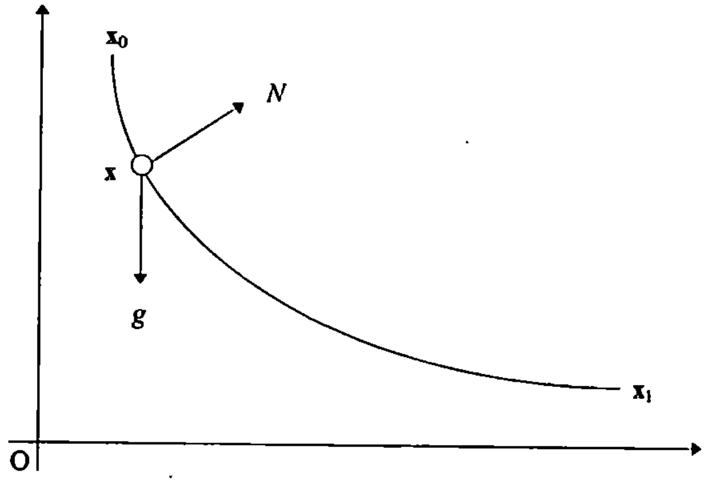


Figure 6.5 The brachistochrone problem

Given a curve $y = f(x)$, the time taken for the bead to travel from $\mathbf{x}_0$ to $\mathbf{x}_1$ is given by

$$t = \frac{1}{\sqrt{2g}} \int_{x_0}^{x_1} \left(\frac{1 + y'^2}{y} \right)^{\frac{1}{2}} dx \tag{6.32}$$

where y' represents $\frac{dy}{dx}$.

Since this equation represents a functional (an operator which acts upon functions) rather than a function, it is not possible to apply the visualisation techniques to it directly - the single parameter $y(x)$ of the functional is a function itself. In order that this problem may be visualised, the functional is applied to a constrained set of curves which may be represented parametrically - in this case sixth order polynomials. These polynomials are constrained to pass through the start and end points (0, 1) and (1, 0), and therefore take the form

$$f(x) = 1 - \frac{\sum_{i=0}^{5} a_i x^{i+1}}{|\mathbf{a}|} \qquad \mathbf{a} \neq \mathbf{0} \tag{6.33}$$

For the purposes of the visualisation, the six coefficients a_i are assumed to lie in the range 0-1. Substituting equation 6.33 into equation 6.32 and multiplying the result by $\sqrt{2g}$ yields the six dimensional test function

$$f_{12}(\mathbf{x}) = \int_0^1 \left(\frac{|\mathbf{x}|^2 + \left(\sum_{i=0}^{5} (i+1) x_i y^i \right)^2}{|\mathbf{x}|^2 - |\mathbf{x}| \sum_{i=0}^{5} x_i y^{i+1}} \right) dy \tag{6.34}$$

which is calculated by trapezoidal approximation. This function is undefined at the origin (since there is a pole at this point) and therefore, for the purposes of the visualisation techniques, the function is set to 0 there.

6.4.6 Results for the Final Test Suite

As for the results upon the prior test suites, the average errors and elapsed times for the experiments upon each of these test functions are given in table 6.7. Once again, the unit basis vectors and errors of the best visualisations for each function are presented in table 6.8.

	PPR		Variable Separation using finite differences		Variable Separation using quadratic approximation	
Function	Time Taken (mins)	Mean Error (%)	Time Taken (mins)	Mean Error (%)	Time Taken (mins)	Mean Error (%)
f_{10}	1154	1.88	68	0.75	145	2.21
f_{11}	3114	25.91	92	6.35	364	3.18
f_{12}	3748	55.53	346	15.65	549	13.56

Table 6.7 Results of experiments upon the final test suite

For the first of the functions of the final test suite it is interesting to note that, although the function is not separable, the best visualisation (identified by the variable separation technique using finite differences) utilises the original basis vectors. Both PPR and variable separation utilising quadratic approximation failed to identify this basis. Interestingly, both PPR and variable separation utilising quadratic approximation settled upon similar bases, although the former generated the better visualisations of this function. For the second of the final test functions, the best visualisations were identified by the variable separation technique utilising quadratic approximation. Perhaps counter-intuitively, given the apparent linear nature of this function, none of the visualisation techniques settled upon the original basis - although for both PPR and variable separation utilising finite differences, four of the six basis vectors of the best visualisations are close to the original bases. For the final function, the best visualisation of both PPR and, to a lesser extent, variable separation utilising quadratic approximation settle upon basis vectors which are similar to the original bases. Despite this similarity, the PPR technique generated the worst visualisations of this function.

