Genetic Neural Networks on MIMD Computers

Nicholas J. Radcliffe

Doctor of Philosophy
Theoretical Physics
University of Edinburgh
1990

Contents

Acknowledgements

Some of this work has been supported by the Science and Engineering Research Council, and some by the Edinburgh Parallel Computing Centre.

It is a pleasure to thank my original supervisors, Ken Bowler and Richard Kenway, for accepting my caprices and my desertion from their subject in good spirits, and David Wallace for allowing me to follow the highly individual route which led me from Lattice QCD to Genetic Algorithms and Neural Networks, for his ever-positive attitude and for providing a means for my continued existence once SERC's money ran out. I am also indebted to Richard Rohwer for useful conversations and moral support.

I would like to thank Gareth Richards and Frank Śmieja especially for entertaining my foolish questions about neural networks, and for many a good pint and dram; Greg Wilson for his energy and blues; Nick Stroud for encouragement; Cathy Quinn for all those sticky situations retrieved; Wullie for the mail, the resistance and the humour; Brian Pendleton for pointing me towards TEX(even if it was via LaTEX!) and for answering (and asking) many a trivial question; Dominic Prior, for being the only other person I know interested in the obscure corners of life which fascinate me; Adam Bennett for putting it all in perspective (you were right); the staff of Seeds, for the finest café in the entire universe; Helen Pankhurst for the friendship, support, meals, films and fun; Tony Clayton for beer (more beer) and inspiration (*tomorrow* we'll save the world); my parents for moral and more tangible support; and Karen, for the delays, the inspiration, and giving it all a purpose.

Abstract

This work draws together two natural metaphors: "genetic algorithms" draw inspiration from natural evolution to attempt to provide a robust, efficient search technique, and "neural networks" form a crude model of information processing in brains which offer the prospect of training computers to solve tasks by example. It is very natural to think of applying the adaptive techniques of genetic algorithms to the problem of searching the space of neural networks, but doing so is extremely hard, and provides the motivation for this work.

It is argued that the key determinant of the success of any particular genetic algorithm is the interaction between the underlying correlations in the search space, the representation of the space adopted, and the genetic operators used to manipulate the representatives of elements in the search space. It is further argued that genetic algorithms as usually formulated are not ideally suited to "training" neural networks, and that in order to make significant progress in this area a broadening of the standard "schema" analysis is required. Such a generalisation, based on the notion of imposing suitable nested sets of equivalence relations over arbitrary search spaces, is proposed and developed. "Design principles" to help construct genetic algorithms for arbitrary problems are suggested in the context of such knowledge of the regularities in the search space as are known. The techniques developed are applied to a number of problem domains and yield some new insights.

Issues of linkage and convergence are also relevant to the application of adaptive genetic techniques to neural network problems. Studies of these are presented. Existing attempts to apply genetic algorithms are also reviewed in the light of the non-standard analysis developed, and the prospects for further progress are discussed.

In recognition of the fact that much of this work was carried out on a large, medium-grained, reconfigurable parallel computer, a study of connection strategies for such machines is also presented.

Notational Conventions

A comprehensive summary of the notation used is given in the appendix, together with notes on equivalence relations, which play a rather major rôle in this work. A few points, however, deserve early comment.

Weights

For obscure reasons, most workers in the field of neural networks write the weight from node i to node j as w_{ji} rather than the more natural w_{ij}. The latter convention is used here, so that products which are conventionally written as matrix-vector multiplications become vector-matrix products.

Reals and Integers

The usual notation $\mathbf{R}^+$ and $\mathbf{Z}^+$ for the positive reals and integers is extended to $\mathbf{R}_0^+$ and $\mathbf{Z}_0^+$ to indicate $\mathbf{R}^+ \cup \{0\}$ and $\mathbf{Z}^+ \cup \{0\}$ respectively.

The integers modulo n are conventionally denoted $\mathbf{Z}_n$. This convention is adhered to, but whereas the representation of $\mathbf{Z}_n$ usually used is $\{0, 1, \ldots, n-1\}$, it will be more convenient to represent the zero element by n. This is not a serious abuse of notation since $n\ (mod\ n) = 0$.

Definitions

The notation $a \triangleq b$ is used generically to mean 'a is defined to be b'.

1 Overview

The work in this study draws together two principal themes—the use of neural networks to perform mappings and the use of genetic algorithms for search. The ultimate goal is the construction of useful training algorithms for neural networks, harnessing the power of genetic algorithms to search the space of networks. In practice, attention is restricted to a familiar sub-class of neural networks which has received much study, so-called 'layered, feed-forward networks'. These have the advantage that a reference training algorithm already exists, in the form of back-propagation, so that it is possible to gauge the success of alternative genetic training algorithms.

There are various motivations for attempting to apply the adaptive techniques of genetic algorithms to network training, and various areas which seem ripe for such application. Perhaps the most obvious, and certainly the most widely-studied, is the problem of choosing a suitable network topology. No principled method for this is currently known, and workers in the field normally restrict themselves to a tiny region of the space of possible network configurations. Searching the full space is extremely difficult, and little progress would be expected with conventional techniques, even where they are capable of being applied.

Additionally, the "standard" training algorithm, back-propagation, while being significantly more robust than any competitor developed to date, is not fast, uniformly successful or even especially accurate. Moreover, it is limited to a very restricted subset of potentially interesting networks, namely feed-forward networks with differentiable activation functions. Training is an optimisation (or search) problem, and in many ways one for which adaptive genetic approaches might be thought particularly appropriate, either because of loose biological analogies or because of the known properties of genetic search. Moreover, a genetic technique offers the possibility of adapting the topology, activation functions and connection strengths simultaneously—a highly attractive prospect. The problem of setting parameters for general training algorithms is also amenable to genetic search.

Conversely, neural networks form an interesting problem domain to tackle with genetic algorithms because they exhibit many of the features which characterise hard search problems, and in particular many of the difficulties which most urgently require further work for the useful development of adaptive search techniques. These difficulties include representation problems, redundancy, epistasis, noise, high cost of evaluation and deception. Successful techniques for tackling any of these in the domain of neural network training are likely to carry over to more general contexts.

1 Organisation

Chapter two gives a fairly general introduction to neural networks. The presentation is not entirely conventional but is appropriate for drawing out some of the subtleties which will be salient in tackling the problem of their training by genetic algorithms. The discussion begins broadly, stressing the generality of the training problem, but is later restricted to the feed-forward networks that form the focus of the bulk of this study.

Chapter three discusses genetic algorithms as usually formulated. As with the previous chapter, the discussion is not entirely standard, but again brings out rather naturally some points which will be relevant later.

Having described the techniques separately, chapter four begins the discussion of the bringing together of genetic and neural techniques, and consists of a fairly extensive survey and critique of work in the field to date. The discussion is up-to-date, and was written after the research in some of the later chapters was carried out, but provides some insights into the reasons for the research taking the directions that were followed.

In chapter five, intrinsic parallelism and schemata—the key ideas underpinning our understanding of genetic algorithms—are examined, and it is found to be possible to extend the standard analysis to cover general representations, equivalence relations and genetic operators. This has important implications for the application of genetic techniques to the problem of training networks, offering the hope of working in more natural representations and perhaps "folding out" some of the redundancy which will be found to plague these powerful training techniques when applied carelessly.

Chapter six comprises a series of studies around the themes of linkage, bias and convergence. The term linkage refers to the tendency of some genes to be passed on together under genetic reproduction, and might be expected to be important when applying genetic techniques to neural networks, but the work in this chapter suggests that the mechanisms usually proposed for utilising linkage are too weak to be of very much use. Bias in crossover operators is also examined, with perhaps surprising results, and the familiar problem of premature convergence is discussed.

Bringing together the threads of the earlier ones, chapter seven discusses various attempts to apply genetic algorithms to neural networks, both those of the author and of others. It is mainly discursive in nature, commenting upon and analysing further existing work and discussing possible ways to tackle some of the difficulties involved in genetic training, based on the experience gained in the course of the earlier work.

In recognition of the fact that much of this work was carried out on a medium-grained MIMD computer, the so-called Edinburgh Concurrent Supercomputer, the final chapter addresses the key question of how best to connect together the different processors in a reconfigurable MIMD machine. While this depends critically on the particular task performed, a variety of measures can be constructed which give information about the strengths and weaknesses of different topologies. The rather surprising conclusion is that over many such performance measures *all* of the configurations in standard use emerge spectacularly poorly. Evidence is presented which suggests that for many applications highly irregular connection graphs perform very much better. A variety of search techniques are examined in this context, including a genetic algorithm, and there is some discussion of general permutation problems and genetic routes to their solution.

The appendix contains a comprehensive guide to the notation used and includes a brief description of equivalence relations and the logical conventions adopted here. Strenuous efforts have been made to adopt a consistent notation throughout the entire work, though a few symbols are used in different sections for different purposes; the author will have failed badly if this ever causes confusion.

2 Chronological Note

The work presented here consists of a series of disparate studies around two or three unifying themes—principally genetic algorithms and neural networks. In trying to "linearise" these studies into a coherent stream there are various approaches that could be taken. An obvious strategy is to present the work in the order in which it was undertaken, which has the advantage that the progression of ideas can be seen, and the way in which earlier experiences focus the research is brought out. The trouble with this approach, however, is that it is hard to provide a commentary on the work in the light of later research which puts it into better perspective. Moreover, although the path followed at the time seemed reasonably obvious, with hind-sight it looks less so even to the author, and the reader might fail to observe it altogether.

For these reasons the approach taken here is instead to try to find a logical order in which to present the material, giving the freedom to comment on earlier research in the light of later work and allowing greater coherence. It so happens that in this case the process of constructing a clear path has involved almost exactly reversing the order in which the research was undertaken. It might be helpful, therefore, to explain the chronological connections between the various pieces of work. For this description only, the first person will be adopted since the thought processes involved are by their nature personal to the author.

My first interest in neural networks grew out of a conversation in a pub with Gareth Richards, who finally gave me a description of them that appealed to me. This description forms the basis of the presentation in chapter 2. I already had some interest in genetic algorithms and had been playing around, along with Greg Wilson, with very simple sorts of reproductive models *à la* Dawkins (1988), and it seemed natural to try to apply genetic search to the problem of training neural networks. It was immediately apparent, however, that the "hidden node problem" (which allows arbitrary permutations of be applied to the hidden node labels without affecting the resulting network) would present a very serious obstacle to progress. At this stage I played around with a few "toy" gannets—"gannet" being the term I, among others, coined for the combination of genetic algorithms and neural networks—but without great success. Some of this work is discussed in chapter 7. I was also interested, at this stage, in comparing the grand formalism of

Holland's genetic algorithms with the much simpler evolutionary approaches taken by Dawkins, work which lead to the study of clustering around optima given in chapter 6.

Meanwhile, Mike Norman had been attempting to use a genetic algorithm to tackle the "transputer configuration problem"—that of finding a good way of connecting together transputers in a Meiko Computing Surface. I became involved with this, and the study grew into the joint collaboration between Dominic Prior, Lyndon Clarke, Mike and me which forms the bulk of chapter 8.

While this was going on I continued to think about the hidden node problem, and became increasingly convinced that the formalism of genetic algorithms was inadequate to allow proper application to neural networks. This was reinforced by the experience of attempting to use a genetic algorithm for the transputer configuration problem. Moreover, I had a growing conviction that the key to extending the analysis lay in the use of general equivalence relations, but could not quite see how to turn this intuition into respectable scientific conclusions.

Another area which particularly interested me was that of linkage: I could not understand why this important topic seemed to be so consistently ignored by the genetic algorithms community, since there seemed to be no conclusive research to show that linkage was not useful. Moreover, I had a strong intuition that linkage was going to be important in studying neural networks with genetic algorithms, because epistasis—the phenomenon whereby groups of genes act in highly non-linear ways in combination—seemed certain to loom large, and the prospect of allowing groups of co-adapted genes to be preferentially transmitted was highly attractive. This led to the other studies in chapter 6.

The last piece of work to be performed was in fact that presented in chapter 5, which in every sense forms the centre of this study: this work pulls together all of the ideas I had been kicking around over a period of two or more years, and brings a semblance of coherence to an otherwise rather idiosyncratic collection of ideas. This explains why the work is not as fully developed as I would have liked, and why I have not applied the ideas contained within it more widely in the rest of the studies. If I had known then, what I believe I know now, the body of this work would look very different.

Hofstader's Law:
It always takes long than you expect,
even when you take into account Hofstadter's Law.
— DOUGLAS HOFSTADTER, Gödel, Escher, Bach: An Eternal Golden Braid (1979)

2 Neural Networks

The field of neural networks is a very broad one, drawing researchers from fields as diverse as physics and neurology, engineering and psychology, biology and computer science. Workers from these fields often have quite different perceptions of what a neural network is, and radically different reasons for studying them. It will be useful first to describe briefly some of these different view-points, discussing general features of neural networks, and then to formulate more precisely what will be meant by the term in the remainder of this study.

1 Perceptions

The neural networks described herein are called by some 'artificial neural networks' in order to distinguish them from the 'real' neural networks which form brains. Certainly the inspiration for this computational model is drawn from models of real brains, but since the discussion here will only ever consider 'artificial' neural networks the qualification will be dropped.[1]

Perhaps the most obvious interpretation of a neural network is as a software model of certain poorly understood neurophysiological processes thought to exist in brains. Brains are known to consist of some 10^{10} neurons, each of which is connected to perhaps 10^4 other neurons. Each neuron receives stimulae from other neurons to which it is connected, and sometimes from external sources, and responds to these stimulae with a pattern of activity, perhaps the most important feature of which is a firing rate.

A typical sequence of neural events may be thought of as

1° stimulation of some neurons by an external event (for example, the appearance of a particular light pattern on the retina);

2° instability during a cascade of activity as stimulated neurons activate or inhibit activation of other neurons;

3° stabilisation of neural activity.

This provides a crude picture of our understanding of processing in the brain. The approach in neural networks research is essentially to model neurons by *nodes* which implement simple non-linear functions. Each node is has a number of input and output connections, and associated with each connection is a *weight* or *connection strength*. The stimulus or *potential* at a node is usually taken to be the sum of the products of the outputs from all the nodes connected to the node under consideration and the associated

[1] Those who take the 'strong AI' position might take issue with the claim that the distinction between 'real' and 'artificial' neural networks exists at all, but a physics thesis is perhaps not the most suitable place to get embroiled in such metaphysical considerations, however tempting the prospect.

connection strengths. The output from this node is then calculated by applying the non-linear function to the input potential.

It will be seen from this description that there is an implicit notion of time in neural network models. In the brain, activity gradually builds up, firing rates changing as the activity in connected nodes varies in analogue fashion. In general, connections can be either excitatory—in which case activity in one neuron tends to be positively correlated with activity in the other—or inhibitory, in which case the correlation tends to be negative. In network models the activities in nodes are usually updated in an iterative manner until the node states stabilise (or fail to stabilise). This introduces a discrete notion of time.

The model described allows computation to be performed by designating some of the nodes as *input units*, and others as *output units*. A pattern (stimulus) is applied to the network by setting the (output) values of the input nodes. The dynamics of the network then cause activity to spread through the network until, in a useful net, the states of the output nodes stabilise and can be read.

Thus a neural network can be thought of as a unit which performs computation by translating input patterns into output patterns. This, to some extent, is the view-point of computer scientists In this sense, the potential range of activities a network can perform can be seen to be the same as the range of activities a Turing machine can undertake: in each case, it is necessary to be able to state the problem in the form 'produce a specified output from a given input'. From a more mathematical perspective, a neural network can be thought of as a device which implements arbitrary mappings.

Perhaps the most widely differing perceptions come from physicists and neurologists. The latter group tend to be interested in building complex models of the brain: for them, the more realistic are the building blocks from which networks are constructed, and the more neural features which they can combine, the greater is the interest, and the greater the validity of testing models and hypotheses using them. For physicists, almost the reverse is true, for physics is fundamentally about reducing systems to their bare essentials. If models with very simple connectivities and activation functions can produce interesting behaviour, especially if that behaviour is relatively unaffected by the details of the particular functions and connectivities used, then so much the better. For the physicist simplicity is the essence, and the more detailed the knowledge of human brains necessary for the construction of models which exhibit interesting behaviour, the less satisfactory are they.

From this point of view, one of the great interests in neural networks is their robustness to damage. In a typical network, information is not localised to anything like the degree that is typical in conventional computational models: one of the aims is to avoid so-called 'grandmother cells' (Śmieja (1989))—single neurons which fire in response to one pattern only, causing recognition of that pattern. In a robust network the information is distributed, at least across the weights, but perhaps also through the variety of activation functions used and so forth. This leads to the possibility of graceful degradation of performance as a parts of a network are perturbed or damaged. This property is one of the attractions of the subject for engineers, for it allows the possibility of realising neural networks in silicon and being able to use many chips produced which contain faults—still a significant problem for chip manufacturers.

2 Training

In order for it to be useful and interesting, it is necessary to have some control over the mapping that a neural network performs. This is determined in combination by the topology of the network, its dynamics, the functions implemented by the nodes and the strengths of the connections between the nodes. In some cases these determinants can be prescribed in such a way that the resulting network is guaranteed to perform a given mapping. Various constructive algorithms exist which build such networks (Frean (1989), Mezard & Nadal (1989)), but this is not, in general, the path followed.

Perhaps the canonical use for a neural network is to perform partially-specified mappings. A network is "shown"[2] a number of representative input-output (or "pattern-target") pairs and the goal is to "train" the network to map the sample inputs to their corresponding outputs. An algorithm for altering the network is employed which tends to cause the network more nearly to implement the desired mapping as these training patterns are repeatedly presented. This process of training takes its inspiration from the way people are commonly taught—by example.

The phenomenon whereby networks are able not only to map training data, but also in some sense to interpolate the map for unseen data, is known as generalisation, and provides one of the main reasons for interest in the subject. A trivial example would be a network which classified digitised images as dogs or cats. The hope would be that after training the network on a representative set of images it would be able to classify correctly pictures of dogs and cats which were not in the training set. More powerfully, this way of working offers the possibility of implementing maps which are not fully known, to perform transformations for which no algorithm is known.

The normal requirement that neural networks be capable of generalisation adds interest and complexity to the task of training. Without it, given suitable constraints on the network parameters and an objective function returning a measure of the performance of the network, training becomes a well-defined optimisation problem, albeit a hard one. When generalisation is required, the task becomes ill-defined and still more problematical.

One of the obstructions faced when attempting to produce a network which generalises is the phenomenon of *over-learning*. Here, the network learns to map the training data accurately, but loses the ability to

[2] anthropomorphisms are hard to avoid in this subject, and will recur . . .

interpolate usefully. A common interpretation of this is that the network learns the individual characteristics of the training patterns, and chooses to map them by recognising these characteristics rather than—as is intended—by extracting general features from the patterns and classifying them on the basis of these. There is a fine balance to be struck between the desire to extract as much useful information as possible from the training data and wishing to avoid over-learning.

3 A More Formal Description

It will be convenient to take a neural network $\mathcal{N}$ to be a system comprising:

1° A set N of *nodes*,[3] together with a decomposition function

$$d : N \longrightarrow \mathsf{P}(N)^2,$$

which designates some of the nodes as *input nodes* I and some as *output nodes* O. There is no requirement that these be disjoint sets. Given this decomposition,

$$N = I \cup H \cup O$$

where the *hidden nodes*, H, are defined to be those nodes which are neither input nor output nodes. It is conventional to think of a node as a simple processing device characterised by a *state*—its output state—which is influenced by, and itself influences, the state of other nodes in the system. This is the model that will be assumed in the text here. In the mathematical presentation, however, the set of nodes is simply an index set.

In general, an *input pattern* is presented to a network by specifying initial states for the input nodes. A specified system of dynamics then updates the states of the other nodes in the system, and possibly of the input nodes also, until at some later time the states of the output nodes have stabilised and an *output pattern* may be read from these. In this way, a network associates one output pattern with each input pattern.

If the node states are real, which is sufficiently general for present purposes, then a network $\mathcal{N}$ effectively acts as a function

$$\mathcal{N} : \mathbf{R}^{|I|} \longrightarrow \mathbf{R}^{|O|}$$

where the notation $|A|$ is used to indicate the number of elements in a set A.

2° A *connectivity*. The connectivity of a network can be viewed as a function

$$c : N \times N \longrightarrow \{0, 1\}$$

[3] Nodes are also sometimes called *neurons* or *units*.

which is defined by

$$c_{ij} \triangleq \begin{cases} 1, & \text{if node } i \text{ is connected to node } j, \\ 0, & \text{otherwise.} \end{cases}$$

Connections are not necessarily symmetric ($c_{ij} \not\equiv c_{ji}$) and self-connections are allowed but not required. (A connection from node i to node j allows node i to influence node j but not the reverse, which is why connections may be asymmetric.)

3° A set W of *weights* or *connection strengths*. There is one weight associated with each (ordered) pair of connected nodes and the weights are assumed to be real-valued so that

$$\forall i \in N \; \forall j \in N : \; (c_{ij} = 1 \implies \exists \; w_{ij} \in \mathbf{R})$$
$$W \triangleq \{ w_{ij} \}.$$

4° A set V of $|N|$ *activation functions*. Given the state of every node and weight in the network the ith activation function is used to update the state of the ith node:

$$V \triangleq \{ v_i : \mathbf{R}^{|N|} \times \mathbf{R}^{|W|} \longrightarrow \mathbf{R} \mid i \in N \}.$$

Thus a neural network is conveniently represented as a 5-tuple $\mathcal{N} = (N, d, c, W, V)$—a collection of nodes with some specified decomposition and connectivity, a weight associated with each connection and an activation function for each node. The set of all such networks will be denoted Ω.

It is convenient at this point to make a few more general definitions for neural networks which will be used later in defining objective functions for the genetic algorithm. The following definitions all assume a network $\mathcal{N} \in \Omega$ with nodes $N = I \cup H \cup O$.

A *training set* $\mathcal{T}$, is simply a set of training patterns. A training pattern is a pattern-target pair as described before, $(p, t) \in \mathbf{R}^{|I|} \times \mathbf{R}^{|O|}$. Given such a training pattern, it is useful to define the network's error, ε, for that training pattern. An appropriate definition for

$$\varepsilon : \mathbf{R}^{|O|} \times \mathbf{R}^{|O|} \longrightarrow \mathbf{R}_0^+$$

is

$$\varepsilon(t, v) \triangleq \sum_{j \in O} (t_j - v_j)^2 .$$

Finally, the *global error*, $\mathcal{E}$, is defined as the sum of the errors for each pattern in a training set. For a training set with n pattern-target pairs this gives

$$\mathcal{E} \triangleq \sum_{r=1}^{n} \varepsilon(t^{(r)}, v^{(r)}) = \sum_{r=1}^{n} \sum_{j \in O} \left(t_j^{(r)} - v_j^{(r)} \right)^2 .$$

4 Hopfield Network

One of the most widely studied categories of neural network models is the *Hopfield network.* Using the notation introduced above, an n-node Hopfield net can be defined as follows:

Nodes:	$I = O = N = \mathbf{Z}_n; \quad H = \emptyset$.
Connectivity:	$\forall i \in N \ \forall j \in N : \ c_{ij} = 1 - \delta_{ij} = \begin{cases} 1, & \text{if } i \neq j, \\ 0, & \text{otherwise.} \end{cases}$
Starting Weights:	$w_{ij} = \sum_r t_i^{(r)} t_j^{(r)}$ (Hebb Rule)
Activation:	$\phi : N \times \mathbf{R}^{\lvert W \rvert} \times \mathbf{R}^{\lvert W \rvert} \longrightarrow \mathbf{R}$
	$\phi_j^{(t)} = \sum_{i \in N} v_i^{(t-1)} w_{ij}$
	$v_j^{(t)} = \begin{cases} 1, & \text{if } \phi_j^{(t)} + \tau_j > 0, \\ -1, & \text{otherwise.} \end{cases}$

From this it can be seen that:

- the node states are ± 1.
- The Hopfield net has no hidden units: every node acts as both input and output. A common use for such a network is the restoration of noisy images. The image with noise is presented to the (trained) network and the net iterates to a stable point which, if the image is close to one of those in the training set, should be the cleaned-up image. In this way the Hopfield network acts as an associative memory.
- When using the Hopfield net, there is a standard prescription (the *Hebb rule*) for initial weight values. This typically does not allow all training pattern to be recalled correctly, and training rules such as the Hopfield-Little algorithm can then be employed to improve performance.
- The activation function v used for each node is conveniently decomposed into a potential ϕ, and a function of one variable applied to that potential. This is common to many types of network. The function applied is in each case a sigmoid, distinguished by a bias which varies between nodes. Having computed the sigmoid, the sign of the result is then taken to yield the node state of ± 1. The bias τ_j is usually taken to be zero.
- The time sequence is important, and is indicated by the parenthetical superscripts. The initial node states $v_j^{(0)}$, are the input pattern presented. Thereafter, the potentials $\phi_j^{(1)}$ are computed and each output state $v_j^{(1)}$ is subsequently modified. This process is repeated

until the node states stabilise, which typically requires very few time-steps. It is possible for 2-cycles to develop so that node states never stabilise, but this is rarely a problem.

The Hopfield network will not feature heavily in this study, but has been presented to show the flexibility of the scheme outlined and the relevance of the training schemes considered to networks of types other than the feed-forward nets which form the focus of the work.

5 Feed-Forward Networks

Layered, feed-forward networks will form the bulk of this study. In these networks nodes are partitioned into disjoint *layers*.

This is shown schematically in figure 1.

Patterns are presented at the input layer (layer 1). The states of these nodes are then used to set the nodes in layer 2, the first hidden layer, and each successive layer is set in similar fashion until finally the nodes in the output layer—layer n in an n-layer net—are set. The output state may then be read off, the lack of any feed-back precluding the need for a period of stabilisation.

It is useful to introduce a function L assigning a layer to each node in an n-layer network:

$$L : N \longrightarrow \mathbf{Z}_n.$$

The restriction on the connectivity is then

$$c_{ij} = 1 \quad \Longrightarrow \quad L_i = L_j - 1.$$

Defining further the *feed* F_j of a node j to be the set of all nodes which can influence it

$$F_j \triangleq \{\, i \in N \mid c_{ij} = 1 \,\}$$

the potential

$$\phi : N \times \mathbf{R}^{|W|} \times \mathbf{R}^{|W|} \longrightarrow \mathbf{R}$$

is given by

$$\phi_j^{(t)} = \sum_{i \in F_j} v_i^{(t-1)} w_{ij}.$$

In principle, any function of this potential may then be used, but in practice a sigmoid is usually employed:

$$v_j^{(t)} = \frac{1}{1 - \exp -\beta(\phi_j^{(t)} + \tau_j)}.$$

Again, this is controlled by a single parameter, the bias at the node. Weight values are unrestricted. The set of layered, feed-forward networks will be denoted Ω_L, and the set of feed-forward networks (not necessarily layered) will be denoted Ω_F.

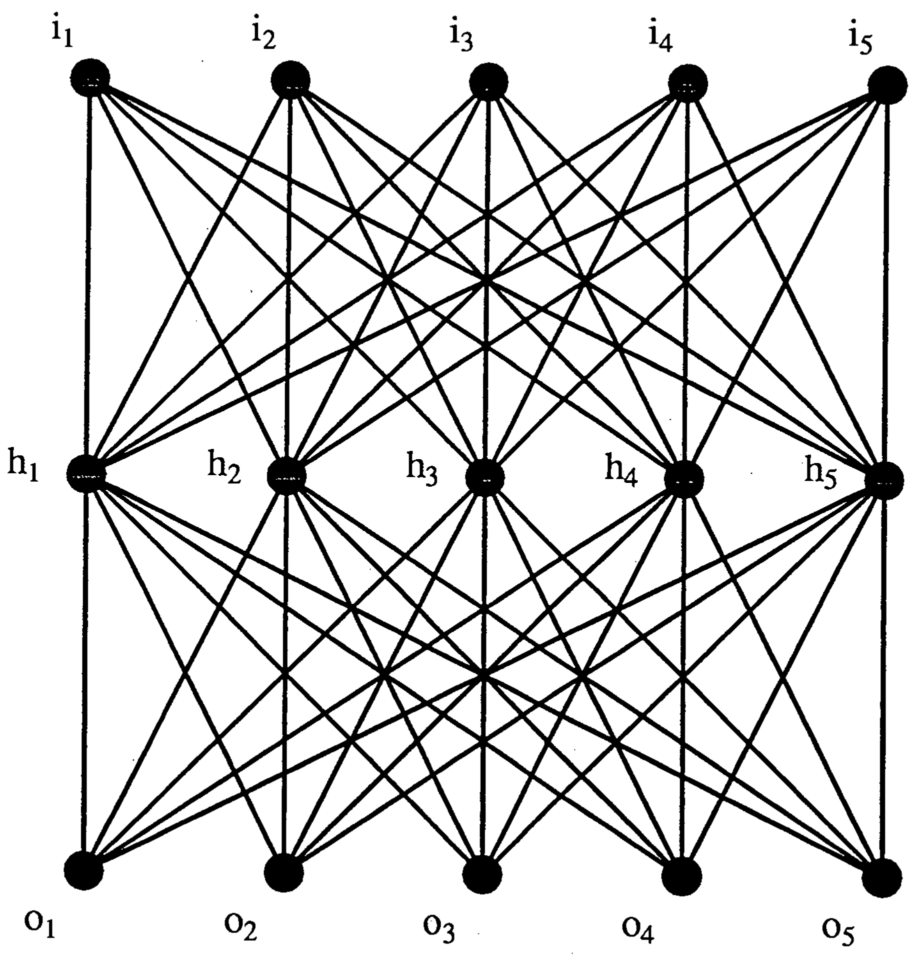


Figure 1: A fully-connected 3-layer, feed-forward neural network

6 Back-Propagation

The best-established method for training feed-forward layered networks uses the *back-propagation* algorithm of Rumelhart, Hinton & Williams (1986). Numerous variations of and acceleration schemes for the technique have been investigated, but the following features of back-propagation and its variants are largely universal:

1° The topology of the network is fixed, that is to say the algorithm does not add, delete or re-label either nodes or connections.

2° The basic weight update scheme is

$$\Delta w_{ij} = -h^{(w)} \frac{\partial \mathcal{E}}{\partial w_{ij}}$$

where E is the error function (omitting functional dependence), and $h^{(w)}$ is the step size. The step size may or may not be the same for each weight, and may or may not vary during the course of the algorithm.

3° Usually, each node in the network has the same activation function, distinguished only by its bias τ. The bias is varied according to

$$\Delta \tau_j = -h^{(b)} \frac{\partial \mathcal{E}}{\partial \tau_j}.$$

Again, $h^{(b)}$ is the step size.

Thus the back-propagation algorithm, $\mathfrak{B}$, and indeed most conventional network training schemes, simplify the 'full' network training problem

$$\Omega \longrightarrow \Omega$$

to

$$\mathfrak{B} : \mathbb{R}^{|W|} \times \mathbb{R}^{|N|} \longrightarrow \mathbb{R}^{|W|} \times \mathbb{R}^{|N|}.$$

7 Momentum and Acceleration Strategies

There are numerous schemes designed to improve the speed and quality of back-propagation. Probably the most widely used involves the addition of a 'momentum' term to the update scheme. With the new terms the equations take the form:

$$\Delta w_{ij}(t) = -h^{(w)} \frac{\partial \mathcal{E}}{\partial w_{ij}(t)} + \alpha \Delta w_{ij}(t-1); \qquad \Delta \tau_j^{(t)} = -h^{(b)} \frac{\partial \mathcal{E}}{\partial w_{ij}(t)} + \alpha \Delta \tau_j(t-1),$$

where the (t) argument now labels the time-step or 'epoch' and α is the fractional momentum. As can be seen, the idea is that some proportion of the previous change is added on to the current change for both weights and biases. When successful, this helps to avoid local optima, and reduces training times. Momentum is very widely used, and though some workers find high values problematical others consistently use values as high as .99. Richards (1988)

Other acceleration schemes abound, and usually focus on adjusting either the momentum or the step-size according to prevailing conditions. One of the more successful schemes is the SAB (Self-Adjusting Back-propagation Scheme) strategy of Tollenaere (1990), in which step sizes are increased by a small amount at each time-step as long as the error continues to decrease. Once this condition is violated, the last step is undone, using information already available for the momentum scheme, and the step-size is gradually reduced again.

8 Assessment of Neural Networks

If the aim of this work is to produce good training algorithms for neural networks, it would be as well to decide what shall be the criteria for judging the effectiveness of an algorithm. There are four main measures which might be considered—robustness, accuracy, speed and generalisation ability.

A robust training algorithm would be one which had wide applicability and which rarely failed to learn on training sets from some suitably chosen class. There are many classes of networks for which no satisfactory training algorithm is known, and there are very few domains for which a reliable algorithm is known to complete in reasonable time, so any improvement in this are would be welcome.

Accuracy is perhaps less of a problem, for when networks learn it is usually the case that acceptably accurate solutions can be found. Moreover, assuming that a network can be found which is close to a good solution with respect to some known training algorithm, the final hill-climbing required to reach that solution can easily be performed by the known technique.

While any speed improvement is always welcome, unless large factors can be gained this is probably not so much of an issue: training networks is always relatively time-consuming, but once the network has been trained it can be used to process patterns at a prodigious rate. Thus for real-time situations the interest is more often in using a trained network, and during training it is usually more important to know that an algorithm is robust than that it will converge quickly, provided always that the times involved are not very long compared with those for other techniques.

The most interesting criterion, as has already been noted, is that of generalisation ability. By its nature, this is something that cannot be controlled directly, but which will often be very important. After all, it will almost always be possible to implement a known function more efficiently with conventional techniques, even if this involves a resort to table lookup, so while models which do not generalise may be of theoretical interest, it seems reasonable to set some store by a network's generalisation capabilities.

This last criterion is the only one in which the comparison between conventional techniques such as back-propagation is at all subtle. For while other approaches are possible, and will be discussed, the natural way to tackle genetic training involves use of the total error $\mathcal{E}$ as an objective function, just as for gradient techniques. The subtlety arises from the fact that this is the *only* information that the genetic algorithm will normally be given about the training patterns and its performance with respect to them. This contrasts with gradient techniques where errors are computed for back-propagation through the system, feeding more information into it about the training data. This point is sometimes used to suggest that the genetic algorithm solves a harder problem than does back-propagation but this point of view seems slightly perverse, since the reason for failing to use the extra information is not that this makes the problem easier (something to be welcomed) but simply that it is not obvious *how* to utilise it in the genetic algorithm, and because an algorithm which fails to use it has wider applicability. The more interesting question is whether using less information about the data during training helps improve the generalisation ability of the network developed by genetic techniques. Of course, until suitable genetic training algorithms are developed, the question remains incapable of resolution, but the hope of better generalisation provides yet another spur to progress.

There was I, trapped.
Trapped like a trap in a trap.
— DOROTHY PARKER, The Waltz.

3 Genetic Algorithms

In 1975 John Holland suggested a novel search technique inspired by adaptation in evolving natural systems (Holland (1975)). He envisaged searching a general space, $\mathcal{S}$, composed of *structures*, which his *reproductive plans* would selectively manipulate and reproduce to yield constantly changing populations of solutions. The manipulation of structures was to be achieved through the application of idealised *genetic operators*, and the aim was to create selection pressures within the population which would drive it towards ever-better regions of $\mathcal{S}$.

One of the key motivations behind the development of Holland's *genetic algorithm* was an attempt to devise an optimal strategy for "allocation of trials" over the search space. Associated with each structure in the population is a measure of its performance, variously called *utility, fitness* or *goodness*. The specification of a utility function is an essential part of the definition of any particular reproductive plan. The utility it yields when a structure is tested (evaluated) is interpreted as a 'payoff', and the aim in developing the algorithm was to maximise expected payoff at each time-step by making optimal use of available information.

The basic problem can be identified as striking the right balance between "exploitation" and "exploration". When the search begins, all parts of the search space are assumed to be equally likely to be rich in good solutions. As the search progresses, however, some parts of the space are seen to exhibit above-average performance, while others are observed to exhibit below-average performance. The art—which in fact can be turned into a science—is to exploit such information as has been gleaned about the search space by concentrating trials in areas which are performing well, while adequately allowing for the possibility that the poor observed results in other areas of the space are due to sampling error. The discussion will return to this theme from time to time.

1 Representation, Schemata and Utility

The structures used in Holland's genetic algorithm are represented by *chromosomes*, a chromosome being an ordered list of values, each drawn from some specified set. The chromosome is conveniently pictured as a linear string. Each position on the chromosome is called a *gene* and the values that any gene can take are referred to as its *alleles*. The search space which the algorithm explores is usually the set of all chromosomes, that is, all possible distributions of alleles over the genes. For example, if a chromosome η has n sites (genes) taking values $(\eta_1, \eta_2, \ldots, \eta_n)$ drawn from sets $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n)$ respectively, then η is a member of the effective search space

$$\mathcal{C} \triangleq \mathcal{G}_1 \times \mathcal{G}_2 \times \cdots \times \mathcal{G}_n.$$

Assuming that the chromosome codes the solution to some problem, as is usually the case, this amounts to an implicit requirement on the representation that all chromosomes represent (valid) solutions to the problem. It is convenient to introduce a representation or coding function

$$\rho : \mathcal{S} \longrightarrow \mathcal{C}.$$

This function must certainly be injective, for different chromosomes must be capable of being decoded unambiguously to yield structures in $\mathcal{S}$. Moreover, it must be surjective, because every chromosome must represent some structure. Thus ρ should be a bijection, which is reassuring, for it means that $\mathcal{C}$ can faithfully represent $\mathcal{S}$.

Having said this, it is not all that uncommon for people to use genetic algorithms in which either several chromosomes represent the same structure in $\mathcal{S}$ (so that ρ is an improper, or "many-valued" function) or in which not all chromosomes correspond to legal solutions. Such representations are to be avoided where at all possible. If structures in $\mathcal{S}$ are represented by more than one chromosome then at very least the space being searched is unnecessarily large, and while some optimisation techniques, such as the analogue neurons of Hopfield & Tank (1984), make use of enlarged continuous spaces to smooth search in difficult discrete spaces it is hard to see such effects operating in the context of genetic algorithms. It is likely to be similarly unhelpful to have constrained representations (that is, ones in which not all chromosomes correspond to structures in $\mathcal{S}$) for this will almost invariably impede the intrinsic parallelism and schema recombination which is about to be introduced.

In order to analyse the behaviour of genetic algorithms it is useful to introduce a set of partitionings on the space of chromosomes. Given a particular partitioning, each partition (subspace) will be characterised by the sharing of some specified genes by its elements. The partitions conventionally used are conveniently expressed as *schemata*, members of the set

$$\Xi \triangleq \mathcal{G}_1^\star \times \mathcal{G}_2^\star \times \cdots \times \mathcal{G}_n^\star,$$
$$\text{where} \qquad \mathcal{G}_i^\star \triangleq \mathcal{G}_i \cup \{\square\}.$$

For example, the schema $\xi = ab\square\square\ldots\square$ specifies the partition containing all chromosomes η which have $\eta_1 = a$ and $\eta_2 = b$. Formally,

$$\eta \in \xi \iff (\forall i \in \mathbf{Z}_n \; (\xi_i \neq \square) : \; \eta_i = \xi_i).$$

Plainly every chromosome is a member—or *instance*—of 2^n schemata. (This can be seen by noting that the replacement of any subset of a chromosome's genes by $\square$ generates a schema which it instantiates, and that there are 2^n such subsets.)

