Published in April 2006 by
The Old Inn Publishing
www.theoldinn.org

Order online via
www.lulu.com/theoldinn

THE UNIVERSITY OF SUSSEX

The Artificial Evolution of Adaptive Behaviour

Inman Harvey

Submitted for the degree of D. Phil.

September 1993

revision April 1995

CONTENTS

LIST OF FIGURES

LIST OF TABLES

THE UNIVERSITY OF SUSSEX

The Artificial Evolution of Adaptive Behaviour

Submitted for the degree of D. Phil.

September 1993

revision April 1995

Inman Harvey

ABSTRACT

A methodology is presented for the design through artificial evolution of adaptive complex systems, such as the control systems of autonomous robots.

Genetic algorithms have largely been tailored towards optimisation problems with a fixed and well-defined search-space; the SAGA (Species Adaptation Genetic Algorithms) framework is introduced for the different domain of long term artificial evolution, where the task domain is ill-defined and can increase in complexity indefinitely. Genotypes should be able to increase in length indefinitely, and evolution will take place in a genetically converged population. Significant changes from normal genetic algorithm practice follow from this.

It is shown that changes in genotype length should be restricted to gradual ones. Appropriate mutation rates are proposed to encourage exploration of the high-dimensional fitness landscape without losing gains already made. Tournament selection, or similar ranking methods, are advocated as a way of maintaining selection pressures at a known rate. A crossover algorithm is introduced, which allows for recombination of genotypes of different lengths without undue confusion. The significance of a developmental process from genotype to phenotype, of co-evolution and of neutral drift through genotype space, are discussed.

As a class of control systems appropriate for evolution, programming languages are dismissed in favour of realtime dynamical recurrent connectionist networks; issues of time, representation and learning in such networks are discussed. A whole complex system, comprised of such a network together with sensory and motor systems, is characterised as a dynamical system with internal state, coupled to a dynamical environment.

Applications of these theoretical frameworks of artificial evolution and of control systems are demonstrated in a series of experiments with mobile robots engaged in navigational tasks using low-bandwidth sensors. Initial experiments are in simulation; the validity of such simulations and the significance of noise is discussed. Then experiments move to a real-world domain, with the use of a specialised piece of hardware which allows the automatic evaluation of populations of mobile robots using real low-bandwidth vision to navigate in a test environment. Evolution of capabilities is demonstrated in a sequence of navigational tasks of increasing complexity.

ACKNOWLEDGEMENTS

I was funded while doing the research for this thesis by SERC, the UK Science and Engineering Research Council.

COGS, the School of Cognitive and Computing Sciences at the University of Sussex, provided an ideal environment while this research was done. I was immensely fortunate that immediately after I arrived, Phil Husbands did also, and became my supervisor for this work. He has completely shared my interest and enthusiasm and belief in the importance of this area of work, and has been an enormous support. Phil Husbands and Dave Cliff and myself formed the Sussex Evolutionary Robotics Group which features in this thesis.

Much debt is due to members of the Alergic discussion group at COGS, and to innumerable people across the world in related fields, met through email or through conferences. I would like to thank Dave Cliff, Horst Hendriks-Jansen, Geoffrey Miller, Michael Wheeler and of course Phil Husbands for discussion of particular issues, and comments on earlier drafts; and particularly Pedro de Oliveira for numerous valuable and detailed comments.

I would like to thank Shirley Kitts for setting me straight on philosophical issues, and for throughout the gestation of this thesis giving it exactly the reverence it deserved — no more and no less. I must acknowledge all the effort and struggle put in through 4 billion years by my ancestors, without whom I would not be here; most immediately and above all, of course, by my parents.

DECLARATION

I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other University for a degree.

CHAPTER 1

Introduction

1.1 Who this is aimed at

A thesis cannot proclaim newly discovered truths valid for all time; it must be a creature of its time and place and culture, and be aimed at some particular people. In so far as it documents the discoveries and realisations and changes of mind of its author over the several years it has taken to come to fruition, it is probably aimed most directly at the fallacies and misapprehensions that the author shared with others a few years back, and which it is now his or her pleasure to correct!

With this in mind, this thesis is directed at two main academic groups. The first is the Genetic Algorithm (GA) community; the second, far more numerous group, is the Artificial Intelligence (AI) and Cognitive Science community. In this thesis is developed a methodology for developing robust controllers to produce adaptive behaviour in complex systems such as robots, advocating that the design process should be through artificial evolution. Hence this work is built on much from these two communities, and I hope contributes something back to both of them.

1.1.1 The Genetic Algorithms audience

The GA community does not in general need to be persuaded that it may be constructive to borrow ideas from evolution and apply these in algorithms to particular ends. But in general those purposes have tended to be restricted unduly to function optimisation, despite advocacy by both John Holland, the father of GAs, and Kenneth De Jong (1992) in favour of widening their scope. In this thesis, the emphasis will be strongly on *artificial evolution* as contrasted with function optimisation. I will demonstrate a considerable number of practical differences in the way GAs should be applied in this different domain. This is, I believe, the most radical and useful part of the thesis, in laying the groundwork for a general theory of artificial evolution that could be applied in any field where there

are long-term incremental design requirements; primarily in a co-evolutionary situation where the problems faced continually get more complex.

1.1.2 The AI and Cognitive Science audience

The particular domain where these ideas are intended to be applied in the first instance is that of designing the control structures, and perhaps simultaneously sensory morphology, of autonomous navigating robots; hence the second academic community to whom these ideas are addressed. In this case the co-evolutionary pressures may well be from the human experimenter. Whenever success is achieved in a limited task domain, the experimenter will extend the domain, 'move the goal posts', so that one could talk here of robots co-evolving with humans.

People within AI can be broadly divided into those who are doing science — they are interested in, perhaps, computational models of cognition and intelligent artefacts as an aid to understanding how humans and animals behave, perceive and plan; and those who are doing engineering — they are interested in creating useful or intelligent artefacts, and treat ideas from biology, neurobiology, evolution, psychology, etc. merely as means to this end. For the purposes of this thesis I am firmly in the engineers' camp; this deliberate stance obviously may be modified by the inevitable constant interchange of useful and interesting ideas between the engineers and the scientists.

Autonomous[1] robots are taken as the subject for artificial evolution because this seems to be the *ultimate* objective for AI as an engineering discipline (unless one is aiming to design tools which are useful for humans to use). 'Ultimate' here means that one is looking eventually towards artefacts that should be able to pass some version of a 'Total Turing Test' (Harnad 1989). Here, rather than the symbolic reasoning tasks characteristic of the original Turing Test and traditional AI, the emphasis has been turned towards the 'situated' and 'embodied' practical skills that are seen as more fundamental by the Artificial Life movement. This change of emphasis is not to deny that language processing and symbolic skills are a crucial part of *human* intellect; but it affirms that so many practical skills, of locomotion, navigation, interaction with the environment, are mediated by very different, phylogenetically earlier, modes of operation. It is denied that such skills (in practice in humans, in principle in robots) are produced by symbol-processing techniques; in other words, the central plank of the orthodox cognitivist position of mainstream AI is

[1] *Autonomy* is not for me a rigid all-or-none concept. To the extent that a robot controls its actions without external supervision, to that extent I call it autonomous.

here denied.

AI people acting as engineers first need to be convinced that an evolutionary approach to design has advantages. I will be arguing that in the long-term for the design of complex systems an evolutionary approach is the only possible one. Further, it needs to be established at what level of primitives the design of control systems needs to be tackled. This inevitably leads into philosophical issues.

1.2 Philosophical issues

For a project of creating perceiving, cognizing robots, creating autonomous artefacts with at least some of the characteristics by which we identify living creatures, it is inevitable that one's approach will be heavily influenced by one's philosophical perspective.

For much work in the area of AI, the underlying philosophical perspective is neither made explicit nor referred to. It is assumed to be shared with the readership. It is frequently a Cartesian objectivist realist position, which perhaps the majority of scientists share — whether they realise it or not. In my case my perspective is that of an extreme relativist, in that I deny the existence, indeed the meaning, of any 'objectively' real world independent of any observer. We have our own worlds, and my world reflects my culture and background, and so will be more or less shared with those readers who have a common culture; and will also be shared to some extent with other animals that have common concerns, of survival and locomotion and navigation. Any robot, in so far as it is autonomous will have its own world, and it is a matter of some interest to consider how much this might be shared with ours.

In the AI world I am today probably in a minority in taking this philosophical perspective. But it is a strong and vociferous minority. In the battle against traditional Cartesian positions, I would consider such writers as Dreyfus, Edelman, Varela, Freeman, Cariani, Brooks, Haugeland, Winograd, Stent, Gibson, Agre and Maturana to be broadly on the same side of this major divide as myself; this still allows for great differences between such people.

I mention these names not as any sort of philosophical argument 'from authority', but merely to make the point that, though a minority position, these views are sufficiently widespread and discussed elsewhere for me not to have to defend them in any depth here. The topic of this thesis is not philosophy but engineering of a specific kind. Because my philosophical approach is a minority one, yet fundamental to the directions pursued here,

I will make it explicit where this seems appropriate and necessary; where many who take an orthodox cognitivist position might get away with leaving it implicit. However, I make no claims to be saying anything new or original in the philosophical passages; it is the practical consequences of this approach that I am concerned with.

1.3 Layout of thesis

Broadly the thesis can be divided into three parts: up until the end of Chapter 5, discussion is of a broad and general nature, giving a particular perspective on the critical issues around the artificial evolution of adaptive behaviour, setting out what it is desirable to attempt to do: Chapters 6 to 10 then develop in some detail the theoretical ideas of how to implement artifical evolution: the next three chapters show the application of these ideas to practical problems. Below I briefly summarise each chapter.

After a chapter surveying work published to date on closely related areas (Chapter 2), there will follow two general introductory chapters. The first of these (Chapter 3) will argue (to an AI or robotics audience) for the use of artificial evolution as a design method — as an alternative to design by hand — for the 'brains' or cognitive architectures of autonomous robots. Then Chapter 4 will develop, from the philosophical stance alluded to above, ideas of what *sorts* of cognitive architectures should be designed through such an evolutionary methodology, in the particular domain of autonomous robots and adaptive behaviour. A program might seem the most attractive candidate to many; evolution of simple programs in languages with the power of a Turing machine will be considered; such as register machines, or the LISP S-expressions used in Koza's Genetic Programming. Then standard connectionist networks, or ANNs (Artificial Neural Nets) will be considered.

In Chapter 5 it will be suggested that neural nets in general have several virtues that make them attractive as the class of cognitive architecture on which artificial evolution should perform its magic. But it will also be pointed out that most work done with ANNs ignores the temporal dimension, as a consequence of many people tending to wear 'computation-tinted' glasses when working with them. It will be proposed that realtime recurrent neural networks should be used — and these should be considered as dynamical systems rather than tools to perform computations from inputs to output. The last part of this chapter discusses learning. This will be an attempt to answer those critics who suggest that any interesting or worthwhile autonomous robot will have to be capable of *learning*, and who suggest that this work does not address such issues at all, merely

producing 'hard-wired' creatures. I shall claim that 'learning' is a descriptive term at the behavioural level, rather than at the mechanical level or implementational level; and that it follows that, for the particular purposes of those creating autonomous agents through artificial evolution, learning poses no problems beyond the choice of suitable primitives.

Following these general themes, the middle section of the thesis moves on to address practical issues of how to develop or extend GAs for use in such a project. The differences between evolution and optimisation will be discussed. GAs tend to be used for function optimisation, but since I am advocating the use of (artificial) *evolution*, which I take to be somewhat different, this results in a form of GA rather different from those usually seen. SAGA (for Species Adaptation Genetic Algorithm) is the acronym for this form of GA developed here. To avoid misunderstanding, I should state that there is no single program or suite of programs which implements SAGA principles; rather, it is a bundle of connected ideas which follow from the notion of implementing (artificial) evolution rather than optimisation; in any particular application any or all of these ideas may be implemented in the programs used.

The practical results of the first two chapters in this middle section (6, 7) include:

The handling of variable-length genotypes, and how changes in genotype length should be restricted.

The acceptance of genetic convergence in this domain.

The appropriateness of tournament selection or rank-based selection, at higher-than-usual rates.

The importance of high rates of mutation.

A consequence of these results is that it is useful to have a grasp on the dynamics of populations under evolution, and hence the following Chapter 8 will discuss genetic drift; it divides naturally into three sections. The first part will discuss a particular published GA example which demonstrates the Baldwin effect, in which an anomaly is mentioned; this anomaly I will demonstrate to be due to random genetic drift, which in turn led me to do an analysis of genetic drift in haploid GAs in terms of diffusion approximation equations. The next part of this chapter will discuss how neutral genetic drift could be a factor of major importance in artificial evolution, with the side effect of making the combinatorics involved much less daunting than might at first have been thought. Examples will be used in this part from Schuster's analysis of early RNA evolution. Then, in the last part of

this chapter, from a particular run of an experiment in our Evolutionary Robotics work it will be shown how a genetically converged population of robot control architectures — or the genotypes which determine them — has indeed drifted significantly through sequence space.

Then Chapter 9 deals with particular problems which the crossover genetic operator faces when GAs are extended to deal with variable-length genotypes. An explicit algorithm will be developed, and a program to implement this given in an appendix.

In the last chapter in this theoretical section, Chapter 10, issues concerning a developmental or ontogenetic process from genotype to phenotype will be discussed. The reasons why such a developmental process might be desirable are considered, and various previous suggestions have their virtues and defects weighed up. A sketch of a proposal of my own will be put forward — this is, however, 'future work to be done'. The purpose of sketching it here is to make the point that all the inevitable consequences of variable-length genotypes that follow from it have already been considered in the previous sections.

The chapters in this middle theoretical section concern algorithms to be used for artificial evolution in *any* domain. As well as theoretical discussion, they include demonstrations of how these ideas work with particular abstract examples.

We then move on to applications of this evolutionary theory. The primary application area is autonomous robotics, but before reaching that there is a chapter on the TSP, Travelling Salesperson Problem, to go some way to filling in a lacuna in the robotics work discussed — Chapter 11. Here tests are made, in the domain of this much-studied problem, of the feasibility of using incremental evolution to tackle a succession of problems of increasing complexity, necessitating increase in genotype length.

The next two chapters describe results in Evolutionary Robotics based on the work developed here. As a consequence of the early research work for this thesis, the Evolutionary Robotics Group at Sussex came into being, initially funded for a year from October 1992 by a Research Development Grant from the University of Sussex, which employed me to to start transferring the theoretical ideas developed here into real applications in navigation for autonomous robots. This group, which initially consisted of Phil Husbands and Dave Cliff as well as myself, was then funded for a further 3 years by the Science and Engineering Research Council, SERC. Chapter 12 surveys the artificial evolution, in simulation only, of navigation behaviours in a simple mobile robot using low-bandwidth vision, using the dynamic recurrent neural networks proposed earlier. Then in Chapter 13 artificial

evolutionary methods are applied to a purpose-built piece of hardware, which allows the automatic evaluation of a mobile robot using real vision. In one experiment a series of tasks of increasing complexity is set for a population of robots — in fact instantiated one at a time in the real robot — and success achieved on each successive task.

A final summary chapter will attempt to assess the significance of what has been covered in the thesis.

1.4 Published work

Many parts of this thesis have already been published, primarily in conference proceedings, in versions more or less similar to the form they appear in here. Chapter 3 is an expansion of a paper originally jointly written with Phil Husbands (Husbands and Harvey 1992). Chapter 5 includes much that was first published as a Technical Report (Harvey 1992c). Chapter 5 includes much that was first published as a Technical Report (Harvey 1992c). The work in Chapter 6 first appeared in (Harvey 1992b). The work in Chapter 7 was presented at Artificial Life 3, in 1992, and is published in (Harvey 1993a). Chapter 9 appeared in similar form in (Harvey 1992a).

Applications of these ideas to robots in simulation and in reality, discussed in Chapter 12 and 13, have been covered in many jointly written papers: (Harvey *et al.* 1993c) (Harvey *et al.* 1993a) (Harvey *et al.* 1993b) (Cliff *et al.* 1993g) (Cliff *et al.* 1993a) (Cliff *et al.* 1993d) (Cliff *et al.* 1993c) (Cliff *et al.* 1993b) (Cliff *et al.* 1993e) (Cliff *et al.* 1993h) (Cliff *et al.* 1993f) (Husbands *et al.* 1993a) (Husbands *et al.* 1995) (Husbands *et al.* 1993c) (Husbands *et al.* 1993b). This work was done jointly with Phil Husbands and Dave Cliff. The different people's contribution to this work is summarised at the start of those chapters.

Chapter 8 incorporates two papers presented at the International Conference on Genetic Algorithms 1993, (Harvey 1993b, Harvey *et al.* 1993a) and an expansion of the latter in a Technical Report (Harvey *et al.* 1993b).

CHAPTER 2

Background

2.1 Introduction

This chapter will give a brief survey of various fields closely related to the work of this thesis, together with references to recent relevant literature. The survey will be broadly divided into 'Genetic Algorithms', 'Artificial Life' and 'Robotics'. Although ideas will be drawn on from Artificial Intelligence, Artificial Neural Networks ('PDP' or 'Connectionism') and Theoretical Biology throughout later chapters, these fields are too broad to cover in general here; particular references will be given later when appropriate.

This survey will of course be selective; and in particular attention will be drawn to specific problems associated with paradigms discussed, that will be of relevance to the work pursued further here. One issue returned to several times in the first section on Genetic Algorithms is that of the *evolvability* associated with different techniques.

2.2 Genetic Algorithms

Genetic Algorithms (GAs) were developed initially by John Holland in the 1960's (Holland 1975) as a form of search technique modelled on Darwinian evolution. The most accessible introduction is by Goldberg (1989b); other sources are Davis (1987), and the Proceedings of the GA and PPSN conferences (Grefenstette 1985, Grefenstette 1987, Schaffer 1989, Belew and Booker 1991, Forrest 1993, Schwefel and Männer 1990, Männer and Manderick 1992). A new journal **Evolutionary Computation** which covers GAs and related fields has begun in 1993.

Given a search problem, with a multi-dimensional space of possible solutions, a 'genetic-code' representation is chosen such that each point in the search space is represented by a string of symbols, the genotype. A number of initial random genotypes are produced, typically by a random number generator, which form the initial population. Each of

the corresponding points in the search space (which can be considered as representing the corresponding phenotypes) is evaluated by the appropriate evaluation function which gives higher scores to those nearest the solution sought, the 'fitter' ones.

The next generation of points is generated from the present population by selection and reproduction. The selection process is based on the scores of the present population, such that the fitter genotypes contribute more to the reproductive pool; typically this is also done probabilistically.

From the reproductive pool of selected genotypes, a new set of genotypes for the next generation is derived, using genetic operators such as crossover and mutation. The crossover operator works by taking parent genotypes in pairs, selecting a crossover point somewhere at random along the length of the genotype, and taking the left-hand section of one parent, as far as the crossover, and joining it to the right-hand section of the other parent; so the offspring inherits genetic material from both parents. The mutation operator just changes at random a very small proportion of the symbols on the offspring genotype to some other valid symbol.

The new population so derived has thus inherited genetic material selectively, but probabilistically, from the parent generation. New search points in the search space have been generated, exploiting the information from the performance of the parents, and improved performance can be expected from the new population. The cycle of selection and reproduction can then be repeated.

Variations on these genetic operators can be used, and there are decisions to be made on suitable sizes of populations, and the rate at which mutation and other operators are applied. In the choice of selection techniques, a recurring problem is keeping a balance between enough selective pressure for continued improvement, against too much selective pressure which leads to premature convergence, with loss of the diversity needed to escape from local optima. Subject to these caveats, however, GAs are a very robust search technique that can operate successfully in many varied, and indeed varying, search domains, and some of these properties can be demonstrated theoretically.

An early theoretical result by Holland (1975) was the Schema Theorem which demonstrates that (subject to certain conditions) schemata, which are the 'building blocks' in his representation, of above average fitness will receive exponentially increasing numbers of trials in successive generations — in other words improvement will take place. A schema is a similarity template which characterises that subset of all possible strings which have iden-

tical specified values at those positions specified by the template; at other positions, often indicated by # in the schema template, any values are allowed. Holland also demonstrated the implicit parallelism of GAs; in a population of size P, the number of these schemata being processed each generation is of order P^3 — in other words, it is an efficient algorithm. Just for once the combinatorial explosion is on our side. These results remain the theoretical underpinning to applications of GAs, although not necessarily the final story.

Typically GAs are used on well-defined problems where the search space is so large that other approaches are computationally infeasible. Within this general framework there are many possible very distinctive styles, with their own advantages and disadvantages, some of which will be covered briefly in the following subsections.

2.2.1 Evolving FSMs and programs

Apparently over the last half-century many people have experimented with using evolutionary techniques applied directly to strings of program code, or equivalent, considered as the genotype. The virtually total lack of early publications, before the last few years, was due to their universal lack of success in reaching any significant results.

An early evolutionary attempt which should be mentioned used small finite-state machines (Fogel *et al.* 1966). From a single parent machine a single offspring was derived by mutation of the parent's state transition table; the better of the two at a sequential symbol prediction task was chosen. No crossover operator was used to gain the power of recombination, and with a minimal size population the effect was basically simple hill-climbing. These early ideas have since been extended more successfully in the Evolutionary Programming paradigm (Fogel 1993).

The hurdles to be overcome with programs are:

1. How can genetic operators avoid making syntactically senseless programs?
2. How can 'meaningful' building blocks in a program be manipulated by genetic operators without effectively destroying any sense that the program has?
3. How can partial success be attributed to programs, with evaluations or fitness functions reflecting this; so as to avoid 'needle-in-a-haystack' search spaces?
4. How can programs with infinite loops be handled?

In an attack on the evolvability of structured programs, Conrad (1988a, 1988b) argues that for a program to be 'evolvable' it must in general have neighbours reachable by a

single structural change, i.e. mutation, which are at a minimum halting programs. Yet as a corollary to the unsolvability of the halting problem it is in general impossible to ensure that this is so, although it may be possible in special cases.

A program in most languages is a string of symbols, which could be considered a genotype. However the dependence on order in the string is so critical that the simple insertion or deletion of a punctuation mark can completely destroy the meaning of a program, and more often than not produce a syntax error — unless the semantics of the programming language are relaxed.

Since the usual genetic operators are disastrous for both the syntax and the semantics of traditional programming languages, emphasis has been placed on less traditional languages with simpler forms. A simplified form of LISP was used by Fujiki (Fujiki and Dickinson 1987) to produce programs for the strategy for the iterated prisoners' dilemma. Cramer (1985) used a subset of the simple algorithmic language PL, a pseudo-assembly language, to attempt to produce two-input, single-output multiplication functions. Many of the issues raised in Cramer's paper were significant precursors to Koza's further development of Genetic Programming, discussed below; specifically, the use of sub-tree crossover, and the evaluation of partial successes.

Cramer attempts to preserve any potentially useful but fragile code in the population from the ruthless genetic operators; the program is divided into blocks which are exchanged as a whole by the crossover operator, and mutation is restricted to 'non-catastrophic' parts of the program. The real problem, however, is that in attempting to evolve a program that performs multiplication, the evaluation function displays what Minsky has termed the *mesa* phenomenon (as quoted in Conrad 1988b); peaks are separated by untraversable flatlands, and there are no intermediate values between (a) the very few programs that perform the required function, and (b) all the rest, which are useless. An attempt was made, with only limited success, to hand-craft an evaluation function which gave partial credit to functions that did something 'multiplication-like'.

In order to investigate an evaluation function which is intended to value, without hand-crafting, 'interesting' programs of increasing difficulty — where 'interesting' would include a two-input one-output multiplication function — I experimented with a primitive assembly language suitable for a simple register machine. Such a language can be Turing equivalent; whereas a Turing machine has an initially fixed and finite number of internal states, but unlimited memory capacity in the form of an endless tape, in a register machine

this is traded off for a finite size memory, a fixed set of registers; but each register can hold an integer of any size. It can be shown (Minsky 1967) that only two registers are necessary for Turing-equivalence.

The register contents are limited to non-negative integers, and only a small set of primitive assembly language instructions are necessary. A program is a list of instructions which are carried out sequentially, except where the instructions themselves require a jump to a different part of the program. An advantage of such a register machine language is that any program, however manipulated by genetic operators, is syntactically correct.

I report this work elsewhere (Harvey 1991); the experiments I ran with this system produced inconclusive results. Of the four hurdles listed above, only the first and third were successfully tackled, and this latter only by means of having a moving evaluation target and an element of co-evolution. These experiments were run at the start of my research, and with hindsight I can see that the idea of seeking evolution rather than optimisation, and SAGA space came out of this. At this stage Koza's work was published, and it became immediately clear that, in so far as it overcame the most prominent hurdle that others had fallen at, any future attempt at evolving programs would have to acknowledge Koza's work, discussed in the next section.

2.2.2 Genetic Programming

Koza (1990, 1992b), extending ideas originally introduced by Cramer (1985), proposed and put into practice in a wide variety of domains a method of applying GAs to programming languages, Genetic Programming. This uses populations of programs which are given in the form of LISP S-expressions; these can be depicted as rooted point-labeled trees with ordered branches. The genotype is thus considered to be a tree rather than a linear string, and the primary genetic operator of recombination swaps complete subtrees between the parents. At the internal nodes of the tree are appropriate functions (mathematical, logical and domain-specific functions); at the leaves of the tree are terminals such as constants and inputs to the program. The output of the program is the value returned by the S-expression composed of the whole tree.

Given a particular problem for which solutions in the form of such programs are to be evolved — example problems include learning a Boolean multiplexer function, pattern recognition, optimal control of the cart-pole problem — first of all a set of functions and terminals appropriate to the domain are chosen. Each member of the initial random

population is constructed by sequentially choosing at random from this set of functions and terminals; a function will have a specified number of branches which need to be filled in turn, a terminal will terminate that particular branch. Usually limits are set on the depth of branching, both in the initial random population and in later generations produced through recombination.

Each S-expression, from the first or later generations, is evaluated. For many problems the fitness is taken to be the sum (taken over all the test cases), or the square root of the sum of squares, of the distances between the point in range space returned by the S-expression, and the correct point in range space. Parents for the next generation are selected in proportion to their fitness, and offspring generated by the recombination operator which swaps randomly chosen subtrees. Typically 90% of the population might be selected (with reselection allowed) from the population with a probability equal to their normalised fitness, and these recombined to produce the same number of offspring for the next generation. Then 10% of the previous generation were similarly selected (with reselection allowed) to make up the balance of the next generation. Mutation is an operator of minor importance, frequently not used at all.

The use of this genetic programming approach has been demonstrated in many domains (Koza 1992b), including Artificial Life and robot control systems. It tackles with some success all four of the hurdles. The problem of 'illegal offspring' is met by having the primary genetic operator of crossover, or recombination, swap complete sub-trees between the parents, which necessarily results in syntactically correct offspring. Since sub-trees may well form useful building blocks, the second hurdle is attempted. The problem of how to give partial credit to a less-than-perfect program is met by computing, for each possible set of inputs (or, with a large input space, each set from a sample of all possible), the distance in the output space between the output given by the program, and the correct output; the fitness function for this program is then based on the sum of all these distances for all the cases considered.

Koza's approach seems to have some success notwithstanding being based on a conventional programming language. In the majority of the examples he gives the functions used within the LISP S-expressions are limited so as to only produce primitive recursive expressions; these do not have the power of Turing-equivalence, and hence the halting problem never arises. Some of his examples, such as block-stacking problems, or finding prime numbers, do have something like a DO UNTIL operator, which gives them the

power of partial recursion, i.e. Turing-equivalence.

Where these programs that potentially may never halt are used, a 'time-out' system is brought into play to place limits on the number of iterations carried out. The point is made by Koza that in natural evolution, unlike simulations on a serial computer, all organisms are effectively evaluated in parallel; and the infeasibility of any one individual does not stop the continued propagation of the others. In the real world, the time taken to compute a decision is necessarily one of the factors to take account of in the evaluation of that decision.

2.2.3 Classifier Systems

There is a discussion of the issues involved in using GAs to search program spaces in (De Jong 1987), where he concludes that production system languages as developed for classifier systems are the most suitable. By and large they jump over or evade the hurdles listed above.

Classifier systems were put forward by Holland (Holland 1975, Holland and Reitman 1978) as a general machine-learning architecture appropriate for GAs to manipulate. These are production system languages, in which rules are strings with a left-hand side which represents the 'if' condition, and a right-hand side which asserts the consequent action. Given (for instance) a binary alphabet {**0, 1**} with the addition of the '**#**' wild-card or 'don't-care' symbol, a rule **01#00#** $\rightarrow$ **110110** would insert a new string or 'message' **110110** into the environment if and only if any of the four messages **010000**, **010001**, **011000**, **011001** were already in the environment. A program can consist of a set of these rules whose left hand sides are in parallel monitoring the environment, and effecting changes to it. Such rules are tailor- made for sensible manipulation by the usual genetic operators of crossover and mutation, and classifier systems have been much developed and are the subject of a considerable literature (Goldberg 1989b, Grefenstette 1985, Grefenstette 1987, Schaffer 1989, Belew and Booker 1991, Forrest 1993, Schwefel and Männer 1990, Männer and Manderick 1992).

In the 'Michigan' style of classifier systems, individual rules are members of the population, and the complete population forms the rule set which is tested against the environment; a GA for producing new rules is but one small part of a complex system, in which a 'bucket brigade' plays a prominent role in apportioning of credit from the relatively few external payoffs to the rules which contributed to them. The different 'Pitt' approach, in

which each member of the population is a complete rule set, is shown for instance in the LS-1 system of Smith (1980).

While classifier systems seem to be a formal mechanism very far removed from the network models of connectionists, there is a mapping from classifier systems to Boolean networks (Miller and Forrest 1989). Boolean networks consist of a set of nodes, each of which has a possible state of 0 or 1, and an associated fixed Boolean function. There are directed connections between nodes, and at discrete time-steps each node updates its state to the Boolean function of its inputs received from other nodes. By mapping all possible messages of the classifier system to nodes; the presence of any message in the environment to the firing, or being in state 1, of the associated node; and mapping the classifier rules into the appropriate connections and Boolean functions in the network, one can preserve the functional behaviour of the classifier system. Where there are strengths associated with each classifier, these can be mapped into a probabilistic function (or 'synaptic weighting') for the Boolean functions that implement the classifier.

2.2.4 Artificial Neural Nets

A 'structurally nonprogrammable' system, such as a connectionist network is gradually transformable. It has, for instance, been shown that a feed-forward network can learn arbitrary classifications using algorithms that either extend the network forward, in the sense of adding new layers, one at a time until the classification is learnt; or extending a network with a fixed number of layers laterally with the same objective (Frean 1989, Honavar and Uhr 1989). To build a connectionist network as a virtual machine on top of a conventionally programmed computer does not alter the fact that the virtual machine may be suitable for evolutionary development whereas the underlying real machine is not — the mutations of structure are at the virtual machine level only. The price paid for this, however, is the computational inefficiency of simulating one type of computation with another.

Genetic algorithms have been used as a technique for setting the weights within a connectionist network, as an alternative to, for instance, back-propagation; indeed, some success has been reported (Whitley 1989a, Montana and Davis 1989). However this setting of weights is for learning in an individual, with a given network architecture; GAs should be able to search through the space of possible architectures for the more appropriate ones for the task in hand.

Approaches to evolving connectionist network architectures that have been taken include those in (Kerszberg and Bergman 1988, Harp *et al.* 1989, Miller *et al.* 1989, Muhlenbein and Kindermann 1989); see also references in (Rudnick 1990). The genotype must specify the characteristics of nodes, and the connections between nodes. This can be done in either a serial descriptive fashion, or through some form of developmental process as will be discussed in Chapter 10. The number of different ways of doing this, such as those in the works cited above, show that it is possible to have a mapping which still has syntactical sense — produces a valid network — after application of genetic operators. If internal noise is added to a network during evaluation, as will be advocated later, this has the effect of smoothing or blurring the fitness landscape; in the sense that a mutation which eliminates or drastically changes a single node can be considered as maximum noise, and hence only different in degree from the noise associated with previous evaluations.

The problem of how to evaluate partial success on some particular task (such as, in a feedforward network, producing a specified mapping from inputs to outputs) is also easy; standard connectionist procedures use such measures all the time. Perhaps the analogous situation in a neural network to a non-halting program is a feedback loop, but there is no problem of 'illegality' here if indeed such an oscillation in real time is fed through to the outputs.

The remaining problem is how to ensure that genetic operations such as mutation and crossover manipulate useful building blocks. This is the Holy Grail that many have sought. One proposal will be put forward speculatively in Chapter 10.

2.3 Artificial Life

The motivation behind Artificial Life (AL) has been expressed by Chris Langton (Langton 1989):

> Artificial Life is the study of man-made systems that exhibit behaviors characteristic of natural living systems. It complements the traditional biological sciences concerned with the *analysis* of living organisms by attempting to *synthesize* life-like behaviors within computers and other artificial media. By extending the empirical foundation upon which biology is based *beyond* the carbon-chain life that has evolved on Earth, Artificial Life can contribute to theoretical biology by locating *life-as-we-know-it* within the larger picture of *life-as-it-could-be*. (Page 1, original emphases.)

Artificial Intelligence (AI) has tended to look at high-level problem-solving types of behaviour, which can fit easily into the computational paradigm. In contrast, AL tends

to start from the bottom up, looking at how processes specified at a low level can produce emergent[1] complex behaviour. While not using the computational paradigm as an underlying methodology for behaviour generation, the technology of computers is used as a laboratory tool for doing experimentation; just as in recent years computers have allowed 'experimental mathematics', which at one time would have seemed a contradiction in terms.

Approaches on these lines have been the subject of much recent interest, and conferences on similar themes have been held under the title of 'Artificial Life' in Santa Fe in 1987, 1990 and 1992 (Langton 1989, Langton *et al.* 1991, Langton 1993); in Europe in 1991 and 1993 (Varela and Bourgine 1992, Goss 1993); with the titles 'Evolution, Games and Learning' and 'Emergent Computation' by the Center for Nonlinear Studies, Los Alamos, in 1985 and 1989 (Farmer *et al.* 1985, Forrest 1989). Simulation of Adaptive Behavior Conferences have been held in Paris, 1990 (Meyer and Wilson 1991) and Hawaii, 1992 (Meyer *et al.* 1993). New journals **Adaptive Behavior** and **Artificial Life** have been started in 1992 and 1993.

Much work in AL has been done with simple 'animats' (Wilson 1985), a simulated animal or autonomous robot, usually in a simple simulated environment. Evolutionary approaches are a common theme. I shall briefly summarise here just two such studies out of many possible ones of some relevance to this thesis.

2.3.1 Evolutionary Reinforcement Learning

Ackley and Littman (1991) construct a simulated AL world, a two-dimensional array of 100×100 cells populated by adaptive agents and non-adaptive 'predators', 'food' and 'obstacles' (I shall hereafter drop the 'scare quotes'). The agents are deemed to receive visual inputs of the closest object up to four cells away in each compass direction; predators can see similarly up to six cells away. Predators move, and potentially harm agents, according to fixed rules mapping from sensory inputs to motion. Agents have an adaptive control system which determines agent actions depending on sensory inputs. According to the consequences of these actions agents gain or lose energy and health; they may gain energy by reaching food (including dead agents or predators), or lose health by bumping into obstacles or meeting a predator. Both agents and predators reproduce after accumulating enough energy from food, or die through starvation or damage. The numbers

[1] I shall discuss the concept of emergence in Chapter 3.

of agents and predators thus fluctuate throughout a run, sometimes to extinction.

The Evolutionary Reinforcement Learning (ERL) control architecture for agents consists of two major components, both of which have some factors specified genetically. The first component is an "action network", a feed-forward neural network with modifiable weights which maps from sensory inputs to actions. The initial weights on the connections of this network, genetically specified, represent innate behaviours inherited from its parents, but these weights are adjusted over time by a reinforcement learning algorithm (the particular algorithm used is 'Complementary Reinforcement Back-Propagation').

The second component is an "evaluation network" with fixed weights, which maps from sensory inputs to a real-valued scalar, which provides the reinforcement learning signal to the first network. Thus the goals specified by this network are fixed throughout the agent's lifetime, and inherited from its parents. When an agent reaches the energy level to trigger reproduction, if there are any agents within a prespecified distance within the AL world, the closest one is chosen as a mate; the genetically specified weights form a string, on which the genetic operator of recombination is applied, followed by mutation. If there are no potential mates close enough, asexual reproduction uses mutation only.

The interactions between learning and evolution in this scenario allowed for various experiments; such as comparing runs with the learning turned off against runs with evolution switched off, to be discussed in Chapter 5. Analysis was made of a particular run where a population of agents survived, through sometimes violent fluctuations in numbers, for some nine million time steps. The idea was used that lack of observed mutations in any part of the gene sequences of the population over a period of time implies that the corresponding part of the phenotype was functionally constrained, mutations there being harmful and eliminated through selection — as compared to areas of the genotype currently 'serving no useful function' where mutations can accumulate. From this it appeared that during the first 600,000 time steps the evaluation network is functionally constrained, and thereafter it is very much less so. The interpretation given is that in the initial periods the successful behaviours are determined by the evaluation network, which through reinforcement allows the action networks to rapidly learn successful actions; whereas later the action networks, through inheritance of appropriate initial weightings, produces successful behaviour from the start, and hence the evaluation network is called into play much less often to give negative reinforcement for mistakes. This is given as an example of the Baldwin effect (Baldwin 1896), to be discussed further in Chapter 8.

The general framework of this scenario is characteristic of many AL studies, in that a simple model of agents and their environment is specified, and the current power of computation allows the observer to look at long-term trends and formulate and test hypotheses. Whether such hypotheses could be transferred to the natural world of biology or ecology depends of course on whether the initial model has captured the essential characteristics of the real world. Even if this is not so, fruitful ideas for use in designing complex systems may arise.

2.3.2 TIERRA

Ray in his TIERRA simulation (Ray 1992) has a virtual world in which organisms are modelled at a very primitive level and allowed to replicate and evolve in an open-ended fashion; the aim is to generate increasingly diverse and complex organisms, in a parallel to the Cambrian explosion — Ray is an ecologist and evolutionist by background. The metaphor used in his model is one where organisms compete for resources of energy and space in the form of CPU (Central Processing Unit) time and memory in a computer (actually in a virtual computer). An organism consists of a self-replicating assembler language program. A finite block of (virtual) memory is designated as the "primaeval soup" which is inoculated with an initial population of prototype creatures 80 assembly language instructions long. These have been designed by hand to be self-replicating, and during replication mutations can cause errors.

Within memory each organism occupies a sequence of RAM, bounded by markers which effectively form a membrane or cell boundary; whereas the programs of any other organism can read or execute code within such a membrane, only the organism itself has write privileges within its cell. Replication of a daughter cell results from the mother writing into a fresh area of memory, but on completion the daughter cell is given its own instruction pointer, and exclusive write privileges over its own section of memory. As a community of organisms lives in the soup simultaneously, the supervising system shares out CPU time to each organism through multi-tasking and time-slicing — in effect each organism has a virtual CPU. The number of organisms increases through replication, and hence needs to be kept within bounds by a mortality system, the 'Reaper queue', which eliminates organisms on the basis of both age and their failure in executing certain instructions. The 'dead' code of killed organisms is left in memory, and may be used and over-written by other organisms.

Initially a single hand-designed genotype of 80 instructions, the 'ancestor', is placed in the memory, which can hold 60,000 instructions. It takes the execution of some 800 instructions for the ancestor to replicate once, and rapidly the memory fills up with copies, some imperfect through mutation. As the community becomes diverse, new classes of genotypes appear, of size and character different from the ancestor. For instance, 'parasites' evolve, which cannot self-replicate in isolation, but manage to replicate in the presence of hosts by executing parts of their hosts' code. Hyper-parasites can then evolve — (Ray 1992) page 383:

> ...which can self-replicate in isolated culture, but when subjected to parasitism, subvert the parasites energy metabolism to augment their own reproduction. Hyper-parasites drive parasites to extinction, resulting in complete domination of the communities.

The translation of concepts such as 'energy', 'parasitism' and so on between the synthesized artificial world and the real world of biology which it models is at a fairly abstract level. Nevertheless, it seems that ecologists and evolutionary theorists find the TIERRA world useful as a testbed in which theories can be examined.

The success of this work in evolving replicating 'organisms' and 'parasites' based on machine-code instructions has been interpreted by some as demonstrating that the inherent brittleness of partial recursive program languages (discussed in earlier sections) can be avoided. But it is avoided in this case because there is no externally imposed evaluation which requires each machine-code program to be 'run until it halts' for its fitness to be evaluated. The real time of the 'organisms' is measured in terms of program steps. So the brittleness of non-halting programs is avoided; under normal circumstances a program with an infinite loop does not produce a result, but in the case of TIERRA it is *necessary* for survival for the programs to be non-halting.

A second type of brittleness, related to hurdle 2, is when a mutation can produce a radically different result (which in the case of an 'organism' could eventually prove fatal in pushing it along the queue towards the reaper). TIERRA is vulnerable to this form of brittleness, though it has been minimised with a small instruction set and having 'similar' instructions close to each other in Hamming-space.

2.4 Robotics

Much traditional robotics is concerned with the control of objects such as robot arms through a well-defined space of possible positions. The domain of interest for this thesis

is movement and navigation of an autonomous mobile robot through a space that is not well-defined. The traditional approach to doing this has used an implicit assumption of functional decomposition, assuming that perception, planning and action can be analysed independently of each other. In this paradigm, the role of perception is, as far as possible, to deliver a well-defined world model for the planning module to deal with. These sorts of traditional approaches to the development of autonomous robot control systems (Brady *et al.* 1982) have made only modest progress, usually with computationally very expensive methods; they have had great problems with robustness in the face of noise.

One approach for control is to treat it as a problem where an input/output mapping must be learnt. Neural networks as the medium of control have been used in such projects as ALVINN with the CMU Navlab (Pomerleau 1991) for real robots. This used a back-propagation network with a human driver as the teaching input; the task was to learn to drive the Navlab van down roads using the input from a video camera. The limitations of using a synthesized environment meant that training sequences were derived from real roads, with techniques developed to ensure that examples of varied and bad driving were also available to be learnt from. The supervised learning technique distinguished between the performance phase, and the training phase in which the human driver provided feedback.

The main alternative to traditional robotics in recent years has been so-called 'behaviour-based' robotics, which challenges the traditional assumption of functional decomposition. The prime mover has been Brooks with his subsumption architecture (Brooks 1986, Brooks 1991).

2.4.1 Subsumption architecture

Subsumption architecture is a design philosophy for building robots that can deliver realtime performance in a dynamic world that has not been specially constructed for it. It is 'behaviour-based' in the sense that (at least in its purest form) each modular part of the control systems directly generates some part of the behaviour of the system; in other words, such a module would incorporate pathways all the way from sensors to actuators. Thus in contrast to traditional functional decomposition, behavioural decomposition carves up a complete control system into separate and to a large extent independent behavioural layers. 'Lower' layers take care of more primitive behaviours, 'higher' layers more sophisticated ones, with internal interactions between such layers limited to simple

suppression and inhibition[2]. These interactions between layers could be thought of as conflict-resolution methods where different layers would otherwise be simultaneously trying to produce different behaviours. All the layers are executed concurrently. Increasingly higher layers incorporate and take advantage[3] of lower layers.

Integral to this design philosophy is the requirement that robots are initially designed with just the most primitive layers of behaviour (e.g. collision avoidance, wall-following); only after these are tested and debugged are more sophisticated behaviours added incrementally, tested and debugged in their turn. Thus at all stages the robot should be able to operate at some level of competence. Because of the insistence that in general each new layer goes all the way from sensors to actuators, this approach has been termed by some 'reactive'. This is misleading as it implies that there is no internal state in any of the layers. In fact each layer is composed of Augmented Finite State Automata (AFSM), with much internal state. The 'augmentation' refers to the explicit incorporation of time: timer units pass on signals after a specified period of time, or repeat actions at regular timed intervals until suppressed or inhibited.

Behavior Language (BL) allows for the design of subsumption architectures in a subset of LISP (Brooks 1990). Groupings of realtime rules for individual layers of behaviour are written in BL. These are compiled in machine-independent form into a specification in terms of AFSMs, which can then be further compiled for the particular machine that implements the control system. Whereas conceptually all the layers act concurrently and asynchronously in parallel, this is in practice implemented on a sequential machine by fine time-slicing.

This use of loosely coupled asynchronous task-achieving behaviours in practice gives a robustness and fault-tolerance which the traditional approach of functional decomposition typically fails on. It also lends itself very well to an evolutionary approach.

2.4.2 Evolutionary Robotics

In a traditional robotics context, mention is made of an evolutionary approach in (Barhen *et al.* 1987). A student of Brooks discussed some of the issues involved, with reference to subsumption architectures, in (Viola 1988). Evolutionary robotics was proposed for philosophical reasons in (Cariani 1989). De Garis (de Garis 1992) proposed using GAs for

[2]'Inhibition' describes the action of one internal unit blocking the action of another; 'suppression' describes the action of one unit substituting its own output for that of another.

[3]'Subsume', hence *Subsumption* architecture.

building behavioural modules for artificial nervous systems, or 'artificial embryology'. Beer (Beer and Gallagher 1992) used GAs to synthesize a walking behaviour for a six-legged agent. It is only recently that serious proposals have been made to use evolutionary approaches to real-world robots (PRANCE 1991, Brooks 1992, Husbands and Harvey 1992).

Most interest in such an approach has probably come from Japan, which currently has significantly more than half of all the robots in existence. Recently the Japanese government research laboratories, ATR in Kyoto, have set up a well-funded research group for Evolutionary Robotics in their Evolutionary Systems department. Similar work is pursued at ETL in Japan, and there is interest from Japanese industry; Mitsubishi sponsored a symposium on Evolutionary Robotics in March 1993, which has since become an annual event. At the Simulation of Adaptive Behavior 1992 conference SAB92 (Meyer *et al.* 1993) in Hawaii, a group of papers were closely related to this field. This was followed up similarly at SAB94 held in 1994 in Brighton, where our Evolutionary Robotics Group were the local organisers.

Here what I consider to be the two most significant proposals (apart from that presented here) will be briefly summarised; that of Brooks, and that of Beer and Gallagher.

Brooks' evolutionary approach

In December 1991 Brooks reported at ECAL-91 in Paris (Brooks 1992) that, following a suggestion by Langton, he was starting work on an evolutionary approach to robotics, based on Koza's Genetic Programming techniques (Koza 1990). He acknowledged the advantages of largely using simulations, but stressed the dangers of using simulated worlds rather than real worlds. Two further points he makes will be endorsed in later chapters of this thesis: firstly that by evolving the control program incrementally the search space can be kept small at any time; and secondly that symmetries or repeated structures should be exploited so that only a single module needs to be evolved, which is then repeatedly used.

Brooks analyses the use by Koza of conditionals in genetic programming, specifically when Koza evolves a subsumption architecture (Koza 1992a). These are usually special forms or macros, such as IF-SENSOR with two arguments. Depending on the value of the sensor reading, either the first or second argument is evaluated. The use of such conditionals as primitives, plus the embedding by hand of critical constants within such predicates, makes the search space of programs drastically smaller than it would be had

the primitives of Common LISP and the necessary constant values had to be assembled through evolution. As BL does not, as it stands, have these advantages for evolvability, Brooks proposes a higher level language, GEN, which could be evolved, and then compiled down into BL and further on down onto the actual machine.

Since this original proposal, I understand that no further significant work has been done.

Beer's evolutionary approach

Important work on an evolutionary approach to simulated robotics using neural networks has been done by Beer and Gallagher (Beer and Gallagher 1992, Gallagher and Beer 1993). They explore the evolution of continuous-time recurrent neural networks as a mechanism for adaptive agent control, using as example tasks chemotaxis, and locomotion-control for a six-legged insect-like agent.

The networks consist of a number of model neurons, the state equation for the ith neuron being of the form:

$$\tau_i \frac{d\gamma_i}{dt} = -\gamma_i + \sum_{j=1}^{N} w_{ji}\sigma_j(\gamma_j) + I_i(t)$$

where γ is the activation of the neuron; $\sigma_j(\xi) = (1 + e^{(\theta_j - \xi)})^{-1}$ is a sigmoidal function; θ is a threshold; τ is a time constant; w_{ji} is the weighting or strength of connection from the jth to the ith neuron; $I_i(t)$ is an external input from a sensor. Arbitrary recurrent connections are allowed. This is an extension of a Hopfield net (Hopfield 1982).

The technique used by Beer and Gallagher to determine the time constants, thresholds and connection weights is a standard GA, as implemented in the GENESIS package (Grefenstette 1983). They report success in their objectives; in the case of locomotion control, controllers were evolved that in practice generated a tripod gait (front and back legs on one side in phase with the middle leg on the opposite side). This was achieved both with and without the use of sensors which measured the angular position of each leg. They tried a number of GA variations in the use of selection only, or crossover only, or mutation only. Their discussion of the issues involved is to be recommended to anybody working in this area. They do not emphasise in the way Brooks (and Pomerleau) does the important distinctions to be made between simulations and reality, as their work reported here is confined to simulations; nevertheless a neural net approach such as theirs should be inherently much less brittle in the presence of noise than a programming approach.

Beer (1992) develops a dynamical systems perspective on control systems for autonomous agents, influenced by early work in Cybernetics (Ashby 1960), which corresponds closely with ideas put forward here in later chapters.

CHAPTER 3

Evolution not Design[1]

3.1 Introduction

In this thesis I am attempting to set out a methodology for the development of the control systems, or 'cognitive architectures', of autonomous mobile robots intended for use in environments that cannot be completely pre-specified. Many competences will be required, and a primary one to consider is navigation. Such robots will probably require active perception or 'animate vision'. In other words, vision is considered as a closed loop; not only do visual inputs influence, via the control system, motor outputs, but *also* the movements of the robot can be thought of as influencing what visual inputs are received (Ballard 1991). The robots will be behaviour-based as introduced in Chapter 2. Although behaviour-based approaches to robot control appear to be far more promising than traditional model-based functional decomposition methods, in this chapter it will be argued that the design of such control systems is still prohibitively difficult.

Artificial evolution requires the evaluation of a great number of candidate control systems. Time constraints suggest that perhaps many of these evaluations should be done in simulation. The requirements and pitfalls of such a simulation system are discussed.

3.2 Interesting robots are too difficult to design

Traditional approaches to the development of autonomous robot control systems have made only modest progress, with fragile and computationally very expensive methods. A large part of the blame for this can be laid at the feet of an implicit assumption of functional decomposition — the assumption that perception, planning and action could be analysed independently of each other. This failure has led to recent work at MIT which bases robot control architectures instead around *behavioural decomposition* (Brooks 1986,

[1] This chapter is partly based on a paper jointly written with Phil Husbands (Husbands and Harvey 1992).

Brooks 1991), using 'subsumption'. Such work rejects the traditional AI approach which manipulates symbolic representations of the world, and places more emphasis on 'knowing how' to do things rather than 'knowing that' the world is in a given state. Viewpoints sympathetic to such an approach can be seen in, e.g., (Rosenschein 1985, Beer 1990, Cliff 1991b, Agre and Chapman 1990).

Such a subsumption-style cognitive architecture for a robot in theory analyses independent behaviours of a robot or *animat*, such that each behaviour can be 'wired in' all the way from sensor input to motor output. Simple behaviours are wired in at first, and then more complex behaviours are added as separate layers, affecting earlier layers only by means of limited suppression or inhibition mechanisms.

However, it is generally accepted that the design of robust mobile robot control systems is highly complex because of the extreme difficulty of foreseeing all possible interactions with the environment; and the interactions between separate parts of the robot itself (Brooks 1991, Moravec 1983). Even where internal interactions are not apparent, there may be interactions between parts of the robot mediated via the environment. The design by hand of such a cognitive architectures inherently becomes more complex much faster than the number of layers or modules within the architecture — the complexity can scale with the number of possible interactions between modules.

One way out of this problem is to try and automate the design process. A possible approach is to view the design of a control architecture as a planning problem. Traditional AI approaches to planning have been shown to be computationally infeasible when applied to such problems (Chapman 1987).

The design of a behavioural layer in a subsumption architecture seems to be design by magic, by sleight of hand, by indirection; in the sense that the desired behaviour can often be described as an emergent by-product of rule-following which does not explicitly mention that behaviour (Horswill and Brooks 1988). Emergence is in itself nothing magic as a phenomenon, if it is considered as emergence-in-the-eyes-of-the-beholder[2]. Something can be characterised as emergent relative to an initial given description if:

1. a system can be set up which corresponds completely to this initial given description.

[2]Michael Wheeler has pointed out to me in discussion that common usage of the word 'emergence' — 'the moon emerged from behind a cloud' — refers to transitory phenomena. Perhaps much confusion would be avoided if in its metaphorical sense it was also used to refer only to a relatively brief period of time.

2. A new description of the behaviour of the system can be made which 'is useful' or 'makes sense' to an observer, and makes use of concepts outside those originally given.

To design such emergent behaviours hence requires either (a) a computationally intractable planning problem or (b) a creative act on the part of the designer — which is to be greatly admired, though impossible to formalise. In both cases it seems likely that the limits of feasibility are currently being tested.

3.3 Let's evolve robots instead

If, however, some objective fitness function can be derived for any given architecture, there is the possibility of automatic evolution of the architecture without explicit design. Natural evolution is the existence proof for the viability of this approach, given appropriate resources — just how much is necessary, in time and perhaps in parallelism, is obviously a matter of great concern. Genetic Algorithms (GAs) use ideas borrowed from evolution in order to solve problems in highly complex search spaces, and it is here suggested that GAs, suitably extended in their application, are a means of evading the problems mentioned in the previous section.

The artificial evolution approach will maintain a population of viable genotypes (chromosomes), coding for cognitive architectures, which will be inter-bred and mutated according to a selection pressure. This pressure will be controlled by a task-oriented evaluation function: the better the robot performs its task the more evolutionarily favoured is its cognitive architecture. Rather than attempting to hand design a system to perform a particular task or range of tasks well, the evolutionary approach will allow their gradual emergence.

The sleight-of-hand, or indirection, problem mentioned above, is avoided with GAs in that there is no need for any assumptions about means to achieve a particular kind of behaviour, as long as this behaviour is directly or implicitly included in the evaluation function.

Brooks' subsumption approach was mentioned above as a contrast to the dogmatic assumptions of functional decomposition implicit in much of traditional robotics. Nevertheless, it is similarly not necessary to be dogmatically committed to an exclusively behavioural decomposition. By allowing both types of decomposition — indeed any kind of decomposition — the evolutionary process will determine where in practice the balance

should lie in the robots' cognitive architecture.

Turing, in his seminal paper (Turing 1950), showed that he saw the attractions of an evolutionary process (page 456):

> We cannot expect to find a good child machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications
>
> Structure of the child machine = hereditary material
>
> Changes of the child machine = mutations
>
> Natural selection = judgement of the experimenter
>
> One may hope, however, that this process will be more expeditious than evolution. The survival of the fittest is a slow method for measuring advantages. The experimenter, by the exercise of intelligence, should be able to speed it up. Equally important is the fact that he is not restricted to random mutations. If he can trace a cause for some weakness he can probably think of the kind of mutation which will improve it.

I argue that Turing's suggestion in the latter part of this quotation, that the experimenter can and should think up which specific mutations will speed up the evolutionary process, is not a viable proposition. I will be arguing that the experimenter should indeed have the analysis which can determine appropriate settings of mutation rates and other genetic operators, but should then stand well back and leave evolution to its own devices. The combinatorics of evolution are not loaded as heavily against success as many people, including perhaps Turing, have thought. These issues will be discussed in Chapters 7 and 8.

3.4 Evaluation

The human roboticist should relinquish the design role to the evolutionary mechanisms of an extended GA, once the capabilities being sought for the robot go beyond toy domains. But unrestricted evolution will have no reason to produce *animats* that the human wishes for, and hence for practical purposes the human must take on a role analogous to the plant breeder or cattle breeder, both of whom act so as to influence the future course of existing species. Hence the task of the human will be to design a set of robot tasks of increasing complexity (and ways to evaluate them), leading from the capabilities of existing robots towards those capabilities required in the future. This is non-trivial, although the creative human input to this process will be at a higher level of abstraction than current design at the nuts and bolts level.