	PPR		Variable Separation using finite differences		Variable Separation using quadratic approximation	
Function	Basis	Error (%)	Basis	Error (%)	Basis	Error (%)
f_{10}	(1.00,0.00,0.00,0.00) (0.13,0.99,−0.02,−0.08) (−0.01,0.00,1.00,0.02) (−0.01,0.01,0.01,1.00)	0.47	(1.00,0.00,0.00,0.00) (0.00,1.00,0.00,0.00) (0.00,0.00,1.00,0.00) (0.00,0.00,0.00,1.00)	0.34	(1.00,0.00,0.01,−0.02) (−0.13,0.97,0.20,0.09) (0.03,−0.08,1.00,0.01) (0.06,−0.06,0.01,1.00)	1.13
f_{11}	(0.49,0.41,0.54,0.12,0.32,0.39) (0.01,0.99,−0.11,0.04,0.00,0.08) (0.54,−0.07,0.52,−0.29,−0.33,−0.53) (0.04,−0.06,0.00,0.97,−0.23,0.00) (−0.01,0.00,−0.07,−0.23,0.96,0.00) [illegible]	5.09	(0.57,0.00,0.51,−0.43,−0.09,−0.52) (0.00,1.00,0.00,0.00,0.00,0.00) (0.00,0.00,1.00,0.00,0.00,0.00) (0.12,0.00,0.12,0.63,−0.75,−0.12) (0.00,0.00,0.00,0.00,1.00,0.00) (0.00,0.00,0.00,0.00,0.00,1.00)	2.31	(0.54,0.02,0.31,−0.29,−0.37,−0.54) (−0.01,0.97,0.23,0.04,−0.02,−0.02) (−0.86,0.01,0.25,0.00,−0.09,−0.44) (−0.03,0.03,0.04,0.66,−0.75,−0.04) (0.12,−0.05,−0.11,0.50,0.84,−0.12) [illegible]	2.15
f_{12}	(1.00,−0.05,−0.02,0.02,−0.02,0.06) (0.00,0.99,0.00,0.00,−0.05,0.13) (−0.01,0.01,1.00,0.01,0.01,−0.06) (−0.03,0.02,0.06,1.00,0.03,0.01) (0.05,−0.13,0.10,−0.10,0.98,−0.01) [illegible]	7.70	(0.56,−0.13,0.43,0.48,−0.26,−0.44) (−0.69,0.43,−0.39,0.25,−0.03,0.31) (−0.32,0.79,0.46,−0.06,0.06,0.13) (−0.48,−0.63,0.17,0.33,0.09,−0.43) (0.72,−0.38,−0.12,−0.03,0.25,0.51) [illegible]	7.24	(0.66,−0.26,−0.52,−0.35,0.10,0.29) (0.04,0.85,0.28,0.23,0.12,−0.35) (0.57,−0.19,0.77,−0.13,−0.10,0.12) (0.24,−0.32,−0.13,0.90,−0.08,−0.06) (−0.07,0.06,0.10,0.38,0.93,−0.23) [illegible]	6.92

Table 6.8 Best visualisations of the functions in the final test suite

As for the previous test suites, it would appear that the use of finite differences for the generation of partial derivative information results in a more robust visualisation technique than does the use of quadratic approximation. For the functions in the final test suite, the variable separation technique utilising finite differences consistenly outperforms the PPR technique both in terms of computational expense and in terms of accuracy.

The best visualisations generated by the variable separation technique utilising finite differences of each function in the final test suite are illustrated in figures 6.6 - 6.8.