Let the utility function which the genetic algorithm uses to guide its search be u. This associates with each chromosome a positive, real measure of its performance:

$$u : \mathcal{C} \longrightarrow \mathbf{R}^+.$$

It is useful to construct from u a measure

$$\mu : \Xi \longrightarrow \mathbb{R}^+$$

which gives the utility of a schema as the average utility of all its members:

$$\mu(\xi) \triangleq \frac{1}{|\xi|} \sum_{\eta \in \xi} u(\eta),$$

where $|\xi|$ is the number of chromosomes in C which instantiate the schema ξ. Noting that $C \subset \Xi$, it is then immediately apparent that

$$\mu\big|_C = u,$$

so that μ can be used to yield the utility of either a schema or a chromosome.

Holland observed that each evaluation of a chromosome, or structure, could be regarded as a statistical sampling event which yields information about the utility of *each* of the 2^n schemata that the chromosome instantiates—a phenomenon known as intrinsic parallelism. Provided that there are correlations between the utilities of different instances of schemata, this information can usefully be used to guide the further exploration of the space. This, in fact, is the key assumption in constructing a genetic algorithm, and provides the sense in which the genetic algorithm allocates trials in near-optimal fashion.

One of the contentions of this work is that the validity of this assumption (that correlations between the various instances of each schema exist) is a prime factor in determining the effectiveness of genetic search. Great care is needed in selecting suitable reproductive plans for tackling the various problems associated with training neural networks. In particular, the schemata should contain chromosomes corresponding to networks whose performance is likely to be correlated.

2 Crossover and Reproductive Plans

The key genetic operator employed in most reproductive plans is a *crossover* operator. A simple crossover operator X has a functional form:

$$X : C \times C \times \mathbb{Z}_{n-1} \longrightarrow C,$$

that is, given two "parent" chromosomes and a "cross point" it yields a single child chromosome. The component-wise prescription is typically as follows:

$$X_i(\eta, \zeta, r) = \begin{cases} \eta_i, & \text{if } i < r, \\ \zeta_i, & \text{otherwise.} \end{cases}$$

For example, $X(\eta, \zeta, 4)$ may be visualised as:

$$\left.\begin{array}{c} \eta_1\eta_2\eta_3\eta_4\eta_5\cdots\eta_n \\ \zeta_1\zeta_2\zeta_3\zeta_4\zeta_5\cdots\zeta_n \end{array}\right\} \stackrel{X}{\longmapsto} \eta_1\eta_2\eta_3\zeta_4\zeta_5\cdots\zeta_n$$

Crossover is understood to be the driving force behind genetic algorithms, and despite its deceptively simple form is carefully designed. Informally, the way that a typical reproductive plan proceeds as follows. An initial population of chromosomes is generated in some fashion, often randomly, and the utility of each is determined. Thereafter, at each step of the algorithm a pair of chromosomes is selected for crossing over. The process is random, the chromosomes being picked with a probability directly proportional to their utility. Thus chromosomes with higher utility more often participate in reproduction. A random cross point is also selected, each of the intergene positions being given equal probability. Any other genetic operators being used are then applied to the "proto-child", which replaces some chromosome chosen from the population randomly and without bias. Such a reproductive plan will be referred to as a reproductive plan of type $\mathfrak{P}$.

The plan described might be thought of as "fine-grained" (or *asynchronous*, or *steady-state*) in the sense that a single individual is added to the population at each time-step, and a single individual is deleted. It is also possible to adopt a procedure whereby an entire new population is generated from the old in a "coarse-grained" (or *synchronous*, or *generational*) update procedure. Each method has its proponents and their merits are discussed in section 4.

3 Crossover, Schemata and Correlations

A reproductive plan of type $\mathfrak{P}$ would make a plausible heuristic search technique, for it is plain that fitter chromosomes (those with higher utility) more often take part in reproduction than their less fit counterparts, and that children bear some similarity to each of their parents. Thus it seems reasonable that the mean utility of the population might be dragged up by repeatedly recombining different solutions. In fact, as has already been indicated, this algorithm is much more than a heuristic, and can be shown to use the information gained from sampling instances of many schemata in such a way as to maximise expected "payoff" given the information accumulated in the population (Holland (1975)).

An intuitive understanding of the power of the algorithm can be gained by thinking about schemata. The *definition points* of a schema are those loci (sites) not filled by $\square$, so that the definition points of $a\square b\square c$ are 1, 3 and 5. The *order* of a schema, $o(\xi)$, is equal to the number of definition points it has, so that $o(a\square b\square c) = 3$, and the *definition length* of a schema, $\ell(\xi)$, is the maximum distance between any pair of definition points, so that $\ell(a\square b\square c) = 5 - 1 = 4$.

One of the important quantities to be able to calculate is the *disruption rate* for a schema—the probability that the child of a parent which instantiates the schema will not itself be an instance of the schema. If the definition length of the schema is small, it is fairly unlikely to be disrupted by a crossover operation because the cross point is unlikely to fall between the extreme definition points. Similarly, low-order schemata are less likely than their higher-order counterparts to be disrupted. Moreover, instances of schemata with high sample averages for fitness will tend to be involved in many crossover operations, because reproduction rates vary with utility. When these factors combine to produce a child which instantiates a high-performance schema containing one of its parents, but which is not the same as that parent, an area of the search space which has exhibited good performance (as witnessed by the high sample average of the schema) is further explored, while another sample point within that space is measured so that its true performance is more accurately established. Since each chromosome instantiates many schemata, each reproduction involves many of these events simultaneously, and it is from this that intrinsic parallelism derives its power. As the algorithm proceeds, instances of short, low-order schemata with high sample averages accumulate in the population. It then becomes appropriate to think of the algorithm testing these schemata in combination. This assertion forms the basis for the "Building Block" hypothesis of Goldberg (1989), discussed in chapter 5.

4 The Selection Algorithm

The selection algorithm determining which chromosomes in the present population breed, and how often, is a vital part of the definition of a reproductive plan, and its accuracy and character can significantly affect the effectiveness of the search.

As has been indicated, a fixed-size population $\mathfrak{B}(t)$ of chromosomes is maintained at time-step t, and each member of $\mathfrak{B}(t+1)$ is generated from one or more of the members of $\mathfrak{B}(t)$ by the application of the idealised genetic operators (typically crossover and mutation). The probability of picking some $\eta \in \mathfrak{B}(t)$ as the principal parent[4] of any $\zeta \in \mathfrak{B}(t+1)$ is taken to be:

$$P(\eta) = \frac{1}{|\mathfrak{B}(t)|} \frac{\mu(\eta)}{\bar{\mu}(t)} \qquad (1)$$

$$\text{where} \quad \bar{\mu}(t) \triangleq \sum_{\theta \in \mathfrak{B}(t)} \mu(\theta).$$

The conventional way of implementing this scheme is by so-called "roulette-wheel" selection, whereby chromosomes are assigned sectors of a notional wheel subtending angles proportional to their fitness, and the wheel is spun against a fixed pointer to see which chromosome is selected. For coarse-grained update, this procedure was traditionally applied once for each individual in the (new) population.

In an exceptionally elegant piece of work, Baker (1987) has pointed out that this algorithm is needlessly inaccurate, because the actual number of trials (children) allocated to any individual can diverge arbitrarily far from that individual's expected number. This extra source of randomness in the reproductive plan serves no useful purpose and Baker set out to provide an efficient algorithm with minimal spread, without

[4] When a child has two parents it is standard practice to choose them both with this probability, though is not what Holland suggested. There are subtleties associated with this issue, some of which are discussed in section 6.

biasing the selection algorithm, in the sense of giving sampling probabilities which differ from those in equation (1).

He offers several solutions, prehaps the most elegant of which he calls "Stochastic Universal Sampling". This efficiently achieves zero bias with minimal spread by the simple expedient of replacing the single pointer used in conventional roulette wheel selection with a set of N pointers separated by angles of $2\pi/N$, where N is population size. This is the selection algorithm consistently used for this work, and for this reason coarse-grained update is adopted.

Other workers, including Whitley & Hanson (1989b) and Syswerda (1989) advocate fine-grained update for a variety of reasons, none of which seem especially compelling to the author. Syswerda cites the advantages as follows:

> The advantages of [steady-state genetic algorithms] are thought to be that (1) schema fitness versus percentage in the population works out properly as the fixed-point of the system, that (2) good members of the population float to the top of the population where they are protected from deletion . . . , and (3) that poor members sink to the bottom where they are more likely to be deleted (but can be parents if lucky).

Points (2) and (3) are in directly contrary spirit to Baker's careful analysis, seem to have little theoretical basis. His first point is unsupported by references and seems to suggest that schemata should occupy proportions of the population which reflect to their utility. The author sees no reason to require this, and is not convinced that it is in general possible. Neither does it agree with the author's own empirical observations which have never shown any discernible difference between the two procedures when both use roulette-wheel selection. When working on parallel machines with population distributed over many processors, however, the attractions of fine-grained genetic algorithms are greater (for example, Tanese (1987), Mühlenbein (1989), Gorges-Schleuter (1989)).

It should also be mentioned, in passing, that De Jong (1975) used a more general scheme whereby some fixed proportion (the "generation gap") of the population was replaced at each time-step, and these techniques are sometimes applied.

5 Convergence and Fitness Scaling

One of the problems which commonly occurs when using genetic algorithms is that the population suffers "premature convergence", losing diversity before finding the optimum. They way that this often happens is that a solution (chromosome) with very much higher utility than any other member of the population emerges and is allocated a very large number of trials (equation (1)). In these circumstances, its genes can quickly come to dominate the population to the eventual exclusion of all others. If the chromosome does not represent a point close to the optimum, this is undesirable.

The rate at which individuals are allocated trials on the basis of their observed performance is governed by the utility function, but in fact the formalism only really requires that there be some function which maps any population of chromosomes into a probability distribution giving each individuals expected rate of reproduction.

It has been pointed out by Radcliffe (1989) and Grefenstette & Baker (1989) that given any utility function μ over the search space, the composition of an arbitrary (monotone) increasing function

$$f : \mathbf{R}^{+} \longrightarrow \mathbf{R}+$$

with μ also yields a valid utility function, $f \circ \mu$, and indeed one which preserves the relative ranking of solutions. Thus there is scope for slowing the rate of convergence of the algorithm by making a suitable choice of utility function, and numerous schemes exist for "scaling" it to improve performance.

This idea can in fact be taken even further in principle, for there are no constraints at all on the utility function except that global optima must have maximum utility. If a fitness function could be devised for some problem which made the search easier by removing local optima or—thinking now in terms of a minimisation problem—cutting a valley through the space of solutions which led smoothly down to the global optimum, this would be an entirely valid (and useful) approach, though clearly if the search were stopped prematurely the solutions currently held might be very far from optimal. It seems hard to imagine how such a function could ever be constructed in the absence of detailed knowledge of the search space (making search redundant) but a possible way forward is suggested by the work of Levy & Gomez (1985) and Yao & Young (1989), who in the context of a different optimisation scheme called the "Tunnelling Algorithm" proposed adding singularities to the objective function whenever and wherever an optimum was discovered. Though it will not be discussed here, this clearly forms an interesting possible research area.

Baker (1985) earlier suggested another innovative approach to the problem of premature convergence, which was to abandon reproduction in proportion to a fixed utility function altogether, and instead to look only at the rank of the chromosomes when determining their reproductive potential.[5] While the problem of how to relate the number of trials allocated to each rank still remains, this should at least avoid the problem of "super-individuals" taking over the population, and given some "raw" objective function allows the same utility function to be used for all problems. This approach is very attractive, and is used in many of the applications in this work. Some workers, including Whitley & Hanson (1989b) in the context of neural networks, further advocate using rank-based selection to speed rather than to slow convergence, and this will be examined further in chapter 7.

[5] Indeed, it is now standard practice to use a "sliding window" to limit the range of fitness values allowed in the population.

6 The Fundamental Theorem of Genetic Algorithms

The way in which instances of above-average schemata accumulate in the population can be quantified. Given an operator

$$\Gamma_i : \mathcal{C} \longrightarrow \mathcal{C},$$

where any other parameters that the operator may take have been omitted for convenience, let p_i^ξ be the probability that the child of a parent instancing ξ is prevented from instancing the schema itself by the application of Γ_i. Formally,

$$p_i^\xi \triangleq P\left(\Gamma_i(\eta) \notin \xi \mid \eta \in \xi\right). \tag{1}$$

Using this idea, Holland (1975) derived the following remarkable result (pp. 102–103), though not in this form:

THEOREM (The Fundamental Theorem of Genetic Algorithms)

Let $n_\xi(t)$ denote the number of instances of a schema ξ at time-step t in fixed-size population whose dynamics are governed by a reproductive plan of type $\mathfrak{P}$. Let $\hat{\mu}_\xi(t)$ denote the sample utility for the schema—the mean utility of every $\eta \in \xi$ in the population at time t—and $\bar{\mu}(t)$ denote the mean utility of the population at this time-step. Moreover, let there be N_o genetic operators Γ_i, each applied with probability p_i. Then the expected number of instances of the ξ at time-step $t + 1$ is governed by:

$$\left\langle n_\xi(t+1) \right\rangle \geq n_\xi(t) \frac{\hat{\mu}_\xi(t)}{\bar{\mu}(t)} \left[1 - \sum_{i=1}^{N_o} p_i p_i^\xi\right]. \tag{3}$$

It is, in fact, extremely easy both to prove this theorem and to fill in bounds for p_i^ξ for the standard operators. The only subtlety concerns the treatment of recombination operators which introduce extra parents.

Assume initially that the operators are all unary (asexual) so that every child has precisely one parent. Then the term outside the brackets follows directly from selection of the parent on the basis of fitness (equation (1)), and the bracketed term reduces the bound to take account of the fact that each operator, when applied, can destroy membership of the schema. (The second term in the bracket is called the *disruption rate*.)

When treating binary (sexual) operators p_i^ξ must be interpreted as the probability that Γ_i destroys membership of a schema given the probability distribution used to select the other parent.

For example, using the conventional one-point crossover, if both parents are selected according to equation (1) then the probability of disrupting a schema ξ is bounded above by the probability that the cross point falls between the outer-most definition points. To see this, it is sufficient to note that picking both parents in this way results in a doubling of the expected number of offspring from each schema to $2n_\xi(t)\hat{\mu}_\xi(t)/\bar{\mu}(t)$ and that if the cross point falls outside the defining region one of the two possible children is guaranteed to instantiate the given schema. Assuming that the cross point is chosen uniformly along the length, this gives $p_X^\xi = \ell(\xi)/(n-1)$, where the subscript X denotes crossover.

Similarly, the probability of losing at least one definition point as a result of mutation is bounded above by $p_m o(\xi)$, where p_m is the point mutation rate. Substitution in equation (1) restores the familiar form of the Fundamental Theorem:

$$\Big\langle n_\xi(t+1)\Big\rangle \geq n_\xi(t)\frac{\hat{\mu}_\xi(t)}{\bar{\mu}(t)}\left[1 - p_X\frac{\ell(\xi)}{n-1} - p_m o(\xi)\right].$$

Holland, assuming that only one of the parents was chosen on the basis of fitness, showed a slightly different result.

7 Correlations and Optimisation Schemes

A straight-forward method for searching an arbitrary space $\mathcal{S}$ exists which is guaranteed to find the global optimum given a sufficiently long time—enumeration. A strictly enumerative search simply tests every structure in the space in some fixed order, but the key feature of enumeration is that the order in which structures are tested is independent of the observed performance of previously-tested structures, so that random search can also reasonably be classified as enumeration. It is important, however, to realise that for all but the most trivial problems, such techniques are *absolutely* impractical. To quote Holland (1975) (p. 17) (changing notation to coincide with that used here):

> The flaw, and it is a fatal one, asserts itself when we begin to ask, "How long is eventually?" To get some feeling for this we need only look back at the first example. For that very restricted system there were 10^{100} structures in $\mathcal{S}$. In most cases of real interest, the number vastly exceeds this number, and for natural systems like the genetic systems we have already seen that numbers like $2^{10,000} \cong 10^{3,000}$ arise. If 10^{12} structures could be tried every second (the fastest computers proposed to date could not even add at this rate), it would take year to test about $3 \cdot 10^{19}$ structures, or a time vastly exceeding the estimated age of the universe to test 10^{100} structures.

Needless to say, the number of possible states of even a small neural network is similarly enormous.

All search techniques other than enumeration make assumptions about correlations between solutions: if correlations do not exist it is plain that it is impossible to improve on an enumerative search.[6] In the case of genetic algorithms, the implicit assumption is that there is correlation between the utilities of the various instances of a schema. A conventional genetic algorithm is capable, in principle, of recognising any such correlations, but probably[7] few others.

[6] This is the case for the so-called "golf-course" problem, where the objective golf hole is one point on an otherwise level golf green.

[7] The "probably" is made more precise in chapter 5.

It should be clear that correlations of this type need not exist. To emphasise this point, notice that if the size of the search space $\mathcal{S}$ is s, there are $s!$ possible bijective coding functions

$$\rho : \mathcal{S} \longrightarrow \mathcal{C},$$

almost all of which render effectively unusable such correlations as exist in $\mathcal{S}$. Nevertheless, the Fundamental Theorem (equation (3)) will be obeyed for *all* of these representations. The search would be expected to be ineffective simply because almost none of schemata would relate chromosomes with correlated performance.

Of course, choosing a random representation is entirely unreasonable, and there are corresponding operations that would render ineffective any other search technique. The intention here is merely to illustrate that schemata *need* not be meaningful if insufficient attention is paid to the representation, rather than to suggest that there are in fact no correlations in the representations typically used. Moreover, it is not necessary that the instances of *every* schema should have correlated performance in order for a genetic algorithm to be able to make useful progress; merely that there be suitably many schemata which capture correlations in the search space, including some of low order. It should be clear, however, that the greater are the correlations between such instances, and the greater are the number of schemata which collect together structures with correlated performance, the more of the information that the genetic algorithm collects and processes will in fact be useful, and the more effective will its search be.

That the Fundamental Theorem holds in all situations, regardless of the quality of the representation, is both its strength and its weakness. Because the measure of schema fitness used is a sample mean, $\hat{\mu}_\xi(t)$, the reproductive plan is entirely at the mercy of the vaguaries of schemata whose instances are poorly correlated, as can be seen from the fact that it holds even given the kind of wrecking representations described above. In such cases the sample averages for schema utilities are completely unreliable indicators, with the result that the utility will not be dragged up in the way usually expected for a genetic algorithm.

This point is dwelt on partly because it might appear, at first sight, to be in conflict with another remarkable result from Holland (1975) (p. 142), which justifies the lengthy quotation given below. In discussing schemata over fixed k-ary representations, he effectively considers the case of an arbitrary representation. He defines "enriched" schemata to be those with an above average number of high-performance structures (or rather, high-performance representatives under ρ). In the following quotation, some symbols have been changed to agree with the notation used in this work, and Hollands ith "detector" δ_i can be taken to be a function which returns the value of the ith gene in the representation $\rho(s)$ of $s \in \mathcal{S}$, i.e. $\delta_i(s) = \rho_i(s)$.

> Let $\mathcal{S}$ contain x structures which are of interest at time t (because their performance exceeds the average by some specified amount). If the attributes are randomly dis-

tributed over the structures, determination of "enrichment" is a straight-forward combinatoric exercise. More precisely, let each δ_i be a pseudo-random function and let $\mathcal{G}_i = \{0, 1\}$, $i = 1, 2, \ldots, n$, so that a given structure $s \in \mathcal{S}$ has property i (i.e. $\delta_i(s) = 1$) with probability $\frac{1}{2}$. Under this arrangement peculiarities of the payoff function cannot bias concentration of exceptional structures in relation to schemata.

Now, two exceptional structures can belong to the same schema only if they are assigned the same attributes at on the same defining positions. If there are h defining positions this occurs with probability $\left(\frac{1}{2}\right)^h$. For j exceptional structures, instead of 2, the probability is $(1/2^{j-1})^h$. Since there are $\binom{n}{h}$ ways of choosing h out of n detectors, and $\binom{x}{j}$ ways of choosing j out of x exceptional structures, the expected number of schemata defined on h positions and containing exactly j exceptional structures

$$\left(\frac{1}{2^{j-1}}\right)^h \binom{l}{h}\binom{x}{j}.$$

For example, with $l = 40$ and $x = 10^5$ (so that the density of exceptional structures is $x/2^l = 10^5/2^{40} \cong 10^{-7}$), $h = 20$ and $j = 10$, this comes to

$$(1/2^9)^{20}(40!(20!20!))(10^5!/(99,990!10!) \cong 3.$$

Noting that a schema defined on 20 positions out of 40 has $2^{20} = 10^6$ instances, we see that the 10 exceptional structures occur with density 10^{-5}, an "enrichment" factor of 100. ... Stated another way, *even when there can be no correlation between attributes and performance*, the set of schemata cuts through the space of structures in enough ways to provide a variety of "enriched" subsets. Intrinsic parallelism assures us that these subsets will be rapidly explored and exploited.
[present author's emphasis]

This is an extremely exciting result, and appears to offer the much sought-after "free lunch", suggesting that the genetic algorithm will be able to make good progress using *any* of the $s!$ bijections discussed above. Further reflection, however, suggests that a reproductive plan will not, in fact, be able to make use of these enriched schemata in the way that Holland describes. For suppose, as Holland effectively did, that a random bijective representation $\rho \in \mathcal{C}^{\mathcal{S}}$ (the set of all possible mappings of structures to chromosomes) is chosen. Then the probability that any chromosome exhibits high performance is *by assumption* independent of the performance of any other chromosome or set of chromosomes (up to finite-size effects). If the Fundamental Theorem depended upon the ratio $\mu_\xi(t)/\bar{\mu}(t)$ then the existence of high performance schemata would ensure that, once they were sampled, the reproductive plan would multiply their numbers, (assuming a suitably low rate of schema disruption by the genetic operators) thus performing effective search. But the use of the sample utility in the ratio $\hat{\mu}_\xi(t)/\bar{\mu}(t)$ precludes this guarantee. Thus there is a "representation problem", a term which will be used repeatedly to refer to the problem of devising a chromosomal representation inducing schemata whose members have well-correlated performances.

This representation problem may be inverted: suppose that for some problem an equivalence relation is known which relates chromosomes with correlated performances. Then it is clearly desirable that the representation used allow the equivalence classes induced by this equivalence relation to be expressed as schemata, for only then is there any reason to suppose that the genetic algorithm will be able to exploit the correlations. This observation forms the basis of much of the work in chapter 5, and is discussed in relation to neural networks in chapter 7.

8 Linkage and Inversion

It should be apparent that within the formulation described thus far the physical location of a gene on the chromosome has some significance. The probability of two genes being separated by a crossover operation is directly proportional to the distance between them. Specifically, the probability that the cross point falls somewhere between sites i and j on a chromosome of length n is

$$P(\textit{cross point falls between } i \;\&\; j) = \frac{|i-j|}{n-1}$$

since there are $n-1$ possible cross points. This bias in the crossover operator—which Eshelman *et al* (1989) call *positional bias*—could be eliminated by the simple expedient of making a random choice at each locus as to which parent the gene was taken from. Indeed, this method is now being adopted by some workers, and the resulting operator is known as "uniform crossover". Holland, following the biological analogue, sought instead to exploit this bias.

The distance between two genes on a chromosome defines their *linkage:* genes which are physically close on the chromosome are said to be *tightly linked,* implying that they are likely to be transferred together during crossover operations, whereas genes which are far apart are said to be *loosely linked,* implying that they are unlikely to be transferred together. Short schemata (those with small definition length) specify correlations between tightly linked sets of genes, and instances of such schemata are relatively likely to produce children, under crossover, which also instance the given schema. If the layout of the genes on the chromosome is random, however, given a fixed number of definition points (order), there is no reason to suppose that the correlations between the performances of instances of short schemata are any greater than those between instances of schemata having larger definition lengths. For this reason, rather than having some pre-determined linkage between different genes on the chromosome, it would be useful to have a mechanism which allowed the reproductive plan itself to determine appropriate linkage. Holland proposed that this be achieved through the application of the widely misunderstood and rarely used *inversion operator.*

The usual description of inversion involves rearranging the order of the genes on the chromosome, but because positions on the chromosome determine their meaning this is a rather misleading way of describing the process. A more helpful description associates with each gene a unique number from $\mathbb{Z}_n$, or equivalently, with each chromosome some permutation π from $\mathcal{P}_n$—the set of all permutations of n objects:

$$\begin{array}{|c|c|c|c|}\hline \pi^\eta_1 & \pi^\eta_2 & \cdots & \pi^\eta_n \\ \hline \eta_1 & \eta_2 & \cdots & \eta_n \\ \hline \end{array}$$

This effectively assigns a second ordering to the genes on the chromosome. The physical ordering on the chromosome is used to determine the meaning of each gene, which ensures that children produced by crossover are well-formed, having exactly one allele at each locus on the chromosome, drawn from one or other parent. The specification of the cross-point, is in terms of the alternative ordering induced by the permutation. For this reason it is convenient to call the permutation associated with a chromosome its *linkage information*, for its sole function is to specify the linkage between the various genes on the chromosome. A crossover operator can then be defined as follows:

$$X : \mathcal{C} \times \mathcal{P}_n \times \mathcal{C} \times \mathbb{Z}_n \longrightarrow \mathcal{C} \times \mathcal{P}_n$$

$$\text{with} \qquad X_i(\eta, \pi^\eta, \zeta, r) = \begin{cases} (\eta_i, \pi_i), & \text{if } \pi_i < r, \\ (\zeta_i, \pi_i), & \text{otherwise.} \end{cases}$$

For example, taking $n = 7$,

$$\left.\begin{array}{c} \begin{array}{|c|c|c|c|c|c|c|}\hline 2 & 5 & 7 & 1 & 3 & 6 & 4 \\ \hline \eta_1 & \eta_2 & \eta_3 & \eta_4 & \eta_5 & \eta_6 & \eta_7 \\ \hline \end{array} \\ \\ \begin{array}{|c|c|c|c|c|c|c|}\hline 7 & 4 & 6 & 1 & 2 & 3 & 5 \\ \hline \zeta_1 & \zeta_2 & \zeta_3 & \zeta_4 & \zeta_5 & \zeta_6 & \zeta_7 \\ \hline \end{array} \end{array}\right\} \overset{X(\ldots,3)}{\longmapsto} \begin{array}{|c|c|c|c|c|c|c|}\hline 2 & 5 & 7 & 1 & 3 & 6 & 4 \\ \hline \eta_1 & \zeta_2 & \zeta_3 & \eta_4 & \zeta_5 & \zeta_6 & \zeta_7 \\ \hline \end{array}.$$

Notice that only one set of linkage information is used, which may be taken without loss of generality to be associated with η, and that because of this the crossover operator X is not symmetric with respect to the two parents. The linkage information for the child is copied directly from the 'principal' parent η. While ideally this would be the case, it is difficult to use the linkage information from both chromosomes. This should not be a serious problem, for the aim is simply to give the algorithm the power to decide the linkage itself, and this it retains: the mechanism is merely weaker.

This said, Goldberg & Lingle (1985) suggested an ingenious way of tackling this problem, through the introduction of PMX—partially-mapped crossover. PMX provides one of a growing number of known ways of recombining permutations (some of which are described in chapter 8), and having developed the operator to tackle the travelling sales-rep problem (TSP), Goldberg & Lingle suggested using it to provide linkage information for a child which is a combination of that of its two parents. In fact, they did not present results for this, and the experiments detailed in chapter 6 suggest that such techniques are unlikely to prove very useful, but the idea remains very appealing.

Regardless of how linkage information is passed to children, in order for adaptation to be able to search for good linkages it is necessary to define an operator whose function is to change the linkage of the genes on a chromosome. This can be viewed as a kind of mutation operator for linkage, either performing independent search or doing so in conjunction with PMX or some other permutation crossover operator. Conceptually, inversion reverses the order of a contiguous portion of the permutation. Thus, given two distinct *inversion points*, p_1 and p_2, and assuming (without loss of generality) that $p_1 < p_2$, inversion can be defined as

$$I : \mathcal{P}_n \times \mathbb{Z}_n \times \mathbb{Z}_n \longrightarrow \mathcal{P}_n$$

$$\text{with} \qquad I_i(\pi, p_1, p_2) = \begin{cases} p_1 + p_2 - \pi_i, & \text{if } p_1 \le \pi_i \le p_2, \\ \pi_i, & \text{otherwise.} \end{cases}$$

This slightly impenetrable definition will be illuminated by the following example where the chromosome length n is taken to be 8:

$$I((13467285), 3, 6) = (16537284).$$

Inversion should play a very useful rôle in a genetic algorithm by subjecting the linkage between different parts of a solution to adaptive search. Inversion is applied to each child after reproduction with some probability p_I, and when it is applied the two inversion points are each chosen uniformly and randomly. Inversion does not in any way affect the meaning of the chromosome,[8] nor its utility, but rather changes the likelihood of various of its genes being transferred to a child *en masse* under a crossover operation. In effect, the presence of inversion allow the genetic algorithm to move the definition points for high-scoring schemata closer together, and in doing so increases the probability of children's instancing the same high-scoring schemata as their parents.

Two related issues merit brief mention here. The first is the observation that few workers seem to implement inversion in genetic algorithms. This appears slightly perverse and might be expected to degrade the performance of the algorithm unless there are *a priori* for supposing that the physical arrangement of the genes along the chromosome is appropriate in the sense that there is some intrinsically greater symbiosis between the parts of the solution represented by neighbouring genes. The work in chapter 6 sheds some light on this.

A possible partial explanation is the work of De Jong (1975), in which he advocates viewing the chromosome as a circle instead of a line segment by establishing linkage between positions 1 and n, and selecting *two* cross points, exchanging genetic material from the region between them. In the context of chromosomes without linkage information, where the cross-point is defined with reference to the physical location of genes, this suggestion is helpful and removes some of the positional dependence of the genes by eliminating the "special" endpoints. (The algorithm will no longer behave differently if the genes are "rotated" on the chromosome, for their linkage is unchanged by this.) On the other hand, if inversion

[8] i.e., the structure in the 'real' search space to which it corresponds

is used it is not immediately apparent that anything is gained from the use of circular chromosomes, for although in effect, one of the cross-points is then always chosen to be between sites n and 1, the meaning of the genes at these sites is no longer determined *a priori*, but rather by the algorithm. The issue of linkage is discussed further in chapter 6.

9 Maintaining the Gene Pool: Mutation

The 'genetic algorithm' so far described is incomplete. One of the problems that can be encountered is that every instance of some allele for a particular gene may disappear from the population. For while crossover never introduces new alleles, the population dynamics allow deletion from the population, and it could happen that some allele is lost to the population entirely. Once this happens, some portions of the space become unavailable to the search.

Mutation is used as a "background operator" occasionally replacing some allele with another selected at random. This ensures that no allele is ever permanently lost to the search, even if the measured performances of chromosomes (and schemata) which use it are low. This is important because the genetic algorithm may lose these alleles as a result of sampling error, (or indeed because schemata do not admit adequate expression of the correlations in $\mathcal{S}$). The mutation rate p_M is usually defined as the point-wise probability of a mutation at any gene. In principle, therefore, the mutation operator takes the form of a set of n operators

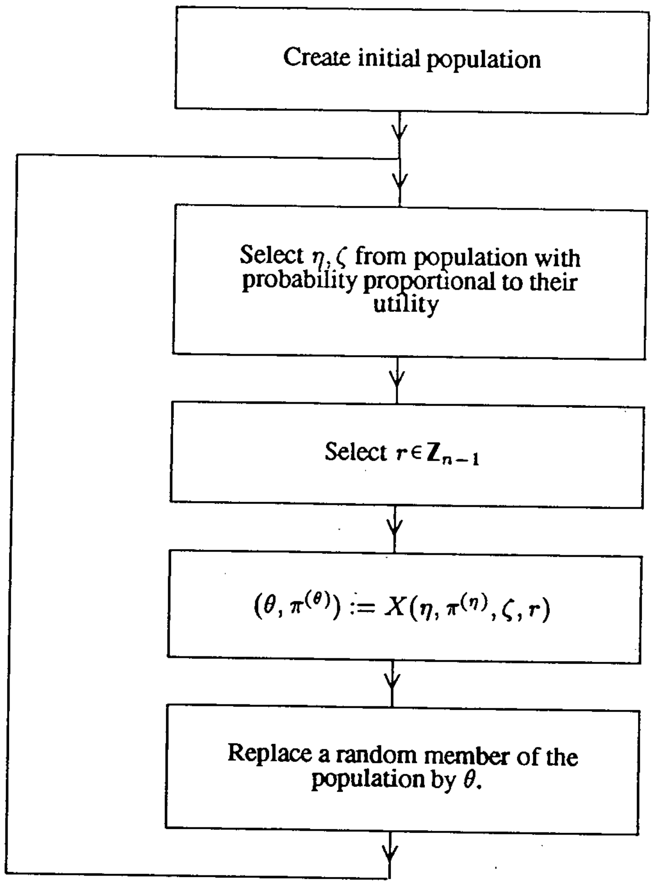
$$M_i : \mathcal{C} \times \mathcal{G}_i \longrightarrow \mathcal{C},$$

each of which replaces the ith gene of a chromosome with the given allele from $\mathcal{G}_i$

$$M_i(\eta_1\eta_2 \ldots \eta_n, a) = \eta_1\eta_2 \ldots \eta_{i-1}a\eta_{i+1} \ldots \eta_n.$$

The aim is to keep the rate of application of mutation as low as is consistent with keeping the gene pool well-stocked, for mutation is intrinsically disruptive, causing alleles from chromosomes (and schemata) which have been performing well to be discarded. Its introduction completes this preliminary survey of genetic algorithms. The following figure summarises the reproductive plans so-far examined.

10 General Reproductive Plan


Create initial population
Select η, ζ from population with probability proportional to their utility
Select $r \in \mathbf{Z}_{n-1}$
$(\theta, \pi^{(\theta)}) := X(\eta, \pi^{(\eta)}, \zeta, r)$
Replace a random member of the population by θ.

What a "reasonable time span" is
depends strongly on the environments (problems)
under consideration,
but in no case will it be a time
large with respect to the age of the universe.
— JOHN HOLLAND, Adaptation in Natural and Artificial Systems (1975)

When this work was started, the intention was to tackle the "full" network design problem for layered, feed-forward networks. In the language of the previous chapters, the search space S was to be Ω_L—the space of all such networks—so that the reproductive plan would have control not merely over the topology of the network, or the connection strengths, but also over the node categorisation (the division of the nodes N into input nodes I, hidden nodes H and ouput nodes O) and the activation functions. The ultimate hope was to abandon even the restriction to feed-forward networks, and to take Ω as the search space. As will be seen in the following survey of work in the field, this is very much more ambitious than the usual approach, in which either the topology is fixed and the reproductive plan manipulates only the weights, or alternatively some other training scheme—almost invariably back-propagation—is used to train networks, while the genetic algorithm manipulates only the pattern of connectivity. In these circumstances, the very ambitious programme outlined requires some justification. This is given both during the commentary on the work of others in the field (presented next), and also in the final section of this chapter.

1 Genetic Topology Selection

Probably the greatest research effort in combining the connectionist and evolutionary approaches to machine learning has focused on using a genetic algorithm to select a topology (connectivity) for a neural network which is subsequently trained using a conventional algorithm such as back-propagation of errors. This approach has many attractions, perhaps the most obvious being that it could be thought crudely to model the way in which biological neural networks are arrived at, evolution providing the "topology" in the form of the neural connections (initially) present, and the learning processes upon which "connectionism" is based working within these constraints to develop brains and minds capable of the extraordinary range of behaviours which humans and other organisms exhibit. The emphasis here, however, while drawing inspiration from nature, and respecting her elegance and parsimony, is ultimately on developing powerful algorithms rather than modelling natural systems, so while this observation is suggestive it will not be taken as constraining.

Probably the more powerful case for applying the machinery of genetic algorithms to the selection of topologies, is our extraordinary lack of knowledge about which connection patterns are in fact appropriate. It is perhaps an exaggeration, but not a wild one, to say as Miller, Todd & Hegde (1989) do, that 'the [topology] design stage remains something of a black art', and that '[t]hose seeking radically new architectures cast off into uncharted darkness'. Certainly it seems to be the case that the methods employed by workers in neural networks rarely consist of more than simple heuristics along the lines of "if it fails to work, add some more hidden nodes" and "if it over-learns, reduce the number of hidden nodes." Seitsma & Dow (1988) have made a more detailed study of some of these processes, and suggest that—at least within the context of learning by back-propagation—more hidden nodes are required in order to train a network than for the processing of patterns by the trained network. They advocate training with a relatively large number of hidden nodes, and then looking for correlations between the firing patterns of these nodes in the trained network with a view to replacing groups of nodes with strongly correlated patterns of activity by a single node. In this way they claim both to reduce training times and to improve the ability of the network to generalise from training data.

These insights notwithstanding, little is known about appropriate network topologies, and the case for any scheme which allows a more systematic approach to selecting a suitable connectivities is powerful. In the words of Miller, Todd & Hegde (1989), '[w]e seek an automated method for searching this vast, undifferentiable, epistatic, complex, noisy, deceptive, multimodal surface'—their surface being the space of possible network topologies. This characterisation of the relatively limited problem of choosing a topology emphasizes how very ambitious is the goal pursued in these pages, namely the selection of a network from Ω by a single, unified search procedure. Yet it also begins to explain why genetic algorithms might provide a suitable choice for this and other parts of the search, for while they have not yet proved themselves to be universally reliable, reproductive plans have successfully been used to tackle problems exhibiting all of the problematic traits mentioned in various combinations, and many of these were important motivations in their initial development (Holland (1975)).

Considerable emphasis already has been placed in this work on the importance of chromosomal representations when using reproductive plans, and in particular on the interplay between the representations adopted, the idealised genetic operators used, and the underlying correlations in the problem under study. Miller, Todd & Hegde (1989) distinguish two approaches to tackling network topology optimisation according to what they call the *developmental specificity* of the representation. When describing the genetic algorithm in chapter 3, the representation or coding function

$$\rho : \mathcal{S} \longrightarrow \mathcal{C}$$

was introduced, but it was pointed out that in practice it is the inverse mapping

$$g : \mathcal{C} \longrightarrow \mathcal{S}$$

which is more often the focus of attention, and strictly it is only this mapping that is required to be well-defined. (If ρ is a bijection then $g \equiv \rho^{-1}$, but if not ρ becomes formally ill-defined and attention is forcibly shifted to the "growth" function g.)

Similarly, while for many purposes it is convenient to regard the evaluation function μ as assigning a utility to each *chromosome*, it is more properly the structures in $\mathcal{S}$ which exhibit utility. The blurring of the distinction between the structure in $\mathcal{S}$ and its representative (or representatives) in $\mathcal{C}$ has disguised a potentially very important issue, hitherto alluded to but not expanded upon. This is that the decoding process itself may be far from trivial. While in simple problems it may be the case that $\mathcal{C} = \mathcal{S}$ (or more carefully, that the "natural" representation of the structure in a computer is a linear string suitable for genetic manipulation) it is hard to imagine such an identification being made when the chromosomes represent network descriptions of some kind. Clearly some *developmental machinery* will be required to perform the *morphogenesis* which converts the *genotype* (chromosome) into its expressed *phenotype* (corresponding structure in $\mathcal{S}$).

While this may appear all to be a matter of definition and trivial distinctions, the issues involved are in fact quite profound, for morphogenesis is the "flip side" of representation. The "developmental specificity" of Miller, Todd & Hegde (1989) refers to the degree of accuracy with which a genotype can specify a particular phenotype. In the context of network topology, "weak" specification schemes are those which describe higher level attributes of a particular topology, leaving the details to be fixed by the developmental machinery encapsulated in the growth function g. Harp, Samad & Guha (1989) propose one such weak specification scheme in which they regard a chromosome as a "blueprint" which describes some of the more important high-level features of the topology. They say that '[i]deally, a representation should be able to describe all potentially "interesting" networks, i.e., those capable of doing useful work, while excluding flawed or meaningless structures'—clearly a far cry from the simple bijection ρ envisaged earlier. Their scheme is based on the notion of "areas", which correspond to groups of nodes. They fix the number of "areas", but allow each area, in addition to a general description of its size, and approximate relative spatial dimensions, to contain a variable number of "projection specification fields"—groups of partially-specified connections to other areas, "addressed" either "absolutely" (in terms of fixed area identifiers) or "relatively" (in terms of relative area identifiers). Additionally, each projection specification field contains some information about the way that the learning algorithm should treat the group of connections.