Such a sequence in evolutionary time parallels the addition of layers of behaviour in a subsumption architecture designed by hand — for instance progression from obstacle-avoidance to wandering to wall-following to simple navigation. The score of an individual robot will at any one time be based on a function of the scores on each of the individual tasks, and at any time a new task can be added which contributes to the score. Species evolution within the SAGA framework (to be introduced in Chapter 6) limits the search process to the appropriate adaptations to networks with already established capabilities. A price that has to be accepted here is that only a minute portion of the global search space is covered; but no other method avoids this once we are beyond toy domains.

Though an animal should not be considered as a solution to a problem posed 4 billion years ago, in the short term adaptations in a species may be usefully interpreted as solving particular problems for that species. So when using the evolution of animals as a source of ideas for the evolution of *animats*, GAs should be used as a method for searching the space of possible adaptations to an existing *animat*, not as a search through the complete space of *animats*.

This of course has strong resemblances to Brooks' incremental approach, wherein 'low-level' behaviours are wired in and thoroughly debugged, before the next layer of behaviour is carefully designed on top of them. The difference with the approach advocated here is that of substituting evolution for design.

3.5 Optimisation

GAs have primarily been used for optimisation. Optimisation problems can be thought of as a search space where each point in the search space, corresponding to particular settings of the variables in the problem, has an evaluation, or fitness. By adding an extra dimension which represents the fitness at such a point in the search space, a fitness landscape can be created. Optimisation means finding the fittest such point, or at least finding some near-optimum.

For difficult problems of interest — such as the classic Travelling Salesperson Problem — the combinatorics are such that exhaustive search is not feasible. Optimisation techniques need to be found for this sort of problem, and GAs are of course one such technique.

For present purposes, what characterises optimisation problems are that they are well-defined problems with a circumscribed search space; and that the fitness function is well

defined. In general the search space is of finite — though large — size. If in fact one or more of the variables to be determined through the optimisation process is continuous, then the search space could be considered as containing an infinity of possible solutions. But a GA approach will in fact approximate the continuous nature of such a variable by discretisation, and hence use genotypes with a finite alphabet.

Since in optimisation problems the fitness function is also well-defined, the requirement is to find the global optimum, or alternatively satisfactory near-optima, from the whole search space.

3.5.1 Evolution, in contrast

If we have a population of giraffes with varying neck lengths, in an environment where nutritious leaves are only found on tall trees, it is easy for us to treat this as an optimisation problem in which those with longer necks find the optimum. The act of casting it as an optimisation problem is one of circumscribing the search space (e.g. to lengths of necks) and specifying the fitness function (ability to reach the available leaves). But the whole process of natural evolution cannot be treated in the same way as this particular isolated problem.

Firstly, it should be noted that the nature of the problem was defined a posteriori. Other species, descended from the same ancestors as giraffes, have developed different lifestyles in which the consumption of leaves from tall trees takes no part. This problem does not exist for them.

Secondly, it is not even the case that different problems pre-existed the species that came, so to speak, to tackle them — some tackling these problems and some tackling others. The environmental niche is only specified in terms of the organisms that occupy that niche. In Varela's terms (Varela *et al.* 1991) the relationship is one of *mutual specification* or *codetermination*, elsewhere described by Lewontin (1983), pages 75–76:

> The organism and the environment are not actually separately determined. The environment is not a structure imposed on living beings from outside but is in fact a creation of those beings. The environment is not an autonomous process but a reflection of the biology of the species. Just as there is no organism without an environment, so there is no environment without an organism.

The opposition between internal and external factors in evolution — between Nature and Nurture — ceases to make sense under this view. In concert with this, Varela develops a view of evolution as *natural drift*, in contrast to the adaptationist view of positive selection for ever-increasing fitness. Rather than following some 'arrow of progress' the

evolutionary path that any species follows is a matter of taking *any* direction that allows for organisms or structures having sufficient integrity to survive. This is *satisficing* rather than optimising.

3.6 The environment of a robot

How do these ideas fit in with the artificial evolution of autonomous robots? Here at first sight it would seem that the environment of a robot, and what behaviour is going to count as desirable, will be completely pre-specified by the humans overseeing the process. To the extent that this is true, then creating robots with the desired behaviour in such an environment would seem to be a straightforward — though difficult — optimisation problem. Surely in this case, we do not have the robot and environment mutually specifying each other; is not the human being firmly in charge and laying down the law?

There are two questions that are going to arise here:

1. Is the robot's environment going to be fully specified?

2. Is the criterion for success going to change over time?

If the robot's environment is so fully specified that it can be formalised, then indeed we are back with an optimisation problem. But as will be discussed in Chapter 4, whereas problems of the chess type that dominated early AI can indeed be formalised, most of the interesting capabilities that count as cognitive are just those where this cannot be done.

In so far as a robot is meant to be autonomous or in charge of itself, it will be intended to cope with the unexpected and unanticipated. Whereas with the traditional cognitivist view of AI it was assumed that the creation of intelligence involved algorithms which could cope with, and calculate appropriate responses to, all potential contingencies within a pre-specified domain, this does not hold under the philosophy espoused here.

Although particular evaluation functions may be well defined and clearly specified by humans as targets to be achieved by robots – - indeed the process of artificial evolution will depend on this — it is only in simulations that the physical characteristics of the environment can be completely formalised. Such formalisations of the physical world can never be guaranteed to contain all potentially relevant facts.

Turning to the second point listed above: if and when we create robots that score satisfactorily highly on our evaluation criteria, what will happen in practice is that we will then move the goal posts; tasks will be made more complex, constraints on the type of

physical environment that the robot is intended to operate in will be loosened. So what can be expected in the medium term is co-evolution between robot species and humans.

3.7 Simulation versus Reality

Any evolutionary technique is going to need large numbers of trials of robots, and practical constraints mean that these should be done on simulated robots if this is viable. Traditionally, and for good reason, those who have built real robots have tended to scorn simulations as implicitly assuming that all the really hard real-world problems have been solved. To quote from (Brooks 1992), pages 4–5:

> First, there is no notion of the uncertainty that the real world presents ...Second, there is a tendency to not only postulate sensors which return perfect information (e.g., the cell ahead *contains food* — no real perception system can do such a thing) but there is a real danger of confusing the global world view and the robot's view of the world. ...

This is a valid criticism of naive simulations, but for some practical purposes adequate simulations have been made. It is standard practice for a commercial pilot to convert to flying a new model of a plane by training in a flight simulator, so that the first real flight in the new plane is carrying passengers. Two things should be noted: the conversion is from one plane type to another similar one, and commercial flight simulators are so complex that, although cheaper than flying the real thing, the cost is not of a totally different order.

The simulator is limited by the knowledge of the programmer of relevant factors to be included, and if, for instance, no account of the effects of wind shear is put into a flight simulator, the first encounter with this in the real world will be hazardous. So any benefits brought to an evolutionary approach by using simulations will inevitably have to be paid for by major effort being put into the realism of the simulator. Trials of evolved architectures on real robots will have to be carried out at frequent intervals for the dual purpose of validating the fitnesses *and* providing feedback for improvement of the simulator.

As robustness is vital for any control system, the use of noise in any simulations is critically important. Since the results on individual runs will vary with noise, a number of runs should be made. Taking the *minimum* score achieved over a number of runs, rather than the average, as the final evaluation will implicitly favour robust designs. If adaptive noise-tolerant units, such as neural nets, are used as the key elements of the control system,

then 100% accuracy is not required. Discrepancies between the simulations and the real world, as long as they are not too big, can be treated as noise; the system can adapt to cope with this.

In the long-term, as the robots become more sophisticated and their worlds more dynamic, will the simulation run out of steam? The simulation of a medium resolution visual system with, for instance, motion detection pre-processing is painfully slow on today's hardware. Techniques to test many generations of control systems in real worlds will have to be developed. In the work done by the Evolutionary Robotics Group at Sussex, to be discussed in Chapter 12, we have found that doing simulations of vision, using ray-tracing, has been so computationally expensive that it has been necessary to build special-purpose hardware to allow automation of multiple evaluations with real vision.

3.8 Conclusions

In Artificial Intelligence, it is always a good idea when in doubt to look again at the only uncontroversial examples of intelligent and adaptive behaviour, those provided by the real world. That these were produced by evolution rather than by design was Darwin's great insight. That a similar approach can be used in developing artificial examples of intelligence and adaptive behaviour has only been appreciated relatively recently.

The distinction has been made between *optimisation* and *evolution*. Techniques for doing the former require a well-defined goal and a well-defined search space. Because for autonomous robots the goal-posts will be moving, and the search-space ill-defined, it will be artificial evolution that is needed. A major problem is the large number of evaluations that will need to be done. Simulations are one possible way around this, but bring problems of their own.

There is no such thing as a free lunch, and the evolutionary approach is not proposed as some sort of magic that requires little effort or time. Under some circumstances simulations may be of use; the ultimate test of this is whether indeed resulting designs for cognitive architectures can be transferred successfully to real robots. It is likely that the advance of technology will mean that in a few years time robots will be small enough and cheap enough for it to be feasible for multitudes of real robots to be used in large scale evolutionary programs.

CHAPTER 4

Cognition, Complex Adaptive Systems and Evolution[1]

4.1 Introduction

Artificial Intelligence arose as a field of study from the belief that intelligent human behaviour could be formalised, and hence could be mechanised. Problem solving must be done according to rules, so this approach went; put the rules in a machine, and we will have an intelligent machine. The greatest successes have been, understandably, in just those fields of human intelligence where the problems can be formally defined, e.g. chess-playing, expert systems in simple domains; even here, success has been limited. Connectionism (McClelland and Rumelhart 1986) uses a different approach; investigating network models which are based on a very simplified model of the brain, namely large numbers of simple processing nodes with many 'wires' or links connecting them, passing activations throughout the network. A major insight to come from this approach is that the behaviour of the whole can look as though it is obeying explicitly programmed rules, even though one can see that this is just an emergent property of the underlying mechanisms.

The conventional AI approach tries to design into a program intelligent behaviour; the connectionist tries to design networks that will produce intelligent or adaptive behaviour. Yet Darwinian evolution shows us that there can be intelligent behaviour without a designer. Hence as one can consider the intelligence and adaptability of humans and other animals to be an emergent property of their evolutionary history in their environment, one can also consider the possibility of the emergence of intelligent and adaptive behaviour in simulated organisms, in simulated environments, with some form of evolutionary algorithm.

The present chapter will attempt to focus on what we might be trying to artificially create; what is cognition, what counts as an adaptive system? We will all agree that humans have cognition, and are complex adaptive systems. Similarly with animals, though

[1] Parts of this chapter first appeared in (Harvey 1992c).

some will want to draw a rigid line between human and animal cognition while others will be more relaxed and consider this a matter of degree. Undoubtedly, if we are seeking to produce intelligent autonomous robots these will be complex and adaptive systems. Attribution of cognition to such systems may well be dependent on one's philosophical stance.

I take the attitude that it would be counter-productive to start with a set of definitions for what constitutes cognition. Rather, just as what counts as art is determined by what people in their social climate treat as art (with violent disputes, and shifts in opinion over time), so also with cognition. Definitions are tools for stabilising terminology for agreement within a group of people, and hence become awkward just at that moment when agreement may be shifting; the potential appearance of intelligent autonomous artefacts could well herald such a shift in opinion, a point fully recognised by Turing (Turing 1950), page 442:

> The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.

So in my discussion of cognition and adaptiveness I will not myself be putting forward definitions; rather, I shall be advocating a certain way of looking at these issues. I shall start with consideration of adaptiveness, which is inextricably bound up with selection.

4.2 Adaptation and selection in the natural world

Let us start from this characterisation of 'adaptiveness' (Meyer and Guillot 1991), page 2:

> In a changing, unpredictable, and more or less threatening environment, the behavior of an animal is adaptive as long as the behavior allows the animal to survive. Under the same conditions, the behavior of a robot is considered to be adaptive as long as the robot can continue to perform the functions for which it was built. (Meyer and Guillot 1991)

So the notion of adaptiveness is in general a normative one. In the case of the robots the norms come from the builder's intentions; in the case of animals, survival, or the successful production of offspring which in itself requires some form of survival, is the 'aim' of adaptive behaviour; there are potentially controversial extensions to this, when genes, or groups, are considered as the units of selection. The concept of *adaptation* is of course central to evolutionary biology, and to the notion of *natural selection*. It has been suggested by many that careless use of these terms has been prevalent and misleading (Varela *et al.* 1991, Lewontin 1983, Gould and Vrba 1981).

The core of the argument is the question of whether adaptation, and selection, imply a pre-existing absolute standard, a universal God-given scale of fitness against which any organism can be measured. The traditional, neo-Darwinian 'right', so the argument is sometimes presented, in claiming that all significant characteristics of organisms are the consequence of generations of selection in favour of those variations that allow the organism to better match its environment, are thereby committed to the position of a pre-existing absolute fitness standard. The radical 'left' criticise this interpretation of natural selection firstly from a philosophical standpoint which rejects the notion of a pre-existing objective world; and secondly with the argument that this over-emphasis on fitness under-estimates many differing mechanisms in evolution.

To list briefly the points raised by Varela, Thompson and Rosch (Varela *et al.* 1991):

1. Linkage between genes, or pleiotropy, may mean that one gene for a 'bad' trait is, through historical accident, always associated with another gene for an 'adaptive' trait; or the same gene is simultaneously associated with two such traits. It is difficult to analyse selection for such traits.

2. Embryological developmental processes, through intrinsic self-organising properties of complex networks, can give order 'for free', without the need for selection (Kauffman 1993, Kauffman 1989).

3. Random genetic drift, and genetic 'hitch-hiking', can have significant effect despite selection — see Chapter 8.

4. Morphological stasis in some species, despite environmental change and despite genetic diversity, needs to be explained.

5. *If* it is allowed that under particular circumstances it can make sense to treat genes as units of selection, or groups, as well as individual organisms, then potential interactions between different such units of selection make a simplistic view of absolute fitness values difficult to sustain.

One suggestion put forward by the 'radical left' is that the word *selection* should be abandoned because of its associations, and instead the neologism of 'exaptation' introduced, which means something like 'selection against' (Gould and Vrba 1981). In this way, it is suggested, the emphasis is changed from the evolutionary process selecting *for*

traits that are fit, towards the idea of selection *discarding* whatever is incompatible with survival and reproduction.

Turning to Darwin's treatment of natural selection (Darwin 1859), 'variation under nature' is discussed immediately after 'variation under domestication'; and natural selection is introduced by reference to the artificial selection practised by farmers over the centuries. But by and large his treatment of selection is compatible with the emphasis on negative selection endorsed above. Of course, given Malthusian principles of exponential increase, in the absence of limitations, of organisms that reproduce successfully, coupled with finite limits on available resources, the disappearance of those organisms that are selected against is the other side of the coin to the proliferation of organisms-that-are-not-selected-against.

At this stage the red herring is often raised, that if 'survival of the fittest' simply means 'survival of the survivors', then it is a tautology of no import. This is to miss the point that what is being asserted is that with the evolutionary requirements of heredity, variation, and selection, survival of *lineages over timespans of many generations* is a consequence of survival through to reproduction of *individuals over individual lifespans*.

Within particular well-defined (and usually short-term, contexts), particular traits such as the length of a giraffe's neck, or a bird's foraging strategy may well be usefully treated as positive and adaptive traits that are selected *for*, but this approach cannot be extended to the broader picture. There is no universal and objectively set fitness landscape against which bacteria, jellyfish and humans can all have their traits measured for adaptiveness.

But what notion of adaptiveness can we use when we come to consider artificial evolution?

4.3 Adaptation and Selection in an artificial world

Here it seems we have come full circle, in that the concept of natural selection in evolution is derived from the selection practised artificially by farmers on their livestock; and now we are borrowing ideas from evolution to apply to the breeding of artificial livestock. Of immediate relevance here is the distinction between evolution and optimisation, discussed in Chapter 3.

There are two different levels at which something can be described as adaptive, and these should be distinguished. A well-shaped axe may be well-adapted to its use, but is not itself an adaptive system. Here I am concentrating on the general nature of adaptiveness which will be required for complex systems, and potentially autonomous robots,

in the immediate future. A large telephone exchange is a complex system, but it would not be considered adaptive unless some sophisticated software was, for instance, capable of re-routing calls when lines were broken or overloaded. To cope with changing and unpredictable circumstances is the hallmark of this second meaning of adaptiveness.

Unpredictability and changeability can come in varying degrees. At the bottom end of the scale it can be fully described in advance, and the appropriate behaviour to respond to it given in a finite set of rules. The telephone exchange will behave differently when somebody dials 1 rather than 0, and we are not sufficiently impressed by that to deem this adaptive behaviour. At the top end of the scale, cataclysmic changes — a meteorite hitting the telephone exchange — cannot possibly be survived.

It is at scales intermediate between these two that we require adaptive behaviour, and the boundaries of this intermediate area cannot, by their very nature, be clearly pointed out. The upper boundary — just how far can we go towards catastrophe and still expect telephone exchanges, robots or humans to be adaptive enough to survive? — cannot be easily predicted; we rely on the fact that large parts of our world does not often change very much. The bottom boundary, since it lies by definition beyond any point which can be characterised by a complete description of all relevant factors, cannot be clearly delineated.

Traditional cognitivist AI works on the assumption that all relevant factors can be formalised completely in terms of rules, and hence misses the point that cognition and adaptation only become interesting once this point has been passed. This lies at the root of the Dreyfus' criticism of AI (Dreyfus 1972). This criticism has often been misunderstood as asserting that AI was in principle impossible. But a careful reading of Dreyfus shows that this was never asserted; rather, that the attempt to create AI by *rule-based* systems was doomed to failure.

4.4 Cognition

Metaphors for the mind or the brain go through fashions, usually based on the prominent technology of the day; hydraulic machinery, telephone exchanges, computers. The computer metaphor has in recent decades been so all-pervasive that its tenets have ceased to be made explicit. It is all the more dangerous when it is taken for granted, and left out of any debate. These assumptions have affected the directions taken in connectionist research, which could be (but rarely are) fitted into a different metaphor.

Connectionism (or Artificial Neural Networks, or Parallel Distributed Processing) is frequently promoted as a parallel form of computation, or of information processing[2]. Many applications are indeed just this, but the danger is that when connectionism is proposed either as a model of the mind, or as a technique for producing an 'artificially intelligent' machines, the computational metaphor still lies unsaid in the background. This thesis is based on the assumption that such a cognitivist approach is flawed.

The alternative view taken here is that cognition, as ascribed to animals or potentially to machines, is something that can only be attributed to the conjunction of an organism and the world that it inhabits. From this it follows that it would be a category error to treat cognition as something 'done' by the brain, or a part of the brain. This view is to a large extent shared by a significant number of people who have in the past been regarded as radical, or more commonly been completely ignored by mainstream cognitive science and AI. The time has surely arrived, by now, when such views can be assumed to be recognised (even if not accepted) by a cognitive science audience.

It will be suggested, here and later in Chapter 5, that the mainstream Cartesian paradigm has gravely restricted the class of cognitive models, and in particular connectionist models that have in practice been investigated. The use of *time* in connectionist networks will be discussed in Chapter 5; in this chapter I will concentrate on the practical effects of an unconsidered notion of *representation*.

When trying to produce an 'artificially intelligent' machine that can behave autonomously within an environment. The 'brain' or 'nervous system' of the machine can be considered as a Black Box connected to sensors and actuators. The behaviour of the machine plus brain within its environment must be seen to be intelligent, or sensible; at a minimum, that it should maintains its viability over a period of time.

The question then is, 'What to put in the Black Box?'. The computationalists will say that it should be computing appropriate outputs from its inputs. Or possibly they may say that whatever it is doing should be *interpretable* as doing such a computation.

4.5 What is the Computer Metaphor?

The concepts of computers and computations, and programs, have a variety of meanings which shade into each other. On the one hand a computer is a formal system with the

[2]Indeed, in the early years of the current resurgence in connectionism, a lot of effort was spent in trying to convince people that these networks *were* doing computation, in order that the field could gain respectability — personal communication, G.E. Hinton

same powers as a Turing Machine (...assuming the memory is of adequate size). On the other hand a computer is this object sitting in front of me now, with screen and keyboard and indefinite quantities of software.

A program for the formal computer is equivalent to the pre-specified marks on the Turing machine's tape. For a given starting state of this machine, the course of the computation is wholly determined by the program and the Turing machine's transition table; it will continue until it halts with the correct answer, unless perhaps it continues forever — usually considered a *bad thing*!

On the machine on my desk I can write a program to calculate a succession of co-ordinates for the parabola of a cricket-ball thrown into the air, and display these both as a list of figures and as a curve drawn on the screen. Here I am using the machine as a convenient fairly user-friendly Turing machine.

However most programs for the machine on my desk are very different. At the moment it is (amongst many other things) running a word-processing program. It sits there and waits, sometimes for very long periods indeed, until I hit a key on the keyboard, when it virtually immediately pops a symbol into an appropriate place on the screen; unless particular control keys are pressed, causing the file to be written, or edits to be made. Virtually all of the time the program is waiting for input in the form of interrupts, which it then processes near-instantaneously. In general it is a *good thing* for such a program to continue for ever, or at least until the exit command is keyed in.

The cognitivist approach asserts that something with the power of a Turing machine is both necessary and sufficient to produce intelligence; both human intelligence and equivalent machine intelligence. Although not usually made clear, it would seem that something close to the model of a word-processing program is usually intended; i.e., a program that constantly awaits inputs, and then near-instantaneously calculates an appropriate output before settling down to await the next input. Life, so I understand the computationalists to hold, is a sequence of such individual events, perhaps processed in parallel.

4.6 What is a Representation?

The concept of symbolic reference, or representation, lies at the heart of analytic philosophy and of computer science. The underlying assumption of many is that a real world exists independently of any given observer; and that symbols are entities that can 'stand for' objects in this real world — in some abstract and absolute sense. In practice, the role

of the observer in the act of representing something is ignored.

Of course this works perfectly well in worlds where there is common agreement amongst all observers — explicit or implicit agreement — on the usages and definitions of the symbols, and the properties of the world that they represent. In the worlds of mathematics, or formal systems, this is the case, and this is reflected in the anonymity of tone, and use of the passive tense, in mathematics. Yet the dependency on such agreement is so easily forgotten — or perhaps ignored on the assumption that mathematics is the language of God.

A symbol P is used by a person Q to represent, or refer to, an object R to a person S. Nothing can be referred to without somebody to do the referring. Normally Q and S are members of a community that have come to agree on their symbolic usages, and training as a mathematician involves learning the practices of such a community. The vocabulary of symbols can be extended by defining them in terms of already-recognised symbols.

The English language, and the French language, are systems of symbols used by people of different language communities for communicating about their worlds, with their similarities and their different nuances and clichés. The languages themselves have developed over thousands of years, and the induction of each child into the use of its native language occupies a major slice of its early years. The fact that, nearly all the time we are talking English, we are doing so to an English-speaker (including when we talk to ourselves), makes it usually an unnecessary platitude to explicitly draw attention to the community that speaker and hearer belong to.

Since symbols and representation stand firmly in the linguistic domain, another attribute they possess is that of arbitrariness (from the perspective of an observer external to the communicators). When I raise my forefinger with its back to you, and repeatedly bend the tip towards me, the chances are that you will interpret this as 'come here'. This particular European and American sign is just as arbitrary as the Turkish equivalent of placing the hand horizontally facing down, and flapping it downwards. Different actions or entities can represent the same meaning to different communities; and the same action or entity can represent different things to different communities. In Mao Tse-Tung's China a red traffic light meant *GO*.

In the more general case, and particularly in the field of connectionism and cognitive science, when talking of representation it is imperative to make clear who the users of the representation are; and it should be possible, at a minimum, to suggest how the convention

underlying the representation arose. In particular it should be noted that where one and the same entity can represent different things to different observers, conceptual confusion can easily arise. When in doubt, always make explicit the Q and S when P is used by Q to represent R to S.

In a computer program a variable `pop_size` may be used by the programmer to represent (to herself and to any other users of the program) the size of a population. Inside the program a variable i may be used to represent a counter or internal variable in many contexts. In each of these contexts a metaphor used by the programmer is that of the program describing the actions of various homunculi, some of them keeping count of iterations, some of them keeping track of variables, and it is within the context of particular groups of such homunculi that the symbols are representing. But how is this notion extended to computation in connectionist networks?

4.7 Representation in Connectionism

When a connectionist network is being used to do a computation, in most cases there will be input, hidden and output nodes. The activations on the input and output nodes are decreed by the connectionist to represent particular entities that have meaning for her, in the same way as `pop_size` is in a conventional program. But then the question is raised — 'what about internal representations?'.

If a connectionist network is providing the nervous system for a robot, a different interpretation might be put on the inputs and outputs. But for the purpose of this section, the issues of internal representation are the same.

All too often the hidden agenda is based on a Platonic notion of representation — what do activations or patterns of activations represent in some absolute sense to God? The behaviour of the innards of a trained network are analysed with the same eagerness that a sacrificed chicken's innards are interpreted as representing one's future fate. There is, however, a more principled way of talking in terms of internal representations in a network, but a way that is critically dependent on the observer's decomposition of that network. Namely, the network must be decomposed by the observer into two or more modules that are considered to be communicating with each other by means of these representations.

Where a network is explicitly designed as a composition of various modules to do various subtasks (for instance a module could be a layer, or a group of laterally connected nodes within a layer), then an individual activation, or a distributed group of activations, can

be deemed to represent an internal variable in the same way that i did within a computer program. However, unlike a program which wears its origins on its sleeve (in the form of a program listing), a connectionist network is usually deemed to be internally 'nothing more than' a collection of nodes, directed arcs, activations, weights and update rules. Hence there will usually be a large number of possible ways to decompose such a network, with little to choose between them; and it depends on just where the boundaries are drawn just who is representing what to whom.

It might be argued that some ways of decomposing are more 'natural' than others; a possible criterion being that two sections of a network should have a lot of internal connections, but a limited number of connecting arcs between the sections. Yet curiously this does not usually hold for what is perhaps the most common form of decomposition, into layers. The notion of a distributed representation usually refers to a representation being carried in parallel in the communication from one layer to the next, where the layers as a whole can be considered as the Q and S in the formula "P is used by Q to represent R to S".

An internal representation, according to this view, only makes sense relative to a particular decomposition of a network chosen by an observer. To assert of a network that it contains internal representations can then only be justified as a rather too terse shorthand for asserting that the speaker proposes some such decomposition. Regrettably this does not seem to be the normal usage of the word. While claiming that my usage of the word representation as outlined above is the careful and principled form that underlies the confused and careless way in which the word is frequently used, I am aware of the dangers of claiming to be the only person in the platoon that is in step. Nevertheless, until I see an alternative formulation clearly laid out, I shall continue to be puzzled by much of what is written on the subject.

In (Hinton *et al.* 1986), reprinted in (Boden 1990), an attempt is made to make sense of distributed representation in connectionist networks. No acknowledgment of any necessity to specify *what* is representing something *to what* is made. Yet that paper can be sensibly interpreted as implicitly taking different layers in a network to be the different *whats*. When a more abstract, philosophical approach to discussion of connectionist representation is taken, as for instance in a collection of papers in (Ramsey *et al.* 1991), the absence of any clarification or specification of the *whats* makes it difficult, from my perspective, to work out what, if anything, is being said.

The gun I reach for whenever I hear the word *representation* has this engraved on it: "When P is used by Q to represent R to S, *who is* Q *and who is* S?". If others have different criteria for what constitutes a representation, it is incumbent on them to make this explicit. In particular I am puzzled as to how they can reconcile (if they believe it is not necessary to specify Q and S) the same symbol representing different things to different communities.

4.8 Are Representations Needed?

With this approach to the representation issue, then any network can be decomposed (in a variety of ways) into separate modules that the observer considers as communicating with each other. The interactions between such modules can *ipso facto* be deemed to be mediated by a representation. Whether it is useful to do so is another matter.

Associated with the metaphor of the mind (or brain, or an intelligent machine) as a computer go assumptions of functional decomposition. Since a computer formally manipulates symbols, yet it is light waves that impinge on the retina or the camera, surely (so the story goes) some intermediate agency must do the necessary translating. Hence the traditional decomposition of a cognitive system into a perception module, which takes sensory inputs and produces a world model; this is passed onto a central planning module which reasons on the basis of this world model; passing on its decisions to an action module which translates them into the necessary motor actions. This functional decomposition has been challenged, and an alternative behavioural decomposition proposed, by Brooks in, e.g., (Brooks 1991).

Marr (in (Marr 1977), reprinted in (Boden 1990)) classifies AI theories into Type 1 and Type 2, where a Type 2 theory can only solve a problem by the simultaneous action of a considerable number of processes, *whose interaction is its own simplest description.* It would seem that Type 2 systems can only be decomposed arbitrarily, and hence the notion of representation is less likely to be useful. This is in contrast to a Type 1 theory, where a problem can be decomposed into a form that an algorithm can be formulated to solve, by *divide and conquer.* Type 1 theories are of course the more desirable ones when they can be found, but it is an empirical matter whether they exist or not. In mathematics the 4-colour theorem has been solved in a fashion that requires a large number of special cases to be exhaustively worked out in thousands of hours of computation (Appel and Haken 1989). It is hoped that there were no hardware faults during the proof procedure,

and there is no way that the proof as a whole can be visualised and assessed by a human. There is no *a priori* reason why the workings of at least parts of the brain should not be comparably complex, or even more so[3]. This can be interpreted as: there is no *a priori* reason why all parts of the brain should be in such a modular form that representation-talk is relevant. The answer to the question posed in the title of this section is *no*. This does not rule out the possibility that in some circumstances representation-talk *might* be useful, but it is an experimental matter to determine this.

4.9 The Dynamical Systems Alternative

If one abandons the computer metaphor, the problem of how to make an intelligent machine becomes: what sort of physical system should be put inside the Black Box of its nervous system so that it behaves appropriately in its environment? The answer suggested here, and developed in Chapter 5, is simply that it should be a dynamical system whose states vary in time, according to the current values of its states (a computer is a subclass of dynamical systems). A cogent argument for a dynamical systems perspective has been put forward by Beer and Gallagher in (Beer and Gallagher 1992, Beer 1992).

The starting point taken by them can be characterised by this quotation (Beer and Gallagher 1992), page 91:

> ...animals are endowed with nervous systems whose dynamics are such that, when coupled with the dynamics of their bodies and environments, these animals can engage in the patterns of behavior necessary for their survival.

The emphasis on dynamics (rather than treating robot control as the solution to a sequence of static problems) means that the dimension of time cannot be ignored; and indeed it is for this reason that I will be arguing that programs and indeed many classes of connectionist networks, which inherently ignore the dimension of time except solely its sequential aspect, are inappropriate. Beer's work fully recognises this issue, and the time constants of the model neurons in his networks is of as much significance as the weights on the connections between them; and there is no reason to restrict networks to feedforward.

Of course, abandoning the computer metaphor does not prevent one from using a computational model of a physical system and calculating its behaviour, just as one can

[3]For the purposes of making an intelligent machine or robot, it has in the past seemed obvious that only Type 1 techniques could be proposed. However evolutionary techniques need not restrict themselves in this fashion — Chapter 3.

calculate the parabola of a cricket ball without claiming that the ball itself is some form of computer.

The computational approach would imply that the 'brain' of a machine has access through its sensors to information about the machine's world, which it can then reason about. We are dismissing this notion, and instead have to rely on processes whereby the physical system within the Black Box can adapt itself according to some given criteria. The present thesis, of course, is that this can be done through evolution.

The view of cognition entailed by this attitude fits in with Varela's characterisation of cognition as *embodied action* (Varela *et al.* 1991), pages 172–173:

> By using the term *embodied* we mean to highlight two points: first, that cognition depends upon the kinds of experience that come from having a body with various sensorimotor capacities, and second, that these individual sensorimotor capacities are themselves embedded in a more encompassing biological, psychological, and cultural context. By using the term *action* we mean to emphasize once again that sensory and motor processes, perception and action, are fundamentally inseparable in lived cognition. Indeed, the two are not merely contingently linked in individuals; they have also evolved together.

4.10 Higher-level cognition

Emphasis laid on low-level sensorimotor processes is one thing that distinguishes the field of Artificial Life, or AL, from traditional Artificial Intelligence, AI. For some people interested in so-called higher-level cognitive capacities, such as planning or natural language, their interests are so remote from this low-level emphasis that they would suggest that there is no practical connection.

There is some merit in this attitude. The possibility of reduction of high-level theories, such as thermodynamics, to a lower-level substrate, such as molecular kinematics, does not of itself deny the validity of theories expressed solely in terms of high-level concepts. However, since so much of evolution is dependent on historical accidents, and since it is clear that language turned up only very recently on the evolutionary time-scale, this raises more of a question-mark on the potential independence of theories of natural language than there is on thermodynamics.

If a methodology of incremental evolution of robot 'nervous systems', starting with low-level sensorimotor capabilities, is ever going to shed light on natural language processing, it must be freely conceded that this will not happen for an exceedingly long time to come. Regardless of this, study of low-level sensorimotor processes has intrinsic merit; and for the engineer trying to build autonomous robots is clearly of prime importance.

CHAPTER 5

Real Time Dynamical Systems[1]

5.1 What building blocks for a control system?

We are relying on evolution for the design of a control system, but we must choose appropriate building blocks for it to work with. Some have advocated production rules (Classifier Systems are the GA version of these). Some propose LISP-like programming languages (Koza 1990). Brooks (Brooks 1992) has proposed using Koza's ideas applied to a high-level behaviour language. Beer (Beer and Gallagher 1992) has used dynamical neural networks. It is only this last approach that broadly speaking is advocated here.

The intuition is — and it will be supported by results in the evolutionary robotics work reported in Chapters 12 and 13 — that the primitives manipulated by the evolutionary process should be at the lowest level possible. Any high level semantic groupings inevitably incorporate the human designer's prejudices, and give rise to a more coarse- grained fitness landscape with more steep precipices; evolution requires that the fitness landscape is in general not too rugged. It might be thought that the use of low-level primitives necessitates aeons of trials before any interesting high-level behaviour emerges, but initial experience with simulations indicates otherwise.

Another factor concerning high-level languages is that the injection of noise into a system seems contrary to the rationale for such languages. The injection of noise into the lowest levels of a control system can be shown to have valuable effects on the dynamics; and has the additional benefit of blurring the fitness landscape and making it less rugged for evolution.

The criteria advocated for deciding on component primitives are:

They are the primitives of a dynamical system.

The system should operate in real time, and the timescales on which the components

[1] Much of this chapter first appeared in a Technical Report (Harvey 1992c).

work must be in some sense appropriate for the world the robot inhabits.

The system should be 'evolvable', not 'brittle'; in the sense that many of the possible small changes in the way components are bolted together should result in only small changes in resulting behaviour.

Incremental change in the complexity of any structure composed of such primitives should be possible.

There may be many possible components and general architectures that meet these criteria. The particular choice focused on here is that of recurrent dynamic realtime networks, where the primitives are the nodes in a network, and links between them — it seems to me likely that different choices that work will be functionally equivalent to this one (for instance, Brooks' use of Augmented Finite State Automata in subsumption architectures is functionally equivalent). The nodes act much as many artificial 'neurons' in a neural network, though with particular characteristics. The links are unidirectional, have time delays between units, and currently are restricted to a fixed unit weight. There is no restriction on the direction of connectivity.

What will be suggested in this chapter is that the mainstream Cartesian paradigm has gravely restricted the class of connectionist models that have in practice been investigated. In Chapter 4 the practical effects of an unconsidered notion of representation were discussed; here I will consider use of *time* in connectionist networks. A sketch of a broader class of connectionist networks will be given.

5.2 Time in Computations and in Connectionism

One particular aspect of a computational model of the mind which derives from the underlying Cartesian assumptions common to traditional AI is the way in which the issue of *time* is swept under the carpet — only the sequential aspect of time is normally considered. In a standard computer operations are done serially, and the lengths of time taken for each program step are for formal purposes irrelevant. In practice for the machine on my desk it is necessary that the time-steps are fast enough for me not to get bored waiting. Hence for a serial computer the only requirement is that individual steps take as short a time as possible. In an ideal world any given program would be practically instantaneous in running, except of course for those unfortunate cases when it gets into an unwanted infinite loop.

A common connectionist assumption is that a connectionist network is in some sense a parallel computer[2]. Hence the time taken for individual processes within the network should presumably be as short as possible. They cannot be considered as being effectively instantaneous because of the necessity of keeping parallel computations in step. The standard assumptions made fall into two classes.

1. The timelag for activations to pass from any one node to another it is connected to, including the time taken for the outputs from a node to be derived from its inputs, is in all cases exactly one unit of time (e.g. a back-propagation, or an Elman network).

2. Alternatively, just one node at a time is updated independently of the others, and the choice of which node is dealt with next is stochastic (e.g. a Hopfield net or a Boltzmann machine).

The first method follows naturally from the computational metaphor, from the assumption that a computational process is being done in parallel. The second method is closer to a dynamical systems metaphor, yet still computational language is used. It is suggested that a network, after training, will when presented with a particular set of inputs then sink into the appropriate basin of attraction which appropriately classifies them. The network is used as either a distributed content-addressable memory, or as a classifying engine, as a module taking part in some larger-scale computation. The stochastic method of relaxation of the network may be used, but the dynamics of the network are thereby made relatively simple, and not directly relevant to the wider computation. It is only the stable attractors of the network that are used. It is no coincidence that the attractors of such a stochastic network are immensely easier to analyse than any non-stochastic dynamics.

It might be argued that connectionists are inevitably abstracting from real neural networks, and inevitably simplifying. In due course, so this argument goes, they will slowly extend the range of their models to include new dimensions, such as that of time. What is so special about time — why cannot it wait? This argument misses the point that for dynamical control systems operating in environments where physical events have natural timescales, the dimension of time is not an optional extra, but fundamental. Clocked networks form just one particular complex subset of all realtime dynamical networks.

A much broader class of networks is that where the timelags on individual links between nodes is a real number which may be fixed or may vary in a similar fashion to weightings

[2]Such assumptions, and the criticisms offered here, do not apply to whole areas of connectionist research, such as attempts to realistically model the dynamics of neurons.

on such links[3].

5.3 Time in neurobiology

Computers tend to be built with a global clock-tick, measured in MHz. Neurons seem to operate in the millisecond time scale, and there is no evidence for global synchronisation in the central nervous system on such a scale. The nodes of most connectionist networks are trivial compared to the complexity of most neurons that neurobiologists study. Treating neurons as complex black boxes that interact with each other via axons and dendrites, it appears that most (though not all[4]) such interactions are in the form of all-or-none spikes, seemingly binary in nature and sharply located in time. One common assumption is that in some sense neurons are passing information between each other 'encoded' in the rate of firing of spike trains; such a code could cater for real-valued numbers. The average discharge rate of neocortical neurons is only of the order of 50 to 100 spikes per second (Crick and Asanuma 1986). Hence periods of 50–100 milliseconds would be needed to start coding real numbers at all accurately. Yet complex behavioural responses to stimuli take place in animals in less than one second, despite involving many neurons in sequence.

With these difficulties facing this interpretation of spikes in real brains, another possibility is the stochastic option; as, in artificial connectionist networks, with Hopfield nets or Boltzmann machines. Here the firing of any one neuron is considered to be stochastic, but the network is such that useful properties result from the statistics of a long series of randomly chosen updates.

There is a third interpretation of the significance of timing of spikes, which I will summarise here by looking at neurobiological experiments by Abeles (Abeles 1982), and somewhat similar theoretical work by von der Malsburg and Bienenstock (Malsburg and Bienenstock 1986). In these, it is suggested that the individual timing of neuronal events could be a major factor in the behaviour of a neural network. I have not previously seen these sorts of proposals carried over to artificial connectionist networks, and I suggest that they give a natural way in which real time dynamics can be incorporated into such networks.

[3] For a simple model without loss of generality any time taken for outputs to be derived from inputs within a node can be deemed to be zero, as long as such a time is added on to the time-delays on all outgoing links from that node.

[4] Axons can propagate graded potentials, but due to attenuation this seems only to be of use over very short distances; e.g. between nearby neurons in the retina.

5.4 Abeles' electrophysiological studies

The core of his experimental technique (Abeles 1982) is the ability to analyse the patterns of activity from a single electrode inserted into the midst of a group of neurons, and distinguish automatically by the different shapes of the spikes detected, which of two, three, or even up to seven different neurons it came from (in the auditory cortex of an unaesthetised muscle-relaxed cat). If two spikes were simultaneous, no match with the templates could be made, but barring this the spikes are individual enough in character for a single stream of output from the electrode to be converted into precise timings of spikes from identified neurons; the interesting results came from analysing three neurons.

The assumption is used that a neuron has no 'memory' of what happened before a spike, and hence a spike train can be treated statistically as a renewal process; all the information of interest can be contained in a representation by renewal density, by plotting histograms of how often each particular neuron fires at varying times after some specific event (spikes/second plotted against milliseconds after event). If the event is the previous firing of the same neuron, auto-renewal density will be plotted; experimental results for these show a slight tendency for there to be more spikes in the the following 100 milliseconds than a purely random process would give; but no signs of periodicity. If the event is the previous firing of some other specific neuron, then cross-renewal density is plotted.

If the inputs to a neuron are considered as the sum of thousands of independent spikes, some excitatory and others inhibitory, firing at random times, the resulting sum will be a wildly fluctuating membrane potential with roughly a gaussian distribution of amplitudes. If for instance this distribution was such that the membrane potential was above threshold value 5/1000 of the time, and if each spike corresponded to a millisecond of being above the threshold, then loosely speaking an average rate of firing of 5 spikes/second would result; the exact calculations are not as important as Abeles' discussion of the relative effects of co-ordinated and unco-ordinated inputs.

Theoretical analysis, using some reasonably plausible assumptions about the connectivity and average firing rate of neurons, and the timescale of the falling postsynaptic potential, gives the result that one 'unco-ordinated' input spike to a neuron contributes on average 1/333 spikes to the output, whereas 29 'co-ordinated' ones suffice to produce one output spike - a factor of 11 to 1 in efficacy. These sort of approximate figures give some handle on the picture of a neuron as a leaky integrator, with (in the example hypothesized) some $20,000 \times 5$ inputs per second.

Abeles then gives experimental results indicating that in the neurons he studied the observed figures were of the order of 0.06 spikes per 'unco-ordinated' input against 7.7 'co-ordinated' ones sufficing to produce an output spike - a much smaller factor of about 2 to 1 in efficacy. He suggests that a larger factor might have been shown but for the bias in his experimental techniques towards sampling pairs of cells with strong synaptic contacts. Nevertheless, the significance of co-ordinated spikes was established.

We now reach the novel part of Abeles' work, looking for those complex interactions between cells which by their very nature will not be observed by simpler experimentation. By timing the spikes on 3 nearby neurons, which his apparatus is able to do with a single simple probe, he builds up compound renewal density histograms giving the relative frequencies at which A fires at time x after a firing of C, and B fires at time y after the same firing of C. Similar graphs of B and C timed after the firing of A, and C and A timed after the firing of B, are built up, giving a picture of the compound renewal densities of A, B and C.

Now as well as the simpler, e.g. common input, relationships, more complex time-locked activity becomes apparent in some triplets. For instance, he shows examples where A fires at a high rate 25 milliseconds after those occasions when B and C fire together; and in the same group ABC, C fires at a high rate shortly after those occasions when B fires 375 milliseconds after A. These patterns, with consistent delays of up to 400 milliseconds, imply consistently repeated patterns which must involve large numbers of intermediate synapses. Since the probabilities of random failure at any one link of a long chain of neurons make the chance of a signal reliably reaching the end negligible, some more robust form of chain must be postulated, and Abeles suggests the 'synfire chain'.

If each member of a set of cells has connections onto each member of a further set, then when the first set fires in synchrony, the numerous simultaneous inputs to each of the second set will, if the parameters are appropriate, cause the second set to fire in synchrony. If the connectivity of the network is such that further sets are subsequently fired, then we can have a long chain of synchronous sets firing in a time-locked fashion, and with that robustness to random perturbations which is lacking from a chain of individual neurons. Other nearby neurons may also be fired with the passing of the activity, without themselves contributing to the propagation of that activity. And cells within such a synfire chain will individually fire due to random outside activity, without setting off the synfire chain. Furthermore, one or many neurons can participate in two or more potential synfire

chains, and the activity of either or both chains could pass through the same neurons without interfering with each other.

5.5 Short-term plasticity and Synaptic patterns

Von der Malsburg (Malsburg and Bienenstock 1986) develops a similar, but more dynamical, approach. Whereas Abeles is concerned to show the stability of behaviour of a synfire chain, or several intertwined such chains, that could exist in the brain, von der Malsburg posits conditions that would allow such signal structures to be developed on the fly.

Taking simple Hebbian rules, he proposes that synaptic weightings, or 'transmission efficacy', should increase or decrease according to whether incoming spikes were positively or negatively related to outgoing spikes. He proposes that these changes are immediate (i.e. effective within a few milliseconds) but limited in magnitude of variation about the current long-term mean; and that the more familiar idea of long-term weightings, often thought of as holding long-term memory, can be a time-integration of these short-term changes.

Whereas the synfire chain could be thought of as a sequence of cell assemblies — where each assembly fires together, and is connected such as to trigger off the next assembly in turn — von der Malsburg looks at pairs of neurons which may be interconnected only indirectly via intermediaries. If nevertheless there are a large number of different pathways from one to the other, such that the transmission time down each pathway is equal, then there will be a better than random chance that a spike in the first will be followed by a spike in the last. Whereas Abeles' probes looked only at neighbouring neurons, these ones might well be some distance apart. Indeed it is suggested that very short paths between the two neurons, since they will be relatively few in number, are unlikely to produce these favourable conditions.

A 'synaptic pattern' is a stable pattern of firing — stable in the sense that it has a higher than random chance of recurring — which is selected out of all the billions of possible patterns of activation by the existence within of many micro-configurations with the favourable pathways mentioned in the previous paragraph. The existence of synaptic patterns in this sense cannot be modelled either in neural networks with firings co-ordinated by a notional universal clock-tick; nor in stochastic simulated networks, such as a Boltzmann machine. Yet within a complexly connected cortex-like network there will be relatively stable states, or attractors, of the dynamics, which the posited short-term

plasticity changes would tend to reinforce.

In contrast to a synfire chain, where the activity proceeds like a travelling wave, there need be no immediately apparent topological structure to a synaptic pattern, with the exception that there will be a higher rate of coincident spike inputs than in a network without a synaptic pattern. Apart from this, there will be nothing statistically significant visible in the behaviour of any single neuron.

5.6 Time in dynamical networks

The point has been made that the usual simplification of neural network models of the brain into, at one end of the spectrum, co-ordinated activation at the tick of a universal clock, at the other end, a randomly ordered updating, inevitably misses out those features which a network with neurons behaving to some extent as co-incidence detectors will develop.

Whether neurons in the brain act as leaky integrators of their weighted inputs, and if so what sort of time constant characterises the leakiness, is a matter to be investigated by experiment. Likewise the question of how synaptic weightings change with activations before and after the synapse, and what sort of time scales this occurs on. A survey of the different proposals that come under the heading of neural networks shows an enormous range of ideas based on the simple primitives of nodes, activations, links and weights. The importance of co-incident timing of spikes, with its consequences, is I suggest an important new addition to the field. For an engineer looking for a relatively easy method to implement a realtime control system, these ideas are very fruitful.

5.7 Networks for Control Systems

Taking into account these issues relating to time, there are good grounds for thinking that a generalised form of connectionist network could be one very appropriate class of control system. Such networks can be defined using a minimal set of just three basic principles.

1. The 'brain' should be a physical system, occupying a physical volume with a finite number of input and output points on its surface.

2. Interactions within the brain should be mediated by physical signals travelling with finite velocities through its volume, from the inputs, and to the outputs.

3. Subject to some lower limit of an undecomposable 'atom' or node, these three principles apply to any physical subvolume of the whole brain.

A justification for the third principle is that of the incremental development of the whole by alterations and additions over evolutionary timescales. The consequence of these principles, as can be seen by shrinking in any fashion the surface containing the original volume, is a network model where internal nodes are the undecomposable atoms, and connections between inputs, internal nodes and outputs are through directed arcs by signals taking finite times. Such a network can be arbitrarily recurrent. The assumption of only a finite number of input/output points on any surface rules out of this model such more general methods of physical interaction as might be assumed to be involved with, e.g. chemical neurotransmitters in the human brain.

No assumptions about the operations of the nodes have yet been made. The simplest assumptions would be those of standard connectionist models. Input signals are weighted by a scalar quantity; all output signals are identical when they leave the node, being calculated from the weighted sum of the inputs. If this weighted sum is passed through a sigmoid or thresholding function then we have the non-linear behaviour we have learnt to know and love. So far the only generalisation this model has when compared with the picture given in (McClelland and Rumelhart 1986) is that timelags between nodes need to be specified. But a whole new universe of possible dynamical behaviours is opened up by this extension, particularly if the timelags can be real-valued, rather than restricted to discrete values.

Such networks are more difficult to analyse, but still possible to synthesize. With an evolutionary approach it may not be necessary to analyse *how it works*, but rather one should be able to assess *how good is the behaviour it elicits*. This is no short-cut recipe, but requires that the internal complexity of the 'brain' (of an organism or a machine) be dependent on the history of interactions with its world; the more the complexity that is required, the longer the history that is needed to mould it.

5.8 Learning

In the last part of this chapter, I shall be considering the how notions of learning, which in many connectionist's eyes is normally related to changes of connection weights in a real or artificial neural network, fits in with the proposals above. To avoid confusion, I had better emphasise that I am not here proposing a theory of learning. I am merely investigating

what minimum conditions are necessary for artificial evolution to produce artefacts that display learning behaviour.

The evolutionary process itself can be considered as a form of reinforcement learning. The species, as assessed by looking at a representative member at any one time, can be treated as learning if one considers a period of time when the environment is fixed and the evaluation according to the fitness function improves. Here I shall be considering the separate question of what is necessary and sufficient to have individuals — cognitive structures or robots — learning within their lifetimes.

Whether the ability of an artificial autonomous system to learn is a 'good thing' or not depends entirely on what sort of world it will operate in; for some, hard-wired behaviour may be adequate. But in many cases, where a system is evolved for a broad class of worlds, yet within its lifetime or periods of its lifetime its world has particular idiosyncratic characteristics, then it makes sense for individual learning to tailor the behaviour of the system to its current environment. The interaction between individual learning and evolution, and the advantages it can bring, are covered in more detail in Chapter 8. A particular example of a system where such advantages are sought is that of Ackley and Littman (1991), using Evolutionary Reinforcement Learning; this was summarised in Chapter 2. On analysing the relative success achieved using learning without evolution, as compared to the relatively poor results using evolution without learning, Ackley and Littman reach the conclusion that:

> Our explanation is that *it is easier to generate a good evaluation function than a good action function.*
> ...it can be much easier to specify *goals* than *implementations* — assuming, of course, the existence of a search and learning process adequate to fill in the details. [Pp. 497–498, original emphases]

5.9 Levels of Description

I will contrast here a behavioural level of description of a system, and a mechanical level of description of the same system. Consider four conditions which might be proposed as a definition of learning in a system:

1. The system changes its behaviour over time
2. This change 'improves' its behaviour in some sense.
3. The change in behaviour was at the cost of some work or effort by the system.

4. Some *particular kind* of internal change of state must have taken place in the system.

The first three conditions refer to a behavioural level of description, and the last to a mechanical level of description. The first two are the least controversial, and I personally would endorse them.

The third condition I have some sympathy with. The sort of situation where it comes to bear is that of the cockroach which apparently has one efficient gait when all its legs are intact, yet on the loss of a leg the dynamics of its control system are such that it slips instantly into a different 5-legged gait. We seem reluctant to say that this change of gait is the outcome of a learning process, and condition 3 seems appropriate to discriminate here. Of course, what counts as 'work' or 'effort' is ill-defined itself; outside (perhaps) mathematics, all definitions bottom out somewhere in some ill-defined consensus.

It is the final suggested condition that I disagree with, insofar as it requires any particular type of internal change, at some mechanical level of description, before some change of behaviour can be characterised as learning. This will be considered in the context of a thought experiment later.

5.10 Change and Improvement

Any one behaviour itself is displayed over some finite time period. So we have:
From time t_A to t_{A+a} behaviour A occurred.
From time t_B to t_{B+b} behaviour B occurred.

Here there are two different timescales being considered; the first, shorter, timescale, is that necessary to establish that the current behaviour is A rather than B, or *vice versa*; the second, longer, timescale, is that which covers periods over which behaviour changes, e.g. from A to B. Simple change of behaviour is not sufficient to characterise a system as 'learning'. After all, thistledown dancing in the wind changes its behaviour all the time, for instance between increasing its height and decreasing its height, yet we do not say it is learning. Some normative criterion is also necessary, such that we can say behaviour B is *better* than behaviour A. This serves to emphasise the point that a simple mechanical, nuts-and-bolts level description of a system is not of itself sufficient to establish whether that system is capable of learning; as such a description has no normative aspect.

If the type of behaviour being considered is at a more abstract level, such as 'ability to do something', or 'representational knowledge', the same point holds true: the normative, non-mechanical part of the description, whether implicit or explicit, is necessary to

characterise 'learning'.

> On Monday she could not spell 'embarrass' (and was not trying to).
> On Friday she could spell 'embarrass' (but was not actually doing so).

This only counts as learning if the Friday follows the Monday; if it was the preceding Friday, we call the change 'forgetting'.

5.11 A thought experiment

Turning to the last proposed condition for learning suggested above, consider this thought experiment.

A mechanical mouse is placed in a T-maze, with two signal lights just before the choice-point of the T, one on each side. The mouse is given 100 trials, at the start of each of which a piece of mechanical cheese is placed (at random, determined by the toss of a coin) at the end of one or other arm of the T.

A separate switch determines whether the signal light on the *same* side as the cheese, or the *opposite* side is turned on for the benefit of guiding the mouse. Before the first trial this switch is set randomly, and after each trial a die is thrown, to give a 1 in 10 chance of reversing the switch.

The mechanical mouse chooses randomly on the first trial. As we observe successive trials we can see that the mouse thereafter biases its choice according to the signal light, based on whether the signal was correlated or anti-correlated with the cheese on the previous trial. Hence it succeeds thereafter every time, except for those occasions when the switch has just been reversed; then it learns from its one failure to adjust its behaviour accordingly. I believe that most people watching such a mouse would agree that this is learning behaviour. These mice conform to the third condition above, in that the failures at the reversal of the switch cause one-shot learning; this can be considered as work or effort in learning the new regime of correlation between lights and cheese.

When the lid is lifted from the brain of the mouse, however, it can be seen that the mechanism to implement this is trivial. It could have been artificially evolved, it could have been written in a very small number of lines of program code, or implemented in the form of a hard-wired electronic circuit. At this stage some sceptics will want to say that this is not really learning.

Patently, there has been some internal change of state involved, however the mouse's brain has been made. But if such a claim is based on the assertion that, for this to be

learning, some *particular* type of internal change must take place, then those who insist on this cannot know whether or not humans are capable of learning. For our knowledge of learning mechanisms in humans is inadequate. Yet it is clearly ridiculous to say that we do not know whether humans are capable of learning.

Hence it is unnecessary to demand that a system must be capable of some *particular* kind of internal change of state, as a pre-condition for it to be able to learn.

5.12 So what do we need for individual learning?

In artificial evolution, the normative criteria come from whatever evaluation function, explicit or implicit, drives selection. No further normative criteria imposed from outside are necessary.

At the mechanical level of description, that of deciding what sort of primitives to put into our artificial 'primaeval soup', it is sufficient that there should be components capable of operating on different timescales. The classical such components in artificial neural networks are the activations within the network which operate on a fast timescale; and the 'weights' which change on a slow timescale. These are by no means the only way in which operation on different timescales need to be made possible.

For instance, it could be argued that within a 'hard-wired' network, or a cellular automaton, in which the rules only covered changes at a single timescale, nevertheless a description of the system's behaviour at a different timescale can be realistic. If such a system in the absence of perturbation from outside is within one basin of attraction, and then a perturbation pushes it into a different basin of attraction, then such a movement between basins is at a different and longer timescale than that of the individual cell or 'neuron' update rules. This is sufficient for an observer, *if* the change in the system's behaviour can be characterised as an improvement, to be justified in calling this learning.