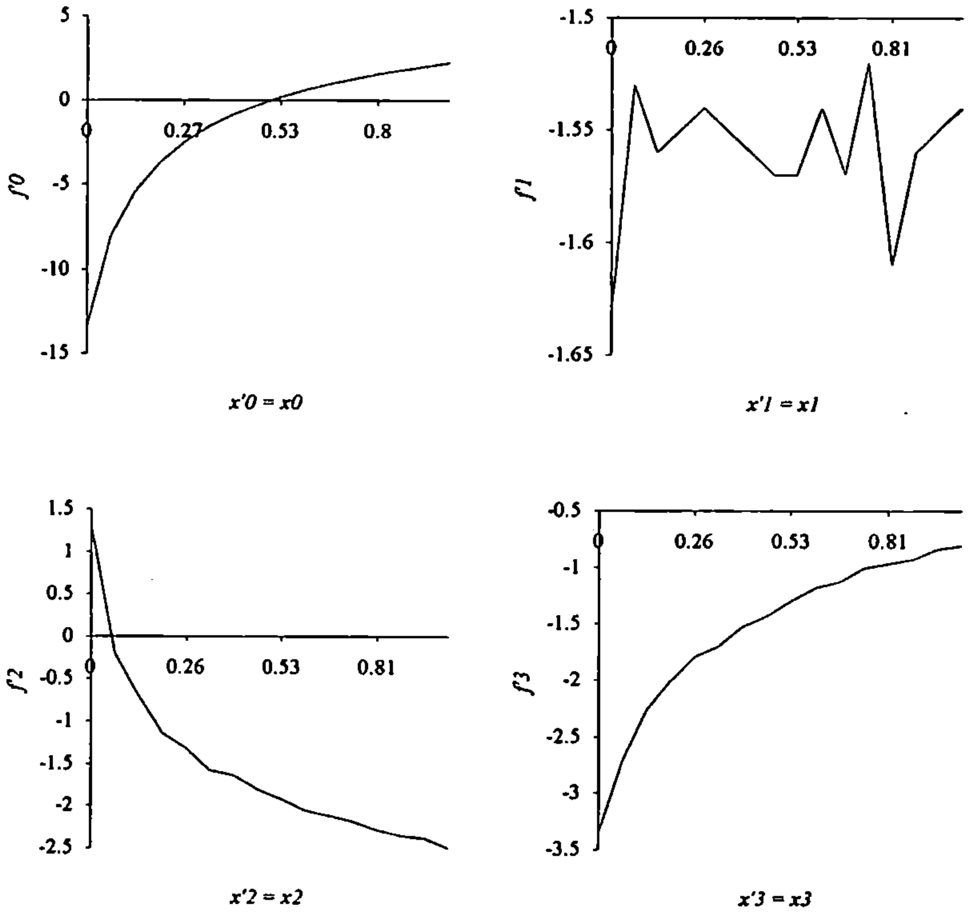


Figure 6.6 The best visualisation of f_{10}

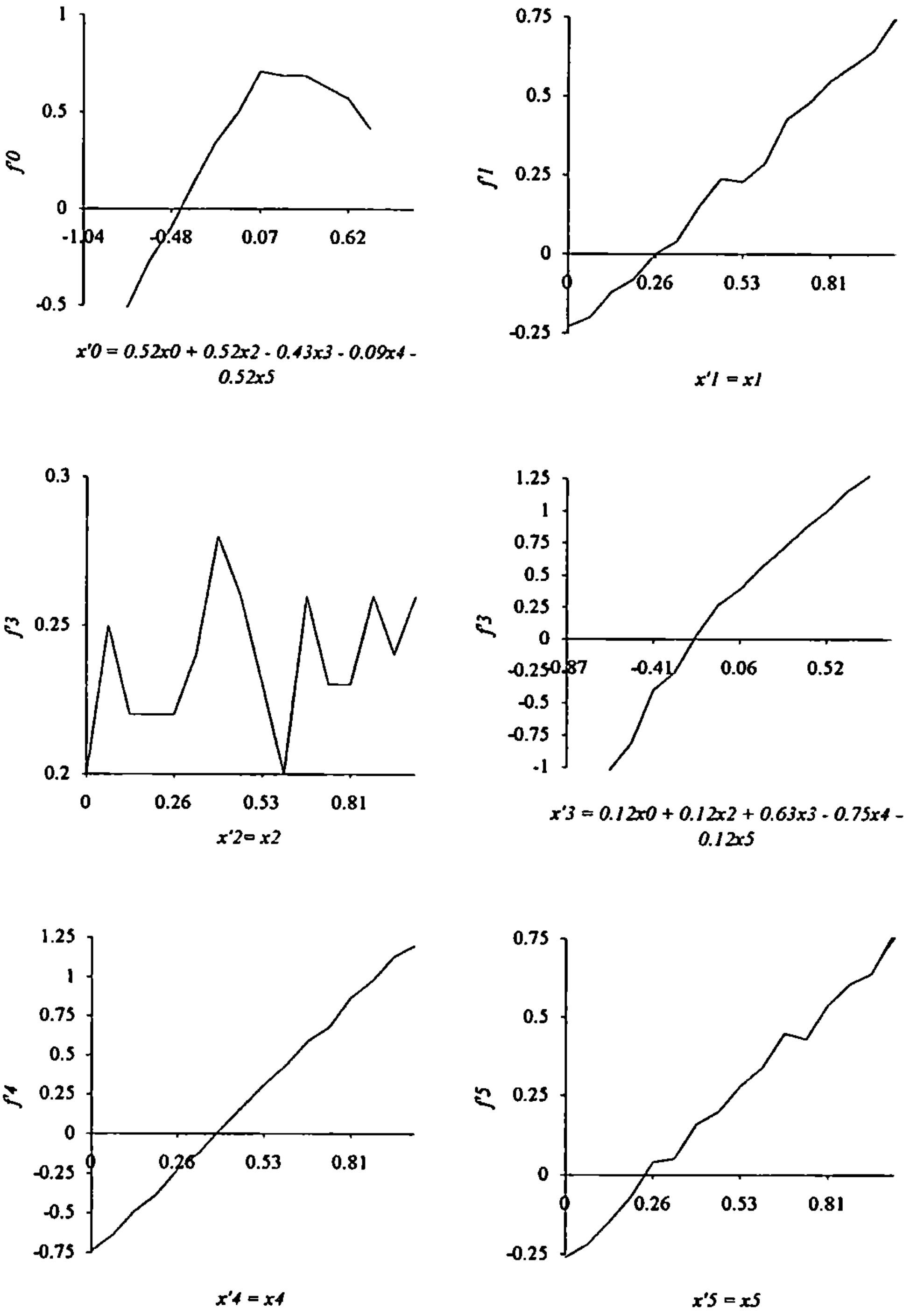


Figure 6.7 The best visualisation of f_{11}

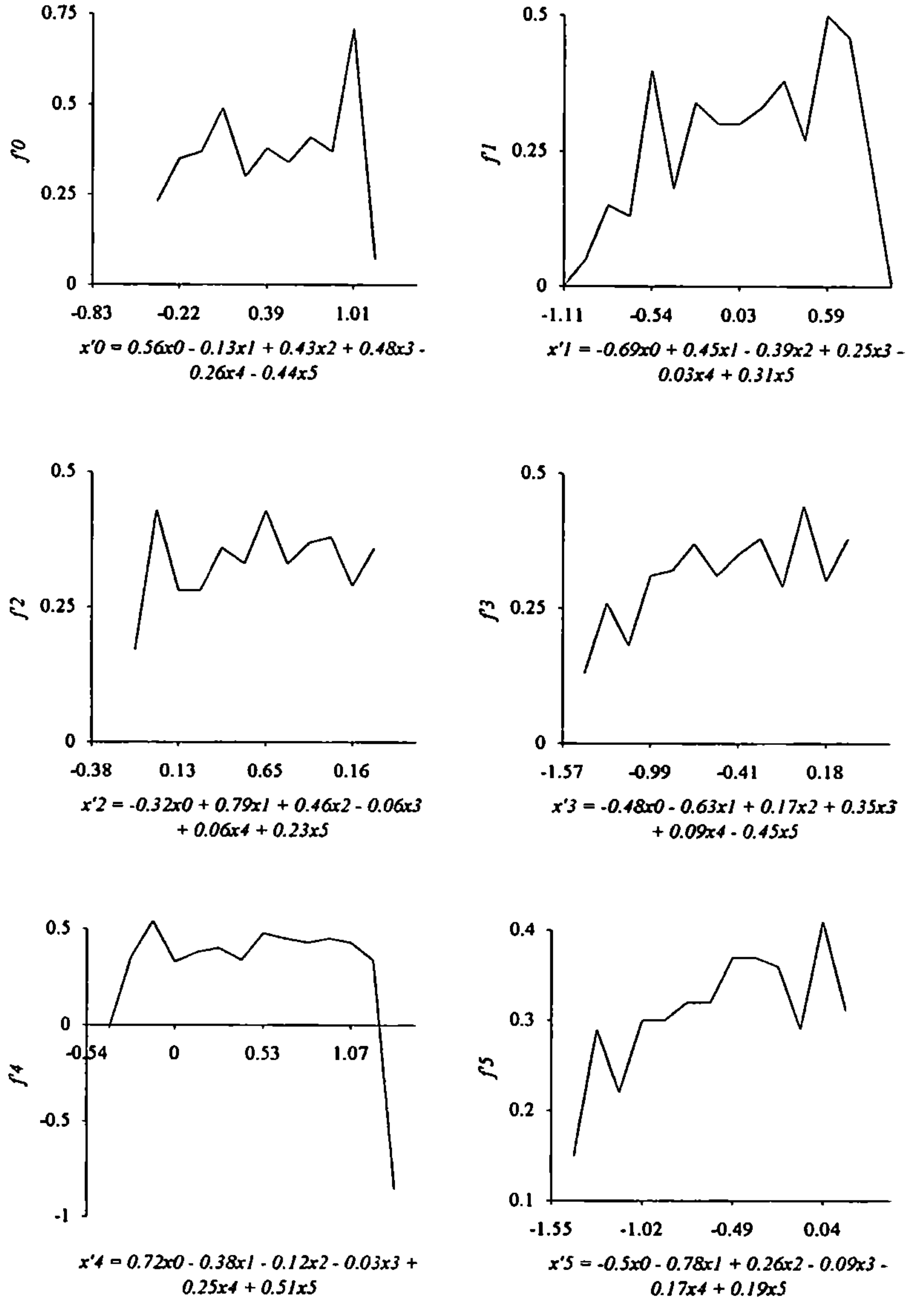


Figure 6.8 The best visualisation of f_{12}

Although the function f'_1 of the visualisation of f_{10} appears to be very noisy, it has a significantly smaller range than the remaining three functions. This would indicate that the parameter x'_1 does not contribute greatly to the function f_{10} and could possibly be ignored with little loss of information (x'_1 is identical to x_1 - a normalised Reynolds number). The remaining functions are smooth and monotonic which is surprising given the complexity of the original function.

As for function f'_1 of the visualisation of f_{10}, function f'_3 of the visualisation of f_{11} appears, at first, to be very noisy. However, it has a significantly smaller range than the remaining five functions - once again indicating that it may be possible to ignore this parameter without significant loss of information. The remaining functions, with the exception of f'_0, appear to be close approximations to linear functions, which is not unexpected given the apparent linear nature of the original function. Note that the functions f_0 and f_3 are not defined for the full range of their respective parameters due to the fact that the original data did not cover these ranges.

As for the previous visualisations, some of the univariate functions of the visualisation of f_{12} are not defined for the full range of their respective parameters. Once again, this is due to the original data not spanning the range of those parameters. Unlike the previous visualisations, each of the univariate functions contributes significantly to the visualisation - the range of each of these function is comparable. As would be expected, given the complexity of the function f_{12}, the forms of these univariate functions are complex and noisy. The error of this visualisation is relatively high (at 7.24%) and this, combined with the complexity of the univariate functions, makes it difficult to draw any firm conclusions about the nature of the original function.