As Miller, Todd & Hegde (1989) comment, blueprint schemes like those of Harp, Samad & Guha (1989) 'may be good at capturing the architectural regularities of large [network topologies] rather efficiently', but 'necessarily involve either severe constraints on the ... search space, or the stochastic specification of individual connections'. For these reasons they choose a "strong" specification scheme, in which the chromosome is isomorphic to the connection matrix for the network, and the search space is the entire set of topologies using (at most) some fixed number of nodes. This scheme certainly allows a more conventional chromosome than the ones required for the "blueprint" representation, binary strings sufficing to code the connection matrix, and in principle allows the use of conventional genetic operators, though this they abjure. The blueprint scheme, on the other hand, involves using a variable-length representation, with extra control information to allow the chromosome both to be decoded properly by the developmental machinery and to be recombined meaningfully.

Before discussing the relative merits of the two models, a third consideration, introduced by Mjolsness, Sharp & Alpert (1988), merits consideration. They begin with the observation (also made by Dodd (1989) and others) that '[t]he human genome has been estimated to contain 30,000 genes with an average of 2,000 base pairs each ... for a total of roughly 10^8 base pairs; this is clearly insufficient to independently specify the 10^{15} synapses ... in the human brain'. From this they conclude that rather than individual connections, it is probably synaptic growth rules which are specified genetically, and go on to make a strong case for restricting attention to *structured* network topologies. Their elegant approach involves the introduction of recursion relations which, properly seeded, allow construction of networks of arbitrary size. They examine the scaling properties of network topologies found to be useful for small problems when their higher-order counterparts are trained on problems which are themselves "scaled-up versions" of the problems used to test and select the smaller networks, and report generally encouraging results.

Though the scaling networks described are very pleasing, they are far from mainstream and will not be discussed further in this context, but the strongly-specified model of Miller, Todd & Hegde (1989) and the weakly-specified "blueprint" model of Harp, Samad & Guha (1989) deserve further examination. Both of these models were designed to study feed-forward networks (though not necessarily strictly layered networks) yet neither has any constraint in the representation to ensure that recurrent networks are not generated. Moreover, it is perfectly possible within either framework to generate topologies in which there is no path from the input nodes to the outputs. Of these, the former is perhaps the more serious, for while a network without connections from input to output may simply be considered to have minimum utility, a network with feedback corresponds to no structure in $\mathcal{S} = \Omega_F$. Both sets of workers tackle this problem by "purifying" the invalid networks—failing to build in feedback connections. Given the representations used, there is perhaps little other choice, but the implications of this are severe indeed, for there will in general be more than one way to "purify" a network, and it is hard to imagine a principled way of selecting between these: the process will inevitably be either stochastic or consistent but arbitrary in its determinism. More fundamentally, however, the introduction of this arbitrary process seems likely to add to several of the difficulties already faced by the reproductive plan, in particular to the epistatic and deceptive effects, and to the noise of evaluation. These points merely emphasize once more how difficult is the problem of finding a good coding, and how littered with pitfalls is the topological landscape.

The problem of finding a reasonable chromosomal representation for neural networks—whether it be connections or weights which are to be stored—suffers even more serious difficulties than these, however. Indeed, the single greatest motivation behind the path followed in these pages has been the "hidden node problem", well known within the neural networks community, but rarely mentioned in connection with genetic algorithms. The problem lies simply in the fact that hidden node labels are arbitrary. If the hidden nodes are distinguished in some way—for example, by having different activation functions—this arbitrariness might not cause difficulties, but in the more common case the problem is extremely severe. Its detailed consequences depend upon the particular representation used and the kind of search being performed, but the overall effect is invariably that "naïve" representations (and often sophisticated representations also) exhibit enormous redundancy. For example, in a layered network, the consequence is that the nodes in any layer may be permuted arbitrarily without in any way affecting the functionality of the network. If this problem is not specifically recognised and avoided the result will be a vastly inflated search space. For example, suppose that a modest 12–10–4 network[9] is being used: the redundancy in naïve representations is $10! \simeq 4,000,000$.

This difficulty was recognised, in the special context of fully-connected, layered, feed-forward networks, by Belew, McInerney & Schraudolph (1990), who share the author's interpretation of this as a truly daunting problem, writing that the 'invariance of BP networks under permutation of the hidden units is such a devastating and basic obstacle to the natural mapping of networks onto a GA string that we might consider ways of normalising network solutions prior to crossover'. They then go on to discount this method on grounds of computational cost, and though the approach was developed some time ago by Radcliffe (1988) and is described in chapter 7, it was never implemented for this reason and others discussed later.

It is worth dwelling on this slightly longer to explain why the redundancy resulting from undetectable hidden node permutations is such a major obstacle to genetic algorithms but not to techniques such as back-propagation. The answer lies in the nature of correlations which genetic algorithms are capable of detecting and utilising. Using conventional operators, it is almost correct to say that the *only* kinds of correlations between chromosomes that a genetic algorithm can "understand" are those capable of being expressed as common membership of some schema.[10] Further, genetic algorithms were specifically designed as a *global* search technique. These characteristics contrast markedly with gradient techniques such as back-propagation, which recognise correlations between solutions which are spatially close, for these techniques are essentially *local*. In consequence, whereas back-propagation will rarely be troubled by the symmetry in network problems (provided that the starting configuration is asymmetric), simply descending into the nearest local (or indeed, the nearest global) optimum, the genetic algorithm has either to recognise and "fold out" the redundancy or to explore the whole space oblivious to the symmetry. In fact, if a genetic algorithm *can* converge under such circumstances, this is a failure rather than a vindication of the technique, for it can do so only by abandoning the global search. (The way in which the problem actually arises in practice, as opposed to the abstract description in terms of schemata, is that two chromosomes when crossed may not be crossing genuinely homologous parts of the solution; that is, the components of the solution are identified only through arbitrary labels, and are likely, therefore, to

[9] in this text the first number is always the size of the input layer, following through the hidden layers in sequence, to the last, which is the number of output nodes

[10] This statement is expanded in chapter 5

be used for entirely different purposes in the two networks.)

Returning to the network representations discussed above, it is instructive to examine how this redundancy manifests itself. In the case of the blueprint scheme of Harp, Samad & Guha (1989), it is the area labels which are arbitrary. Since there is no *a priori* differentiation between any of these areas, a chromosome allowing up to h areas would be expected to exhibit $h!$ redundancy from this source alone. In the case of the higher-specificity scheme of Miller, Todd & Hegde (1989) the redundancy expresses itself more directly, as columns and rows in the connection matrix representing hidden node connections may be permuted arbitrarily.

Given these observations it is surprising (and perhaps worrying) that both groups have found good results with their approaches. Further analysis must await the construction of more analytic machinery in chapter 5

2 Genetic Training Techniques

Although the greater focus of attention has thus far been on the application of genetic search techniques to finding good topologies for neural networks, equally valid (if less biologically grounded) is the pursuit of genetic training algorithms for neural networks. The lack of biological grounding for this approach is taken to be extremely important by some workers especially Mühlenbein & Kindermann (1989), who distinguishes between three types of learning—"genotype learning" (which has here been called topology selection), "learning during development" (morphogenesis) and "phenotype learning", (adjusting weights within a fixed topology). Mühlenbein & Kindermann (1989) argues that '[t]here is general agreement that genetics, environment and learning [determine] the construction and the higher level functions of the nervous system. ... Extensive information processing capabilities of neural networks are possible with a very low contribution of genetically determined network characteristics, if time and memory characteristics would be unlimited for an individual.' His view is that phenotype learning (using some non-genetic technique) should shape the behaviour within generations while genetic evolution operates between the generations, and has developed a system called "Pandemonium II" in which networks with genetically-specified topologies learn to play the "prisoner's dilemma" (Axelrod (1987)). Mühlenbein (198?) writes that '[l]earning is done on a fast time scales, evolution on a slow time scale. During learning the meta-control variables of the system (which can be called the genome) are held fixed. On top of this, evolution tries to increase the performance by changing the meta-control variables to cope with the changing environment.'

Mhlenbein's reservations nowithstanding, some workers have investigated genetic approaches to training, perhaps because of the rather limited success and applicability of conventional techniques. The non-genetic methods currently used for training include the ubiquitous back-propagation (Rumelhart, Hinton

& Williams (1986)) for feed-forward networks, perceptron learning (Rosenblatt (1962)) for perceptrons, the Hopfield-Little algorithm (Hopfield (1982), Little (1974)) for Hopfield networks and constructive techniques such as those of Mezard & Nadal (1989) and Frean (1989). Each of these has fairly limited scope and while there are convergence theorems for some of the techniques, these do not guarantee completion in reasonable time. More particularly, since the present focus is on feed-forward networks, while back-propagation has had notable successes and remains by far the most robust technique known for training these networks, it will not reliably converge, is somewhat sensitive to initial conditions (weights) (Kolen & Pollack (1990)) and performs extremely poorly on nets with more than a few layers, presumably because the strength of the back-propagating error-correcting signal attenuates and loses accuracy as it propagates further away from the output nodes. Moreover, back-propagation is if anything less biologically plausible than genetic approaches, and requires differentiable activation functions in order even to be a candidate technique.

The problem of finding a suitable set of weights for network is far from trivial even in cases where the topology is known to admit solutions. Borrowing the catalogue of difficulties compiled by Miller, Todd & Hegde (1989) in the context of network connectivities, the weight space might be characterised as potentially undifferentiable, epistatic (since weights can have highly non-linear combined effects), noisy or explicitly misleading (since the error function measured is not necessarily the one which will be used to make the final assessment of the network, as for example when generalisation is required), deceptive (since it is hard to imagine representations in which schemata would accurately lead to the solution, and since similar weight configurations can have wildly differing performances), and multimodal (since radically different networks can exhibit near-identical performance).

The most consistent exponent of using genetic training techniques has been Darrell Whitley (Whitley (198?), Whitley, Starkweather & Bogart (1989), Whitley & Hanson (1989b), Whitley & Hanson (1989a)). His approach involves mapping a network having fixed topology onto a binary chromosome in the most straight-forward of manners and simply applying the standard genetic operators of crossover and mutation in order to perform the search. There are many reasons for supposing that this approach will not work well, yet surprisingly Whitley claims good results. He places much emphasis on the use of fine-grained update in genetic algorithms, and rank-based selection, points which will be discussed further in chapter 7.

A more interesting approach was taken by Montana & Davis (1989?). Davis's approach to problems generally involves defining a large number of operators, designed according to various heuristic considerations, and then either performing a series of tests to discover which operators are the most effective or using a meta-algorithm to controls the rates of application of all operators (Davis (1989)). The approach Montana & Davis took to training neural networks was to work directly with the network (at least conceptually) designing operators which manipulated its identified functional sub-components. They suggested six "mutation" operators, one of which was essentially a back-propagation step, and three recombination operators; of these, the more successful were determined and used. Several heuristic principles lie behind the operators they considered. The first is that the weights leading into a common node form a logical unit and might usefully be manipulated *en masse.* The intuition here is that 'networks succeed because of the synergism between their various weights, and this synergism is greatest among weights from ingoing links to they same node.' Although this statement is not justified in the paper, the logic is presumably that the functions of the input pattern which a node can compute are constrained by the incoming weights. Their second principle is that since selected parents in general have above average performance, information they hold should not lightly be discarded. In particular, rather than drawing new alleles at random from the initial distribution, this distribution should be centred about the value the weight currently holds. Their third principle, however, is perhaps the most profound and is certainly the closest to the approach taken here: they argue that when recombining networks, it is *functionally homologous* parts which should be swapped. In order to try to achieve this, they use the ingenious strategy of passing test patterns through the two networks to be recombined and attempting to identify nodes which have correlated firing patterns. (Parallels with the work of Seitsma & Dow (1988) are very strong and useful in this connection.) The hidden nodes from the two parent networks are then aligned as far as possible before recombination is performed, in a process they call "crossover-features".

Clearly the approach of Montana & Davis is markedly different from that of Whitley, and each has its strengths and weaknesses. Whitley's approach is capable of immediate analysis using the standard machinery of schemata and intrinsic parallelism, and is amenable to any general improvements suggested by mainstream work on genetic algorithms. Unfortunately, when schema analysis is performed the theory appears to suggest that his algorithm should fail catastrophically because of representational inadequacies, in accordance with the author's intuition, but possibly in conflict with Whitley's experimental results. In contrast, Montana and Davis's approach seems like an excellent heuristic way forward, and appears from their results to give good performance by "poaching" operators and ideas from other techniques and combining them in the unified framework of a reproductive plan. But it is not even obvious what a schema *is* in their framework, let alone that their operators manipulate them satisfactorily.

There is thus a genuine dilemma in choosing whether to follow Whitley's lead and coerce the problem until it conforms to the standard model of genetic algorithms and is encoded as a k-ary string, or to take the much more appealing and pragmatic approach of Montana and Davis, thus losing the opportunity either to appeal to the formalism of standard genetic algorithms analysed by schemata and with it the opportunity to carry over any useful results to other problem domains. The approach taken here seeks a route between these, and attempts to extend the schema analysis in such a way that it does apply to non-standard techniques such as those of Montana and Davis. This, in effect, is the goal of chapter 5.

3 Parameter Selection

The third area which might seem fruitful for genetic search is that of optimisation of parameters for other training techniques. Harp, Samad & Guha (1989) included specification of an initial step-size and its decay constant in each of their projection specification fields for groups of connections, and Shaffer *et al* (1990) included both the step-size and momentum parameters on the chromosome in their brief study of network topology optimisation. These are rather trivial extensions, but no doubt worthwhile.

Belew, McInerney & Schraudolph (1990) took the idea further by using a genetic algorithm to search for good sets of starting weights for back-propagation. Recent work by Kolen & Pollack (1990) indicates that the performance of back-propagation is significantly affected by the initial conditions used, so an automated technique for finding suitable starting configurations would be valuable. As Belew, McInerney & Schraudolph (1990) note, '[t]here is a pleasing symmetry to this search, in that the best initial weight vector (found by the GA) is obviously the same as the final weight vector (found by BP) . . . It is important to note, however, that the two algorithms are coupled in this way only if the range of initial weight values being explored by the GA is coextensive with the domain of solution weight vectors discovered by the gradient descent procedure.' Indeed, this project is appealing for many reasons. If it is only initial weight vectors which are required, it seems reasonable to assume that a coarser-grained weight description might suffice, and that the range of allowed values required might be significantly smaller than for the full problem. Belew, McInerney & Schraudolph (1990) further point out that '[w]hile it is true that the GA is seeking [initial weight configurations] that are "close to" the solution ultimately found by a gradient technique . . . the relevant measure is not the natural (e.g. Euclidean) distance between initial and final weight vectors . . . Rather, good [initial weight matrices] are close to good final solutions *with respect to the gradient procedure being used*' (original authors' emphasis). Thus it appears that there is a potential rôle for genetic search even in the context of networks with fixed topology and a classical training algorithm, though the problems of redundancy described in the contexts of topology selection and training will be present in the search for initial weight configurations also, if perhaps in less acute form. Belew, McInerney & Schraudolph (1990) further discuss fascinating "compromise" strategies in which the trained solutions are "reverse transcribed" onto chromosomes so that the evaluation process effectively improves the genetic quality of the chromosome's offspring, but point out that this is possible, in general, only if the spaces searched by the genetic algorithm and the training algorithm are coextensive, something which has already been argued to be unlikely. Although these ideas will not be pursued further here, they serve once more to emphasize the wealth of interesting and difficult problems presented by constructing a neural network to perform some required mapping.

4 Justification

The foregoing survey makes clear that there is significant scope for the application of genetic techniques to neural network construction, but also that the problems are in almost all areas extremely complex and very far from comprehensively solved. The case has not, however, been adquately made for tackling the topology and training problems together.

The case for attempting to automate the selection of suitable network topologies in some fashion is powerful, and genetic algorithms appear to be suitable candidates, in view both of their known characteristics and the initial successes suggested by the work of Miller, Todd & Hegde (1989) and Harp, Samad & Guha (1989). Once this is accepted, the question becomes whether it is efficient to separate this process (as is almost invariably done) from the training of the network. The implication of doing so is that some technique such as back-propagation in effect becomes part of the evaluation function. From the perspective of the implementation of the genetic algorithm, this has its attractions: computationally intensive evaluation functions allow the luxury of using rather more sophisticated (and computationally intensive) genetic techniques with relative impunity, so long as the time spent on the "genetics" remains small compared to the evaluation of the chromosome.

From the point of view of solving neural networks problems, however, the scheme appears less convincing in general, and it is sensible to consider the circumstances in which the use of such a "heavy" technique as a genetic algorithm could be justified. One such possible set of circumstances might arise when no topology could be found which allowed the particular problem to be solved, in other words, a situation in which the learning process was likely to be long. Even if the reproductive plan used were exceptionally efficient, however, this would suggest that many hundreds of training sessions would be required. Another set of circumstances might involve producing a large set of networks to perform different but related mappings. It might then become feasible search for a topology which reliably allowed efficient training on each of the different training sets, or indeed one which dramatically reduced the time required for the training scheme to locate a solution.

Belew, McInerney & Schraudolph (1990) argue that the scheme outlined is increasingly sensible simply because of the advent of large parallel computers, but the author prefers to adopt a less profligate stance. His feeling is that the approach is likely to prove too computationally expensive in most situations, and that while it might be necessary to adopt it if a unified approach to finding suitable connection patterns *and* connection strengths cannot be found, the unified scheme remains the greater—if substantially more ambitious—goal.

More fundamentally, if genetic search techniques are as powerful as they sometimes appear, it seems perverse to break down a problem artificially into the selection of a topology and the subsequent search for good connection strengths within that connectivity. For while this is admittedly not the standard view of the problem, it is surely the case that the ultimate interest lies in finding a good network in terms of its ability to perform some mapping, possibly within some constraints on the network, but rarely within some pre-defined connectivity.

5 Objectives

If the ultimate goal is to construct a genetic algorithm which is capable of searching some suitably defined subset of Ω, it is clear that this is very much too great a challenge to expect the solution to be forthcoming without breaking down the problem significantly. The developments over the two or so years since this project began have been encouraging, but modest, and it will be useful to summarise the principal objectives both as they appeared two years ago, and as they appear today.

5·1 Hidden Node Redundancy

As was stated earlier, perhaps the most serious obstacle to progress, and certainly the one which has most incessantly driven the research presented in these pages, remains the very basic problem of arbitrary hidden node labels. The most ingenious scheme for tackling this problem presented to date is surely that of Montana & Davis (1989?). Their solution is far from perfect, involving very significant computational costs and even then not lifting the redundancy entirely,[11] but provides an extremely valuable starting point.

Sadly, however, it is not obvious that it helps lift the redundancy when it is network topologies rather than the weights which are being optimised. Clearly, much work remains in this area. The ideas in chapter 5 were inspired directly by considerations such as these, in a quest to find a language which allowed more accurate expression of the difficulties posed by the permutation of hidden nodes, and in which insights might be gained which would allow further progress towards overcoming these hurdles.

5·2 Schema Analysis and Intrinsic Parallelism

It was argued in chapter 3 that the success of any reproductive plan depends on coding the problem in such a way that the underlying correlations in the search space are capable of expression as schemata, and that the genetic operators must manipulate the solutions in meaningful ways. Yet schemes which are either not amenable to analysis by schemata (because they violate grossly some of the conventional design rules), schemes such as those of Montana & Davis (1989?) and Harp, Samad & Guha (1989) appear to perform well. This suggests that the conventional view of the genetic algorithm as analysed by

[11] though it arguably lifts all the redundancy that matters . . .

schemata is incomplete, usefully describing a part of the space of reproductive plans, but failing to admit the possibility of more general sorts of intrinsic parallelism and genetic search.

Thus there is a case for trying to extend the formalism to cover more general cases, such as arise when neural networks form the search space. Clearly this is an ambitious goal, rather in the cavalier spirit of exceptional souls such as Newton, whose inclination when seeing a physical problem he was unable to tackle within the existing formalism of mathematics was invariably to extend the mathematics rather than to simplify or approximate his physical systems so that they became amenable to existing techniques, but since the formalism of genetic algorithms is rather less extensively developed than the mathematics that Newton inherited it is possible that the task will not prove entirely unfeasible.

5·3 Representation Issues

If representations are as significant as has been argued hitherto, then a major focus for research could usefully be a search for better ways of coding networks on chromosomes. The variety of techniques already discussed is quite wide, ranging from the very natural representations of Montana & Davis (1989?) and Miller, Todd & Hegde (1989), through the recursive, structured representations of Mjolsness, Sharp & Alpert (1988) to the very abstract "blueprint" representations favoured by Harp, Samad & Guha (1989), but the possibilities are boundless. No doubt as more becomes understood about the way in which neural networks work, the scope for sophisticated representations will increase, perhaps leading, in the words of Harp, Samad & Guha (1989), to those which 'describe all potentially "interesting" networks . . . while excluding flawed or meaningless structures'.

Even such basic questions as the most suitable representation for a real number are not yet resolved, for while the "mainstream" solution is undoubtedly to use binary coding, there are those such as Caruna & Shaffer (1985) who prefer Gray coding (to minimise the Hamming distance between adjacent values) and even those such as Montana & Davis (1989?) who prefer to code a real number as a floating point value. All options remain open.

5·4 Understanding Existing Results

If, as has been suggested, there is apparent conflict between theory and experiment in this area then clearly a principal objective should be the resolution of that conflict: if Whitley's coding involves massive redundancy and the genetic algorithm is incapable of recognising that redundancy, how does it work? Similar questions shroud many of the combinations of evolutionary and connectionist models described thus far, and some of them will be revisited in chapter 7.

5 Formae

The description of the conventional understanding of genetic algorithms in chapter 3 placed much emphasis on the "representation problem" and the importance of finding representations which allow schemata to capture such regularities and correlations as are likely to be present in the search space $\mathcal{S}$. The notion of intrinsic parallelism—the phenomenon whereby each n-gene chromosome instantiates 2^n schemata—was introduced, which has been the key theoretical tool for analysing and understanding genetic algorithms. In this context, schema analysis as conventionally understood, provides powerful arguments for using binary genes in order to maximise the degree of intrinsic parallelism available. It was pointed out, however, that not all problems, find natural expression as binary—or indeed, k-ary—strings. This chapter shows that the notion of intrinsic parallelism (and the associated "Fundamental Theorem") can be extended to non-string representations through the introduction of arbitrary equivalence relations and the replacement of schemata by their equivalence classes. It further provides a framework within which arbitrary genetic operators can be analysed. In doing so, a significant generalisation of the formalism under-pinning genetic algorithms is achieved, and insights are gained into general principles for designing representations and operators when applying reproductive plans in new problem domains. When applied to well-known domains this approach often reproduces some of the more successful operators discovered to date, and provides new insights into their successes.

1 From Schemata to Equivalence

Recall that a chromosome $\eta \in \mathcal{C}$ will be taken to be a string of n genes $(\eta_1, \eta_2, \ldots, \eta_n)$ drawn from sets of alleles $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n)$, so that the space of chromosomes is

$$\mathcal{C} \triangleq \mathcal{G}_1 \times \mathcal{G}_2 \times \cdots \times \mathcal{G}_n,$$

and schemata are members of the set

$$\Xi \triangleq \mathcal{G}_1^\star \times \mathcal{G}_2^\star \times \cdots \times \mathcal{G}_n^\star,$$
$$\text{where} \qquad \mathcal{G}_i^\star \triangleq \mathcal{G}_i \cup \{\square\}.$$

Schemata can be identified as the equivalence classes of a set of equivalence relations, Ψ, which identify chromosomes which share genes. The set of all such equivalence relations is conveniently represented by

$$\Psi \triangleq \{\, \square, \blacksquare \,\}^n,$$

where $\blacksquare$ is used to indicate genes which must match for equivalence, and $\square$, as usual, "matches" any allele. Taking $n = 4$, a particular equivalence relation is then $(\square, \blacksquare, \square, \blacksquare)$, which is conveniently abbreviated to $\square\blacksquare\square\blacksquare$. Intuitively, the idea is that two chromosomes are equivalent under this equivalence relation if they have the same alleles wherever the definition has the $\blacksquare$ symbol. More carefully, calling each $\square$ or

■ symbol in the string describing an equivalence relation a component, given any equivalence relation $\sim\in\Psi$, with components $\sim_1, \sim_2, \ldots, \sim_n$ and given chromosomes $\eta, \zeta \in C$

$$\eta \sim \zeta \iff (\forall i \in \mathbb{Z}_n\ (\sim_i = \blacksquare)\ :\ \eta_i = \zeta_i),$$

where $\mathbb{Z}_n = \{1, 2, \ldots, n\}$. That $\sim$ satisfies the conditions of symmetry, reflexivity and transitivity, and is therefore an equivalence relation, follows immediately from this definition and the properties of $=$.

The purpose of introducing the equivalence relations which induce schemata is to raise the possibility of imposing other equivalence relations on the search space (whether $\mathcal{S}$ or C) which will induce different equivalence classes. For schemata, fundamental as they have been to understanding genetic algorithms, are merely a mathematical tool for analysing and designing their behaviour. The population maintained and manipulated by any reproductive plan consists of individual chromosomes and it is their utility which is actually measured. Holland's inspired interpretation of such measurements as statistical sampling events over the space of schemata, however forceful, is simply that—an interpretation. There is freedom to analyse the algorithm in any way desired, through the introduction of such equivalence relations and classes as may be useful, and the objective of this work is to suggest a framework within which non-standard equivalence relations and equivalence classes may be expoited. The careful formulation of the Fundamental Theorem in chapter 3,

$$\left\langle n_\xi(t+1)\right\rangle \geq n_\xi(t)\frac{\hat{\mu}_\xi(t)}{\bar{\mu}(t)}\left[1 - \sum_{i=1}^{N_o} p_i p_i^\xi\right].$$

is equally valid if ξ is interpreted as an equivalence class of *any* equivalence relation $\sim$ on C (or equivalently, given a bijective coding function ρ, on the real search space $\mathcal{S}$) provided only that the coefficients p_i^ξ are calculated correctly according to

$$p_i^\xi \triangleq P\left(\Gamma_i(\eta) \notin \xi \mid \eta \in \xi\right). \tag{1}$$

A general method for bounding these coefficients is now discussed.

A fairly general recombination operator X has the functional form

$$X : C \times C \times \mathcal{A}_X \longrightarrow C,$$

where $\mathcal{A}_X$ is a set of *control parameters* that determine which of the typically many possible crosses between two chromosomes occurs. For example, in the case of one-point crossover (Holland (1975)), $\mathcal{A}_X = \mathbb{Z}_{n-1}$, the set of possible cross points. Both two-point crossover (De Jong (1975)) and partially-mapped crossover (PMX, Goldberg & Lingle (1985)) use the control set $\mathcal{A}_X = \mathbb{Z}_{n-1}^2$, the set of all pairs of cross points, and uniform crossover (for example, Syswerda (1989)) has $\mathcal{A}_X = \{0, 1\}^n$, the set of all n-bit binary masks. In the case of a few crossover operators (such as the "Heuristic" crossover for the travelling sales-rep problem (TSP) of Grefenstette *et al* (1986)) the control set depends upon the two chromosomes being crossed, so that it becomes appropriate to write $\mathcal{A}_X(\eta, \zeta)$ as the set of all control parameters governing which of the possible children that η and ζ could produce is in fact produced.

Given this structure, an often useful upper bound on the coefficient p_i^ξ of equation (1) can be calculated as follows. Let

$$\mathcal{A}_X^\xi \triangleq \{\, a \in \mathcal{A}_X \mid \forall \eta \in \xi \ \forall \zeta \in \mathcal{C} \ : \ X(\eta, \zeta, a) \in \xi \,\},$$

the set of parameter settings for which membership of ξ is passed to the child from the principal parent (η), regardless of the partner (ζ) chosen. Then p_X^ξ can be bounded by

$$p_X^\xi \leq \left(1 - w^\xi \frac{|\mathcal{A}_X^\xi|}{|\mathcal{A}_X|}\right), \tag{2}$$

where w^ξ is a weight to take account of the possibility that control parameters from $\mathcal{A}_X$ are not all selected with equal probability. In most cases (including all the crossover operators listed above) the choice is conventionally unbiased so that $w^\xi \equiv 1$. This bound is, in fact, the one used by Holland to derive the Fundamental Theorem, and is typically implicitly used in deriving variations for other operators.

A similar approach can be taken for mutation operators. Conventional point mutation can be viewed as a collection of n operators

$$M_i : \mathcal{C} \times \mathcal{A}_i \longrightarrow \mathcal{C}$$

with $\mathcal{A}_i \equiv \mathcal{G}_i$, the allele sets. Then

$$M_i(\eta_1\eta_2 \ldots \eta_n, a) = \eta_1\eta_2 \ldots \eta_{i-1} a \eta_{i+1} \ldots \eta_n.$$

The coefficients p_i^ξ are then given by

$$p_i^\xi = \begin{cases} 0, & \text{if } \xi_i = \square, \\ (|\mathcal{G}_i| - 1)/|\mathcal{G}_i|, & \text{otherwise.} \end{cases}$$

If each gene is drawn from a set of k alleles this yields

$$\sum_{i=1}^{n} p_i^\xi = o(\xi) \left(\frac{k-1}{k}\right).$$

2 Representations

There is little theory surrounding good representations for genetic algorithms. Holland (1975) suggested subjecting the representation itself to adaptation, but the author is aware of no implementation in which this approach is adopted outside the domain of classifier systems. The Argot Strategy (Shaefer (1989)) does alter the representation during the course of the search, but not in the manner suggested by Holland, nor in a way which is amenable to this analysis. Goldberg (1989), however, suggested the following two principles for good representations:

> *The Principle of Meaningful Building Blocks:*
> The user should select a [representation] so that short, low-order schemata are relevant to the underlying problem and relatively unrelated to schemata over other [defining] positions.
>
> *The Principle of Minimal Alphabets:*
> The user should select the smallest alphabet that permits a natural expression of the problem.

The analysis presented here focuses on the interaction between the chromosomal representation, *some* set of equivalence relations Ψ over the chromosomes, and the genetic operators used. Goldberg's principles, on the other hand, are formulated with respect to conventional chromosomal representations (n-tuples of genes drawn from sets of alleles) analysed with conventional schemata.

His first principle requires three things. First, it emphasizes the point made in the previous section, that as many equivalence classes (schemata) as possible should contain chromosomes which have correlated performance. Secondly, by seeking to reduce the defining length and order of good schemata it attempts to minimise the likelihood of disruption by the genetic operators. Finally, it tries to ensure that recombination of (instances of) different schemata works in a useful manner. The second principle attempts to maximise the degree of intrinsic parallelism available to the algorithm by ensuring that each chromosome instantiates many schemata.

3 Formae

The above considerations (and others) lead to the following proposals for constructing useful equivalence relations, good representations and suitable sets of operators. These principles are not all necessary for an effective genetic algorithm, and are certainly not sufficient for it, but might be expected to characterise good representations. To emphasise the link between these equivalence classes and schemata, the former will be referred to as *formae*,[12] and the number of formae induced by an equivalence relation will be referred to as the *precision* of the relation and the formae it induces.[13] From this point on, Ξ will be interpreted as the set of all formae induced by the equivalence relations in Ψ.

In order to express the design principles succinctly, it is useful to introduce the notion of compatible formae. Two formae ξ and ξ' will be said to be *compatible* if it is possible for a chromosome to instantiate both of them. Denoting this by $\xi \bowtie \xi'$, a more careful statement is

$$\xi \bowtie \xi' \iff \xi \cap \xi' \neq \emptyset .$$

3·1 Design Principles

1. (Minimal redundancy) *The representation should have minimal redundancy; such redundancy as exists should be capable of being expressed in terms of the equivalence relations used.*
This is highly desirable in order to minimise the size of the search space. If redundant solutions are related by one of the equivalence relations used then the genetic algorithm should effectively be able to "fold out" the redundancy (see principle 4); otherwise it is doomed to treat redundant solutions as unrelated.

2. (Correlation within formae) *Some of the equivalence relations, including some of low precision, must relate chromosomes with correlated performance.*
This ensures that information can usefully be gathered about the performance of a forma by sampling chromosomes which instantiate it, and that this information can usefully be used to guide the search. The emphasis is placed on low-precision formae because membership of these will generally be less likely to be disrupted by the application of genetic operators, and are also more likely to be compatible with one another.

3. (Closure) *The intersection of any pair of compatible formae should itself be a forma.*
This ensures that solutions can be specified with different degrees of accuracy and allows the search gradually to be refined. Clearly the precision of formae so-constructed will be at least as high as that of the higher-precision of the intersecting formae.

4. (Respect) *Recombining two instances of any forma should produce another instance of that forma.*
Formally, it should be the case that

$$\forall \xi \in \Xi \ \forall \eta \in \xi \ \forall \zeta \in \xi \ \forall a \in \mathcal{A}_X : \ X(\eta, \zeta, a) \in \xi ,$$

[12] Although Holland chose the neuter form for the Latin noun schema, there is no option but to choose the feminine form of its synonym, forma.

[13] In the case of schemata and genes with k alleles, the precision is k^o, where o is the order of a schema.

where X is the recombination operator. In this case the recombination operator will be said to *respect* the equivalence relations (and their formae). This is necessary in order that the algorithm can converge on good formae, and implies, for example, that $X(\eta, \eta, a) = \eta$ (assuming that equivalence relations of maximum precision specify chromosomes completely). It also effectively reduces the disruption rate in the Fundamental Theorem, though a more accurate value for p_X^ξ than that given in equation (2) is needed in order to see this.

5. (Proper assortment) *Given instances of two compatible formae, it should be possible to recombine them to produce a child which instantiates both formae.*
Formally,

$$\forall \xi \in \Xi \ \forall \xi' \in \Xi \ (\xi \bowtie \xi') \ \forall \eta \in \xi \ \forall \eta' \in \xi' \ \exists a \in \mathcal{A}_X : \ X(\eta, \eta', a) \in \xi \cap \xi'. \tag{3}$$

This relates to Goldberg's "meaningful building blocks", of which he writes (Goldberg (1989), p. 373)

> Effective processing by genetic algorithms occurs when *building blocks*—relatively short, low order schemata with above average fitness values—combine to form optima or near-optima.

A crossover operator which obeys equation (3) seems very much more likely to be able to recombine "building blocks" usefully, and any crossover operator which obeys this principle will be said *properly to assort* formae.

6. (Ergodicity) *It should be possible, through a finite sequence of applications of the genetic operators, to access any point in the search space $\mathcal{S}$ given any population $\mathfrak{B}(t)$.*
This is the *raison d'être* for the mutation operator.

3·2 Deception

Sadly, the formulation in terms of formae, rather than schemata, does nothing to eliminate the problem of *deception*, some degree of which is highly likely for non-trivial problems. Deception is normally understood as the property whereby good schemata combine to give poorer schemata of higher order. An example of a completely deceptive problem would be a volcano if the objective was to find the minimum: this is in the centre, surrounded by the worst points in the search space.

Deception can be stated in a general way as follows. Assume that there is a unique global optimum represented by $\eta^\star \in \mathcal{C}$, i.e.

$$\forall \eta \in \mathcal{C} \setminus \{\eta^\star\} : \ \mu(\eta) < \mu(\eta^\star).$$

Let the formae induced by any relation $\sim \in \Psi$ of precision k be $\xi_\sim^{(1)}, \xi_\sim^{(2)}, \ldots, \xi_\sim^{(k-1)}, \xi_\sim^\star$, where $\eta^\star \in \xi_\sim^\star$.

Then a representation is said to be deceptive with respect to Ψ if

$$\exists \sim \in \Psi : \max_i \mu \left(\underset{\sim}{\xi}^{(i)} \right) > \mu(\underset{\sim}{\xi}^{\star}). \tag{4}$$

In other words, a representation is deceptive (with respect to the equivalence relations in Ψ) if the global optimum is not in the equivalence class (forma) with highest utility for all of the equivalence relations.

4 Crossover and Formae

It is instructive to examine the way standard crossover operators interact with schemata (the "standard" formae) to see whether they respect and properly assort them in the sense of principles 4 and 5. The crossover operators which have traditionally been used are 1- and 2-point crossover. More recently, attention has focused on multi-point crossover and the so-called "uniform" crossover operator. Eshelman *et al* (1989) have also discussed "shuffle" crossover operators. Recall that uniform crossover makes an independent random choice as to which of the parents the allele at each locus is drawn from, and shuffle crossover shuffles the (effective) order of the genes before crossing over, removing "positional" bias in the sense of Eshelman *et al* (1989). All of these operators respect schemata, for it is plain that under all of them a child will instantiate any schema which both of its parents instance. Only the uniform and shuffle crossover operators, however, properly assort schemata.

To see this, consider the chromosomes and schemata $1010 \in 1\square 1\square$ and $0101 \in \square 1\square 1$. Plainly the two given schemata are compatible, with intersection 1111, but neither 1- nor 2-point crossover can cross them to produce 1111 in a single step. It should be clear that this kind of problem will arise for n-point crossover with any *fixed n*. Both uniform and shuffle crossovers, however, can recombine the two chromosomes as required (albeit with low probability) and it should be apparent that they always respect schemata.

5 Random, Respectful Recombination

The foregoing analysis has extended the scope of intrinsic parallelism from k-ary string representations analysed by schemata to arbitrary representations analysed by formae. The Fundamental Theorem has been shown to be equally valid in the context of general formae when suitably formulated, and principles have been suggested for identifying useful formae, representations and genetic operators. The aims of this project have been to liberate us from the constraints of k-ary representations, to challenge the hegemony of binary representations, to provide new insights into the successes and failures of current algorithms (and operators) and to suggest methods for tackling problem domains which have so far proved difficult. Having suggested a unified framework for the analysis of genetic algorithms, it would be extremely useful to examine similarities between operators used in different problem domains, and if possible to suggest general operators defined with respect to formae rather than representations. If this could be achieved, even if the operators were not ideal in all circumstances (which would seem too much to hope for), the application of adaptive techniques to new problem domains would be eased, and it might become possible more easily to transfer insights gained and developments made from one problem domain to another. Moreover, if operators were to be defined in terms of equivalence classes (formae) in the search space $\mathcal{S}$, rather than the representation space $\mathcal{C}$, the issue of representation would be moved away from centre stage. These are the (perhaps over-ambitious) goals towards which this section works.

5·1 Respect Revisited

The constraints on representations and formae expressed by principles 1–3 (minimal redundancy, correlation within formae, and closure of formae under intersection) are by their nature problem-specific. The remaining principles—respect, proper assortment and ergodicity—are more general in nature and form a suitable starting point for discussing general, representation-independent genetic operators.