It remains true, however, that any autonomous system must by definition be able to cope with changes in its world at whatever timescale those changes occur. Hence the choice of primitives should be influenced by the timescales of the system's world, and it may well be a good idea to choose a range of primitives whose dynamics act over a range of timescales.

5.13 How should we organise the primitives?

There is an enormous and respectable literature on machine learning, on reinforcement learning techniques such as Q- learning and TD-learning; and of course on all sorts of neural network techniques for learning. The Ackley and Littman example mentioned above is a case where ideas from this domain have been bolted on to an artificial evolutionary scenario, to form a hybrid system, with some success.

Nevertheless, for exactly the reasons given in Chapter 3 in favour of evolution rather than design, I take the hard line and say that the human overseer should make no attempt to influence the way in which the evolutionary process arranges the primitives available to it. Of course, should subsequent analysis of evolved systems show that they can be interpreted as carrying out some classical learning technique, this would be of considerable interest. I would still maintain that the human's role here should be restricted to ensuring that the primitives available have a dynamic range on timescales appropriate to the environment.

5.14 Conclusions

As a preamble, nothing said herein should be taken as denying that connectionist networks can be used for doing a form of parallel computation. Nor should I be taken to be claiming that it is impossible that any part of the human brain could be usefully interpreted as performing some such parallel computation — or that such techniques should never be used in an intelligent machine.

What *is* being asserted is that the exclusive interpretation of connectionist networks within the computational metaphor — a pervasive practice grounded within the mainstream paradigm – - has severely limited the types of networks investigated. Particular consequences of this misdirection have been followed up.

Already, in Chapter 4, I have discussed how the carry over to connectionist networks of the often inappropriate notion of representation is associated with a desire to decompose networks into modules, often layers, which can be seen to be communicating interpretable messages between each other; in Marr's terms, a Type 1 decomposition. This has militated against the investigation of arbitrarily recurrent networks (with the exception of Hopfield-type nets using stochastic updates, which themselves tend to only make sense as a component within some larger computational system).

In this chapter the point has been made that the irrelevance of time in serial com-

putation, except as a dimension for ordering program steps, means that for the most part only two impoverished subclasses of networks have been analysed; where all internal interactions take a unit time step, or where individual nodes are updated in stochastic order. When timelags between nodes are given individual real values, in the same fashion that weightings often have, then a wider class of realtime control systems is possible; the complexity of behaviour may be more difficult to analyse, but with artificial evolution this need be no barrier to their use.

Further, it has been suggested that learning ability in some system is a behavioural-level description of that system. And that the only implication for the 'working parts' of a system able to learn is that components operating on differing timescales are necessary; slow-moving synaptic weight changes are but one possible technique.

CHAPTER 6

SAGA[1]

6.1 Introduction

In previous chapters I have given arguments for an evolutionary approach, and discussed the notion of adaptation in the context of cognition. Now I shall turn to theoretical discussion of the form of Genetic Algorithm appropriate for this domain. Since evolution implies the potential for indefinite increase in complexity over time, and since such increase in complexity of phenotype requires (given a fixed mapping from genotype to phenotype) comparable increase in genotype length, we must consider GAs with genotypes of varying length.

Given a long evolutionary run in which, in effect, the dimensionality of the current search space changes over time, a observation window focusing on just a small period of time may well allow analysis in terms of genotype lengths fixed at the current values; but such an analysis would inevitably miss the longer term changes of interest.

Holland's Schema Theorem has provided the theoretical underpinning for GAs (Holland 1975, Goldberg 1989b); this Schema Theorem assumes that all the genotypes in a population are the same length, and remain so through successive generations. In the messier world of natural evolution these assumptions do not hold, which prompts questions such as:

> Could some more generalized version of this theorem be extended to include variable length genotypes?
>
> Are there circumstances in which they might be of use in GAs?

In speaking of variable length genotypes I will be making some assumptions, spelt out later, about how those extra parts on long genotypes, not present on the shorter ones, contribute to their fitness. But the answers to these two questions will be, firstly: no,

[1] A version of this chapter has been published as (Harvey 1992b).

there is no such immediate generalisation, but rather a very different process is at work as genotypes change length, which must be analysed independently. And secondly: for traditional function optimisation problems they are unlikely to be of use, but they will be in Artificial Life and artificial evolution.

Manipulation of schemata in the conventional analysis of GAs can be interpreted in terms of intersections of hyperplanes in the predefined search-space — for instance in the case of binary genotypes of length l, the search-space is a hypercube of l dimensions. The schema corresponds to a hyperplane which may be associated with above average fitness; the GA then concentrates the search towards the intersections of these hyperplanes. With genotypes of potentially unbounded lengths, however, we no longer have a search space of known dimensionality. An alternative characterisation of a genotype search space, perhaps less familiar to the GA community, will be borrowed from theoretical biology; this lends itself more easily to variation in length of genotype.

It will be argued that for progress through such a space to be feasible, it only makes sense for genotypic variation in length to be relatively gentle. It follows that instead of attempting a generalisation of the Schema Theorem to genotypes of any length, the analysis of the convergence of a population of nearly uniform length can and should be decoupled from the analysis of changes in length. A general trend towards increase in length is associated with the evolution of a *species* rather than global search. The word *species* I am using to refer to a fit population of relative genotypic homogeneity.[2]

As to the question of under what circumstances variable lengths might be of use in GAs, it would seem that for such traditional GA concerns as function optimisation in a predefined domain, one would do best to stick to fixed-length genotypes. If, however, an animat is evolving in an environment with unknown complexity, then variability in genotype length becomes relevant; the use of classifier systems is one way in which this may, in effect, be achieved. A genotype space can be open-ended if the environment itself alters over time, perhaps in response to the evolution of the animat itself. The classic case is the *Red Queen* (Valen 1973, Stenseth and Smith 1984) (or *Arms Race*) phenomenon of co-evolution of different species interacting with each other, where one can expect over time both the phenotype complexity and the genotype length to increase.

The notion of a search space is a metaphor which is frequently a useful one. It is,

[2]It will follow from this that crosses between members of the same species have a good chance of being another fit member of the same species; whereas crosses between different species will almost certainly be unfit.

however, very often taken to imply a space of predefined extent, with a predefined or recognizable goal. In the natural world, tempting though it may be for any one species to think of evolution as a 4 billion year search for a goal of something very like them, it is evident that any such notion of a goal can only be *a posteriori*. So in order to distinguish the space of possibilities that a species can move in from that of a conventional search space, I shall use the term *SAGA space*[3]. This corresponds to the acronym for Species Adaptation Genetic Algorithms, the altered and extended version of GAs necessary to deal with such a space.

6.2 Variable lengths in GAs

Variable length genotypes have been used in GAs in, for instance, Messy GAs (Goldberg *et al.* 1990), LS-1 classifiers (Smith 1980), Koza's genetic programming (Koza 1990, Koza 1992b). The first of these in fact uses an underlying fixed-length representation. The analyses offered in the other two examples do not satisfactorily extend the notion of a schema such that schemata are preserved by the genetic operators.

Smith with his LS-1 system uses genotypes, of rule sets, of varying length. He suggests that for a hyperplane analysis an alternative 'attribute space' representation should be used. For one of the particular domains LS-1 works in, analysis would show, he suggests, that there are a finite number, k, of distinct attributes, relative to the task at hand. The search can be seen as one for combinations of these attributes yielding high performance, and hence as optimisation in a k-dimensional space, where each dimension represents the presence or absence of a particular attribute. A hyperplane analysis can then be applied to the structures represented as points in this k-dimensional space, provided that it can be shown that the genetic operators applied to the genotype tends both to explore new untested hyperplanes in the attribute space, and to exploit previously tested hyperplanes by generating new instances of them; but unfortunately this last condition need not be true in the general case.

In Koza's genetic programming (see Chapter 2) where recombination swaps complete sub-trees between the parents, if these sub-trees are of different size then the offspring will have genotypes of different lengths from their parents.

Koza suggests that the equivalent of a schema in the search space of such programs can

[3] "**Saga** ...story of heroic achievement or adventure; series of connected books giving the history of a family etc. [Old Norse = narrative]." Concise Oxford Dictionary.

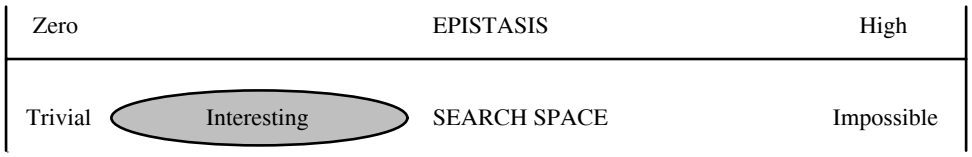

Figure 6.1: *Low, but non-zero, epistasis is associated with a search space that is possible, but non-trivial.*

be specified initially by any one specific sub-tree. Since the set of all potential programs containing that sub-tree is infinite, Koza finds it necessary to partition it into finite subsets indexed by the length of the program, and it is these subsets that are considered as schemata. The number of occurrences in the reproductive pool of examples of a particular schema which, as sampled in the parental pool, shows above-average fitness, will indeed tend to increase. But this does not cater for the fact that the crossover operator will in general turn the offspring into programs of different lengths, and hence disrupt the schema which has been defined (in part) by program length. A possible way to minimize this disruption would be to restrict the possible variations in length to only minimal changes, and indeed this will be echoed in the conclusions reached further on in this chapter.

The obvious way to extend the crossover operator from fixed-length to variable-length genotypes is by randomly choosing different crossover positions for each of the two parents; an offspring may then inherit two short portions, or two long portions, and in general will have a genotype of significantly different length. Potential drastic repercussions of this will be discussed in Chapter 9.

6.3 Epistasis

A gene is the unit of analysis in determining the phenotype, and hence its fitness, from the genotype; it is coded for by a small subsection of the genotype. The term epistasis refers to the linkage between genes on the genotype, such that the expression of one gene modifies or over-rules the expression of another gene.

If there is no epistasis, in other words if the fitness contribution of each element on the genotype is unaffected by the values of any of the others, then optimisation can be carried out independently on each element; simple hill-climbing is adequate. At the other end of the epistatic scale, where there are many dependencies between the elements, the only useful building blocks that a GA tries to manipulate are too long, and easily disrupted by genetic operators. Indeed in the limit of maximum epistasis only random search is feasible.

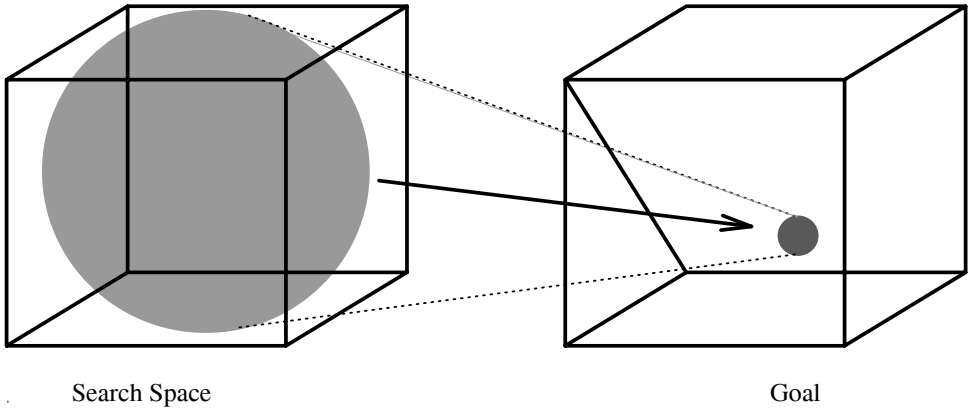

Figure 6.2: *The evolution of a standard GA in a fixed-dimensional search space; the population initially spans the whole space, and in the end focuses on the optimum.*

The appropriate region on the epistatic scale suitable for GA type search is between these two extremes, and GA representations need to be chosen with this in mind (Davidor 1990).

6.4 Uncorrelated Landscapes

A model of a genotype search space which allows explicit setting of low or high degrees of epistasis is based on the concept of a protein space, originally introduced in (Maynard Smith 1970). This space has a point for each possible example of a genotype, and a neighbourhood metric which gives all those other points which can be reached by a single mutation from a given point. Compared with the traditional GA analysis, which takes a global view of the whole search space, and considers how a population of points in this search space can effectively range across it by use of recombination, we now have a very different perspective. Here mutation is the only genetic operator, instead of just a background one to prevent irremediable loss of an allele.

Kauffman (Kauffman and Levin 1987, Kauffman 1989) has extended this model to produce a general theory of adaptive walks on rugged fitness landscapes — where the distribution of fitness values across the space is visualised as a landscape with fitness represented by the height. It should be noted that the fitness values are ascribed to points on a lattice rather than a continuum. Nevertheless the landscape can be imagined as a mountain range, where ruggedness implies a relative lack of correlation of heights of nearby points, which in turn is associated with high epistasis on the genotype.

Gillespie's assumptions (Gillespie 1984), that the mutation rate is slow compared to the assimilation of any fitter mutation by the population as a whole, are being used. The

population as a whole is considered to be at a single point in the space, with mutations of single members sampling the immediately neighbouring points. Any less fit mutations die out rapidly, whereas any fitter one causes, by this assumption, the whole population to move to that point.[4]

Where the genotype is a string of N elements, each able to take one of B possible values, then a single mutation involves taking any one element, and altering it to one of the $(B-1)$ values other than its present one; i.e. there are $N(B-1)$ one-step neighbours. If the fitnesses of these neighbours are completely uncorrelated with the fitness of the original point, i.e. if the fitness values of each point in the space is drawn at random from some fixed underlying distribution, then the landscape is maximally rugged.

The rate of progress of an adaptive walk via fitter variants through such an uncorrelated landscape can be analysed, assuming that this walk rejects any step that leads to a decrease in fitness, and chooses at random among those steps which increase fitness. This covers the posited case of a slow mutation rate, such that mutations occur one at a time, and the first fitter one encountered is chosen as the next step in the population walk. Let all the points in the space be ranked from worst, 1 to best, $T = B^N$, and suppose that the walk starts at the least fit point, rank 1. On average, the first step will be a move to a position half-way to the top in rank order, and successive steps will on average close the gap to the top by a half each time. Hence after R steps the expected relative rank order X/T is

$$\frac{X}{T} = 1 - \frac{1}{2^R}$$

In an adaptive walk on a maximally rugged landscape if the first step upwards, from the bottom rank of fitness, takes one unit of time, then the next step upwards, where only half the neighbours are fitter, takes on average 2 units, then 4, 8, ..., doubling each time (Kauffman and Levin 1987).

6.5 Correlated landscapes

The above discussion is for a completely uncorrelated landscape – - which can be considered equivalent to maximum epistasis between the genes on the genotype. In most fitness landscapes there is, however some local correlation, in that neighbours will tend to have

[4] This is a more restricted assumption than that in (Eigen and Schuster 1979), where a population under the influence of selection and a low mutation rate in general moves to form a *quasi-species*, with a probability distribution centred about a point.

similar fitness values, and certainly this is true of any search space in which GAs are to be of use. Let length be defined in this space using the distance metric of how many point mutations are necessary to move from one genotype to another. A long jump is defined to be the equivalent of several *simultaneous* mutations, long enough to jump beyond the correlation lengths in the landscape[5]. Moves via such long jumps will in general display important similarities with the characteristics of uncorrelated landscapes (except that in the limit of long jumps all points are accessible, and hence the notion of a local optimum becomes meaningless). In particular the above result still holds: that the waiting time until finding a fitter variant by such long jumps doubles after each such improvement.

Kauffman further considers a different assumption from that used above; suppose that instead of a single mutant being sampled at each unit of time, there is a large population of fixed size simultaneously sampling different mutants, and the population then moves as a whole to the fittest of any improved variant encountered. It is shown that the above result on waiting times remains almost unchanged.

This search process is of course very different from that analysed in conventional GAs, where a population of spread-out points effectively spans the search space, and recombination allows effective moves to predominate. The distinction between these two types of search process must be kept in mind when we turn to looking at variable length genotypes.

6.6 Variable length genotypes

Let us spell out some assumptions about a genetic system with variation in the length of genotypes, within which many different types of representation, or fixed mappings from genotype to phenotype to fitness, could be allowed[6]. These assumptions correspond to the NK model of Kauffman which is discussed later in this chapter.

> Firstly, it is assumed that the genotype can be analysed in terms of a number of small building blocks, or genes, that are coded for individually on it; possibly by a single symbol, or a sequence of symbols. These genes can be uniquely identified, either by their position by reference to an identified end of the genotype, as in conventional GAs; or by an attached tag or template, such as those used in messy GAs (Goldberg

[5]The 'correlation length' of any particular fitness landscape can be worked out (Kauffman 1989), such that in general two points in sequence space less than this (Hamming) distance apart have correlated fitness, and more than this distance apart do not.

[6]The possibility of a mapping from genotype to phenotype that changes over time is not considered here.

et al. 1990). Longer genotypes will code for genes that are not present at all on shorter ones.

Secondly, it is assumed that each gene makes a separate additive contribution to the fitness of the whole; but that the contribution of any one gene can be modified by epistatic interactions with a number K of the other genes. This number K is much less than the total number of genes available.

Thirdly it is assumed that the total of all these additive contributions is then normalized in some way such that the final fitness remains within some predefined bound even if the number of genes is unbounded; for instance, dividing by the number of genes. many genes there are.

The second assumption means that this gives a locally correlated landscape, with epistasis not too high. (If $K = 0$ there would be zero epistasis; if $K = N - 1$ where N is the current genotype length, maximum epistasis would give a completely uncorrelated landscape.) The last condition reflects the fact that any fitness function, as defined for a biological population, is only relevant in so far as it affects the selection process. On average in the long term each member of a viable population will be replaced by just one offspring. Less than one and the population is heading for extinction, more than one implies exponential growth. But there are always finite physical resource limitations which prevent such unlimited growth, and this has to be taken account of in the fitness function.

When genotype lengths are fixed, these three assumptions are compatible with the use of a standard GA. But now, in addition to the normal genetic operators of mutation and crossover, we assume that there are further operators, perhaps *cut* and *splice*, or *increase-length*, which allow offspring to have their length changed by arbitrary amounts, although still retaining at least some genetic material from their parent(s).

Suppose that there are a total of G different genes represented in the population, some perhaps represented in all genotypes and some in only a few. Then by adding an extra allele for each gene, to indicate whether it is 'absent' in a particular genotype, a new representation of the population can be formed in which every gene is represented in every member. Genetic operators which do not introduce a completely new gene into the population allow this to be analysed as a normal GA.

Suppose now that the genetic operators allow, by lengthening a genotype, the creation of a *single* new gene, giving a new total of $G+1$. By the second and third assumptions made

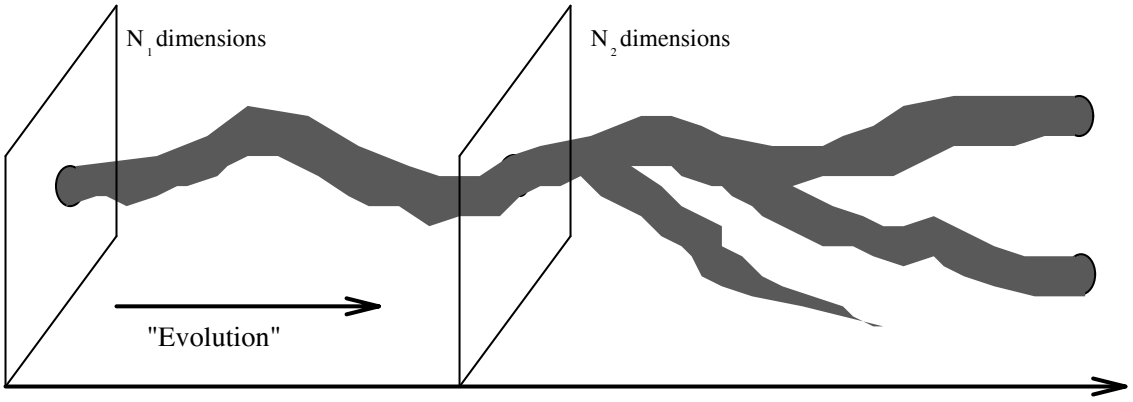

Figure 6.3: *The progress of the always compact course of a species; the horizontal axis indicates both time and the (loosely correlated) number of dimensions of the current search space. The other axes represent just two of the current number of dimensions. The possibility of splitting into separate species, and of extinction, are indicated in the sketch, although not here discussed.*

explicit above, the epistasis of this new gene is similar to that of the previous ones. The new population can now be considered as being spread across a new $(G+1)$-dimensional search space, except that all bar one member is confined within the previous G-dimensional sub-space. This can still be analysed as a normal GA with an initially skewed population. If a single advantageous new gene appears in the population, it can become widespread through crossovers.

In contrast to this, an alternative possibility is that the genetic operators allow the creation in one generation of a *large number* g of new genes on one genotype. In the new $(G+g)$-dimensional search space, the old population is based entirely inside the original (in relative terms, very small) G-dimensional subspace, with just the one new point exploring elsewhere. If g is large enough such that this is beyond the correlation length of the lanscape, then this is obviously a 'long jump' as defined in the previous section, and the fitness will be uncorrelated with that of any of the previous generation. If such a long jump is successful, in the sense that the new genes are retained in the population, with a resulting general increase in the fitness of the population, then the chances of a successful further long jump will be significantly less. Any such long jump adaptation will suffer from the problem of the doubling of waiting time after each jump.

The picture now emerges of two very different processes going on at independent timescales in this SAGA space. Given a genetic operator which allows unrestricted changes in length of genotypes, we can expect the following sequence of events in a locally correlated landscape:

> An early population could fluctuate in length through 'long jump' adaptation which effectively acts in an uncorrelated landscape; but as average fitness increases the doubling of waiting times will slow down this process drastically.
>
> Thereupon the traditional GA operators of crossover and mutation will take over, and Holland's Schema Theorem will be applicable to this phase of the search.
>
> Those applications of the change-length operator which result in minimal changes of length will be moves on a correlated landscape, and therefore are feasible even if major changes are increasingly unlikely.
>
> If there are selectionary pressures which encourage the genotype lengths to increase, the population will become a nearly-converged 'species', with an almost uniform length that increase in small steps.[7]

6.7 An objection

One immediate potential objection that needs to be dealt with is that of gene duplication[8]. Surely, it is argued, in the natural world there are many examples of polyploidy, where the duplication of a complete chromosome or significant part of the whole genotype results in an organism which phenotypically is almost identical to another organism without this duplication. This is a counter-example to the suggestion above that only small changes in genotype length can be viable.

This objection is indeed valid, but analysis of why such gene-duplication in the natural world leads to little immediate phenotypic variation is instructive. The duplicated genetic material does not result in any different proteins being coded for at the cellular level, but rather, under the same triggering conditions, more of the same. The DNA of the natural

[7] These ideas should be neutral in respect of the punctuated equilibria controversy. A succession of small steps may or may not be rapid in geological time — indeed there may well be good reasons why there should on occasion be such a cascade. What is being ruled out here is any single large step.

[8] A simulation which produces interesting results using gene-duplication is (Lindgren 1991); discussed later in Section 6.9.

genotype is not a string of instructions that is read off in sequence — as has till now been the practice in genetic algorithms — but rather a major constraint (not necessarily the only one) on the process of morphogenesis. Particular polypeptide chains present in the cell initiate, through template-matching with appropriate parts of the DNA, the transcription and translation of the thus-indicated parts of the genotype so as to make further polypeptide chains. Hence a doubling of the relevant parts of the genotype merely doubles the potential resources for enabling the same sort of chain-reaction as before. The virtues of this sort of mechanism, and its possible incorporation into artificial evolution, will be advocated in Chapter 10.

If any such genotype-to-phenotype system is used, then the principle that only gentle changes in genotype length can be in practice viable must be altered to meet this objection. One possible approach towards doing this would be to invoke the information content of the genotype as measured by the Lempel-Ziv metric which can be calculated by the Kasper-Schuster algorithm (Kasper and Schuster 1987, Lopez and Caulfield 1991). This metric for bit-strings is based on the Kolmogorov definition of complexity, calculating the minimal length of a computer program which can generate the bit-string (or in the current context, genotype).

To be precise, Lempel and Ziv do not calculate the length of a minimal program — indeed a general algorithm to do this cannot be given — but a number $c(n)$ which is a useful measure of this length, i.e. an approximate measure of Kolmogorov complexity. The time complexity of $c(n)$ lies between n (for a string of identical digits) and a worst case of n^2 (for a string of random digits); hence it is a practical algorithm to use. It can be characterised by considering a simple self-limiting learning machine which scans digits from left to right, adding a new word to its memory every time it discovers a substring of consecutive digits not previously encountered. A flow diagram for the algorithm is shown in Figure 6.4.

The initial digit will of course be an insertion. Thereafter at each stage of extending the generation of the string, one calculates the maximum extension that would consist of a copy of a substring of what has already been seen to date, plus the insertion of a single fresh digit on the end to differentiate it; adding this extension completes this stage, whereupon the process is repeated until the end is reached. The measure $c(n)$ is the number of such insertions (plus one if the very last extension copy does not have a fresh digit added to the end). Or as an alternative way of looking at it, $c(n)$ is counted as one

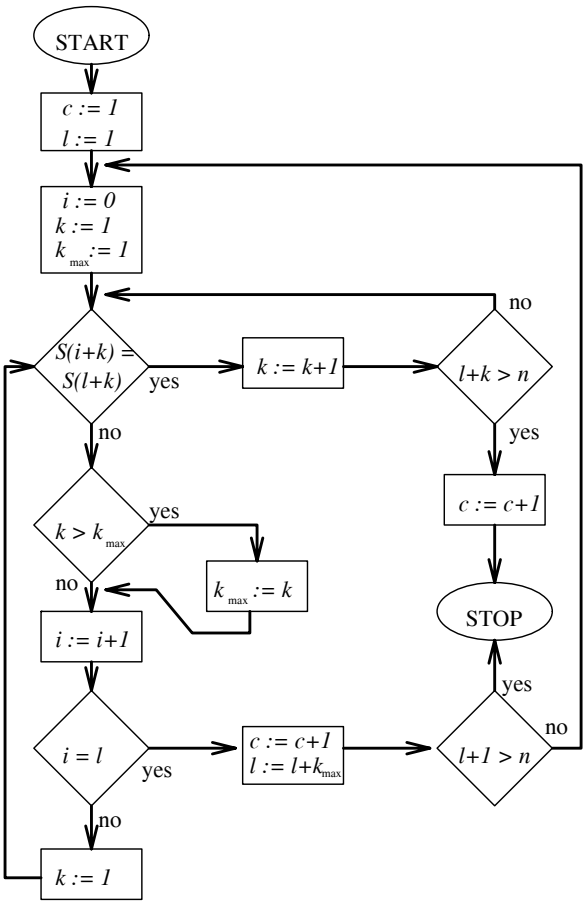

Figure 6.4: *A flow diagram for the Lempel and Ziv algorithm. This diagram is derived from (Kasper and Schuster 1987), except for a* yes/no *transposition on the upper-right decision box, correcting what appears to be an error in the original publication.*
The complexity count is c, length of string S is n, where S(i) is the ith member of the string. k and k_{max} are used to keep track of temporary length, and maximum length to date, of substrings being checked to see if they have occurred previously.

for the first digit, plus one for every stage as sketched above that is gone through. The string 000000 is decomposed into $0 \cdot 00000$ with a complexity score of 2; whereas the string 0001101001000101 is decomposed into the substrings $0 \cdot 001 \cdot 10 \cdot 100 \cdot 1000 \cdot 101$, giving a complexity score of 6.

With such an easily calculable measure of complexity, which 'devalues' simple gene-duplication in an intuitively attractive way, one could recast the principle of gradual change of genotype length into the form that any increase in this Lempel-Ziv quantity can only be gentle for viability to remain feasible.

However, one should remember that the addition of large quantities of 'junk DNA'

would increase such a measure by a large amount in one step, yet if not expressed in the phenotype would not alter the viability of the same. Hence it would seem to be necessary to discard those parts of the genotype which are not expressed in any fashion in the phenotype, before applying this metric.

Such provisos start to make the practical application of such a metric seem less than trivial. So, for the time being, perhaps one should stick with the original principle that changes in genotype length should be gradual, and simply add the qualification that, where the sort of developmental process outlined above takes place, duplications of genetic material are exempted from this principle. There is a strong case to be argued, in Chapter 10, that genetic duplication, followed by mutation of one of the copies, is potentially a powerful method for evolutionary progress.

6.8 SAGA and the Schema Theorem

A schema defines a subset of possible genotypes which share the same values at a specified number of genes. If there is no upper limit to the possible length of the genotype, these subsets will be infinite in size, and estimates of the 'average fitness' of a schema based on any finite sample become problematical.

We might be tempted to avoid this by saying, in this particular example we will restrict the space of possibilities to genotypes of length less than, e.g., 1000. But then 'nearly all' possible instances of a particular schema will refer to genotypes very close to this upper limit, and there may be no reason to expect the average fitness of this schema to bear any significant relationship to the fitness of the same schema restricted to genotypes of maximum length 100, 500 or even 950.

However, consider the case where all the population have the same G gene sites (though with variations in the values at each gene site); and we are considering the addition of one extra gene to one or more of the population. We can recast our analysis in terms of a population all of genotype length $G + 1$, with the extra gene having one additional possible value of 'absent'. For any two schemata S_1 and S_2 that have the extra gene fixed as 'absent', let S_1' and S_2' be the corresponding schemata with the extra gene value allowed any value (including 'absent'). Given the assumption of low epistasis, the relative fitnesses of schemata S_1 and S_2 will be closely correlated with the relative fitnesses of schemata S_1' and S_2'. This will still hold true if we allow an extra g genes rather than one, provided that g is small in relation to G and the assumption of low epistasis holds. It will not hold

true when g is large, or epistasis is high.

Hence in the short term of small changes in genotype length in a population of nearly uniform genotype length, we can still apply the Schema Theorem.

6.9 Would variable lengths be useful?

Turning now to the second question posed in the introduction to this chapter, under what circumstances might it be useful to have a genetic operator which allows an increase in the number of genes represented on the genotype? If the problem being tackled is basically a function optimisation one, where there is a predefined search space with a fixed number of factors that can be coded for on the genotype, then it would be folly not to put them all in at the start, represented in such a way as to minimise the epistasis, and put one's trust in the Schema Theorem.

A non-traditional GA, such as Koza's genetic programming (Koza 1990, Koza 1992b), is more interesting. The discussion above of his analysis of sub-trees indexed by the length of program as being equivalent to schemata pointed out the flaw in that the crossover operator in general changes the program length. If this operator was restricted to allow only minimal changes in program length – - which was suggested in the previous section to be the only effective long-term method of increase — then this gap might well be plugged. A super-schema $S(T, P, P+p)$ can be defined in terms of neighbouring schemata specified by the same sub-tree T and indexed by program lengths P to $(P+p)$. The recombination operator would have a high probability of not disrupting such a super-schema if only minimal changes in length were allowed, as this would minimise the net 'leakage' caused by offspring programs which had sub-tree S but whose lengths fell outside the range P to $(P+p)$.

A major group of problems which cannot be specified in terms of a predefined search space involve co-evolution of one population with another (or several) which in turn is affected by the first. Since one population is part of the environment for the other, the environment is continually changing (Hillis 1991, Husbands and Mill 1991). The same requirements, of relatively few epistatic interactions between a gene and those aspects of the environment which it affects and is affected by, hold if an evolutionary process is going to be more than random search.

There are many co-evolutionary worlds where an increase in complexity in one population stimulates an increase in complexity of the other, and so on, perhaps indefinitely.

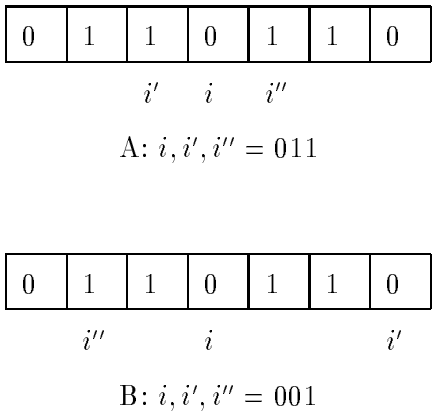

Figure 6.5: *Genotypes in an NK model. At the top, gene i is linked to neighbours i', i''. The values 011 point into a fitness look-up table for i. Below, i' and i'' are no longer immediate neighbours.*

So in as much as length of genotype is associated with complexity of the phenotype, we can expect that there is selective pressure for long-term growth in their lengths. Lindgren (Lindgren 1990, Lindgren 1991) models a population of individuals competing with each other at the iterated Prisoner's Dilemma with noise — the population in practice breaks into sub-populations with different strategies. There is no recombination, the only genetic operators being mutation and gene doubling. The particular representation used treats a binary genotype of length 2^h as a look-up table; the history of the last h interactions between competing prisoners, coded in 0's and 1's and considered as a binary number, generates a pointer into this look-up table to determine the strategy. Application of the gene-doubling operator does not in itself generate new strategies, but allows later mutations to generate finer discriminations within that strategy. Using the Lempel-Ziv metric, the complexity only increases by 1 during doubling. His results show periods of stasis alternated by periods of unstable dynamics, with a long-term growth in the lengths of the successful sub-populations.

6.10 NK Simulation

The NK model (Kauffman and Levin 1987, Kauffman 1989) assumes a binary genotype of length N, where each position represents a gene which is affected by linkage with K others. This is an abstract model in which it is assumed that the fitness of the phenotype can be directly calculated from the values on the genotype.

The three assumptions itemised above in Section 6.6 hold for the fitness function. In

	000	0.141
value of B	001	0.592
	010	0.653
value of A	011	0.589
	100	0.793
	101	0.233
	110	0.842
	111	0.916

Figure 6.6: *Fitness table for gene i in the NK model, filled with random numbers between 0 and 1. i, i' and i'' determine fitness contribution of gene i to fitness of the whole genotype. A and B refer to the genes in the previous figure.*

the case of $K = 2$, the fitness contribution of gene i depends on the two others to which it is linked (which may be specified as immediate neighbours, or may be specified at random positions). The binary alleles of i and its 2 neighbours specify a 3-bit number which picks a fitness from an 8-place table of fitnesses associated with gene i — there are N such fitness tables prepared at the start of the simulation, with each place containing a fitness randomly chosen in the range 0 to 1 (see Figures 6.5 and 6.6). The fitness of the genotype is then assessed by adding up the fitnesses thus determined for all N genes, and dividing by N. It can be seen that in this case of $K = 2$, the flipping of a single bit on the genotype will affect the fitness contributions of on average just 3 genes; the other $N - 3$ being unaffected, this gives a reasonably correlated fitness landscape. In the limit of $K = N - 1$, where the fitness table associated with each gene would have 2^N places, the flipping of a single bit would alter everything, and the fitness landscape is totally uncorrelated.

This model can be extended to allow for changes in genotype length. The simplifying assumption is made that any new gene appears at the right-hand end of the genotype, and that the identity of the gene is uniquely determined by its position in the genotype. In the case of $K = 2$, if linkage is with immediate neighbours to left and right, the ends are assumed linked in a loop to avoid boundary problems. A set of tables of random fitness values for each gene is set up for the minimal-length genotype. For each new gene added one new table is generated for it, and two further replacement tables for those genes which are neighbours of the newcomer. This can easily be generalised for $K > 2$, and also for choice of epistatic linkages to newcomers being randomly selected rather than restricted

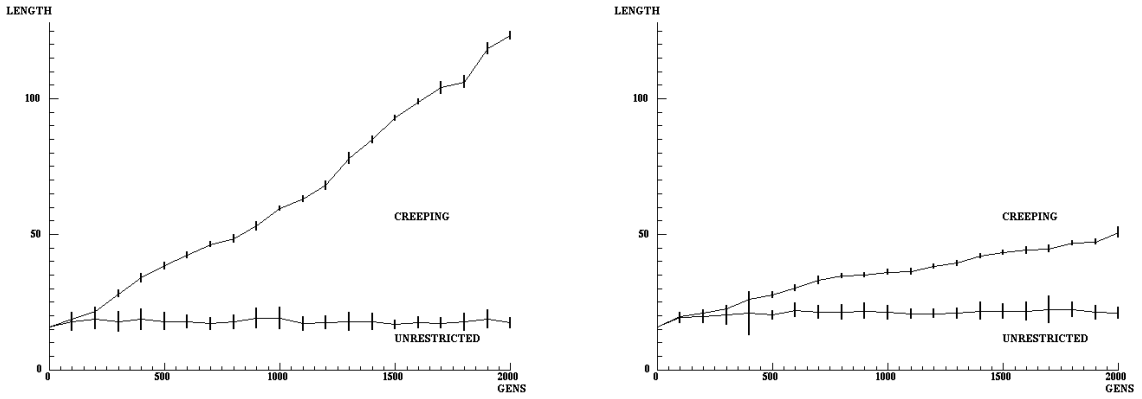

Figure 6.7: *Average genotype lengths against generations; vertical bars show standard deviations. Effects of 'creeping' and 'unrestricted' increase-length genetic operators on a population with the same fitness conditions, epistasis $K = 2$. Left graph, linkage with neighbouring genes. Right graph, random linkage.*

to neighbours.

This allows the setting up of models for simulations with genotypes of any length, with epistasis of any degree; increase in the lengths of genotype can be allowed under conditions to be specified. Such simulations allow experimentation with variable lengths, in an abstract context, without the difficulties of choice of representation that normal problems give.

Experiments have been run with a population of 100 genotypes, all of initial length $N_{init} = 16$, with epistasis given by $K = 2$, linkages being with neighbouring genes on the genotype; there was an additional genetic operator which allowed the lengths to increase. The purpose of these experiments was to test whether big increases or small increases in length in the short term had differing effects on the way genotype lengths increased over the long term. For this purpose it was necessary to tweak the fitness function on which selection operated, so as to favour long-term increase in genotype length. Without such tweaking, since the NK-fitness is conventionally normalised to fall within the range 0.0 to 1.0 regardless of length N, there would be no incentive for long-term increase in N.

If the normalisation was eliminated, however, so that the fitness of a genotype composed of the sum of N fitness contributions lay within the range 0 to N, then large increases in N would almost inevitably lead to large increases of fitness, regardless of the gene values. This other extreme would be equally undesirable for the purposes of this experiment, hence the necessity for tweaking to provide selective pressure for long-term increase in genotype

length, but *not* regardless of gene values. After some failures a method was found which produced the desired effect by taking the initial NK fitness of any genotype, as defined by look-up tables of random numbers and normalised for length; then adding a factor proportional to $N/(N + \overline{N})$, where N is the length of the particular genotype and $\overline{N}$ is the current average genotype-length of the population. The constant of proportionality was chosen such that there was a selectionary pressure in favour of longer genotypes comparable to the selectionary pressures given by the initial fitnesses. It was found that a factor of r=0.2 worked for the genotype lengths considered here, and hence if f_0 was the normalised NK fitness, an adjusted fitness f_1 was calculated by:

$$f_1 = (1.0 - r)f_0 + r\frac{N}{(N + \overline{N})}$$

After this adjustment fitness-proportionate selection or rank-based selection could be used. The choice used here was to further rescale the fitnesses so that the fittest would have an expected 2.0 offspring against an average of 1.0, and then to use fitness-proportionate selection.

In the first trial the genetic operators were crossover, mutation, and an increase-length operator which in 10% of offspring allowed a genotype to increase in length by a random amount between 0 and 50% of its current length. In the second trial the 'creeping' increase-length operator only allowed an increase of genotype length in the offspring by exactly one gene. Despite this restriction, the average length increased steadily in the 'creeping' trial as compared to virtually no increase in the 'unrestricted' trial. Figure 6.7 shows results from four runs: with and without restrictions on increase in length, firstly with linkage to neighbouring genes and secondly with linkage to randomly placed genes. The population size was 100; running the same experiments 10 times using different random number seeds produced qualitatively similar results.

On separate trials with the epistatic linkages being with randomly placed genes instead of with neighbouring genes, the results were similar, although with the 'creeping' increase in length at a slower rate. In both sets of trials there was much more variation in lengths within the population in the 'unrestricted' case, compared to the 'creeping'.

Both the theoretical arguments leading one to expect this, and the experimental conditions used above, rely on assumptions listed in Section 6.6. Undoubtedly one could create special-purpose fitness landscapes in which these assumptions, and these results, do not hold. Nevertheless I take it to be a useful working hypothesis that many real world

problems conform sufficiently to the assumptions for these conclusions to be valid.

6.11 SAGA and Development

By working with the evolution of a nearly converged species increasing in genotype length and in phenotype complexity over time, we have moved away from the usual GA notion of evolution as a search technique towards a notion of 'evolution as a tinkerer' (Jacob 1989), always adding to or altering something that is already viable.

The cumulative process of additions and alterations implies that a phenotype can be considered as being produced from a genotype by a developmental process (see Chapter 10). It will not be surprising if 'Ontogeny recapitulates Phylogeny', subject to the small but ever-present possibility of a later alteration bearing on a significantly earlier stage in the developmental process. The application of this approach to, for instance, the evolution of subsumption architectures for robots (Brooks 1991) would seem to correspond to the effective, tinkering, incremental approach that practical designers take.

One consequence will be that a species will only reach those parts of a SAGA space that are connected by a continuous chain of viable ancestors to the origin. Thus within the space of all possible genotypes of length G there may well be a host of fit and viable points or islands which, through isolation and lack of a viable pathway from the origin, are unattainable.

6.12 SAGA and Genetic Operators

In addition to the usual GA operators of mutation and/or crossover, an operator which allows change in genotype length is necessary. The example in the NK model simulation above is the most trivial such operator, and depends on the identity of any gene being given by its position relative to one end of the genotype. Lindgren's (Lindgren 1991) doubling operator uses a representation which has this same dependency on position.

If the identity of a gene is given by a tag, or by template-matching as seems to happen in the real world of DNA, then absolute positions of genes on the genotype need not be maintained. This allows for duplication of a section of the genotype, after which mutations can differentiate the duplicated parts. The crossover operator can still be used in a fairly homogeneous population with slight variations in genotype length, although given any random crossover point in one parent, a 'sensible' corresponding crossover point in the other parent must be chosen. This can be uniquely defined as that point (or in some cases,

any of a contiguous group of points) which maximises the longest common subsequences on both sides of the crossover. A version of the Needleman and Wunsch algorithm makes this computationally feasible (Needleman and Wunsch 1970, Sankoff 1972). This will be set out in full in Chapter 9.

6.13 Conclusions

With fixed-length genotypes one can afford to think in terms of a fixed, predefined search space with a finite number of dimensions which, even if it is immense, is at least theoretically knowable by God or Laplace.

When one allows genotypes to vary in length the search space is potentially infinite and it stops making sense to think of it as defined in that sense. Nevertheless, in the real world, evolution has taken place in such a fashion that we have very distant ancestors whose genotypes were much shorter than ours; the problems we face are not the problems they faced.

When looking at evolution, talking about 'problems being solved' can be very misleading. However, people using GAs are usually hoping to use lessons from evolution in order to find solutions to a problem that faces them. If they really do know the problem they have to solve, then they can define in finite terms the search space, and fixed length genotypes are appropriate. If, however, they are trying to evolve a structure with arbitrary and potentially unrestricted capabilities, then the problem space is not predefined, genotypes must be unrestricted in length, and a new approach is needed. Hence this discussion is probably more relevant to those looking at the evolution of animats or cognitive structures than it is to those looking at GAs as function optimisers.

One of the lessons demonstrated, given the assumptions of Section 6.6, is that if genotypes can potentially increase indefinitely, they will in practice only do so on a slow timescale, so that within a population all genotypes will be very nearly the same length. Indeed, there will be a high degree of uniformity in the genotypes[9], and any significant variations, including changes in length, will spread through the whole population before the next variation occurs. This is in contrast to the relatively fast timescale on which the crossover operator, which is the power-house of standard GAs, very efficiently mixes and matches fitter schemata.

One factor to bear in mind here is that there is a relationship between mutation rate

[9] ...modulo possible gene duplications, as discussed above.

and the length of a genotype that can effectively evolve. Too little mutation, and there is not the variation to allow change; too much, and there is not sufficient stability to maintain fitness. In contrast to the approach used in Holland's Schema Theorem, or the hyperplane analysis of schemata, where the population can effectively sample the whole search space, we must visualise a population in our new, infinitely though slowly expandable, search space as a localised cloud (with a high degree of consistency within the population) which can only sample 'nearby' points (those that can be moved to by one or a small number of applications of the genetic operators.) The question ceases being 'Where in this whole search space is the optimum?' and becomes instead 'From here, where can we move to that is better?'.

CHAPTER 7

Selection and Mutation[1]

7.1 Introduction

In the context of species evolution the metaphor of hill-crawling as opposed to hill-climbing will be introduced in this chapter, and appropriate mutation rates discussed. On both pragmatic and theoretical grounds, it will be suggested that there are good reasons for using Tournament Selection in evolutionary robotics.

Whereas in standard GAs the crossover operator is the main engine driving the search, from an initial population widely spread across the search space towards homing in on an optimal area, in SAGA the main operators operating on a converged population are mutation and change-length operators. The mutation operator, rather than being at a very low-level background rate, will be shown to be of the order of magnitude of 1 mutation per genotype. The crossover operator is used to 'mix-and-match' any improvements made by the other operators, and must be carefully crafted to minimise disruption.

In general, the 'problem-solving', or 'goal-seeking' metaphor for evolution is misleading. Within a SAGA space, however, it can still be useful to use this metaphor in the restricted sense of searching around the current focus of a species for neighbouring regions which are fitter, or in the case of neutral drift, not less fit. Such a search takes place through application of genetic operators such as mutation or change-length.

As discussed in Chapter 6, a change-length genetic operator, when subject to the restrictions of only allowing small changes in length in any single application, can be translated into a form equivalent to mutation by sleight-of-hand. In the case of an increase by one gene in a binary genotype with possible alleles 0 and 1, this gene can be considered instead to have 3 possible values 0, 1, and $-$, where the latter value is equivalent to 'absent'; the new appearance of this gene can be considered as a mutation from $-$ to either 0 or 1. If more than one gene appears in one reproduction event, then this would

[1] The work here was presented at Artificial Life 3, and has been published as (Harvey 1993a).

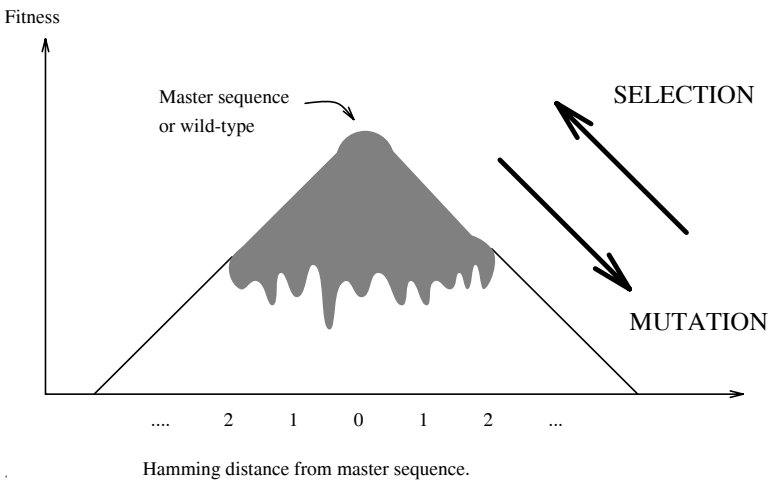

Figure 7.1: *The opposing forces of mutation and selection on a population centred around a local optimum, where Hamming distance from master sequence is directly related to fitness ranking. The vertical axis denotes fitness, and the many dimensions of genotype space are figuratively collapsed onto the horizontal plane; thus the 'mountain' sketched as 3-dimensional is more accurately multi-dimensional. Since Hamming distances are integers only, the sketch should not be taken too literally.*

be equivalent to a simultaneous set of mutations, which for low mutation rates is highly unlikely.

For this reason in the next sections of this chapter we will consider mutation only, though bearing in mind that through this sleight-of-hand it will be possible to extend some of the conclusions to an increase-length operator which works at very low rates. In addition we will start by assuming that there is no recombination; i.e. reproduction is asexual. Having sketched out the important factors using this assumption, the impact of recombination on this sketch will be discussed.

7.2 Species hill-crawling

Usually in GAs premature convergence of a population is something to be avoided. In SAGA, we are continually working with a converged population, and are interested in encouraging search around the local focus while being careful not to lose the gains that were made in achieving the current *status quo*. In the absence of any mutations, selection will concentrate the population at the current best. The smallest amount of mutation will hill-climb this current best to a local optimum. As mutation rates increase, the population will spread out around this local optimum, searching the neighbourhood, but if mutation

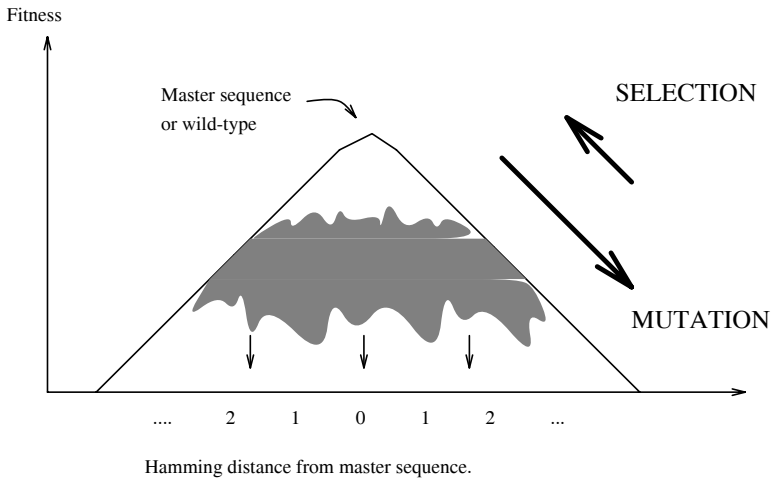

Figure 7.2: *When mutation outweighs selection so that the fittest rank can be lost, Muller's ratchet inexorably drives the population down the hill. See comments on Fig. 7.1.*

rates become too high then the population will disperse completely, and the search will become random with the previous hill-top lost.

The problem is that of *Muller's ratchet* (Maynard Smith 1978). Call the genotype that represents the very peak of the hill the master sequence (or the 'wild-type'). As a converged population, the other members will be quite close in Hamming distance to this master sequence, and hence far more mutations will increase this Hamming distance than will decrease it. The only force opposed to this pressure is that of selection preferentially reproducing the master sequence and its nearest neighbours in sufficiently large numbers to allow an occasional copy of the master with fortuitously no mutation; the other possibility, of fortuitous back-mutation from a near neighbour to the master sequence, is so small as to be usually negligible.

Figure 7.1 sketches the effects of mutation on a population centred around a local optimum. The vertical axis represents fitness or selective values. The horizontal axis indicates distance in sequence space from the master sequence at the top of the hill. The shape of the hill indicates the assumption that within this particular local neighbourhood increase in numbers of mutations is monotonically related to decrease in fitness. Figure 7.2 demonstrates the effect of Muller's ratchet when mutation is high enough to cause loss of information. Figure 7.3 sketches the effects when mutation is high enough (without bringing Muller's ratchet into play) for some elements of the population to crawl down the hill far enough to reach a ridge of high selective values. As discussed in (Eigen *et al.* 1988), this results under selection in a significant proportion of the population working their way

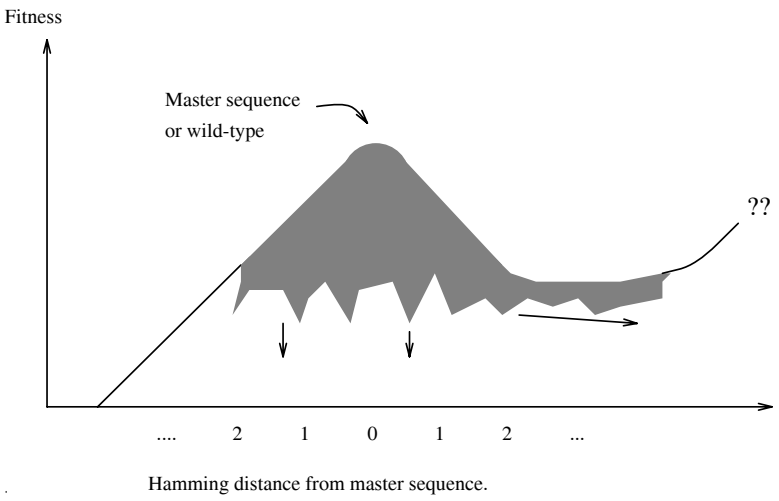

Figure 7.3: *If the population can crawl down the hill far enough to reach a ridge of relatively high fitness, it will spread along it, potentially reaching new hills. See comments on Fig. 7.1.*

along this ridge, and making possible the reaching of outliers further in Hamming-distance in that particular direction from the master sequence.

If any such outliers reach a second hill that climbs away from the ridge, then parts of the population can climb this hill. Depending on the difference in fitness and the spread of the population, it will either move *en masse* to the new hill as a better local optimum, or share itself across both of them.

So in a SAGA setup of evolution of a converged species, we want to encourage through the genetic operators such hill-crawling down towards ridges to new hills, subject to the constraint that we do not want to lose track of the current hill. To quote from (Eigen *et al.* 1988), pages 6886–6887:

> In conventional natural selection theory, advantageous mutations drove the evolutionary process. The neutral theory introduced selectively neutral mutants, in addition to the advantageous ones, which contribute to evolution through random drift. The concept of quasi-species shows that much weight is attributed to those slightly deleterious mutants that are situated along high ridges in the value landscape. They guide populations toward the peaks of high selective values.

7.3 SAGA and mutation rates

Although progress of a species through a fitness landscape is not discussed in the standard GA literature, in theoretical biology there is relevant work in the related field of molecular quasi-species (Eigen and Schuster 1979, Eigen *et al.* 1988). In particular, analysis of

'the error catastrophe' shows that, subject to certain conditions, there is a maximum rate of mutation which allows a quasi-species of molecules to stay localised around its current optimum. This critical maximum rate balances selective forces tending to increase numbers of the fittest members of the population against the forces of mutation which tend, more often than not, to drag offspring down in fitness away from any local optimum. But a zero mutation rate allows for no further local search beyond the current species, and other things being equal increased mutation rates will increase the rate of evolution. Hence if mutation rates can be adjusted, it would be a good idea to use a rate close to but less than any critical rate which causes the species to fall apart. A further possibility, in the spirit of repeated annealing of metals, is to temporarily allow the rate to go *slightly above* the critical rate — to allow exploration — and then cut it back again to consolidate any gains thus made. In the context of artificial evolution, where mutation rates and rates of selection can be under explicit control, this can be arranged relatively easily.

For an infinite population, in the particular context of molecular evolution, Eigen and Schuster show (Eigen and Schuster 1979) that these forces just balance for a mutation rate

$$m = \frac{\ln \sigma}{l}$$

where l is the genotype length and σ is the *superiority* parameter of the master sequence — the factor by which selection of the master sequence exceeds the average selection of the rest of the population. The diagrams they show for the very sharp cutoff at the critical rate refer to a fitness landscape with a single 'needle' peak for the master sequence, all the rest of the population taken to be equally (un-)fit; where the hill slopes more gently from the master sequence, the cutoff is less abrupt (as will be shown in Figure 7.5). For typical values of σ between 2 and 20, the upper limit of mutation before a quasi-species 'loses its grip' on the current hill would be between $0.7/l$ and $3/l$. When the population is of finite large size, yet small enough for stochastic effects of genetic drift to start having an effect, the same overall picture holds except for a reduction in this critical mutation rate (the 'error threshold')(Nowak and Schuster 1989). Expressed in terms of the single-digit accuracy of replication $q = 1 - m$, then the critical value of q for a population of size N is related to that for an infinite population by

$$q_N = q_\infty \left(1 + \frac{2\sqrt{\sigma - 1}}{l\sqrt{N}} + \ldots\right)$$

Nowak and Schuster suggest that the approximations made in deriving this equation mean that it should only be relied on for large populations of significantly more than 100.

Nevertheless, the presence of l, genotype length, in the denominator suggests that for genotypes of length order 100 or larger, and populations of size order 100, the error threshold will be extremely close to that for an infinite population.