6.5 Conclusions

The variable separation technique provides a rapid, and in general, effective tool for the visualisation of high dimensional surfaces generated by functions of many parameters. The results show a favourable comparison to PPR for this technique in terms of the errors of the visualisations for the various functions in the test suites. In terms of computational expense, the variable separation technique compares very well indeed although, as was stated at the beginning of this chapter, the PPR algorithm used here is not necessarily the most computationally efficient and some allowances must therefore be made.

The use of finite differences for the approximation of derivative information is consistently more computationally efficient than the use of quadratic approximation. Furthermore, although the use of quadratic approximation is often an improvment upon the use of finite differences in terms of the errors of the visualisation, the use of finite differences would appear to provide a more robust visualisation technique. For a number of the test functions, it could be argued that the variable separation technique utilising quadratic approximation for the generation of derivative information overfits the data.

The variable separation technique has a number of additional advantages which have not been discussed. Firstly, the variable separation technique generates the basis vectors for the projection independently of the functions of the transformed parameters. This allows the user to apply a wide range of curve fitting techniques to the data without the necessity of recalculating the basis vectors at each stage. For example. an initial visualisation may use a small number of linear splines to approximate the univarate functions of which the model is comprised. At a later stage, more splines may be used to improve the accuracy of the visualisation without the necessity of recalculating the basis vectors - a computationally expensive task.