Recall that in order for a recombination operator X to respect a set of formae Ξ, it is necessary that whenever two (parent) chromosomes are each instances of some forma ξ, all of their possible children under X instantiate that forma also. This notion can be formalised through the introduction of a *similarity set* for each pair of chromosomes η and ζ. The idea is that the similarity set contains all those chromosomes which are similar to η and ζ in the same sense that they are similar to each other. A suitable definition is

> *the similarity set of a pair of chromosomes η and ζ is the highest precision forma containing both η and ζ.*

This similarity set will be written $\eta \oplus \zeta$, and is defined formally as follows:

$$\oplus : \mathcal{C} \times \mathcal{C} \longrightarrow \mathbb{P}(\mathcal{C})$$

$$\text{with} \qquad \eta \oplus \zeta \triangleq \bigcap \{ \xi \in \Xi \mid \eta, \zeta \in \xi \}.$$

For example, in the familiar case where the formae are schemata,

$$\begin{array}{r} 1\;0\;1\;1\;1\;0\;0\;1 \\ \oplus\; 1\;0\;0\;1\;0\;0\;1\;0 \\ \hline 1\;0\;\square\;1\;\square\;0\;\square\;\square \end{array}$$

justifying the remark that the similarity set is the set of all chromosomes which are similar to η and ζ in the same sense that they are similar to one another. (The similarity set here is the set of all chromosomes sharing the same genes that η and ζ share.)

Given this definition, respect can be restated as the requirement that the offspring of any pair of chromosomes under recombination is a member of their similarity set, that is

$$\forall \eta \in \mathcal{C}\ \forall \zeta \in \mathcal{C}\ \forall a \in \mathcal{A}_X\ :\ X(\eta, \zeta, a) \in \eta \oplus \zeta.$$

5·2 Proper Assortment

Proper assortment, it will be recalled, depended upon the notion of compatible formae. If ξ and ξ' are compatible, so that $\xi \cap \xi' \neq \emptyset$, proper assortment requires that the recombination operator X be capable of generating a member of the intersection $\xi \cap \xi'$ given one parent $\eta \in \xi$ and another $\eta' \in \xi'$. Let $\mathcal{A}_X$ be the control set associated with X, and define further

$$\mathcal{A}_X^{\zeta}(\eta, \eta') \triangleq \{ a \in \mathcal{A}_X \mid X(\eta, \eta', a) = \zeta \},$$

the set of control parameters which cause X to produce ζ when recombining η and η'. Proper assortment can then be expressed as the requirement that there be some chromosome ζ in the intersection of ξ and ξ' for which $\mathcal{A}_X^{\zeta}(\eta, \eta')$ is non-empty. Formally,

$$\forall \xi \in \Xi \; \forall \xi' \in \Xi \; (\xi \bowtie \xi') \; \forall \eta \in \xi \; \forall \eta' \in \xi' \; \exists \zeta \in \xi \cap \xi' : \; \mathcal{A}_X^{\zeta}(\eta, \eta') \neq \emptyset .$$

5·3 Random Respectful Recombination

The design principles suggested require that the recombination operator simultaneously respect and properly assort the formae in Ξ, but for a general set of formae there is no guarantee that this is possible. Indeed, formae will later be introduced in the context of periodicity for which it is impossible simultaneously to achieve these goals. A set of formae Ξ will be said to be *separable* if it is possible for some recombination operator both to respect and properly to assort its members.[14]

Given this definition it is possible to construct a recombination operator, which will be called *random, respectful recombination*, R^3, which respects and properly assorts any set of separable formae with respect to which it is defined. Informally, given a pair of chromosomes, this operator chooses a random element from their similarity set as their child. More carefully, a binary recombination operator for finite spaces

$$R^3 : \mathcal{C} \times \mathcal{C} \times \mathcal{A}_r \longrightarrow \mathcal{C}$$

will be said to be a random, respectful recombination operator, provided that

$$\forall \eta \in \mathcal{C} \; \forall \zeta \in \mathcal{C} \; \forall a \in \mathcal{A}_r : \; \frac{|\mathcal{A}_r^{\theta}(\eta, \eta')|}{|\mathcal{A}_r(\eta, \zeta)|} = \begin{cases} 1/|\eta \oplus \zeta|, & \text{if } \theta \in \eta \oplus \zeta, \\ 0, & \text{otherwise.} \end{cases}$$

Of course, for any particular set of equivalence relations all random, respectful recombination operators so defined differ only in the structure of their control sets, and are therefore essentially equivalent. The continuous case is a trivial generalisation, introduced in the next section.

[14] "Separable" is chosen because of certain intuitions the author has about the implications of separability which do not yet merit setting down on paper, but which may in time make the choice seem reasonable.

Clearly[15] random, respectful recombination properly assorts the (separable) formae as well as respecting them, for it generates *every* solution a respectful operator is permitted to generate with non-zero probability.

6 Real-Valued Problems

Before going on to examine recombination and formae in general, it will be useful to look at an example problem domain where the ideas may prove useful. An enormous concentration of effort has gone into using genetic algorithms for real-valued function optimisation in various-dimensional spaces. This therefore forms a natural domain to study.

Conventional wisdom holds that real-valued problems are best tackled using binary representations because in this way the maximum level of intrinsic parallelism with respect to schemata is achieved. (Recall that each chromosome instantiates 2^n schemata, and that n is maximised for binary genes.) In practice, however, the intrinsic parallelism is useful only insofar as the schemata relate solutions with correlated performance, and it is far from clear that schemata are the most useful formae in this context.

The great strength of binary representations lies in their versatility: different schemata relate chromosomes on quite different bases. Indeed, their robustness is demonstrated by the wide variety of problems which have been tackled successfully using binary representations. For example, using the natural coding for a real number in the range $[\alpha, \beta]$, with N divisions

$$\rho(x) = \left\lfloor N \frac{x - \alpha + \delta}{\beta - \alpha} \right\rfloor$$

where $\delta = 1/2N$. The schema $1\square\square\cdots\square$ then specifies the upper half-space $x > (\alpha + \beta)/2$, whereas the schema $\square\square\cdots\square 1$ specifies alternate strips of width 2δ across the space, capturing some possible periodicities.

The problem lies in the fact that the schemata are not uniform over the space. Suppose that $\alpha = 0$ and $\beta = 1.5$ with 16 divisions. Then $\rho(0) = 0000$, $\rho(0.7) = 0111$, $\rho(.8) = 1000$ and $\rho(1.5) = 1111$. It is possible to specify the interval $[0.8, 1.5]$ exactly using the low-order schema $1\square\square\square$, whereas there is no schema of *any* order which specifies any range which crosses the "Hamming Cliff" between 0.7 and 0.8. Caruna & Shaffer (1985) advocate using Gray coding to avoid this (and other) problems, but the interpretation of schemata is then even less obvious. Nor is locality the only problem: while some periodicities of powers of two in the discretisation size are easily characterised using schemata, periodicities of three, for example, are incapable of being so represented. The problem seems to lie in the

[15] Clearly is used in the technical sense, and should not be taken to imply that it was obvious to the author before suitable notation was introduced, or that he did not spend many hours trying to prove the result.

fact that relatively few of the schemata seem to be induced by equivalence relations which group together "useful" sets of points.

Whether more useful equivalence relations can be developed depends very much on how much insight can be gained into likely kinds of structure in the problem. For function optimisation over intervals in $\mathbf{R}^n$, locality and periodicity seem like—though not useful—starting points.

6·1 Locality

Simplifying to functions of one real variable ($\mathcal{S} \subset \mathbf{R}$), suitable equivalence relations for capturing locality are intervals specified by a position and a radius. Let $\mathrm{B}[c, r)$ be the half-open interval $c - r \leq x < c + r$, $r \in \mathbf{R}^+$, and let $\mathrm{B}[c, 0) = \{c\}$. Then the equivalence relation specified by position p and radius r is

$$\eta \sim \zeta \iff (\exists k \in \mathbf{Z} : \eta, \zeta \in \mathrm{B}[p + 2kr, r))$$

with formae

$$\{ \mathrm{B}[p + 2kr, r) \mid k \in \mathbf{Z} \}.$$

Thus any interval $[\alpha', \beta')$ is an equivalence class under *some* equivalence relation.

Moving back to the more general problem of searching a space $\mathcal{S}$ which has n real-valued parameters:

$$\begin{aligned} & \mathcal{S} = \prod_{i=1}^{n} \mathcal{I}_i, \\ \text{where} \quad & \mathcal{I}_i = [\alpha_i, \beta_i] \subset \mathbf{R}, \end{aligned}$$

a suitable set of "locality" equivalence relations Ψ^L can be defined as

$$\begin{aligned} & \Psi^L = \prod_{i=1}^{n} \mathcal{I}_i^{\star}, \\ \text{where} \quad & \mathcal{I}_i^{\star} = \mathcal{I}_i^2 \cup \{\square\}. \end{aligned}$$

(The "don't care" character is strictly redundant, but is left in for notational convenience.) This induces formae which can be described using exactly the same set as for Ψ^L, namely

$$\Xi^L = \prod_{i=1}^{n} \mathcal{I}_i^{\star}.$$

An example of such a forma ξ with $n = 3$ can be written as $\langle \mathrm{B}[0.2, 0.1), \square, \mathrm{B}[0.5, 0.2) \rangle$, with the interpretation that a chromosome η instantiates ξ if $0.1 \leq \eta_1 < 0.3$ and $0.3 \leq \eta_3 < 0.7$. Formally,

$$\eta \in \xi \quad \iff \quad (\forall i \in \mathbf{Z}_n \; (\xi_i \neq \square) : \; \eta_i \in \xi_i).$$

If these equivalence relations are to be used, then a crossover operator should be constructed which both respects and properly assorts the formae they induce. Standard crossover with real genes would respect them, but would fail properly to assort them. An example should make this clear. The sub-formae[16] $B[0.4, 0.2)$ and $B[0.6, 0.2)$ are compatible with intersection $B[0.5, 0.1)$, but given genes $0.3 \in B[0.4, 0.2)$ and $0.7 \in B[0.6, 0.2)$ it is impossible for standard crossover to generate any value in $B[0.5, 0.1)$ since the result of such a cross will always either be 0.3 or 0.7. The Hamming cliffs also make it immediately clear that standard crossover with binary genes will not respect these formae.

The random, respectful recombination operator for this problem is:

$$X^F : \mathcal{C} \times \mathcal{C} \times [0,1]^n \longrightarrow \mathcal{C}$$

$$\text{with} \quad X_i^F(\eta, \zeta, r) = r_i |\eta - \zeta| + \min\{\eta, \zeta\},$$

which will be called *flat* crossover.[17] Given a pair of real-valued genes, this operator returns a random value within the interval between them. The choice is uniform provided that each r_i is chosen uniformly. (The control set here is $\mathcal{A}_F = [0,1]^n$.) Plainly this operator respects formae from Ξ^L, for if the two genes have the same value then the interval they define has zero width. Moreover, compatible formae ξ and ξ' have overlapping intervals at each locus. Given $\eta \in \xi$ and $\eta' \in \xi'$, it is clearly possible to choose a set of r_i such that each gene of the child sits within the intersection $\xi \cap \xi'$.

Thus X^F respects and properly assorts formae from Ξ^L, composed of intervals of arbitrary widths in the search space $\mathcal{S}$. A genetic algorithm using this might be expected to perform well on a real-valued problem for which locality *is* the appropriate kind of equivalence to impose on solutions, utilising the intrinsic parallelism of the many formae that each chromosome instantiates.

Two related problems, however, remain. The first concerns a bias in the operator, namely that it systematically biases the search away from the ends of the intervals, violating ergodicity in the sense of principle 6. The second is the question of a suitable mutation operator.

Recall that the rôle of mutation for k-ary string representations is usually understood to be that of keeping the gene pool well-stocked, the fear being that if an allele for some gene is not present in any member of the population, crossover will never be able to generate it and will thus not have access to the entire search space. This observation, which motivated the principle of ergodicity, suggests that the two problems mentioned can be tackled together by defining a mutation operator which inserts only extremal values into the gene pool, thus countering the bias of X^F. As before, given n genes per chromosome (now real-valued), a set of n point mutation operators are defined according to

$$M_i^R : \mathcal{C} \times \{\alpha_i, \beta_i\} \longrightarrow \mathcal{C},$$

[16] defined on a single gene

[17] known affectionately as "top hat", because of the graph of its distribution function

$$\text{with} \quad M_i^R(\eta_1\eta_2\ldots\eta_n, a) = \eta_1\eta_2\ldots\eta_{i-1}a\eta_{i+1}\ldots\eta_n.$$

The difference between this and standard mutation is that instead of using the interval $[\alpha_i, \beta_i]$ as the control set $\mathcal{A}_i$, only the end-points α_i and β_i are now used. If both parents are selected according to fitness, such mutations should be applied *before* crossing over, in order to reduce the probability of generating a child of very low fitness which then fails to reproduce.

In order to test these ideas, De Jong's standard test suite of functions (De Jong (1980)), described in tables 1 & 2 were examined using both a standard binary representation with uniform crossover, and a real-valued representation using flat crossover as defined above.

function	domain	resn.	description
$f_1(x) = \sum_{i=1}^{3} x_i^2$	$\lvert x_i\rvert \leq 5.12$	0.01	3-D parabola
$f_2(x) = 100(x_1^2 - x_2)^2 - (1 - x_1)^2$	$\lvert x_i\rvert \leq 2.048$	0.001	Rosenbrock's saddle
$f_3(x) = \sum_{i=1}^{5} \lfloor x_i \rfloor$	$\lvert x_i\rvert \leq 5.12$	0.01	5-D step function
$f_4(x) = \sum_{i=1}^{30} x_i^4 + R, \quad R \sim N(0,1)$	$\lvert x_i\rvert \leq 1.28$	0.01	Quartic with Gaussian noise
$f_5(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]$	$\lvert x_i\rvert \leq 65.536$	0.001	Shekel's Foxholes

Table 1: De Jong's Test Functions

The positions of the foxholes for f_5 (the a_{ij}) are traditionally taken to lie in a square lattice with spacing 16, centred on the origin. In the tests presented, f_6 is the same as De Jong's f_5 except that the coordinates of the foxholes were chosen at random over $[-65.536, 65.536)$ at the start of each run. Although the domains are invariably quoted as closed intervals $[-a, a]$, it is in fact half-open intervals $[-a, a)$ which are always used to give a number of points in the search space which is a power of 2.

Of the five functions, good performance might reasonably be expected on f_1, f_2 and f_4, which are (essentially) smooth, whereas very poor performance would be expected on f_5. Reasonable performance might also be anticipated on f_3, which while not smooth, is reasonably local in nature. The results for off-line[18] and best-seen performance are shown in the graphs 1–5. An extra function, f_6 is also included,

[18] "Off-line" performance is running average of the best result seen so far: this contrasts with "on-line" performance which is the average of all evaluations so far. The nomenclature is due to De Jong (1980).

fn.	dim	space size	description
f_1	3	1.0×10^{9}	parabola
f_2	2	1.7×10^{6}	Rosenbrock's Saddle
f_3	5	1.0×10^{15}	step function
f_4	30	1.0×10^{72}	noisy quadratic
f_5	2	1.7×10^{10}	Shekels foxholes
f_6	2	1.7×10^{10}	Random foxholes

Table 2: De Jong's functions (description)

which is a variation on Shekel's foxholes in which the positions of the foxholes are random, rather than in a regular grid.

A comparison is shown between the same genetic algorithm using binary and real representations, with parameters selected to give good performance with binary representations. Following Shaffer *et al* (1989), the point mutation rate was made inversely proportional to the chromosome length, and was thus higher when using real representations (with fewer genes) than for their binary counterparts. The Stochastic Universal Sampling procedure of Baker (1987) and rank-based selection, broadly *à la* Baker (1985) were used. Flat cross-over and extremal mutations, as described above, were used for the real-valued case, and uniform crossover was used for the binary trials. The results are all averages over 100 runs.

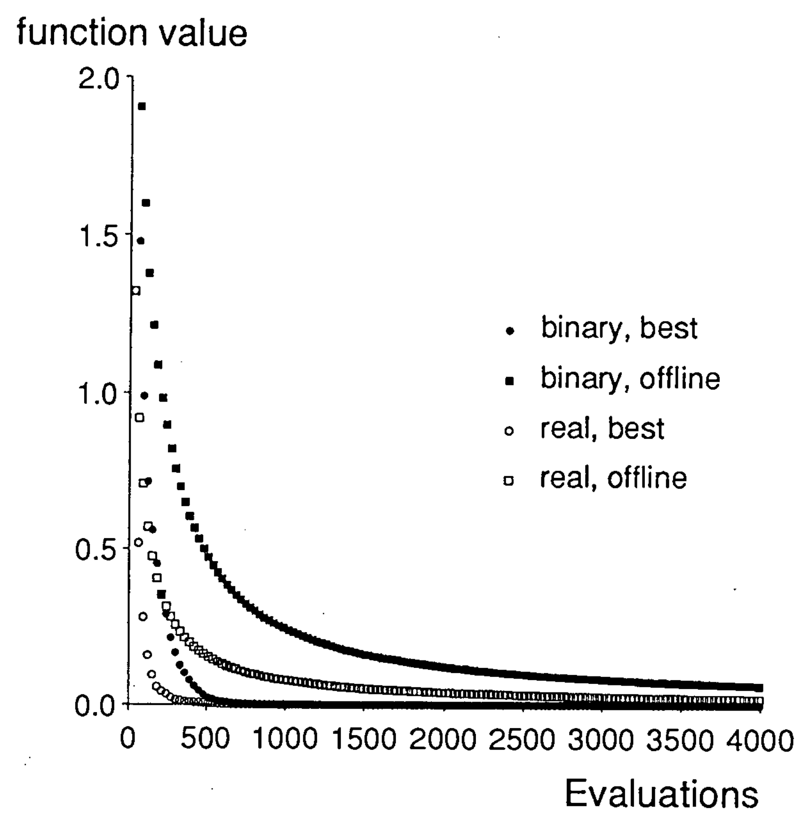
As predicted, flat-crossover with real genes performs extremely well on the smooth f_1, f_2 and f_4, out-performing binary representations. On f_3, although less effective than the binary case, the global optimum is still consistently found in reasonable time.

The results for Shekel's foxholes are rather more surprising. With the standard foxhole configuration, (a five-by-five grid with spacing 16) the binary representation appears superior, though the real representation performs amazingly well considering that the crossover operator it uses was only designed to respect *locality* formae, which have no obvious relevance to this problem. Notice, however, that points differing by 16.384 are very close in Hamming Distance under the binary representation, making it easy to hop from one foxhole to another. For this reason, a second set of trials was performed using fox-hole coordinates each chosen at random. In this case, the real representation using flat crossover gives slightly superior performance to the binary representation.

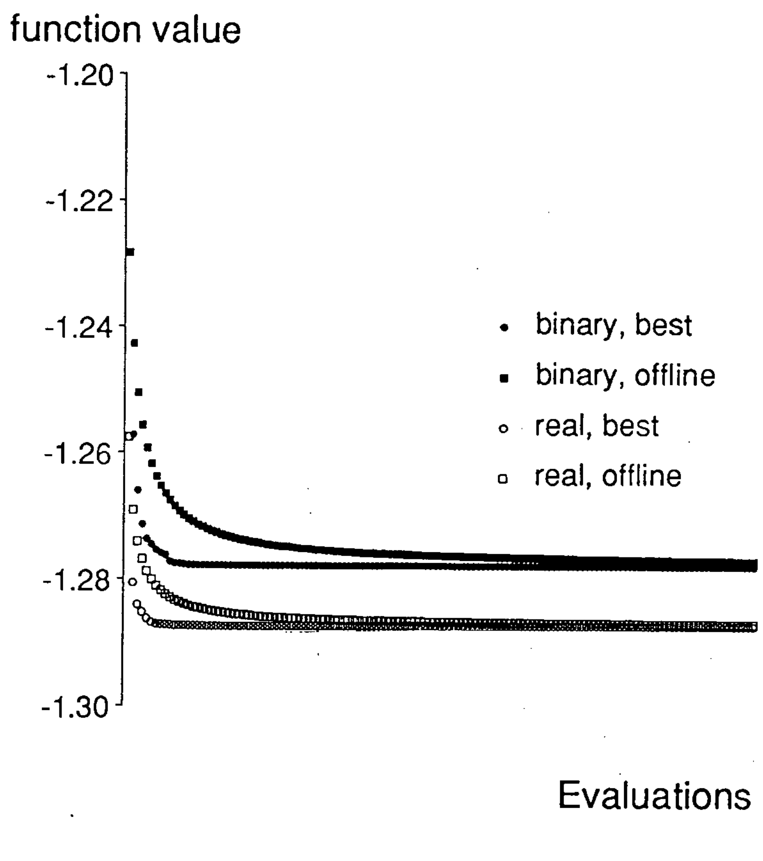
6·2 Periodicity

Dealing with general periodicities, unsurprisingly, is harder. Constructing equivalence relations Ψ^P capable of capturing general periodicities is not difficult: suitable relations are specified by a position p, a

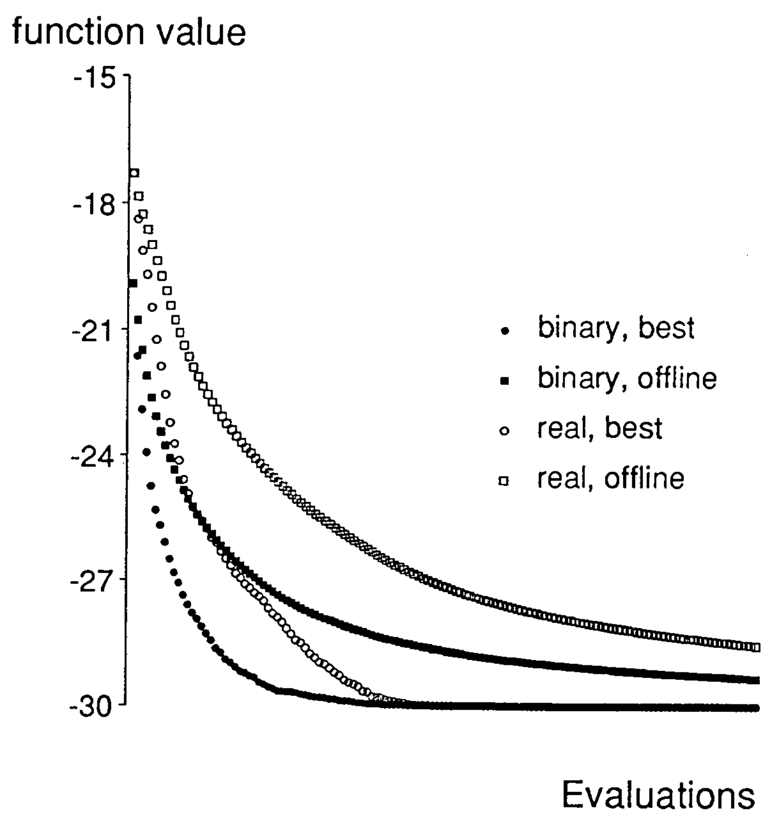
De Jong's f1



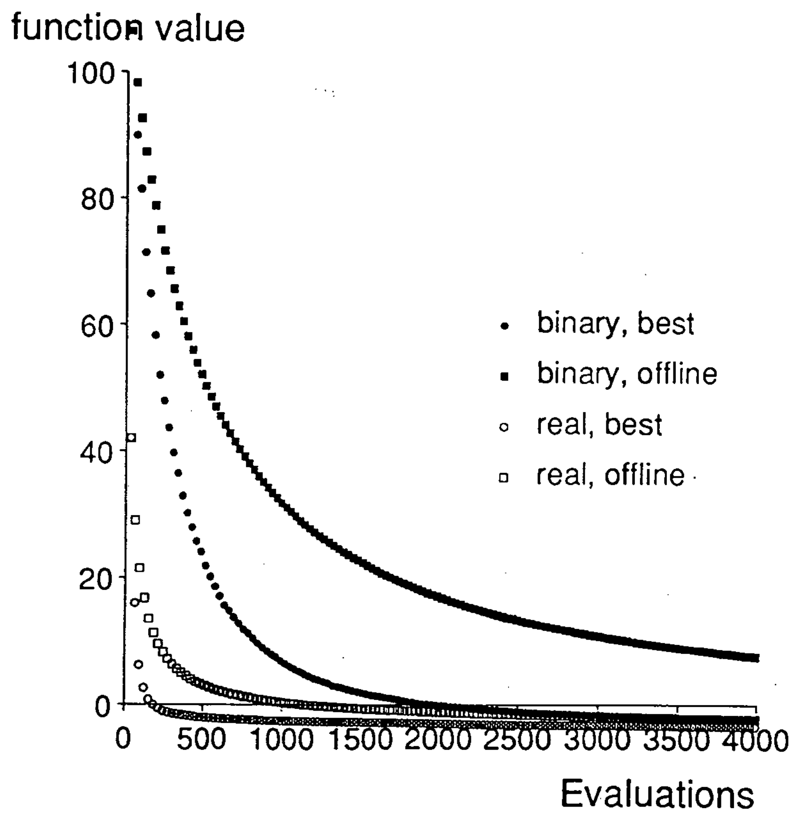
De Jong's f2



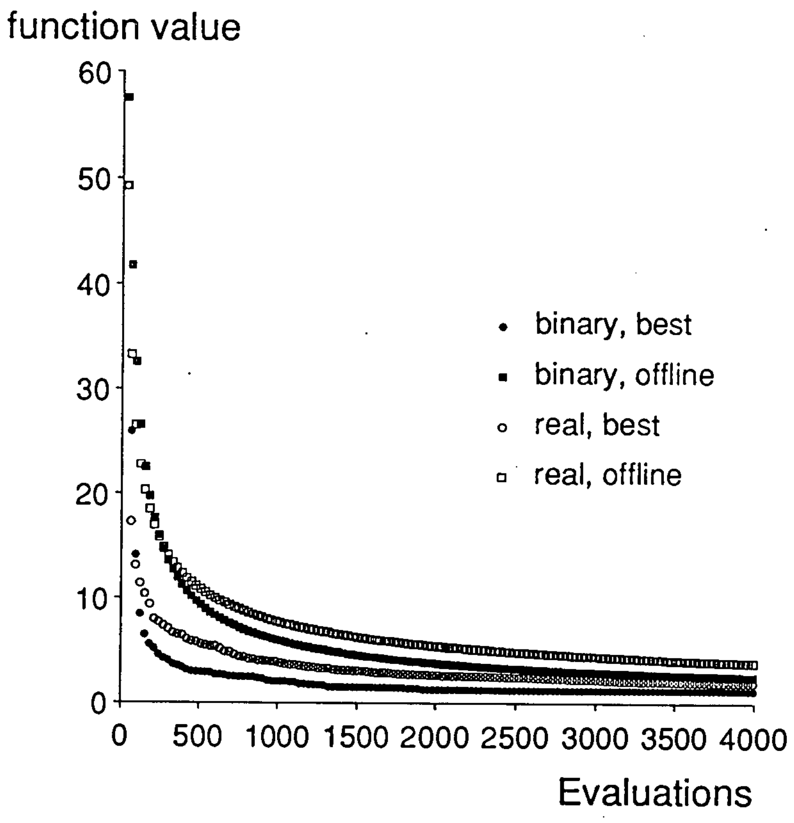
De Jong's f3



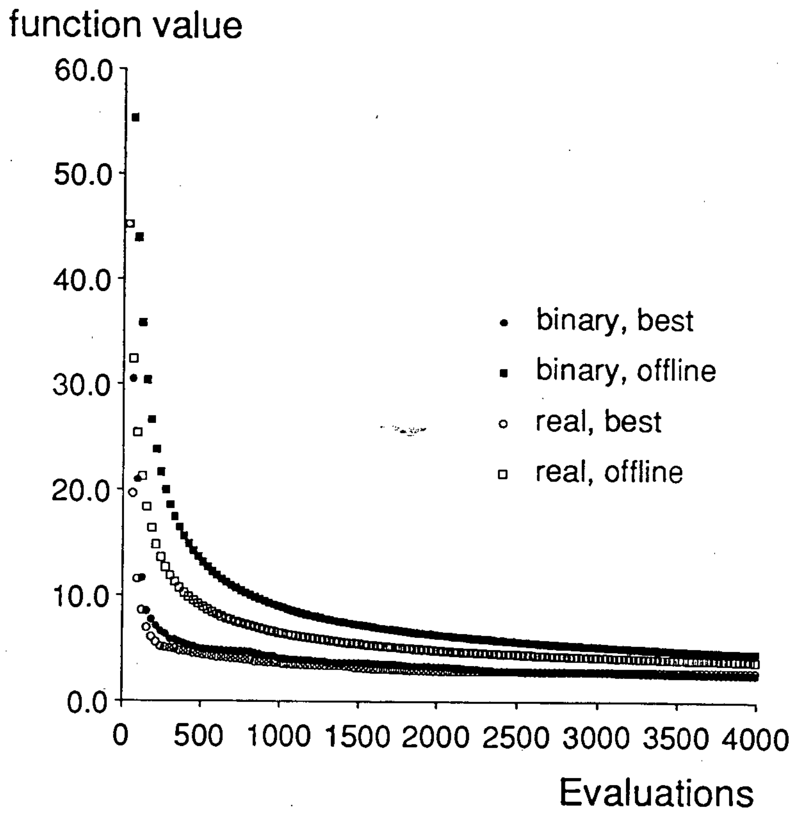
De Jong's f4



De Jong's f5



Random Foxholes (f6)



radius r (to allow for fuzziness) and a period T which is an integral multiple of r. Given these, and again simplifying to functions of one real variable, two chromosomes are equivalent if they lie in intervals of radius r centred about points separated by a multiple of the period T. Formally,

$$\eta \sim \eta' \iff \left(\exists\, k, k' \in \mathbf{Z} : \ \eta \in \mathrm{B}[p + kT, r) \text{ and } \eta' \in \mathrm{B}[p + k'T, r) \right),$$

These equivalence relations are extremely flexible, subsuming the previous "locality" relations immediately by setting T to zero. If a crossover operator could be constructed which both respected and properly assorted these relations it might be expected that an extremely powerful algorithm for real-valued problems would result.

Sadly, no such operator exists. To see this, consider the formae ξ_1 to ξ_4 in the table, each with radius $r = 0.5$:

ξ_1	0		5		10		15		20
ξ_2	0	4		8		12		16	20
ξ_3	0								20
ξ_4		4			10			16	

The numbers above indicate the centres of the intervals which the formae comprise, so that ξ_3 consists of $\mathrm{B}[0, 0.5)$ and $\mathrm{B}[20, 0.5)$. Notice that $\xi_3 = \xi_1 \cap \xi_2 \neq \emptyset$ so that $\xi_1 \bowtie \xi_2$. Consider chromosomes $10 \in \xi_1 \cap \xi_4$ and $4 \in \xi_2 \cap \xi_4$. If a crossover operator X^P is to respect ξ_4 then it must be the case that for all $a \in \mathcal{A}_P$: $X^P(4, 10, a) \in \xi_4$; that is, all possible children of 4 and 10 must be members of ξ_4. If it is to assort ξ_1 and ξ_2 properly then there must be some $a' \in \mathcal{A}_P$ for which $X(4, 10, a') \in \xi_1 \cap \xi_2$, that is, it must be possible to cross 4 and 10 to produce a chromosome which instantiates both ξ_1 and ξ_2. These conditions are incompatible, however, because $\xi_4 \cap \xi_1 \cap \xi_2 = \emptyset$.

It should be emphasised that this is not a failure of the forma analysis, which has simply revealed that general periodicities are extremely hard for a genetic algorithm to be sensitive to. It has been demonstrated that no recombination operator can both fully respect and properly assort the formae Ξ^P induced by Ψ^P, but it is quite possible for an operator partially to respect and assort them. Indeed, uniform crossover does this. Whether an operator can be constructed which better respects and/or assorts Ξ^P remains an open question.

7 Recombination Operators and Formae

The previous section showed one application of a random, respectful recombination operator in the context real-valued genes for function optimisation, and this was mildly encouraging, but while looking at individual cases can be useful, the greater goal is to develop *general* insights, methods and results. In order better to assess the usefulness of random, respectful recombination operators it will be instructive—if slightly tedious—to examine some standard recombination and see whether they are random, respectful operators in the sense defined previously.

7·1 k-ary Chromosomes

The comparison between conventional crossover operators and random, respectful operators is complicated by issues of linkage. It has already been suggested that in the absence of linkage information and inversion, one- or two-point crossover is a strange choice of operator, and that uniform crossover is a more appropriate choice. Syswerda (1989) provides supporting evidence for this, though the work in chapter 6 suggests that the arguments are not as straight-forward as might naïvely be supposed. With linkage information and inversion, one-point crossover can be seen to respect, but not properly assort schemata, but its lack of proper assortment is far from accidental: the purpose of linkage information is precisely to bias crossover towards producing children which inherit sets of co-adapted genes with higher probability than other combinations. For these reasons the comparison is made with uniform crossover, and this choice will be partially vindicated by evidence in chapter 6 that this is the better choice of operator in any event.

A formal definition of uniform crossover is as follows:

$$X^u : \mathcal{C} \times \mathcal{C} \times \{0,1\}^n \longrightarrow \mathcal{C}$$

$$\text{with} \qquad X^u_i(\eta, \zeta, a) = \begin{cases} \eta_i, & \text{if } a_i = 0, \\ \zeta_i, & \text{otherwise,} \end{cases}$$

the set of n-bit binary masks being the control set $\mathcal{A}_u$. Plainly this operator respects schemata, since membership of $\eta \oplus \eta'$ merely requires that wherever the two chromosomes share a gene, their progeny share it also: since a child inherits every gene from one or other of its parents, this is automatic. Moreover, uniform crossover properly assorts schemata, for compatible schemata cannot disagree on the value of any gene. This being so, if $\eta \in \xi$ and $\eta' \in \xi'$, the binary mask a defined by

$$a_i = \begin{cases} 0, & \text{if } \xi_i \neq \Box, \\ 1, & \text{otherwise,} \end{cases}$$

ensures that $X^u(\eta, \eta', a) \in \xi \cap \xi'$.

Thus uniform crossover fully respects and proper assorts schemata over chromosomes with k-ary genes, but it is clear that it is not, for general k, a random, respectful recombination operator. Interestingly, however, in the best-studied and most widely-advocated case, that of binary genes ($k = 2$) uniform crossover *does* reduce to a random, respectful operator. (To see this it is sufficient to note that if some gene differs in the two parents it must be 1 in one and 0 in the other, so that making a random choice between the parents is equivalent to making a random choice from the binary allele set for that gene.)

This immediately raises a plethora of questions: how does random, respectful recombination differ from uniform crossover for higher k? Is the reason for the greater general success of binary representations over those using higher k explicable in these terms? Equivalently, which is better for high k—uniform crossover or random, respectful recombination? Are schemata the appropriate tools for analysing k-ary representations? Given that use of a different set of formae leads to a different random, respectful recombination operator, do other formae exist under which random, respectful recombination reduces to uniform crossover?

Not all of these questions will be tackled; in particular, no attempt will be made at this point to conduct empirical studies into the strengths and weaknesses of random, respectful recombination. In part, this is because the purpose of introducing this class of operators was largely for their theoretical interest, rather than their practical use, but also because it will probably be more productive to continue the quest for patterns that are independent of representation.

Analysing recombination with respect to any set of formae for an unspecified problem is in many ways the very antithesis of the approach that has thus far been advocated; the constant theme has been the idea that the crucial factor in determining the effectiveness of a reproductive plan is the interplay between the representation, the regularities (correlations in the search space) and the genetic operators used. It is at very least sensible, therefore, to consider the circumstances in which a k-ary representation might be appropriate before proceeding to examine other possible equivalence relations and formae which might be appropriate. In the following, k is taken to be the same for each gene, but this is merely to simplify the description: the analysis holds equally well if k depends upon locus.

An obvious case to consider arises when the "parameters" describing the structure in the real search space not sensibly divisible below k: for example, if a traffic light can take on colours red, amber or green it would seem positively perverse to try to introduce a binary coding. Part of this is because the coding would require at least two bits, which would leave a value without an obvious interpretation; more fundamentally, however, decomposing colours in this context seems to make little sense.

Another case arises when the parameters take on an ordered range of values. One of the curious features of standard crossover is that it effectively discards the ordering information present, since the algorithm is (statistically) invariant under a permutation of the allele labels. The "real-valued" genes in the last section are an example of parameters taking k ordered values since these would almost invariably be implemented as fixed precision numbers. More generally, in any system where the numerical values assigned to alleles are not arbitrary but indicate relations between the objects, it might seem natural to code the parameters as k-ary variables.

These considerations suggest a number of kinds of formae. Certainly schemata as conventionally conceived might seem like the obvious kind to use to capture characteristics like redness. For example, using the coding 0 for red, 1 for amber and 2 for green, and assuming that the chromosome represents a set of traffic lights to be controlled (the fitness function perhaps combining expected number of crashes with rate of traffic flow) the chromosome configuration 0022 would indicate the first two lights at red and the second two at green. In terms of the real search space the equivalence classes are then partial specifications of the entire configuration such as *red*, □, *green*, □.

Following from the "locality" formae of the previous section, a more obvious kind to use when considering genes representing numerical quantities might be ranges such as 2–4. Such formae could be represented thus

$$1 \begin{pmatrix} 1 \\ \downarrow \\ 4 \end{pmatrix} \Box \begin{pmatrix} 2 \\ \downarrow \\ 3 \end{pmatrix},$$

with the natural interpretation.

Finally, a more general kind of forma still might involve allowing an arbitrary set of alleles to be specified at any locus thus:

$$1 \begin{Bmatrix} 1 \\ 3 \\ 5 \end{Bmatrix} \Box \begin{Bmatrix} 2 \\ 3 \end{Bmatrix},$$

with the obvious interpretation that where the component of the forma is a set, membership of the forma requires that the gene at the corresponding locus in the chromosome is a member of that set. These proposed sets of formae both close under intersection (when compatible), as required by principle 3. The latter set, however, is not separable. This should come as no particular surprise, since every partition of the space of chromosomes is included in it. Interestingly, however, if rather than allowing each component of the forma to take on up to k values, a maximum of two is imposed a separable set of formae does result. This set is sufficiently interesting to deserve more careful definition. The allele sets $\mathcal{G}_i$ can all be taken without loss of generality to be $\mathbb{Z}_k$. Then

$$\Xi = \mathcal{G}'_1 \times \mathcal{G}'_2 \times \ldots \times \mathcal{G}'_n$$
$$\text{where} \qquad \mathcal{G}'_i = \mathbb{Z}_k \cup \mathbb{Z}_k^2 \cup \{\Box\}.$$

In other words, each component of these formae is either a single allele, a pair of alleles or a "don't care" symbol. The interpretation is the obvious one, that to match a single allele the gene in a chromosome must take that value, to match a pair of alleles the gene must take one of the two allelic values, and that any gene matches □. It is easy to verify that the formae are separable. The reason that they are of particular interest is that under them random, respectful recombination reduces to uniform crossover. There are a number of ways to see this, perhaps the simplest being to note that the similarity set of a pair of chromosomes can always be expressed as:

$$\eta \oplus \zeta = \left\{ \begin{matrix} \eta_1 \\ \zeta_1 \end{matrix} \right\} \left\{ \begin{matrix} \eta_2 \\ \zeta_n \end{matrix} \right\} \cdots \left\{ \begin{matrix} \eta_n \\ \zeta_n \end{matrix} \right\},$$

and random respectful recombination simply makes an arbitrary choice at each locus, in exactly the same way as uniform crossover. The range formae work in a manner very similar to the locality formae of the previous section, and merit little further discussion.

8 Outlook

Some progress has been made in this chapter towards the goal of generalising the analysis of genetic algorithms to non-string representations. This permits a shift of emphasis away from finding a representation within which schemata make sense, and allows genetic operators to be thought of as manipulating the structures in S directly. The prospect of a "free lunch", in the form of a representation which makes search easy *regardless* of the nature of the correlations in the underlying space has receded, but in its place there another appealing prospect, that of designing operators which manipulate equivalence classes (formae) of general sets of equivalence relations in useful ways. Inevitably, attention is now focused upon the regularities in the particular problem to hand, and—as always—it will be the case that the more that is known about the problem, the more suitable will be the formae that will be able to be constructed, and the more successful is the genetic search likely to be.