Eigen and Schuster show that for a given selective pressure, the maximum length of genotype that can be reliably held in a tight distribution at an optimum is of the order of magnitude of the reciprocal of the mutation rate. Mutation rates of 5×10^{-4} per base by single-stranded RNA replication is adequate for a phage with length 4500. The lower mutation rates of order 10^{-9} associated with DNA replication and recombination in eukaryotes allow for the genotypes of length order 10^9 that humans have.

Since it is the natural logarithm of the *superiority* parameter σ which enters into the equation for m, variations in this of an order of magnitude do not affect the error threshold as significantly as variations in genotype length. In conventional GAs, choice of mutation rates tends to be a low figure, typically 0.01 or 0.001 per bit as a background operator, decided upon without regard to the genotype length. This despite suggestions from experimentation in (Schaffer *et al.* 1989) that optimal rates $m_{opt} = \alpha/(N^{0.9318} l^{0.4535})$, for some constant α; in (Hesser and Männer 1991) that after earlier higher values should decrease exponentially towards $m_{opt} = \alpha'/(N\sqrt{l})$; and in (De Jong 1975) quoted in (Hesser and Männer 1991) as recommending $m_{opt} = 1/l$.

In support of this analysis, practical results given by Beer and Gallagher (1992) in their work discussed in Chapter 2, show that on testing different mutation rates of roughly 0.02, 0.8 and 8 mutations per genetic string, the value 0.8 gave best results.

But these rates, and also the error threshold given by Eigen and Schuster, are based on particular assumptions about the selective forces on the population. Tournament selection will be focused upon later in this chapter, and this provides a significantly different range of selective forces to a population, which means the above analyses cannot be relied on.

In Figure 7.4, an artificial fitness space was created with two fitness hills. The fitnesses of any point in the space was given by the (negative of) the minimum of the Hamming distances to each hill. The population was started entirely concentrated at the peak of one hill, and then allowed to drift under various mutation rates, with tournament selection, until the other hill was first arrived at by a single member. Obviously with zero mutation it would never be found; with excessively large mutation, random search would result, also taking excessively long times. The typical U-shaped curve, of time taken against mutation rate, is shown in Figure 7.5. In this case the difference between with and without

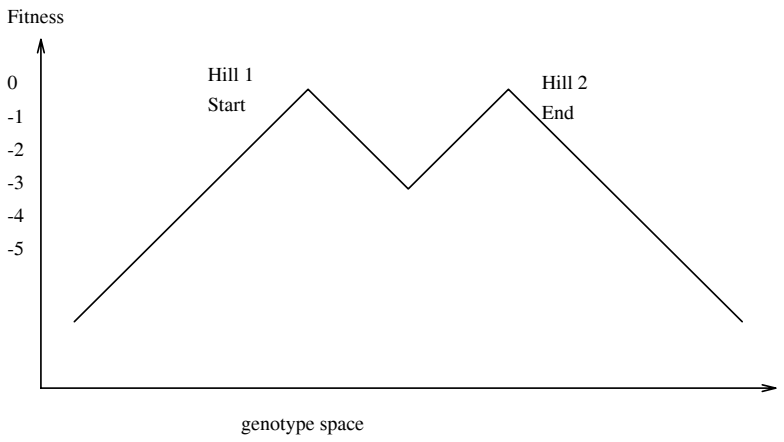

Figure 7.4: *Two genotypes are designated as start and goal, each of them with maximum fitness (of 0). The fitness of any other genotype is defined as (the negative of) the minimum of the Hamming distances to start and to goal. This gives a fitness landscape of two hills as shown. The population all starts off at the peak of the 'start hill'.*

recombination was only marginally in favour of the former.

Recently in the GA community there has been some discussion of the surprising success (in some circumstances) of what has come to be called *Naive Evolution*; i.e., mutation only, contrary to normal GA folk-lore which emphasises the significance of crossover. It would be interesting to check on those circumstances where it has been found useful, and see whether the population is in fact converged, with hill-crawling being the motive force for progress. The optimal mutation rates that are appropriate when hill-crawling is feasible have been obscured in the GA literature by the usual practice of quoting mutation rates per bit or per symbol, rather than per genotype. It is the optimal mutation rates per genotype that can be found within a band that is nearly invariant over all genotype lengths.

7.4 Tournament Selection for practical reasons

In the context of evolutionary robotics, realistic simulations take time to run, and it will be necessary to do a large number in parallel. The problems of simulation have been already discussed, and where they can practically be used they will in practice be complex; parallel machines with SIMD are no use, and for instance an individual workstation per simulation would be appropriate. In almost all networks of workstations there is a vast unused computational capacity which can be used effectively by running background processes. It then becomes attractive to use an evolutionary algorithm which allows asynchronous processing of the individuals.

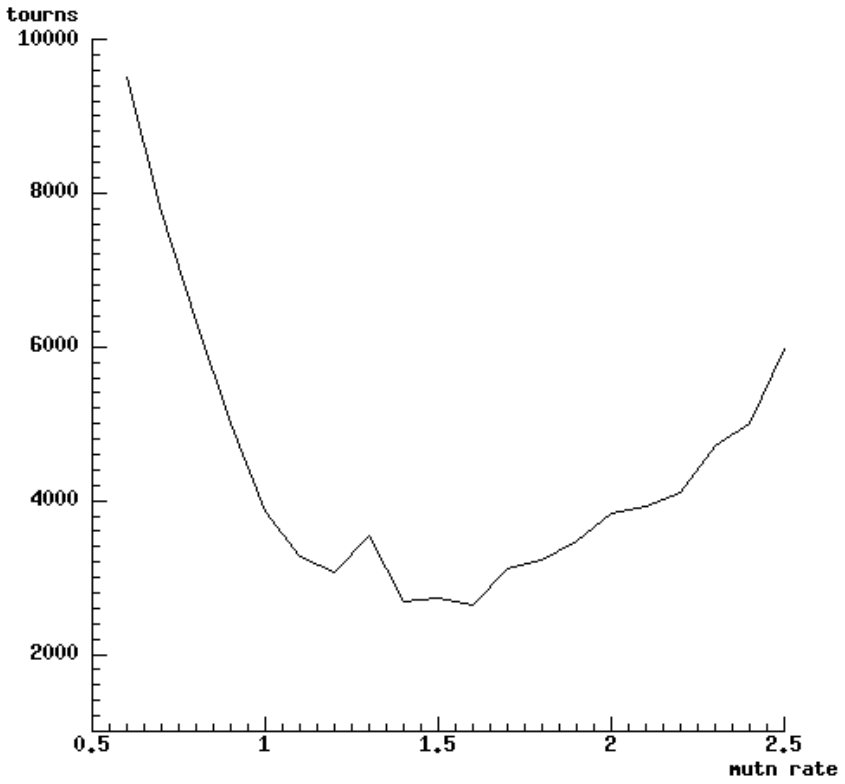

Figure 7.5: *Number of tournaments for a population centred at one 'hill-top' to have a first member reach a nearby hill-top. Rate specified is the average number of mutations per genotype. Recombination has been used.*

Standard GAs tend to evaluate the whole current population, select from these and apply genetic operators to produce the next generation. A steady-state algorithm such as GENITOR (Whitley 1989b) replaces an individual at a time rather than a generation at a time. But since it always replaces the currently worst member of the population, it requires global communication of statistics about the whole population before carrying out such a replacement. In a network of processes running asynchronously, with the possibility of individual machines being down for periods of time, this negates some of the benefits of parallelism.

Tournament selection operates by taking two (or sometimes more) members of the population chosen at random, and choosing the best of this tournament to contribute genetic material to a new individual. There are a variety of ways to choose which old individual should sacrifice its place for the new. In a tournament of size two a copy of the winner, after application of genetic operators such as mutation or crossover (recombination), could replace the loser, or replace a randomly chosen member of the population. Sexual

recombination can take place between the winners of two different tournaments.

The practical advantage of this procedure when using a network of workstations is that it can be truly asynchronous and decentralised. Communication between machines is largely limited to occasional passing of genotype strings. If one, or a whole group of machines, slows due to loading or even is down for a time, the algorithm can carry on regardless on the remaining machines. It even becomes feasible to use several geographically dispersed different networks only occasionally communicating with each other by, e.g. electronic mail, with no need for any centralised control. Most tournaments would be 'local' within one network, as very little transmission of genetic material is needed between two otherwise isolated genetic pools for them to stay together. It thus becomes possible to use enormous amounts of computing power currently little used on present facilities.

For robotics applications it should be noted that the use of tournament selection reduces the evaluation of the robots to a very simple question: of two given robots which is (probably) the 'best'? If the robots are tested (in reality or in simulation) on a series of tasks of increasing complexity in a noisy environment, then the evaluation will become something like: which of the two got further before stopping? It is the power of evolution that complexity can be built up through a succession of such trivial 'questions and answers' each containing at the very maximum 1 bit of information. As with any selection mechanism which is equivalent to ranking, it is not necessary to have an evaluation function that returns a scalar value, which may simplify matters greatly. However, the extent to which noise in tournament selection — occasionally selecting the less fit by mistake — affects the stability of a species will be considered below.

In the game of *Twenty Questions* each reply can provide a maximum of 1 bit of information, hence at the end potentially discriminating between 2^{20} choices; but answers to inappropriate questions will provide much less than 1 bit, and frequently zero information. In an evolutionary approach to robotics it is the task of the genetic operators such as mutation, change-length, and recombination to generate new test cases that 'ask new questions' which tend to be appropriate. The following sections look at the problem from a theoretical perspective, and demonstrate that tournament selection can aid the genetic operators in this task.

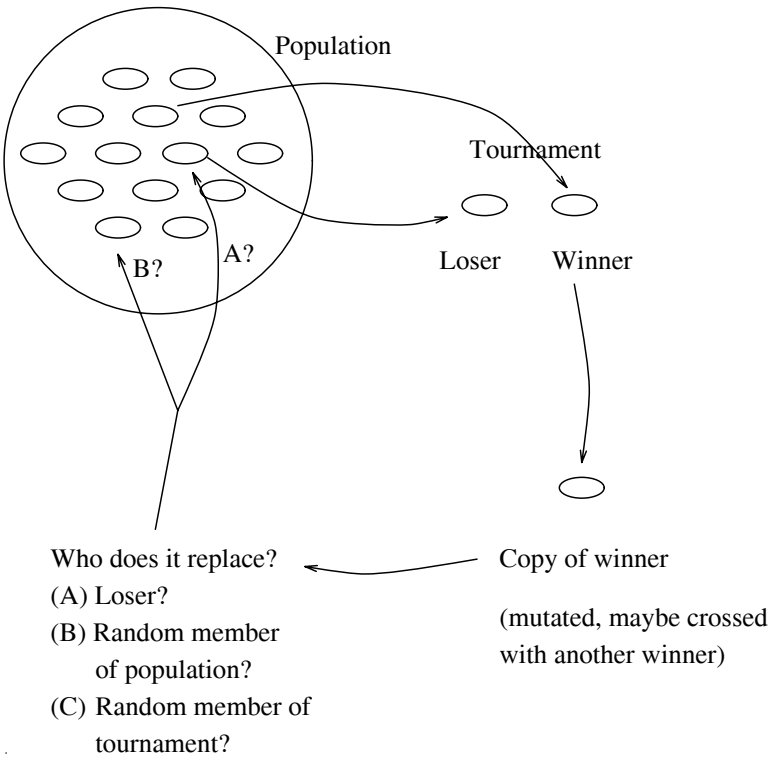

Figure 7.6: *Tournament Selection. Possible choices for who is to die to make way for the copy of the winner.*

7.5 Theoretical basis of Tournament Selection

Tournament selection relies on selecting randomly from the population a small number of contestants for a tournament, and taking the winner for further genetic processing to contribute to the future population. We will only consider a tournament of size 2, but within this constraint there is still a wide range of possibilities.

Tournament selection in GAs can be done on a generational basis; i.e. the current generation of population size P is held fixed, and for each of P members to be produced for the following generation, either one tournament is held to select a winner for cloning and mutation (if asexual); or two tournaments held to select two winners as parents, for recombination and mutation to generate an offspring. It can be shown (Goldberg and Deb 1990) that this is equivalent to a ranking scheme[2], in which the highest rank on average contributes 2 members to the genetic pool, the middle rank 1 member, and the bottom rank none. In Eigen and Schuster's terms, the superiority $\sigma = 2$. Such a generational method loses the advantage of asynchronous parallel computing mentioned earlier.

A steady-state method, which can be done asynchronously, involves generating a single

[2]The advantages of a ranking scheme are discussed in (Whitley 1989b)

offspring, either from the winner of one tournament (if asexual), or the offspring of two parents selected from two tournaments. This single offspring is then inserted as a replacement for one other member of the whole population, which otherwise remains unchanged. The interesting question is, who is to die so as to make way for the offspring, and some possibilities are (see Figure 7.6);

1. A randomly chosen member of the population.
2. A randomly chosen member of the tournament.
3. The loser of the tournament.
4. A subtle variation — do *not* reproduce from the winner, but remove the loser of the tournament and replace it by an offspring of a randomly chosen member of the population.

The effects of these methods will be described in terms of a notional generation; in N tournaments (where N is the population size), each involving two members, N new offspring are generated. The first method is similar in effect to the original generational basis, except that the superiority of the first rank is approximately e rather than two[3]. This superiority only holds true while each member of the population is in general separately ranked, and ceases being valid as soon as a significant number of the tournaments are draws — in other words, Eigen and Schuster's analysis no longer becomes directly applicable as their selection mechanism is radically different from tournament selection.

The second method has the same effect as the first – a randomly chosen member of the tournament will, in the long run, be just as 'averagely fit' as a random member of the whole population. So the σ value is e again, but the variance around this average figure over many runs is reduced compared to the first method, and there is no chance of removing the elite; except insofar as it will be replaced by a copy of itself that may have been subjected to some genetic operators. Since in half of the tournaments using method 2, the winner will be replacing itself, then $2N$ such tournaments are equivalent, as far as σ is concerned, to N tournaments using method 3. Hence a full notional generation of N

[3] An unique best member has an expectation of $2/N$ of appearing in a tournament, which it would win; and an expectation of $1/N$ of being the unlucky one chosen randomly to make way for a new offspring. Hence expected number of copies after 1 tournament is $(1 + 2/N - 1/N)$, and after N such trials is $(1 + 1/N)^N$. But $(1 + 1/x)^x \to e$ as $x \to \infty$.

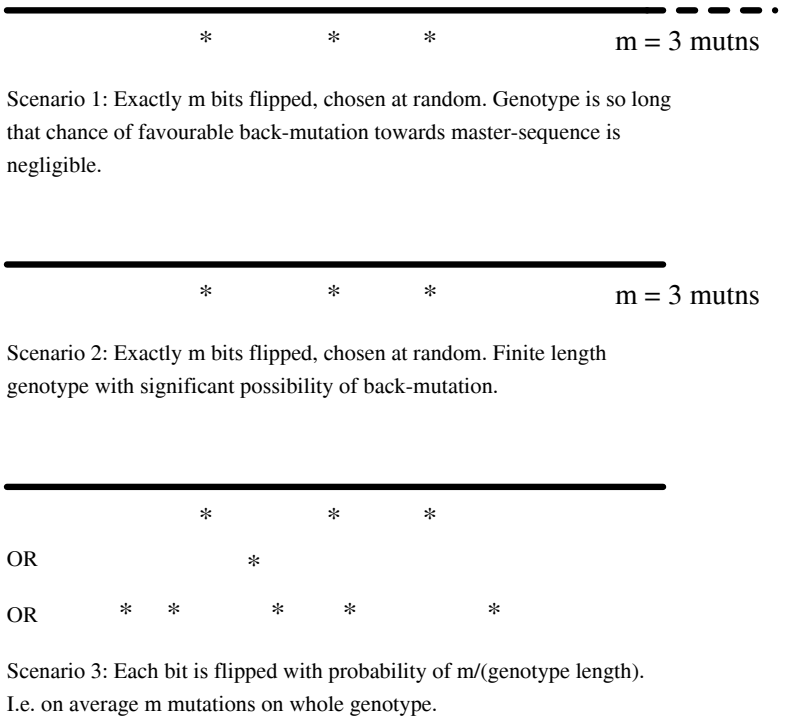

Figure 7.7: *Possible ways to apply a given mutation rate.*

tournaments with method 3 yields a σ value of ϵ^2. Surprisingly, method 4 gives the same value of σ as methods 1 and 2[4].

Method 3 will be the one discussed here, and the effect of mutation rates on a converged species (with a binary genotype) will be assessed for three different scenarios (see Figure 7.7):

1. A mutation rate of μ bits flipped per genotype, where μ is a small integer and the genotype length is so long that the possibility of back-mutations towards the current master-sequence can be ignored.

2. A similar mutation rate of μ bits per genotype on a genotype of length l, with the possibility of back-mutations.

3. An average mutation rate of μ bits per genotype, calculated independently at the rate of μ/l at each locus.

[4]In the generation of a single offspring, the unique best has $1/N$ chance of being chosen at random to be copied, and zero chance of losing a tournament and thus disappearing. Thus after one tournament its expected number of copies is $(1 + 1/N)$, as before. Once again the variance about this expectation is much reduced, in comparison to method 1. There is no possibility of the elite member being removed.

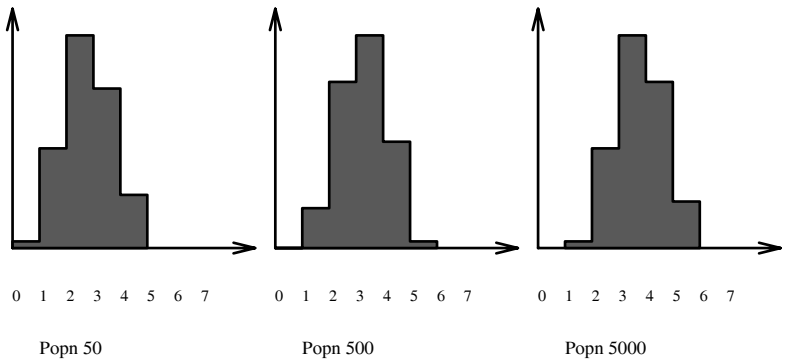

Figure 7.8: *Numbers in each class, ranked by the Hamming-distance from the master-sequence. Effectively infinitely long genotype, in that there are no back-mutations, but exactly one mutation in genotype at replication of winner of tournaments. Results shown are after simulations of number of tournaments equal to 4000 times population size.*

7.5.1 Long genotypes

In the first scenario, we can classify each member of the population by the Hamming-distance from the master sequence. This will increase by μ at each replication, giving possible distances of 0, μ, 2μ,..., so without loss of generality we need only consider $\mu = 1$. In the context of hill-crawling, our interest is in how the population distributes itself within different mutant-classes of size r_i, whose members have Hamming-distance i from the master-sequence. Given a tournament between members of distance i and j, for $i < j$ the winner i will remain in the population, and j will be replaced by a mutant of distance $i + 1$. A tournament-draw between members of the same distance i results in a winner i and the loser replaced by $i + 1$.

For maximum hill-crawling without losing the master-sequence (of distance 0) from the population, the long-term fate of this master-sequence should be considered. It can be seen that all tournaments between a 0 and a 0 result in the loss of one 0 to the population, and there is no other way in which 0s can be gained. If all tournaments are constrained to be between two different individuals, then r_0 will soon reduce to one member which will thereafter survive alone for ever. This member, the 'wild-type', will win all its tournaments and continually replenish the flow of mutants down the hill away from it. Histograms of results from populations of various sizes run in a computer simulation is shown in Figure 7.8. An equation which allows one to iteratively derive the expected size of each class is derived in Appendix A.

If the same individual can be chosen twice for a tournament, resulting in the replacement of itself by a mutated copy, then the wild-type will eventually be lost through just such an incident, and Muller's ratchet will start to operate. However, if tournaments are between different individuals, then the wild-type will never be lost, whatever the size of μ. We thus have a selection mechanism which can move the bulk of the population crawling down the hill as much as is desired, without ever encountering the error threshold of Eigen and Schuster. The banding into multiples of μ can be broken up by alternating between two different integers for μ.

But careful ...

There is a dangerous potential flaw in this. We are relying on the choice of winner of a tournament being 100% reliable, and in the context of evolutionary robotics, as discussed earlier, this may very easily not be the case. If the reliability of choice is $p < 1$, then sooner or later the wild-type will be lost and Muller's ratchet will start. A possible counter to this will be to only mutate the replica with probability $q < 1$, and otherwise leave it unchanged. In Appendix B it is shown that this will save the situation in an infinite population for values of $q < (2p - 1)/p$, this being independent of the value of μ. For example, if $p = 0.9$, we should have $q < 0.888\ldots$. In the case of a finite population, q should be reduced further to allow for the stochastic effects of genetic drift. The analysis of this case is not attempted here.

7.5.2 Shorter genotypes

Experiments run in simulation with populations of various sizes with genotypes of length 100, and with a mutation rate of exactly one bit flipped at replication, demonstrate what happens when the possibility of back-mutation is no longer negligible. The histograms in Figure 7.9 demonstrates a similar shape to those in the previous case, except that much more of the population stays close to the wild-type. It can be seen that in the case of unreliable choice discussed above, this effect will supplement that of any given value of q in countering the loss of the wild-type.

7.5.3 Mutations assessed independently at each locus

When the same experiment is run on genotypes of length 100, but in this case instead of exactly one bit flipped per genotype, there is a 1/100 chance of flipping at each locus on

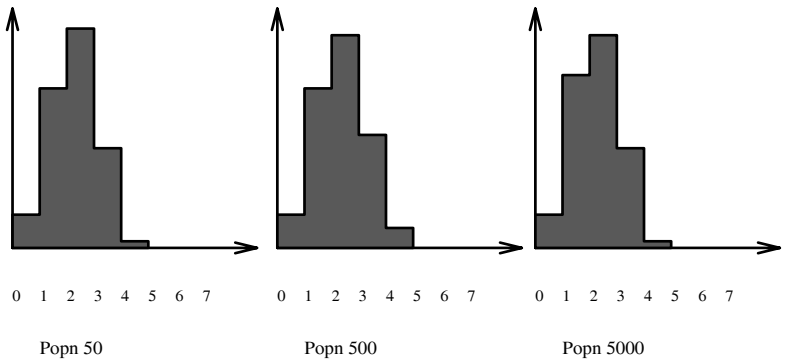

Figure 7.9: *Similar simulations to those shown in previous figure, except that genotypes are of length 100. Exactly one mutation per genotype at replication, which means a significant number of back-mutations towards the master-sequence.*

the genotype, the results shown in Figure 7.10 are startlingly different. The reason for this is that although the expected number of mutations per genotype is 1, this is made up from a probability of about $1/e$ of no mutations, about $1/0.99e$ of one mutation, about $1/1.98e$ of 2 mutations, and so on. The significant probability of there being no mutation has a similar effect to that given by the deliberate introduction of a probability $1-q$ of no mutation, discussed above in the long genotype scenario and in Appendix B. Appendix C shows that the proportion of the population expected to be the wild-type is $2/(1+e^{\mu})$, which when $\mu = 1$ gives a value of about 0.538. When there is a probability p of making a mistake in a tournament, it is also shown in this appendix that in an infinite population the wild-type will not be lost if $p > e^{\mu}/(1+e^{\mu})$, which for $\mu = 1$ gives a minimum value for p of 0.731. To use this equation in the other direction, if it is known that $p > 0.9$ then the maximum value of μ would be $\ln 9 \approx 2.19$. These figures would be worse in a finite population due to stochastic effects — indeed there is no way of 100% guaranteeing retention of the wild-type.

7.6 SAGA and recombination

It has been suggested above that the application of a change-length genetic operator at the very low rates required in SAGA can be treated in a similar fashion to low mutation rates, although the sleight-of-hand used is equivalent to increasing the number of possible alleles at the relevant loci. In the context of adjusting genetic operators so as to be able to influence hill-crawling without losing the current wild-type, the introduction of recombination makes a major impact.

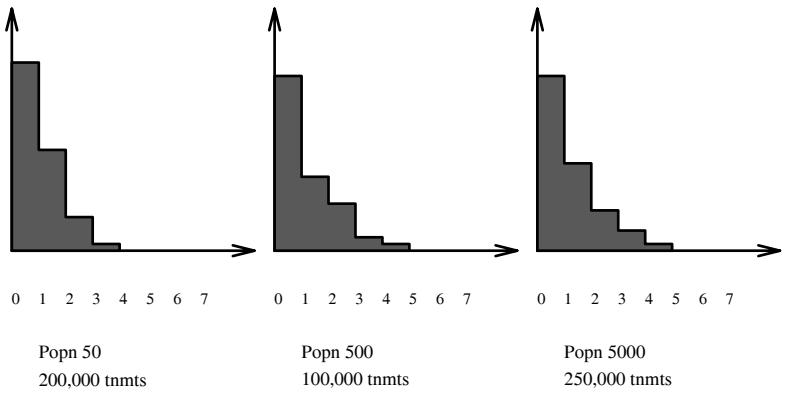

Figure 7.10: *Similar conditions to previous figure, except that there is a 1/100 chance of a mutation at each bit of the genotype, which is of length 100.*

One virtue of recombination within a species is that when two different favourable but improbable mutations take place within two different members of a population, then sexual replication can at a stroke produce an individual combining both; whereas asexual replication would require both improbable events to occur within the same single line of descent. It is this virtue which is stressed in standard GAs, yet here we will concentrate on another virtue — the other side of the same coin — which is that recombination functions as a form of repair mechanism protecting against Muller's ratchet (Maynard Smith 1978).

With tournament selection, candidates for recombination would be the winners of two separate tournaments, and the two offspring, after crossover and mutation, can replace the two losers. In general, the crossover will produce one offspring closer to the wild-type than the average of the two parents' distances, and another offspring further away than this average; after which mutation adds its toll. This constitutes a force producing a restorative flow towards the wild-type, allowing larger mutation rates without loss of the current local optimum. Simulations confirm this.

There are practical computational problems in dealing with recombination with the variable-length genotypes that are necessary in evolutionary robotics; given a crossover point in one parent genotype, where should the crossover point in the other parent genotype be? A discussion of this problem, and an algorithm which provides an efficient technique, are presented in Chapter 9.

7.7 Elitism in noisy finite populations

We have been seeking ways of avoiding loss of the wild-type, while promoting appropriate exploration. In GAs, the policy of retaining the best, unchanged, for inclusion in the

next generation is known as *elitism*, and we have seen that tournament selection gives you elitism for free, when the tournaments are 100% reliable and 'method 4' is used (see Section 7.5). In the presence of noise, we can counter the operation of Muller's ratchet by some of the measures mentioned above which guarantee preservation of the wild-type in non-zero proportions in infinite populations. For finite populations of a practical size, stochastic effects are significant, and the counter-measures are less effective.

7.8 Conclusions

An evolutionary approach to robot design, working from simple towards more complex cognitive architectures, implies species evolution within the SAGA conceptual framework. This requires a very different analysis from standard GAs, and abandoning the goal-seeking metaphor associated with them. A new metaphor of hill-crawling of a converged species has been introduced, and this needs an analysis of the conflicting forces of exploitation and exploration — which here means efficiently searching down the current hillside and along high-value ridges in the fitness landscape while being careful not to lose track of the current hilltop.

Whereas theoretical biologists are trying to analyse the selection mechanisms they believe exist in the natural world, in simulated evolution we can choose our own selection mechanism. Arguments have been presented that tournament selection can be used for hill-crawling, with significantly higher mutation rates than are used in conventional GAs; higher mutation rates enable a faster rate of evolution. It has also been argued that with the complex simulations that would be needed in evolutionary robotics, requiring serious computing power for each individual being evaluated, tournament selection allows a practical evolutionary setup to be highly distributed over an asynchronous network or networks of machines with minimal intercommunication. In addition, tournaments reduce the selection process to a succession of binary decisions as to which of two individuals is the better, avoiding scaling problems with any evaluation function.

Analytical results have been shown for the effects of tournament selection in the case of infinite populations, with and without reliable tournament decisions. Stochastic effects of genetic drift in small populations alter these results. Results from simulations with finite populations under different conditions have been shown, and practical ways to overcome Muller's ratchet have been suggested.

CHAPTER 8

Drift[1]

8.1 Introduction

This chapter has three naturally separate parts. What they have in common is that they all show processes where the forces of selection are overridden or bypassed — where the effects of finite populations produce consequences other than those predicted by infinite population models. When discussing artificial evolution, of a converged finite population over long periods, then the issue of drift becomes much more significant than it might be in many standard function optimisation problems.

In the situation discussed in the first part, some unexpected anomalies turn out to be due to random genetic drift; in this context the drift is a 'bad' thing, and the discovery of it here prompted me to analyse random genetic drift in haploid populations. The full analysis is given in an appendix.

Following this, I turn to work by Schuster on a model of RNA fitness landscapes, which demonstrates the potential for neutral drift to take a genetically converged population travelling long (Hamming) distances across sequence space. This form of neutral drift can be a 'good' thing; at least it is symptomatic of how evolutionary search spaces (in so far as 'search space' is a useful metaphor) are much more traversable in practice than might at first sight have been thought.

Turning then to a particular experimental run of an evolutionary robotics task which will be fully discussed in a later chapter (Section 12.4.1), analysis of the population shows that, although genetically converged from an early stage, it has nevertheless drifted considerable distances through sequence space. Finally, some brief comments will be made about sexual selection.

[1] This chapter incorporates two papers presented at the International Conference on Genetic Algorithms 1993, (Harvey 1993b, Harvey *et al.* 1993a), and an expansion of the latter in a Technical Report (Harvey *et al.* 1993b).

8.2 Genetic Drift

If a coin is tossed 100 times, then on average it will be heads 50 times, but it is unlikely to be exactly 50. The same holds if 100 random selections are made with replacement from 50 heads and 50 tails, without turning any over. In the selection case, repetition of the process will on each occasion on average give you the same result as on the previous occasion, but the variance allows significant change in this average over time. If at any stage the selection resulted in all heads (or tails), then future change would be impossible.

The consequence is that, in the absence of mutation, and even without any selection, a population will eventually converge to all one value or all the other. This also holds true for low values of mutation and/or selective bias, and the critical values which permit genetic drift to be significant can be calculated. The theory of genetic drift is analysed in the field of population genetics, but usually for a diploid population with fitnesses affected by dominant and recessive alleles.

8.3 The Hinton & Nowlan paper

In an important and elegant paper (Hinton and Nowlan 1987), Hinton and Nowlan demonstrate with a deliberately simple example the Baldwin effect, wherein the ability of a phenotype to adapt in its lifetime (ability to 'learn') alters the fitness landscape of the corresponding genotype. This has the consequence that selection within a population moves the genotypes towards the region where the adaptations, that were originally made in the lifetime of the phenotypes, are genetically fixed. This has the appearance of Lamarckism, but is not so, as there has been no direct flow of information from the adapted phenotype to the genotype.

The model chosen as an example uses genotypes with a number of genes that can be specified as incorrect, correct, or open to adaptation during the lifetime of the phenotype. The evaluation function only favours those phenotypes that, within a finite lifetime, find a perfect solution through a combination of 'correct' genes, and 'adaptive' genes which successfully adapt. It is demonstrated that with the application of a standard genetic algorithm (GA) to the population as specified, the number of incorrect alleles on the genotype rapidly decreases to zero; the number of correct alleles increases at first rapidly and then slows down; the number of undecided (adaptive) alleles decreases slowly. If the same experiment is tried out only with correct and incorrect genes, and no adaptive ones, then the 'needle in a haystack' nature of the single perfect solution means that only

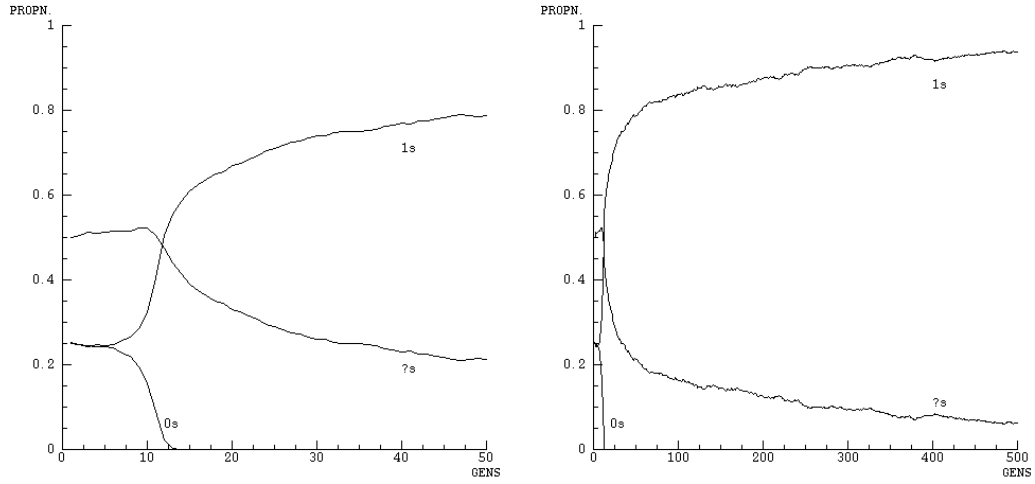

Figure 8.1: *The proportions of incorrect, correct, and undecided (adaptive) alleles (***0***s,* **1***s,* **?***s) in the whole population, against generations. On the left, the first 50 generations of a run, and on the right the same continued for 500 generations.*

random search works, and takes an unreasonably long time.

The main thrust of Hinton and Nowlan's paper is endorsed here, but a subsidiary matter that is mentioned as an aside there is taken up as the main point for investigation here (Hinton and Nowlan 1987) page 497:

> One interesting feature of [the figure] is that there is very little selective pressure in favor of genetically specifying the last few potential connections, because a few learning trials is almost always sufficient to learn the correct settings of just a few switches.

The figure in question indicates that there could be an asymptote at a relative frequency of about 0.45 below which the number of undecided alleles will not fall.

My own re-implementation of the model usually shows an asymptote at between 0.05 and 0.2. A typical run is shown in Figure 8.1, showing the dramatic changes in the first 50 generations, and the longer term behaviour over 500 generations. The variations between runs is indicated in table 8.1, showing the values at the end of 20 runs of 500 generations each. The re-implementation by Belew (Belew 1989) shows 'an almost steady-state' at about 0.3. He asserts that the curve is 'in fact asymptotically approaching ...0.0'. This I will demonstrate to be false, in the general case; the analysis of what is really happening shows that the combination of genetic drift and the hitch-hiking effect so completely swamps the selective pressures that some of the genes are completely converged to the undecided value, rather than the 'correct' one.

Strictly speaking, if there is even the smallest amount of mutation in the system, applied

Propn of ?s at 500 gens.	Loci having >50% ?s	Loci having 100% ?s	Propn of ?s at 500 gens.	Loci having >50% ?s	Loci having 100% ?s
0.063	1	1	0.108	2	2
0.109	2	2	0.093	2	1
0.082	1	1	0.150	3	3†
0.123	2	2	0.150	3	3†
0.118	2	2	0.112	2	2
0.074	1	1	0.100	2	2†
0.107	2	2	0.093	2	1
0.200	4	4†	0.121	2	1
0.134	3	2	0.115	2	2
0.092	2	0‡	0.115	1	1

Table 8.1: *The final proportions of undecided alleles after 20 runs each of 500 generations, with no. of loci converging or converged on* **?**. *4 runs*† *have in fact completely converged at all 20 loci, only one run*‡ *does not yet have a locus with* **?** *fixed.*

independently at each locus on each genotype, then if you are willing to wait long enough you will see *any* population state; even a population entirely composed of incorrect alleles, or entirely composed of undecideds. This would be a transitory phenomenon, and the necessary timespans are way beyond those being considered here. The diffusion equation analysis of genetic drift given below gives a picture of those circumstances under which genetic drift can be expected to be a significant force.

Genetic drift — the consequences of random fluctuations in relatively small populations — is a matter of fundamental concern to population geneticists, but is frequently ignored by people using GAs. Since 1000 is often 'relatively small' in this context, and GAs frequently use population sizes of 100 or less, to ignore genetic drift is commonly, as in this example, to ignore one of the fundamental processes underlying the phenomena.

I include a number of graphs (figures 8.3 and 8.4) indicating the relative influences of selection, mutation and genetic drift for different parameter values. I also give a reworking of standard genetic drift analysis taken from population genetics theory, and adapted to the haploid models common in GAs.

8.4 The model

For fuller details of the model used in the demonstration I refer you to the original paper, and to a subsequent analysis by Belew (1989). As a brief summary, the model has a population of 1000, each with genotypes with 20 genes having possible values **0**, **1** and **?**. In the initial population these are randomly selected with probabilities 0.25, 0.25 and 0.5. The derived phenotypes are taken to be a set of 20 switches, which undergo a series of up to 1000 trials each. The allele **0** at a particular gene specifies that the corresponding switch is set incorrectly, **1** specifies that the switch is set correctly, and a **?** indicates that the corresponding switch is flipped randomly at each trial. The series of trials on a phenotype is stopped when all switches happen to be set correctly, on trial number i, or alternatively at $i = 1000$, the final trial, if there is no success. Of course, if any of the alleles in the genotype are **0**, i.e. some switch is genetically fixed at the incorrect position, inevitably the trials will run the full course until $i = 1000$.

The fitness F is then calculated from i by the formula $F = 1 + 19(1000 - i)/1000$. This gives an all-perfect phenotype, which needs no trials to reach success, a fitness of 20; while one which never succeeds ($i = 1000$) either through being born without a chance (one or more alleles of **0**) or through failing despite having a chance, has a fitness of 1. The necessary equations to calculate the expected fitness are given in Appendix D. If q is the number of undecided alleles in a genotype which otherwise is correct, then for $q > 14$ the expected fitness is near to 1; for $q < 5$ the expected fitness is near to 20. The sharp transition is shown in Figure 8.2.

8.4.1 Early stages ...

At each generation the relative attained fitnesses of each member of the population determine the probability of that member contributing to the reproductive pool for the next generation. In the early stages, virtually all the members will have the same minimum fitness. Something similar will happen also at the later stages, after the incorrect (**0**) alleles have been eliminated; virtually all members will have small q-values, and hence, because of the flatness of the curve for $F(q)$ at small q, nearly identical fitnesses. At both these stages there is very little selective pressure.

However, as Figure 8.1 indicates, typically around generations 5 to 15 successful members emerge with a fitness nearly 20 times as great as that of the original random members. This enormous selective differential operates near-exponentially for a few generations, giv-

q	F(q)	q	F(q)
0	20.000		
1	19.962	11	4.965
2	19.924	12	3.140
3	19.848	13	2.113
4	19.696	14	1.568
5	19.392	15	1.287
6	18.784	16	1.144
7	17.569	17	1.072
8	15.233	18	1.036
9	11.649	19	1.018
10	7.868	20	1.009

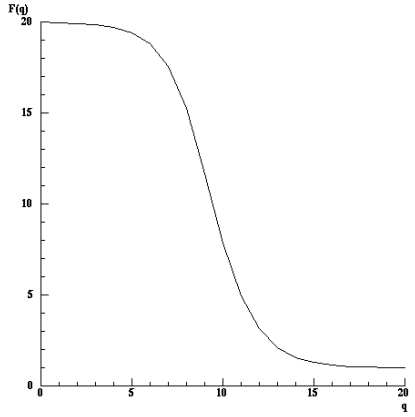

Figure 8.2: *The expected fitness $F(q)$ of a gene with q undecided alleles and $(20-q)$ correct ones. For $q = 0$ it has been assumed that success was on the 'zero-th' trial, to give a fitness of 20.*

ing the sharp swings indicated in the figure. If the fitness function is adjusted to give a spread of fitnesses from 1 to 2, rather than 1 to 20, this transition is typically delayed until perhaps generation 50, but due to its fundamentally exponential nature it is then a similarly sharp transition.

8.4.2 ...Hitch-hiking...

During this transition the genetic material of the first high-scorers dominates the reproductive pool. By marking the genetic material of the first 'winner', and then tracing the marked genes in later generations as they are selected and recombined with others, it can be seen that typically within 10 generations of appearance 50% or more of the whole genetic pool is derived from that first winner. Hence the accidental pattern of **1**s and **?**s in that first winner has a strong chance of dominating future generations after selection has ceased to be a major force — the 'hitch-hiking' effect.

8.4.3 ...then Genetic Drift

Once that has happened, genetic drift will allow the proportion of **?**s at any one locus vary until it has reached either 0% or 100%, when in the absence of mutation change will cease; and even in the presence of low mutation a stable state can be expected. In the

complete absence of selection, then since expected changes from generation to generation do not alter the expected mean, from an initial position of $x\%$ **?** alleles one can expect $x\%$ of the time convergence to all **?**s, and $(100 - x)\%$ of the time convergence to all **1**s.

The run shown in Figure 8.1, which is also the first example in table 8.1, has at 500 generations one locus 100% converged to **?**s. Hence the appearance in the graph of a long-term trend towards no **?**s is deceptive, as the asymptote will be at 5%. Table 8.1 gives an idea of the variations in these figures over 20 separate runs.

In (Hinton and Nowlan 1987), no mention is made of any mutation, and I am unable to account for the much higher asymptote indicated at 45%. It may be an artefact of some idiosyncracies in the programming of the algorithms, or the fact that the graph is hand-drawn may show that it is meant to be loosely indicative only.

In Belew's paper the asymptote appears to be at about 30%. This is significantly outside the range covered in my simulations. Mutation is mentioned, without specifying a rate. In the two-bit coding for each locus there described, what is in my terminology **0**, **1**, and **?** translate into respectively **10**, **11**, and either **00** or **01**. The early selection to eliminate **0**s (in my terminology) would eliminate **1**s from the left-hand bit of each pair; the occasional mutation in these left-hand bits will be swiftly eliminated by strong selection. The right-hand bits would then distinguish between (in my terminology) **1**s and **?**s. No mutation rate here can explain the high asymptote shown.

Belew gives three versions of an explanation for this asymptote, which can be summed up as suggesting that selective pressures are so low that 'the probability of producing more than an average number of offspring is infinitesimal'. For a selection as implemented in a standard GA, this would not only be the wrong answer, but it would be almost diametrically opposite to the truth. At a single locus which has exactly 300 out of the 1000 population set to the one value, the probability of the next generation having exactly 300 set again, in the absence of selection, is

$$\binom{1000}{300} (0.3)^{700}(0.7)^{300} = 0.0421$$

I.e. only some 4% of the time will exactly 300 of the next generation have this same allele at this locus, and of the balance some 48% of the time more than 300 will (and some 48% of the time less than 300 will). It is this variation, the *improbability* of remaining at the same proportion from generation to generation, which constitutes genetic drift.

An additional factor to increase genetic drift in this particular model is the fact that

q	F(q)	s(q)	2sN
0	20.000		
1	19.962	0.00190	3.80
2	19.924	0.00191	3.82
3	19.848	0.00383	7.66
4	19.696	0.00772	15.44
5	19.392	0.01568	31.36
6	18.784	0.03237	64.74
7	17.569	0.06916	138.31
8	15.233	0.15335	306.70

Table 8.2: *The selection s for a gene with q undecided alleles and* $(20 - q)$ *correct ones is calculated from* $s(q) = (F(q) - F(q-1))/F(q)$. *The population size N is 1000.*

the fitness evaluation is non-deterministic, and has a very large variance. Where the fitness is deterministic this would obviously have no variance. In most non-deterministic evaluations, by taking a reasonably large sample size the variance is normally reduced to insignificance, as the variance of a sample of size N is $1/N$ times the variance of the population it is sampled from.

However, Belew mentions that he uses the GENESIS (Grefenstette 1983) GA simulation facility. Into this (or at least the later versions) is built the ingenious selection algorithm, due to Baker (1987), which "guarantees that the number of offspring of any structure is bounded by the floor and the ceiling of the (real-valued) expected number of offspring". In other words, although the expected number of offspring is maintained at the correct value, the variation about this value — and it is this variation which is associated with genetic drift — is reduced to a minimum. Nevertheless, this variation is still significant.

8.5 A Diffusion equation approach to Genetic Drift

Goldberg and Segrest (1987) give a finite Markov chain analysis of genetic drift. The alternative approach is using a diffusion approximation. In Appendix E I set out such a diffusion approximation which starts from Roughgarden (Roughgarden 1979), where an analysis is given for diploid systems, and makes the necessary alterations for a standard haploid genetic algorithm.

The underlying basis for this approach is that of considering one experiment with a particular set of parameters such as population size, selective bias, and mutation rate; considering one particular locus, and taking a census across the population to find the distribution of the possible alleles at this locus, *after* sufficient generations have passed for any initial transitional phenomena to have died away.

For instance, in a particular experiment the proportion of **0**s at this locus will be x in the range $[0.0, 1.0]$. But because of the stochasticity, a series of experiments will give different values of x. Hence a whole ensemble of such experiments are considered — using the same parameter settings for the whole ensemble. The census results on different members of the ensemble will vary, but the probabilities of different census results can be calculated analytically.

The size of the population N is assumed large enough for it to be valid to make a continuous approximation to the discrete steps actually taken — the proportions of any allele can in fact only change in steps of size $1/N$. It is assumed that the proportions in a population of each allele at one locus can be analysed independently of what is happening at other loci, which are taken to be either fixed or with no interdependence on this locus. An ensemble of populations is considered, all acting under the same forces of selection, mutation and drift. It is assumed that from any starting position, this ensemble will spread out under these forces until some equilibrium is reached. This equilibrium will be shown to have strikingly different features depending on the values of the parameters. The figures later on characterise the features of such an ensemble, and hence give a perspective on what might plausibly happen in any one individual population.

The diffusion can be analysed with the same equations as are used for physical processes. A 'diffusion equation' is introduced which approximates the Markov chain. In the analysis of a physical system of diffusion we let $\rho(x,t)$ denote the density of particles at location x at time t. (The translation to our ensemble of populations is: let $\rho(x,t)$ denote the proportion of populations in the ensemble that at time t give a census return of x for the proportion of 1s at the relevant locus.) The flow across a surface at x is $J(x,t)$. The change in density at a location is equal to the spatial derivative of the flow.

$$\frac{\partial}{\partial t}\rho(x,t) = -\frac{\partial}{\partial x}J(x,t) \tag{8.1}$$

In the current context the expression for $J(x,t)$ contains a term for external forces — mutation and selection — and another term for diffusion.

$$J(x,t) = M(x)\rho(x,t) - \frac{1}{2}\frac{\partial}{\partial x}V(x)\rho(x,t) \tag{8.2}$$

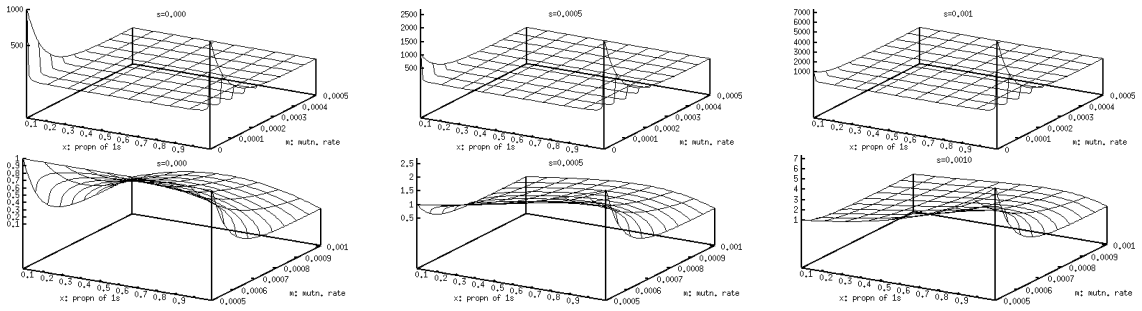

Figure 8.3: *Equilibrium distributions, varying m for particular values of s. The horizontal scale is the proportion x of the allele being selected for, in the range x = 0.001 to 0.999. The vertical scale varies from graph to graph, hence for the U-shaped curves, only the general shape is indicative.*

In a short time interval Δt, $M(x)\Delta t$ is the average distance travelled from a point x under force of mutation and selection. $V(x)\Delta t$ is the variance of the distances travelled.

We are interested in the equilibrium distribution $\hat{\rho}(x)$, where it exists.

A selective force s is defined assuming that the schema fitness of the allele **0** is f_0 and of allele **1** is f_1. The average fitness $\overline{f}$ depends on the proportion x of **1**s in the current population, $\overline{f} = (1-x)f_0 + x f_1$. We shall define the selective force s in favour of allele **1** as

$$s \equiv \frac{f_1 - f_0}{f_0} \tag{8.3}$$

The population size is N, the mutation rate is m. In Appendix E it is shown that:

$$\hat{\rho}(x) = \frac{c(1+sx)^{2N}}{[x(1-x)]^{1-2mN}} \tag{8.4}$$

where c is a normalising constant.

Here it is clear that the term on the top relates to the forces of selection; whereas the denominator, symmetrical in x and $(1-x)$, shows different characteristics depending on whether the exponent is positive or negative — which depends on the relationship between mutation rate and population size.

8.5.1 Varying the mutation rate

In the case of zero selective force, $s = 0$, this becomes

$$\hat{\rho}(x) = c x^{2mN-1}(1-x)^{2mN-1} \tag{8.5}$$

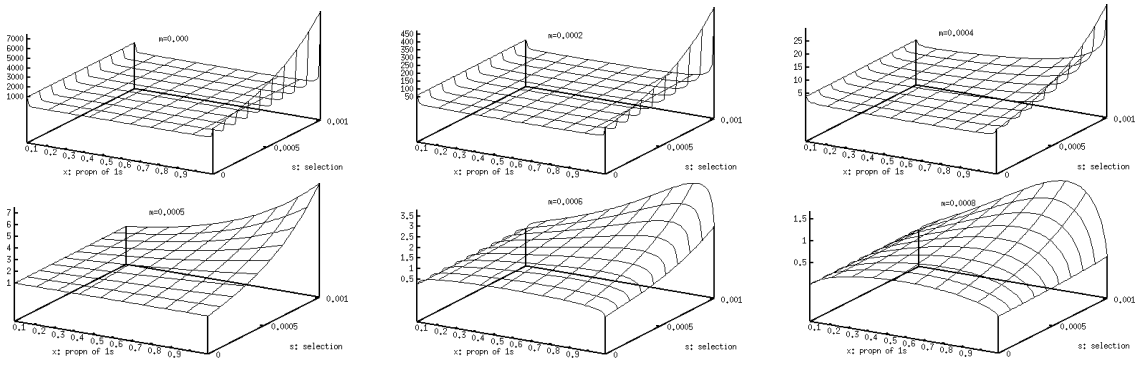

Figure 8.4: *Equilibrium distributions, varying s for particular values of m. See caption to Figure 8.3*

The behaviour of eqn. 8.5 varies dramatically according as to whether $2mN < 1$ or $2mN > 1$. In the former case of low or non-existent mutation the curve is the U-shaped $\frac{1}{x(1-x)}$, demonstrating that the population will converge completely on one allele or the other. In the latter case of mutation significantly high in relation to the population size, then the reverse will happen and the distribution will be centred on $x = 0.5$. If selection is positive rather than zero, then either the U-shaped curve or the humped curve, as appropriate, will be skewed towards the side favoured by selection.

The graphs in figures 8.3 and 8.4 are indicative only of the general shape. In particular, for the U-shaped curves demonstrating genetic drift, the constant c in (8.5) would need to be zero for the area under the curve to be unity. This gives a vertical bar at $x = 0$ and at $x = 1$ (emphasized in the figures), with zero elsewhere. For this reason the graphs are only shown for x =0.001 to 0.999.

8.5.2 Varying selection

When m=0, then (8.4) becomes

$$\hat{\rho}(x) = \frac{c(1+sx)^{2N}}{x(1-x)} \tag{8.6}$$

which is the limit of a U-shaped curve. The denominator is symmetrical in x and $1-x$. For an indication of the relative proportions of the converged population that settle at $x = 0$ or $x = 1$, the numerator should be considered for these values; although not much reliance should be placed on this, as it is exactly here that the diffusion approximation breaks down with a finite size population.

Nevertheless, for $s = 0$ the numerator is constant, and as s increases the numerator is more at $x = 1$ than it is at $x = 0$. $(1+s)^{2N}$ becomes $\mathcal{O}(e)$ when $2sN = 1$, and increases

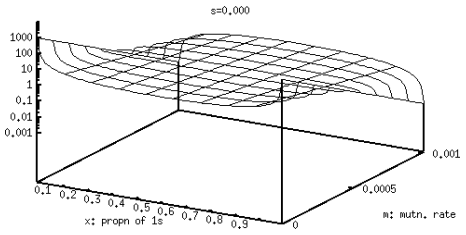

Figure 8.5: *Eqn. 8.5 for $c = 1$, and m ranging from 0 to 0.001. Here the vertical axis is log-scaled, and selection is zero. The transition between U-shaped and $\bigcap$-shaped curves at $2mN = 1$ can be clearly seen.*

exponentially as s increases above this value. Hence when $2sN \ll 1$ we can expect the two arms of the U-shaped curve to be nearly equal in size (i.e. selection is insignificant), and when $2sN \gg 1$ the arm that selection favours will predominate.

Returning to the Hinton and Nowlan example, the value of s can be calculated when there are q undecided alleles and $20-q$ correct ones. The selective force s at a locus which could change one undecided allele to a correct one can be calculated from the schema fitnesss in Figure 8.2. For any q, it is $(F(q-1) - F(q))/F(q)$, which is calculated in table 8.2. It can be seen that, whereas convergence on the 'wrong' value can be expected for $2sN \ll 1$, in this case convergence on the wrong value occurred for q as great as 4, and hence $2sN$ as big as 15. This can be explained as due to the hitch-hiking effect, where before genetic drift could take over as the population settled down to equilibrium, the swamping of the population by the very high selection in favour of the first successful genotype has resulted in near-convergence on the 'wrong' value.

8.6 Summarising the Diffusion equation results

A puzzling anomaly in the Hinton and Nowlan paper has been explained as the result of genetic drift, due to low selective forces, following a period when very high selective forces and the 'hitch-hiking effect' have distorted the proportions of alleles at temporarily irrelevant loci. By analysing this case in detail, the significance of genetic drift has been brought out, and using the diffusion equation approach some more general results demonstrated.

These can be summarised as the following, where m is the mutation rate, N is the population size, and s the *selective force* in favour of a particular allele 1, at a binary locus, is defined as $(f_1 - f_0)/f_0$, f_1 and f_0 being the schema fitness of alleles 1 and 0:

If and only if $2mN < 1$ then the population will converge at this locus on one

value or another; and if $2sN \ll 1$ it will be almost equally likely to converge on the 'wrong' value as the 'right' one.

Where 'hitch-hiking' takes place, even for $2sN$ somewhat larger than 1, convergence on the wrong value can still happen.

8.7 RNA Sequence space and Shape space

Schuster (1992) discusses issues of potentially great relevance to the work of this thesis, but in the rather different context of RNA evolution.

Sequence space for all RNA molecules of a fixed length ν is clearly defined, with a metric based on Hamming distance, or the number of point mutations required to change one sequence into another. If this is thought of as genotype space, then one possible phenotype space for RNA molecules is *shape space*, defined as the set of all RNA secondary structures formed by all sequences of chain length ν derived from a given base pairing alphabet.A metric for shape space is defined by converting RNA secondary structure into equivalent trees. The secondary structure of RNA, in terms of stacks (paired complementary sections of RNA) loops (between stacks) and free ends, is uniquely equivalent to a tree (though in general one tree is equivalent to many sequences); the conversion from sequence to tree can be done by an algorithm. The trees can be handled algorithmically, and a "tree distance" is defined in terms of the minimal number of tree editing steps required to convert one tree into the other. The tree distance induces a metric on the shape space.

So there are now separate metrics on both sequence space and shape space — if you like, genotype space and phenotype space. In the discussion that follows, it will be suggested that some of the properties displayed by this RNA world may also hold in the different world where genotype space is that of genotypes specifying robot control and sensory architectures, and phenotype space is that of the corresponding robot behaviours. The differences between these two worlds must be acknowledged — an obvious one being that experiments with RNA involve population sizes many orders of magnitude larger than is likely with robots.