A further advantage is that it is relatively simple to constrain the basis identification technique so that the transformed parameters do not include terms of different physical dimension. For example consider the function

$$V = l_1 \times l_2 \times v \times t \tag{6.35}$$

which describes the volume of material, V (m^3), which flows through a rectangular pipe with cross section l_1 (m) by l_2 (m) at a velocity v (ms^{-1}) over time t (s). For a meaningful visualisation it may be desirable that the length, velocity and time terms are not linearly combined. For the variable separation technique this is simply achieved during the diagonalisation of the Hessian - any step which requires placing a non-zero value in an element of **D** which corresponds to the combination of two such parameters is skipped. For the PPR algorithm these disallowed combinations act as constraints for the optimisation process. Such constraints may seriously compromise the performance of traditional optimisation techniques, although the simplex (polytope) algorithm used during this study will not violate this type of constraint provided that the initial sample of points are feasible (do not violate the constraints). The visualisation techniques were compared upon this constrained function and the results of this experiment are given in tables 6.9 and 6.10.

PPR		Variable Separation using finite differences		Variable Separation using quadratic approximation	
Time Taken (mins)	Mean Error (%)	Time Taken (mins)	Mean Error (%)	Time Taken (mins)	Mean Error (%)
1154	33.97	32	21.21	150	11.52

Table 6.9 Results of experiment upon the constrained function

PPR		Variable Separation using finite differences		Variable Separation using quadratic approximation	
Basis	Error (%)	Basis	Error (%)	Basis	Error (%)
(0.60,0.80,0.00,0.00) (0.06,1.00,0.00,0.00) (0.00,0.00,1.00,0.00) (0.00,0.00,0.00,1.00)	8.67	(0.71,0.71,0.00,0.00) (−0.71,0.71,0.00,0.00) (0.00,0.00,1.00,0.00) (0.00,0.00,0.00,1.00)	8.53	(0.62,0.78,0.00,0.00) (−0.63,0.78,0.00,0.00) (0.00,0.00,1.00,0.00) (0.00,0.00,0.00,1.00)	8.00

Table 6.10 Best visualisations of the constrained function

It is interesting to note that although the variable separation technique utilising finite differences does not generate the visualisations with the smallest errors, it does identify the optimal basis. This may be a result of over-fitting to the data on the part of the other techniques.

6.6 Further Work

Clearly, the second partial derivatives at the sample points do not provide a sufficiently robust measure of the codependance of parameters. As has been shown, high frequency (or order) components of the function invariably lead to unnecessary errors in the visualisation. Clearly, one of the first tasks in improving the effectiveness of the variable separation technique would be to define a better measure of codependance - one for which these high frequency (or order) components were not so disruptive.

Furthermore, the matrix diagonalisation technique used for the diagonalisation of the Hessian is merely a adaptation of a technique for the diagonalisation of real valued symmetric matrices. It would be of great advantage, therefore, to develop a technique specifically for the diagonalisation of vector valued matrices. Unfortunately, preliminary investigations into the possibility of developing such a technique have, so far, proven unsuccessful.

The model to which both the variable separation technique and PPR seek to fit the data bears a great deal of similarity to the model to which a single hidden layer feed-forward neural network (Muller & Reinhardt, 1990) attempts to fit data and such neural networks may be considered as a form of non-linear regression. Figure 6.9 shows the structure of one such neural network where the x_i are the input parameters, the w_{ij} and v_j are weights, the S_j are sigmoid functions and the y is the output. Where a number of lines converge, the inputs which they represent are assumed to be summated.

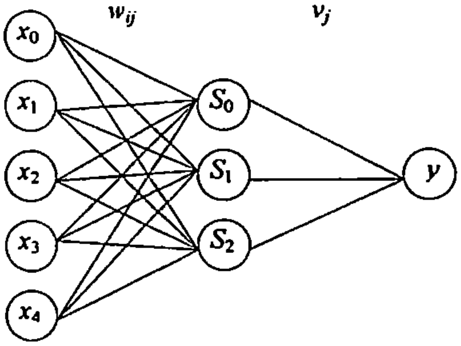


Figure 6.9 A feed-forward neural network

In algebraic notation, the neural network in figure 6.9 is given by

$$y = \sum_{i=0}^{4} \sum_{j=0}^{2} v_j S_j \left(w_{ij} x_i \right) \tag{6.36}$$

and this may be consider as a less general form of the model to which the data is fitted under variable separation and PPR. Previous research has compared PPR and neural networks, and the two techniques have been shown to yield similar results (Ripley, 1994). It would be of interest to compare the relative performance of such neural networks, variable separation and PPR. A clear difficulty for the variable separation technique is that it uses second partial derivatives which, as has been explained, are not meaningful for the scatter data to which PPR and neural networks are usually applied. However, both neural networks and PPR implicitly assume that a regression surface for the data exists, and this ideal regression surface does have meaningful derivatives. The locally weighted quadratic approximation approach for the generation of second partial derivatives, or an improvement of it, may possibly generate a good approximation to the second partial derivatives of the ideal underlying regression surface. If so, this approach will allow the generalisation of the variable separation technique to scatter data. Furthermore, before the variable separation technique may be compared to such neural networks, it is necessary to adapt the technique so that the number of univariate functions in the model is not constrained to the dimension of the input data. This is a desirable feature for a visualisation technique, but not for data approximation.

7 Conclusions

Each of the chapters of this thesis present conclusions upon the research documented therein. This chapter reiterates those conclusions and furthermore draws conclusions upon the research project as a whole, taking into consideration the original research goals.