Of course, this work has raised many more questions than it has answered. One concerns the characterisation of separable formae—those capable of being simultaneously respected and properly assorted under recombination. Another concerns the development of more powerful representation-independent operators, ones which like random, respectful recombination can be used with any set of formae, but which will presumably be more powerful than this rather simple exemplar.

Another idea which has considerable appeal is to try to introduce what might be called *basic formae,* which would be defined with respect to an arbitrary set of formae in much the same way that basic vectors are defined in (arbitrary) linear spaces. A basis might consist of a set of low-precision formae from which all higher precision formae could be constructed by intersection. Equipped with these it would be possible to embark on a more careful analysis of the way in which formae can be—and perhaps should ideally be—recombined. It is not ridiculous to begin to see emerging ideas such as *fair assortment,* which would specify that recombination not merely be capable of constructing members of the intersections of compatible formae given suitable parents, but did so with probabilities prescribed by their precision and imputed utility. The possibilities at this stage seem boundless.

9 Summary

Intrinsic parallelism, the key concept under-pinning genetic search, has been shown not to be restricted to k-ary string representations. Given a suitable set of equivalence relations and a crossover operator which both respects and properly assorts its equivalence classes (formae) without excessive disruption, any genetic algorithm should exhibit intrinsic parallelism. These ideas have been applied to standard crossover operators to provide another insight into the apparent superiority of the uniform crossover operator over traditional 1- and 2-point crossover, and to apply genetic algorithms more effectively to some real-valued problems. They could equally well be applied to other problems for which k-ary string representations and schemata are not obviously appropriate.

No amount of physical acumen suffices
to justify a meaningless string of symbols.
— N. G. VAN KAMPEN, Stochastic Processes in Physics and Chemistry (1981)

A musician would be horrified if his art were to be summed up as
'a lot of tadpoles drawn on a row of line';
but that's all the untrained eye can see in a page of sheet music
The grandeur, the agony, the flights of lyricism and
discords of despair: to discern them among the tadpoles is no mean task.
They are present, but only in coded form.
In the same way, the symbolism of mathematics is merely its coded form,
not its substance. It too has its grandeur, agony,
and flights of lyricism.
However, there is a difference. Even a casual listener can
enjoy a piece of music.
It is only the performers who are required to understand
the antics of the tadpoles
— IAN STEWART, The Problems of Mathematics (1987)

6

Linkage and Convergence

Part I: Linkage

Neural networks are known to be highly non-linear systems. In some cases, connections can be deleted with hardly noticeable consequences, leading to the notion of graceful degradation of performance and a distributed information base. Equally, apparently very similar networks can display wildly differing performances, while very dissimilar networks[19] can implement almost identical functions. It therefore seems very likely that when using a genetic algorithm to select good networks, the phenomenon of *epistasis*—whereby the effect of a gene or some group of genes is very strongly dependent on which other alleles are present on the chromosome—will be prevalent.

The mechanism which Holland (1975) appealed to in the hope of ameliorating epistatic effects was *linkage*, described briefly in chapter 3. Recalling that tightly-linked genes are those which have a tendency to be carried over together under recombination, it can be seen that genes which are close together on a conventional chromosome are more tightly linked than those which are further apart. The inversion operator was introduced to allow reproductive plans to subject the linkage of the chromosome to adaptation in much the same way that the gene values are adapted. Proposals to extend this scheme through the introduction of permutation crossover operators such as PMX (Goldberg & Lingle (1985)) were also discussed.

The intention with all of these schemes is that co-adapted sets of alleles will come to be grouped together by the selective advantage that their chance grouping will confer on them. This relative selective advantage would be expected to be especially significant in the case of "mutually epistatic" sets of genes, which are more sensitive to disruption than others.

Curiously, very few workers appear to use linkage in their implementations. This is in part because the most common exposition of inversion (in terms simply of excising, reversing and replacing a portion of the chromosome) seems to make little sense:[20] except in very special problems, there is no reason to suppose that an allele which is useful at one locus will be useful at another, even if the allele sets for the various loci are the same so that moving the allele is legal. The confusion is perhaps increased by the fact that in the particular case of permutation problems such as the TSP, inversion *is* used in the "naïve" way, simply to reverse the order in which a contiguous portion of the chromosome, because in these special cases this plainly is a reasonably thing to do. In such cases, however, inversion is primarily playing the rôle of a mutation operator, rather than re-linking chromosomes.

There are, however, many people who do understand the rôle of linkage, and of inversion as its manipulator in genetic algorithms, who choose not to make use of it. This is curious because there seems to be very

---

19 that is, even after permutation of hidden units and so forth

20 to the author's personal knowledge, this is a misunderstanding of the rôle of inversion which many people have.

little reported work which suggests that it is not useful. Such work as the author is aware of is now discussed.

1 Survey

The bulk of the work on linkage appears to have been performed between 1967 and 1972 by Holland's students, the three principal studies being the doctoral theses of Bagley (1967), Cavicchio (1970) and Franz (1970).[21] Of these, only Franz's considered inversion in the form suggested by Holland (1975). Although his research failed to show any positive effect of using inversion, this has been attributed by both Bethke (1981) and Goldberg (1989) (citing Bethke) to the relative simplicity of the problems studied by Franz.

While various permutation crossover operators have now been introduced, only Goldberg & Lingle (1985) seem to have suggested using these to manipulate linkage information, and surprisingly they presented no results of performing such linkage manipulation themselves. The only other work of which the author is aware which uses inversion in anything like the sense described by Holland is that of Whitley (1987), which is now discussed.

2 Whitley's Cards and "Inversion"

Whitley (1987) investigated using genetic algorithms for the interesting, if artificial, problem of searching for winning 5-card poker hands. In this work he allowed "five of a kind" and gave no credit for straights or flushes so that, in effect, credit was only assigned for multiple collections of a single value, with special cases for "full houses" and "two pairs". Although not stated explicitly in the paper, it is not clear from his discussion that suits were not used either. This is an interesting problem because it presents another kind of generic chromosome to consider—one which represents an unordered *collection* of objects. (The term "set" will not be used because the convention is that repetition is of no consequence in a set, so that $\{a, a, b\} \equiv \{a, b\}$.)

It is worth digressing and asking what formae might be introduced to analyse this sort of problem. It is desirable that the formae do not distinguish between two chromosomes merely because they present the elements in a collection in a different order. The natural forma to use (or at least, the natural extension of schemata to this case) specifies a sub-collection of elements which the chromosome must contain in order to instantiate the forma. Thus, taking Whitley's cards as an example, the chromosome 54437 would be an instance of the forma 347□□, among others, where the convention adopted is that that defining positions

[21] The discussion of work before 1985 draws heavily on a discussion on (pp. 167–170) in Goldberg (1989). This is largely because this is the only work which references the pre-1985 studies in the context of linkage of which the author is aware, and for this reason he became aware of these early studies only recently.

are filled from the left, and follow the numerical order of the cards. This would then suggest that before performing a conventional cross between two chromosomes, one of them should be re-ordered in such a way that any shared elements are aligned: only in this way will crossover respect the formae.

Whitley (in the absence of the formalism of formae, of course) chose a different path, performing crossover as usual but using the "naïve" inversion operator also. In this case using naïve inversion is quite acceptable because the location of an allele on the chromosome does not affect its phenotypic expression. Whitley's experiments show that using levels of inversion around 20–30% improves performance significantly over just using crossover, though it should be noted that these results are in the absence of a mutation operator. His conclusions are as follows (Whitley (1987)):

> The results of the experiments indicate that where real valued[22] features are used on problems where ordering does not affect the value of [chromosomes][23] inversion does more than change the linkage of features. It provides a type of non-destructive noise that helps crossover to escape local maxima. Compared to tests using crossover alone, the search using inversion was consistently more efficient; optimal and near optimal solutions were found faster and more high-valued combinations of features were found.

He goes on to say that

> [o]ne of the problems of using only crossover is that each position in the list of features is isolated. Its erratic performance may be due to the fact that crossover cannot exploit all the information in the pool[24] and may not be able to get at certain segments of information that develop in the hands. If the pool is viewed as a two dimensional array with each row being a hand and the features appearing in the columns, it means that information in one column cannot be accessed in another column. For example, it is possible that no ace card appears in the search pool as the first card of a hand.[25] Such a coincidence makes it impossible to find the optimal solution when only crossover is used. ... Inversion "stirs up" the columns and thereby "stirs up" the available information.

A number of points follow from this. First, Whitley calls inversion (as he uses it) "non-destructive" because the real structure in $\mathcal{S}$ (an unordered collection of cards) to which the chromosome corresponds is unaffected by it. Under inversion which only affected linkage, however, it would be the case that if two chromosomes were the same apart from their linkage (and thus corresponded to the same structure in $\mathcal{S}$) then all of their possible offspring under (conventional) crossover would also correspond to that structure. This is not true under Whitley's régime, however, for

$$\mathcal{X}(11122, 22111, 4) = 11111,$$

[22] Whitley's use of the term "real valued" does not seem to mean real in the sense of $\mathbb{R}$ rather than $\mathbb{N}$ since his examples all use integer values.

[23] Whitley uses the term "schemata" here, having earlier explained that '[t]he term schema will refer to any information structure to which genetic operators apply'—a highly non-standard use of the term.

[24] Whitley uses the term "pool" to refer to the set of all alleles in the population at *any* locus.

[25] though it would simple, as Holland (1975) (p. 110) pointed out, to remedy this since k suitable initial strings suffice to ensure that every locus has at least one copy of every allele for k-ary genes.

which is plainly a different structure. In one sense this is exactly what he is aiming for and validates his point that (his) inversion does more than alter linkage, but the example also shows an effect which Whitley does not describe. For having eliminated runs and flushes, the good solutions are all sets of common values: any "n of a kind" beats any "$n - 1$ of a kind" and so forth, subject to exception rules for a "full house" or "two-pairs". But these are exactly the kinds of solutions that inversion will tend to generate. Assume that aces are "high" (which in fact they were in Whitley's example). Then even if the entire population began with the single chromosome $A2222$ (up to permutations) where A represents an ace, under Whitley's scheme it would be possible, if unlikely, to produce the optimum after only three crosses and three inversions:

$$\left.\begin{array}{c} A2222 \\ 2A222 \end{array}\right\} \xmapsto{X(\ldots,2)} \left.\begin{array}{c} AA222 \\ 22AA2 \end{array}\right\} \xmapsto{X(\ldots,3)} \left.\begin{array}{c} AAAA2 \\ 2AAAA \end{array}\right\} \xmapsto{X(\ldots,5)} AAAAA,$$

where in each case the lower chromosome is obtained from the upper one (which results from the cross) by inversion. While this is wholly desirable in the particular case of searching for solutions like $AAAAA$, this property would *not* hold for a general collection of optima, or indeed for the real poker-hand optimisation where the optimal solution is a royal flush (10-Jack-Queen-King-Ace, all of the same suit). Indeed, in reality each card is unique so that while "naïve" inversion is still permissible the kind of crossover required is more akin to a permutation recombination than a conventional crossover. Whitley's version of inversion might still be useful, but it is hard to see that it would produce the dramatic improvements seen in his simpler experiments.

Having said this, Whitley's strategy (preferably with the addition of mutation to keep the gene pool properly stocked) might be a good way to tackle the cards problem as formulated, even with different optima. The point of this discussion was to examine his analysis, to point out that his test problem was particularly amenable to his approach (in the sense that transfer of information from one locus to another is a very efficient way of generating good solutions given the particular characteristics of the optima) and to provide another interesting application of forma analysis.

It might further be pointed out that in the particular case of searching for "n of a kind", standard crossover with alignment to ensure respect of the formae introduced above, would be very unhelpful in the absence of other operators because it would never be able to generate a hand containing more aces than the hand in the initial population with the most aces. The introduction of standard mutation would remedy this problem, but to deal with this particular short-coming allowing the possibility of copying an allele (with low probability) from one locus to another would suffice and perhaps be better. Interestingly, random, respectful recombination (defined with respect to these formae) does not suffer from this limitation since having aligned any common alleles, it would be free to place any values at the remaining loci.

This concludes the survey of work on inversion and linkage.

3 Uniform Crossover

If no operator is to be used to re-link the chromosomes, as seems to be the case with the overwhelming majority of implementations of genetic algorithms, then the bias towards schemata whose definition points are clustered together on the chromosome seems positively bizarre, and it is not surprising that many workers in the field have explored alternatives to Holland's simple one-point crossover operator. The only justification for the bias in one-point crossover could be that the genes were laid down on the chromosome in a way which was known *a priori* to reflect the linkages between suitable parts of the solution. In order for this to be possible, the worker would need considerable insight into the structure of solutions the problem, something which cannot in general be assumed.

The simple generalisation due to De Jong (1975) which interprets the chromosome as a ring and allows the selection of two cross points has been discussed, but while this effectively reduces the definition lengths of some schemata, it does not remove the fundamental bias towards short schemata. A more obvious choice of crossover operator in the absence of linkage information is the so-called *uniform* crossover operator, discussed in chapters 3 & 5:

$$X^{(u)} : \mathcal{C} \times \mathcal{C} \times \{0,1\}^n \longrightarrow \mathcal{C}$$

$$\text{with} \quad X_i^{(u)}(\eta, \zeta, a) = \begin{cases} \eta_i, & \text{if } a_i = 0, \\ \zeta_i, & \text{otherwise,} \end{cases} \tag{1}$$

This operator is such an obvious choice that it is hard to know how to assign credit for its origin, but its properties are discussed by Syswerda (1989) and Eshelman *et al* (1989). Syswerda explicitly makes the connection between (lack of) linkage information and uniform crossover, writing:

> One operator that we might be able to dispense with is the often mentioned but seldom implemented inversion operator. Inversion changes the ordering of [genes][26] on the chromosome without changing their meaning. The hope is that inversion will, over time, move together co-adapted sets of [alleles] in order to protect them from disruption and to make it more likely that they will be crossed with similar groupings. In other words, inversion attempts to reduce the [disruption] rate and increase the combination rate of crossover. With uniform crossover, however, use of inversion is not necessary and will have no effect, since uniform crossover is completely indiscriminate in choosing [genes] to mask ... When using uniform crossover, [genes] which are far apart have the same chance of being masked simultaneously as [genes] close together.

Clearly it is true that if uniform crossover as described by equation (1) is used then inversion has no effect (since it acts only on the linkage information which is not used in this definition) but it is far from clear that uniform crossover plays a similar rôle to crossover with inversion, or that it will perform better, or that it is 'unnecessary'.

[26] This discussion was based on binary genes, but applies with equal validity to genes with arbitrary sets of alleles. The words in square brackets were all 'bit' or 'bits' in the Syswerda's paper with the exception of [disruption].

Under uniform crossover, the probability of generating a bit mask with r 1's is binomial provided that the control set $\mathcal{A}_u = [0, 1]$ is sampled appropriately. More precisely, letting N_1 be a random variable taking the number of 1's in a mask of length n, and letting $q = 1 - p$, where p is the probability that any bit in the mask is 1,

$$P(N_1 = r \mid X = X^{(u)}) = \binom{n}{r} p^r q^{n-r}. \tag{2}$$

For one-point crossover $X^{(1)}$ (and two-point crossover $X^{(2)}$) *with linkage* the cross point is usually chosen uniformly between 1 and $n - 1$ (inclusive), so that

$$P(N_1 = r \mid X = X^{(1)} \text{ or } X = X^{(2)}) = \frac{1}{n-1}. \tag{3}$$

Though this was suggested by Holland, and is the norm, the cross point could be chosen to be binomial for one- or two-point crossover also. An interesting comparison between uniform and traditional crossover operators can be made after a few definitions.

DEFINITION The *random re-linking operator*

$$L : \mathcal{P}_n \longrightarrow \mathcal{P}_n,$$

when applied to (the linkage information from) any chromosome of length n, is defined to the that which returns a randomly-selected permutation from $\mathcal{P}_n$. □

The following algorithm can then be defined:

DEFINITION A genetic algorithm which uses the one-point crossover operator $X^{(1)}$, with cross point r chosen from the binomial distribution (equation (2)), and in which the linkage information is chosen for the child by application of the random re-linking operator L, shall be said to use *Binomial Crossover with Random Linkage.* □

Using the definition of equivalence given below (which is the natural one) it is easy to show that such Binomial Crossover with Random Linkage is precisely equivalent to uniform crossover.

DEFINITION Two crossover operators are equivalent if the probability of generating any given child from any given pair of parents is equal under the two operators. □

THEOREM (Equivalence of uniform crossover and Binomial Crossover with Random Linkage)

The uniform crossover operator $X^{(u)}$ is equivalent to Binomial Crossover with Random Linkage.

Proof:

The distribution for the number of genes taken from the first parent, r, is in each case given by equation (2), by assumption. Under uniform crossover $X^{(u)}(\eta, \zeta, a)$, every gene from η is equally likely to be included in the r genes actually taken from η. Since the permutation used for $X^{(1)}(\eta, \pi^{\eta}, \zeta, r)$ is randomly selected by L, this is also the case under $X^{(1)}$. Thus every gene is transferred with the same probability under the two operators, and it follows that they produce each child with the same probability. QED

COROLLARY It follows trivially from the above that two-point crossover is also equivalent to uniform crossover provided that the separation of the cross points is chosen to be binomial and re-linking of the child is achieved using L. □

4 Preliminary Discussion

It has been shown that relatively minor alterations to Holland's original formulation produce an algorithm which is equivalent to one using uniform crossover. The only changes required are the substitution of a binomial choice of cross point for the conventional uniform choice, and the use of a much more disruptive operator than inversion (indeed, a completely disruptive operator) to re-link the chromosome. An obvious question, then, is whether Holland's original formulation is better or worse than the slightly modified version which is equivalent to the use of uniform crossover. Before answering this question it will be useful review the work of Eshelman *et al* (1989), in which a number of crossover operators are classified according to their "positional" and "distributional" bias.

Positional bias is the tendency of a genetic algorithm (or its crossover operator) to transfer groups of genes with different probabilities depending on their position on the chromosome. One-point crossover without inversion exhibits positional bias towards genes which are close together on the chromosome. $X^{(1)}$, together with inversion, attempts to exploit this bias by linking the chromosome in such a way that the sets of genes transferred interact with each other in useful ways. It is the positional bias of conventional one- and two-point crossover operators which places short schemata at an advantage relative to their longer counterparts of equal order.

Distributional bias is the tendency of a genetic algorithm (or crossover operator) to transfer some favoured amount of genetic material preferentially from one parent to the child. One-point crossover with a uniformly selected cross point exhibits no distributional bias because all possible amounts of genetic material are equally likely to be transferred (equation (3)). Uniform crossover with $p = 0.5$ (or one-point crossover with a binomially-selected cross point) on the other hand, biases the algorithm towards passing on roughly equal amounts of genetic material from the two parents to the child because the binomial coefficient $\binom{n}{\lfloor n/2 \rfloor} \gg \binom{n}{1}$.

The studies by Syswerda (1989) and Eshelman *et al* (1989) suggest strongly that uniform crossover is superior to one-point crossover in the absence of inversion, and it is generally accepted that two-point crossover is superior to one-point under these conditions. Syswerda also concluded that uniform crossover was generally superior to two-point crossover. Neither study, however, investigated inversion. (Eshelman *et al* also considered k-point shuffle crossover operators for arbitrary k. The shuffling corresponds to random re-linking by L, and the number of cross points biases the amount of genetic material transferred: as k approaches n, the chromosome length, the bias increases towards transferring half the genetic material.) The demonstration that uniform crossover can be regarded as one- or two-point crossover with binomial selection of cross points and random re-linking provides an interesting way to separate the differences between traditional and uniform crossover operators, and to test which effects are more significant.

5 The Genetic Algorithm

The five test functions originally investigated by De Jong (1980) seem to have become established as a reference set, and these have been used for the experiments detailed below. These were discussed in chapter 5. The particular genetic algorithm used for these experiments had the following characteristics. Synchronous ("generational") update was used with Baker's "Stochastic Universal Sampling" selection algorithm (Baker (1987)). Rank-based selection was used (broadly *à la* Baker (1985)), with the utility $\mu(\eta_k)$ of the kth-ranked chromosome being given by:

$$\mu(\eta_k) = \frac{p_s q_s^{k-1}}{1 - q_s^n},$$

where $p_s = 0.1$ and $q_s = 1 - p_s = 0.9$. (This fitness is equivalent to stepping down the list, fittest first, selecting the current element with probability p_s, the "selection probability".) Grefenstette's parameter settings were used for population size (30) and crossover probability (0.95) (Grefenstette (1986)) but the point mutation rate was made inversely proportional to the length of the chromosome with the constant of proportionality chosen such that an average of one mutation occurred per update. Scaling the mutation rate in this way is suggested by the work of Shaffer *et al* (1989), though they recommended a lower constant of proportionality.

Binary coding was used for functions f_1, f_3 and f_4, and Gray coding for f_2 and f_5. This choice was made because f_2 and f_5 sometimes required very large number of cycles to converge to the optimum, which resulted in very large error bars on those runs. (Caruna & Shaffer (1985) discuss the advantages of Gray coding for genetic algorithms.) Following Eshelman *et al* (1989) the total number of evaluations required to find the global optimum was recorded for all functions except f_4, where the total number of evaluations to reach a value below 2.0 was recorded (Eshelman (1990)). (The function f_4 has Gaussian noise, so the global optimum is not well-defined.)

Finally, and most importantly, the assignment of bits to positions on the chromosome was randomly chosen at the start of each run. This is useful for isolating the effect of linkage re-arrangement, since conventional ordering (whereby the bits for each parameter are grouped together, in order) may be close to optimal.

6 Results for Inversion

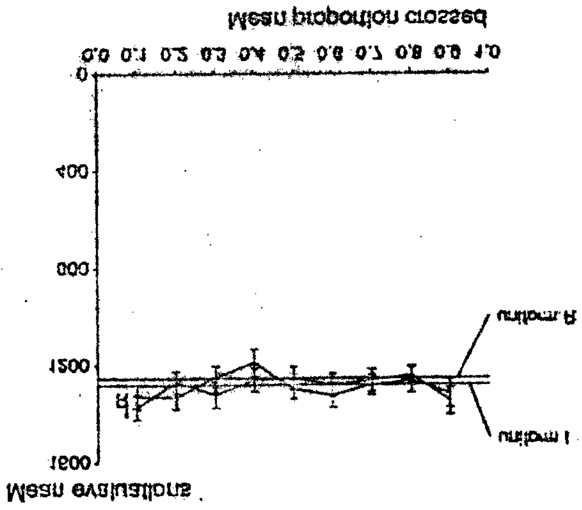
The first series of experiments attempts to determine the effectiveness of inversion in re-linking solutions in ways that enhance the search. Two point crossover was used, and the probability of performing an inversion varied from 0 to 1 in steps of 0.1. In all cases the cross point was chosen uniformly along the length of the chromosome, following equation (3).

The results of these experiments are shown in graphs 1–5. The results are all averages over 50 runs, with the exception of f_2 and f_5, where averages are over 400 runs because the error was consistently much higher for these functions than for the others. (As Shaffer *et al* (1989) noted, finding the global optimum is hard for f_2, but finding good solutions is very easy.) The error bars show the standard error.

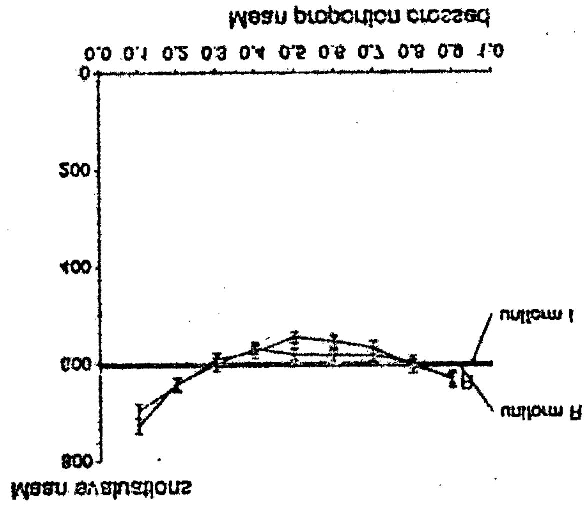
Although the data is quite noisy, the graphs show little evidence that inversion is playing any useful rôle in enhancing search, at least with respect to the time taken to achieve the optimum. Bearing in mind that the order of the bits is random, this is perhaps surprising. Runs were also performed (though are not shown) using the "natural" assignment of bits to loci on the chromosome, rather than choosing random loci, and these showed no perceptible difference from the results given. This seems to indicate that linkage is not important in these problems. Given this, the failure of inversion to enhance performance seems less surprising. Interestingly, these results contrast slightly with those of Syswerda (1989), who found that using the standard ordering for Shekal's foxholes (f_5), the traditional operators out-performed uniform crossover, whereas using an arbitrary ordering uniform crossover performed better. The effects he saw were very weak, and he used binary-coding, whereas Gray coding has been used here. It is not hard to believe that locus is more significant for binary coding than for Gray.

More generally, these experiments appear to fit into the pattern of research which searches for and fails to find linkage effects. A number of possible explanations come to mind for these. First, it could be that linkage is of no significance in the problems studied, or at least that the linkage between the genes is similar over all pairs. Secondly, it could be that the time-scale on which the linkage rearrangement is achieved is too long to show up in these results: only about fifty generations are seen here. Thirdly, it could be that the selection pressure is to weak to have any noticeable effect; it is the *utility* of the chromosome which determines reproductive success, so that there is no immediate benefit derived from good linkage. The selection pressures for achieving good linkages are thus very much lower than the corresponding pressures towards good solutions. In this context it would be interesting to investigate the use of *reproductive evaluation* (Whitley (1987)), a scheme whereby a parent's utility is modified to reflect that of its offspring (clearly a scheme well-suited to fine-grained rather than coarse-grained update).

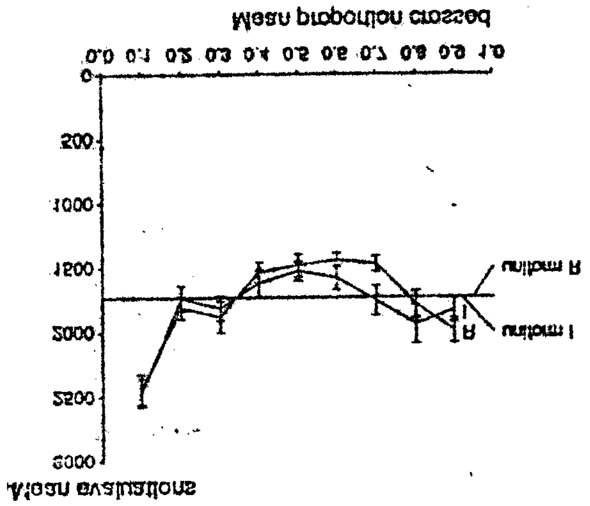
Graph 6: De Jong's f1



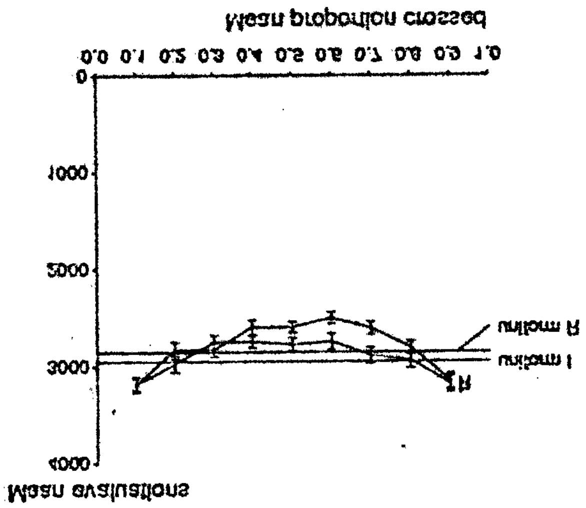
Graph 7: De Jong's f2



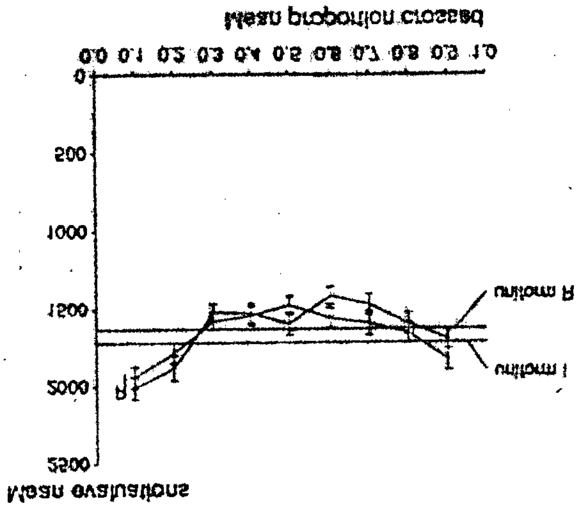
Graph 8: De Jong's f3



Graph 9: De Jong's f4



Graph 10: De Jong's f5



In fact, the paper in which Whitley discusses reproductive evaluation is the same one as discusses his inversion, but it has already been argued that it is hard to draw general conclusions about inversion as a re-linking operator from his trials.

A fourth possibility is that inversion, only a mutation operator in the context of linkage, is inadequate for searching the space of good solutions. In this context the various permutation recombination operators offer a way forward.

Finally, the possibility must be admitted that there is simply no effect to be seen. The reason for playing down this explanation in spite of a having provided no evidence of an effect is that it is very hard to understand how such an effect could be entirely absent, especially since inversion is known to exist in nature. The fundamental idea underlying adaptation is simply that the combination of a mechanism for changing structures and selective reproduction leads to "evolution". So powerful and all-pervasive an idea is this that it is difficult to accept that such a system can fail to evolve.

To expand on this point, Dawkins (1982) described the fundamental agents in evolution as "active, germ-line replicators". A replicator is any system which reproduces or is reproduced. For example, a piece of paper can be regarded as a replicator if it undergoes reproduction by photocopying. There is an implicit assumption that no process of replication is completely reliable so that changes in the form of errors will invariably occur from time to time. Dawkins then classifies all replicators according to two orthogonal criteria—active or passive, and germ-line or dead-end. Active replicators are those whose characteristics affect their likelihood of reproduction, as opposed to passive replicators which reproduce at the same rate regardless of their characteristics. It may seem at first that a piece of paper is a passive replicator, but Dawkins points out that its contents *do* affect the likelihood of a piece of paper's reproduction, for people are more likely to copy interesting papers than uninteresting ones. The distinction between germ-line and dead-end replicators is that germ-line replicators have no limit *in principle* to the number of generations they can spawn, whereas dead-end replicators cannot ever spawn an arbitrarily large number of generations. In other words, although the tree of descendents from a germ-line replicator may well terminate (because it or its descendents happen to fail to reproduce) it need not in principle, whereas the corresponding tree for a dead-end replicator is guaranteed to terminate.

The argument then goes that given active germ-line replicators, those which have characteristics which make them more likely to reproduce are bound, given time, to come to dominate numerically over those with characteristics which make them less likely to reproduce. Further, the infidelities in the copying process ensure that even if errors introduced almost always reduce the reproductive fitness of the replicator, occasionally chance improvements are bound to be made. Thus, given a sufficiently long time, evolution is bound to occur. The addition of recombination only enhances these arguments provided that the recombination increases the probability of producing good (fit) offspring.

This potent argument generates some reluctance to sacrifice the notion of usefully adapting linkage in genetic algorithms, though it is not hard to believe that the time-scales involved make its use impractical. For while it may not be immediately obvious, linkage information is an active germ-line replicator, because although re-linking a chromosome does not affect its utility, it does affect the (average) utility of its offspring under crossover. If the problem is that applying a mutation operator to linkage (in the form of inversion) is inadequate for good search, the way forward may be provided by permutation crossover operators.

7 Results for Cross Points

In this set of experiments the first cross point was again chosen uniformly along the length of the chromosome but the separation of the cross points was chosen using the Gaussian approximation to a Binomial distribution with p varying from 0.1 to 0.9. The experiments were repeated using both inversion, applied with probability half, and the random re-linking operator, L, applied with probability 1. (Clearly the rate at which inversion is applied will be of little consequence, given the earlier experiments.) Recall that use of L together with binomial cross point is exactly equivalent to uniform crossover with the probability of each bit of the mask taking the value 1 set to p.

Again, test results are averages over 400 runs for f_2 and f_5, and 50 for the other functions, and are shown in graphs 6–10. For reference, horizontal lines are shown indicating the performance when the separation of the cross points is chosen to be uniform, again using both I and L. Although error bars are not shown on these, the results are averages over 400 runs, and are small.

The pattern here is very striking, at least for functions f_2 to f_5, and strongly indicates that there is a small, but definite benefit in transferring roughly half the genetic material from each parent.

8 Summary

This work, in common with other work on inversion, has failed to demonstrate any useful effect of using inversion to re-link chromosomes. The author suggests that the explanation for this is that linkage does not vary much between the pairs of genes in these problems, and that inversion is too weak an operator to make search effective in reasonable time given the relatively small selection pressures available to it.

Uniform crossover with $p = 0.5$ has been shown to perform significantly better than two-point crossover (with inversion). The differences between these two crossover operators can be characterised as the greater distributional bias of uniform crossover, and its lower positional bias. The bulk of the difference in performance is accounted for by the distributional bias towards taking half of the genetic material from each parent.

Part II: Convergence

Much has been made thus far of the global nature of the search which genetic algorithms perform, and their robustness over wide classes of problems. There is a hint that the search is not always this powerful, however, in Whitley's various results suggesting that the genetic algorithm sometimes fails to search the space properly, for if the search remained truly global it is hard to see how the problem of hidden node redundancy could fail disrupt the search for good sets of connection strategies given his representation. It is, in fact, common for genetic algorithms to converge to the wrong optimum. Many have tried to understand this, and a number possible solutions have been suggested, both on theoretical grounds and as a result of experiment.

The most interesting ways of tackling this have involved the use of "crowding" (De Jong (1975)) "sharing functions" (Goldberg & Richardson (1990)) and isolated sub-populations (for example, Cohoon *et al* (1987)). The idea underlying each of these approaches is much the same, that of *speciation* as a way of allowing a population to track more than one optimum, though the method for achieving speciation is markedly different in each case. De Jong's crowding mechanism attempts to ensure that offspring replace chromosomes which are similar to themselves: it achieves this by comparing the chromosome representing the child to those for each member of a randomly-selected sub-population and replacing whichever member of that sub-population has the highest "overlap" with the child. Goldberg and Richardson's sharing functions limit the total amount of utility available to individuals in each "niche"—an area either of the search space S (for "phenotypic" sharing) or the representation space C (for "genotypic sharing"). The aim is to generate a population which at steady state has numbers in each niche proportional to the utility of the points in that niche. With isolated sub-populations the mechanism is more basic, simply allowing only low levels of inter-breeding between individuals in otherwise separately-maintained sub-populations. This method is particularly suitable for implementation on parallel machines.

Each of these methods has had some success in tackling the problem of "premature convergence", though the problem is far from solved, but since they are all based on the idea of speciation there is a relatively small limit to the number of optima they can track. This is particularly relevant to studies of neural networks since the levels of redundancy are high in the kinds of representations typically used. But of course, in these cases the aim is *not* to try to get the genetic algorithm to track all of the optima, which would be a stupendous waste of effort since they are all equivalent: rather, the aim is to devise ways of allowing the algorithm to recognise and "fold out" the redundancy. This work is an attempt to try to understand *how* genetic algorithms converge prematurely in the hope that this will provide greater insight into the problem, allowing better understanding of existing results and possibly suggesting new ways of seeking to resolve it.

To this end a simple problem with a single global optimum is introduced, and then modified by the addition of a particularly pernicious kind of local optimum. A novel measure of diversity, the *view* of the population, is then introduced and used to analyse how the algorithm fails to achieve the correct solution. The view seems to give some useful insight into the way that loss of diversity leading to premature convergence occurs. The results obtained suggest that loss of diversity can occur without being noticed by standard diversity measures, often very early in the course of the optimisation.

1 Framework: The Problem Without Local Optima

The simple form of the problem considered is that of finding a particular sentence in the space of all strings of given length composed of letters and spaces. This problem is simple to understand and has a particularly natural representation as a chromosome. Moreover, there is a natural utility function which awards one point to a test string for every character in it that correctly matches the corresponding character in the reference string being sought. Formally, each allele on the chromosome is drawn from the set

$$\mathcal{G} = \{\triangle, \text{`a'}, \text{`b'}, \ldots, \text{`z'}, \text{`A'}, \text{`B'}, \ldots, \text{`Z'}\},$$

where $\triangle$ codes a space and single quotes represent some coding of the character inside, typically ASCII. Then, if the target sentence (global optimum) ω uses n characters, the search space is $\mathcal{C} = \mathcal{G}^n$ and a suitable utility function

$$\mu : \mathcal{C} \longrightarrow \mathbb{N}$$

is defined by

$$\mu(\eta) = \sum_{i=1}^{n} \delta_{\eta_i \omega_i}.$$

(δ_{ij} is the Kronecker Delta, which is 1 if $i = j$, and zero otherwise.) This problem has a single optimum at ω, which is particularly easy to find since there is no epistasis, deception, or multimodality; no discrete optimisation strategy should fail. The problem as stated is therefore uninteresting, but provides a useful base upon which to build difficult optima.

2 Adding Pernicious Local Optima

To state the obvious, and anthropomorphising freely, local optima cause problems because, they lure the genetic algorithm towards points other than the global optimum ω. In doing so they can be said to *obscure* the global optimum, and the more successfully they obscure it the more pernicious shall they be said to be. The point of this remark is that pernicious local optima can be viewed as those which are hard to escape because they blind the the algorithm to other optima.

To incorporate these ideas, the problem outlined above is modified to include a set Q of optima, $Q = \{\omega^0, \omega^1, \ldots, \omega^{N_L}\}$. For reasons described below, ω^0 will always be the global optimum and the other N_L optima will be local. Having added these, the utility function is modified to become $\mu : \mathcal{C}^n \longrightarrow \mathbb{N}$ where

$$\mu(\eta) = \max_{\omega^j \in Q} \sigma(\eta, \omega^j),$$

$$\text{and} \quad \sigma(\eta, \omega) = \sum_{i=1}^{n} \delta_{\eta_i \omega_i} (2 - \delta_{\eta_i \triangle}) \,.$$

The motivation for this fitness function is as follows:

- The function $\sigma : \mathcal{C}^2 \longrightarrow \mathbb{N}$ gives a measure of the similarity of two strings.
- The term in parentheses weights letters twice as heavily as spaces. This ensures that perfectly matching a target string with a given number of spaces yields lower utility than perfectly matching a target string having fewer spaces. The global optimum ω^0 will have fewer spaces than any other target string ω^i.
- All optima are maximally pernicious: a string feels selection pressure only in the direction of the optimum (or optima) to which it is nearest. The population gains no extra information about other optima as a result of the evaluation of a string that can see only one optimum.

3 Loss of Diversity: Monitoring a Genetic Algorithm

The usual explanation for genetic algorithms' convergence to the wrong optima is based on the notion of *premature convergence*. What happens under this explanation is that a chromosome is generated which lies significantly closer to a (local) optimum than do most other chromosomes in the population. This gives the schemata it instances a high sample average for utility, so that a great many more instances of such schemata are generated. These quickly come to dominate the population and the optimisation therefore settles in the local optimum.

There are two natural ways to monitor population diversity within any genetic algorithm. Monitoring the spread of utilities seen within the population requires very little extra work since the utility of each member is at all times needed anyway. Apart from possible problems arising from redundant representations (different chromosomal representations of equivalent solutions) it is also easy, if computationally expensive, to compute some measure of diversity based upon the chromosomes themselves. A natural measure for many problems is the sum of all pairs of Hamming distances between solutions. For the problem described above, and a population $\mathfrak{B}$ of size N, this gives the diversity $\Delta : \mathcal{C}^N \longrightarrow \mathbb{N}$ as

$$\Delta(\mathfrak{B}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \sum_{k=1}^{n} (1 - \delta_{\eta_k^i \eta_k^j})$$

where η^i is the ith member of $\mathfrak{B}$.