8.7.1 Shape space covering

There are far less shapes than sequences. For RNA the number of secondary shapes is approximately $1.485 \times \nu^{-3/2}(1.849)^{\nu}$, which is much smaller than the number of sequences of length ν of four bases, 4^{ν}. An analysis of the distribution shows that relatively few structures (shapes) are common in sequence space, and many structures are rare. In fact

a Zipf law holds, relating frequency of structure against frequency-ranking of structure. Sequences folding into the same secondary structure are essentially randomly distributed through sequence space. Since there are relatively few common structures, within any relatively small patch of sequence space examples of *all* common structures can be found.

To quote from (Schuster 1992), page 12:

> For natural **AUGC**-sequences of length $\nu = 100$ a sphere of radius $h = 16$ (in Hamming distance) is sufficient to yield already the global distribution of structure distances. In this case we can expect all common structures to be found in such spheres in sequence space. There are as many as 6.2×10^{25} sequences in such a ball. Although this number is large, it is nothing compared to the total number of sequences of this chain length: $4^{100} \simeq 1.6 \times 10^{60}$.

The implication is with RNA *all common phenotypes are realised within a relatively small neighbourhood of any random sequence.*

8.7.2 Percolation

In the RNA world, through computer simulations "neutral paths" through shape space were searched for; in other words, paths connected in sequence space by single mutations, Hamming distance one apart, but "neutral" in the sense that all points on this path folded into the same secondary structure. Such a neutral path ends when no further sequence giving the same shape can be found in the neighbourhood of the last sequence.

The maximum length of such a neutral path is of course the chain length, ν. The simulation of RNA space for $\nu = 100$, using the shape-from-sequence, tree-from-shape, and tree-editing algorithms mentioned above, showed that about 20% of the neutral paths have the maximum length. Such paths lead through sequence space to an end sequence differing in every position from the start sequence, yet both ends of the path and every intermediate point share the same shape.

This percolation property means that in artificial evolution — Schuster uses the example of evolutionary biotechnology — it can be advantageous to adopt alternations of selection cycles with low and high error rates (below and above the critical rates discussed in Chapter 7). At the low error rates normal search takes place in the neighbourhood of the current consensus sequence. If no improvement is found, then a switch to a higher error rate allows the population to spread, through this percolation property, along neutral paths to other regions of sequence space; these can then be explored in detail when the error rate is reduced again.

8.7.3 Tractability

The lesson to be learnt from this is that, at least in the case of RNA space as modelled above, the problem of evolution is *tractable*. To quote again from Schuster (1992) page 13:

> The structure of shape space is highly relevant for evolutionary optimization in nature too. It provides a firm answer to the old probability argument against the possibility of successful adaptive evolution. How should nature find a given biopolymer by trial and error when the chance to guess it is as low as $1 : \kappa^{\nu}$? .....The numbers of sequences that have to be searched in order to find adequate solutions in adaptive evolution are many orders of magnitude smaller than those guessed on naive statistical grounds. In the absence of selective differences populations drift readily through sequence space, since long neutral paths are common. This is in essence what is predicted by the "neutral theory of evolution", and what is often observed in molecular phylogeny by sequence comparison of different organisms.

It should be realised that these conclusions in RNA space depend on the models and simulations made. Particular factors that should be borne in mind are:

1. In RNA shape space, the fact that folding depends entirely on complementary matching of characters in the string means that inevitably, given one string, the string that is complementary at all points will fold into the same shape; this obviously has implications for the completeness of percolation through sequence space.

2. The swapping of any 2 complementary bits that are matched in folding results in an identical folded structure, at the expense of a move of length 2 in sequence space.

3. The reversal of a bit that is unmatched during folding similarly results in unchanged structure, at the expense of a move of length 1 in sequence space.

4. In general Schuster's analysis of movement within such spaces assumes population sizes many orders of magnitude larger than those considered in genetic algorithms — but see comment in Chapter 7 that suggests some of the results are applicable with population sizes of the order of 100.

Despite particular factors in the case of the RNA model, these results help to change peoples' intuitions regarding sequence space — distances in high-dimensional landscapes work in a very different way from the 3-dimensional landscapes we often visualise. These properties of high-dimensional landscapes will work in our favour for genetically determined behaviours also, and artificial evolution can be many orders of magnitude easier than might be guessed on naive statistical grounds (Kauffman 1993).

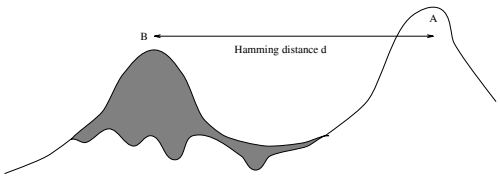

Figure 8.6: *In the fitness landscape a population searches preferentially along 'ridges' of relatively high fitness ...*

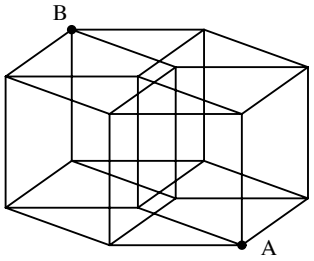

Figure 8.7: *... and in a high-dimensional space there are an enormous number of shortest and nearly shortest paths between two points.*

8.8 Neutral drift

The most visible difference between genotypes under evolution, and those in a standard GA for function optimisation, is that at all times (bar perhaps the very start) the population is virtually genetically converged. In a standard GA this is usually considered the end of the story. The received folklore is that recombination is the driving force for genetic search, and mutation is only a background operator. To quote from (Smith and Goldberg 1992), page 255:

> Clearly the $\mathcal{O}(n^3)$ estimate [for implicit parallelism] is based on a *diverse* population, where many schemata are represented. However, as exponential allocation of observed-best schemata accrues, one can expect that the number of building blocks processed will decrease. This is an inevitable consequence of convergence in the [vanilla-flavour] GA outlined above. After convergence, the GA population will be composed primarily of copies of one individual. The only diversity maintained in the population after convergence is a result of mutation. Note that mutation is a completely random operator that is unguided by the algorithm's observations of fitness values over time.

Now whereas it is undisputed that mutation within an individual is completely random, it does not follow that random mutation on the individuals within a converged population under selection results in random undirected movements of the population across the fitness landscape. Treating the converged population as being currently centred around some local hilltop, then mutations can be thought of as explorations away from the peak; with long genotypes the chance of a back-mutation is insignificant. But successive rounds of the mutation-selection cycle do not explore further away in an undirected fashion, but rather seek out any ridges of relatively high fitness in the landscape that may lead to even higher peaks, as discussed in Chapter 7 (Fig. 8.6).

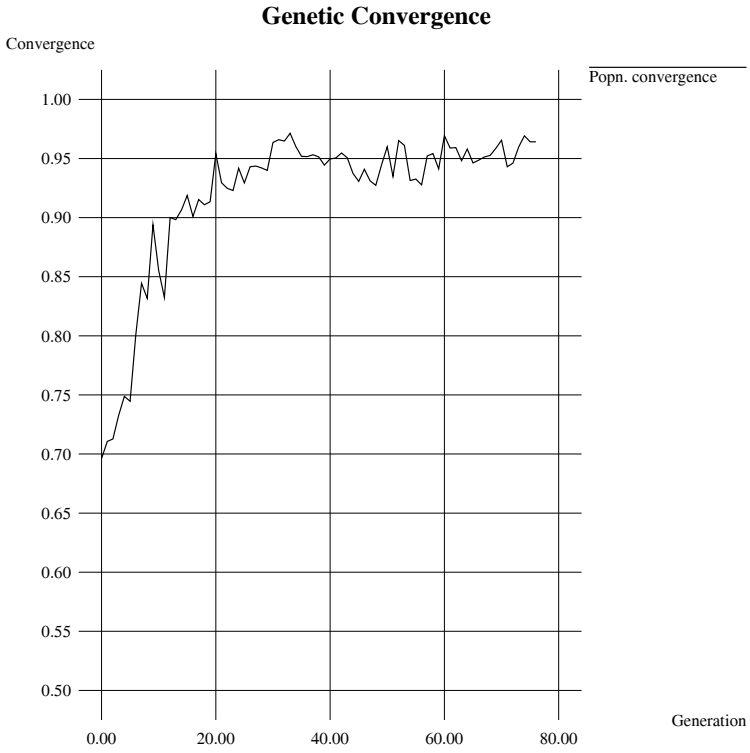

Figure 8.8: *Genetic convergence against generations, calculated as the percentage agreement between pairs of genotypes taken from the population.*

The fitness landscape metaphor is potentially misleading, in that high-dimensional spaces have properties very different from our intuitions about 2-D or 3-D spaces. Whereas in a normal 3-D landscape there can at best be a single ridge between two hills taking the direct shortest route, this is no longer the case in sequence space, which can be thought of as having n dimensions where n is the genotype length. As indicated in Figure 8.7, between two points Hamming distance d apart in binary genotype sequence space, there are $d!$ shortest paths, and far more slightly longer ones. This is why, in any high-dimensional landscape that is smooth enough for there to be some correlation in height or fitness between neighbouring points, any local optimum (other than the global one) is almost inevitably connected by short paths, without any intermediate points of much lower fitness, to other better regions — hyper-spatial bypasses.

In the n-dimensional sequence space, defined by binary genotypes of length n where Hamming-neighbours are connected, suppose that through mutation points up to Hamming-distance d from the current position can be sampled. There are $\mathcal{M}(d,n) = \sum_{i=1}^{d} n!/i!(n-i)!$ of these. For the current position to be effectively a local optimum from which escape is impossible, *all* of these points must be less fit. But with increase in n, increase in $\mathcal{M}(d,n)$

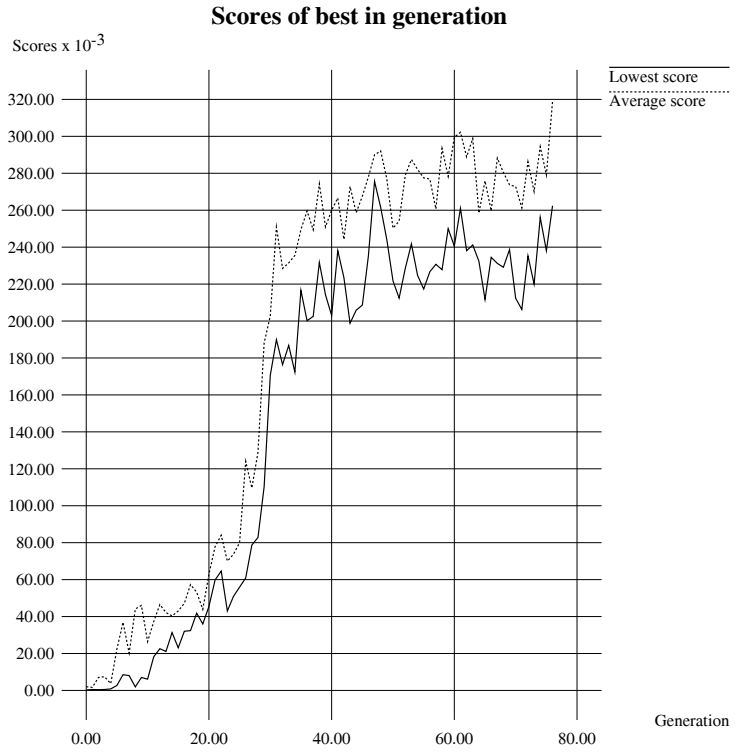

Figure 8.9: *The scores of the best member of each generation. Shown are the average score that the best member achieved over 8 (noisy) trials; and the lowest score of its 8 trials — it is this figure that is used as the evaluation.*

is roughly $\mathcal{O}(n^d)$; the higher the dimension, the more hyper-spatial bypasses there are.

8.9 Convergence and drift in a Species of Evolved Robots

This section looks at these issues of convergence and drift, taking a particular experimental run, of some 77 generations with a population of 60, at the task which will be described in a later chapter, Section 12.4.1.

The starting population was randomly initialised, and only the single task was used for evaluation; so there has not yet been any attempt to increase the complexity of the task over time. Selection was rank-based.

Measuring genetic convergence in a population with varying lengths is non-trivial, even though the lengths in general remain nearly equal. A number (here 8) of pairs of genotypes were selected at random, and for each pair the longest common subsequence (LCSS) was calculated (Hirschberg 1975). Convergence for this pair was taken to be the length of the LCSS divided by the average length of the pair; population convergence was taken to be the average convergence of those pairs sampled.

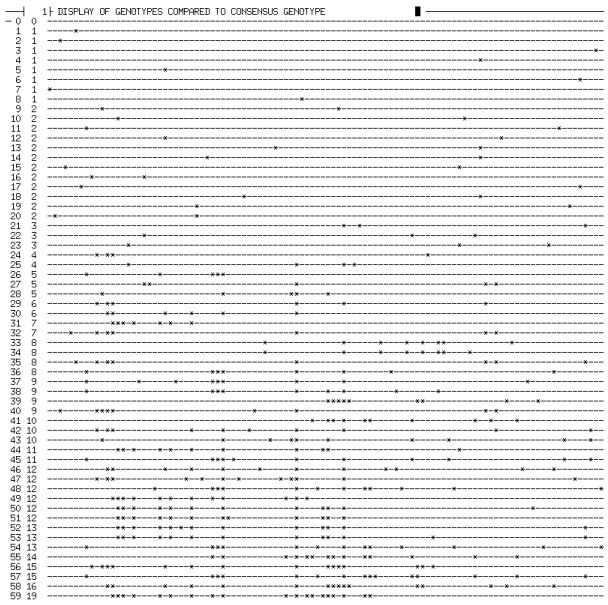

Figure 8.10: *All 60 genotypes in the 76th generation are listed according to their differences from the consensus sequence. Those 224 loci on the genotype with 100% agreement are ignored, the others displayed as '-' where they agree with the consensus, '*' where they differ. They are ordered in terms of Hamming distance from the consensus, distances shown on left.*

The population converged to around 95% after only some 20 generations (Fig. 8.8), driven by the strong selection, even though absolute scores were low. The sharp rise in fitness around 30 generations, followed by a prolonged gradual improvement, occurred *after* this degree of convergence had already been reached (Fig. 8.9).

The run was stopped arbitrarily after 77 generations. At this stage, all the genotypes were the same length except for one minimally shorter and one minimally longer. These two were 'edited' to conform in length, and the whole population converted into binary format, which was then 330 bits long. In 224 of these places there was 100% convergence. These identical values were discounted, the consensus sequence calculated (a sequence with the most popular value for each position), and the population of genotypes displayed in terms of their difference from the consensus sequence (Fig. 8.10). In this figure they are listed in order of Hamming distance from the consensus sequence, these distances ranging from 0 to 19. In fact the consensus sequence itself was present in this particular population, though in the general case this need not be so. The first 21 so listed are at a maximum Hamming distance of 2 from the consensus, indicating a tight cluster.

Some correlation is distinguishable by eye amongst those further from the consensus.

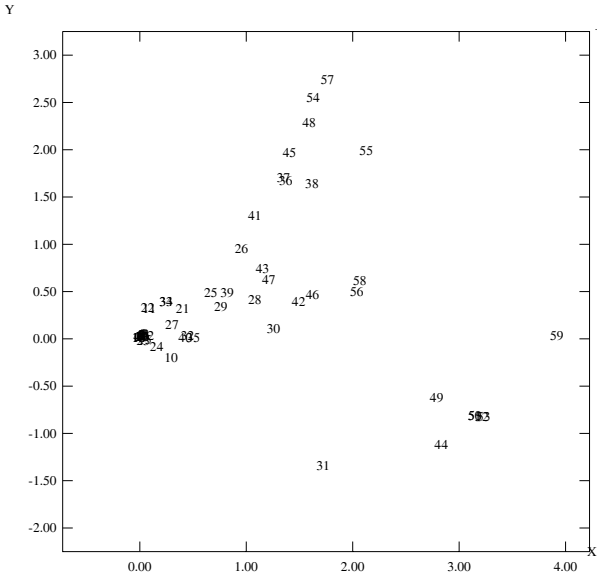

Figure 8.11: *A principal components analysis of all 60 members of the population in the 76th generation. The numbering here is in order of distance from the consensus sequence, itself numbered 0, hidden in the cluster at the origin, (0,0).*

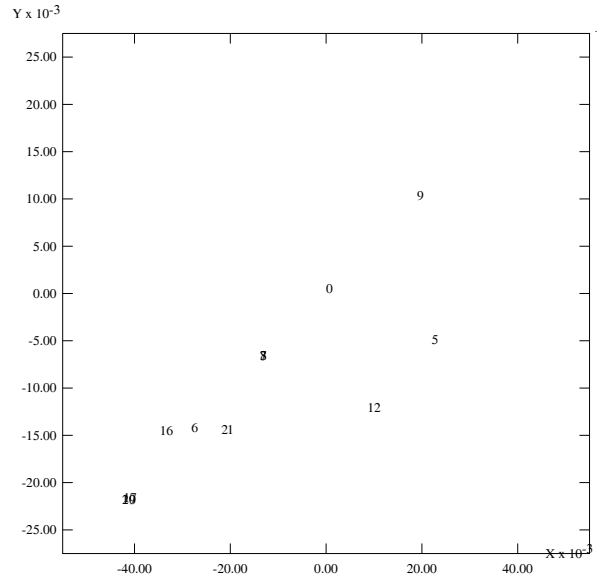

Figure 8.12: *Focusing on the central group shown in the previous figure — both axes are now in units of* 10^{-3}.

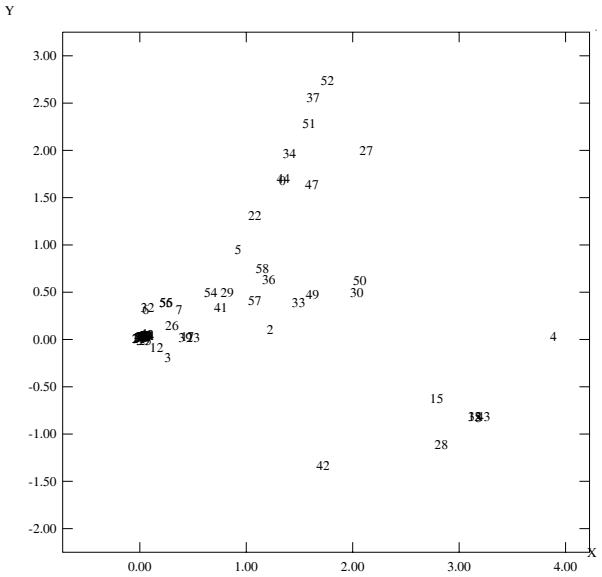

Figure 8.13: *This time the same final population is numbered in order of scores, 0 being the best. This best is contained within the 'ridge' towards the north-north-east, far from the central cluster. Another high-scorer, 4, is isolated towards the east.*

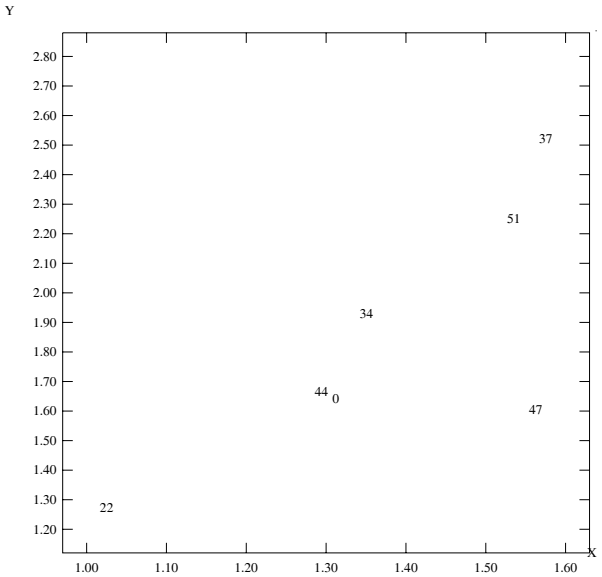

Figure 8.14: *Focusing on the north-eastern 'ridge' reveals the best scorer 0, amongst some relatively indifferent ones.*

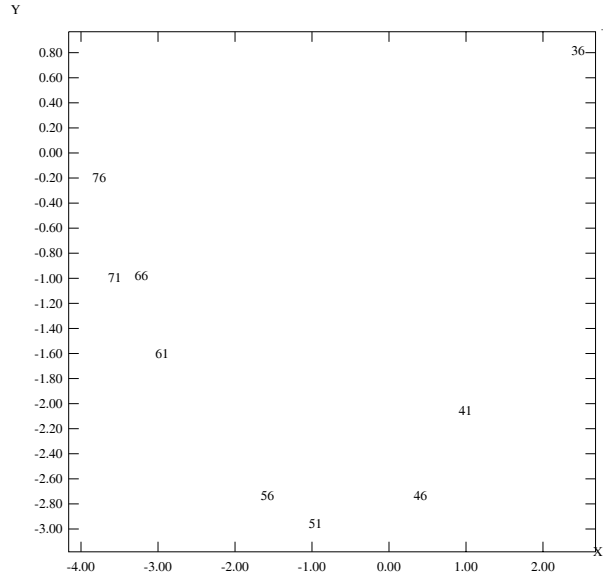

Figure 8.15: *A principal components analysis of the top scorers in every 5th generation from 36 to 76.*

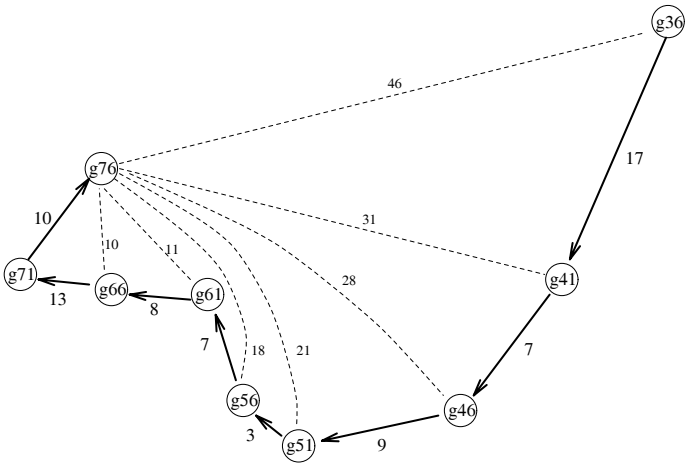

Figure 8.16: *The same top scorers are shown with their Hamming distances both to the next-displayed one (solid lines), and to the one displayed for generation 76 (dashed).*

A principal components analysis (PCA), of the first and second components, is shown in Fig. 8.11, confirming the presence of a strong correlation. The central cluster is shown in more detail in Fig. 8.12, and includes those nearest the consensus. The PCA is here used as a convenient tool to give a 2-dimensional snapshot of what here is a 106-dimensional space. For the first component, a vector is chosen though the consensus sequence in the direction such as to maximise the variation in the projections of all the points onto this vector; the second component is an orthogonal vector chosen so as to maximise the remaining variation. The figure shows what could, if the fitnesses were relatively high, be considered as a 'ridge' to the north-north-east of the centre of this graph.

By numbering the individuals within the population according to their ranking, it can be seen that the current winner is in fact within this 'ridge', and hence will be contributing to a larger cluster around there when it is preferentially copied in large numbers in the next generation.

The history of the population was recorded from generation 36, long after it had converged, every 5 generations until the 76th generation. A PCA of the best of each of these generations, Fig. 8.15, shows a continuous trajectory. The Hamming distances of the best in these generations from the consensus sequence of the last, 76th, generation, are shown in Fig. 8.16. This figure also shows the distances moved by the best-of-generation every 5 generations, and it can be seen that relatively large distances can be traversed across the sequence space, despite the high degree of convergence. It should be borne in mind that at this stage, with genotypes of around 330 bits, no two points in the sequence space are further apart than 330. Thus it can be seen that the early genetic convergence is no barrier to movement across sequence space, and possible continued improvement.

If a population was 'trapped' in sequence space at a local optimum, then this implies no further movement through sequence space. Where movement is possible through neutral drift, the theoretical rate of movement is easily calculated, for genotypes long enough for back-mutation to be insignificant. First consider an asexual population. Through the workings of random genetic drift, inevitably in the long-term the whole population will be descended from just a single member of the population of time t_n. Hence of all the new mutations that appeared in generation t_n, only those from a single member will survive in the long-term. For a mutation rate of m bits per genotype, the expected number of surviving mutations from generation t_n is exactly m, regardless of population size; similarly for generation t_{n+1}, t_{n+2} .... So in the asexual case the expected rate of movement through

sequence space is m per generation, if neutral drift is the mechanism. The argument can be extended similarly where there is recombination, by considering the long-term survival rate of mutations at each locus, rather than within each genotype; exactly the same rate results. Since for our example $m = 0.9$, the movement shown in Figure 8.16 accords extremely closely with this theoretical prediction. Of course comparable distances can be moved by positive selection also.

8.10 Sexual Selection

Sexual selection is an aspect of evolution which, though mentioned by Darwin from the beginning (Darwin 1859), has not received as much notice as natural selection. Recently Miller and Todd have drawn attention to the possible significance of sexual selection in the fields of Artificial Life and, indeed, engineering design (Miller and Todd 1993). Two particular mechanisms are of interest.

The first one is assortative mating, which means that in choosing a mate for recombination preference is given to those members of the population closest in sequence space. This is obviously easy to implement. The expected advantage, in comparison with random choice of mates, is that there will be less tendency for offspring to be positioned in sequence space near the centre of the species or quasi-species, and more tendency for desirable, high-value ridges to be explored further. I have made some preliminary experiments to test this hypothesis in an abstract NK fitness landscape, with as yet no confirmation; this seems worth further investigation.

The second mechanism is that of runaway sexual selection, the sort of evolutionary dynamics that is thought to have made peacock's tails so large despite being a physical handicap to movement. There are arguments, discussed in (Miller and Todd 1993) that in the natural world, many mate preferences in practice pick out viability indicators and hence reinforce natural selection. Further, that since the brightness of plumage depends (roughly) on the amount of energy left after a bird has gone through its daily chores necessary for survival, such brightness magnifies any variance in fitness. Yet a further point raised: sexual selection can allow spontaneous sympatric speciation (Todd and Miller 1991), which could allow exploration of more than one mountain range in the fitness landscape.

Runaway sexual selection in the natural world can be thought of as a macro-evolutionary equivalent of mutation at the micro-evolutionary level. Where neutral drift is possible —

and preceding sections have argued that it is — then sexual selection in a world of artificial evolution would indeed be a driving force to encourage wider and faster exploration of sequence space. It can, of course, be considered as one manifestation of a Red Queen phenomenon. It is, I feel, too early to know whether one should advocate some functional equivalent of sexual selection in the evolution of robots, and how one would promote or encourage it; clearly it is a potentially important issue which should be pursued.

8.11 Conclusions

Having first been drawn to an analysis of genetic drift as an explanation of an anomaly in the Hinton & Nowlan paper, an analysis through the diffusion equation has been given. The RNA model has been used as an example of how neutral drift can, given the imbalance in size between phenotype and genotype spaces, allow movement through sequence space regardless of selective forces. In other words, the metaphor of a population being clustered on a local hilltop may well be misleading in a high-dimensional space.

Analysis of a particular experimental run with a population of robot control architectures shows that despite the genetic convergence, mutation is a sufficiently powerful force for genetic movement along 'ridges' to potentially fitter regions. Principal components analysis has been introduced as a useful visual tool for analysing the movement of populations across sequence space. Finally, under the heading of sexual selection, assortative mating has been discussed as something easy to implement, with benefits that have yet to be tested in practice. Runaway sexual selection is more problematical, but is worthy of further consideration.

CHAPTER 9

Crossover[1]

9.1 Introduction

Genetic Algorithms (GAs) have traditionally tended to use genotypes of a predetermined fixed length. The designer of a particular GA, for use as an optimisation technique within a given search space, decides which parameters are to be represented on the genotype, how they are to be coded, and hence the genotype length. For each parameter there is a given position or set of positions on the genotype which unambiguously code for it. This can be loosely translated as: the allele (parameter or feature value) for a particular gene (parameter or feature) is coded for at a particular locus (genotype position). This makes it simple for a recombination genetic operator, therefore, to take the same crossover point in each parent genotype, and exchange homologous segments.

When variable-length genotypes (VLGs) are used, absolute position of some symbols on the genotype can usually no longer be used to decide what feature those symbols relate to. Some examples of ways around this problem are given in the next section. A related problem is, how can one organise a recombination operator so that the resulting offspring genotypes are, firstly, sensibly interpretable, and secondly, have inherited meaningful 'building blocks' from both parents.

VLG GAs have been proposed in various domains where they seem to allow a natural genetic representation for the problem under consideration, and the variety of domains is reflected in the variety of representations suggested. In this chapter the motivation for needing VLGs is that of wanting to extend GAs so as to allow for open-ended evolution. Although GAs have borrowed ideas from natural evolution to use in function optimisation, what they have ignored is perhaps the most impressive feature of natural evolution: how over aeons organisms have evolved from relatively simple forms to ever more complex ones, with associated increase in genotype lengths.

[1] This chapter was first published in similar form as (Harvey 1992a).

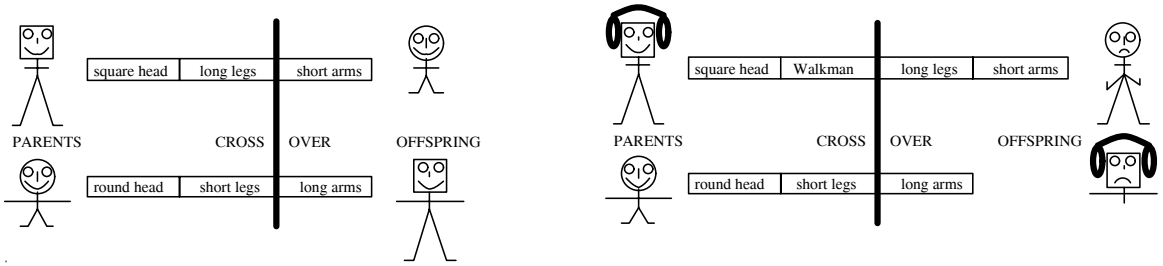

Figure 9.1: *A crossover operator which works well with fixed lengths may have sad consequences when unthinkingly applied to variable length genotypes.*

It will be suggested that in this context the identification of the locus of a 'gene', or that section of a genotype which codes for some particular feature, will necessarily be by use of an identifying template. The problem for recombination becomes then, given a randomly selected crossover point in one parent genotype, how to identify an appropriate place to break the other parent genotype so as to exchange homologous sections as far as is possible. In this we will be aided by the fact that within the SAGA framework the genetic pool of a population will be largely converged to form a *species* or *quasi-species*.

As a matter of practical concern, therefore, an algorithm needs to be developed which can determine on 'syntactic' grounds rather than 'semantic' ones how to exchange homologous segments. This can be quantified as maximising the similarity (under some appropriate measure) of the segments exchanged. This is of course a problem which nature, at the level of molecular biology, has found its own method of tackling, so an investigation of the relevant literature is suggested. It turns out that molecular biologists have developed algorithms for their own rather different, but related purposes. They are interested in quantifying on 'syntactic' grounds the similarities between two given nucleotide or amino-acid sequences, and doing so with computational efficiency, and it turns out that their algorithms can be adapted and extended for our present purposes. The method of doing so will be here presented; C code for implementing this is given in Appendix F.

9.2 Examples of Variable-length systems

VLGs have been proposed for a number of purposes, e.g. Smith's LS-1 classifiers (Smith 1980), Koza's Genetic Programming (Koza 1990, Koza 1992b), Goldberg's Messy GAs (Goldberg *et al.* 1990), Harp and Samad's genetic synthesis of neural network architectures (Harp and Samad 1991). Care needs to be taken that a crossover operation exchanges meaningful building blocks. In the case of LS-1 this is relatively simple, as a genotype is

effectively a list of rules each coded as a fixed-length string. The number of such rules is not fixed, and the ordering of them on the genotype has no significance; hence provided that a crossover exchanges homologous sections of an individual rule, the resulting offspring genotype can still be interpreted sensibly.

In Koza's Genetic Programming, the genotype is interpreted as LISP S-expressions, which can be depicted as rooted point-labeled trees with ordered branches. This allows a recombination operator to swap complete sub-trees between parents. The result is syntactically sensible, and preserves and transmits the building blocks that the sub-trees effectively constitute. This solution relies on the hierarchical tree-decomposition of the genotype, and would not extend to genotype representations where the interactions between 'building blocks' cannot be so decomposed.

In Goldberg's Messy GAs, each locus on the genotype in effect carries its identification tag around with it. Instead of a crossover operator, *cut* and *splice* operators are used, which allow genotypes of any length to develop over time. But the number of loci is fixed at the start, and everything is in effect based on an underlying fixed-length representation, which may be underspecified or over-specified. In the former case, where a genotype does not contain an allele for every locus, the deficiencies are made up by a 'competitive template' scheme. In the latter case, conflicts can arise where the identity tag for a specific locus occurs more than once with different associated values; in this case an arbitrary rule is used, such as choosing the one nearest a specified end of the genotype. It should be noted that the solution to the under-specification problem relies on there being a predetermined number of loci, and cannot be extended to arbitrary numbers of loci.

Harp and Samad (Harp and Samad 1991) use a linear genotype to code for a neural network by having building blocks of a fixed length on the genotype code for the specification of an individual layer in the network, including the connectivity from that layer to other layers. The format for each block is the same, but the number of them is not fixed. A crossover operator can therefore be used which, when a crossover point in one parent genotype occurs inside a block, ensures that the crossover in the other parent genotype occurs in the same position within a block, thus exchanging homologous segments. But unlike Koza's system, the interactions between layers in the network, represented by blocks on the genotype, are not restricted to a hierarchical organisation. So a method must be found for coding within each block so as to identify the other blocks representing network layers projected onto. The solution is adopted of giving each block an ID code,

and having two forms of addressing. The first is by absolute address, where the ID (for the block being projected to) is explicitly listed on the genotype; the alternative is by relative addressing, where a relative address is indicated which specifies a block by its position relative to the block being projected from.

The explicit purpose of these alternative forms of addressing is to allow relationships between blocks (and hence projections between layers in the neural network) to develop and be sustained and generalised across generations. Absolute addressing allows a target block to be identified no matter where it finishes up in a genotype in later generations; relative addressing allows groups of blocks close together on the genotype to maintain their mutual interactions. A version of this has been used in the Evolutionary Robotics work reported on in Chapter 12.

It can be seen that Harp and Samad's approach avoids the restrictions inherent in Smith's, Koza's and Messy GAs. Nevertheless there remains one restriction which prevents it being satisfactorily used as it is for genotypes of completely unlimited length. The number of bits on the genotype that code for the IDs, and for either absolute or relative addresses being referred to, must be pre-specified. Whereas for instance 4 bits might seem adequate (and 8 bits more than adequate) for genotypes coding for networks with less than 10 layers, for eventually 500 layers or more it would become inadequate. This will of course seem a practically irrelevant restriction to those who know the computational requirements of a network with many layers.

Nevertheless, both from a purist perspective, and from a practical perspective when the building blocks are not layers in a network but some smaller design primitives for a system being constructed, there are reasons for wanting to solve this addressing problem satisfactorily for arbitrary numbers of addresses. In particular this is so within the SAGA framework.

9.3 Template Addressing

Harp and Samad's addressing system was limited to addresses of fixed size. The obvious extension is to allow addresses of unlimited length. Assume that any building block on the genotype has an ID string at one end to identify it; and where within another block reference needs to be made to the first ID string (so as to identify a projection or interaction between the two features or modules coded for by the two blocks) this reference is coded for by a reference string. Both ID string and reference string can be of any length, and it

no longer is necessary to think of them as ID *numbers*, but instead as strings that need to be somehow matched. In molecular biology such matching of nucleotide strings can be done through the physical interactions between them.

An initial solution would be to use the identical string for both the address marker and the reference marker:

```
ID1 code1 ID2 code2 ref-ID1 ID3 code3 ref-ID1
```

But here the string for ID1 appears three times, and when a call is made to ID1 a simple string search for the string will not know which of the three to find; unless some additional code, or some transformation of the ID, is used to distinguish the two uses. In binary code a simple transformation would be to use the inverse as a template. Ray was probably the first to propose using template matching as a system of addressing, based on molecular biology, in the context of a synthetic evolutionary system (Ray 1992). Other equivalent methods can be devised to tackle the problem.

Thus we could implement absolute addressing by means of ID strings, or templates, of any size; indeed some system equivalent to this is necessary. Relative addressing for relative jumps coded by a string of arbitrary length is also easy to implement, but is no longer necessary. A form of template addressing is both necessary and sufficient to have the addressing power of Harp and Samad's system extended to genotypes of arbitrary length.

9.4 Where to cross

When evolving systems of arbitrary increasing complexity within the SAGA framework, it will be assumed that there are building blocks coded for along a linear genotype, and that interactions between such building blocks are mediated by some addressing system, as discussed earlier. For recombination it will be relevant that the population will be largely converged; any two parent genotypes will be broadly similar.

The SAGA cross has therefore the requirements that, given any chosen crossover point in one parent genotype, a crossover point in the other parent genotype needs to be chosen so as to minimise the differences between the swapped segments. This can be rephrased as: we should maximise the similarities between the two left segments that are swapped, and between the two right segments that are swapped. Please note that the VLG crossover problem that the SAGA cross handles *only* refers to the choice of the second complementary crossover.

The similarity has to be based on 'syntactic' measures (the characters in the genotype) rather than 'semantic' (the significance of such characters when expressed in the phenotype), so can only be based on equality of symbols, where the genotype is considered as a string of symbols. The ordering of symbols is also relevant. This leads to the use as a measure of similarity of the longest common subsequence (LCSS).

Algorithms for efficiently computing this have been developed for quantifying similarities between two given nucleotide sequences, starting with the Needleman and Wunsch algorithm (Needleman and Wunsch 1970, Sankoff 1972); and for the problem of how many editing operations are needed to change one string to another (Wagner and Fischer 1974). Here a method will be presented of using the algorithm to solve the VLG crossover problem. The starting place will be Hirschberg's exposition (Hirschberg 1975).

Hirschberg defines an 'Algorithm B' which accepts as input strings A_{1m} and B_{1n} of lengths m and n; and produces as output vector L_{0n} of length $(n+1)$. L_j will contain the length of the LCSS of string A_{1m} and substring B_{1j}. An array of size $2(n+1)$ is used for intermediate calculations, $K_{01,0n}$.

ALGB(m,n,A,B,L)

1. Initialisation: $K(1,j) \leftarrow 0\ [j = 0 \cdots n]$;
2. **for** $i \leftarrow 1$ **to** m **do**

   begin
3. $K(0,j) \leftarrow K(1,j)\ [j = 0 \cdots n]$;
4. **for** $j \leftarrow 1$ **to** n **do**

   if $A(i) = B(j)$ **then**

   $K(1,j) \leftarrow K(0,j-1) + 1$

   else

   $K(1,j) \leftarrow \max\{K(1,j-1), K(0,j)\}$;

   end
5. $L(j) \leftarrow K(1,j)\ [j = 0 \cdots n]$

The rationale behind this algorithm is as follows: The length of the longest common subsequence of two strings A_{1i} and B_{1j} is to be written into $L(i,j)$. If $L(i-1,j-1)$, $L(i,j-1)$ and $L(i-1,j)$ are known, then $L(i,j)$ can be derived from them, the value depending also on whether or not the i^{th} symbol of A and the j^{th} symbol of B match.

$L(i,j)$ must be at least equal to the best of $L(i-1,j)$ and $L(i,j-1)$; and if the symbols do match, then $L(i,j)$ will be one better than $L(i-1,j-1)$. Algorithm B keeps track of the necessary amounts, and updates them within the j and i loops.

In Hirschberg's development, a further algorithm C is used to recursively use Algorithm B, by dividing a given problem into two smaller problems, bottoming out of the recursion when there are trivial subproblems. This is used to output the sequence which is the LCSS of A and B. My purposes here are rather different, and I have developed an algorithm D to solve the VLG crossover problem.

The initial step is to add a feature to algorithm B so that it will work with substrings, and equally well when comparing two strings enumerated from one end or from the other end. For this it is necessary to explicitly pass as inputs the initial and final indices for the substrings of A and B.

Algorithm D accepts input strings A_{1m} and B_{1n} of lengths m and n, and an integer c which represents the crossover point in A. As output it returns a vector M which keeps track of the current best-so-far candidates for a crossover point in B (which may be one point or a sequence of them). For intermediate calculations two vectors $L1(n+1)$, $L2(n+1)$ are used, which contain the outputs from two separate calls to algorithm B. Internal integer variables r, s and t are used respectively as the current best score, the number currently equal to the best-so-far, and a temporary store.

ALGD(m,n,A,B,c,M)

1. $ALGB(1, c, 1, n, A, B, L1)$
2. $ALGB(m, c-1, n, 1, A, B, L2)$
3. $r \leftarrow 0; s \leftarrow 0;$
4. **for** $i \leftarrow 1$ **to** $n+1$ **do**

   begin
5. $t \leftarrow L1(i) + L2(n-i)$
6. **if** $t > r$ **then**

   $s \leftarrow 0; r \leftarrow t;$
7. **if** $t = r$ **then**

   $M(s) \leftarrow i; s \leftarrow s+1;$
8. **end**

The rationale behind this algorithm is:

In line 1, algorithm B is applied to the 'left-hand' substring of A, from the start up to the crossover point, and to the whole of string B. The result is output in $L1$.

In line 2, algorithm B is applied to the right-hand substring of A, and to the whole of B, but treating each string in reverse order, starting from the right-hand ends. The result is output in $L2$. Since increasing the length of one of a pair of strings can only either

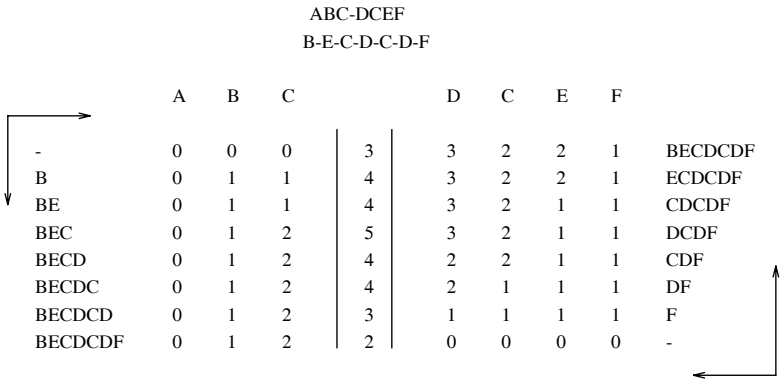

Figure 9.2: *Algorithm D on ABC-DCEF (cross between C and D) and BECDCDF (best cross to be determined). On left, algorithm B gives in column C best scores matching substrings against ABC. On right, working backwards, best scores in column D. Central column shows best total (5 matches) given by splitting BEC-DCDF.*

retain or increase the length of the LCSS, both $L1$ and $L2$ have this property.

The loop started in line 4 places, for each possible cross point i in B, the sum of LCSSs for left and right segments into a variable t. As i increases the value of t will each time either increase or remain steady, until it reaches a peak value or a plateau; thereafter t will decrease with occasional level stretches.

The purpose of lines 6 and 7 is to monitor this, and to store in M the values of i for the current best, or several best-equal, values of t. Hence when the loop finishes, the first s values in M contain the proposed crossover positions for maximising t, the sum of left and right LCSSs.

It is then possible to select at random one of the optimal positions, and return this as the proposed crossover point. The C code, given in Appendix F, also economises on memory; rather than using a separate array M, there is enough space to keep track of the optima as we go in array $L1$, since we will never overtake what we are reading from $L1$ with what we are writing into it. An application of the algorithm is shown in Figure 9.2.

9.5 Two Point Crossover

Only one-point crossover has been considered here. This has the feature that building blocks near each end of one genotype are much more likely to get separated than ones nearer the middle. For some purposes it may be better to use a two point crossover which avoids this bias. Two crossover points are chosen at random in one parent genotype, and two complementary points need to be selected in the other parent genotype; the offspring

are made by swapping the middle sections of each parent.

The present algorithm can be extended to handle this, or indeed multiple-point crossovers, by adding further outer loops.

9.6 Computational requirements

This algorithm requires for one-point crossover memory space of order $(m + n)$ where m and n are the lengths of the two genotypes. The main loops are in algorithm B, and the time requirements are of order (mn). The time taken is independent of the similarity or otherwise of the two genotypes. The only test on symbols on the genotype is for equality, so whether the genotype uses a binary alphabet or any larger one makes no difference. On a reasonably loaded Sun4 using two genotypes each of length 1000 characters, approximately one second is needed.

In GAs the computational requirements for fitness-evaluation generally far outweigh those for genetic operations, and this could also be expected for any system which needed genotypes of this length.

9.7 Conclusions

Both biological evolutionary theory, and GA theory, rely on the notion of genes or building blocks being expressed in compact sections on the genotype. For inter-relationships between such building blocks to be incorporated, a method of identification is needed, and this issue comes particularly to the fore when crossover is considered. For fixed length GAs the issue can be solved by identification being implicit in position on the genotype, but not in general when genotype lengths are variable.

Addressing methods for several variable length genotype GAs have been surveyed, and their restrictions noted. Harp and Samad's approach avoids many of these restrictions, but nevertheless does not extend immediately to genotypes of completely arbitrary length. It has been suggested that for present purposes some form of template addressing will be both necessary and sufficient.

A recombination operator needs to be designed that, given any crossover point on one parent genotype, can choose a complementary crossover on the other parent, when the genotypes are of arbitrary, differing, length. The choice must be made on purely 'syntactic' grounds; i.e. through operations solely on the symbols of the genotype, not on their interpretation. Nevertheless the crossover must exchange homologous segments as

far as is possible. The fact that in a SAGA system, of gradually increasing complexity and gradually increasing genotype lengths, the population will be largely converged, means that there will be a high degree of similarity between two parent genotypes.

It should be emphasised here that this crossover operator is completely impartial as to where the *initial* choice of a cross on the *first* parent genotype is; it merely then selects where to cross on the *second* parent genotype. The first cross may, for instance, be deliberately chosen for different reasons to be more or less disruptive of schemata (De Jong and Spears 1990). The SAGA cross is only concerned with the complementary crossover; this is a problem which conventional GA practice never has to face, as with fixed-length genotypes the complementary position is trivially obvious.

Building on algorithms developed for the Longest Common Subsequence problem, a novel algorithm has been presented which allows a random crossover point in one parent genotype to be optimally matched by a specified crossover point (if relevant, a restricted range of possible points) in the other parent genotype. The criterion for optimality is well-defined in syntactic terms, being that of maximising the sum of the length of the LCSS in the left-hand segments and the length of the LCSS in the right-hand segments. The algorithm is computationally efficient.

CHAPTER 10

Development

10.1 From Genotype to Phenotype

In this context 'development' is used in the biological sense of morphogenesis, the process by which an embryo develops to become an adult. This happens under the control, or perhaps more carefully stated under the *influence* of the genotype; there are many more factors such as the immediate cellular environment that are also relevant. Here, for the most part I will be ignoring such factors, and considering that in artificial evolution the genotype alone is sufficient to specify the phenotype; subject to two caveats mentioned in the next two sections.

10.2 Order for free

The first caveat is the theory put forward by Kauffman (Kauffman 1993, Kauffman 1989) (and similar views held by Goodwin (1990) and others), that in complex systems with many interacting parts there are elements of large-scale order that, so to speak, come for free, without the necessity for selection to produce such order by acting on the phenotypic expressions of genotypic variations. To take an early example of Kauffman's, with his work on random boolean networks (Kauffman 1974) where he interprets 'genes' as boolean switches that turn on and off other genes, he demonstrated that (subject to some constraints on the number of other genes which one individual gene affects) there was a surprisingly small number of basins of attraction. With g binary genes there are 2^g possible states of the system, yet he shows that there are generally only something of the order of $\sqrt{g}$ basins of attraction. Combining this model of genes as boolean switches interconnected in a network with a model of cells as basins of attraction of such recurrent networks, this could be an explanation of why animals have such a small number of cell-types in comparison with the number of their genes.

Another way of looking at this result — and this thought underlies much of Kauffman's work — is that in large complex systems very often 'order comes for free'. The relatively small number of cell types implies that the overall organisation of the system can be described much more simply than if the number of cell-types was closer to the number of possible internal states of the cell, or random boolean network. Without this result it might be thought that such simplicity of organisation could only have arisen by strong *selection* for simplicity; this result implies that it can arise even without selection.

10.3 Neural Darwinism

In his writings on Neural Darwinism, e.g. (Edelman 1989), Edelman proposes a theory of brain function that follows selectional principles — this selection acts *within* individual ontogenetic development. The theory is called TNGS, for theory of neuronal group selection.

The first tenet of TNGS is that the anatomical pattern of neural connections is underspecified by the genotype, and hence the *actual* connections formed are determined by a selectional process involving populations of neurons engaged in competition. The resulting population of neural networks, known as the primary repertoire, does not have a specific wiring diagram specified by the genotype. Rather, the genetic code imposes a set of constraints on the selectional process.

The second tenet involves another selectional process acting through strengthening or weakening of populations of synapses as a result of behaviour, leading to the formation of a secondary repertoire of neuronal groups, or functioning circuits. The third tenet is that reentrant linking between different areas of the brain, in the form of maps, means that the selectional processes in any one such area influence the selectional processes in other areas.

The unit of selection in TNGS, according to Edelman, is a closely connected collection of cells, a neuronal group. Variation in directions in which neuronal connections are formed, or in strengths of synaptic connections, due to random environmental influences, can be seen as an analogue to variation in evolution. I have difficulty, however, in working out what, in his metaphor of neural Darwinism, is the equivalent to heredity. For this reason I am sympathetic to Francis Crick's comment, cited in (Edelman 1989) that "If a term has to be used for the whole set of ideas I would suggest Neural Edelmanism." Crick claims that is not possible to make a worthwhile comparison between the theory of

natural selection and what happens in the developing brain.

Despite this scepticism I share about TNGS, in the present context it serves to highlight the multifarious possible processes that can take place during development, for which genetic influences are merely one constraint among many. At some time in the future it may well be appropriate to incorporate comparable processes into artificial morphogenesis.

10.4 Coding for artificial neural networks

In biology, morphogenesis is one of the least-understood areas. In the artificial evolution of neural networks we know of no very satisfactory method for modeling a developmental process from genotype to network, and this currently presents the greatest challenge. For the time being *ad hoc* solutions must be used.

Of course, even with variable-length genotypes the genotype can still be in effect a straight description of the phenotype. If the network that forms the control system in the evolved robots to be described in Chapter 12 is considered as the phenotype, then this is true here also. There is a case to be argued, however, that the phenotype here on which selection operates is the *behaviour* of the robot; in which case the relationship with the genotype is far more complex.

The information in the DNA is too small by many orders of magnitude to uniquely specify each connection in the brain, and hence there must be some form of modularity in this specification. One presumes that there are building blocks (for instance, layers with a certain pattern of projection for the connectivity between them) that are useful in many different parts of the brain; rather than each occurrence being individually coded for in the genotype, such a building block may be coded for in perhaps just one place, and this information used many times over in the development of the brain structure.

10.4.1 Mjolsness

One method for compactly defining hierarchically structure networks is the Recursive Learning Network formalism (Mjolsness *et al.* 1987), which was not originally proposed in the context of GAs, although simulated annealing was used. A major part of the rationale for this formalism was that it allows networks that have, through some learning technique, achieved success on a small scale problem to be automatically generalised so that they can be applied at a larger scale to a scaled-up version of the problem.

In this formalism, the connectivity of a network is expressed initially in terms of a

connection matrix, with 1s and 0s at each place in the matrix denoting the connection, or absence of connection, from the node represented by row-number to that represented by column-number. Any uniformities at a large scale within the matrix, such as repetition of quadrants, can be expressed by means of templates; and this process can be recursively applied to such smaller regions in turn. With the additional notion of a *lineage tree*, there is no need for such sub-divisions to be restricted to regular shapes such as quadrants.

Through the recursive application of the templates as prescribed by the lineage tree, the formalism embodies the principle of divide-and-conquer. The summation makes it possible for several networks to be superimposed, a technique generally useful in network design.

The Recursive Learning Network formalism has great expressive power. It allows networks to contain arbitrary interconnections, including cycles. (Mjolsness *et al.* 1987), page 167.

10.4.2 Kitano

Kitano (Kitano 1990) uses the genotype to encode a graph generation grammar. The graph L-system is an extension of Lindenmayer's L-system (Lindenmayer 1971). Kitano's system allows a linear genotype to operate on a square matrix of characters, initially 1×1. Each act of rewriting expands the matrix into 4 quadrants each the same size as the previous matrix, with the contents of each quadrant specified by the genotype. At the end of a succession of n such rewriting steps, one finishes with a matrix of size $2^n \times 2^n$. The characters in this final matrix are interpreted as 1s, for a connection, or 0s for no connection, in a connection matrix specifying all connections in a network of 2^n neurons or nodes.

In this way scalability and modularity, as defined below, start to be implemented in a compact genetic encoding of large regular networks.

10.4.3 Gruau

Gruau (Gruau 1992, Gruau 1993) discusses Kitano's work, and also acknowledges earlier work by Wilson (Wilson 1987). He defines 7 properties of a genetic encoding of neural networks that should be considered. These are:

1. **Completeness**: Any NN architecture should be capable of being encoded.

2. **Compactness**: One encoding scheme is more compact than the second if for any NN architecture the first genetic encoding is shorter than that given by the second.

3. **Closure**: implies that any genotype encodes some architecture.

4. **Modularity**: A genetic encoding would be modular if parts of the genotype specify subnetworks of the complete network, and other parts specify the connections between such subnetworks. This decomposition could be recursive.

5. **Power of expression**: A NN encoding has a power of expression greater than (perhaps this should be *greater than or equal to*), for example "Turing machine language" if for every program P written with this language there exists a code for a NN that simulates the program P.

6. **Scalability**: this would be the property of an encoding scheme which could specify networks with part of the encoding specifying a scale parameter; such that small and large problems of the same type could be handled by small or large networks whose encoding differed only in respect of this scaling parameter.

7. **Abstraction**: an encoding would be abstract if it can describe a NN at a functional level, rather than at the level of specifying connections.

Gruau's proposed Cellular Encoding (CE) is a form of "neural network machine language", which he claims has all the above desirable properties. This is a form of rewriting grammar, where the rewriting is considered as a form of developmental process involving "rewriting neurons" or "cells". Rewriting operators include PAR which divides a cell into two cells that inherit the same input and output links as their parent; CLIP which can cut links; WAIT which delays rewriting operations so as to change the order in which later operations are carried out.

Further operators SPLIT and CLONE allow for the desirable property of modularity to be achieved. In total 13 operators are used for CE to achieve all seven listed desirable properties. The final property of abstraction allows a program written in a high level language, via this CE encoding, to be used to build a NN which efficiently simulates the given program.

It is in a sense because EC was probably designed with this sort of application in mind that I feel that, though it may well be appropriate for some purposes, it is not appropriate for the purposes of this thesis. The properties of *completeness*, *closure* and *modularity* are

indeed generally desirable, but the other properties listed are more appropriate for the 'instructionist' approach that a human designer must take in designing a network, than they are appropriate for the 'selectionist' approach.

10.5 What are the virtues of development?

In the context of artificial evolution, are there any good reasons why a control system, perhaps some sort of connectionist network, should be derived from a genotype through some analogue of a morphogenetic process? Yes, there are, and I shall reach the same conclusion through two different routes. The effect on evolution will be that, given the sorts of genetic operations that are likely to happen, particular types of phenotypic changes will be associated. Genetic operators to be considered are mutation, crossover, translocation or inversion, and gene duplication.

10.5.1 Symmetries and repetition

Just as organisms and robots tend to have physical symmetries, such as that between left and right, these are likely to be reflected in control systems also. If a genotype was in essence a straight description of a control system which contained similar sub-parts for the control of left and right sides, then evolution would have to allow through selection the genetic specification for each side to come about completely independently. If instead there was just a single 'description' of the necessary sub-part, which was then 'called' once for the left side and once for the right, in the sense that a sub-routine in a computer program can be called several times, then the evolutionary pathways that can reach this situation are much easier. For multiple repetition, rather than simple bilateral symmetry, this factor is of even more significance. When repeated general patterns within a layer of primate brains is seen, such as in early visual processing, the potential power of something equivalent to sub-routining in the genetic specification is apparent. The operations of gene duplication followed by mutation are likely to be very powerful in this context.