7.1 An Empirical Investigation of the Building Block Hypothesis

This chapter documents the development of a technique for the extraction of highly sampled building blocks from the populations of the GA. These building blocks were subsequently used for the empirical analysis of the building block hypothesis. The extraction technique was shown to isolate those building blocks occurring most regularly within the populations of the GA at relatively little computational expense through the use of a clustering algorithm.

The building block hypothesis relies upon the assumption that the fitness of a chromosome is dependant upon the fitness of the building blocks from which it is constructed. If this were so, the fitness of a schema constructed from a pair of building blocks would depend upon the fitness of those constituent building blocks. Through combinatorial analysis it was shown that, upon a range of test functions, fit building blocks do not combine to yield significantly fitter schemata than those that result from the addition of randomly generated building blocks to fit building blocks (for the purpose of this analysis, the fitness of a building block was taken to be the static fitness of that building block - the mean fitness of the chromosomes which are matched by it).

These results suggest that the static building block hypothesis (the building block hypothesis where fitness is taken to mean static fitness) is fundamentally flawed.

The static building block hypothesis is not generally considered to be an accurate picture of the mechanics of the GA. The building block hypothesis, in its most general form, relies upon the dynamically changing fitnesses of building blocks defined by the mean fitness of the chromosomes in the current population which are matched by it. The relationship between the static fitness of a building block and its utility to the GA was therefore examined. The utility of a building block was defined to be the number of times it was sampled by the GA during the search process - the assumption that fit building blocks propagate justifies this choice of measure of utility. The dynamic fitness of a building block is, unfortunately, an unsuitable measure for such an analysis since it changes from generation to generation. A linear regression of the static fitness of building blocks and their utility revealed that these measures are related and consequently that the building block hypothesis is likely to be more closely related to the static building block hypothesis than is generally believed.

7.1.1 Further Work

The empirical analysis of building blocks documented in this chapter was performed upon the De Jong test functions. Clearly, this is a restricted domain and it would be of advantage to apply the same analysis upon a wider class of function - particularly the Royal Road functions developed to encourage the exploitation of building blocks by the GA.

7.2 An Alternative Description of the Action of Crossover

The description of crossover as a rotation in the parameter space yielded a pair of performance models of the GA upon the simple function $f(x)=|x|^2, x \in [0,1)$. These models were shown to approximate the performance of the GA accurately, in both the short and the long term. The accuracy of these models provides validation of the initial assumption - that crossover can be described as a rotation in the parameter space.

The role of crossover in the GA is generally believed to be in the recombination of building blocks. However, the analysis of the building block hypothesis revealed that it was unlikely that the GA was combining building blocks to construct solutions. Some alternative account of the operation of the GA, and in particular of crossover, is therefore required. The description of crossover as a rotation in the parameter space provides such an account since it does not rely in any way upon the notions of schema or building block. Furthermore, unlike the assumptions of the building block hypothesis, this description is valid for a wide range of chromosomal representations (real coding, for example).

The mechanics of the GA search process may be explained in terms of this rotational description of crossover. Initially, the population consists of a diverse set of parameter sets (defined by the randomly generated chromosomes). The effect of crossover during the early generations of the GA is therefore to make large scale changes to these parameter sets. The selection process rejects those parameter sets of low fitness and the population begins to converge on the fitter regions of the search space. The effect of crossover upon the newly created, less distributed, population is therefore less marked. The GA may be described as initially performing a coarse search, making large scale changes to the population and allowing the survival of poor solutions. As the population converges, the search becomes more and more fine grained and the increasing selection pressure reduces the likelihood of poor solutions surviving. This search process is not unlike that of simulated annealing and this description goes some way to explain the similarities in performance between these techniques.

7.2.1 Further Work

The description of crossover as a rotation is an approximation - the distribution of offspring more closely represents a rectangle. A model built upon a more accurate approximation of the effect f crossover would itself be more accurate. The use of these models as predictive tools is possible, although not very accurate. It would be of great advantage therefore if a model could be derived for a more general class of functions. For example, consider the class of quadratic functions

$$f(\mathbf{x}) = a\mathbf{x}^{\mathrm{T}}\mathbf{x} + \mathbf{b}^{\mathrm{T}}\mathbf{x} + c$$

for arbitrary scalars a and c, and vector **b**. Such functions may be used to approximate small regions of a surface - a model built for such a function might therefore still hold predictive power upon arbitrary classes of functions during the latter stages of the GA, once the population has converged and the neighbourhood of each pair of chromosomes may be accurately approximated by such a quadratic.

A number of alternative crossover techniques have been proposed for the real coded GA, most of which attempt to emulate binary crossover to some extent. The similarity between crossover and a rotation in the parameter space suggests that such a rotation could be used in place of crossover - providing a new alternative crossover operator.

7.3 A Technique for the Visualisation of High Dimensional Surfaces

The variable separation visualisation technique compares favourably against the statistical technique of projection pursuit regression (PPR). In terms of computational expense, variable separation consistently outperforms PPR, and in terms of the errors of the visualisations variable separation performs well in comparison. As was predicted, the variable separation technique encounters difficulties when presented with functions containing high frequency (or order) components. Clearly, this represents a major failing of the technique and as a result, variable separation is not suitable for a wide range of functions.

An advantage of the variable separation approach is that it is simple to constrain the basis vector to include only terms of the same physical dimension. It may be desirable to the user that the visualisation does not include bases which combine velocity terms and distance terms, for example. This is more difficult for PPR, since it represents a constraint upon the optimisation cycle. Traditional optimisation technique may encounter difficulties when faced with such constraints - possibly resulting in a significant degradation in performance of the PPR algorithm.