In the case of the problem already outlined, artificial by its very nature, but for this reason also useful for analysis, there is available a third measure of diversity. This measure reports for each optimum in Q, the number of members of the population which can 'see' it. For the optimum ω^i this is the number of chromosomes η whose similarity $\sigma(\eta, \omega^i)$ to ω^i is greater than to any other optimum ω^j. Where a chromosome is equally close to k optima each is awarded credit $1/k$ for its utility. This measure will be called the *view* $\Upsilon : C^N \longrightarrow \mathbb{R}^{|Q|}$ and the ith component Υ_i of Υ (that corresponding to the ith optimum ω^i) is defined by

$$\Upsilon_i(\mathfrak{B}) = \sum_{\eta \in \mathfrak{B}} v(\eta, \omega^i),$$

$$\text{where} \quad v(\eta, \omega) = \frac{\delta_{\mu(\eta)\sigma(\eta,\omega)}}{\sum_{k=0}^{|Q|} \delta_{\mu(\eta)\sigma(\eta,\omega^k)}}.$$

If the genetic algorithm is searching the space thoroughly each component Υ_i of Υ would be expected to be non-zero at least until solutions have been found which have better utility than the corresponding optimum ω^i. Flat distributions over Υ show the population exploring widely, and will be referred to as *broad* views. In contrast, peaked distributions show it concentrating its efforts around the optima corresponding to the peaks and correspond to *narrow* views.

4 Results of Monitoring

As a demonstration of the power of mutation combined with selection pressures, Dawkins (1988)) demonstrated that this process could produce the string

```
METHINKS IT IS LIKE A WEASEL
```

very quickly. Expanding on this example the following five sentences have been chosen as the optima.

`Methinks It Is Like A Weasel`	(ω^0)
`Premature      Convergence`	(ω^1)
`A Local Optimum`	(ω^2)
`One Two Three Four Five Six`	(ω^3)
`Francis  Bacon  Writ This`	(ω^4)

The particular genetic algorithm implemented used fine-grained (asynchronous) update, used crossover all the time, inversion half the time, and gave each allele a 4% chance of mutation during reproduction. The choice of members for reproduction was as described originally by Holland (1975), using "roulette-wheel" selection.

The graphs in figures 1 and 2 show the way that the population's view Υ, its diversity Δ, and the utility of its best solution vary with the total number of solutions which have been assessed. Notice that a logarithmic scale is used on the x-axis, to allow important detail in the early stages to be seen. Each line on the graphs showing the *view* corresponds to one optimum and shows the number of members of the population which can see that optimum. Twenty runs were performed with these parameters, and on four occasions the global optimum, ω^0, was found. Figure 1 shows that in these cases after only 1000 sentence evaluations every string could see only that optimum. Notice that at this stage the best solution in the population has a utility of only 16 compared with the optimal 51, and the diversity appears to be declining smoothly: the genetic algorithm has not converged, and takes an order of magnitude longer to do so.

Of the twenty runs, another four resulted in convergence to optimum ω^1. Figure 2 shows a very similar pattern averaged over these four runs.

5 Discussion and Implications for Gannets

The results above reveal an interesting and perhaps unexpected phenomenon. What is seen is not so much premature convergence as a premature clustering around a single optimum. The solutions are not particularly close to their "chosen" optimum at this stage, but they are all closer to it than to any other. This is the case even though the level of diversity as measured by Δ is high over the population.

As has been emphasised, the optima studied thus far in this paper are deliberately pernicious, and it seems probable that few real applications will involve optima as difficult to escape as these. It is not so much the fact that the population gets stuck in a slightly less good optimum than the best that is worrying as the very early stage at which it stops exploring other areas of the search space. The results suggest that the problems usually associated with premature convergence may if fact emerge far earlier than is usually thought to be the case.

In the context of genetic training algorithms for neural networks, these results are slightly more double-edged. Some surprise was expressed earlier that Whitley (Whitley (198?), Whitley, Starkweather & Bogart (1989), Whitley & Hanson (1989b), Whitley & Hanson (1989a)) consistently claimed to be able to train neural networks with genetic algorithms despite his having made no apparent attempt to tackle the permutation problem for hidden nodes. Commenting on these results, Belew, McInerney & Schraudolph

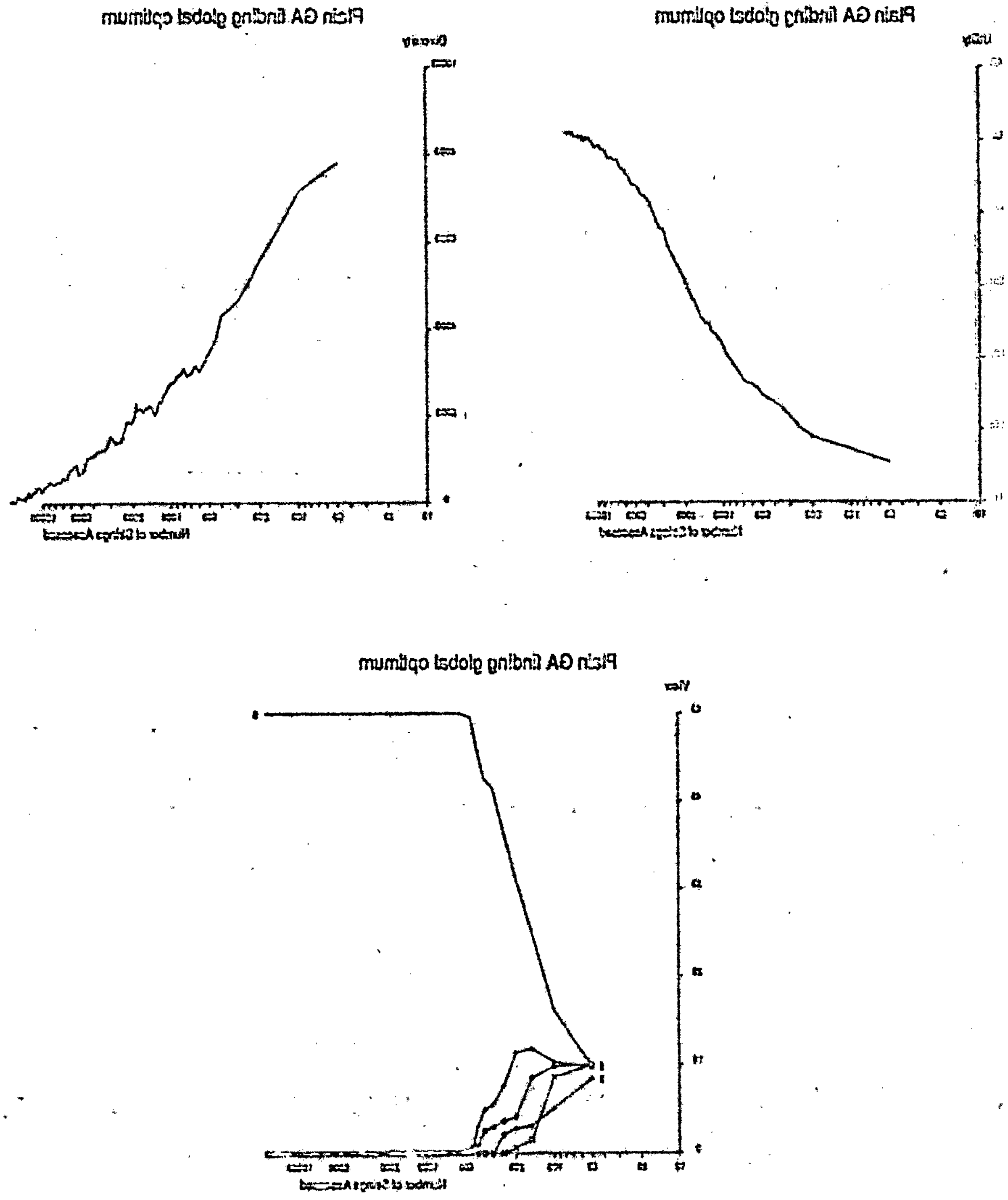
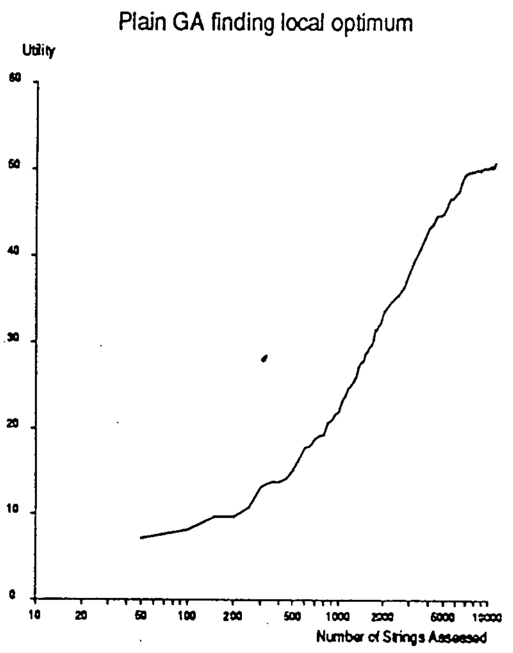
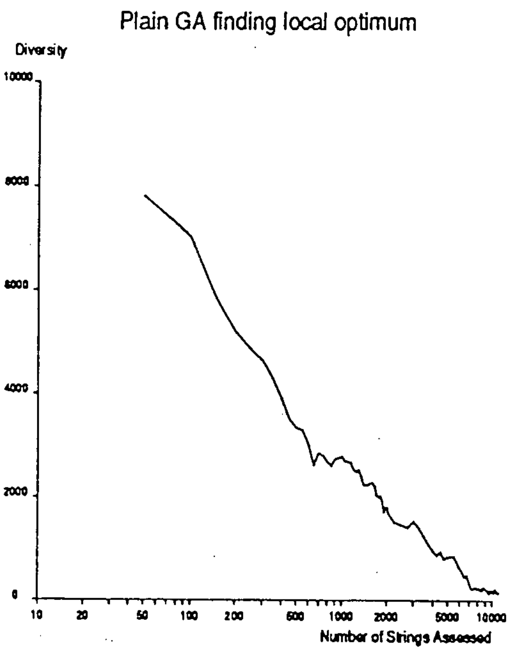


Figure 1: Progress of a simple genetic algorithm having pernicious local optima but converging to the global optimum.





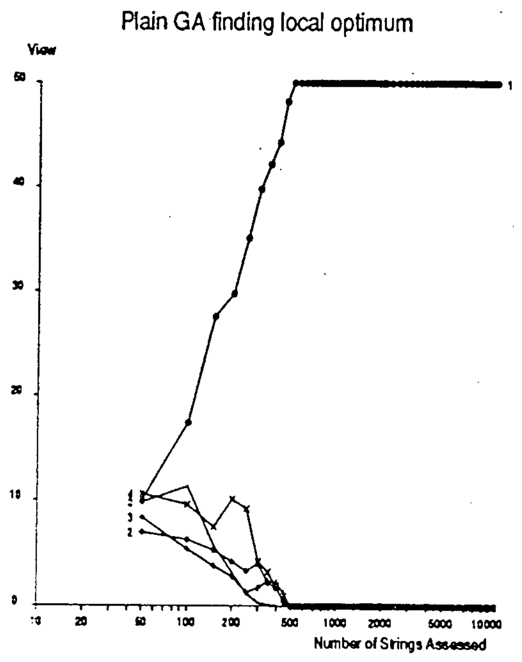


Figure 2: Progress of a simple genetic algorithm having pernicious local optima and failing to find the global optimum.

(1990) write:

> If very small populations are used with the GA, there is not "room" for multiple alternatives to develop. In this case, whichever solution is discovered first comes to dominate the population and resist alternatives. This approach has been used by Whitney (sic) (Whitley & Hanson (1989b)), and in some experiments of our own (Wittenberg & Belew (1990)).

Whitley's various reports do not entirely support this interpretation, and slightly more complex effects could be at work, especially since he reports that increasing his population size from 200 to 1000 produced a performance improvement, for example, on the 4-2-4-adder problem (Whitley & Hanson (1989b)). He places great emphasis on the differences between his genetic algorithm, which he calls Genitor, and "standard" genetic algorithms. To quote (Whitley & Hanson (1989b)):

> *GENITOR* is an acronym for *GEN*etic Implemen*TOR*, a genetic search algorithm that differs in marked ways from the standard genetic algorithms originally developed by John Holland and his students. In a standard genetic algorithm, the parents are replaced by their offspring after recombination. The idea is that the parent's (sic) constituent hyperplanes have a good chance of surviving. Thus, the genetic material of the parents may survive and remain in the population, although the parents themselves do not. In the *GENITOR* approach the offspring do not replace their parents but rather a low ranking individual in the population. Another key difference is that the *GENITOR* algorithm is designed to allocate reproductive trials to genotypes based on their rank in the population rather than an absolute measure of fitness relative to the population average (Whitley (1989))

There is nothing particularly unusual about any of these variations: Genitor uses fine-grained update, which Holland (1975) discussed, and rank-based selection following the lead of Baker (1985). Holland's recommendation, however, was that the member of the population to be replaced was chosen uniformly from the population, to ensure the same statistical behaviour as for coarse-grained ("synchronous") update, whereas Whitley discards a low-ranking individual. This is not a particularly uncommon practice: (Syswerda (1989), for example, describes using inverse-ranking to determine deletion in "steady-state genetic algorithms") but is non-standard. Moreover, he claims (Whitley (1989)) that '[r]anking, coupled with one-at-a-time reproduction gives the search greater focus. Once the *GENITOR* algorithm finds a good genotype it stays in the population until displaced by a better string. This means that the algorithm is less prone to "wander" in the search space and will maintain an emphasis on the best schemata found so far.'

The results presented here have suggested that there are occasions when genetic algorithms cluster around one optimum rather early in the search. In fact, this research (which was one of the first genetic algorithms the author used) used an algorithm that bore some similarities to Whitley's Genitor: indeed, at one stage trials used inverse rank-based deletion (so that lower ranking chromosomes had a higher probability of begin replaced than higher-ranked ones) but this was abandoned because of poor results. Indeed, it would be expected that biasing deletion towards the bottom would increase the probability of premature convergence and clustering around optima as described. Whitley's *GENITOR* package has been used for various problems, including standard suites, but seems to have concentrated on gannet problems. For example, he writes (Whitley & Hanson (1989b))

> The standard genetic algorithm has been tuned on [De Jong's functions] for over a decade (De Jong (1975)). The optimization problems posed by neural networks provide a new and more challenging test of genetic algorithms.

In this context it is possible to begin to understand how it is that Whitley achieves his results with gannets. With Genitor, the top-ranking individual in the population will remain in place until succeeded by an individual with higher utility, and this top-ranking individual will receive the largest number of reproductive trials. This means that its replacement is very likely to be one of its own progeny, inheriting most of its characteristics. This, combined with the fact that the members deleted from the population are always of low-rank will tend to result in a kind of "gauge-fixing": in effect, the environment is very hostile towards any chromosome which does not use its hidden nodes in a way similar to the fittest individual. Any scepticism about this explanation might be assuaged by noting that the problems Whitley typically studies (Whitley & Hanson (1989b), Whitley (1989), Whitley & Hanson (1989a)) are the 4-2-4 encoder, 2-2-1 XOR and a 2-4-3 adder, the largest of which only has a potential $4! = 24$-fold symmetry. As Whitley & Hanson (1989a) note, genetic algorithms must be tested on very much larger problems than those they have tackled: the author's strong suspicion is that these will prove very much more difficult to solve using Whitley's methods.

Longevity, Fecundity, Fidelity
— RICHARD DAWKINS, Replicator Selection and the Extended Phenotype (1978)

7 Gannets

1 Equivalent Neural Networks

If the work in chapter 5 is to be taken seriously then the principal question which has to be addressed is 'When are two neural networks equivalent?', for it is the development of appropriate equivalence relations which allow the selection of proper formae, and once these are known the task of finding suitable genetic operators for their manipulation is a very much more restricted and manageable one. The question as stated, however, is not well posed, for it is not a single equivalence relation that is sought, but rather a set of equivalence relations, preferably separable, with the property that compatible formae induced by them close under intersection. Despite this, it is useful to consider the maximum precision equivalence relation that might be sought.

One relation it is tempting to impose would classify two networks as equivalent if they produced the same output for all possible input patterns. Such "functional equivalence" would subsume trivial equivalences such as the equivalence between two networks that are the same up to a permutation of hidden node labels, and would further capture equivalence between two networks one of which had some completely redundant connections, or which implemented with several nodes a function which another network implemented with one node. It would in fact go further than this, for in principle there might be two networks with completely unrelated internal structures which happened to perform the same mapping, and this equivalence relation would identify these. The set of networks, Ω, has been so broadly defined that there will always be such dissimilar but functionally equivalent networks if only because given any network

$$\mathcal{N} : \mathbb{R}^{|I|} \longrightarrow \mathbb{R}^{|O|},$$

a network which is functionally equivalent can be defined simply by making the activation function v_j of the jth output node the same as the jth component of $\mathcal{N}$ and giving full connectivity between input and output nodes, omitting to use any hidden nodes.

Of course, this example is of no direct relevance to the real case, for in practice the set of available activation functions will always be restricted, and in any case the example given is directly contrary in spirit to the conventional neural networks approach, where ideas of distributed representations, fault tolerance and neural plausibility are all to greater or lesser extents important. It is also an equivalence relation which while easy to specify precisely in the abstract, would be almost impossible to express in a form which allowed the formae to be extracted, and genetic operators to be designed which manipulated these formae usefully. But this is more a failure of the kind of equivalence relation chosen than the set Ω of networks considered, for it is no more reasonable (or perhaps even desirable) to expect an equivalence relation to capture all functionally equivalent networks than it is to ask one to extract all points of the same utility for a general problem. Moreover, it will not necessarily be the case that all functionally equivalent networks will be awarded the same utility; for example, other things being equal, small networks might be awarded higher utility than larger ones for their parsimony.

Instead of looking at functional tests for equivalence, it might be sensible to consider when two networks are similar in terms of their structure and likely chromosomal representation. Suppose that a network $\mathcal{N}'$ is obtained from $\mathcal{N}$ by the application of some operator Γ. Under what circumstances might the mapping performed by the two networks be expected to be similar?

To repeat a point made tediously often already, but which will continue to bedevil this study, if the operator does no more than permute hidden node labels then the operation is essentially null, and there is guaranteed to be no change in the performance of the network. This immediately suggests that *any* equivalence relation used to induce formae ought not to distinguish between networks on this basis. Further, a great many networks are known to be little affected by the deletion of a small number of weights so it might be appropriate to examine equivalence relations which associate networks that have a large number of common weights. If the activation functions $v_j \in V$ used are continuous and reasonably smooth it would additionally be expected that small perturbations in weight values (or indeed the activation functions themselves) would have relatively small effects on the network's functionality, suggesting another possible set of formae.

It is not difficult to extend this list in the abstract, but it turns out to be extremely difficult to turn any of these into useful formae. In order to see why it will be useful to return once again to the hidden node permutation problem.

2 The Permutation Problem Revisited

In order to emphasise how difficult is the problem of folding out the hidden node redundancy, the problem of aligning network topologies will be examined. An algorithm will be introduced which constructs a standard ordering for any 3-layer, feed-forward network in such a way that network topologies which are equivalent up to permutations of hidden node labels are mapped to the same "standard" representation, and non-equivalent networks map to different representations. What is meant by this is essentially that if the networks are drawn on paper in the manner of figure 1, with the hidden nodes in numerical sequence, the same picture will result from networks which are equivalent up to permutations of node labels if the ordering prescribed by the algorithm is used.

DEFINITION The term *topological connectivity* shall refer to those properties of a connection diagram for a neural network which are independent of the labelling of the hidden units. Two networks will then be said to be *topologically equivalent* if they have the same topological connectivity, i.e., if one may be transformed into the other by a permutation of the labels of the hidden nodes. Notice that, trivially, topological equivalence is an equivalence relation. □

DEFINITION A description of the connectivity of a network shall be called a *representation* of that network. Representations may refer to hidden units by labels, though these will be understood to be arbitrary. □

ALGORITHM (Constructing a Standard Representation for a 3-layered Net)

Given a set I of input nodes

$$I = \bigcup_{j=1}^{N_i} \{i_j\},$$

a set O of output nodes

$$O = \bigcup_{j=1}^{N_o} \{o_j\},$$

and a set of hidden nodes H having cardinality N_h, define $N_e = N_i + N_o$ and let $E = I \cup O$. Label the elements of E by defining new labels for the external nodes e_i, $i \in \{1, 2, \ldots, N_e\}$ as

$$e_j \triangleq \begin{cases} i_j, & \text{if } j \le N_i, \\ o_{j-N_i}, & \text{otherwise,} \end{cases}$$

so that

$$E = \bigcup_{j=1}^{N_e} \{e_j\}.$$

Then the problem is effectively transformed into a two-layer problem without loss of generality. Let the connections between a hidden node $h_j \in H$ and external nodes $e_k \in E$ be represented by a connection c_k^j where

$$c_k^j \triangleq \begin{cases} 1, & \text{if } h_j \text{ is connected to } e_k, \\ 0, & \text{otherwise,} \end{cases}$$

so that the upper label on c refers to the hidden node and the lower one to the external node. The introduction of an index set

$$Z \triangleq \{1, 2, \ldots, N_h\}$$

to label the hidden nodes facilitates the description of the algorithm. An arbitrary initial ordering is imposed upon H so that

$$H = \bigcup_{j \in Z} h_j,$$

but it should be understood that this is not the ordering that the algorithm produces.

In order to follow the description of the algorithm the reader may find it helpful to refer to figure 1, which presents a worked example.

The algorithm begins by inductively constructing a sequence $\{H^\alpha\}$ of sequences $\{H_j^\alpha\}$ of sets of hidden nodes. The initial sets H_j^1 are defined by

$$H_0^1 \triangleq H$$

$$H_j^1 \triangleq \begin{cases} \{h_k \in H_{j-1}^1 \mid c_j^k = 1\}, & \text{if this set is non-empty,} \\ H_{j-1}^1, & \text{otherwise.} \end{cases} \qquad (1a)$$

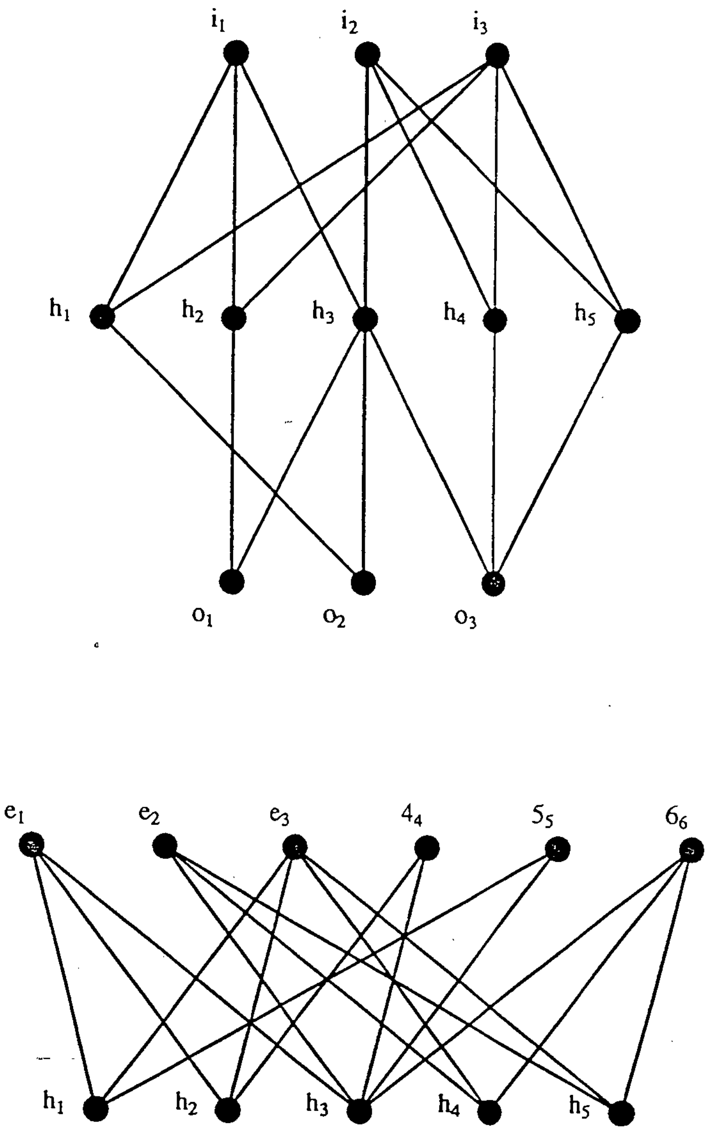


Figure 1: Constructing a standard representation

The diagram at the top shows the original network, and the one below shows the network after transforming to a single layer. In the derivation below, the primed values are the hidden node label assignments furnished by the algorithm.

$H_0^1 = H;\ H_1^1 = \{h_1, h_2, h_3, \};\ H_2^1 = \{h_3\} == H_\star^1;\ \Longrightarrow h_1' = h_3.$
$H_0^2 = \{h_1, h_2, h_4, h_5\};\ H_1^2 = \{h_1, h_2\};\ H_2^2 = \{h_2\} = H_\star^2;\ \Longrightarrow h_2' = h_2.$
$H_0^3 = \{h_1, h_4, h_5\};\ H_1^3 = \{h_1\} = H_\star^3;\ \Longrightarrow h_3' = h_1.$
$H_0^4 = \{h_4, h_5\};\ H_1^4 = H_0^4;\ H_2^4 = H_1^4;\ H_3^4 = H_2^4;\ H_4^4 = H_3^4;\ H_5^4 = H_4^4;\ H_6^4 = H_5^4 = H_\star^4;$
$\Longrightarrow h_4' = h_4,\ h_5' = h_5$ or $h_4' = h_5,\ h_5' = h_4.$

[Thus H_1^1 comprises those hidden nodes connected to e_1 provided that there is at least one such node. If there is not, $H_1^1 \triangleq H_0^1 (= H)$. H_2^1 is then constructed by taking all those nodes in H_1^1 which are connected to e_2, again provided that there is at least one such node; otherwise, $H_2^1 \triangleq H_1^1$. This process continues recursively until the termination condition described below is reached.]

Such sets are generated for increasing j until either a singleton is produced or $H_{N_e}^1$ has been reached. In the former case the hidden unit in the singleton has successfully been identified as having different connectivity from all other units, and it may thus be labelled with the first free label from the index set Z. In the latter case a set of hidden units have been identified all of whose members share a common connectivity. Formally,

$$\forall k \in Z\ (h_k \in H_j^1)\ \forall k' \in Z\ (h_p \in H_j^1)\ \forall l \in \mathbb{Z}_{N_e}\ (e_l \in E)\ :\ \left(c_l^k = 1 \iff c_l^{k'} = 1\right).$$

These cannot be distinguished and may labels may be distributed to them arbitrarily. Let n_1 be the number of nodes in this set, select the next n_1 labels from the index set and distribute these over the elements of H_j^1. In either case let $H_\star^1$ be the set of elements so far successfully labelled.

After this process at least one node is guaranteed to have been labelled. Having generated an initial sequence of sets $\{H_j^1\}$, further sets $\{H_j^\alpha\}$ are now inductively generated in the manner described below, proceeding as for $\{H^1\}$ to generate ever smaller subsets until only equivalent nodes remain—those with the same external connections. Specifically, define

$$H_0^\alpha \triangleq H \setminus \bigcup_{k=1}^{\alpha-1} H_\star^k.$$

and by induction on j define

$$H_j^\alpha \triangleq \begin{cases} \{h_k \in H_{j-1}^\alpha \mid c_j^k = 1\}, & \text{if this set is non-empty,} \\ H_{j-1}^\alpha, & \text{otherwise,} \end{cases} \tag{1b}$$

until either a singleton is produced or run out of external nodes j to test against. This process is exactly analogous to the one described for H_j^1 except that some nodes have now been deleted from the initial set H_0^α. Again, the final set is denoted $H_\star^\alpha$ and contains only equivalent nodes. An appropriate number of the remaining unused labels are drawn in sequence from the index set Z to label these units.

When the inductive processes described above have been exhausted, all hidden nodes which are connected to at least one external node will have been labelled, and this labelling will be well-defined up to permutations of nodes with identical connections to external units. The only remaining task for the procedure is to label those hidden nodes with no external connections. Yet these are manifestly equivalent to each other, so they may be arbitrarily labelled using the remaining indices from the index set. This completes the algorithm, all hidden units having been assigned a labelling. □

THEOREM Given two representations η and ζ of 3-layered networks the algorithm[27] will furnish new representations η' and ζ' (respectively) such that

1. If ζ and η are topologically equivalent, then $\zeta' = \eta'$
2. If ζ and η are not topologically equivalent, then $\zeta' \neq \eta'$.

Proof

To verify that the algorithm does indeed take different representations of topologically equivalent networks to a common representation assume that $\pi \in \mathcal{P}_{N_h}$ is a permutation of the labels of the the hidden units of two topologically equivalent networks η and ζ for which

$$\forall k \in Z : {}^{\eta}h_{\pi(k)} = {}^{\zeta}h_k. \tag{2a}$$

Leading superscripts like these will consistently be used to indicate which network a quantity is defined with respect to. Then by definition,

$$\forall j \in Z\ \forall k \in Z : {}^{\eta}c_j^{\pi(k)} = {}^{\zeta}c_j^k. \tag{2b}$$

In the algorithm the labellings of the hidden units are determined only by the sequences of sets generated, and these are governed by the equations (1a) and (1b). Comparing equation (1b) for the two representations ζ and η, in the former case:

$$ {}^{\zeta}H_j^{\alpha} = \begin{cases} \{ {}^{\zeta}h_k \in {}^{\zeta}H_{j-1}^{\alpha} \mid {}^{\zeta}c_j^k = 1 \}, & \text{if this set is non-empty,} \\ {}^{\zeta}H_{j-1}^{\alpha}, & \text{otherwise,} \end{cases} \tag{3}$$

and in the latter case

$$ {}^{\eta}H_j^{\alpha} = \begin{cases} \{ {}^{\eta}h_k \in {}^{\eta}H_{j-1}^{\alpha} \mid {}^{\eta}c_j^k = 1 \}, & \text{if this set is non-empty,} \\ {}^{\eta}H_{j-1}^{\alpha}, & \text{otherwise.} \end{cases} \tag{4}$$

Substituting in (3) from (2a) and (2b) gives:

$$ {}^{\zeta}H_j^{\alpha} = \begin{cases} \{ {}^{\eta}h_{\pi(k)} \in {}^{\zeta}H_{j-1}^{\alpha} \mid {}^{\eta}c_j^{\pi(k)} = 1 \}, & \text{if this set is non-empty,} \\ {}^{\zeta}H_{j-1}^{\alpha}, & \text{otherwise.} \end{cases}$$

But of course, in this rôle $\pi(k)$ is simply acting as a label running over the elements in a set, which could validly be replaced by k; $\pi(k)$ is merely a perverse choice of dummy variable. On making this substitution it can be seen that

$$ {}^{\zeta}H_j^{\alpha} = \begin{cases} \{ {}^{\eta}h_k \in {}^{\zeta}H_{j-1}^{\alpha} \mid {}^{\eta}c_j^k = 1 \}, & \text{if this set is non-empty,} \\ {}^{\zeta}H_{j-1}^{\alpha}, & \text{otherwise,} \end{cases}$$

[27] Constructing a Standard Representation for a 3-layered Net

which shows that the sets defined by the algorithm with respect to the two representations (η and ζ) at step α are identical provided that this was true for previous steps. An almost identical argument applies to equation (1a), and when this is performed this completes an inductive proof that the sets are indeed the same since the initial set for the inductions is H in each case. Thus the labelling of the hidden nodes, which depends only on these sets, must be the same under these two representations up to permutations of equivalent nodes, which are indistinguishable.

This suffices to demonstrate that the subsets produced by the algorithm are the same when the labellings of the hidden nodes in η are simply a permutation of those in ζ, and thus that different representations of topologically equivalent networks are mapped to a common representation by this algorithm. It should be immediately apparent that topologically non-equivalent networks cannot possibly ever share a representation, and so provided it is accepted that the algorithm does indeed furnish a valid representation of any network the proof is complete. □

3 Unbounded Problems

One of the difficulties of genetic training techniques that has not really been discussed here is that even when the problem is restricted to finding good combinations of weights in the context of a network with fixed topology and activation functions, the parameter space is unbounded. This problem was alluded to by Belew, McInerney & Schraudolph (1990), who avoided it by searching only for sets of initial weights which a gradient technique would subsequently take as its starting point, but is rather a serious problem, and one not restricted to the domain of neural networks.

It might be thought that this problem would disappear if connection strengths were to be stored as floating point values, which certainly take on sufficiently wide range of values for these purposes, but while this overcomes the storage problem it does not tackle the question of ergodicity satisfactorily. The work in chapter 5 suggested a set of formae suitable for use with smooth, real-valued functions, the locality formae (half-open balls), and the simple random, respectful crossover operator for these was “flat crossover”. While no particularly strong case is being made for R^3 operators in this work, flat crossover seemed to work reasonably well for a surprisingly wide class of functions and would seem to be a reasonable choice of operator for recombining weights, once they have been suitably aligned. The question of counteracting its bias towards central values, however, is rather harder to overcome in the unbounded case, for “extremal mutation”, at least in the simple form presented thus far, is not a satisfactory solution in an (essentially) infinite space.

Various ways of tackling the general problem might be considered. One of the most interesting studies relevant to this discussion is the “ARGOT” strategy of Shaefer (1989). Argot is not really a genetic algorithm in the sense described so far, but it is an adaptive system of broadly the same ilk, employing selective reproduction, the same idealised genetic operators and so forth. The critical difference is that the mapping

$$g : C \longrightarrow S$$

which effects the morphogenesis is itself subjected to adaptation. Quoting Shaefer

> The ARGOT strategy consists of a substructure, roughly equivalent to [a standard genetic algorithm], that employs crossover, mutation, and selection to perform the standard, implicitly parallel, hyperplane analysis, plus numerous triggered operators which modify the intermediate mapping that translates the chromosomes into trial parameters. These triggered operators are controlled through three types of internal measurements performed upon the chromosome population or the population of trial solutions.

Shaefer introduces the notion of "roving" parameter boundaries, which move when the population clusters near them. He further has various techniques for changing the resolution of the genes coding parameters and for effecting various other apparently useful changes. His methods certainly offer a way forward, and undoubtedly deserve further investigation, but seem to involve sacrificing some of the analytical machinery developed, something which will be accepted only reluctantly.

The way that Whitley tackled the problem (Whitley, Starkweather & Bogart (1989)) involved simply choosing a large weight range and relatively low accuracy—usually -127 to 127 with zero occurring twice, and using the standard operators to explore the range. Montana & Davis (1989?), in contrast, used floating point weights with an initial distribution given by an exponential. They then ensured that the whole space was accessible through a number of their operators, including a gradient operator, a mutation operator that generated values from the same distribution as that used to seed the initial population initial distribution and another "creep" mutation operator which perturbed weights by values taken from the initial distribution. On the whole, this scheme is more appealing than Whitley's, but still feels rather arbitrary.

If flat crossover were to be adopted, perhaps the most natural approach would be to monitor the maximum and minimum value for each weight across the population and to use a mutation operator which inserted values beyond, but not very far outside the range defined by these extrema. This would seem to satisfy the requirements of ergodicty and tackle the bias inherent in flat crossover, and perhaps has relevance to other operators. No systematic experiments have yet been carried out to investigate these ideas, largely because the work on formae has only recently been completed, but there seem to be some grounds for hope that this form of mutation would allow flat crossover or similar operators to be used satisfactorily.

4 Discussion

The construction of a standard ordering for three layered, feed-forward network topologies can be extended relatively trivially to deal also with weights provided that exact matching is required. Essentially the weights are cycled though on the basis of their external connectivities to resolve any ambiguities in the hidden node ordering generated by the algorithm given, priority in labelling being given (say) to the highest-valued weights. The lifting of the redundancy inevitably makes appeal to the external nodes and connection strengths between hidden and external nodes, for it is these which are *not* arbitrary and allow the redundancy to be lifted. There are two principal obstacles to using these methods to sort hidden units before recombining the chromosomes representing the networks under consideration.

The first is simply the computational intensiveness of the algorithm. Although more efficient algorithms that the one given can undoubtedly be devised, it is hard to imagine one which would not involve a fairly large amount of computation compared to other genetic operations and the evaluation of a network's performance over a reasonable-sized training set. But even if the time could be afforded, a much greater problem lies in the fact that the algorithm implements a mapping which is highly discontinuous. In the case of network topologies (where the notion of continuity is not strictly applicable, but where one can still talk loosely of mappings which take neighbourhoods to neighbourhoods) the swapping of a single pair of connections, or the deletion or addition of a single connection, would be likely dramatically to alter the standard representation furnished by the algorithm. In the continuous case, if redundancies are to be lifted by sorting weights, perturbing a weight in such a way that its value remained bounded by the weights from the network which previously bounded it would have no effect on the "standard representation", but crossing one such boundary would again be likely to have catastrophic effects. It is hard to imagine algorithms which do not suffer from such non-local behaviour not merely at a few points in Ω_F, but over much or most of it. In these circumstances the algorithm is computationally useless, for ultimately there is no interest in recombining two nets which are identical up to hidden node permutations: this cannot yield new information, at least from a respectful operator defined with respect to formae that are invariant under hidden node permutations. Instead the interest lies in making the best possible match between two networks before recombination to ensure that the maximum amount of useful information is preserved. Thus however desirable it may be, it is hard to see how operators will be able to be constructed which respect and properly assort *any* formae induced by equivalence relations with the desirable properties listed above, even supposing that these could be defined.

All of this goes some way to reinforce the comments of Belew, McInerney & Schraudolph (1990), who write

> It seems, for example, that at least in the case of BP networks with a single hidden layer, the differential weighting of the "anchored" (i.e., constant and nonarbitrary) input and output layers might be used to recognize similarly functioning hidden units. Even establishing correspondence among the hidden units of two three-layer networks which have been trained to solve the same problem appears to be computationally intractable, even when we assume that the only difference between the two networks' solutions is a permutation of the hidden units. In realistic networks many other different solutions to the same problem can be constructed, for example by reversing the sign of all the weight,

or taking other "semi-linear combinations".[28] We therefore conclude that attempting to normalise networks before combining them is not feasible.

Thus the picture is somewhat depressing: having identified early in the work a fundamental difficulty facing the application of the machinery of genetic algorithms to training neural networks (the ever-present permutation problem), the course adopted was to investigate extensions to the formalism of genetic algorithms which would justify the use of non-string, and more particularly non-binary representations. This work was reasonably successful, resulting in the notion of formae as an extension of Holland's original schemata, and allows powerful analytical machinery to be brought to bear on more general sorts of genetic algorithms, using operators defined with respect to general equivalence relations. The forma analysis has been shown to be quite useful in a number of problem areas, and certainly allows the goals towards which work on Gannets needs to progress to be formulated more precisely. Sadly, however, after such a reformulation the problems look more daunting than ever. One thesis is not enough.

[28] We speak a bit loosely here. Because BP networks depend on nonlinear "squashing" functions, simple linear combinations are not quite adequate. (Belew *et al*'s footnote.)