10.5.2 Useful brains first

Another way of viewing the same effect is to consider, for any given genotype-to-phenotype mapping, the ordering of phenotypes that corresponds to a strict ordering of genotypes in terms of length; within each possible length, all possible genotypes to be listed consecutively. *If* the genotypes were, for instance, straight descriptions of networks, then the

corresponding list of networks would be in order of size. Ordered networks of size N would appear at a similar place in the list to randomly wired up networks of the same size N. And networks that were broadly similar except that the second contained additional parts that were, for instance, multiple copies of the parts of the first, would be far apart on this listing.

Since incremental evolution necessarily means something like exploring individual points on the genotype list in something loosely approximating to sequential order (shorter genotypes will be explored before longer ones), this seems to imply that the list of networks, of phenotypes, will be explored in an intuitively unsatisfactory order. Large, complex, semi-organised networks can only be reached after networks of all possible lesser sizes have been explored.

10.6 A sketch of a proposal

Desirable characteristics for an artificial morphogenetic coding for a network are:

1. Genotypes of all lengths and any ordering of characters within the allowable alphabet should generate a valid phenotype (Gruau's 'Closure').

2. All valid networks should be coded for by some genotype ('Completeness).

3. Regularities in networks should be, in some sense, 'compactly encoded' ('Modularity').

4. 'Zen principles' should be respected.

The last item refers to informal intuitions expressed by Goldberg (Goldberg 1989a). Rather than trying to force the desired results, a more indirect approach should be taken of merely specifying ground conditions which allow the desired results to emerge. With these various points in mind, here is a sketch of a possible way to achieve this.

Much use will be made of biological theories, or simple caricatures thereof, to do with the morphogenesis of multi-cellular organisms. It should be remembered here that this is not an attempt to be faithful to biological processes, but rather to take inspiration from a partial understanding of them in order to produce engineering techniques. With this proviso in mind, to assist in visualisation in the sketch which follows concepts such as *cells, DNA, enzymes, peptides* and *polypeptides* will be used freely and no doubt, for a biologist, inaccurately.

10.6.1 Multicellularity

The underlying metaphor used in this proposal is that of the development of a multi-cellular organism by cell division from a single initial cell. Every cell contains the same DNA, the genotype, which acts as a constraint on an intra-cellular dynamics of transcription and translation of enzymes which themselves initiate or repress the production of further enzymes.

For any one cell, the intra-cellular dynamics are to a small but sometimes significant extent influenced by inter-cellular 'signals', enzymes which are received through the cell membrane from neighbouring cells. In turn, internal production of particular types of enzymes results in versions of these being exported as signals to neighbours. The production of particular enzymes initiates cell-splitting; other particular enzymes, when they are first produced, signal the completion of the developmental process.

After all the cells have completed their development, there will be a number, potentially large, of cells that can be considered as positioned in some space with neighbourhood relations. Although all containing the same DNA, the cells can be differentiated into different classes by the particular distinctive internal dynamics of their enzyme production process. Thus at this stage the whole group of cells can be interpreted as a structure with organisation; for instance, as a neural network with different cells being 'neurons' with specific characteristics, and with connections specified between them.

10.6.2 Dynamics within a cell

At any given time, as well as the fixed DNA — a long directed bit-string — within a cell, there will be a number of smaller directed bit-strings floating around; these can be thought of as polypeptide chains. These smaller bit-strings are translated, 3 bits at a time, into a different 8-letter alphabet from **A** to **H**. Any spare bits on the end that do not form a triplet are ignored. So here, in contrast to the 64 possible triplets in the natural 4-base code being collapsed through redundancy into coding for 20 peptides, we have a non-redundant coding via 8 possible triplets in a binary code, allowing for 8 peptides.

The polypeptide chain thus formed can be considered in two ways. Firstly, the symbol **A** to **H** at the head of the chain distinguishes what role this chain plays:

1. **A, B** are **start** polypeptides.

2. **C, D** are **stop** polypeptides

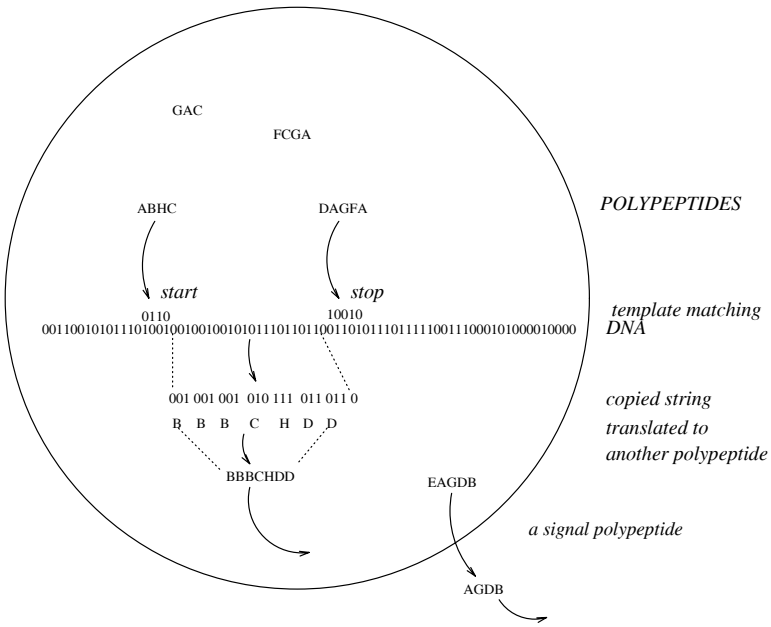

Figure 10.1: *The polypeptide chains floating around the cell, through template-matching with the DNA, promote transcription of further polypeptides.*

3. **E** is a **signal** polypeptide

4. **F** is a **tag** polypeptide

5. **G** is a **split** polypeptide

6. **H** is a **completion** polypeptide.

Secondly, the polypeptide can be re-interpreted as a **template** bit-string, by translating **A, C, E, G** as 0, and **B, D, F, H** as 1. This **template** is some third of the length of the original bit-string it was derived from via the two coding processes.

Those polypeptides distinguished by their head character as **starts** are passed along the DNA to see where they template-match; i.e., where parts of the DNA form the exact mirror-image of their **template**. At all such places a start-marker is placed. Similarly by template-matching, the **stop** polypeptides are used to place stop-markers. Any **signal** polypeptides have their distinguishing first character (**E**) stripped off, and are then passed out through the cell wall and into neighbouring cells, where they then participate in the internal dynamics of those cells.

Transcription can then take place, starting immediately after the template-identified start-markers on the DNA, and continuing to the first stop-marker encountered. In this way new polypeptide chains are created, which in their turn contribute to the internal

dynamics.

10.6.3 Cell splitting

If and when a **split** chain emerges within an individual cell, then the cell divides into two. The cells must be located in some space with a neighbourhood metric, so that splitting cells are next to each other, and in some fashion 'push aside' any surrounding cells that they are within. The neighbourhood relationships within this space determine where any **signal** polypeptides target themselves.

After splitting, both cells have the same DNA, but some arbitrary decision must be made in apportioning the polypeptides active in the parent cell between the two offspring cells. This allows for symmetry breaking and differentiation of cell dynamics.

10.6.4 After development

Individual cells cease further splitting when the first **completion** polypeptide is transcribed. There would need to be some constraints to deal with the very possible situation that no such polypeptide could ever arrive; for instance, a time-out mechanism could abandon further splitting after some arbitrary time, or after some notional 'energy' resource is exhausted[1].

Once the system does reach a multi-cellular structure with completion reached on all cells, there now starts the process of interpreting this structure. One possible way would be to interpret each cell as a 'neuron' in a neural network, with connections between them specified by **signal** polypeptides from the originating cell choosing target cells through template matching (using the early part of the polypeptide only) with **tag** polypeptides of the target. Other distinguishing characteristics of the neural network, such as neuron differentiation into input, output and hidden units, and weighting or delay times on the inter-neuron connections, are left (in this sketch of a proposal) as an exercise for the reader.

[1] This problem with potential infinite recursion is the same problem that has been commented on with relation to Koza's Genetic Programming, in Chapter 2. It is worth commenting that cancer cells in humans and animals are cells where cell-splitting carries on 'out of control', with harmful consequences. In a physical system, as opposed to a computational system, finite resources mean that such potential exponential growth is always at the expense of something else.

10.6.5 Assessment of the sketch

It is clear that there are many loose ends in this sketch, and quite possibly fatal flaws in, for instance, the combinatorics which would require a radical rethink. To produce a fully working method of artificial morphogenesis along lines such as this would be something of major significance.

The purpose of outlining this sketch here is to give a flavour of the sort of system that would satisfy the criteria for desirability listed above. All possible genotypes would code for a valid network. To show that any given network could be created by at least one genotype, it should be noted that template-matching with templates of arbitrary length allows for an arbitrary degree of specificity. Hence by hand-crafting the DNA such that all possible polypeptides created were long enough to form unique templates, and perform one specific function, one could expect to be able, at enormous computational cost, to translate this morphogenetic system into another one where the genotype was effectively a serial *description* of the phenotype. Since the serial description method can, in its cumbersome way, describe any network, this would also demonstrate the universality of the underlying morphogenetic system. In general one would not expect evolution to shape the DNA in such an explicit fashion; there is a tradeoff between explicitness and conciseness.

The third desirable criterion listed above is that regularities in networks should be in some sense 'compactly encoded'. The system sketched out above shows some similarities to — indeed its creation owes some debt to — Kauffman's work on Random Boolean Networks where attractors in the dynamics are treated as models of cells, discussed earlier in this chapter.

The final criterion of 'Zen principles' is clearly respected by the sort of morphogenetic system sketched out above. Given that there is a tradeoff between explicitness and conciseness, it is can be seen that the relationship between a phenotype network and its determining genotype will be opaque in the extreme. Although this would make the task of a human trying to hand-craft the genotype near-impossible (without the subversion of such a system to a cumbersome genotype-as-description method), this opacity need be no barrier to evolution. This point touches on the 'Central Dogma' of molecular biology, that genetic information can only flow in one direction: from DNA or nucleic acid to protein, but not from protein to nucleic acid. The real reason why Lamarckian processes, which rely on the possibility of such a reverse flow of information, are in practice unworkable is the practical impossibility of inverting any 'interesting' or non-trivial genotype-to-phenotype

developmental process. This still leaves, of course, Baldwin effects as mentioned in Chapter 8; the evolution of lifetime learning capabilities, coupled with cultural transmission or its analogues, can alter the dynamics of evolution in ways comparable to Lamarckianism, without contravening the Central Dogma.

10.7 Morphogenesis and SAGA

Though they have not yet been applied to such artificial morphogenetic systems, the SAGA framework has been from the start built up with the intention of such application. Variable-length genotypes fit in naturally with such a system, which can handle genotypes of arbitrary length in a simple, e.g. binary, alphabet. The interpretation of a genotype can be completely divorced from the genetic manipulation of such a genotype. Syntax can be divorced from Semantics, in the machinery of the GA.

If one characterises individual 'genes' on the genotype by identity and by function, then such genes are identified syntactically through template-matching; their function depends on how the transcribed polypeptide interacts with other genes, with other cells through signals, with cell splitting and completion, and with cell-identification by tags. The crossover mechanism described in Chapter 9 is explicitly designed for such conditions, operating in a genetically converged population, or species.

10.8 Conclusion

The perennial problem with any evolutionary algorithm is to try and have the genotype to phenotype mapping such that useful 'building blocks' can be conserved and encouraged to flourish. Desirable characteristics have been listed, and various proposals considered. All of them are either less than satisfactory, or incomplete.

Finding an appropriate method of morphogenesis is the biggest problem to be faced in the context of artificial evolution as proposed in this thesis. It ranks top in the list of 'future work to be done'.

CHAPTER 11

Applications — TSP

11.1 Introduction

At the start of this thesis it was suggested that the issues discussed here are *engineering* ones. In other words, what is proposed is intended as a practical method for attacking real-world problems. So after assembling the theoretical framework of the preceding chapters, it should be tested on some practical problems.

Normally a proposal for a new approach is tested in comparison with other approaches; typically for a class of problems there are benchmark tests allowing such comparisons. However, the approach advocated in this thesis, of incremental evolution, cannot simply be tested against any single benchmark; it is intended for application to a potentially open-ended sequence of problems of increasing complexity, and should be judged on that, rather than on any single problem.

I am not aware of any other engineering methodology which aims to do this sort of thing and which allow easy comparison. Similar incremental issues occur in the long-term development and maintenance of complex software systems, but not in a form which allows comparison with SAGA. The prime target for the work of this thesis is Evolutionary Robotics, combining the modified use of genetic algorithms associated with SAGA, together with the use of dynamic recurrent real-time neural nets as advocated in Chapter 5. Results in this area will be covered in the following two chapters, but although they will include a series of tasks of increasing complexity, this work has not yet reached the stage of needing significant long-term increases in genotype length.

Allowing for such long-term increases has been one of the guiding principles of the theory developed here. In Chapter 6 the NK model was used to test some of these ideas, but this is potentially open to the criticism that it may not reflect sufficiently accurately the difficulties presented by a real world problem. Hence in the present chapter SAGA principles are applied to a standard optimisation problem, the Travelling Salesperson Problem

(TSP). This domain simply and clearly allows for long-term increase in complexity of the problem, through the increase in the number of cities for which a tour is desired; and this is naturally associated with increase in genotype length.

11.2 TSP — the Travelling Salesperson

In this classic problem, the positions of a number of cities are given, usually on a two-dimensional plane, and the problem is to find the tour of minimal length that passes through each city once. The distances between any two cities may be calculated in terms of the direct straight line between them, or an alternative version assumes distances that do not directly correspond to this; in this version, all such distances must be specified rather than calculated from city positions, and there is the option of having the distance from A to B not equal to that from B to A.

This is an NP-complete problem, one of the class of problems that are in some theoretical sense equivalently difficult. Although a perfect solution to such a problem may be easy to spell out once found, there is no known way (and very possibly never will be) of reliably finding the best solution within a length of time that is any polynomial function of the size of the problem — in the case of TSP, the number of cities. As a benchmark for GAs it has two immediate advantages: the use of the tour itself as the genotype is a very natural one, and the evaluation of different tour lengths is computationally trivial and fast. But it has at least one disadvantage: a recombination operator combining two tours will in general result in an illegitimate tour, unless special precautions are taken. Much of the GA literature on TSP concerns different methods for recombination which avoid this problem.

The appropriateness of the TSP as a testbed for SAGA ideas of incremental evolution lies firstly in the easy way it lends itself to extending the lengths of tours; and secondly in the simplicity of calculation of evaluations, in what is nevertheless a very tough domain. Some sort of comparison can also be made with other GA approaches to a given TSP, subject to the qualifications above on this score of comparing like goals with like.

11.3 A SAGA approach

The usual N-city TSP problem, for a given number N cities, can be transformed into a form suitable for an incremental SAGA approach by considering the sequence of N-city problems as N increases from 1 to an indefinitely large number. The transition from

an N-city problem to an $(N+1)$-city problem (with the addition of one new city to an otherwise unaltered list of N) is a small increase of complexity; and one in which it could be expected that good solutions for the 'after' problem bear strong, but not necessarily always trivial, similarities to good solutions to the 'before' problem.

So for the purpose of testing out the SAGA approach, a predefined set of N cities was used, with specified inter-city distances. Initially 3 cities were selected at random, and a population of identical 3-city tours initiated the trial. Then an as-yet-unselected city was chosen at random, and added to all the members of the population, after which a tournament style GA was operated for a specified number of tournaments; thereafter a new city was selected, and thus the cycle was repeated until eventually all N cities were included in the tours.

On the addition of a city to all the members of the population, at the start of a cycle, one option could have been to add this in at a randomly chosen place within each tour. Instead, the decision was made to use some domain-specific knowledge, and add it in each tour at a position to one side or the other of a neighbour relatively close in distance. In practice, given a population size P, a current list of cities (in any order) L, and a new city to be added A, from L the cities were ranked in order of their distances from A, and the first $P/2$ closest were chosen. Then in each member tour of the population A was inserted to the left or the right of each of these closest cities. Given 2 sides to each of $P/2$ selected neighbours, this meant a potentially different point of insertion in each case; unless $P/2$ was greater than the length of the current list L, in which case some repetition was necessary.

The selection operation was through a tournament of size 2 to select the mother, a similar tournament to choose the father, both on the basis of lowest tour length; and a third tournament of size 2 to choose on the basis of highest tour length a member to be removed to make way for the single offspring. When a crossover operator was not applied, the offspring was derived solely from the mother, subject to possible mutation and translocation operators.

A mutation operator chose a single city at random, and moved it to another random position in the current tour. In use this operator was found to be too disruptive in comparison to a segment-inversion operator; this required the random choice of 2 cities in the current tour, and the reversal of the segment inbetween. For a crossover operator, the usual problems of illegitimate recombined tours loomed.

The SAGA cross described in Chapter 9 could not be applied directly, but something similar in spirit could. Firstly a segment is chosen at random from the mother. Then a search is made through the father tour for any segment of the same length that contains the identical cities to those in the mother segment, in any order. Only if such a segment is found is recombination allowed, which in these circumstances produces a legitimate tour for the offspring. In many attempts at recombination no such legitimate crossover could be found, and out of the legitimate crosses that were found, many resulted in swapping identical segments, and hence were still ineffective. Nevertheless, the fact that results with the use of the crossover operator differed from otherwise identical trials without crossover, indicated that there were some legitimate swaps which made a difference.

Using the SAGA approach with a generally homogeneous population means in any case that recombination does not have the major emphasis that is given to it in standard GA optimisation problems. Experimentation with different values in the two environments listed below showed that in many circumstances use of the segment-inversion operator alone, without any recombination, gave the best results.

11.4 Comparisons

Two TSP problems were taken from the literature, so that some comparison with results achieved by other methods could be made.

11.4.1 50 cities

In (Whitley *et al.* 1989) an edge recombination operator is proposed and implemented on, among other problems, a specific 50 city TSP problem originally specified by Eilon. Over 10 runs with a population size of 600 and a total of 25,000 recombinations which produce 25,000 new individuals, an average score of 439 with a best score of 428 was achieved. The GA used was Whitley's steady-state GENITOR (Whitley 1989b), using rank-based selection. For testing the SAGA approach I used the 50 cities listed in (Whitley *et al.* 1989), together with the method used there for calculating integral approximations to Euclidean distances between them.

Improvements to this recombination operator resulted in the Enhanced Edge Recombination in (Starkweather *et al.* 1991). There it is reported that on the same 50-city problem the optimal 428 city solution was found 30 times in 30 runs, using a population of 1000 and 30,000 recombinations.

	10 trials			100 trials		
crossover	0.0	0.5	1.0	0.0	0.5	1.0
average	460.2	462.4	456.2	461.8	457.7	459.1
std. dev.	8.28	15.51	13.83	12.14	10.55	11.54
best	443	437	443	434	437	436
Best known (Whitley) 428						

Table 11.1: *Results of series of trials with 50 cities, using different rates of crossover.*

Populations of various sizes from 30 to 600 were used for the SAGA-style approach, and 50 found to be a population size that obtained good results. Starting with 3-city tours, a fixed number of (sets of 3) tournaments was used after each city was added, until full tours of 50 cities were reached. This number of tournaments was one of the parameters varied during experiments, along with the rate at which crossover and segment-inversion operators were applied; as mentioned above, mutation was soon abandoned. In terms of expense, each tournament only involves the comparison of scores; the real expense should be considered to lie in the evaluation of the new offspring, as in real world problems this effort vastly outweighs any machinery of a GA such as tournament comparisons.

If one was trying to obtain a straight comparison with Whitley's results, then a comparable number of individual tour evaluations should be used. But in his case, all the evaluations were of full-length 50-city tours, whereas with the incremental approach the tours varied from trivial lengths of 3 or 4 at the start, to the final value of 50 only at the end. If one considered the complexity of as tour as linearly related to its length, then perhaps 50,000 evaluations overall in the incremental scenario might be equivalent to the 25,000 evaluations used by Whitley; those 50,000 evaluations come out approximately as 1000 evaluations done at each level of tour length, from 4 up to 50; this figure of 1000 was used. This must be a minimum bound on the number to make a 'fair' comparison, as most notions of 'tour complexity' would not scale linearly with the tour length, but rather in a way that increased far more rapidly, perhaps exponentially.

In fact, comparing the incremental approach to Whitley's figures is comparing apples to oranges, as in the examples given here I am not trying to demonstrate that SAGA matches performance with other algorithms on a given problem, but rather that SAGA-style approaches do indeed work sensibly in a potentially open-ended problem-space.

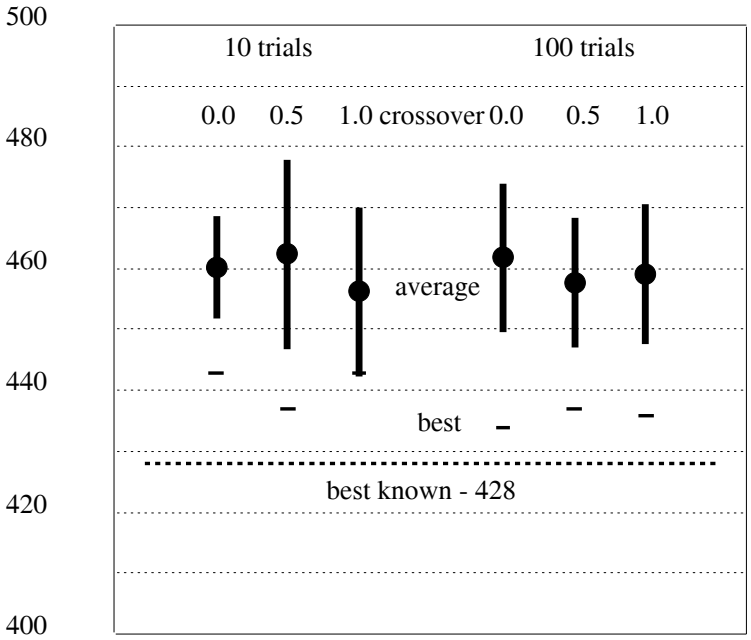

Figure 11.1: *The results of trials with 50-city tours as in Table 11.1, displayed pictorially. For 10 or 100 trials, with different crossover rates of 0.0, 0.5, 1.0, the average length is shown with the vertical bars displaying standard deviation. The best of each set of trials is shown underneath, and can be compared with Whitley's best shown as the dotted line.*

On using the segment-inversion operator alone, at rates of application varying up to a probability of 1.0 (certainty) of being used on each newly generated offspring, values of around 0.5 were found to give the best results. Various rates for the crossover rate were tried, and did not seem to make very much difference; probably because only relatively rarely was a legitimate non-trivial recombination found, and at all other times the crossover operator was ineffectual. Results here are given only for a segment-inversion rate of 0.5, with crossover rates set at 0.0, 0.5 and 1.0.

The results are shown, firstly for runs of 10 trials and then for runs of 100 trials, in Table 11.1, which is displayed pictorially in Figure 11.1. The best achieved in all these trials, 434, is just some 1.4% longer than Whitley's best of 428; this comparison should be treated with caution, as it is subject to the caveats above on just how comparable the two scenarios are.

	normal		linear		every 10		final	
crossover	0.0	0.5	0.0	0.5	0.0	0.5	0.0	0.5
average	128883	129261	128478	129680	130190	129229	128410	129925
std. dev.	3344.2	3650.0	4231.9	2559.0	2736.8	3210.6	4388.2	5101.1
best	123098	123957	121107	125904	126718	126162	122929	124685
Best known 118282								

Table 11.2: *Results of series of trials with 127 cities, using the different methods given in the text. Each series is 10 trials.*

11.4.2 127 cities

A list of 127 cities was selected from a public library of TSP problems, TSPLIB 1.1, compiled by Gerhard Reinelt of Heidelberg University and Robert Bixby of Rice University[1]. The list used was bier127, apparently based on the locations of 127 beergardens in the Augsburg area, with a minimum known tour length of 118282. The co-ordinates in two dimensions are specified, and the distances between cities are once again calculated as the Euclidean distance rounded to the nearest integer.

A series of 10 runs was made with the crossover rate set at 0.0, and then a further series with the rate at 0.5. As with the 50-city examples, there was no mutation, and the segment-inversion rate was set at 0.5 throughout. The results are shown in Table 11.2 and Figure 11.2.

As discussed in the previous section, having the same number of tournaments undertaken after (e.g.) the 100th city is added as were undertaken after the 4th city was added seems to be an inefficient way of distributing a given number of evaluations. Although the TSP happens to be a difficult problem in which (conveniently) evaluations of tours are computationally trivial, in more general real-world problems evaluations are expensive and time-consuming; hence allocation of such trials in a relatively efficient way is important. One possibly more efficient way of distributing the same number of evaluations in the TSP is in such a way that the number of evaluations at a given tour length is linearly related to that tour length; it is not claimed that this is the 'fairest' distribution, merely that this is 'fairer' than the previous flat distribution.

Using this method, with otherwise similar parameters to those above, the results are

[1] Available in the public domain by anonymous ftp from softlib.cs.rice.edu, in directory pub/tsplib.

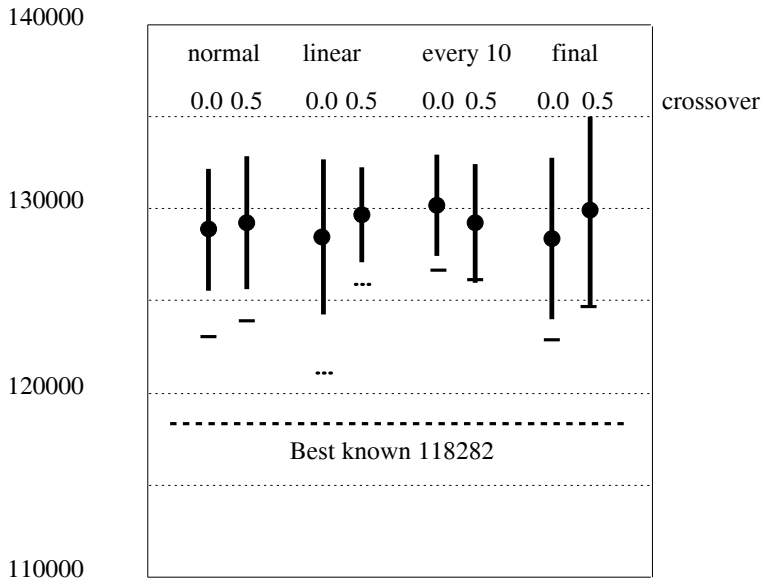

Figure 11.2: *The results of trials with 127-city tours displayed in a similar fashion to Figure 11.1.*

shown under the heading 'linear'.

On the 127 cities, further tests were done with the cities being added 10 at a time (equivalent to 'long jumps'). After each such addition ten times as many tournaments were run; it was arranged so that the intervals of 10 terminated with the last cities being added, so that the final set of evaluations was run with the full list of cities. These results are shown under the heading 'every 10'.

As a final comparison on the 127 cities, with the same parameters the same total number of evaluations were run, but entirely on the full list of cities; in other words, no incremental pathway was taken. These results are shown under the heading 'final'.

On comparing the results, in Table 11.2 and Figure 11.2, there seems relatively little to choose between these 4 methods, bearing in mind the standard deviations. In general the use of crossover made the performance slightly worse, except in the 'every 10' case. The results in the 'every 10' trial with zero crossover were slightly worse than the 'normal' equivalent trial. The best achieved in the linear case, 121107, is just some 2.4% worse than the best known value. The average figure achieved under the 'normal' regime is some 9% worse.

11.5 Conclusions

The incremental SAGA approach has been applied to a standard difficult problem, which allows some comparison with other methods for tackling the same problem. The results achieved on the full list of tours were not as good as the best known results, but were still respectable. In making this comparison it should again be stressed that to some extent it is not comparing like with like, as en route to producing a solution to the 127-city problem the incremental method also solved the N-city problems for N from 3 to 126.

The results have demonstrated that, in this domain at least, the incremental SAGA approach with gradual increase in genotype length over the long term is comparable with the application of a similar GA to the end-problem only. The SAGA approach has however used less resources, in that the earlier evaluations were on shorter tours; and it has produced results for the intermediate-length tours throughout.

CHAPTER 12

Applications — Evolutionary Robotics in Simulation

12.1 Sussex Evolutionary Robotics

The theoretical ideas developed in this thesis are intended for practical application in fields such as Evolutionary Robotics, and this led to the formation of the Evolutionary Robotics Group at Sussex. Initial support from a University of Sussex Research Development Fund award employed me to start transferring these ideas from theory into practice with autonomous robots. While waiting for hardware to be built, these ideas were tested in simulation in the work reported in this chapter, done jointly by Phil Husbands, Dave Cliff and myself. The vision simulation was programmed by Dave Cliff, and the simulation of the physics of the robot world was done by Phil Husbands, along with the implementation of the simple neural model and the genetic encodings. The artificial evolution used, and the class of control systems used, were along the lines that I have advocated here in earlier chapters, and I programmed the genetic algorithms. A number of papers on this work, jointly authored, have been published or accepted for publication, and were listed at the end of Chapter 1. Figures used here came from these publications.

From these beginnings in 1992 the Evolutionary Robotics Group at Sussex has progressed further. The next chapter covers later work with a specialised piece of hardware allowing real vision to be used in a robot that can have a succession of control systems rapidly and automatically evaluated in sequence. Related work within the group by Jakobi (1994, In Press 1995) and Thompson (In Press 1995) will be mentioned below. What this work has in common, apart from the evolutionary approach, is that it is centred around navigational behaviour for wheeled robots in an office-type environment, using minimal low-bandwidth sensing; touch sensors, and either vision, infra-red sensing, or sonar.

The robot on which the simulation work is based is roughly cylindrical in shape, approximately 40 cms in diameter and 30 cms in height. It is mounted on two wheels, one at each end of a diameter; each wheel is independently driven by its own motor. A third

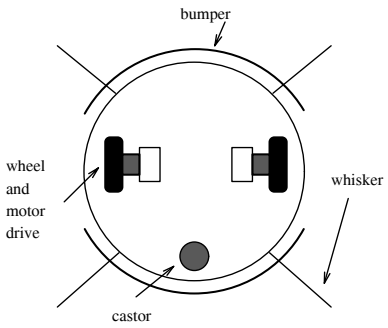

Figure 12.1: *Plan view of simple six-sensor robot.*

trailing castor gives the robot stability. Around the periphery of the robot are touch-sensors or whiskers (Figure 12.1). The robot is powered by onboard batteries; a notebook computer (Compaq 3/20 PC-compatible), powered by its own batteries, carried by the robot can implement the control system, or simulate the operation of a neural network that implements the control system. This robot platform is capable of carrying simple visual sensors or sonars, and the sensors considered here are limited to the touch sensors and limited vision capabilities; motor outputs are limited to those to the left and right wheels. The robot is designed to operate on flat floors in an office-type environment. It is currently being used for evolution of hardware control systems (Thompson In Press 1995).

The work reported in this chapter attempts to simulate in an accurate and realistic manner these robots. and plausible environments; and using this simulation to evolve control structures for simple navigational tasks, without vision and with vision. Successful experiments in simulation will be reported below. A followup to these experiments is to transfer such control structures to the real robot and hence validate the simulations. While this has not been done with the simulations here, it has been done successfully in related work within our group (Jakobi 1994) using the similar but much smaller *Khepera* robot from EPFL in Lausanne, Switzerland.

12.2 Neuron models

In Chapter 5 dynamic real time recurrent neural networks were advocated for such robots. This still leaves room for many choices of what sort of neuron model to use. Depending on the motivation for any particular series of experiments, different classes of neuron models could be used. These alternatives I characterise as:

1. Biologically plausible neurons.

2. Electronically plausible neurons.

3. Computationally convenient neurons.

12.2.1 Biological models

It is generally acknowledged that the neuron models used in Artificial Neural Networks (ANN) tend to be gross simplifications and caricatures of the sort of neurons that neurobiologists study; and indeed for this reason the use instead of the word 'node' may well beg less questions. Nevertheless, whether neurobiologists like it or not, the use of the word *neuron* in the context of ANNs has become widespread.

However, if neuron models are proposed as being to some degree biologically plausible models of those seen in, for instance, insects or relatively simple animals with navigating abilities; and control systems using these neuron models are artificially evolved in robot navigation tasks that are comparable to the navigation tasks carried out by these simple animals; then comparison of the artificially evolved networks with those seen in the real animals could be of interest and practical use to the neurobiologists. Hence the use of *biologically plausible* neurons could assist in scientific studies of animals.

12.2.2 Electronic models

Turning from science to engineering, if ones motivation is to build navigating robots that perform tasks useful to us, then one can think in terms of the process of artificial evolution doing the design work on a control system that will ultimately be built in electronic hardware. In which case, the primitive nodes of the control system should have the characteristics of those electronic hardware items that would ultimately be used; if you like, *electronically plausible neurons*. Recent work within our Sussex group on these lines is discussed in (Thompson In Press 1995).

The minimalist vision system being used is thought of in terms of the electronic equivalent of insect ommatidia; in other words, simple photoreceptors with some specific sensitivity to light, and given angles of acceptance. In practice, there is simultaneous evolution of the visual morphology, in other words potentially the number of such photoreceptors, their angle of acceptance and their placing relative to each other and relative to the robot body. Hence what is actually used is a CCD camera, with software subsampling of the image to give signals comparable to those given by such photoreceptors. This leads to the third class of neuron model.

12.2.3 Taking advantage of computers

Still staying with the engineering rather than scientific stance, it can be acknowledged that the software manipulation of virtual pixels subsampled from a CCD image is enormously easier than the physical placement and alteration of physical photoreceptors; and similarly the alteration and manipulation of control networks simulated in software is enormously easier than rewiring electronic components. Hence the idea of *computationally convenient neurons.*

With this approach, one accepts that one will at all times be using computers to run — 'simulate' if you like — control networks, and hence one should abandon, as an engineer, any attempt to emulate the incidental and unnecessary characteristics of either biological models of neurons, or of electronic components. This leaves the question of: what characteristics are necessary?

Looking back at the discussion in Chapter 5, what is required is a network architecture operating in real time, with nodes that implement some input/output function rapidly. In biological networks the components were, for instance, biological neurons, axons, dendrites that use chemical and electrical signals. In the computer, perhaps one should use minimal nodes that can easily be implemented in programs, and which run fast. A simple threshold operation, testing whether summed inputs exceed a specified threshold and giving a binary output as the result, would be the minimal such operation.

Despite the fact that one is using programs on the computer to run control networks that are genetically specified, it should at all times be remembered that these networks are not implementing computations, but are real time dynamical systems. In practice serial machines are used. Events take place at nodes, and have knock-on effects downstream at future times determined by the delay-lengths of the paths between nodes. If these delay-lengths are real numbers rather than small discrete integers, in practice all future events will be asynchronous. This implies that within such a control network, in general no two events will take place at two or more nodes at the same time. This should be qualified in that in practice delay-lengths will not be quantified in real numbers, but rather numbers of a precision determined by the machine and programming language; and will ultimately depend on the clock-speed of the machine. Such a qualification need not worry us, as for networks of size, e.g., 1,000 nodes with events at any one node happening on average 10 times a second, running on a machine with a clock-speed of 66MHz, an event will happen on average every 6,600 clock cycles. Potentially synchronous events will be handled in

arbitrary order, and this will happen sufficiently rarely for it to have no practical effects, bearing in mind that such networks have to be evolved to be robust in the face of noise.

This means that if all 'instantaneous' thresholding events at nodes are for practical purposes asynchronous and never coincide, a serial machine can handle such a network through running a simple scheduler in the form of a linked list. All such events listed on the schedule have a time-stamp indicating the real time at which they are due to occur. Any change in sensory inputs at an input node are immediately put on the top of the schedule and dealt with immediately; in the absence of such inputs, a realtime clock is watched until the time is reached for the first scheduled event to occur.

Such an event will consist of a simple computationally convenient thresholding event at that node, resulting in either no change or a change in the binary output of that node. In the former case of no change, the event is removed from the scheduler and the waiting process starts again. If there is a change, this will have knock-on effects downstream at nodes connected to the output of that neuron. Such effects will be scheduled to occur at future times dependent on the time-delays on the links, and hence will be inserted into the scheduler's linked list at appropriate places. In general the length of such a scheduled list of pending events will shrink and grow, but given a finite number of nodes the list will remain strictly bounded in length.

This kind of computationally convenient neuron model is currently being tested in the Sussex Evolutionary Robotics work. But in the results discussed below, a slightly biologically plausible neuron is used, simplified so as to be computationally fairly convenient to deal with.

12.3 Early work without vision

Early experiments used simulations of the robot, without vision, navigating in a simulated office-type environment. The particular networks used, in this and later experiments, have a fixed number of input nodes, one for each sensor, and a fixed number of output nodes, namely two attached to the left and two to the right motors. All the nodes are noisy linear threshold devices with outputs in the range $[0.0, 1.0]$, and two units for a motor give a signal in the range $[-1.0, 1.0]$. This range of signal arriving at a motor is divided into 5 segments, giving 5 possible speeds for the wheel: full ahead, half ahead, stop, half-speed reverse, full-speed reverse.

In addition to the fixed number of input and output nodes, there is an arbitrary num-

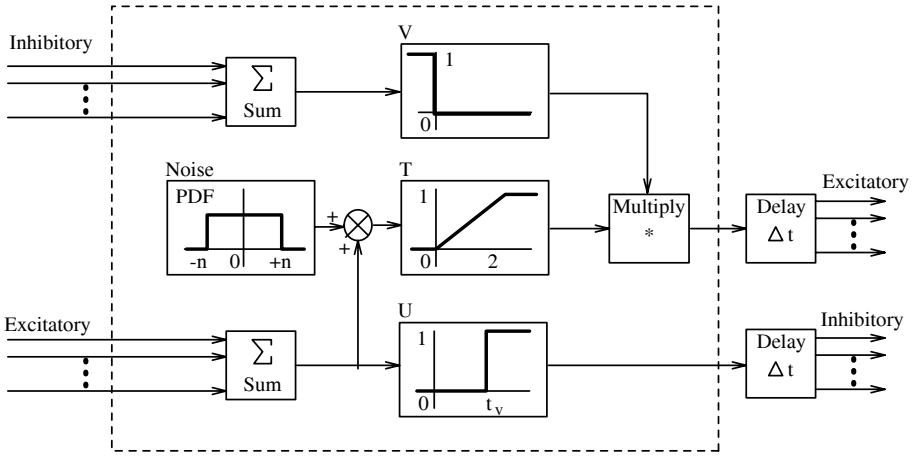

Figure 12.2: *Schematic block diagram showing operations within a single model neuron.*

ber of internal or 'hidden' nodes. Within the networks there are two separate types of connection: normal or excitatory, and veto or inhibitory, with separate threshold values for each. A normal connection is a weighted link joining the output of one unit to the input of another. A veto connection is a special infinitely inhibitory link between two units. If there is a veto connection between units a and b, and a's output exceeds its veto threshold, then all normal output connections from b are switched off – although further veto outputs from b are not affected. The veto threshold is always much higher than the normal threshold. A block diagram of an individual unit is shown in Figure 12.2.

The genotype specifies properties of each unit, and the connections and connection-types (normal or veto) between them. As the total number of units is not fixed, the genotype is of variable length.

12.3.1 Genetic encoding

The genetic encoding used, shown in Figure 12.3, is in effect a description of the network, and does not use developmental mechanisms as proposed in Chapter 10. It is interpreted sequentially. Firstly the input units are coded for, each preceded by a marker. For each node, the first part of its gene can encode node properties such as threshold values; there then follows a variable number of groups each representing a connection from that node. Each group specifies whether it is a normal or veto connection, and then the target node indicated by jump-type and jump-size. The jump-type allows for both relative and absolute addressing. Relative addressing is provided by jumps forwards or backwards along the genotype order; absolute addressing is relative to the start or end of the genotype. These modes of addressing mean that offspring produced by crossover will always be legal.

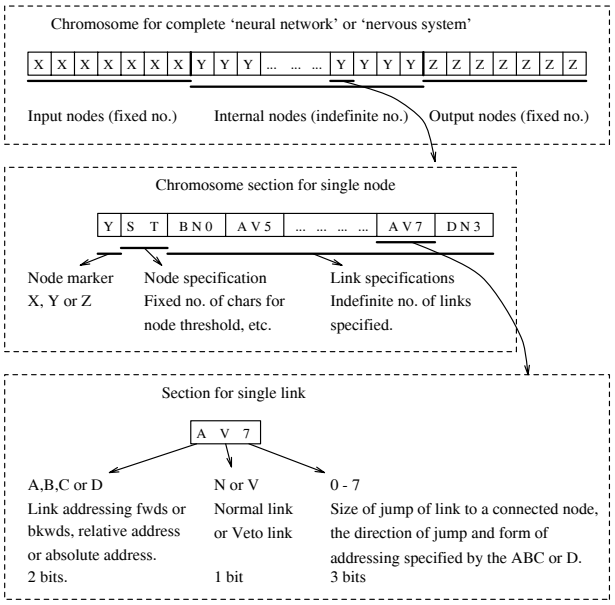

Figure 12.3: *The genetic encoding scheme.*

The internal nodes and output nodes are handled similarly with their own identifying genetic markers. Clearly this scheme allows for any number of internal nodes. The variable length of the resulting genotypes necessitates a careful crossover operator which exchanges homologous segments. In keeping with SAGA principles, when a crossover between two parents results in an offspring of different length, such changes in length (although allowed) are restricted to a minimum.

This method of genetic encoding could easily be adapted to allow genetic specification of weights and thresholds for each node, and delay times on each connection. In practice to date, in the experiments reported here, all delays were set to unity, as were all non-zero weights. The threshold values and noise parameter were fixed. Thus the only aspects of the network under genetic control were the number of hidden units, and the specific connections between units.

12.3.2 Why use noise?

Internal noise was added to each node, which provides interesting properties to the dynamics of the network. Figure 12.4 shows the input/output transfer function over 10 sets of inputs. The internal noise in the networks significantly alters their dynamics. It also helps to make the control system more evolvable and less brittle. Incidentally, it underlines the point that it is not very useful to try and interpret the networks as 'computing' outputs from inputs; rather, the network is a particular dynamical system perturbed by

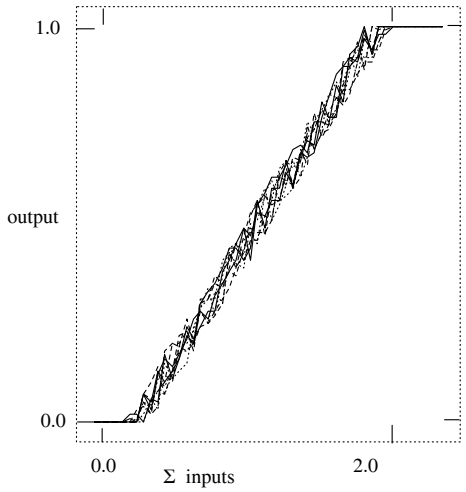

Figure 12.4: *Noisy neuron transfer function.*

the inputs.

In addition, in the simulations at the physical level a significant amount of noise is deliberately added to sensor inputs, motor outputs, and movement in the world such as that arising from collisions. Each robot is scored over a number of trials, and the *worst* of these scores is used as the final evaluation. In this way it is possible to encourage robustness, and on transfer from simulation to the real world the goal is that the 'real' values of parameters should lie within the 'envelope of noise' that is created around the parameter values used in simulation.

Work done by Craig Reynolds (Reynolds 1993) in evolving robot controllers (in simulation) using a Genetic Programming approach shows that whereas success was easy to achieve in a non-noisy environment, on adding noise he was unable to produce a robust, noise tolerant controller. This result supports the choice here of networks rather than programs for the medium of control, in real-world noisy domains.

In the simulations the continuous nature of the system was modelled using a fine-time-slice simulation. All nodes of the network were updated, in effect synchronously, for about 50 iterations (with a variance of up to ± 5 to counter potentially distorting periodic effects). The outputs to the motors were then calculated, and the movement of the robot for one time step was calculated. After this new sensor readings were calculated (for input to the network), and the cycle repeated. The movement of the robot depended on specified wheel velocities which were subject to noise. Collisions with obstacles were modelled realistically, and depended on the speed and the angle of incidence as well as the shape of the obstacle.

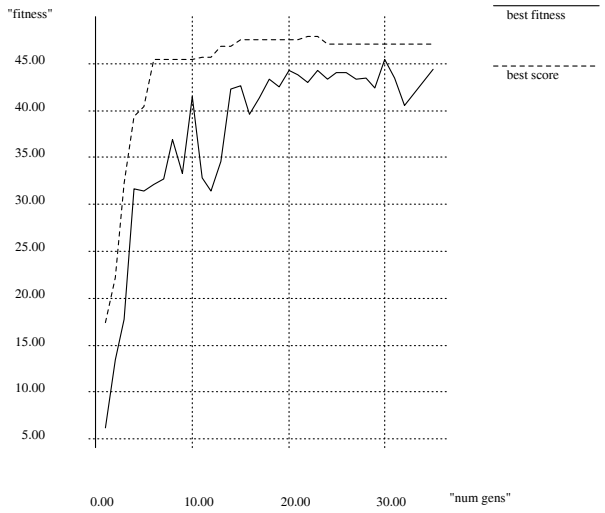

Figure 12.5: *Results of simple experiment. See text for further explanation.*

12.3.3 The experiments

Each of the following experiments was run for 50 to 100 generations, and with a population size of 40 or 60. The crossover rate was set at 1.0, while the mutation rate was of the order of 1 bit per genotype.

A control network was evolved using an evaluation function which encouraged wandering in a cluttered office-type environment containing walls, pillars and doorways. The robots were started from rest with no initialising signals; internal noise was sufficient to allow fitter nets to settle into useful initial states (one illustration of the potentially useful effects of noise). The lower line on the graph in Figure 12.5 shows the fitness (i.e. *worst* score of its 8 runs) of the best individual in each generation; the upper line shows the best score achieved by any member of the population. Successful evolution of robust control networks is shown by the increase in scores, and the convergence between these two lines. Figure 12.6 shows a short run by a robot controlled by one of these networks.

Figure 12.7 shows a typical behaviour generated by a network evolved under a evaluation function describing a much more difficult task. The evaluation function measured the area of the enclosed polygon formed by the robot's path over a finite time period. The robot was always started at random locations with a random orientation.

Figure 12.8 gives comparative results, based on the same task, but with and without the selection measures which were designed to promote robustness. The left graph was obtained by taking the fitness to be the *average* of the small number of runs in the evaluation set; the right graph was obtained by taking the fitness to be the *worst* of the

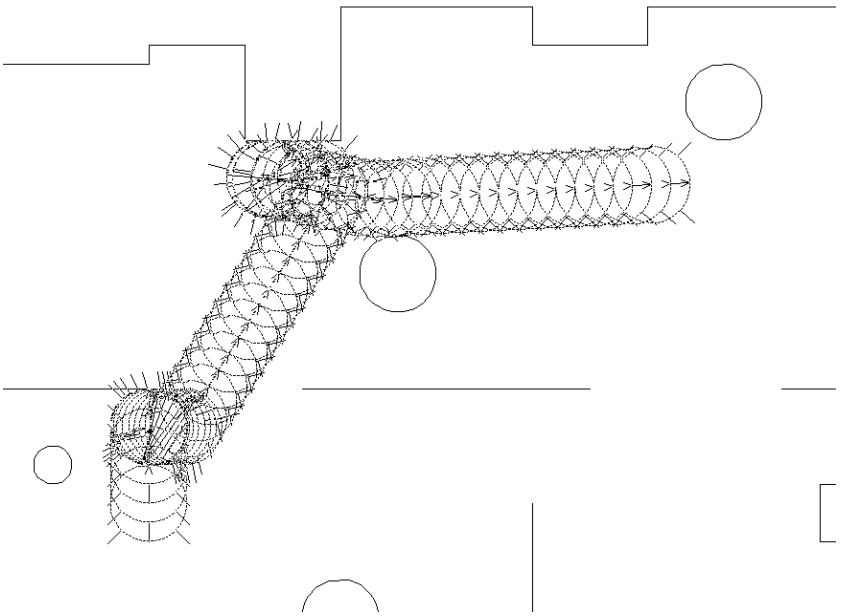

Figure 12.6: *Motion of a single robot controlled by an evolved network.*

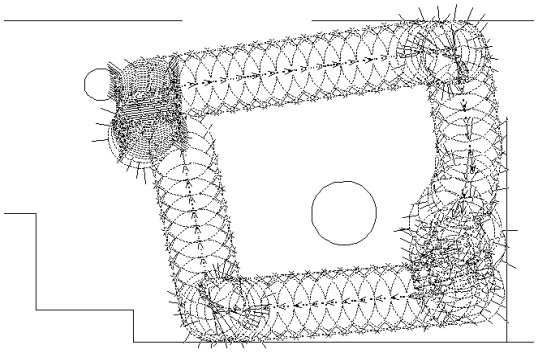

Figure 12.7: *Motion of a robot evolved to maximise the area of the bounding polygon of its path over a limited time period.*

runs. The results show that evaluating from the average gives a better average performance but a very poor noisy worst performance. Evaluating from the worst pushes the worst and average much closer together, providing a far more robust solution.

12.4 Adding vision

Simple visual capabilities were then added to the robot for use in navigation. In keeping with the incremental approach favoured by artificial evolution, such visual capabilities should initially be minimal; in fact one can start with the equivalent of very small numbers of simple light detectors. Rather than imposing on the robot visual sensors with completely predetermined properties, in keeping with the evolutionary approach this work involves

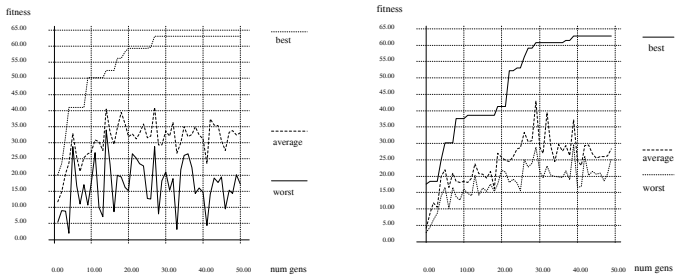

Figure 12.8: *Comparative results for different fitness functions. Left-hand graph is where fitness is measured by the* average *of a series of tests; right-hand graph where fitness is measured as the* worst *performance in a series of tests. Abscissa is generation number; ordinate is fitness value.*

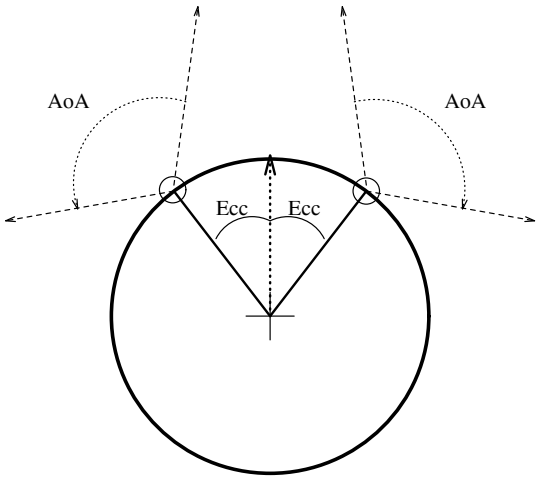

Figure 12.9: *Angle of acceptance and eccentricity for the two-photoreceptor robot. A top-down view of the robot, and the relevant angles.*

investigating the concurrent evolution of visual sensors and control networks. At this stage, the number of photoreceptors is fixed at two, and the variation under genetic control is limited to the angle of acceptance of each receptor (the angular width of the cone that represents its field of view) and the angle of eccentricity (the offset each photoreceptor has from the forward direction of the robot). They are bilaterally symmetric; see Figure 12.9. In the gantry work reported in the next chapter, the number of photoreceptors will be under genetic control.

In this simulation study, the visual input to each photoreceptor is calculated on the basis of a square cross-section to its visual field. As an anti-aliasing measure sixteen rays (in a regular 4×4 grid) are traced for each pixel, the result interpreted as giving a grey-level in the range $[0, 16]$; see Figure 12.11. The world in which the robot operates is a simple (simulated) circular arena, with white floor and ceiling, black walls (Figure 12.10).

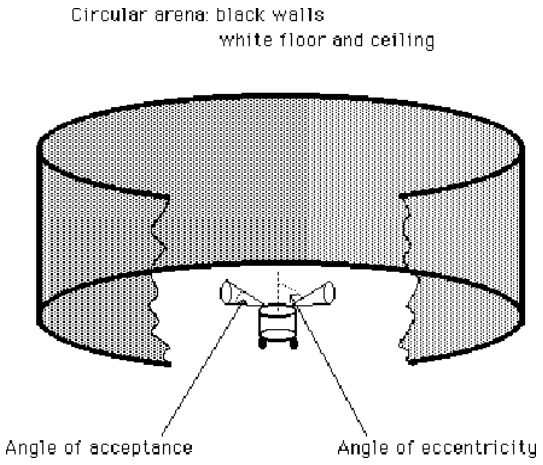

Figure 12.10: *The cylindrical arena in which the robot is tested.*

The calculations were done by embedding the robot in the SYCO vision simulator system (Cliff 1991a), which was originally developed for simulating the visual inputs of a hoverfly. The 'altitude', potentially variable in SYCO, was clamped at a fixed value related to the height of the robot's photoreceptors above the flat floor.

12.4.1 Visual task

The task is set implicitly by the evaluation function, and the robots are rated on the basis of how much time they spent at or near the centre of the arena. This was done by measuring the distance d of the robot from the centre, and weighting this distance by a Gaussian $\mathcal{G}$ of the form:

$$\mathcal{G} = \exp(-d^2/c)$$

for some constant c, which ensures that $\mathcal{G} \approx 0$ for positions of the robot near the perimeter of the arena. Over 100 timesteps the value of this Gaussian was summed, to give the final score. The robot was always started near the perimeter, facing in a random direction.

A run started with a population of size 60 of initially random genotypes. The part of the genotype specifying visual morphology was of fixed length, and in practice treated as a separate 'chromosome'. For selection, a quadratic used to convert ranking into expected contribution to the next generation. The first i members of a population size n have between them a quota proportional to $1/\sqrt{i}$. When $n = 60$, this means that the the first 16 all have above average quotas, and the very first contributes an expected $\sqrt{60} \simeq 7.75$ to the pool for the next generation. This is, compared to standard GAs, abnormally high selection which the rank-based method maintains indefinitely. The mutation rate was set

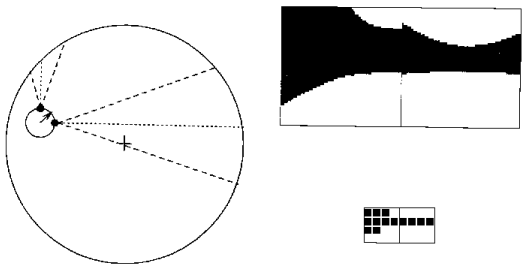

Figure 12.11: *Illustration of the ray-tracing system. The left-hand figure shows the robot's position and orientation within the cylinder, which has black walls and white floor and ceiling. At upper right is a pair of relatively high-resolution images, traced from the robot's position inside the cylinder. The lower-right figure shows the two* 4×4 *images traced prior to averaging, with* $\alpha = 1.571$ *and* $\beta = 0.628$*. The final two photoreceptor brightness levels are derived by averaging the* 4×4 *images.*

at an expected 0.9 bits flipped per genotype. There was 100% recombination, with the single crossover point arranged so that, despite the possible variations in lengths between recombining genotypes, there was minimal change in length in the offspring.

Each genotype was evaluated by decoding the vision chromosome to determine the visual characteristics; decoding the rest of the genotype to determine the control network. Then eight tests were run on the same resulting robot, starting at different random positions near the edge of the arena, using the evaluation function discussed above. The *worst* score achieved over 8 tests was used for the rank-based selection, to encourage robustness. It took around 24 hours on a Sun 4 SPARC workstation to evolve one such population of size 60 for 100 generations. 8 separate populations were evolved in parallel, on 8 workstations. The genotype with the highest fitness from each population was analysed. Typically from 8 populations, between 3 and 5 had shown moderate improvement over the initial random genotypes, whereas the others had evolved to produce near-perfect performance.