7.3.1 Further Work

The main failing of the variable separation technique is its poor performance upon functions with high frequency (or order) components. This results from the use of second partial derivatives as a measure of the codependance of pairs of parameters. An alternative measure must therefore be found - one for which a suitable diagonalisation technique may be devised.

The technique for the diagonalisation of the Hessian matrix is simply an adaptation of a technique for the diagonalisation of real valued symmetric matrices and does not therefore guarantee optimal results. It would be of benefit, therefore, to develop a diagonalisation technique for symmetric vector valued matrices, although preliminary investigations into the possibility of such techniques have not been fruitful.

Finally, there are many similarities between the model to which the variable separation technique and PPR fit the surface data and a single hidden layer feed-forward neural network. Such networks can be considered to be a less general form of regression technique. Comparative studies have shown that PPR and neural networks yield similar results, and it would therefore be of interest to compare variable separation to such neural networks. Some work must be done to adapt the variable separation technique before this is possible. In general, neural networks are applied to scatter data for which no meaningful derivatives exist. Furthermore, it would be desirable to increase the number of univariate functions in the model to which variable separation fits the data above the dimension of the parameter set.

7.4 The Original Research Goals

As has been explained in the introductory chapter of this thesis, the original research goals were to create an accountable optimisation technique, which in addition to the generation of an optimal or near optimal solution would provide reasons for the optimality of that solution. This was to be achieved through the extraction of building blocks from the populations of the GA. Given the building block hypothesis, it was reasoned that those building blocks which were sampled most regularly by the GA would represent important relationships between parameters. The relationships could be used to justify the solution upon which the GA had converged. Furthermore, it was intended that the information contained within these building blocks could be used for the visualisation of the problem space.

The first of the research goals was to develop a technique for the extraction of building blocks from the populations of the GA. This extraction process was to concentrate upon the building blocks of highest utility and involve as little computational expense as possible. This goal was met, and computational expense was kept to a minimum through the use of a clustering algorithm (although for one of the De Jong test functions, computational expense was a serious issue resulting in the redefinition of that function).

The second of the original research goals, the development of an accountable optimisation system, was not completed. Doubt cast upon the validity of the building block hypothesis by the previous research task indicated that they could not be used to provide an account for the solutions upon which the GA converges. This facility was fundamental to the development of the accountable optimisation technique and it was therefore necessary to abandon this phase of the research.

The final goal of the original plan of research was to use high utility building blocks for the visualisation of the functions to which the GA was applied. Although this was not possible, the development of a visualisation technique for high dimensional functions remained a major research goal. This goal was met by the variable separation visualisation technique which compared favourably with the statistical technique of PPR (used as a visualisation technique).

References

Aarts, E., & Korst, J. (1989). *Simulated annealing and Boltzmann machines : A stochastic approach to combinatorial optimization and neural computaing.* John Wiley & Sons Ltd.

Beauchamp, K.G. (1975). *Walsh functions and their applications.* London : Academic Press.

Belew, R.K. & Booker, L.b. (Eds.) (1991). *Proceedings of the Fourth International Conference on Genetic Algorithms.* San Mateo : Morgan Kaufmann Publishers.

Bethke, A.D. (1981). *Genetic Algorithms as function optimizers.* Doctoral Dissertation, University of Michigan.

Booker, L.B. (1982). *Intelligent behaviour as an adaptation to the taks environment.* Doctoral Dissertation, University of Michigan.

Cavicchio, D.J. (1970). *Adaptive search using simulated evolution.* Unpublished doctoral dissertation, University of Michigan.

Cheng, B. & Titterington, D. (1994). Neural Networks: A Review from a Statistical Perspective. *Statistical Science.* 2-30, V9, No 1 (1994). The Institute of Mathematical Statistics.

Chernoff, H. (1973). The Use of Faces to Represent Points in k-Dimensional Space Graphically. *Journal of the American Statistical Association.* 361-368, 68 (June 1973). The American Statistical Association.

Dasgupta, D., & McGregor, D.G. (1991). *A structured genetic algorithm : The model and first results.* Research report, University of Strathclyde.

Davis, L. (Ed.) (1987). *Genetic algorithms and simulated annealing.* Research Notes in Artificial Intelligence. London : Pitman.

Davis, L. (1991). *Handbook of genetic algorithms.* New York : Van Nostrand Reinhold.

Davis, L. (1991b). Bit Climbing, Representational Bias, and Test Suite Design. *Proceedings of the Fourth International Conference on Genetic Algorithms.* 18-23,

Davis, T.E. & Principe J.C. (1993). A Markov Chain Framework for the Simple Genetic Algorithm. *Evolutionary Computation.* 269-288, V3, No 1 (1993). Massachusetts Institute of Technology.

De Jong, K. (1975). *An analysis of the behaviour of a class of genetic adaptive systems.* Doctoral Dissertation, University of Michigan.

Drebin, R., Carpenter, L. & Hanrahan, P. (1988). Volume Rendering. *ACM SIGGRAPH '88 Proceedings.* 65-74, 22(4).

Earnshaw, R. & Wiseman,. N. (1992). *An Introductory Guide to Scientific Visualization.* Berlin : Springer-Verlag.

Forrest, S. & Mitchell, M. (1992). Relative Building-Block Fitness and the Building-Block Hypothesis. *Foundations of Genetic Algorithms 2.* 109-126.