8 Connection Strategies

1 MIMD Machines

Much of the work in this study was carried out on a large parallel computing device known as the Edinburgh Concurrent Supercomputer (ECS). This machine is a Meiko Computing Surface, consisting of several hundred Inmos T800 floating-point transputers together with suitable supporting hardware. A transputer is a "computer on a chip", having a central processing unit (CPU) which performs the main computations, a floating-point unit (FPU) for fast floating point computation, 4Kbytes of on-chip memory and four bi-direction communications links, each having a pair of link controllers. In the ECS, each transputer has an additional 4Mbytes of dedicated external memory. There is parallelism built into the transputer: the CPU, FPU and the eight communications links can all process simultaneously without degradation in performance. Pairs of links[29] from two processors may be connected together without any intervening hardware to form a pair of communications paths between the processors. Each transputer can be connected to up to four others in this manner. In the ECS these connections may be made electronically in a process known as electronic reconfiguration.

There is no shared memory in the ECS so in order for different transputers to share data they must engage in *message passing*. Communication between pairs of processors not directly joined together is achieved by explicitly routing messages across the processor network, each transputer in a chain leading from source to destination processor passing the message forward. For this reason the ECS is an example of a message-passing architecture. It is also a *multiple instruction, multiple data* (MIMD) machine because different processors in a single processing network can simultaneously perform different sequences of instructions on different data. This contrasts with *single instruction, multiple data* (SIMD) machines where the same operation is performed on an array of data simultaneously, and with vector processors in which data is sent along a pipe-line through a series of processors, each of which performs some specific task.

2 Topology

Processing on reconfigurable MIMD machines such as the ECS is a new and relatively poorly understood field. One of the key questions in this area concerns network topology: given a reconfigurable machine such as the ECS, what pattern of processor connectivity will obtain maximum performance from the computing surface? In some cases the communications pattern make the best connectivity very obvious. For example, if the problem under consideration is two-dimensional, and the parallel implementation is achieved by a geometrical decomposition of the data over the computing surface, then provided only local

[29] The term *link* will be used to denote variously a unidirectional communications link to or from one transputer, the connection formed when a pair of such links are joined, an in-out pair of 'half-links' to or from a transputer, and the pair of connections formed when two such pairs of half links are joined. This abuse is standard, and it should always be clear from context which use is intended.

information is needed it will not be possible to improve upon a two-dimensional square lattice. Given a more general communications pattern, however, the issues become much more complex.

The approach taken here is to define a set of measures of the expected performance of any processor graph, to examine various graphs in the light of these performance measures and to use the measures as objective functions in optimisation schemes for constructing processor graphs. This approach is useful provided that there is some reason to believe that the performance of programs is governed by the sorts of measures constructed.

Before defining these measures, it will be useful to make a general few observations about some of the factors that limit the performance of processor networks. Minimising the distance that each message has to travel is almost always desirable, partly because the time taken increases with the distance, and partly because the greater the number of links a message has to traverse, the greater is the proportion of the total link bandwidth of the processor network it occupies. On systems where the routing is performed by the CPU, every extra processor that a message has to be routed through reduces the amount of useful computation which that processor is able to perform, providing another reason for seeking to minimise message distances. The distance between two processors will be taken to be the graph-theoretic distance, defined below. This amounts to an assumption that latencies introduced by travelling along wires do not depend significantly on the length of the wires. One aim, therefore, will be to produce graphs which are *compact*. At the same time, however, it will be important that graphs be *well-balanced* in terms of the likely use of each (processor) and link, for if many messages need to pass through a small number of processors or links, bottle-necks will develop and gains from compactness will be lost.

3 Compact Graphs

It will be helpful to introduce notation for discussing graphs. It is conventional to regard a graph K as comprising a set of *edges*, which correspond to the link-pairs described above, and *vertices*, which correspond to the nodes or processors. The number of links (edges) emanating from a node is called the *arity* a (or valency) of that node, and the arity of the graph is usually taken to be the highest arity of any node in the graph. The set of graphs of arity a or less, having n nodes will be denoted $\mathcal{K}(a, n)$, with $\mathcal{K}_\infty \triangleq \mathcal{K}(\infty, \mathbf{Z}_0^+)$—the set of graphs of arbitrary arity and size. It is convenient, given K, to let $K^\star$ denote the set of the graph's nodes and then to take the *size* of the graph to be the number of nodes that it has, so that $|K| \triangleq |K^\star|$.

The graph-theoretic measure will be used when considering the 'distance' between two nodes. Given a graph K,

$$s : K^\star \times K^\star \longrightarrow \mathbf{Z}_0^+$$

gives the distance between a pair of nodes i and j as

$$s(i,j) \triangleq \min\{\, \textit{number of edges traversed in travelling from } i \textit{ to } j \textit{ on } K \,\}.$$

Two measures will be used to measure the compactness of a graph—the *diameter* D and the *mean inter-node distance* ι.

The *diameter* of a graph

$$D : \mathcal{K}_\infty \longrightarrow \mathbf{Z}_0^+$$

is defined by

$$D(K) \triangleq \max_{i,j \in K^\star} s(i,j),$$

the maximum graph-theoretic distance between any pair of nodes on the graph. For example, graph 1 has a diameter of 4.

The *mean inter-node distance* of a graph

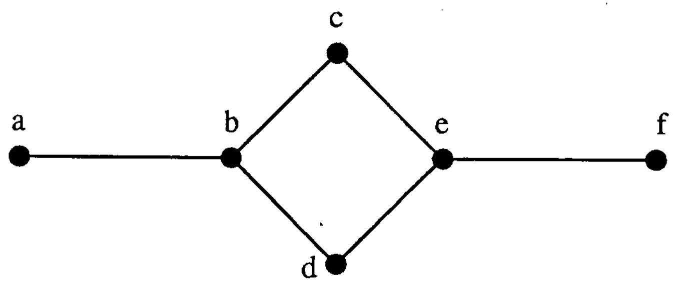
$$\iota : \mathcal{K}_\infty \longrightarrow \mathbf{R}_0^+$$

is the mean distance between each of the $|K|^2$ (ordered) pairs of nodes:

$$\iota(K) = \frac{1}{|K|^2} \sum_{i,j \in K^\star} s(i,j).$$

For example, Graph 1 has $\iota = 1.\dot{5}$.

Diameter is a useful measure for applications which are tightly-coupled, so that the greatest communication time in the system dominates the speed at which the entire application runs. This is the case, for example, with a distributed calculation in which every processor in the system needs to know the result from every other processor between each step of the calculation. Mean inter-node distance, on the other hand, is an appropriate measure when the communications pattern is not pre-determined and any pair of processors are approximately equally likely to need to communicate with one another. A distributed data base would be a good example of this, as would any application for which scattered-spatial decomposition would be appropriate (Fox & Furmanski (1989)). In general, ι gives a good indication both of the number of nodes in the network which are involved in sustaining a typical 'conversation' between two nodes, and the amount of link bandwidth such a conversation absorbs.



Graph 1

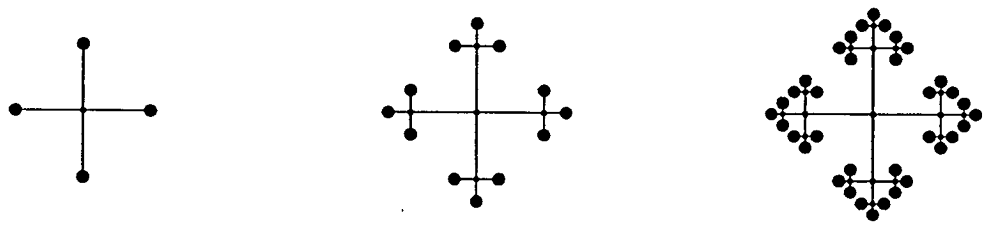


Figure 1: Three generations of Moore Graphs

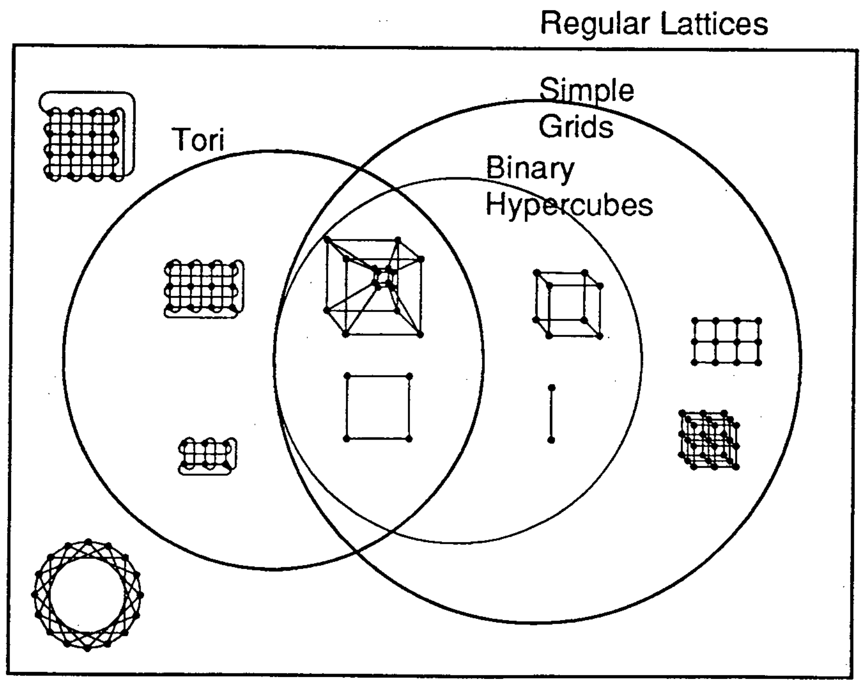


Figure 2: Regular Lattices

4 Well-Balanced Graphs

At least three measures are useful in determining whether a graph is likely to suffer from bottle-necks. Clearly the load on the network is heavily dependent both on the pattern of communications being performed, and on the routing strategy being used. It will be convenient to define measures based on the case where every processor (node) in the graph passes one message to every other processor, and to itself, each unit time-step. The routing strategy used will be a random one based on shortest routes, now described.

Random-Routing Strategy $\mathfrak{R}$

- Whenever a processor holds a message *en route* to another processor, at least one of its links will lead to a processor closer to the destination processor.
- If there is only one such link, it is used, but where there is more than one link which satisfied this condition a random choice is made between these links.

Clearly, under this strategy, every message travels along a route of minimum length from source to destination, but different messages between the same pair of processors may use different paths.

The *most-loaded link load*

$$\lambda : \mathcal{K}_\infty \longrightarrow \mathbf{R}_0^+$$

is defined to be the maximum number of messages carried, on average, by any pair of links in the network as the $|K|^2$ messages are passed under the random-routing strategy $\mathfrak{R}$. For example, the following table gives the load on each of the links in graph 1, and it can be seen that the heaviest load here is 10. In this example, all the links happen to carry integral loads, but in general link-loads can take on fractional values under $\mathfrak{R}$.

link	*ab*	*bc*	*be*	*ce*	*de*	*ef*
load	10	9	9	9	9	10

Similarly, the *most-loaded node load* (or 'worst through-routing load')

$$\kappa : \mathcal{K}_\infty \longrightarrow \mathbf{R}^+$$

is defined as the maximum number of the $|K|^2$ messages routed through any node under strategy $\mathfrak{R}$. For the network in graph 1, the following node loads are obtained:

node	*a*	*b*	*c*	*d*	*e*	*f*
load	11	20	15	15	20	11

Notice again that in the general case node-loads may be fractional.

Finally, it is useful to define a quantity called the *narrowness* (or "worst-cut") of a graph. The idea behind narrowness is to try to gauge the worst communications-to-calculation ratio for any part of the network.

Given a graph K, let $\mathbb{S}(K)$ be the set of all subgraphs of K and $\mathbb{S}_{\bullet}(K)$ be the set of subgraphs of K containing no more than half of K's nodes. Let the number of link-pairs from a subgraph $S \in \mathbb{S}(K)$ to the rest of K be $\mathcal{L}[S]$. Then the narrowness $\nu(K)$ is defined as

$$\nu(K) \triangleq \max_{S \in \mathbb{S}_{\bullet}(K)} \frac{|S|}{\mathcal{L}[S]}.$$

Large values of narrowness correspond to poor communications-to-calculation power for some group of processors in the network. Ideally, since the processing power of the network is fixed, every processor should have as much communications power available relative to its processing power as is possible. Low values for narrowness are therefore desirable.

It seems likely that any 'narrow' graph will have a relatively high most-loaded link load, since high narrowness indicates that a small number of links will carry a lot of message traffic. It is, in fact, possible to bound the narrowness $\nu(K)$ in terms of the most-loaded link load $\lambda(K)$.

Consider again the case where every processor in the network sends one message to every other processor in the network, and one to itself. Given a subgraph $S \in \mathbb{S}_{\bullet}(K)$ let $\overline{S}$ be that portion of K not included in the subgraph. Then clearly $|S| \leq |\overline{S}|$, (from the definition of $\mathbb{S}_{\bullet}(K)$) and since no link in the system carries a load greater than $\lambda(K)$,

$$\frac{2|S||\overline{S}|}{\mathcal{L}[S]} \leq \lambda(K).$$

(This follows because the links joining the S to $\overline{S}$ must carry at least the $2|S||\overline{S}|$ messages between them.) Rearranging, and using the definition of narrowness above,

$$\nu(S) = \max_{S \in \mathbb{S}_{\bullet}(K)} \frac{|S|}{\mathcal{L}[S]} \leq \max_{S \in \mathbb{S}_{\bullet}(K)} \frac{\lambda(K)}{2|\overline{S}|}.$$

Since $\lambda(K)$ is fixed and $|S| \leq |\overline{S}|$, it must be the case that $|\overline{S}| \geq |K|/2$. Thus

$$\nu(S) \leq \frac{\lambda(K)}{|K|}.$$

5 Moore Graphs

It is quite easy to derive tight lower bounds on the diameter and mean internode distance for any graph of fixed arity and size. Moore used simple arguments (Moore (c.1956)) to place a bound on the number of nodes that could be present in a graph of given diameter having nodes with a given arity. His arguments can be inverted to compute bounds on the diameter, and the mean internode distance as follows.

Consider some node $k \in K^*$. The maximum number of nodes which can be at distance one from k is a, the situation when every link connects to a different node. Each of these nodes can then be connected to a maximum of $a-1$ other nodes, so that at most $a(a-1)$ nodes can be at a distance two from k. Plainly the sequence of maximum numbers of nodes that can be present at any given distance (figure 1) is:

distance	0	1	2	3	...	j
number of nodes	1	a	$a(a-1)$	$a(a-1)^2$	...	$a(a-1)^{j-1}$

Thus the maximum number of nodes that can be within a distance D of any node k is

$$\begin{aligned} n &\leq 1 + \left(a + a(a-1) + a(a-1)^2 + \cdots + a(a-1)^{D-1}\right) \\ &= 1 + \sum_{r=0}^{D-1} a(a-1)^r \\ &= 1 + \frac{a\left((a-1)^D - 1\right)}{(a-1)-1}. \end{aligned} \tag{1}$$

Solving for D gives

$$D \geq \log_{a-1}\left(\frac{(n-1)(a-2)}{a} + 1\right).$$

This gives the "D-bound", a lower bound on the diameter of any graph. A graph for which the inequality is in fact met is called a Moore graph: in this case every node is at the centre of a branching tree as shown in figure 1.

A bound on the mean internode distance ι can be obtained in similar fashion. Consider again some node k and assume that it consists of a Moore graph with r "full shells" together with a partially-filled $(r+1)$th shell containing n' nodes. Then the mean distance ι' of nodes from k is

$$\iota' = \frac{1}{|K|^2}\left(0.1 + 1.a + 2.a(a-1) + 3.a(a-1)^2 + \cdots + r.a(a-1)^{r-1} + n'.a(a-1)^r\right)$$

since at distance 0 there is one node, at distance 1 there are a, and so forth. Summing the series gives

$$\iota \geq \iota' = \frac{\frac{a}{a-2}\left(r(a-1)^r - \frac{(a-1)^r - 1}{a-2}\right) + (r+1)n'}{|K|}.$$

This is the "ι-bound".

6 Regular Graphs

The vast majority of MIMD machines are configured in what might be termed 'regular topologies'. The Venn Diagram in figure 2 suggests a classification scheme for these. Simple grids consist of a regular lattice (in any number of dimensions) on which there are connections between nearest-neighbour sites only. To each simple grid there corresponds a torus, which is obtained by connecting nodes on opposite edges of the simple grid. The special case where the extent of a simple grid is two in every dimension gives rise to the class of binary hypercubes, which in even dimensions are also tori. Remaining 'regular' topologies include the helical torus, and the chordal ring, which are topologically equivalent.

The following discussion will focus on tori, for these always perform better than simple grids. Arity four will usually be used for examples because transputers have four links. Where appropriate, binary hypercubes in higher dimensions will also be discussed. In order to provide a reference against which to measure, most of the comparisons will feature the D- and ι-bounds, as well as 'random graphs'.

7 Random Graphs

As well as comparing regular graphs to theoretical bounds, it would be useful to devise some kind of a reference which represented an "average" or "random" graph. It is theoretically possible to sample the entire set of graphs of given size and arity and to compute mean values for the various measures defined above. This might be thought to give a guide to the expectation values for these properties if a graph were to be built by connecting all the links up at random. A procedure for building such a graph is formalised in algorithm $\mathfrak{A}$:

Algorithm $\mathfrak{A}$

1° Begin with a graph having all the nodes present but no links.

2° Repeatedly select from the set of all unconnected links any pair and join them together.

3° Continue performing step 2 until all links have been connected.[30]

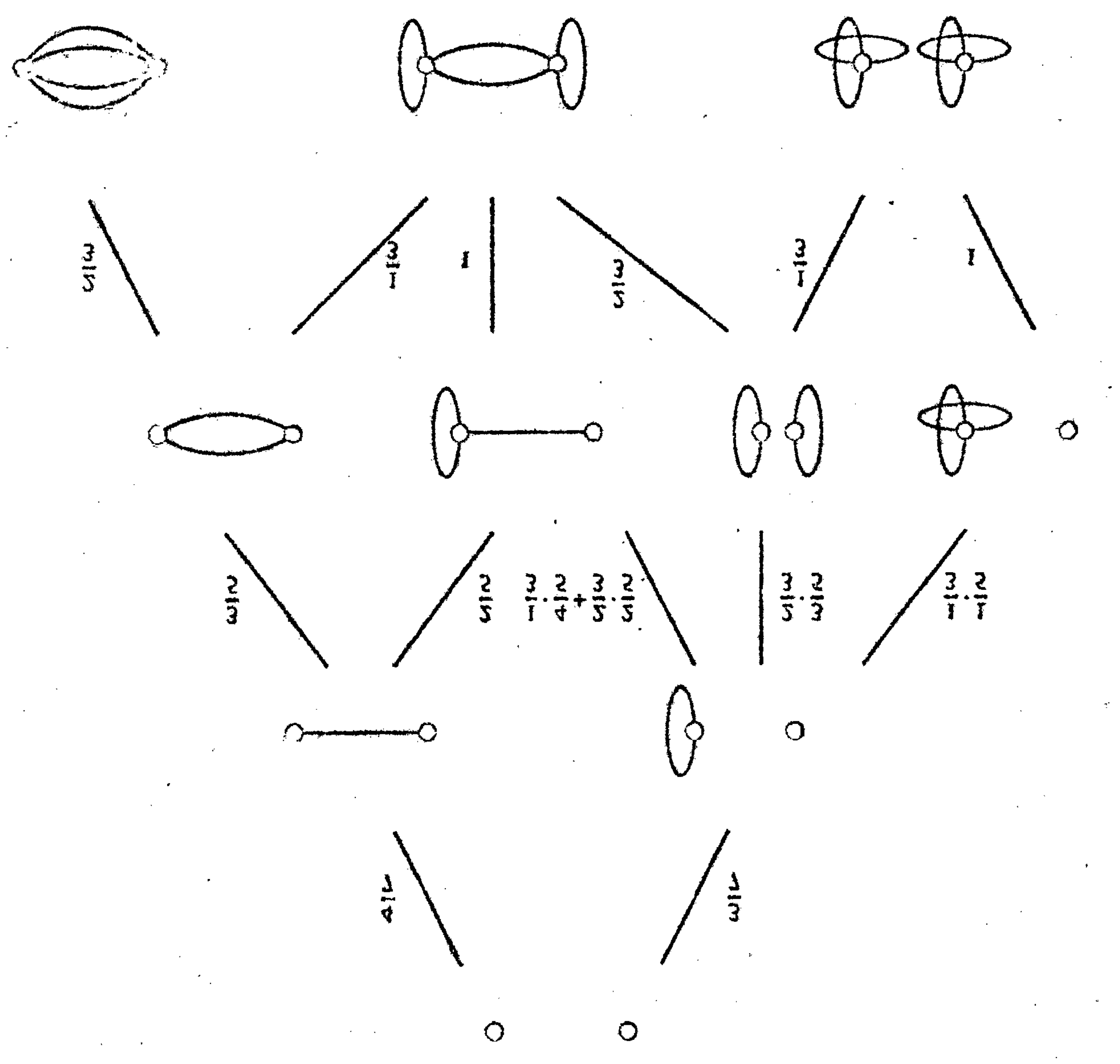
An obvious question arises from this definition: does algorithm $\mathfrak{A}$ sample uniformly the set of all graphs of given size and arity in which all links are used? Perhaps surprisingly, it does not: a proof by example is given below,

THEOREM (Non-uniformity of Algorithm $\mathfrak{A}$)

Algorithm $\mathfrak{A}$ does not sample uniformly the set of all fully-linked graphs of given node-number and arity.

Proof: Consider the set of graphs with two nodes each of arity four. There are only three topologically-distinct fully-linked graphs in this set, the three shown at the bottom of figure 3. If the algorithm sampled these uniformly it would generate each with probability one-third. The branching tree in figure 3 represents the possible ways in which the algorithm can proceed. The numbers associated with the

[30] In the case of graphs of odd node-number and odd arity, one link is left unconnected.



$\frac{3}{7}$
$\frac{4}{7}$
$\frac{3}{5}$
$\frac{2}{5}$
$\frac{1}{3}\cdot\frac{4}{5}+\frac{2}{3}\cdot\frac{2}{5}$
$\frac{2}{3}\cdot\frac{3}{5}$
$\frac{1}{3}\cdot\frac{1}{5}$
$\frac{2}{3}$
$\frac{1}{3}$
1
$\frac{2}{3}$
$\frac{1}{3}$
1

$\frac{4}{7}\cdot\frac{3}{5}\cdot\frac{2}{3}$ $=\frac{8}{35}$

$\frac{4}{7}\cdot\frac{3}{5}\cdot\frac{1}{3}+\frac{4}{7}\cdot\frac{2}{5}\cdot\frac{1}{1}$ $+\frac{3}{7}\cdot\frac{8}{15}\cdot\frac{1}{1}+\frac{3}{7}\cdot\frac{6}{15}\cdot\frac{2}{3}$ $=\frac{24}{35}$

$\frac{3}{7}\cdot\frac{6}{15}\cdot\frac{1}{3}+\frac{3}{7}\cdot\frac{1}{15}\cdot\frac{1}{1}$ $=\frac{3}{35}$

Figure 3: Random Graph Construction Algorithm A

branches on the tree give the probabilities of moving from the partially-constructed graph at the vertex above to the graph at the vertex below. Only topologically-distinct graphs are shown. It can be seen from this illustration that the three graphs are generated with probabilities $8/35$, $24/35$ and $3/35$, rather then $1/3$ each. Thus the algorithm does not sample the space uniformly.

Q.E.D.

It is clear from this that, perhaps counter-intuitively, a distinction must be made between "choosing a graph at random from a space of graphs" and "building a 'random graph' by connecting up the links in an arbitrary order". Graphs generated by algorithm $\mathfrak{A}$ are easier to produce than graphs selected at random from the set, and perhaps correspond more closely to the way a "random graph" might be constructed in practice. For these reasons the figures given below for random graphs are obtained by sampling a number of graphs generated by algorithm $\mathfrak{A}$.

8 Regular and Random Graphs: A Comparison

A comparison between the mean inter-node distances for tori and random graphs at various numbers of processors with fixed arity 4 is shown in figure 4. The ι-bound is also shown for reference. Error bars are not included in the case of the random graphs because they are considerably smaller than the marks used to display the points. It is immediately apparent that as the number of processors increases the mean internode distance rises very much faster for a torus than for a "random graph". Moreover, the random graphs do not diverge to any significant degree from the ι-bound, but maintain mean inter-node distances which exceed the theoretical bound only by a small, roughly constant amount. This immediately suggests that regular lattices are not ideal topologies for any application whose performance might reasonably be expected to be governed by the the mean inter-node distance.

It is possible to understand the way in which the regularity of grid-like graphs increases significantly the distances that messages have to travel. The characteristic pattern of connections on Moore graph ensures that there is a unique shortest path between every pair of nodes. On regular lattices the pattern is almost the reverse: there is a multiplicity of routes between almost every source-destination pair, each equally short. This militates against compactness, for on a compact graph these different routes would lead to different nodes.

9 De Brujn Graphs

The regular graphs so far discussed are highly non-compact, but are not the only kinds of regular graphs which can be considered. de Brujn (1946) discovered a class of graphs repreatedly discovered since 1894, which are known under various names including de Brujn's and "shuffle networks". They can be defined only for nodes of even arity $a = 2b$, and for networks with b^k nodes, $k \in \mathbb{Z}^+$. The simplest way of

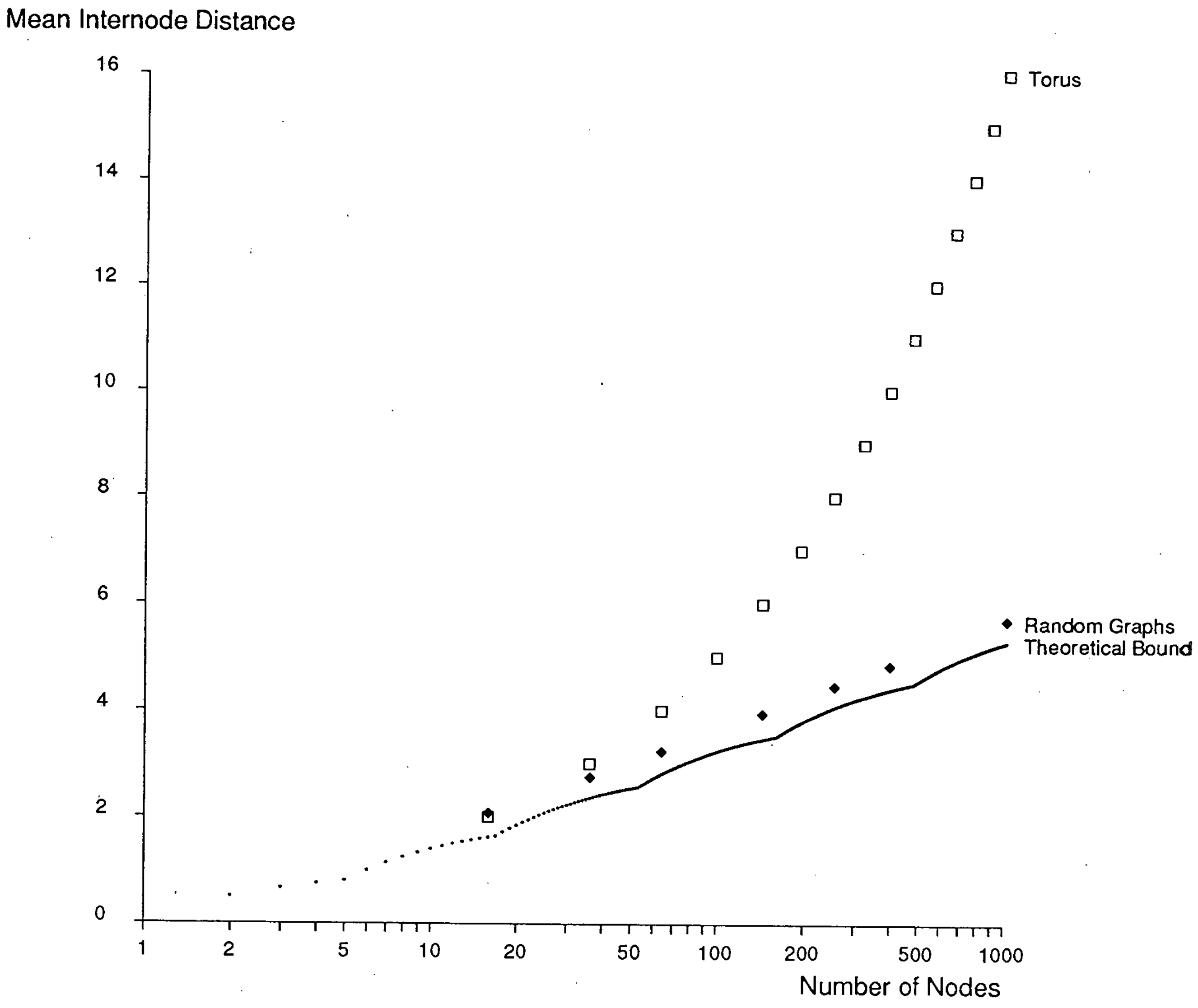


Figure 4: Random Graphs, Moore Graphs and Tori

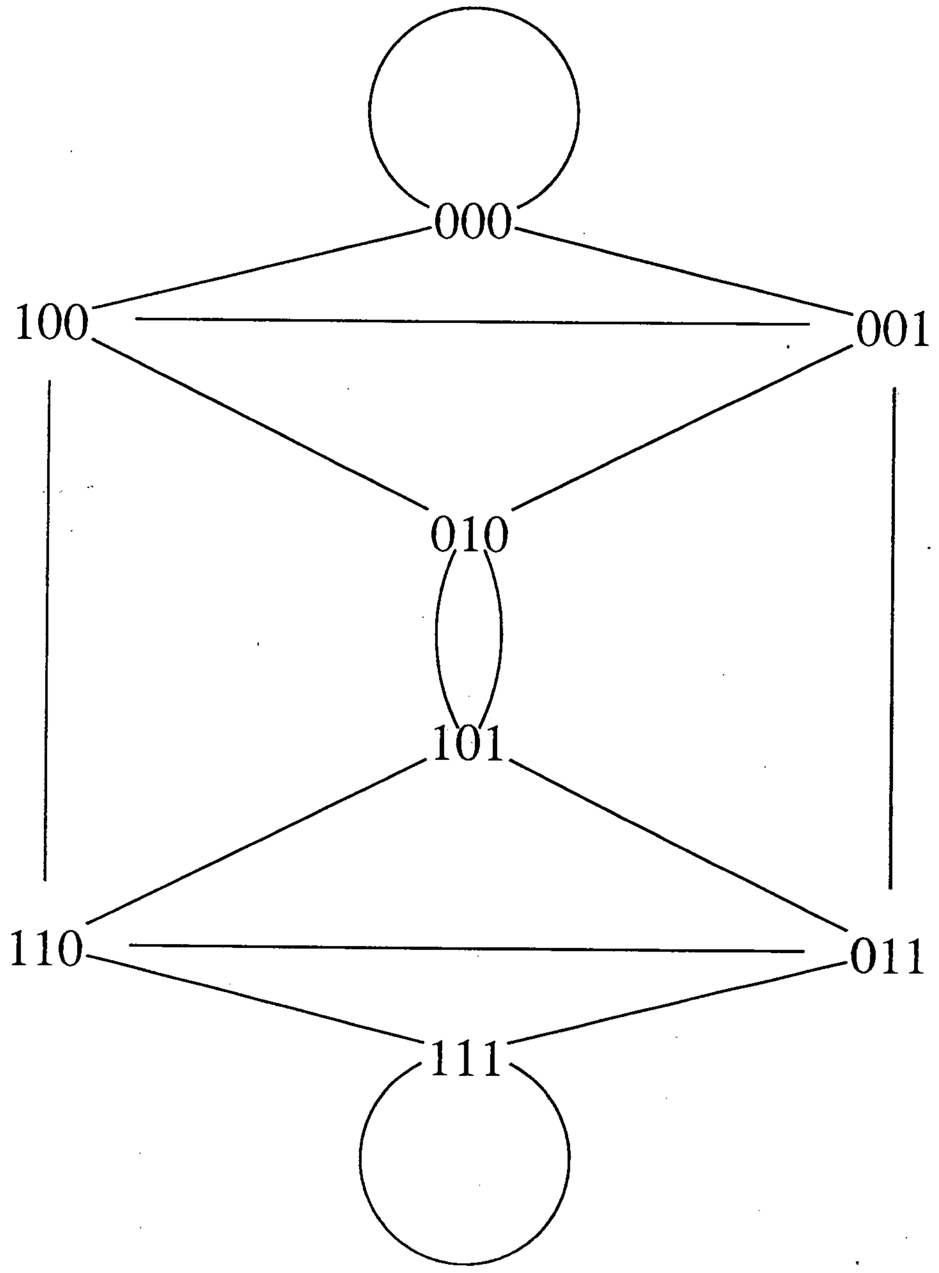


Figure 5: A de Brujn Graph

describing them involves labelling each node with a number in the range 0 to $b^k - 1$, expressed in base b. The de Brujn graph is then constructed by connecting to node α all those nodes whose labels can be obtained by shifting α by one place (left or right) and filling the "spare" digit created with any base b digit. The example in figure 5 uses arity 4, giving binary coding, and takes $k = 3$ for $2^3 = 8$ nodes. The links are shown in the table below:

node	linked to			
000	000	100	000	001
001	000	100	010	011
010	001	101	100	101
011	001	101	110	111
100	010	110	000	001
101	010	110	010	011
110	011	111	100	101
111	011	111	110	111

The regularity of de Brujn graphs is a different kind of regularity from that previously described, but does not permit a simple rule-based routing strategy to be devised. It can be seen from figure 6 that while de Brujn graphs are more compact than the regular lattices examined, they are very much less compact than random graphs.

10 Optimising Graphs

Before examining other measures, it will be appropriate to introduce optimisation schemes which might allow the construction of graphs with even better properties than the "random" graphs. Three techniques will be examined—a "greedy" algorithm, a technique for generating 2-opt graphs (see below) and a genetic algorithm. In all cases the objective functions for the optimisation will reward compactness alone, and will in fact be the mean inter-node distance ι. The most-loaded link- and node-loads will then be calculated, and will be found to be acceptably low. The three techniques are described in turn.

10·1 'Greedy' Graphs

"Greedy" optimisation techniques are usually described as "taking the best they can at each step and never back-tracking", after the "greedy algorithm" for tackling the TSP (Lawler *et al* (1985)). Thus, in constructing a graph K using a greedy algorithm the aim will be (loosely) to add at each step the link which maximally reduces $\iota(K)$. The greedy algorithm can be implemented in the following way:

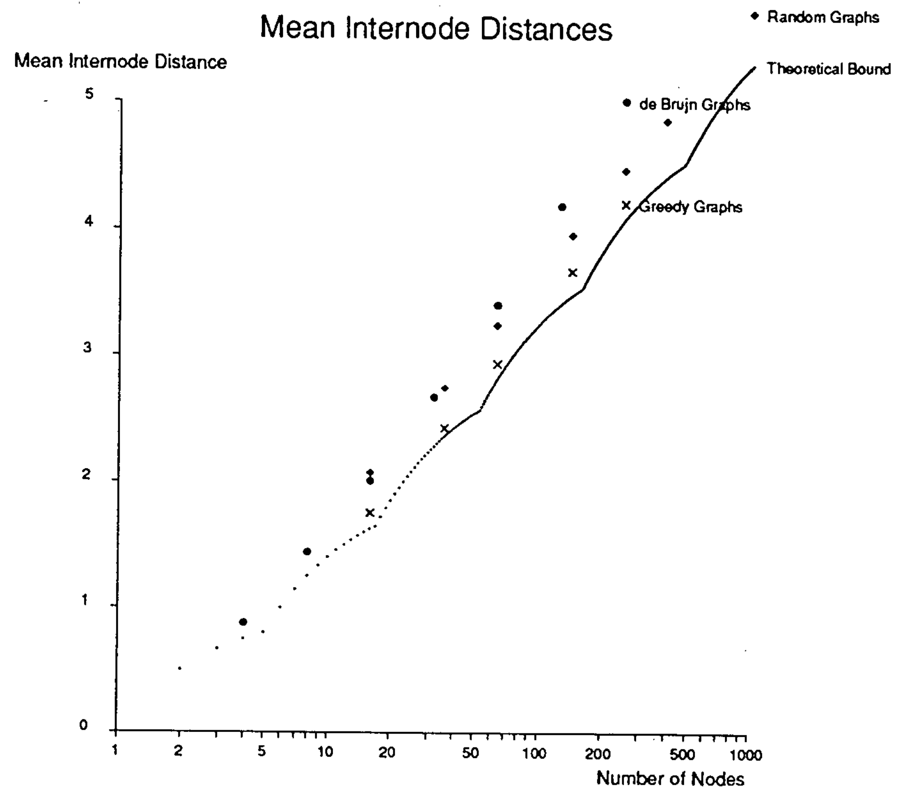


Figure 6: de Brujn & Greedy Graphs

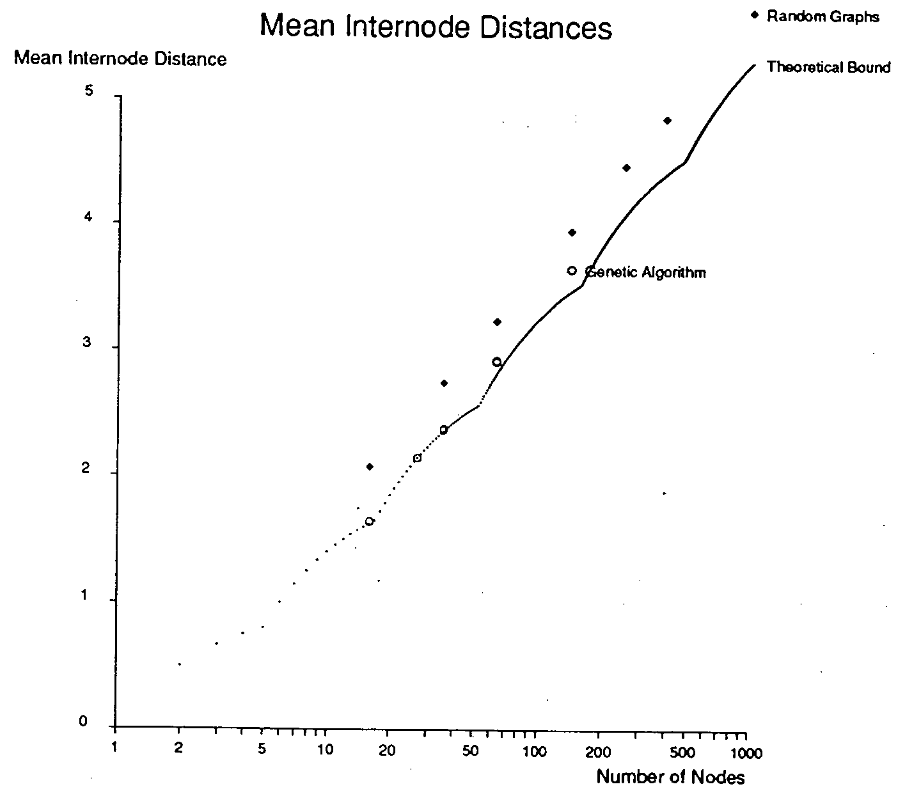


Figure 7: Optimised Graphs

The Greedy Algorithm

Phase I: (Produce a connected graph)

1° Begin with a graph having all the nodes present but no links.

2° Select any node and label it n.

3° While there are still nodes which are not connected, repeatedly select any one of these, connect it to n, and reassign the label n to the new node just connected.

Phase II: (attempt to compact the graph)

4° Select any pair of nodes which have maximum separation on the graph under construction, and which both have at least one spare link, and make a connection between these nodes.