Two different successful control systems, termed C1 and C2, have been described elsewhere (Cliff *et al.* 1993c). In both cases, the robots make a smooth approach towards the center of the arena, and then circle there, either on the spot or in a minimum radius circle. Before analysing these results fully, it had been speculated that the control system might be recognising that the centre of the arena had been reached by using the absolute level of the light input at the centre; there it is at a maximum, as far more of the white walls and floor are within the angle of acceptance of each photoreceptor than at any other position nearer the wall.

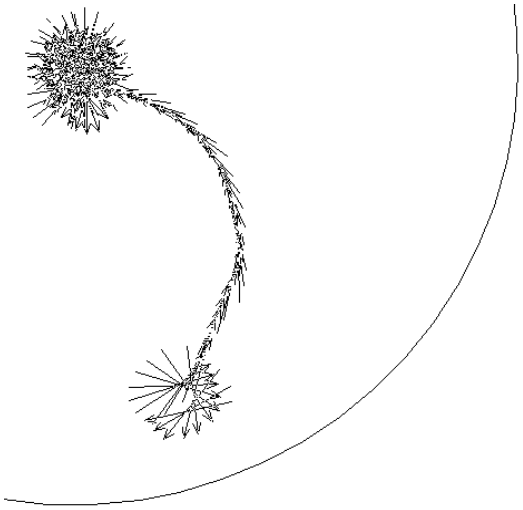

Figure 12.12: *Typical path of a successfully evolved robot, which heads fairly directly for the centre of the room and circles there, using input from 2 photoreceptors. The direction the robot is facing is indicated by arrows for each time step, largely superimposed.*

With this in mind, a more difficult task was set up, where the height of the wall could vary over one order of magnitude, from 4 to 40 units; the diameter of the arena remained at 40, the height of the wall in the original tests was fixed at 15. The full range of possible heights was divided into 10 equal sections, and each robot was given 10 trials, with a height of wall chosen at random from each in turn of the 10 sections. In this way it could be ensured that it was tested across the full range. Thus no use could be made of absolute light values; as is usual in these experiments, the evaluation of the robot was based on the *worst* score it obtained across its trials. Success was similarly achieved in this more difficult navigation task.

12.4.2 Analysis

An analysis of the genetic drift of the population during one run of these simulations was given earlier in Chapter 8, Section 8.9. Now we move to an analysis of the networks described here, which has been developed by Dave Cliff and Phil Husbands.

Typical behaviour for a particular network C1 is shown in Figure 12.12. The robot starts at the edge, moves to the centre, and then stays there, spinning on the spot; the evaluation function does not penalise continued movement. The vision chromosome specifies that the two photoreceptors have 45^o acceptance angles, and face just 6^o each side of the straight-forward direction. The network for C1 is shown in Figure 12.13.

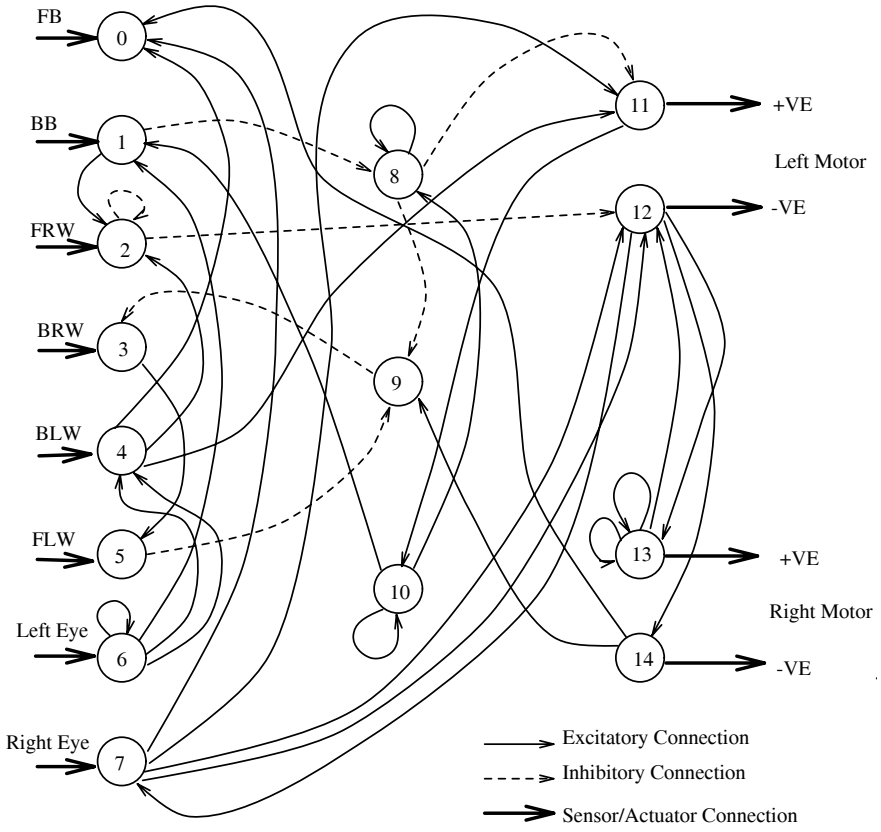

Figure 12.13: *C1 control network. The left-hand column are units originally designated as input units: FB=Front Bumper; BB=Back Bumper; FRW=Front Right Whisker; BRW=Back Right Whisker; BLW=Back Left Whisker; FLW=Front Left Whisker. Right-hand column shows output units to the motor units for left and right wheels. Centre column shows 'hidden units'.*

Analysis of the C1 network starts with identification of redundant units and connections (e.g. unit 0 has no outputs and can be disregarded). Identification of particular sensory-motor pathways which mediate identifiable patterns of behaviour is aided displaying records as shown in Figure 12.14, which gives inputs, outputs, activity levels of nodes, and such robot variables as speed, orientation and distance from centre of the arena. This allows identification of those nodes which are largely inactive and can be eliminated. It can be seen that since early stages of the evolution have allowed visual signals to prevent the robot from bumping into the walls, the touch-sensors are unused. The results of eliminating redundant nodes are shown in Figure 12.15.

From this, it can be seen that evolution has opportunistically taken over some of the otherwise-redundant tactile input units (nodes 1, 2 and 4) to act as 'hidden' units, or 'interneurons', used in sensory-motor pathways between visual inputs and the motor outputs.

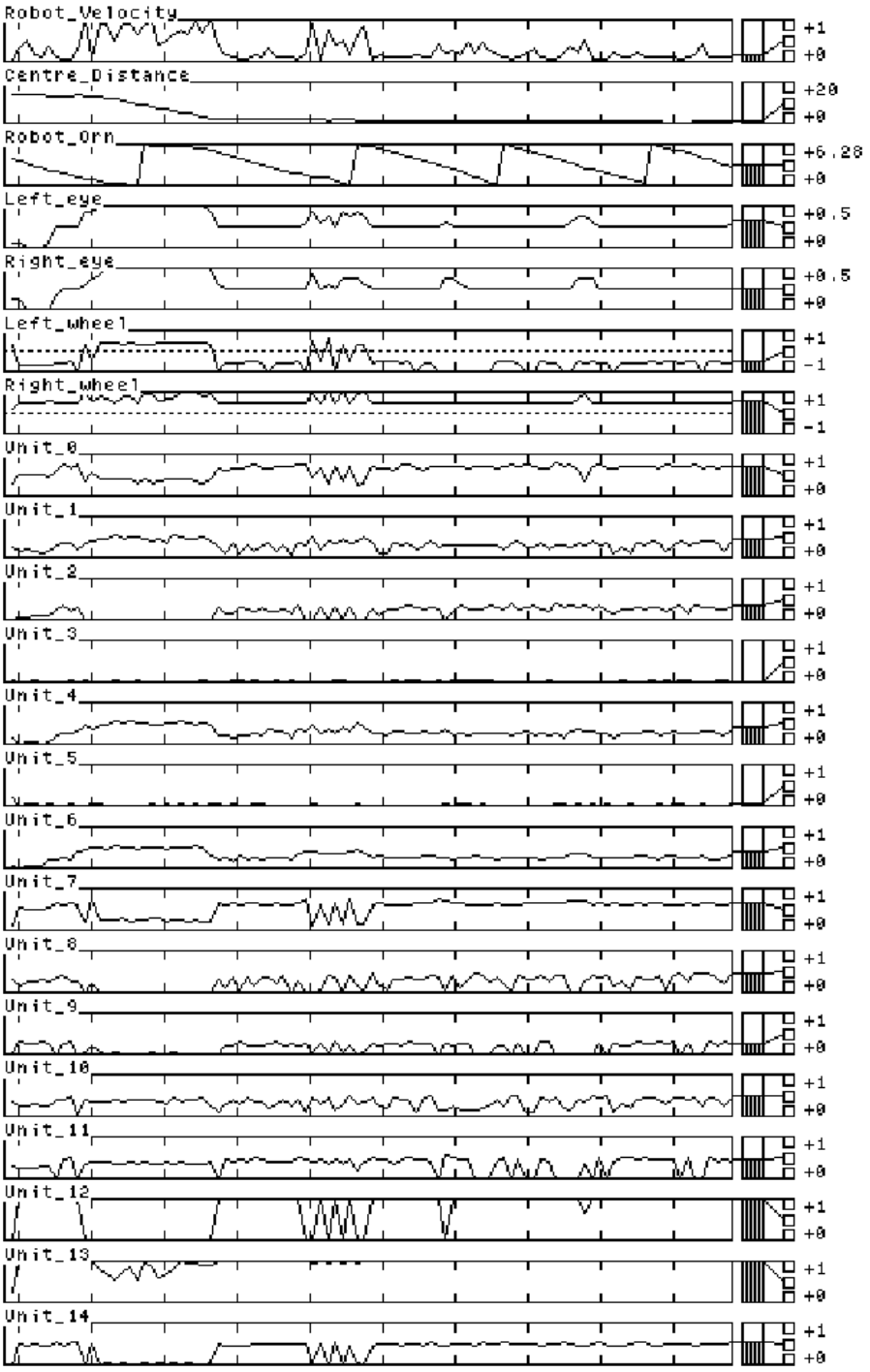

Figure 12.14: *Record of observables and activity levels for the robot. Horizontal axis is time. From top: robot's velocity; robot's orientation; visual input to left photoreceptor; visual input to right photoreceptor; output of left wheel; output of right wheel; activity levels in the control network units.*

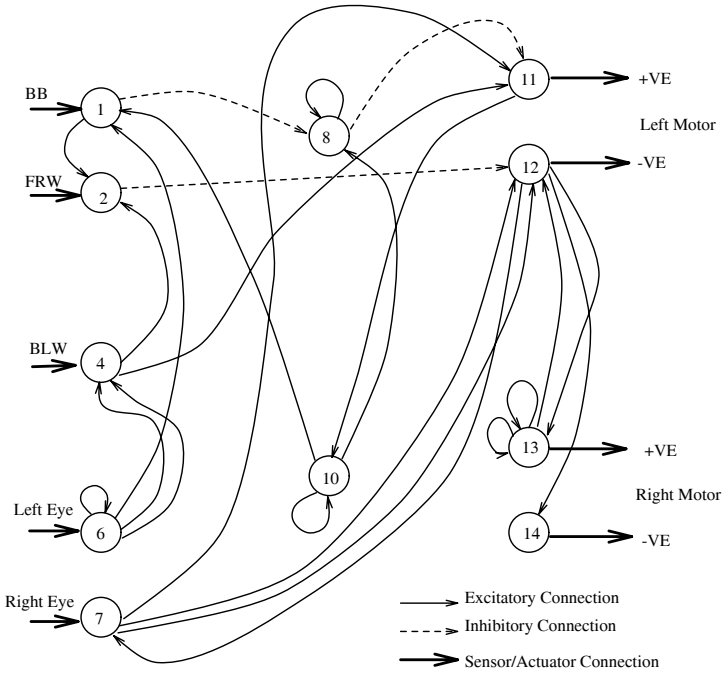

Figure 12.15: *Network with redundant and non-visual units deleted.*

One can also see how the robot behaves in the absence of the noisy conditions under which it was evolved. In these relatively simple networks, such analysis allows one to redraw the active part of the network, and give a full explanation of how it operates. It seems to me problematic, however, to extend such analysis to more complex evolved networks.

12.4.3 Phase portrait analysis

Because of the symmetry of the simple circular arena, the robot's sensory inputs (for a given sensor morphology) are determined solely by its distance r from the centre of the arena, plus its orientation ϕ with respect to the centre of the arena. This allows for the production of a relatively simple phase portrait for a state space giving the visual input of either eye (16 grey-levels, averaged over the noise) for all values of r and ϕ. Any path the robot takes can be translated into a trajectory in this $r\phi$ space.

Starting from this, a different method of analysis has been developed by Phil Husbands; it is described fully in (Husbands *et al.* 1995). By analysing the behaviour of the network for all different possible visual inputs, all possible trajectories can be mapped out. When this is done, a single attractor can be seen, corresponding to a tight orbit around the centre of the arena.

Figure 12.16 shows the actual grey level inputs to the left and right eyes represented

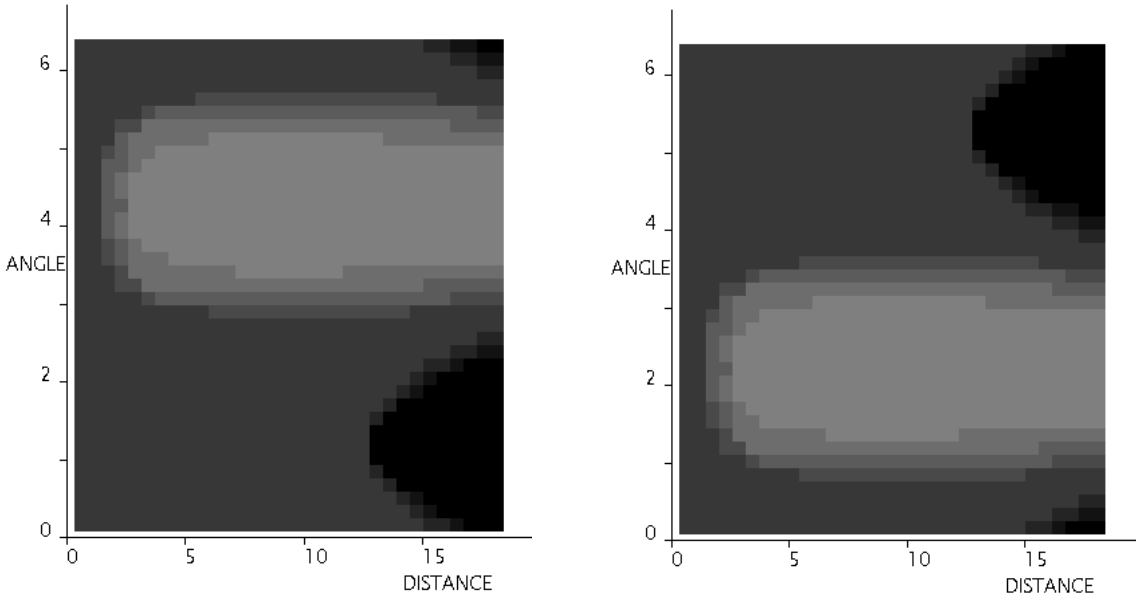

Figure 12.16: *Grey level inputs to left and right eyes in $r\phi$ space.*

in $r\phi$ space for this wall height. The dominant motor behaviours for all possible regions in this space were deduced through network analysis and hence the phase portrait of Figure 12.17 was produced; a vector flow field was constructed across the regions. Where there is a noisy oscillation between two behaviours, a vector average was used. There appears to be a single attractor at $(rad, \frac{\pi}{2})$.

Once again, this analysis has depended on the relative simplicity of the network, plus the very important symmetries of the environment to give a low-dimensional phase space. It is an open question how far such an analysis can be extended to more complex networks and environments. A similar type of analysis has been given by Gallagher and Beer (1993).

12.5 Minimal Analysis, and Conclusions

A related and possibly powerful framework for analysing such dynamical systems is that presented by Thomas in (Thomas 1991), which uses the fact that in these domains interactions both within a network and often outside are *doubly non-linear* or basically sigmoid in character; as inputs to some unit range from minimum to maximum values, outputs typically are saturated at some minimum or maximum value over much of the range, with an intermediate region of intermediate values. The logical idealisation of such a sigmoid as a step function "remarkably preserves the essential qualitative behavior of the systems" (Thomas 1991), and allows identification of internal feedback loops and steady states. This

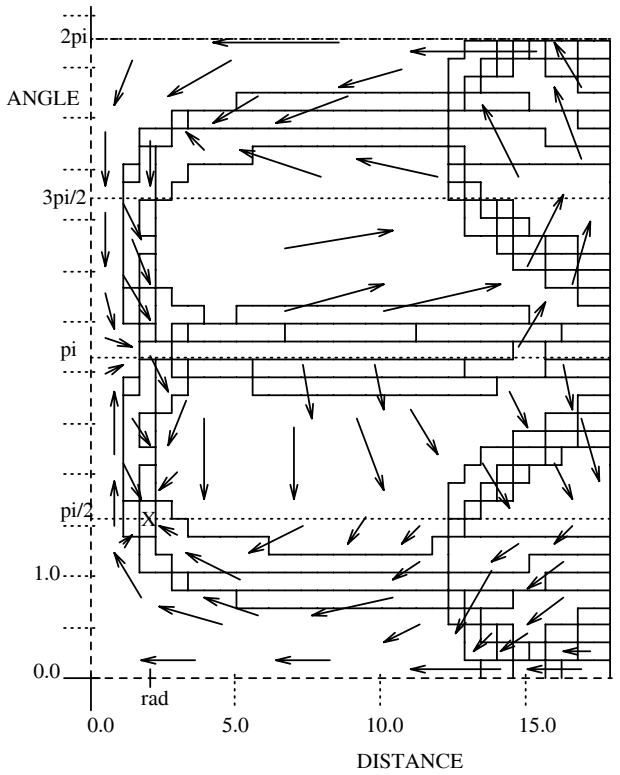

Figure 12.17: *Vector flow field across $r\phi$ space.*

could be done automatically for any size of regulatory net, and the necessary calculations scale polynomially with the number of units. The problem of modelling the environment is, however, more difficult than modelling such control systems.

For scientists attempting to understand how things work, such analyses should of course be attempted. For an engineer wishing to produce a working complex system, however, such analyses need not be relevant if artificial evolution is being used. An evolutionary approach to design will need the specific kinds of analysis given in this thesis:

1. How should artificial evolution operate?

2. What primitives of a control system should be available for evolution to assemble?

In this chapter the application of the principles worked out has been demonstrated with simulated robots using minimal bandwidth vision in a simple environment. The simulation was detailed, and included significant noise. Success was demonstrated at simple, but non-trivial, navigational tasks.

However the history of AI is full of proposals which have worked adequately in simulations, but have failed to transfer to the real world, or to scale up reasonably as the tasks

to be tackled have got more complex. Some start in this direction will be covered in the next chapter.

CHAPTER 13

Applications — Evolutionary Robotics with the Gantry[1]

13.1 From Simulation to Reality ...

The previous chapter discussed work with a simulated robot, whose motion was calculated to a considerable degree of verisimilitude, based on measurements taken from a real robot. Collisions with walls were modelled, as were noisy motor properties. The theory was advanced that *if* control systems could be evolved to handle robustly a certain amount of noise deliberately added to the simulation, and *if* the differences between the simulation, and the real robot on which it was based together with a real environment, were of lesser magnitude than this 'envelope of noise', then successful transfer of such control systems from the simulation to the real robot should be possible.

Recent success in a transplantation — indeed results superior in the real robot to those achieved in a noisy simulation — have been reported in a control system based on neural networks (Meeden *et al.* 1993). Within our Sussex group success has also been achieved in a similar transplantation with the *Khepera* robot (Jakobi *et al.* In Press 1995). Even so, a lot of questions are still raised by that second *if*, and perhaps the biggest question-mark lies against the simulation of the optic array available to a robot in the simulated, as compared to a real, environment.

Hence plans were made to perform the whole evolutionary process with a real robot, moving in the real world. Artificial evolution requires the evaluation of large numbers of robot control systems, so it is advisable to automate evaluations. With navigation tasks, the absolute position of the robot during its trials should be available to an overseeing program which evaluates the robots for the genetic algorithm — while of course this information should not be available in any way to the individual robot control systems. Automatic re-positioning of the robot at fixed or random positions for the start of each trial is also necessary. A specialised piece of visuo-robotic equipment was developed fulfilling

[1] A version of this chapter appears as (Harvey *et al.* 1994).

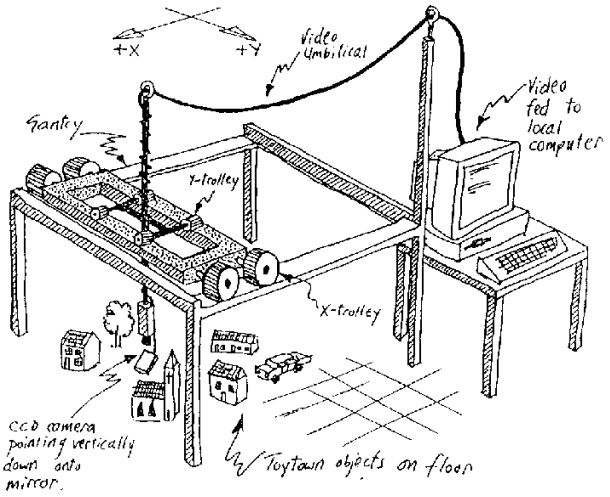

Figure 13.1: *The Toytown gantry (cartoon by Dave Cliff).*

these requirements — the gantry-robot.

An early design sketch for this is shown in Figure 13.1, and a picture of the actual equipment in Figure 13.2.

This was built initially with funds from the University of Sussex development fund, and later research funded by the Science and Engineering Research Council. Tony Simpson, Martin Nock, Jerry Mitchell should be thanked for engineering design and construction work on the gantry; Harry Butterworth for design and building of the Frame Grabber, Single Board Computer (SBC), and the interfacing between computers. The assembly language programming of the SBC was done by myself, and the programming of the GA and the artificial neural networks by Phil Husbands and myself.

In this chapter I present results from experiments with this gantry, in which visually guided behaviours are artificially evolved in the real world. As far as is known, this is the first time this has been achieved. The experiments include a progression of tasks of increasing complexity.

13.2 The Gantry-Robot

13.2.1 Introduction

The gantry-robot occupies a position conceptually partway between a physical mobile robot with two wheels and low-bandwidth vision, and the simulation thereof within a simulated environment. The robot is physically built, cylindrical, some 150mm in diameter, and moves in a real environment — the term 'robot' is here used to refer to that part

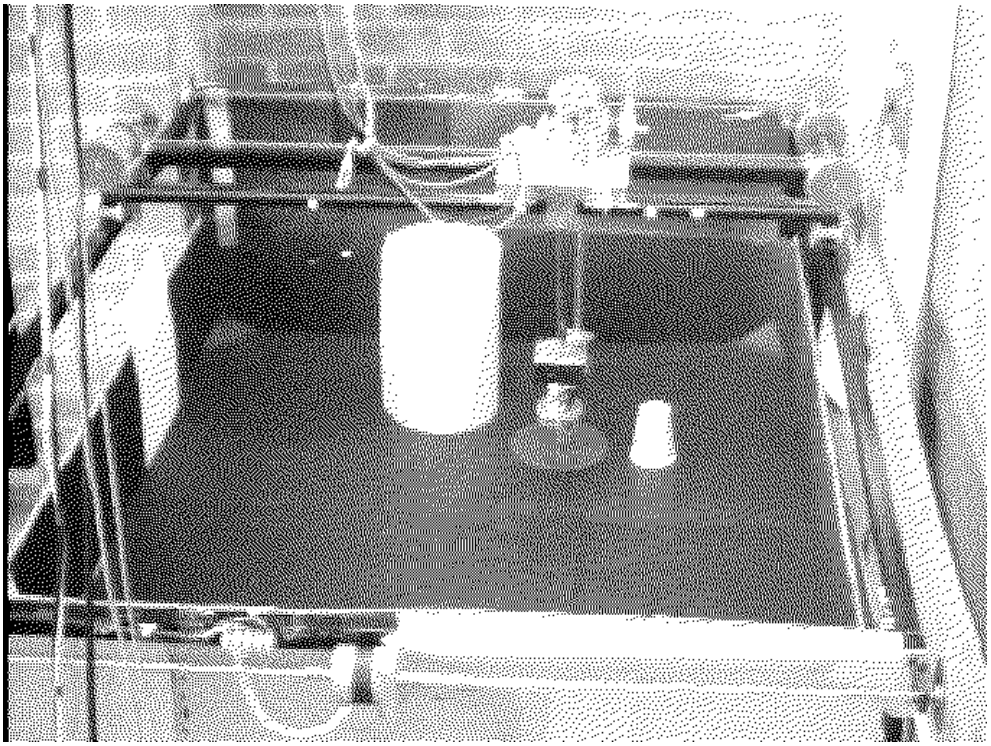

Figure 13.2: *The Gantry viewed from above. The horizontal girder moves along the side rails, and the robot is suspended from a platform which moves along this girder. White targets can be seen on the wall and the floor of the predominately dark environment.*

which moves around and has the sensors mounted on it. Instead of two wheels, however, the robot is suspended from the gantry-frame with stepper motors that allow translational movement in the X and Y directions, relative to a co-ordinate frame fixed to the gantry (see Figure 13.2). Such movement, together with appropriate rotation of the sensory apparatus, can be thought of as corresponding to that which would be produced by left and right wheels. The visual sensory apparatus consists of a CCD camera pointing down at a mirror inclined at 45^o to the vertical (see Figure 13.3). The mirror can be rotated about a vertical axis so that its orientation always corresponds to the direction the 'robot' is facing. The visual inputs undergo some transformations en route to the control system, described in detail below. The hardware is designed so that these transformations are done completely externally to the processing of the control system. If all the transformations made on the sensory inputs and the motor outputs accurately reflected the physics of a real mobile robot, then in principle, a control system successfully evolved on the gantry should be able to be transplanted to a mobile robot with two genuine wheels, and with photoreceptors instead of the vision system described below. Such a transplantation has

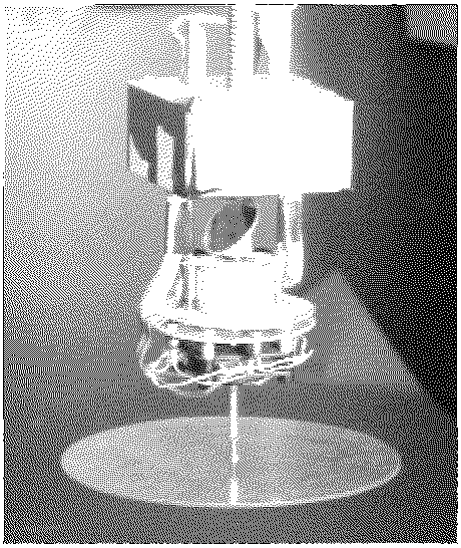

Figure 13.3: *The gantry-robot. The camera inside the top box points down at the inclined mirror, which can be turned by the stepper-motor beneath. The lower plastic disk is suspended from a joystick, to sense bumps.*

not been attempted yet, and is not a prime concern of the current work with this apparatus. Indeed, there are current limitations, discussed below, which would probably hinder it. For the time being the experiments discussed here are best considered as having conditions comparable in complexity and difficulty to those conditions met by a free-running mobile robot; the aim is a fairly general investigation of the artificial evolution of sensorimotor control systems. The most complex and least plausible part of previous simulations — the optic array available for the robot to sample — is in these experiments the real thing.

Conceptually, the control system for the robot is a recurrent dynamic neural net, genetically specified, and in practice implemented on a fast PC (personal computer), the 'Brain PC'. During each robot trial this PC is dedicated solely to running the neural net simulation, and it receives any changes in visual input by interrupts from a second dedicated 'Vision PC'. A third single-board computer, the SBC, similarly sends to the Brain PC, via interrupts, any tactile inputs resulting from the robot bumping into walls or physical obstacles. The only outputs of the control system are motor outputs specified by values on particular nodes of the neural network; these values are sent, via interrupts, to the SBC, which generates the appropriate stepper motor movements on the gantry.

Thus all interactions between the three computers used (Brain PC, Vision PC and SBC) are mediated by interrupts (see Figure 13.4); and the overall system is deliberately

designed so that these interrupts, although inherently asynchronous and unpredictable, are nevertheless sufficiently infrequent for them not to clash with the inherent timescales of the neural network, vision and stepper motor processing.

13.2.2 The Vision PC

There is a 33MHz 486 PC dedicated solely to processing the camera output and calculating (to an appropriate degree of accuracy) the scalar values produced by averaging the signal over genetically specified receptive fields (corresponding to 'virtual photoreceptors') receiving input from parts of the visual field defined by the camera and mirror (further details below). At the beginning of a set of trials for a particular robot, the genetic specification for the visual morphology is passed to this Vision PC. During each trial, whenever the orientation of the robot changes (the full circle is discretized into 96 orientations) a single byte is sent to the Vision PC from the SBC specifying the new orientation; whenever calculations show that the visual input to any of the receptive fields changes in value (scaled in the range 0 to 15) then the details of such a change are sent as single-byte interrupts to the Brain PC (see Figure 13.5).

The constant source of data to the Vision PC is derived from the output of a small CCD monochrome camera, originally designed for use as a security camera. With a wide-angle (about 40^o) fixed-focus lens about 6mm in diameter, this is housed in a box about 70mm square vertically above the angled mirror of the robot, facing downwards. The CCD produces composite video output of some 1 volt peak to peak, with a video bandwidth of 4MHz. A purpose-built Frame-Grabber samples the image at 50 Hz into a high-speed 2K CMOS dual port RAM, completely independently and asynchronously relative to any processing of the image by the Vision PC.

The image (64 pixels square) is placed directly into screen memory, where it is visible to the user in real time, as a monochrome image without visible flicker, but with noticeable noisiness in individual pixel values. This view is of the angled mirror from above; to convert to a horizontal view the right way up it is necessary to take into account the current orientation ϕ of the mirror. Using precalculated lookup tables for speed, a suitably rotated circular image of the mirror, some 50 pixels in diameter, is placed on the screen beside the original square image; the next stage is to sub-sample this image to produce signals from the genetically specified 'receptive fields', conceptually corresponding to photoreceptors of particular angles of acceptance and positions. The angle of acceptance of the CCD camera

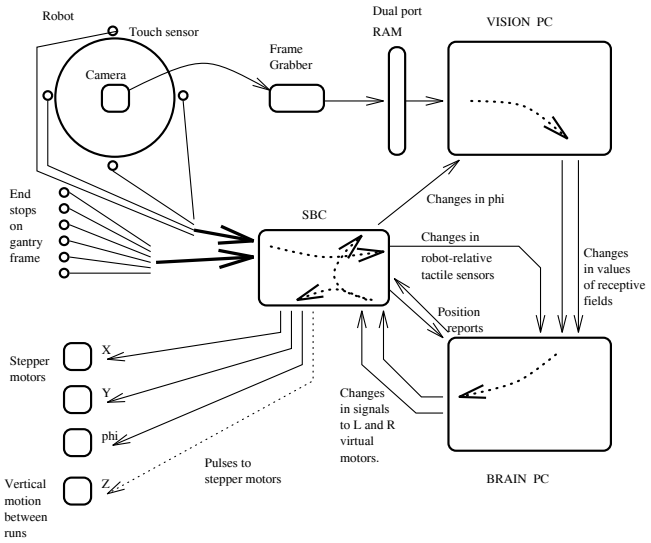

Figure 13.4: *The different rôles of the Vision computer, the Brain computer and the SBC.*

(via the mirror) is about 30^o; the maximum angle of acceptance of a receptive field is about 8^o, and its maximum angle of eccentricity away from a line straight ahead is about 11^o.

Up to 256 such receptive fields can be specified with, to 8-bit accuracy: the diameter of the field; its distance from the centre of the camera's field of view; and the angle from the vertical made by a line from the centre of the camera's field of view to the centre of the field. To calculate the signal from such a field, the average is taken of 25 'jittered' pixels in the camera image spread across the appropriate area. In this way a value (4 bits) can be calculated for each receptive field at least as fast as the camera image is updated; if it has changed, the new value is sent to the Brain PC. The only visual inputs available to the genetically designed robot control system are such scalar values.

In practice, even when the robot is stationary in a stationary environment, noise results in field values changing perhaps once a second. When the robot, and its field of view, is rotating, field values may be changing perhaps 20 times a second. As part of the philosophy of minimalist vision espoused here, if the field values fluctuated too wildly it would be advisable to coarsen the precision used (to perhaps only 2 or 3 bits instead of 4 bits).

13.2.3 The Brain PC

This is a 66MHz 486 PC which has two separate groups of tasks to do at different times. Firstly, the Genetic Algorithm (GA) code is run on this machine, reproduction, crossover and mutation are performed here in between generations, and at the start of a set of trials

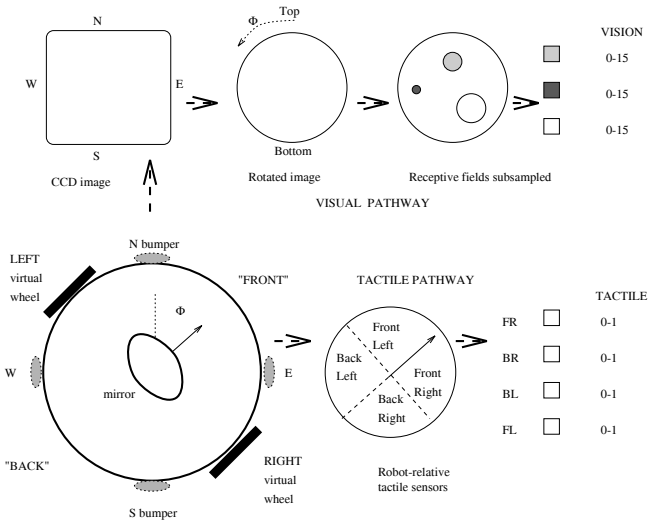

Figure 13.5: *A schematic of the sensory pathways, visual and tactile, from the gantry-robot hardware at bottom left through to the nodes of the neural network on the right.*

for each robot architecture the specification of the visual morphology is transmitted to the Vision PC. As with most GAs, however, the amount of time spent running the genetic machinery is trivial compared with the time spent running the evaluations, and this latter constitutes the second group of tasks.

While running an individual evaluation, the Brain PC receives signals representing sensory inputs from the Vision PC and the SBC — the former sends the values of each genetically specified visual receptive field, the latter sends information from the touch sensors of the robot — and the only signals that the Brain PC sends out indicate changes in values of the left and right virtual motors of the robot. These values, which are restricted to integers from -2 to 2 for each virtual motor independently, are passed on as single-byte interrupts to the SBC (see Figure 13.6).

Apart from such communications, the Brain PC is dedicated to running the genetically specified control system – (simulated) neural network – for a specified period. At intervals during an evaluation, a signal is sent from the Brain PC to the SBC requesting the current X and Y co-ordinates of the robot, and the current angle at which the robot is 'facing', for use in keeping score according to the current evaluation or fitness function. At the end of a run, a byte sent from the Brain PC to the SBC requests the return of the robot to the 'origin' of the gantry.

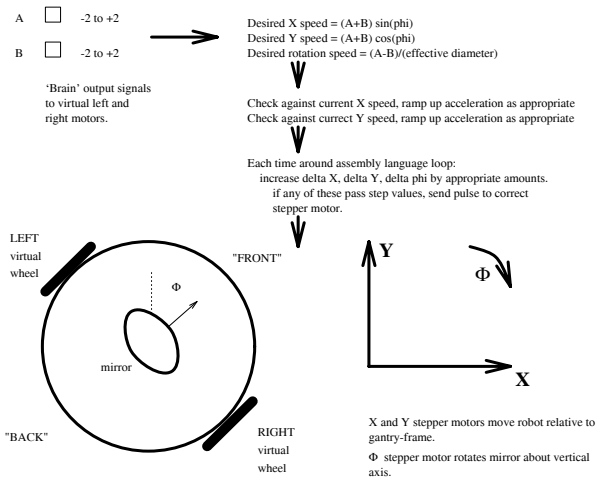

Figure 13.6: *The conversion from the motor outputs signalled by the neural network to the stepper motor movements which move the gantry-robot.*

13.2.4 The SBC, Single Board Computer

The SBC is a minimal 16 bit 68000 system with 256K of RAM and 128K of ROM, running at 10MHz. It has memory mapped ports that connect it to the Vision PC, the Brain PC and the various switches and motors attached to the gantry; a single byte to each PC, and 4 bytes available to the gantry. The code in the SBC, written in assembler, does all the transformations between hardware-relative and robot-relative signals, plus some housekeeping.

The X and Y stepper motors provide translational movement of about 1.2mm per step; the mirror motor turns through 96 steps in a full rotation. As the SBC produces the pulses sent to the relevant stepper motors, it is here that the current values of X, Y and ϕ are stored. Occasional interrupts from the Brain PC will notify new values of the desired speeds of the virtual left and right wheels. These are translated into desired speeds in the X and Y directions, and desired angular velocity of the mirror; for speed of processing, the sines and cosines used here are pre-calculated into a lookup table. Instantaneous changes in desired speed cannot be translated directly into instantaneous changes in stepper motor pulse frequency, due to the momentum of the masses these motors must move. Hence speeds are ramped up relatively slowly towards the desired speed — from zero to full speed in about 2 seconds — decelerations are ramped down rather more swiftly. As the mirror is so light, such ramping was not deemed necessary for rotation — with unexpected side-effects described below.

Maximum attainable speeds in the X and Y directions are about 200mm/sec; the

maximum angular velocity required for the mirror is well within the limits of that stepper motor. On any change in ϕ, corresponding to a step of the mirror, the SBC sends a single-byte interrupt to the Vision PC, notifying it of the new value of ϕ.

Signals from the end-stops for maximum movement along the gantry-frame, and signals from the touch-sensors on the robot, are also processed by the SBC. On the robot, a plastic disc in the horizontal plane is suspended on a joystick vertically below the body (see Figure 13.3). This detects contact, on each of 4 sides of the robot (in gantry-relative co-ordinates) on 4 switches, and since 2 neighbouring switches may be closed simultaneously allows discrimination into 8 (not particularly accurate) sectors. The SBC converts any such signals into robot-relative co-ordinates, according to the current value of ϕ; and then groups the possible outcomes into 4 possible sectors: Front Left, Front Right, Back Left, Back Right. Any changes in the values of these (virtual) sensors are communicated via an interrupt to the Brain PC.

For safety reasons there are further switches acting as a 'Dead Man's Handle'. On movement beyond the limits, or excessive movement on the joystick which senses touch-contact on the robot, a relay which over-rides the software prevents any further motor movement.

13.3 The robot dynamics

Some issues relating to the physical dynamics have already been mentioned; the ramping up and down of stepper motor movements broadly (and perhaps inaccurately) relates to the momentum of a freely mobile wheeled robot.

Collisions with walls or obstacles were loosely modelled on the collisions of a free mobile robot, while recognising that such modelling may well be inaccurate. But if the virtual physics of collisions as handled on the gantry is comparable in complexity to the actual physics of collisions on a free mobile robot, then success in evolving control systems on the gantry gives good reason to expect success in evolving control systems for a real physics.

On collision with a wall, all further movement into the wall was prevented, as was any translational movement along the wall; in other words, of any desired robot motion, only that component perpendicularly away from the wall was allowed, until contact with the wall was lost. Angular velocity that attempted to turn the robot further in towards the wall was forbidden, and only angular velocity away was allowed. Noise could be added to such collisions.

One puzzling phenomenon often observed, particularly in initial randomly generated populations, was that of a robot turning on the spot in a noisy fashion. On reflection, this turned out to be an artefact of the way translational momentum had been implemented in the SBC code, but without angular momentum. This clearly showed how the virtual physics as currently implemented did not accurately reflect the real physics of a free mobile robot.

The translation of the tactile sensor mounted on the robot (via a joystick) to robot-relative co-ordinates means that any such contact is allocated to one of 4 possible sectors; such allocation should be accurate to within about 22.5^o. Though fairly crude, this is adequate for many purposes. The visual inputs are currently subject to various limitations.

Firstly, the lower part of the robot body is supported from the upper half with two thin vertical bolts, which come into the field of view when the mirror is facing towards them. These appear as dark bars 2 to 3 pixels wide on the CCD image, and affect the values of any receptive fields sampling from this area. In principle this could directly provide visual information for two fixed directions for the robot to 'face'. In addition, these bars tend to occlude any distant target used in navigation trials; for early crude experiments this may not be too significant, but it certainly will matter when finer resolution is needed, and these bars produce greater effects than background noise levels. In future work it should be possible to fit a new head on the gantry which overcomes this problem.

Secondly, the fact that the mirror turns in discrete jumps, of 3.75^o at the moment, means that either the angles of acceptance of the receptive fields, or alternatively the horizontal angle subtended by any significant visual features, should be somewhat greater than 3.75^o. This could be overcome with a finer resolution motor.

Thirdly, the visual inputs are naturally noisy (see Section 13.4.2). The natural variation in daylight, as day progresses into night, causes particular problems. When the gantry-frame was exposed to such variation, it was discovered that evolved systems that worked well in the daytime did not work well under artificial light alone at night-time, and vice versa. Individual robot systems were evaluated over a period of perhaps 3 minutes only, and hence it is no surprise that robustness against such longterm variations was not achieved. Since the recognition of this problem the gantry has been largely shielded against daylight variations. In the future the plan is to deliberately vary lighting conditions *within* each robot trial, to try to achieve robustness against such variations.

13.4 Experiments

The following sections describe experiments that have been carried out, demonstrating how these methods cope with the move from simulations to the real world. These begin by exploring primitive visually guided behaviours in static environments, concentrating on target approaching. However, as we shall see, some of the evolved control systems showed surprising degrees of adaptiveness when tested on more general versions of the task they were evolved for. Success has been achieved with a sequence of tasks of increasing difficulty.

13.4.1 Networks and Genotypes

In all of the experiments reported here the same networks and genetic encoding schemes were used as in the earlier simulation work. This made it possible to see how well they transferred to real world tasks. However, they are the simplest, perhaps least powerful, of the classes of networks and genetic encodings that have been advocated here, and future experiments will explore more sophisticated methods. Briefly, the evolutionary algorithms search concurrently for a network architecture and visual morphology capable of generating behaviours resulting in a high score on an evaluation function that implicitly describes a visually guided task. This is achieved by using a genetic algorithm acting on pairs of 'chromosomes' encoding the network and visual morphology of a robot control system. One of the chromosomes is a fixed length bit string encoding the position and size of three visual receptive fields as described above. The other is a variable length character string encoding the architecture of the control network. Each net has a fixed number of input nodes and output nodes, one input for each visual receptive field and one for each of the four tactile sensors described earlier. There are four output nodes, two for each 'virtual motor'. The output signals of these pairs are subtracted to give motor signals in the range [-1,1]. The genotypes encode for a variable number of hidden units and for a variable number of unrestricted excitatory and inhibitory connections between the nodes.

The model neurons use separate channels for excitation and inhibition. Real values in the range [0,1] propagate along excitatory links subject to delays associated with the links. The inhibitory (or veto) channel mechanism works as follows. If the sum of excitatory inputs exceeds a threshold, T_v, the value 1.0 is propagated along any inhibitory output links the unit may have, otherwise a value of 0.0 is propagated. Veto links also have associated delays. Any unit that receives a non-zero inhibitory input has its excitatory output reduced to zero (i.e. is vetoed). In the absence of inhibitory input, excitatory

outputs are produced by summing all excitatory inputs, adding a quantity of noise, and passing the resulting sum through a simple linear threshold function, $F(x)$, given below. Noise was added to provide further potentially interesting and useful dynamics. The noise was uniformly distributed in the real range [-N,+N].

$$F(x) = \begin{cases} 0, & \text{if } x \leq T_1 \\ \frac{x-T_1}{T_2-T_1}, & \text{if } T_1 < x < T_2 \\ 1, & \text{if } x \geq T_2. \end{cases} \tag{13.1}$$

The networks are run on the Brain PC; their continuous nature is modelled by using very fine time slice techniques, In the experiments described here the following neuron parameter setting were used: N=0.1, T_v=0.75, T_1=0.0 and T_2=2.0. The networks are hardwired in the sense that they do not undergo any architectural changes during their lifetime; they all had unit weights and time delays on their connections.

13.4.2 Experimental Details

In each of the experiments a population size of 30 was used with a genetic algorithm employing a linear rank-based selection method, ensuring the best individual in a population was twice as likely to breed as the median individual. Each generation took about 1.5 hours to evaluate. The most fit individual was always carried over to the next generation unchanged. A specialised crossover allowing small changes in length between offspring and parents was used — at any one time a single extra node in the network could be inserted or deleted. and the number of connections from a node could also be changed by one. Mutation rates were set at 1.0 bit per vision chromosome and 1.8 bits per network chromosome.

With the walls and floor of the gantry environment predominantly dark, initial tasks were navigating towards white paper targets. In keeping with the incremental evolutionary methodology, deliberately simple visual environments were used initially, as a basis to moving on to more complex ones. Lighting was provided by fluorescent lights in the ceiling above, with the gantry screened from direct daylight but still subject to significant indirect daylight variations. However, the dark surfaces did not in practice provide uniform light intensities, neither over space nor over time; even when the robot was stationary, individual pixel values would fluctuate by up to 2 units, on a scale of 0 to 15. Varying illuminance of different parts of the walls provided potential visual information.

The following sequence of tasks of increasing difficulty were used:

1. Forward movement.

2. Movement towards a large target.

3. Movement towards a small target.

4. Distinguishing a triangle from a square.

13.4.3 An Initially Converged Population

For a continuing sequence of experiments, with tasks of added complexity, the starting point in each case is the population that succeeded before, but there are different choices for the very first population. One could start with a randomly generated population (i.e. their genotypes are randomly generated from valid symbols), which would be the normal GA technique. But often in a normal GA problem, different parts of the genotype contribute semi-independently to the evaluation function, and through the Schema Theorem (Holland 1975) progress of some sort can be made from such a random start. Here, however, the genotypes describe control systems which in turn generate behaviour, with no simple correlation between the genotypes and the behaviour; which means that, at least with encoding like the one used here, two different genotypes which both produce promising behaviour will, on recombination, almost always produce a genotype with near-average performance — i.e. useless performance. It is only once the population has largely converged that recombination is likely to be useful.

For this reason, from a start with a randomly generated population, the early stages will do no more than allow some early promising candidate to dominate the population. In this case one can speed up the process, and help give some desired initial direction, by personally observing the first random population, choosing by eye the most promising, and seeding the next generation with clones of this one. Thereafter the population settles down to its asymptotic degree of genetic convergence from above, rather than from below.

For the experiments reported here, an initial randomly generated population of size 30 was judged by eye on the intuitive criterion of 'interesting' behaviour. Two members displayed forward-moving behaviour, which altered in character when the white target was within view of the visual system, and one of these two was selected. The informal criterion of 'interestingness' allowed a clear choice, whereas the 'official' evaluation function used thereafter did not give clear preferences on this initial random population, as the scores it gave there were dominated by noise. This use of different evaluations over time is

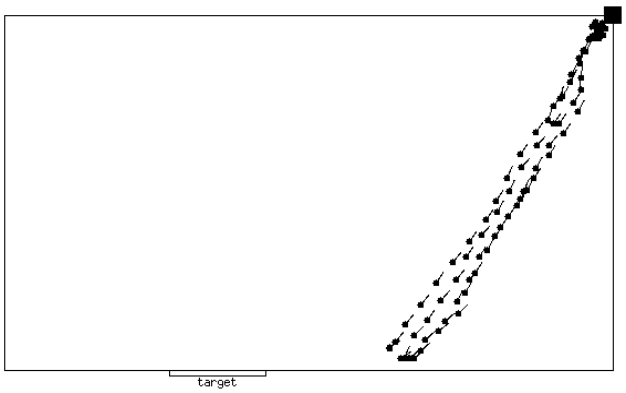

Figure 13.7: *From those evolved for the first task, this is the behaviour of the one best at the 2nd evaluation function — view from above. The dots, and trailing lines, show the front of the robot, and its orientation. Coarsely sampled positions from each of 4 runs are shown, starting in different orientations from the top right corner.*

completely consonant with the underlying philosophy of this approach, of 'co-evolution' between the robots and the human experimenters.

However, the successes achieved with initially converged populations are from too small a sample of experiments to have any statistical significance. It should also be noted that the genetic encoding scheme plays an important role in determining how effective crossover is in early generations.

13.4.4 Big Target

In this next stage, one long gantry wall was covered with white paper, to a width of 150cm and a height of 22cm; the mirror on the robot, which effectively determines the position of the visual inputs, came about 2/3 of the way up on this white wall. The evaluation function $\mathcal{E}_1$ to be maximised implicitly defines a target locating task, which it was hoped would be achieved by visuomotor coordination:

$$\mathcal{E}_1 = \sum_{i=1}^{i=20} Y_i \tag{13.2}$$

where Y_i are the perpendicular distances of the robot from the wall opposite that to which the target is attached, sampled at 20 fixed time intervals throughout a robot trial which lasted a total of about 25 seconds; the closer to the target the higher the score. For each robot architecture 4 trials were run, each starting in the same distant corner, but facing in 4 different directions; these directions were approximately in 4 different quadrants, to give a range of starts facing into obstacle walls as well as towards the target. As the final fitness

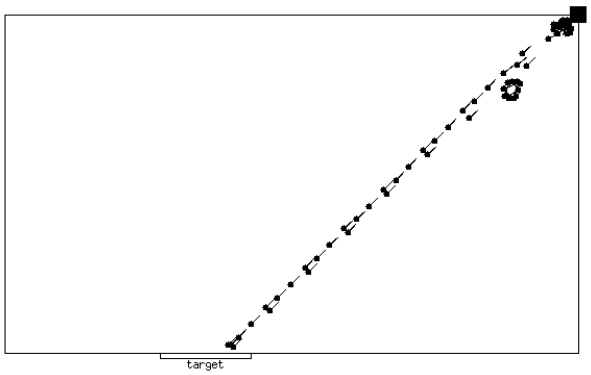

Figure 13.8: *Behaviour of the best of a later generation evolved under 2nd evaluation function. Format as in previous Figure.*

of a robot architecture was based on the *worst* of the 4 trials (to encourage robustness), and since in this case scores accumulated monotonically through a trial, this allowed later trials among the 4 to be prematurely terminated when they bettered previous trials. In addition, as some mutations led to non-moving robots, any that had not moved after the first 6 time intervals were aborted.

In two runs using this method, continuing on from the cloned population chosen on the earlier criterion, very fit individuals appeared in less than 10 generations. From a start close to a corner, they would turn, avoiding contact by vision alone[2]. The best would rotate until the target was in their visual field and then move straight towards it, stopping on hitting the wall when they reached it.

13.4.5 Small Target

The experiment continued from the stage already reached, but now using a much narrower target (22cm) placed about 2/3 of the way along the same wall the large target had been on, and away from the robot's starting corner (see Figure 13.7), with evaluation $\mathcal{E}_2$:

$$\mathcal{E}_2 = \sum_{i=1}^{i=20}(-d_i) \tag{13.3}$$

where d_i is the distance of the robot from the centre of the target at one of the sampled instances during an evaluation run. Again, the fitness of an individual was set to the worst evaluation score from four runs with starting conditions as in the first experiment. The initial population used was the 12th generation from a run of the first experiment (i.e.

[2]They were forced into this by a software error, only discovered later, which meant that all the tactile sensors were turned off. This made this initial task far harder than had been intended.

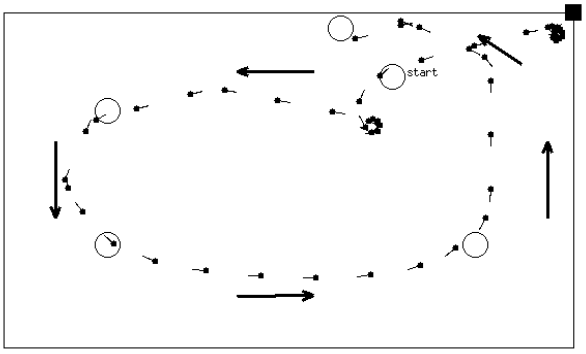

Figure 13.9: *Tracking behaviour of the control system that generated the behaviour shown in previous Figure. The unfilled circles show the position of the target at a number of points on its path (starting position indicated). The arrows roughly indicate the path of the target.*

there was incremental evolution on top of the existing behaviours). The behaviour of the best of this initial population is shown in Figure 13.7. Interestingly, this was not the best at the previous task – that individual did very poorly on the new task.

Within six generations a network architecture/visual morphology had evolved displaying the behaviour shown in Figure 13.8. This control system was tested from widely varying random starting positions and orientations, with the target in different places, and with smaller and different shaped targets. Its behaviour was adaptive enough to cope with all these conditions for which it had not explicitly been evolved.

For comparison a second evolutionary run using $\mathcal{E}_2$ throughout was undertaken; this time $\mathcal{E}_1$, and the big target, were not used as a stepping stone for the first phase. The run started from the same initial converged population as was used for the first task. High scoring individuals emerged after 15 generations. When tested on more general versions of the task their performance was much worse than the best of the incremental run. This result is suggestive, but there is not enough data to be able to report anything statistically significant, about the advantages of doing incremental evolution at this low-level of task.

13.4.6 Moving Target

Following a moving target can be thought of as a generalised version of static target approaching. Hence a number of the evolved small target locators were tested with a white cylinder (of similar width) substituted for the target; this was pushed around the gantry area in a series of smooth movements. The tracking behaviour of the control system

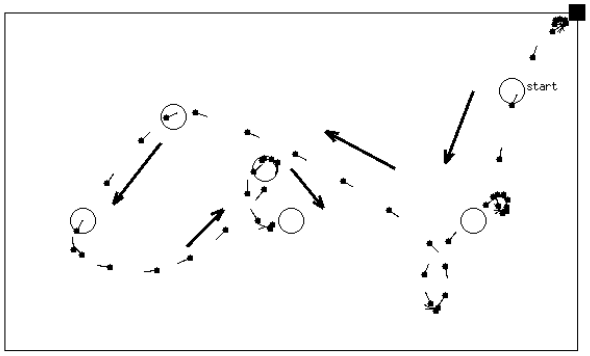

Figure 13.10: *Further tracking behaviour of the control system that generated the behaviour shown in previous Figure.*

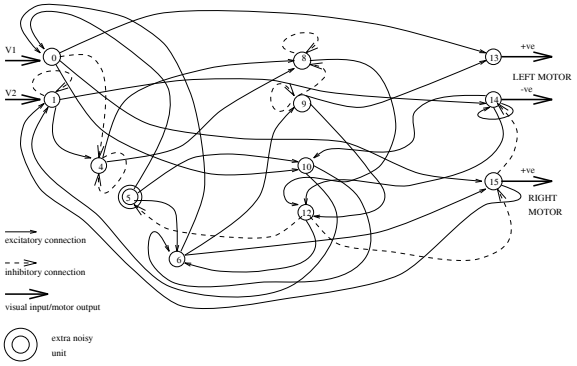

Figure 13.11: *Active network of the best tracker. V1 and V2 are visual inputs from receptive fields 1 and 2.*

that generated the behaviour shown in Figure 13.8 is illustrated in Figures 13.9 and 13.10. To understand how this was achieved, the system could be analysed.

13.4.7 Control System Analysis

In (Husbands *et al.* 1995) it is shown in detail how evolved control systems of the type developed here can be analysed in terms of network dynamics and the way in which the visual morphology couples the control system with the environment. The active part of the network can be characterised in terms of major feedback loops and visual pathways. The active part of the network that generated the behaviours shown in Figures 13.8, 13.9 and 13.10 is shown in Figure 13.11, and its coupled visual morphology is shown in Figure 13.13. On analysis it was seen that in practice only receptive fields 1 and 2 are involved in generating visually guided behaviours.