Friedhoff, R.M. & Benzon, W. (1989). *Visualization.* New York : Harry N. Abrams Inc.

Friedman, J.H. & Tukey, J.W. (1974). A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computers.* 881-890, V c-23, No 9 (1974). IEEE Computer Society.

Gill, P.E., et al. (1981). *Practical optimization.* London : Academic Press.

Goldberg, D.E. (1983). *Computer-aided gas pipeline operation using genetic algorithms and rule learning.* Doctoral Dissertation, University of Michigan.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization & machine learning.* Reading, MA : Addison-Wesley.

Grefenstette, J.J. (Ed.) (1985). *Proceedings of an International Conference on Genetic Algorithms and Their Applications.* Hillsdale, NJ : Lawrence Erlbaum Associates.

Grefenstette, J.J. (Ed.) (1987). *Genetic algorithms and their applications : Proceedings of the Second International Conference on Genetic Algorithms.* Hillside, NJ : Lawrence Erlbaum Associates.

Grefenstette, J.J. (1991). Strategy acquisition with genetic algorithms. *Handbook of Genetic Algorithms.* 186-201.

Hamming, R.W. (1950). Error Detecting and Error Correction Codes. *Bell System Technical Journal.* 147-160, v29(2).

Harris, R.J. (1975). *A Primer of Multivariate Statistics.* London : Academic Press.

Hastie, T.J. & Tibshirani, R.J. (1990). *Generalized Additive Models.* Monographs on Statistics and Applied Probability 43. London : Chapman & Hall.

Holland, J.H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor : University of Michigan Press.

Hollstein, R.B. (1971). *Artificial genetic adaptation in computer and control systems.* Doctoral dissertation, University of Michigan.

Huber, P.J. (1984). Projection Pursuit. *The Annals of Statistics*. 435-475, V13, No 2 (1985). The Institute of Mathematical Statistics.

Jarvis, R.A. & Patrick, A. (1973). Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*. 1025-1034, V C-22, No 11 (1973). Institute of Electrical and Electronics Engineers.

Jenkins, W.M. (1991). Structural optimisation with the genetic algorithm. *The structural engineer*. 418-422, V69, No 24 (1991)

Kirkpatrick, S. et al (1983). Optimization by Simulated Annealing. *Science*. 671-680, V220, No 4598.

Klockgether, J. & Schwefel, H.P. (1970). Two-phase nozzle and hollow core jet experiments. *Proceedings of the 11th symposium on engineering aspects of magnetohydrodynamics*. 141-148. California Institute of Technology.

Michalewicz, A. (1993). A heirarchy of evolution programs : An experimental study. *Evolutionary Computation*. 51-76, V1, No 1 (1993). MIT Press Journals.

Muller, B. & Reinhardt, J. (1990). *Neural Networks: An Introduction*. Berlin : Springer.

Pars, L.A. (1962). *An Introduction to the Calculus of Variations*. London : Heinemann Educational Books.

Parmee, I. (1990). *Pneumatic hydropower devices*. PhD Thesis, Polytechnic Southwest, UK.

Peck C.C. & Dhawan, A.P. (1995). Genetic Algorithms as Global Random Search Methods: An Alternative Perspective. *Evolutionary Computation.* 39-80, V3, No 1 (1995). Massachusetts Institute of Technology.

Pinebrook, W.M., & Dalton, C. (1983). Drag minimisation on a body of revolution through evolution. *Computer methods in applied mechanics and engineering.* 179-197, V39 (1983). Elsevier Science Publishers B.V. (North-Holland).

Rawlins, G. (Ed.) (1991). *Foundations of Genetic Algorithms.* San Mateo : Morgan Kaufmann Publishers.

Rechenberg, I. (1965). *Cybernetic solution path of an experimental problem.* Royal Aircraft Establishment, library translation 1122, Hants, UK. : Farnborough.

Ripley, B.D. (1994). Neural Networks and Related Methods for Classification. *Journal of the Royal Statistical Society (Series B).* 409-437, V56, No 3 (1994). The Royal Statistical Society.

Schaffer, J.D.. (Ed.) (1989). *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications.* San Mateo, CA : Morgan Kaufmann.

Schwefel, H.P. (1989). *Proceedings of the International Conference on Computer-Aided Optimum Design of Structures.* 218-219, 1989, ed. Brebbia, C.A., & Hernandez, S.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms.* 2-9.

Tufte, E.R. (1983). *The Visual Display of Quantitative Information.* Connecticut : Graphic Press.

Vose, M.D. & Wright, A.H. (1994). Simple Genetic Algorithms with Linear Fitness. *Evolutionary Computation*. 347-368, V2, No 4 (1994). Massachusetts Institute of Technology.

Wade, G., Roberts, A. & Williams, G. (1994). Multiplier-less FIR filter design using a Genetic Algorithm. *IEE Proceedings - Visual Image Signal Processing,* V141, No 3 (June 1994).

Whitley, L.D. (1989). The Genitor algorithm and selection pressure : Why rank-based allocation of reproductive trials is best. *Proceedings of the Third International Conference on Genetic Algorithms*. 116-121.

Whitley, L.D. (Ed.) (1992). *Foundations of Genetic Algorithms 2.* San Mateo : Morgan Kaufmann Publishers.

Wolff, R.S. & Yaeger, L. (1993). *Visualization of Natural Phenomena*. New York: Springer-Verlag.