5° Repeat step 4 until all links have been allocated.

It should be clear that this algorithm reduces the diameter of the graph under construction, $\iota(K)$, by the maximum possible amount at each step, but that this in no way guarantees the construction of graphs with optimal values for $\iota(K)$.

10·2 k-opt Graphs

A graph is said to be k-opt with respect to some objective function if there is no permutation of k links which improves the performance of the graph with respect to this function (Lin (1965)). In particular, a graph is 2-opt if there is no transposition of links which improves its performance. Various algorithms exist for producing 2-opt graphs. The simplest involves starting from an arbitrary graph and repeatedly testing transpositions, accepting any which improve the performance of the graph until the graph becomes 2-opt. The testing order is unimportant in the sense that as long as every node is tried, the algorithm will be guaranteed to generate 2-opt graphs. Different testing schedules may, however, produce graphs of different characteristic goodness. The implementation used for this work enumerated all possible transpositions and evaluated them in order. Given n links there are $n(n-1)/2$ possible transpositions so the algorithm is complete when there have been $n(n-1)/2$ unsuccessful attempts at transposition.

10·3 Genetic Algorithm

The use of a genetic algorithm was by far the most ambitious optimisation scheme attempted in this work. An implementation of a genetic algorithm for topology optimisation already existed in Edinburgh, written by Norman (1988b). This program was used in this study, with limited success, and it was in part experiences with this genetic algorithm that inspired the investigation of representation problems in genetic algorithms detailed in chapter 3.

A number of features distinguished the genetic algorithm in question from the standard algorithms described in chapter 3. The more significant differences were:

- *employment of non-standard genetic operators.* In particular, the crossover operator exchanged *homologous subgraphs* between parents. Details of this and the other operators can be found in Norman (1988b).

- *division of the population into isolated sub-populations.* Each sub-population occupied on a separate processor on the ECS. This was an early implementation of an idea now gaining much currency within the community of workers on genetic algorithms. In addition to providing a particularly convenient way of implementing genetic algorithms on MIMD machines, this allows different populations to explore different parts of the search-space independently of each other. As is observed in living systems, when populations are isolated from each other they generally develop differently, and this encourages a more thorough search of the space. The aim is not, however, simply to implement n small runs of the same problem on n different processors of the machine, but rather to have the different processors cooperating on one large optimisation. For this reason a solution (chromosome) from one processor occasionally "migrates" to a new processor where it displaces a member of its sub-population. This ensures that the information gained during isolated development on the various nodes is eventually shared across the network.

- *ranking of solutions.* Instead of reproducing in strict proportion to fitness, a ranking of solutions on the basis of fitness is used. The probability of reproduction for a chromosome η in a population of size n is then taken to be
$$P(\eta) = \sum_{i=0}^{\infty} p_s^{R(\eta)+in}$$
where $R(\eta)$ is the rank of η and p_s is the so-called "stepping probability". (The idea is to step through the list, selecting each element with probability p_s.)

It was stated earlier that finding a suitable respresentation for the problem to be studied using a genetic algorithm was the key determinant in its success. Equivalently, the task can be seen as one of finding suitable genetic operators, for it is the interaction between the operators and the chromosomes which determine the meaning of "schemata", which in turn determine the effectiveness of the algorithm.[31] Of

[31] more carefully, this interaction controls the way in which the genetic algorithm explores the space, and thus determines which set(s) of equivalence relations are most useful in analysing the dynamics of the system.

course, if non-standard operators are in use it may well be that schemata also have to be redefined. Using a genetic algorithm for topology optimisation raises all of these difficulties in fairly acute form.

In order to embark on a serious discussion of the application of genetic algorithms to this problem, it would be necessary first to discuss in some detail the Quadratic Assignment Problem (QAP), of which the TSP is an instance, for this is in many ways a simpler form of the transputer configuration problem. This in itself would require many pages and this discussion is therefore reserved for another forum.

11 Results

Some results for mean-internode distance have already been presented. In this section further results are collated which begin to justify the claim that regular graphs are a less than ideal choice for many applications. Unless otherwise stated, results are for arity 4.

11·1 Mean Internode Distance and Diameter

The graphs in figure 6 and figure 7 give an indication of the performance of the various optimisation schemes considered when using mean internode distance as the objective function. It can be seen that while de Brujn graphs diverge very much more slowly from the "ι-bound" than the regular lattices, they still have mean internode distances very much greater than graphs constructed randomly by algorithm $\mathfrak{A}$.

When using diameter to assess the quality of solutions, tori can be seen from figure 8 to perform even less well. The figures for "greedy" graphs are averages over twenty runs, and the comparison shown uses the "D-bound" derived above and the exact results for a tori.

11·2 Binary Hypercubes

It is often claimed that binary hypercubes are unusually compact, but the results shown in figure 9 for diameter and figure 10 for mean internode distance show that this is misleading: large binary hypercubes have high arity, and the graphs show that if the many links emanating from each node are connected at random (following algorithm $\mathfrak{A}$) very much more compact graphs can be constructed.

11·3 Link Loading

The graphs in figure 11 show the link loads in "greedy" graphs, and compares these to the loads on square tori with the same numbers of arity-4 nodes. The communications pattern assumed is that each processor communicates once per cycle with every processor in the system. Thus given N processors there are N^2 communications per unit time step. The y-axis shows the number of links in the system which suffer the

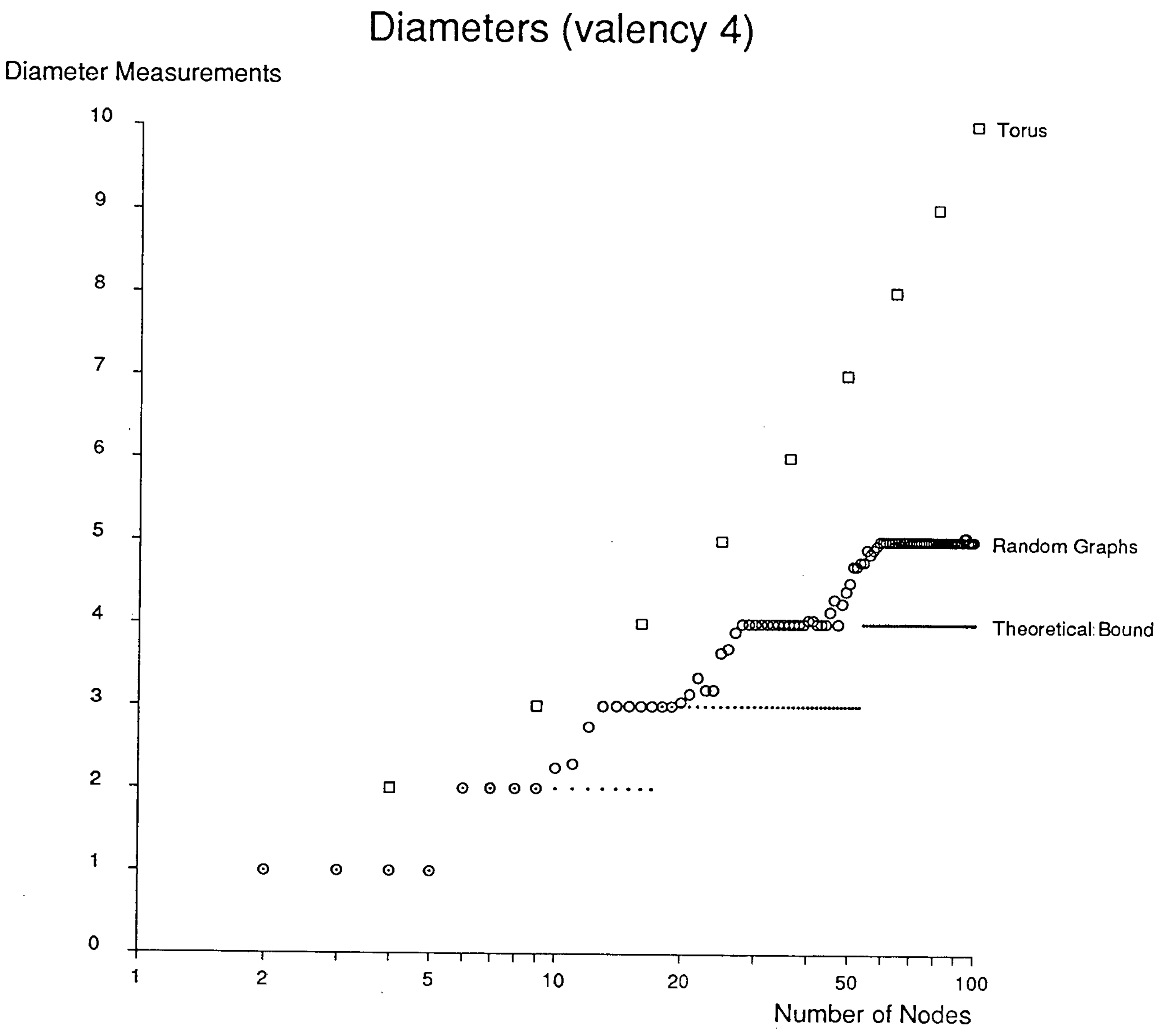


Figure 8: Tori and Greedy Graphs (diameters)

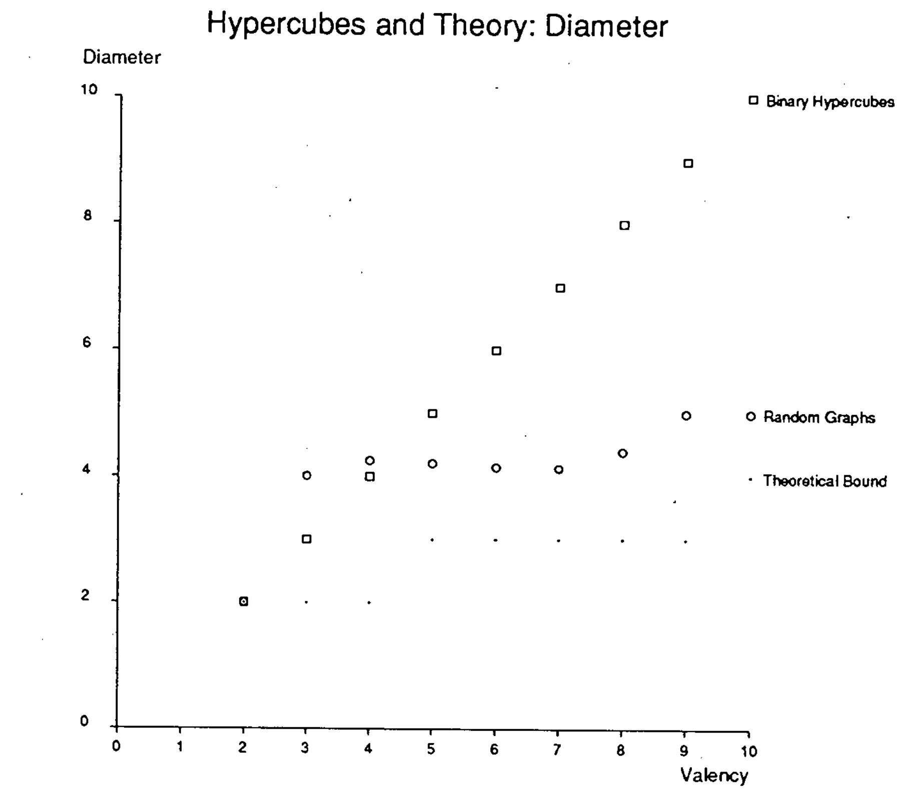


Figure 9: Binary Hypercubes (diameter)

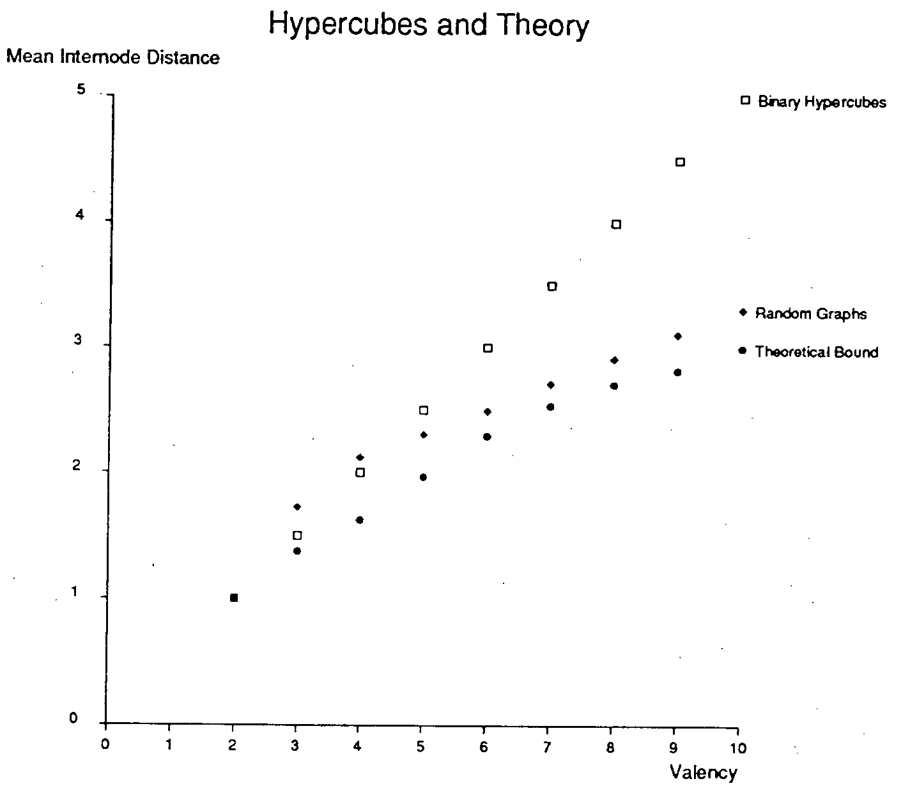


Figure 10: Binary Hypercubes (mean internode distance)

load shown on the x-axis—the number of messages which pass through a link under the random routing strategy $\mathfrak{R}$. Three different system sizes are shown (36, 64 and 100 nodes) and it can be seen that while the loads in the "greedy" graphs are non-uniform, even for the smallest graph no link suffers a load greater than that which *every* link carries on the torus. Moreover, link loads degrade very much more slowly for the "greedy" graphs as the number of nodes rises: for a 100-node graph even the most heavily-loaded link carries around 25% less traffic than each link in the corresponding torus.

11·4 Node Loading

The results for node loading under similar conditions are shown in figure 12, and follow a very similar pattern. Even for 36 processors the highest node load on the random graphs is noticeably smaller than the load on the corresponding torus.

11·5 Narrowness

An obvious question to ask about the irregular graphs discussed here is whether there is not some group of nodes which has a very poor communications-to-calculations ratio, and is thus "narrow" in the sense defined in section 4. A result was derived earlier which showed that narrowness can be bounded in terms of the most-loaded link-load. The table below shows a comparison between the actual narrowness of a torus for various number of nodes, and the *bound* on narrowness for the graphs produced by the "greedy" algorithm, and confirms that narrowness is not a problem with these graphs. As usual, arity 4 is used for comparison

Nodes	Torus	Greedy
36	1.50	1.45
64	2.00	1.81
200	2.50	2.04

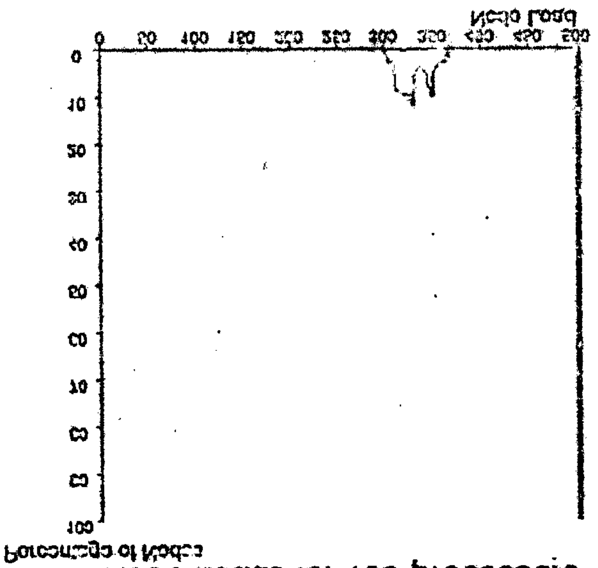
It might be expected that the actual narrowness of the "greedy" graphs (which is hard to calculate) would be rather even lower than these bounds.

12 Applications

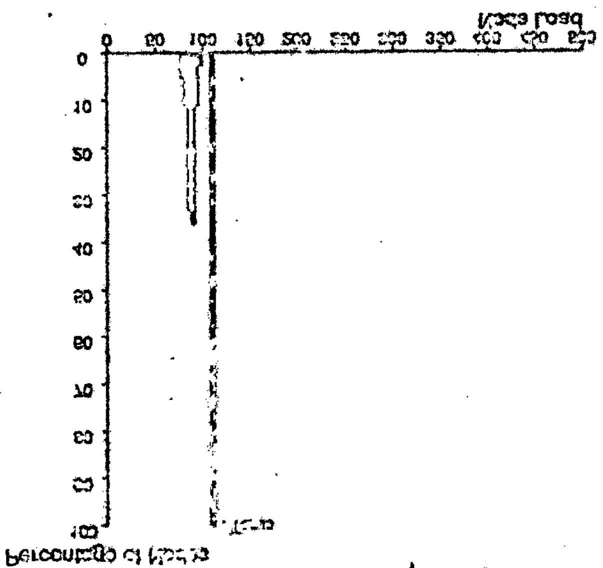
12·1 Speed Increase in a Real Application

An existing application (Norman (1988a)) which used a topology-independent message-passing system (Norman (1988c)) was taken and loaded onto a 2-opt hamiltonian graph with respect to mean interprocessor distance. The program had been designed to run on a helical torus, but despite this achieved an immediate speed increase of around 10% on the most computer-intensive part of the calculation in the

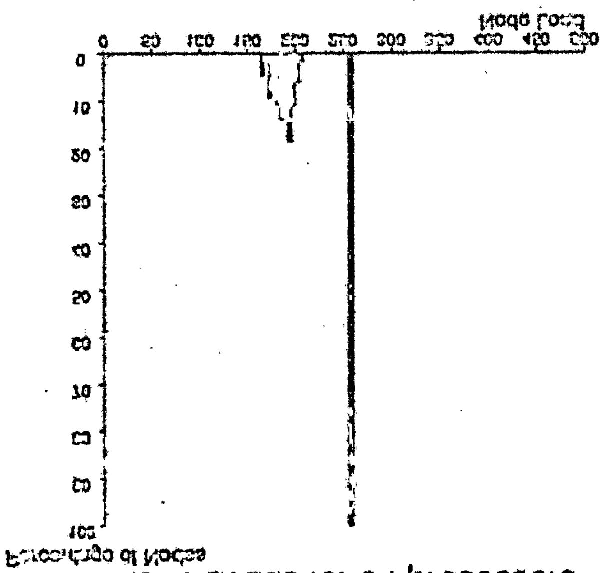
Figure 12: Node Loads



Node Loads for 100 processors



Node Loads for 36 processors



Node Loads for 64 processors

program. (The overall speed-up was greater.)

This is an extraordinary result, for the network used consisted of only 64 processors, and the mean internode distance for the 2-opt graph was only some 30% less than for the helical torus. Moreover, the same object code was used in each case: a topology independent harness allows speed improvements such as this to be gained immediately upon finding a better graph.

Whereof one cannot speak,
thereon one must remain silent.
— LUDWIG WITTGENSTEIN.

9 Appendix

1 General and Set-Theoretic Notation

$f : A \longrightarrow B$	a mapping with domain A and co-domain B
$\{f(k) \mid g(k)\}$	'the set of $f(k)$ *for which* $g(k)$'
$\|A\|$	the number of elements in a set A
$\in$	'is an element of'
$A \subset B$	'A is a subset of B' (not necessarily proper!)
$A \supset B$	'A is a superset of B' (not necessarily proper!)
$A \cup B$	'the union of A and B'
$\bigcup A$	'the union of all sets $B \subset A$
$A \cap B$	'the intersection of A and B'
$\bigcap A$	'the intersection of all sets $B \subset A$
$\vee$	'or' (disjunction)
$\wedge$	'and' (conjunction)
$\emptyset$	the empty set { }
$\triangleq$	'is defined to be equal to'
$\Longrightarrow$	'implies'
$\equiv$	'is identically equal to' (for all values)
$\longmapsto$	'maps to'
$\sim$	'is equivalent to' ('twiddles')
$\exists$	'there exists' (the existential quantifier)
$\forall$	'for all' (the universal quantifier)
$:$	'it is the case that' or 'maps'
$\overline{\operatorname{sgn}}(x)$	'top-heavy' sign function
$\mathcal{P}_n$	the set of permutations of n objects
$\mathsf{P}(A)$	the 'power set' (set of all subsets) of A
$\mathbb{R}$	the set of real numbers
$\mathbb{R}^+$	the set of positive real numbers
$\mathbb{R}_0^+$	$\mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}$
$\mathbb{Z}$	the integers
$\mathbb{Z}^+$	the positive integers $\{1, 2, \ldots\}$
$\mathbb{Z}_0^+$	$\mathbb{Z}_0^+ = \mathbb{Z}^+ \cup \{0\}$
$P(X)$	the probability of event X
$g\|_S$	the restriction of f to (domain) S

2 Neural Networks

$\mathfrak{B}$	the back-propagation learning algorithm
c	the connection mask for a network
c_{ij}	1 if node i is connected to node j
d	decomposition of the set of nodes
e_j	the jth external node in a network
E	the set of external nodes $E = I \cup O$
$\mathcal{E}$	the global error (over *all* patterns)
F_j	the set of nodes i for which $c_{ij} = 1$
h_j	the jth hidden node in a network
$h^{(b)}$	step size for a bias in gradient descent
$h^{(w)}$	step size for a weight in gradient descent
H	the set of hidden nodes in a network
I	the set of input nodes in a network
i_j	the jth input node
N	the set of nodes in a network
$\mathcal{N}$	a neural network (as a function)
O	the set of output nodes in a network
o_j	the jth output node
p	an input pattern for a network
t	a target output pattern for a network
$\mathcal{T}$	a training set (of pattern-target pairs)
V	the set of activation functions
v_i	the activation function for node i
W	the set of weights in a network
w_{ij}	the weight from node i to node j
Z	an index set for the hidden nodes
α	(fractional) momentum term in back-propagation
ε	the network error for one pattern
π	a permutation of hidden node labels
ϕ_j	the potential at the jth node
τ_j	the bias for node j
Ω	the space of all neural networks

3 Genetic Algorithms

■	a gene with specified value
□	a gene with any value
$\bowtie$	'is compatible with'
$\sim$	an equivalence relation
$\eta \oplus \zeta$	the similarity set of η and ζ
$\mathcal{A}$	a set of control parameters for an operator
a	a member of a control set $\mathcal{A}$
$\mathfrak{B}(t)$	the population at timestep t
$\mathcal{C}$	the set of all chromosomes (in some domain)
g	the genotype $\longrightarrow$ phenotype decoding function
$\mathcal{G}_i$	set of alleles for the ith gene
$\mathcal{G}_i^\star$	$\mathcal{G}_i^\star = \mathcal{G}_i \cup \{\square\}$
I	the inversion operator
$\mathcal{I}$	an interval in $\mathbb{R}$
$\ell(\xi)$	the definition length of a schema ξ
L	random-relinking operator
M	the general mutation operator
$n_\xi(t)$	the number of instances of a schema or forma ξ at time t
N_o	the number of genetic operators
Γ_i	the ith genetic operator
o	the order of a schema
p_X	probability of crossover (each reproduction)
p_I	probability of inversion (each reproduction)
p_M	point mutation rate
p_s	"stepping" probability for rank-based selection
$\mathfrak{P}$	a type of reproductive plan
Q	the set of optima in $\mathcal{C}$
R	the rank of a chromosome in a population
R^3	random, respectful recombination
s	a structure in $\mathcal{S}$
$\mathcal{S}$	the search space of all structures
u	utility function for chromosomes
X	a crossover operator
δ_i	the ith detector
Δ	the diversity of a population

η	a chromosome
η_i	the ith gene on chromosome η
ζ	a chromosome
μ_ξ	utility of schema or forma ξ
$\hat{\mu}_\xi(t)$	mean utility of instances of ξ in population at time t
$\bar{\mu}(t)$	mean utility of population at timestep t
ξ	a schema or forma (equivalence class)
Ξ	the set of formae or schemata
π^η	the linkage information for a chromosome η
ρ	the representation (or coding) function
$\sigma(\eta,\zeta)$	the similarity of two chromosomes
Ψ	the set of equivalence relations inducing schemata or formae
υ	the view of an optimum
Υ	the view of the set of optima
ω	an optimum in Q

4 Graphs

$\lvert K\rvert$	number of nodes in a graph (= $\lvert K^\star\rvert$)
a	the arity (number of links to) a node
$\mathfrak{A}$	Algorithm for constructing 'random graphs'
D	the diameter of a graph
K	a graph
$K^\star$	set of nodes in graph K
$\mathcal{K}(a,n)$	the set of graphs of arity up to a and having size n
$\mathcal{K}_\infty$	the set of graphs of arbitrary arity and size
$\mathcal{L}[K]$	number of external links in a graph K
$\mathfrak{R}$	the random routing strategy
$s(i,j)$	graph-theoretic distance between nodes i and j
S	a subgraph
$\mathbb{S}(K)$	the set of all subgraphs of a graph K
$\mathbb{S}_\bullet(K)$	subgraphs S of a graph K with $\lvert S\rvert \le \frac{1}{2}\lvert K\rvert$
ι	the mean inter-node distance for a graph
κ	the most-loaded node load
λ	most-loaded link load
$\nu(K)$	the narrowness of a graph

5 Equivalance Relations

DEFINITION (equivalence relation)

An a relation $\sim$ over a set A is said to be *equivalence relation* if it satisfies the following three conditions:

Reflexivity:	$\forall a \in A,$	$a \sim a.$
Symmetry:	$\forall a, b \in A,$	$a \sim b \Longrightarrow b \sim a.$
Transitivity:	$\forall a, b, c \in A,$	$a \sim b \wedge b \sim c \Longrightarrow a \sim c.$

EXAMPLE (equivalence relation)

The $=$ relation (equality) is an equivalance relation over the set of real numbers $\mathbb{R}$.

Proof

Reflexivity:	$\forall a \in \mathbb{R},$	$a = a.$
Symmetry:	$\forall a, b \in \mathbb{R},$	$a = b \Longrightarrow b = a.$
Transitivity:	$\forall a, b, c \in \mathbb{R},$	$a = b \wedge b = c \Longrightarrow a = c.$

Q.E.D.

6 Logical Conventions

Because conventions vary in logic, those adopted here will be described briefly. The general form of a quantified expression is

$$Q(x_i) : S(x_i)$$

where $Q(x_i)$ quantifies some variables x_i and $S(x_i)$ is a predicate which is true under the conditions specified by $Q(x_i)$. For example,

$$\forall n \in \mathbb{Z}\, \exists n' \in \mathbb{Z} : n' = n + 1$$

reads 'for every integer n there exists an integer n' for which it is the case that $n' = n + 1$'. All of the logical statements in this work have this general form, the quantification preceding the colon and the predicate following it. The only extra complication arises with qualifiers to the quantification, which are put in brackets. For example,

$$\forall n \in \mathbb{Z}^+ \ (n \neq 1)\, \exists n' \in \mathbb{Z}^+ : n = n' + 1$$

asserts that every positive integer with the exception of 1 is the successor to some other positive integer. The exception need not be rare, so that it would also be quite acceptable to say

$$\forall n \in \mathbb{Z}^+ \ (n \ even)\, \exists n' \in \mathbb{Z}^+ : n = 2n',$$

though this is tautological to an absurd extent.

10 Bibliography

Axelrod (1987) Axelrod, Robert, *The Evolution of Strategies in the Iterated Prisoner's Dilemma*, in *Genetic Algorithms and Simulated Annealing*, Pitman (London) 1987.

Bagley (1967) Bagley, J. D., *The Behaviour of Adaptive Systems Which Emply Genetic and Correlation Algorithms*, Ph.D. thesis, (Ann Arbour: Univerity of Michagan) 1967.

Grefenstette & Baker (1989) Grefenstette, John J., & Baker, James E., *How Genetic Algorithms Work: A critical Look at Intrinsic Parallelism*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

Baker (1985) Baker, James Edward, *Adaptive Selection Methods for genetic Algorithms*, in *Proceedings of an International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates (Hillsdale) 1985.

Baker (1987) Baker, James Edward, *Reducing bias and inefficiency in the selection algorithm*, in *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates (Hillsdale) 1987.

Belew, McInerney & Schraudolph (1990) Belew, Richard K., McInerny, John, & Schraudolph, Nicol N., *Evolving Networks: Using the Genetic Algorithm with Connectionist Learning*, CSE Tech. Report #CS90–174, UCSD (La Jolla) 1990.

Bethke (1981) Bethke, A. D., *Genetic Algorithms and Function Optimizers*, Ph.D. thesis, (Ann Arbour: Univerity of Michagan) 1970.

Caruna & Shaffer (1985) Caruna, Richard A. & Shaffer, J. David, *Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms* in *Proceedings of the 5th International Conference on Machine Learning*, Morgan Kaufmann (Los Altos) 1988.

Cavicchio (1970) Cavicchio, D. J., *Adaptive Search Using Simulated Evolution*, Ph.D. thesis, (Ann Arbour: Univerity of Michagan) 1970.

Cohoon *et al* (1987) Cohoon, J. P., Hegde, S. U., Martin, W. N., Richards, D., *Punctuated equilibria: a parallel genetic algorithm* in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates (Hillsdale, New Jersey) 1987.

Davis (1989) Davis, Lawrence, *Adapting Operator Probabilities in Genetic Algorithms*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann

(San Mateo) 1989.

Dawkins (1988) Dawkins, Richard, *The Blind Watchmaker*, Penguin (Harmondsworth) 1988.

Dawkins (1982) Dawkins, Richard, *The Extended Phenotype*, Oxford University press (Oxford) 1982.

Dawkins (1978) Dawkins, Richard, *Replicator Selection and the Extended Phenotype*, Zeitschridt für Tierpsychologie **47**.

de Brujn (1946) de Brujn, N. G., *A Combinatorial problem*, in *Koninkl Nederl. Acad. Wetensch. Proc. Ser. A*, **49** (1946).

De Jong (1975) De Jong, Kenneth A., *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*, Doctoral Dissertation, University of Michigan, 1975.

De Jong (1980) De Jong, Kenneth A., *A Genetic-Based Global Function Optimization Technique*, Technical Report 80-2, University of Pittsburgh, 1980.

Dodd (1989) Dodd, Nigel, *Optimisation of Network Structure using Genetic Techniques*, Royal Initiative in Pattern Recognition (RIPR) at RSRE, Malvern, UK, RIPREP/1000/63/89 (Malvern) 1989.

Eshelman (1990) Eshelman, Larry J., personal communication.

Eshelman *et al* (1989) Eshelman, Larry J., Caruna, Richard A. & Shaffer, J. David, *Biases in the Crossover Landscape*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

Fox & Furmanski (1989) Fox, G., & Furmanski, W., *Philosophical Transactions of the Royal Society*, **A** volume 326 (1988).

Franz (1970) Franz, D. R., *Non-linearities on Genetic Adaptive Search*, Ph.D. thesis, (Ann Arbour: Univerity of Michagan) 1972.

Frean (1989) Frean, Marcus *The Upstart Algorithm: a method for constructing and training feed-forward neural networks*, Edinburgh Preprint (1989).

Goldberg (1989) Goldberg, D. E. *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley (Reading, Mass) 1989.

Goldberg & Lingle (1985) Goldberg, D. E. & Lingle Jr, Robert, *Alleles, Loci and the Travlling Salesman Problem*, in *Proceedings of an International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates (Hillsdale) 1985.

Goldberg & Richardson (1990) Goldberg, David E., & Richardson, Jon, *Genetic Algorithms for Multimodal Function Optimisation*, in *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates (Hillsdale) 1987.

Gorges-Schleuter (1989) Gorges-Schleuter, Martina, *ASPARAGOS Asn Asynchronous Parallel Genetic Optimization Strategy*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

Grefenstette (1986) Grefenstette, J. J., *Optimization of Control Parameters for Genetic Algorithms* in *IEEE Transations on Systems, Man & Cybernetics* 16-1, 1986.

Grefenstette *et al* (1986) Grefenstette, John, Gopal, Rajeev, Rosmaita, Brian, & Van Gucht, Dirk, *Genetic Algorithms for the Travelling Salesman Problem*, in *Proceedings of an International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates (Hillsdale) 1985.

Harp, Samad & Guha (1989) Harp, Steven Alex, Samad, Tariq & Guha, Aloke, *Towards the Genetic Synthesis of Neural Networks*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

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

Hopfield (1982) Hopfield, J. J., Proc. Natl. Acad. Sci. USA B 1 3088 (1984).

Hopfield & Tank (1984) Hopfield J. J. & Tank, D. *Neural Computation on Decisions in Optimisation Problems*, Biol. Cyber. 52 (1984).

Kolen & Pollack (1990) Kolen, John F., & Pollack, Jordan B., *Backpropagation Is Sensitive to Initial Conditions*, in *Complex Systems*, 4 3 (1990).

Lawler *et al* (1985) Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B., *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimisation*, Wiley 1985.

Levy & Gomez (1985) Levy. Alejandro V. & Gomez Susana, *The Tunnelling Algorithm applied to Global Optimization*, in *Numerical Optimization 1984*, Boggs, Byrd and Schnabel, eds., p. 213-244. (SIAM 1985).

Lin (1965) Lin, S. *Computer Solutions of the Travelling Salesman Problem*, Bell. Syst. Tech. J., 2245–2269 (1965).

Little (1974) Little, W. A., Math. Biosci., **19**, 101 (1974).

Mezard & Nadal (1989) Mezard, M., & Nadel, J. *Learning in Feedforward Layered Networks: the tiling algorithm*, Journal of Physics A 22 12.

Miller, Todd & Hegde (1989) Miller, Geoffrey F., Todd, Peter M., & Hegde, Shailesh U., *Designing Neural Networks using Genetic Algorithms*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

Minsky & Papert (1988) Minsky, Marvin L., & Papert, Semour A, *Perceptrons*, MIT Press (Michigan) 1988.

Mjolsness, Sharp & Alpert (1988) Mjolsness, Eric, Sharp, David H., & Alpert, Bradley K., *Scaling, Machine Learning and Genetic Neural Nets*, Los Alamos National Laboratory (LA-UR-88-142) 1988.

Montana & Davis (1989?) Montana, David J., & Davis, Lawrence, *Training Feedforward Neural Networks Using Genetic Algorithms*, in *Machine Learning*.

Moore (c.1956) Moore (c.1958).

Mühlenbein (198?) Mühlenbein, H. *Adaptation in Open Systems: Learning and Evolution*.

Mühlenbein (1989) Mühlenbein, H., *Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

Mühlenbein & Kindermann (1989) Mühlenbein, H, & Kindermann, J., *The Dynamics of Evolution and Learning — Towards Genetic Neural Networks*, in *Connectionism in Perspective*, Pfiefer, P., Schreter, Z., Fogelman-Soulié, F., & Steels, L. (eds), North-Holland, 1989.

Norman (1988a) Norman, M. G., & Fisher, R. B., *Surface Tracking Within Three-Dimensional Datasets using a Generalised Message-Passing Sub-system*, in *Proceeding of the 8th Technical Meeting of the Occam User Group*, (Amsterdam) 1988.

Norman (1988b) Norman, Michael, *A Genetic Approach to Topology Optimisation for Multiprocessor Architectures*, Edinburgh Preprint (1988).

Norman (1988c) Norman, M. G., & Wilson, S. *TITCH: Topology Independent Transputer Communications Harness*, Edinburgh Concurrent Supercomputer Project User Note, 1988.

Radcliffe (1989) Radcliffe, Nicholas J., *Early Clustering Around Optima*, Edinburgh Preprint (1989).

Radcliffe (1988) Radcliffe, Nicholas J., *The Permutation Problem*, unpublished manuscript.

Richards (1988) Richards, G. D., *Documentation for Rhwydwaith*, Edinburgh Concurrent Supercomputer Project, ECSP-UG-7 (1988).

Rosenblatt (1962) Rosenblatt, F. *Principles of Neurodynamics*, Spartan Books, New York (1962).

Rumelhart, Hinton & Williams (1986) Rumelhart, D. E., Hinton, G. E., & Williams, R. J., *Learning Representations by Back-Propagating Errors*, in *Nature* 323 1986.

Shaefer (1989) Shaefer, Craig, G., *The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique* in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

Shaffer *et al* (1990) Shaffer, J. David, Caruna, Richard A., Eshelman, Larry J., *Using genetic Search to Exploit the Emergent Behaviour of Neural Networks*, in *Proceedings of the Conference on Emergent Computation*, Physica D (1990).

Shaffer *et al* (1989) Shaffer, J. David, Caruna, Richard A., Eshelman, Larry J., Das, Rajarshi, *A Study of the Control Parameters Affecting Online Performance of Genetic Algorithmsfor Function Optimisation*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

Seitsma & Dow (1988) Seitsma, J. & Dow, R. J. F., *Neural Net Pruning — Why and How*, in *Proceedings of the IEEE Conference on Neural Networks*, vol. II 1988.

Syswerda (1989) Syswerda, Gilbert, *Uniform Crossover in Genetic Algorithms*, in *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (San Mateo) 1989.

Śmieja (1989) Śmieja, Frank J., *Learning and Generalisation in Feed-Forward Neural Networks*, Doctoral Thesis, UNiversity of Edinburgh (1989).

Tanese (1987) Tanese, Reiko, *Parallel Genetic Algorithm for a Hypercube*, in *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates

(Hillsdale) 1987.

Tollenaere (1990) Tollenaere, Tom *SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties,* in *Neural Networks,* (to appear) 1990.

Whitley (198?) Whitley, Darrell, *Applying Genetic Algorithms to Neural Network Problems: A Prliminary Report,* unpublished manuscript (198?).

Whitley (1987) Whitley, Darrell, *Using Reproductive Evaluation to Improve Genetic Search and Heuristic Discovery,* in *Proceedings of the Second International Conference on Genetic Algorithms,* Lawrence Erlbaum Associates (Hillsdale) 1987.

Whitley (1989) *The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reprodutive Trials is Best,* in *Proceedings of the Third International Conference on Genetic Algorithms,* Morgan Kaufmann (San Mateo) 1989.

Whitley, Starkweather & Bogart (1989) Whitley, Darrell, Strakweather, Timothy & Bogart, Christopher *Genetic Algoriths and Neural Networks: Optimising Connections and Connectivity,* Colorado State university Technical Report CS-89-117 (1989).

Whitley & Hanson (1989a) Whitley, Darrell, & Hanson, Thomas, *The Genitor Algorithm: Using genetic Algorithms to Optimize Neural Networks,* Colorado State University technical Report CS-89-107 (1989).

Whitley & Hanson (1989b) Whitley, Darrell, & Hanson, Thomas, *Optimizing Neural Networks Using Faster, More Accurate Genetic Search,* in *Proceedings of the Third International Conference on Genetic Algorithms,* Morgan Kaufmann (San Mateo) 1989.

Wittenberg & Belew (1990) Wittenberg, G, & Belew, R. K., *Simulated Genetic Search finds Rapidly Trainable Neural Networks,* Science (submitted) 1990.

Yao & Young (1989) Yao & Young, *Dynamic Tunnelling Algorithm for Global Opbimization,* IEEE Trans. SMC, in press 1989.