Due to the same software error mentioned earlier in relation to the tactile sensors,

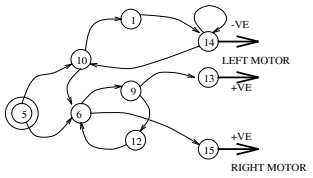

Figure 13.12: *Subnetwork responsible for rotations in absence of visual input.*

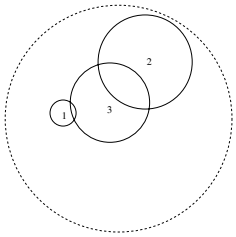

Figure 13.13: *The large dotted circle indicates the extent of the entire visual field available via camera and mirror. The smaller circles indicate the relative positions and sizes of the genetically specified visual receptive fields (no. 3 is not used) for the robot tested on tracking.*

unit 5 (one of the tactile input units) acted as a source of noise over the range [0,0.25]. This unintended property of the unit was exploited by evolution to produce a tightly self-regulating system. Unit 5 feeds into the two coupled feedback loops shown in Figure 13.12. It can be shown that the resulting subnetwork is responsible for generating a noisy turn-on-the-spot behaviour when visual inputs to receptive fields 1 and 2 (*v1* and *v2*) are both low (the robot is facing a dark object). When *v1* is low and *v2* is very high, unit 1 self-inhibits and the same rotational behaviour follows. When *v1* is low and *v2* is medium high the robot rotates in a medium radius circle. When *v1* is high a straight line motion follows. Due to inhibition between motor signals this straight line motion is maintained as long as *v1* is high, irrespective of *v2*.

13.4.8 Rectangles and Triangles

Now, using the population that has become competent with the small target, the task was made significantly more difficult. Two white paper targets were fixed to one of the gantry walls; one was a rectangle 21cm wide and 29.5cm high, the other was an isosceles triangle 21cm wide at the base and 29.5cm high to the apex. The robots were started at four position and orientations near the opposite wall such that they were not biased towards

either of the two targets. The evaluation function $\mathcal{E}_3$ to be maximised was:

$$\mathcal{E}_3 = \sum_{i=1}^{i=20} [\beta(D_{1_i} - d_{1-i}) - \sigma(D_{2_i}, d_{2_i})] \tag{13.4}$$

where D_1 is the distance of target-1 (in this case the triangle) from the gantry origin; d_1 is the distance of the robot from target-1; and D_2 and d_2 are the corresponding distances for target-2 (in this case the rectangle). These are sampled at regular intervals, as before. The value of β is $(D_1 - d_1)$ unless d_1 is less than some threshold, in which case it is $3*(D_1 - d_1)$. The value of σ is zero unless d_2 is less than the same threshold, in which case it is $I - (D_2 - d_2)$, where I is the distance between the targets; I is more than double the threshold distance. High fitnesses are achieved for approaching the triangle but ignoring the rectangle.

After some 10 further generations of a run using as an initial population the last generation of the incremental small target experiment, fit individuals emerged capable of approaching the triangle, but not the rectangle, from each of the four widely spaced starting positions and orientations. The successful networks were of a similar complexity to those shown here. Whereas the fit control systems for the previous experiments only made use of one visual receptive field at a time, those successful at this new task made use of two simultaneously. The visual morphology/networks evolved such that robots rotated on the spot when both visual inputs were low; moved in a straight line when only one visual input was high; and rotated when both inputs were high. The two active receptive fields were so arranged such that when the robot was turning, and its visual field sweeping across the scene horizontally, they would cross a vertical dark/light boundary nearly simultaneously; whereas when crossing a dark/light boundary which slanted from bottom-left to top-right, there was a significant period when one receptive field was in the dark and the other in the light (see Figure 13.14). This was sufficient for the network to switch the motors from rotating the robot to propelling it in a straight line. It would often initially fixate on the edge of the rectangle but as it moved towards it both visual signals would go high, resulting in a rotation towards the triangle.

Hence it would perhaps be more accurate to describe the control system as an 'oblique dark/light boundary detector' rather than a 'triangle detector'. Nevertheless, within the environment in which it was used, it did perform the required task of detecting the triangle, and rejecting the square. This illustrates that tasks such as these can be achieved with extremely minimal vision systems and very small networks.

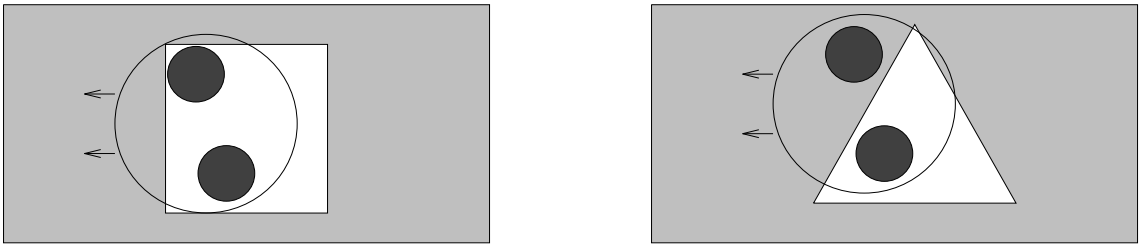

Figure 13.14: *The visual field passing over a square or a triangle*

13.5 Future Work

Encouraged by the initial results with the gantry apparatus more complex experiments are planned. In these it will be possible to use networks with much richer intrinsic dynamics, and more sophisticated genotype to phenotype developmental processes allowing a less restricted open-ended evolutionary process. Behaviours in cluttered and dynamic environments, and under changing lighting conditions, can be explored.

Evaluations with the gantry using a real optic array take less than one order of magnitude longer than the early simulations which used ray-tracing in a very simple environment. But whereas ray-tracing simulations rapidly scale up in computational requirements as the environment is made more complex, with the gantry there is no such constraint.

13.6 Conclusions

This chapter has described a specialised piece of visuo-robotic equipment allowing the artificial evolution of visually guided agents in the real world. The work that has demonstrated that the theory developed in this thesis, and the initial simulation results (Chapter 12), can be carried over into the real world. It has been possible to evolve robust visually guided behaviours with very small populations in very few generations, even though the visual signals in the real world are far more noisy than in the simulations; this is in contrast to the difficulties experienced by others using evolutionary techniques, but with different control system building blocks. A sequence of tasks of increasing complexity has been used, culminating in a combination of control network and visual morphology which allows a triangle to be distinguished from a square, despite the minimal bandwidth vision in a real world system with noisy light values.

A number of the evolved control systems showed interesting levels of adaptation when tested on generalised versions of the task they were evolved for, even though they use only

one or two visual receptive fields and a very small network. In the work to date only the simplest, and least powerful, types of networks, and genotype to phenotype mappings, have been used from the classes that have been advocated; nevertheless promising results have been obtained.

CHAPTER 14

Conclusion

This thesis has attempted to follow through the practical consequences for AI of taking seriously two related major shifts in world view of the last 150 years. Firstly, the death of God (as an explanatory principle for natural phenomena), largely at the hands of Darwinian evolution, which showed how purposeful creatures could arise in a world without a purposeful Creator. Secondly the ultimate failure of science (as a substitute for God) to replace God's view of the universe with an objective impersonal understanding of reality — often proclaimed as the 'scientific' view.

The Darwinian shift is probably accepted in token form by most scientists, though often without any real understanding of its full implications (particularly within AI). The second, anti-objectivist, shift has yet to attain even that token measure of acceptance within much of the scientific and engineering community. I have tried to show here how accepting these two shifts, and taking both of them seriously, has practical consequences for how one sets about building autonomous robots.

Of these two rather grandiose background themes, the Darwinian emphasis has of course been obvious throughout. The anti-objectivist slant has manifested itself in the contrasting of evolution with optimisation, the emphasis on neutral drift, and in the non-representational paradigm advocated for control systems.

In this chapter I shall summarise what I see as the solid achievements of this work; and the aspects most open to criticism. I shall conclude with suggestions for future work.

14.1 The achievements

These I shall divide into those pertinent to the two potential audiences mentioned in Chapter 1: the Genetic Algorithm audience and the Artificial Intelligence audience.

14.1.1 Contribution to GAs

The SAGA framework developed here has been one answer to the call by Kenneth De Jong (1992) and others for GAs to used in a wider domain than just function optimisation; namely, artificial evolution. The critical insight reached here was that in the artificial evolution of ever more complex phenotypes, which in turn requires ever longer genotypes, the inevitable result is that at all times (after the earliest days) the population will be genetically converged (to a large extent). Whereas biologists have always looked at species in the natural world, nobody has previously done so in GAs.

From this basic insight much flowed naturally. The following results have not previously been reported by others in the GA literature:

1. Increases in genotype length should be restricted to gradual ones, appropriate to the correlation length of the underlying fitness landscape (Chapter 6).

2. Mutation rates should be maintained at around one per genotype, subject to the selection pressure (which can be maintained through Tournament or other ranking selection procedures, Chapter 7).

3. Analyses have been given of rank sizes in Tournament selection, with and without noise (Appendices A, B, C).

4. Mutation rates may usefully be fluctuated above and below the critical rate, in a form of annealing (Chapter 7).

5. Recombination between genotypes of varying lengths should be done according to the algorithm presented in Chapter 9, so as to swap as far as possible homologous segments (provided the genotype mapping fits the relevant conditions).

6. Neutral drift of a converged species is possible through sequence space (Chapter 8).

The second, fourth and sixth items here come directly from work in theoretical biology referenced in Chapters 7 and 8, but have not previously been used or reported in GAs. In Chapter 11, application of incremental evolution in the TSP domain showed that it is possible to generate reasonable (though sub-optimal) solutions to a sequence of tour problems of increasing complexity.

In addition, in Chapter 8 an explanation has been given of a puzzling anomaly in (Hinton and Nowlan 1987). The explanation in terms of random genetic drift led to a

discussion of its significance for GAs, and the diffusion approximation analysis of drift given in Appendix E.

14.1.2 Contribution to AI

For AI, the advocacy of artificial evolution is relatively new, despite reference in Turing (1950). Arguments have been presented here that, for the design of control systems for autonomous robots, a major goal for AI, alternative methodologies will hit a complexity barrier which artificial evolution can surmount.

The advocacy of a dynamic systems metaphor, rather than the computational metaphor, for control systems is one that several others have made in recent years (Beer 1992, van Gelder 1992). Indeed, it is a throwback to the days of Cybernetics from the 1940's and 1950's (Ashby 1960, Wiener 1961). It is also one that fits in naturally with Brook's subsumption architecture. Though such ideas appear elsewhere, they deserve the repetition and expansion they have in this thesis. The analysis of how issues of representation (Chapter 4) and time (Chapter 5) have often been dealt with in the Connectionist literature points up the reasons why realtime recurrent dynamical networks have rarely been advocated previously. The discussion of learning in Chapter 5 shows that in the context of artificial evolution it can be treated in a radical new way. The proposal of 'computationally convenient neurons' in Chapter 12 is a minimal class of such dynamical systems, appropriate both for artificial evolution and for implementation on computers.

14.2 Evolutionary Robotics

Growing out of the initial theoretical work for this thesis, application of these ideas to the evolution of control systems for autonomous robots has been demonstrated. These experiments have been focused on mobile wheeled robots for use in office-type environments, performing navigational tasks using low-bandwidth sensors, tactile and visual. Initial experiments with simulations of such robots led to a discussion of the validity of such simulations, and the use of noise.

Then experiments moved to a real-world domain, with the construction of a specialised piece of hardware, the gantry-robot. This allowed the automatic evaluation of populations of mobile robots using real vision in a test environment. The results show that the introduction of artificial evolutionary techniques to robotics can produce interesting, and unanticipated, architectures which succeed at the given tasks, including a sequence of

tasks of increasing complexity..

14.3 The weak points

The history of AI is littered with proposals that have started off with successes in simple toy domains, and then petered out as the complexity of the real world has struck home. The approach of artificial evolution inherently requires that one learns to walk before one can run. Ultimately the only justification for this approach will be impressive results; we do not have them yet.

14.3.1 Resources needed

Despite arguments presented in Chapter 8 that evolution does not have (quite) the enormous effective search space that may be naively thought, it is clear that in the natural world vast resources, in time and in the parallelism of large populations, have been used. The advantages that artificial evolution has (ability to manipulate mutation and selection rates, the ability to take some motor and sensor systems as given) pale in comparison with natural resources. It should be made quite clear that artificial evolution cannot be some magical shortcut, and progress can only be made slowly through many expensive trials.

It can be argued that application of human insight can allow leaps forward with relatively few trials, and hence in some domains this may remain the method of choice for all the conceivable future.

14.3.2 Designing the task sequence

Instead of applying human insight to design, the artificial evolution approach may require just as much insight in designing an appropriate ladder of tasks to be climbed; in the end, nothing may have been gained. Proposals to counter this by use of co-evolution, and sexual selection, are as yet highly speculative and unproven.

14.3.3 Developmental constraints

The discussion in Chapter 10 on morphogenesis gave reasons why it was imperative to have some form of morphogenesis or developmental process between genotype and phenotype, in order to encourage modularity. Although a possible approach was speculatively put forward, nobody has yet demonstrated any developmental system for practical purposes that comes anywhere near to the desired characteristics.

Whereas DNA can indeed be characterised merely as a string of symbols, it is through development using the molecular machinery in which it operates that the final phenotype emerges. This molecular machinery is itself the product of evolution, and we may well have been too naive in assuming that all the important inheritance lies solely in the DNA. Whereas genotypes, as strings of symbols, lend themselves naturally to manipulation in a computer, complex three-dimensional molecular structures do not. Although enormous strides have been made in molecular biology in recent decades, there is still room for radical shifts in perspective, which might imply that much of the power of natural evolution has not yet been understood. As artificial evolution is loosely modelled on natural evolution in a manner which, it is hoped, should capture much of its power, so its assumptions and presumptions are vulnerable to any such radical shift.

14.4 The Future

It is of course appropriate, given the SAGA metaphor, that although only one pathway has led to this point, there are many possible future pathways, not all of which will be taken; artificial evolution will always be work-in-progress. Some promising directions will be briefly examined here.

14.4.1 Theoretical work

The immediate need, as mentioned above, is for developmental systems with the necessary characteristics. For particular purposes it is fairly easy to construct some *ad hoc* system, but without potential for future expansion. The desirable goal would be to have, for a powerful class of control systems defined by the primitives available and their characteristics and connectivity, a developmental system that allowed for arbitrary complexity while preserving the virtues of modularity. Until this barrier is crossed, it seems to me that only limited task domains can be tackled.

In the longer term, it has become increasingly apparent to me as I have done this research that the more one studies evolution the more tricks, and shifts in perspective, and interactions between different conceptual levels there are to be discovered. I had been told this before, but now I truly appreciate it. There is no end in sight to the theoretical analysis that can be done.

14.4.2 Evolutionary Robotics

The research undertaken in the course of this thesis, initially just theoretical, has helped to promote a thriving Evolutionary Robotics Group at Sussex, with research in various directions. One goal is to continue using the gantry described in Chapter 13 to develop more sophisticated navigation abilities than have been demonstrated, here or elsewhere, so far. For instance, given some reasonably cluttered environment, and two separate distinguishable goals at different places, a robot could be given a fixed period of time to 'explore' the domain. Then, on being a specification of just one of the two goals, it will be evaluated on the time it takes to reach it. Success at this task would seem worthy of being called limited navigation. The problems encountered, both in attempting to evolve such behaviour on the gantry, and in transferring any successful behaviours to free mobile robots, will no doubt help shape further experiments.

Within the next few years the technology should be available, at an appropriate price, for radio links to small robots to allow the artificial evolution of a population of free-ranging robots. Our group is already working with a *Khepera* robot, from LAMI at EPFL, Lausanne, Switzerland, which is planned to have these capabilities, plus the ability to autonomously dock and recharge batteries. Once these facilities are available, then co-evolution and sexual selection become practical possibilities. In addition, issues of accuracy of simulations, and transference between simulations and real robots, and between the gantry robot and free robots, will be superseded.

Other promising research being conducted within our group is artificial evolution directly into hardware of control systems for robots, as discussed in Chapter 12 Section 12.2.2. Evolvable hardware, not necessarily restricted to use with robots, is a research topic pursued by groups in Switzerland and Japan as well as Sussex, and what is understood to be the first ever successful application in real hardware was demonstrated here recently (Thompson In Press 1995).

Evolutionary Robotics has in the last year or so become a recognised area of research, with a number of research groups worldwide entering the field. There is now an annual Evolutionary Robotics symposium in Japan, and several conferences and workshops in autonomous robotics hold sessions centred on this area. However, although many groups are applying genetic algorithm techniques to the design of such control systems, as yet they have not in general taken on board the distinctions between optimisation and evolution which have been central to this thesis.

14.4.3 Commercial prospects

The incremental nature of evolution, both natural and artificial, means that any new adaptation builds on the inheritance accumulated from all its ancestors, without which it would have been inconceivable — literally as well as metaphorically. Such an inheritance can only be achieved through enormous numbers of trials, typically expensive. To achieve a complex system capable of coping with (in order of difficulty) tasks A–K will have cost a lot; to go further from K–L will cost significantly less than starting from the beginning and achieving all the capabilities A–L. This is equally true in the world of, for example, aircraft design, where nowadays no aircraft manufacturer could start up without buying in expertise derived from other's design and manufacturing experience.

Accumulated experience in artificial evolution can be expressed, given some specific genotype-to-phenotype mapping, simply in the string of numbers that form the genotype. This provides a basis for, and also limits and constrains, future pathways of evolution. The value of such a genotype can easily be protected by copyright or patent laws, and licensed for use by others. Future progeny can readily be identified as related by use of algorithms such as those discussed in Chapter 9. Given the arguments of Chapter 3, it seems to me probable that in the next century much design work for complex systems — not just robot control systems, but aircraft design, computer chip design, etc. — will start to be done by artificial evolution. The commercial prospects for any firm that achieves a head start in some field, and then licenses its genotypes for use by others with a licence fee payable on all progeny, could be phenomenal. Such a licence fee could be insignificant on any one unit; as evolution continues it could have additional small fees added for the benefit of those who have added value; long-term returns on all future progeny could nevertheless be enormous. As other firms build on and add value, for their own benefit, to the original genotypes, the commercial position of those genotypes relative to any competition becomes further strengthened — the founder effect translated directly to the commercial world.

For theoretical ideas to be transferred to the practical domain, not only must they work but also there should be strong motivations for them to be applied. In the field of biotechnology the application of artificial evolution techniques has recently sprung into prominence, and promise to mushroom further. If some of the underlying premises of this thesis are correct, then something similar can be anticipated in such engineering areas as complex control systems.

14.5 A Final Thought

Arguments by Gott (1993), based on Copernican principles produce estimates on how long our own species can be expected to survive. If we assume that *homo sapiens* is about 200,000 years old, and that our position today as intelligent observers belonging to that species is not privileged, then it follows that we can predict at the 95% confidence level that our species will last between about 5,000 and 8 million years more, compared with some 4 billion years since life began. We are only temporary inhabitants of this planet. Evolutionary theory can help to explain the past, but can provide very little prognosis for the future.

APPENDIX A

Rank sizes with long genotypes

For context, see Chapter 7, Section 7.5.1.

Consider a population of size N, with binary genotypes, and classify each individual by Hamming-distance from the current master sequence or wild-type, which without loss of generality can be taken to be a genotype of all 0s. It is assumed that fitness monotonically decreases with this distance, and without loss of generality the negative fitness can be the number of 1s in the genotype. At each replication there is a single mutation, and the genotypes are so long that the chance of a back-mutation is negligible; i.e. all mutations are deleterious.

The number of individuals with i 1s is defined as r_i. The loss from and gain to this class should balance at equilibrium.

Something of rank i is lost in a tournament if there is a tournament between a rank i and and a rank $< i$, with probability

$$2\frac{r_i}{N}\sum_{j=0}^{i-1}\frac{r_j}{(N-1)},$$

and also for one between two different members of rank i, probability

$$\frac{r_i}{N}\frac{(r_i-1)}{(N-1)}.$$

Something of rank i is gained in a tournament between rank $(i-1)$ and a rank $> i$, with probability

$$2\frac{r_{i-1}}{N}\sum_{j=i+1}^{N}\frac{r_j}{(N-1)},$$

and also for one between two different members of rank $(i-1)$, probability

$$\frac{r_{i-1}}{N}\frac{(r_{i-1}-1)}{(N-1)}.$$

Setting gains equal to losses, and eliminating $N(N-1)$ we have

$$2r_i\sum_{j=0}^{i-1}r_j + r_i(r_i-1) = 2r_{i-1}\sum_{j=i+1}^{N}r_j + r_{i-1}(r_{i-1}-1).$$

This gives a value for r_i based on values for lower is. Since we know that $r_0 = 1$, we have

$$2r_1 + r_1(r_1 - 1) = 2(N - 1 - r_1).$$

This quadratic equation yields a positive solution of

$$r_1 = \frac{1}{2}(-3 + \sqrt{8N + 1}).$$

For $N = 100$ this gives $r_1 \approx 12.65$. Successive values for higher i can be found by iteratively solving successive quadratics.

APPENDIX B

Noisy decisions in an infinite population

For context, see Chapter 7, Section 7.5.1.

Consider a similar situation to that in appendix A, but where there is a probability p of correctly deciding a tournament, and hence $(1-p)$ of making a mistake. We will be considering the long-term possibility of losing all members of rank 0, and will assume a population of infinite size so as to ignore stochastic effects of genetic drift. Let the proportion of the population in rank 0, the master sequence, be a. To keep up the value of a, we will impose a probability q of mutating on replication, and hence $(1-q)$ of there being no mutation.

A wild-type of rank 0 will be lost from the population when rank 0 meets rank 0 and there *is* a mutation, probability a^2q; and also when rank 0 meets rank ≥ 1 and the wrong one wins, probability $2a(1-a)(1-p)$.

A rank 0 will be gained when a rank 0 meets a rank ≥ 1, the right one wins, and there is *no* mutation, probability $2a(1-a)p(1-q)$.

Setting gains equal to losses, and dividing by a, we have

$$aq + 2(1-a)(1-p) = 2(1-a)p(1-q)$$

$$a(4p + q - 2pq - 2) = 2(2p - pq - 1)$$

We can assume that $p > 0.5$, say $p = 0.5 + s$ for positive s. The factor on the l.h.s. of the equation, multiplying a, then becomes $(2 + 4s + q - q - 2sq - 2)$, which reduces to $2s(2-q)$. We know that s is positive and $(2-q)$ is positive, so the condition for a to be positive is that the r.h.s. of the above equation is also.

Hence $2(2p - pq - 1) > 0$, which gives $q < (2p-1)/p$ as the condition for the proportion a of rank 0 to remain positive.

APPENDIX C

Mutations assessed independently at each locus

For context, see Chapter 7, Section 7.5.2.

Consider an population of N binary genotypes each of length l. N is assumed to be large enough to avoid stochastic genetic drift, and l is assumed large enough for approximations to be made below.

If there is on average μ mutations per genotype, this is a probability of μ/l at each locus. So the chance of there being no mutation at all l loci on a genotype is $(1 - \mu/l)^l$. For small μ and large l this is close to $e^{-\mu}$.

Let the proportion of rank 0 in the population be a. Then a rank 0 will be lost to the population when rank 0 meets rank 0 and there *is* a mutation, giving a probability of

$$a^2 \left(1 - e^{-\mu}\right),$$

a rank 0 will be gained when a rank 0 meets a rank ≥ 1 and there is *no* mutation, a probability of

$$2a(1 - a)e^{-\mu}.$$

Setting gains equal to losses, and multiplying through, we have

$$a(e^{\mu} - 1) = 2(1 - a).$$

Hence the proportion of the population expected to be the wild-type at equilibrium is:

$$a = \frac{2}{1 + e^{\mu}}.$$

Let us now reconsider this scenario with noise added, when the winner of a tournament is selected with probability p. A rank 0 will still be lost to the population when rank 0 meets rank 0 and there *is* a mutation, giving a probability of

$$a^2 \left(1 - e^{-\mu}\right),$$

but a rank 0 will *also* be lost when rank 0 meets rank ≥ 1, and noise makes it lose; probability is

$$2(1-p)a(1-a).$$

A rank 0 will be gained when a rank 0 meets a rank ≥ 1, wins, and there is *no* mutation, a probability of

$$2pa(1-a)e^{-\mu}.$$

Setting gains equal to losses, and multiplying through, we have

$$a(e^{\mu} - 1 + 2e^{\mu}(1-p)(1-a)) = 2p(1-a).$$

Hence

$$a(e^{\mu}(2p-1) - 1) = 2p - 2(1-p)e^{\mu}.$$

The contents of the bracket on the l.h.s. are always positive, so that the condition for $a > 0$ is that the r.h.s. is positive,

$$p > (1-p)e^{\mu}.$$

Hence for preservation of the wild-type we need

$$p > \frac{e^{\mu}}{1+e^{\mu}}.$$

The proportion of the population expected to be the wild-type at equilibrium is:

$$a = \frac{2}{1+e^{\mu}}.$$

APPENDIX D

The Hinton & Nowlan model — Expected fitness of potential winner

For context see Chapter 8, Section 8.4.

To calculate the expected fitness of a genotype composed of q question-marks and $(20-q)$ **1**s.

Define $p = 1/2^q$ prob. success on one trial.

$r = 1-p$ prob. failure on one trial.

$R = r^{1000}$ prob. failing all 1000 trials.

If success comes on the ith trial, for $i \leq 1000$, then the actual fitness is then given by $1 + 19(1000-i)/1000$. The chance of first succeeding on the ith trial, which necessitates failing the preceding $(i-1)$ trials, is given by $r^{i-1}p$.

Hence the expected fitness $F(q)$, bearing in mind the chance R of failing all 1000 trials with a resulting fitness of 1, is given by:

$$\begin{aligned} F(q) &= R + \sum_{i=1}^{1000} r^{i-1} p \left(1 + \frac{19(1000-i)}{1000}\right) \\ &= R + 20p \sum_{i=1}^{1000} r^{i-1} - \frac{19p}{1000} \sum_{i=1}^{1000} i r^{i-1} \end{aligned}$$

But we can use:

$$\sum_{i=1}^{1000} r^{i-1} = \frac{1-r^{1000}}{1-r}$$

and by multiplying each side by r and then differentiating w.r.t. r:

$$\begin{aligned} \sum_{i=1}^{1000} i r^{i-1} &= \frac{d}{dr}\left(\frac{r(1-r^{1000})}{1-r}\right) \\ &= \frac{1 - 1001 r^{1000} + 1000 r^{1001}}{(1-r)^2} \end{aligned}$$

Substituting we get:

$$F(q) =$$

$$R + 20p\frac{(1-r^{1000})}{1-r} - \frac{19p(1-1001r^{1000}+1000r^{1001})}{1000(1-r)^2}$$

$$= 20 - \frac{19(1-r^{1000})}{1000(1-r)}$$

This is used to calculate the figures in table 8.1, although care must be taken with the precision in computing as very small numbers are involved in the intermediate calculations.

D.1 Expected number of winners at start

In a member of the initial random population, the probability of having no **0**s, i.e. of being a potential winner, is $(3/4)^{20}$. Such a member will be all **?**s and **1**s, with **?**s being twice as likely as **1**s at any locus.. The chance of having exactly q **?**s in a potential winner, given by the binomial expansion of $(2/3 + 1/3)^{20}$, is

$$\left(\begin{array}{c} 20 \\ q \end{array}\right) \left(\frac{2}{3}\right)^q \left(\frac{1}{3}\right)^{20-q}$$

and then the probability of actual success is $1 - (1 - 0.5^q)^{1000}$.

So the probability of this initial random member being a winner is

$$\left(\frac{3}{4}\right)^{20} \sum_{q=0}^{20} \left(\begin{array}{c} 20 \\ q \end{array}\right) \left(\frac{2}{3}\right)^q \left(\frac{1}{3}\right)^{20-q} (1 - (1 - 0.5^q)^{1000})$$

$$\simeq 0.000558$$

This figure differs from the value 0.028 given in (Belew 1989). The probability of there being no winner in an initial random population of 1000 is $(1 - 0.000558)^{1000} \simeq 0.572$.

APPENDIX E

The Diffusion approximation analysis of Genetic Drift

For context see Chapter 8, Section 8.5.

In the analysis of a physical system of diffusion we let $\rho(x,t)$ denote the density of particles at location x at time t. (The translation to our ensemble of populations is: let $\rho(x,t)$ denote the proportion of populations in the ensemble that at time t give a census return of x for the proportion of 1s at the relevant locus.) The flow across a surface at x is $J(x,t)$. The change in density at a location is equal to the spatial derivative of the flow.

$$\frac{\partial}{\partial t}\rho(x,t) = -\frac{\partial}{\partial x}J(x,t) \tag{E.1}$$

In the current context the expression for $J(x,t)$ contains a term for external forces — mutation and selection — and another term for diffusion.

$$J(x,t) = M(x)\rho(x,t) - \frac{1}{2}\frac{\partial}{\partial x}V(x)\rho(x,t) \tag{E.2}$$

In a short time interval Δt, $M(x)\Delta t$ is the average distance travelled from a point x under force of mutation and selection. $V(x)\Delta t$ is the variance of the distances travelled.

We are interested in the equilibrium distribution $\hat{\rho}(x)$, where it exists. At equilibrium $J(x)$ in (E.2) will be constant, and in this context zero. Hence

$$\frac{1}{2}\frac{d}{dx}V(x)\hat{\rho}(x) = M(x)\hat{\rho}(x) \tag{E.3}$$

Introduce $g(x) \equiv V(x)\hat{\rho}(x)$.

$$\begin{aligned}
\frac{1}{2}\frac{d}{dx}g(x) &= \frac{M(x)}{V(x)}g(x) \\
\frac{1}{g(x)}dg(x) &= 2\frac{M(x)}{V(x)}dx \\
d\ln[g(x)] &= 2\frac{M(x)}{V(x)}dx \\
\ln[g(x)] &= 2\int\frac{M(x)}{V(x)}dx + constant
\end{aligned}$$

$$
\begin{aligned}
g(x) &= c \exp\left(2 \int \frac{M(x)}{V(x)} dx\right) \\
\hat{\rho}(x) &= \frac{c}{V(x)} \exp\left(2 \int \frac{M(x)}{V(x)} dx\right)
\end{aligned} \tag{E.4}
$$

where c is an appropriate normalizing constant to make the area under $\hat{\rho}(x)$ equal to 1 in the range of x from 0 to 1.

So far the calculations have followed Roughgarden (Roughgarden 1979) exactly, but now we make adjustments appropriate for a haploid GA. With a mutation rate of m then Δx due to mutation in one generation is

$$
\Delta x_{mut} = x(-m) + (1-x)m = m(1-2x) \tag{E.5}
$$

To calculate the Δx in one generation due to selection, we shall assume that the schema fitness of the allele **0** is f_0 and of allele **1** is f_1. The average fitness $\overline{f}$ depends on the proportion x of **1**s in the current population, $\overline{f} = (1-x)f_0 + x f_1$. We shall define the selective force s in favour of allele **1** as

$$
s \equiv \frac{f_1 - f_0}{f_0} \tag{E.6}
$$

$$
\Delta x_{sel} = x\left(\frac{f_1}{\overline{f}} - 1\right) = \frac{sx(1-x)}{1+sx} \tag{E.7}
$$

Using the population size N, we now convert Δx to a new time scale where N generations equals one unit of time.

$$
\begin{aligned}
M(x) &= N(\Delta x_{mut} + \Delta x_{sel}) \\
&= mN(1-2x) + \frac{sNx(1-x)}{1+sx}
\end{aligned} \tag{E.8}
$$

On calculating $V(x)$ we use the fact that the variance of Δx over one generation is $x(1-x)/N$. Converting to the same time scale as above we have

$$
V(x) = N\frac{x(1-x)}{N} = x(1-x) \tag{E.9}
$$

Substituting (E.8) and (E.9) into (E.4) we have the following (the constants on integration can be assimilated into the normalizing constant c):

$$
\hat{\rho}(x) = \frac{c}{x(1-x)} \exp\left(W(x)\right)
$$

where we define $W(x)$ to be

$$
\begin{aligned}
&\equiv && 2N \int \left(\frac{m(1-2x)+\frac{sx(1-x)}{1+sx}}{x(1-x)} \right) dx \\
&= && 2mN \int \frac{dx}{x(1-x)} - 4mN \int \frac{dx}{1-x} + 2sN \int \frac{dx}{1+sx} \\
&= && -2mN \ln\left(\frac{1-x}{x}\right) + 4mN \ln(1-x) + 2N \ln(1+sx)
\end{aligned}
\tag{E.10}
$$

So we have $\hat{\rho}(x)$

$$
\begin{aligned}
&= && \frac{c}{x(1-x)} \left(\frac{1-x}{x}\right)^{-2mN} (1-x)^{4mN} (1+sx)^{2N} \\
&= && cx^{2mN-1}(1-x)^{2mN-1}(1+sx)^{2N} \\
&= && \frac{c(1+sx)^{2N}}{[x(1-x)]^{1-2mN}}
\end{aligned}
\tag{E.11}
$$

APPENDIX F

C code for crossover algorithm

For context refer to Chapter 9.

```
#include <stdio.h>
#include <string.h>
#define MAXLEN 1000  /* Max length of genes */
#define max(a,b) ((a)>(b) ? a : b)

/**********************************
A and B contain genes as char strings
as, ae are start and end of substring of A
lenb is the length of B
L is the output vector mentioned in text
**********************************/
algb(A,as,ae,B,lenb,L)
char *A,*B;
int *L;
int as,ae,lenb;
{
  int this,last=0;
    /* this/last alternate between 0 and 1 to
       identify different rows in K[][]     */
  int fwd;
    /* flag to identify whether we are running
       forwards or backwards along string A */
  int i,j;
  int K[2][MAXLEN+1];
```

```
    /* array described in text              */

  fwd=(ae>as ? 1 : -1);
    /* set fwd flag                         */

  for (j=0;j<lenb+1;j++)
    K[1][j]=0;           /* clear row of K  */

  for (i=as;i*fwd<ae*fwd;i+=fwd)
    /* runs backwards if nec */
  {
    last=1-(this=last); /* flip this/last   */
    for (j=0;j<lenb;j++)
      K[this][j+1]=
        (A[i]==B[(fwd==1 ? j : lenb-j-1)] ?
                /* are the chars matching ? */
          K[last][j]+1 :          /* yes or */
          max(K[this][j],K[last][j+1]));
                                        /* no */
  }

    /* internal calculations finished;
       copy into output                     */
  for (j=0;j<lenb+1;j++)
    L[j]=K[this][j];
}
```

```
/**********************************
A and B are char strings for the genes, of lengths lena and lenb.
cross1 is the selected crossover point in A.
algd will return the proposed position for crossover point in B
**********************************/
int algd(A,B,lena,lenb,cross1)
char *A,*B;
int lena,lenb,cross1;
{
  int L1[MAXLEN+1],L2[MAXLEN+1];    /* used by algb                        */
  int best=0;                       /* keep track of best                  */
  int numbest=0;                    /* and how many equal-best             */
  int temp,i;

  algb(A,0,cross1,B,lenb,L1);       /* left part of A, and all B           */
  algb(A,lena-1,cross1-1,B,lenb,L2);/* right part of A, all B, BOTH BKWDS */

    /* Now go through keeping track of max of L1[i]+L2[lenb-i].           */
  for (i=0;i<=lenb;i++)
  {
    temp=L1[i]+L2[lenb-i];
    if (temp>best)
      {numbest=0; best=temp;}
    if (temp==best)
      L1[numbest++]=i;
  }

    /* Now choose at random from the best   */
  return L1[random() % numbest];
}
```

```
/*********************************
Test program to read in file containing 2
gene strings, choose a random crossover point
in first, and select appropriate crossover
point in second.
*********************************/
main()
{
  FILE *fp;
  char gene1[MAXLEN]; /* strings for genes */
  char gene2[MAXLEN];
  int len1,len2;      /* lengths of genes  */
  int i,j,k,displace1,displace2;
  int cross1,cross2;  /* crossover points  */

  fp=fopen("genefile","r");
  fscanf(fp,"%s",gene1);
  fscanf(fp,"%s",gene2);
  fclose(fp);

  len1=strlen(gene1);
  len2=strlen(gene2);

    /* make sure cross1 is within gene1    */
  cross1=1+(random()%(len1-1));
  cross2=algd(gene1,gene2,len1,len2,cross1);

  printf("\n%d   %d\n",cross1,cross2);
}
```

BIBLIOGRAPHY

[Abeles 1982] M. Abeles. *Local Cortical Circuits, An Electrophysiological study.* Springer-Verlag, 1982.

[Ackley and Littman 1991] D. H. Ackley and M. L. Littman. Interactions between learning and evolution. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life II: Proceedings Volume of Santa Fe Conference Feb. 1990*, pages 487–509, Redwood City CA, 1991. Addison Wesley: volume XI in the series of the Santa Fe Institute Studies in the Sciences of Complexity.

[Agre and Chapman 1990] P.E. Agre and D. Chapman. What are plans for? In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 17–34. MIT/Elsevier, 1990.

[Appel and Haken 1989] K. Appel and W. Haken. Every planar map is four colorable. *American Mathematical Society, Contemporary Mathematics*, 98, 1989.

[Ashby 1960] W. Ross Ashby. *Design for a Brain.* Chapman, 1960.

[Baker 1987] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, Hillsdale NJ, 1987. Lawrence Erlbaum Associates.

[Baldwin 1896] J. M. Baldwin. A new factor in evolution. *American Naturalist*, 30:441–451, 1896.

[Ballard 1991] D. H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–86, 1991.

[Barhen *et al.* 1987] J. Barhen, W.B. Dress, and C.C. Jorgensen. Applications of concurrent neuromorphic algorithms for autonomous robots. In R. Eckmiller and C.v.d. Malsburg, editors, *Neural Computers*, pages 321–333. Springer-Verlag, 1989.

[Beer and Gallagher 1992] R. D. Beer and J. C. Gallagher. Evolving dynamic neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, 1992.

[Beer 1990] R. D. Beer. *Intelligence as Adaptive Behaviour: An Experiment in Computational Neuroethology.* Academic Press, San Diego CA, 1990.

[Beer 1992] R.D. Beer. A dynamical systems perspective on autonomous agents. Technical Report CES-92-11, Case Western Reserve University, Cleveland, Ohio, 1992.

[Belew and Booker 1991] R. K. Belew and L. B. Booker, editors. *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann.

[Belew 1989] R. K. Belew. When both individuals and populations search: Adding simple learning to the genetic algorithm. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 34–41, San Mateo CA, 1989. Morgan Kaufmann.

[Boden 1990] M.A. Boden, editor. *The Philosophy of Artificial Intelligence.* Oxford University Press, 1990.

[Brady *et al.* 1982] M. Brady, J.M. Hollerbach, T.C. Johnson, T. Lozano-Perez, and M.J. Mason, editors. *Robot Motion: Planning and Control.* MIT Press, Cambridge MA, 1982.

[Brooks 1986] R. A. Brooks. Achieving artificial intelligence through building robots. A.I. Memo 899, MIT AI Lab, May 1986.

[Brooks 1990] R. A. Brooks. The behavior language; user's guide. Technical Report A.I. Memo 1227, MIT AI Lab., April 1990.

[Brooks 1991] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[Brooks 1992] Rodney A. Brooks. Artificial life and real robots. In *Proceedings of the First European Conference on Artificial Life.* MIT Press/Bradford Books, Cambridge, MA, 1992.

[Cariani 1989] P. Cariani. *On the Design of Devices with Emergent Semantic Functions.* PhD thesis, State University of New York at Binghamton, 1989.

[Chapman 1987] David Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.

[Cliff *et al.* 1993a] D. Cliff, I. Harvey, and P. Husbands. Explorations in evolutionary robotics. *Adaptive Behavior*, 2(1):71–104, 1993.

[Cliff *et al.* 1993b] D. T. Cliff, I. Harvey, and P. Husbands. Evolved recurrent dynamical networks use noise. In *Proceedings of Intl. Conference on Artificial Neural Networks, ICANN'93*. Amsterdam, Sept. 1993, 1993.

[Cliff *et al.* 1993c] D. T. Cliff, I. Harvey, and P. Husbands. Incremental evolution of neural network architectures for adaptive behaviour. In M. Verseylen, editor, *Proceedings of the First European Symposium on Artificial Neural Networks, ESANN'93*, pages 39–44. D facto Publishing, Brussels, 1993.

[Cliff *et al.* 1993d] D. T. Cliff, I. Harvey, and P. Husbands. Visual sensory-motor networks without design: Evolving visually guided robots. In B. Svensson, editor, *Proceedings of the International Workshop on Mechatronical Computer Systems for Perception and Action (MCPA93)*, pages 269–278, Sweden, 1993. CCA/Högskolan Press.

[Cliff *et al.* 1993e] D. T. Cliff, P. Husbands, and I. Harvey. Analysis of evolved sensory-motor controllers. In *Proceedings of Second European Conference on Artificial Life, ECAL93*. Brussels, May 1993, 1993.

[Cliff *et al.* 1993f] D. T. Cliff, P. Husbands, and I. Harvey. Evolving recurrent dynamical networks for robot control. In R. Albrecht, C. Reeves, and N. Steele, editors, *Proceedings of ANNGA93, the Intl. Conf. on Neural Networks and Genetic Algorithms*, pages 428–435. Springer-Verlag, 1993.

[Cliff *et al.* 1993g] D. T. Cliff, P. Husbands, and I. Harvey. Evolving visually guided robots. In J.-A. Meyer, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 3, Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92)*, pages 374–383. MIT Press/Bradford Books, Cambridge MA, 1993.

[Cliff *et al.* 1993h] D. T. Cliff, P. Husbands, and I. Harvey. General visual robot controller networks via artificial evolution. In *Proceedings of SPIE93, Society of Photo-optical Instrumentation Engineers*. Boston MA, Sept. 1993, 1993.

[Cliff 1991a] D. Cliff. The computational hoverfly: A study in computational neuroethology. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats: Proceedings of The First International Conference on Simulation of Adaptive Behavior*, pages 87–96. MIT Press/Bradford Books, Cambridge, MA, 1991.

[Cliff 1991b] D. T. Cliff. Computational neuroethology: A provisional manifesto. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats: Proceedings of The First International Conference on Simulation of Adaptive Behavior*, pages 29–39. MIT Press/Bradford Books, Cambridge, MA, 1991.

[Conrad 1988a] M. Conrad. The price of programmability. In R. Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 285–307. Oxford University Press, 1988.

[Conrad 1988b] M. Conrad. Prolegomena to evolutionary programming. In M. Kochen and H. M. Hastings, editors, *Advances in Cognitive Science: Steps Toward Convergence*, pages 150–168, Boulder, Colorado, 1988. AAAS Selected Symposia Series, Westview Press.

[Cramer 1985] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In J.J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Hillsdale NJ, 1985. Lawrence Erlbaum Associates.

[Crick and Asanuma 1986] F. Crick and C. Asanuma. Certain aspects of the anatomy and physiology of the cerebral cortex. In J. L. McClelland and D. E. Rumelhart, editors, *Parallel Distributed Processing, Vol. 2*, chapter 20, pages 333–371. MIT Press/Bradford Books, 1986.

[Darwin 1859] C. Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* Murray, London, 1859.

[Davidor 1990] Yuval Davidor. Epistasis variance: A viewpoint on representations, GA hardness, and deception. *Complex Systems*, 4(4), 1990.

[Davis 1987] L. Davis, editor. *Genetic Algorithms and Simulated Annealing.* Pitman Press, London, 1987.

[de Garis 1992] Hugo de Garis. The genetic programming of steerable behaviors in GenNets. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 272–281. MIT Press/Bradford Books, Cambridge, MA, 1992.

[De Jong and Spears 1990] K. De Jong and W. M. Spears. An analysis of the interacting roles of population size and crossover in genetic algorithms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, pages 38–47. Springer-Verlag, 1990.

[De Jong 1975] K. De Jong. *Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, Univ. of Michigan, 1975.

[De Jong 1987] Kenneth De Jong. On using genetic algorithms to search program spaces. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale NJ, 1987. Lawrence Erlbaum Associates.

[De Jong 1992] K. A. De Jong. Are genetic algorithms function optimizers? In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 3–13. North-Holland, 1992.

[Dreyfus 1972] H.L. Dreyfus. *What Computers can't do: a Critique of Artificial Reason*. Harper, New York, 1972.

[Edelman 1989] G. M. Edelman. *Neural Darwinism*. Oxford University Press, 1989.

[Eigen and Schuster 1979] M. Eigen and P. Schuster. *The Hypercycle: A Principle of Natural Self-Organization*. Springer-Verlag, 1979.

[Eigen *et al.* 1988] M. Eigen, J. McCaskill, and P. Schuster. Molecular quasi-species. *Journal of Physical Chemistry*, 92:6881–6891, 1988.

[Farmer *et al.* 1985] J.D. Farmer, A. Lapedes, N. Packard, and B. Wendroff, editors. *Evolution, Games and Learning: Models for Adaptation in Machines and Nature. Proceedings of the 5th Annual International Conference of the Center for Nonlinear Studies, Los Alamos, N.M.*, May 1985. Physica D, v.22, 1986.

[Fogel *et al.* 1966] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, 1966.

[Fogel 1993] D. B. Fogel. Evolving behaviors in the Iterated Prisoner's Dilemma. *Evolutionary Computation*, 1(1):77–97, 1993.

[Forrest 1989] S. Forrest, editor. *Emergent Computation: Self-organizing, Collective and Cooperative Phenomena in Natural and Artificial Computing Networks. Proceedings of the 9th Annual International Conference of the Center for Nonlinear Studies, Los Alamos, N.M.*, 1989. Physica D, v.42, June 1990.

[Forrest 1993] S. Forrest, editor. *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

[Frean 1989] Marcus Frean. The Upstart algorithm : a method for constructing and training feed-forward neural networks. Technical Report 89/469, Physics Department, Edinburgh University, 1989.

[Fujiki and Dickinson 1987] Cory Fujiki and John Dickinson. Using the genetic algorithm to generate LISP source code to solve the Prisoner's Dilemma. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale NJ, 1987. Lawrence Erlbaum Associates.

[Gallagher and Beer 1993] J. C. Gallagher and R. D. Beer. A qualitative analysis of evolved locomotion controllers. In J.-A. Meyer, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92)*, pages 71–80. MIT Press/Bradford Books, Cambridge, MA, 1993.

[Gillespie 1984] J.H. Gillespie. Molecular evolution over the mutational landscape. *Evolution*, 38:1116, 1984.

[Goldberg and Deb 1990] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. Technical Report TCGA-90007, TCGA, The University of Alabama, 1990.

[Goldberg and Segrest 1987] David E. Goldberg and Philip Segrest. Finite markov chain analysis of genetic algorithms. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale NJ, 1987. Lawrence Erlbaum Associates.

[Goldberg *et al.* 1990] David E. Goldberg, K. Deb, and B. Korb. An investigation of messy genetic algorithms. Technical Report TCGA-90005, TCGA, The University of Alabama, 1990.

[Goldberg 1989a] D. E. Goldberg. Zen and the art of genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 80–85, San Mateo CA, 1989. Morgan Kaufmann.

[Goldberg 1989b] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading MA, 1989.

[Goodwin 1990] B. C. Goodwin. The evolution of generic form. In J. Maynard Smith and G. Vida, editors, *Organizational constraints on the dynamics of evolution*, Proceedings in Nonlinear Science, pages 107–117. Manchester University Press, 1990.

[Goss 1993] S. Goss, editor. *Proceedings of Second European Conference on Artificial Life, ECAL93*. Brussels, May 1993, 1993.

[Gott 1993] J. Richard Gott. Implications of the Copernican principle for our future prospects. *Nature*, 363(27 May):315–319, 1993.

[Gould and Vrba 1981] S. J. Gould and E. S. Vrba. Exaptation — a missing term in the science of form. *Palaeobiology*, 8(1):4–15, 1981.

[Grefenstette 1983] J.J. Grefenstette. A user's guide to GENESIS. Technical Report CS-83-11, Computer Science Department, Vanderbilt University, 1983.

[Grefenstette 1985] J. J. Grefenstette, editor. *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Hillsdale NJ, 1985. Lawrence Erlbaum Associates.

[Grefenstette 1987] J. J. Grefenstette, editor. *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale NJ, 1987. Lawrence Erlbaum Associates.

[Gruau 1992] F.C. Gruau. Cellular encoding of genetic neural networks. Technical Report 92-21, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, May 1992.

[Gruau 1993] F. Gruau. A learning and pruning algorithm for genetic boolean neural networks. In M. Verleysen, editor, *Proceedings of the First European Symposium on Artificial Neural Networks, ESANN'93*, Brussels, 1993. D facto Publishing.

[Harnad 1989] S. Harnad. Minds, machines and Searle. *Journal of Theoretical and Experimental Artificial Intelligence*, 1:5–25, 1989.

[Harp and Samad 1991] S.A. Harp and T. Samad. Genetic synthesis of neural network architecture. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 202–221. Van Nostrand Reinhold, 1992.

[Harp *et al.* 1989] Steven A. Harp, T. Samad, and A. Guha. Towards the genetic synthesis of neural networks. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo CA, 1989. Morgan Kaufmann.

[Harvey *et al.* 1993a] I. Harvey, P. Husbands, and D. T. Cliff. Genetic convergence in a species of evolved robot control architectures. In S. Forrest, editor, *Genetic Algorithms: Proceedings of Fifth Intl. Conference*, page 636, San Mateo CA, 1993. Morgan Kaufmann.

[Harvey *et al.* 1993b] I. Harvey, P. Husbands, and D. T. Cliff. Genetic convergence in a species of evolved robot control architectures. Technical Report CSRP 267, School of Cognitive and Computing Sciences, Univ. of Sussex, 1993.

[Harvey *et al.* 1993c] I. Harvey, P. Husbands, and D. T. Cliff. Issues in evolutionary robotics. In J.-A. Meyer, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92)*, pages 364–373. MIT Press/Bradford Books, Cambridge MA, 1993.

[Harvey *et al.* 1994] I. Harvey, P. Husbands, and D. T. Cliff. Seeing the light: Artificial evolution, real vision. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour (SAB94)*, pages 392–401. MIT Press/Bradford Books, Cambridge MA, 1994.

[Harvey 1991] Inman Harvey. The artificial evolution of behaviour. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats: Proceedings of The First International Conference on Simulation of Adaptive Behavior*, pages 400–408. MIT Press/Bradford Books, Cambridge, MA, 1991.

[Harvey 1992a] Inman Harvey. The SAGA cross: the mechanics of crossover for variable-length genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 269–278. North-Holland, 1992.

[Harvey 1992b] Inman Harvey. Species adaptation genetic algorithms: The basis for a continuing SAGA. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 346–354. MIT Press/Bradford Books, Cambridge, MA, 1992.

[Harvey 1992c] Inman Harvey. Untimed and misrepresented: Connectionism and the computer metaphor. Technical Report CSRP 245, School of Cognitive and Computing Sciences, Univ. of Sussex, 1992.

[Harvey 1993a] I. Harvey. Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. In C. Langton, editor, *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVI*, pages 299–326. Addison Wesley, 1993.

[Harvey 1993b] Inman Harvey. The puzzle of the persistent question marks: A case study of genetic drift. In S. Forrest, editor, *Genetic Algorithms: Proceedings of the Fifth Intl. Conference*, pages 15–22, San Mateo CA, 1993. Morgan Kaufmann.

[Hesser and Männer 1991] J. Hesser and R. Männer. Towards an optimal mutation probability for genetic algorithms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*. Springer-Verlag, Lecture Notes in Computer Science Vol. 496, 1991.

[Hillis 1991] W.D. Hillis. Co-evolving parasites improve simulated evolution as an optimization parameter. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life II: Proceedings Volume of Santa Fe Conference Feb. 1990.* Addison Wesley: volume XI in the series of the Santa Fe Institute Studies in the Sciences of Complexity, 1991.

[Hinton and Nowlan 1987] Geoffrey E. Hinton and Steven J. Nowlan. How learning can guide evolution. *Complex Systems*, 1:495–502, 1987.

[Hinton *et al.* 1986] G.E. Hinton, J.L. McClelland, and D.E. Rumelhart. Distributed representations. In D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, ed-

itors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 77–109. MIT Press, 1986.

[Hirschberg 1975] D.S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communications of the A.C.M.*, 18(6):341–343, 1975.

[Holland and Reitman 1978] J. Holland and J. Reitman. Cognitive systems based on adaptive algorithms. In P. Waterman and F. Hayes-Roth, editors, *Pattern directed inference systems*. Academic Press, 1978.

[Holland 1975] John Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, USA, 1975.

[Honavar and Uhr 1989] V. Honavar and L. Uhr. A network of neuron-like units that learns to perceive by generation as well as reweighting of its links. In *Proceedings of the 1988 Connectionist Models Summer School*, pages 472–484, San Mateo CA, 1989. Morgan Kaufmann.

[Hopfield 1982] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[Horswill and Brooks 1988] I. D. Horswill and R. A. Brooks. Situated vision in a dynamic world: Chasing objects. In *Proceedings AAAI—88*, pages 796–800, 1988.

[Husbands and Harvey 1992] P. Husbands and I. Harvey. Evolution versus design: Controlling autonomous robots. In *Integrating Perception, Planning and Action, Proceedings of 3rd Annual Conference on Artificial Intelligence, Simulation and Planning*, pages 139–146. IEEE Press, 1992.

[Husbands and Mill 1991] Philip Husbands and Frank Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, pages 264–270, San Mateo CA, 1991. Morgan Kaufmann.

[Husbands *et al.* 1993a] P. Husbands, I. Harvey, and D. Cliff. Evolving control systems for autonomous agents. *Journal of Information Science and Technology*, 1993.

[Husbands *et al.* 1993b] P. Husbands, I. Harvey, and D. T. Cliff. Analysing recurrent dynamical networks evolved for robot control. In *Proceedings of ANN93, the Third International Conference on Artificial Neural Networks*, pages 158–162. IEE Press, 1993.

[Husbands *et al.* 1993c] P. Husbands, I. Harvey, and D. T. Cliff. An evolutionary approach to situated AI. In A. Sloman, editor, *Prospects for Artificial Intelligence: Proceedings of AISB93*, pages 61–70. IOS Press, 1993.

[Husbands *et al.* 1995] P. Husbands, I. Harvey, and D. Cliff. Circle in the round: State space attractors for evolved sighted robots. *Journal of Robotics and Autonomous Systems. Special Issue on "The Biology and Technology of Intelligent Autonomous Agents".*, 1995.

[Jacob 1989] François Jacob. *The Possible and the Actual.* Penguin, 1989.

[Jakobi *et al.* In Press 1995] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proceedings of Third European Conference on Artificial Life.* Springer-Verlag, In Press, 1995.

[Jakobi 1994] N. Jakobi. Evolving sensorimotor control architectures in simulation for a real robot. M. sc. thesis, COGS, University of Sussex, 1994.

[Kasper and Schuster 1987] F. Kasper and H. Schuster. Easily calculable measure for the complexity of spatio-temporal patterns. *Physical Review A*, 36(2):842–848, 1987.

[Kauffman and Levin 1987] Stuart Kauffman and Simon Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128:11–45, 1987.

[Kauffman 1974] S. A. Kauffman. The large scale structure and dynamics of gene control circuits: an ensemble approach. *Journal of Theoretical Biology*, 44:167, 1974.

[Kauffman 1989] Stuart Kauffman. Adaptation on rugged fitness landscapes. In Daniel L. Stein, editor, *Lectures in the Sciences of Complexity*, pages 527–618. Addison Wesley: Santa Fe Institute Studies in the Sciences of Complexity, 1989.

[Kauffman 1993] S. A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[Kerszberg and Bergman 1988] Michel Kerszberg and Aviv Bergman. The evolution of data processing abilities in competing automata. In Rodney M. J. Cotterill, editor, *Computer Simulation in Brain Science*, pages 249–259. Cambridge University Press, 1988.

[Kitano 1990] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476, 1990.

[Koza 1990] John R. Koza. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Department of Computer Science, Stanford University, 1990.

[Koza 1992a] J. R. Koza. Evolution of subsumption using genetic programming. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 110–119. MIT Press/Bradford Books, Cambridge, MA, 1992.

[Koza 1992b] J. R. Koza. *Genetic Programming*. MIT Press/Bradford Books, Cambridge MA, 1992.

[Langton *et al.* 1991] C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors. *Artificial Life II: Proceedings Volume of Santa Fe Conference Feb. 1990*. Addison Wesley: volume XI in the series of the Santa Fe Institute Studies in the Sciences of Complexity, 1991.

[Langton 1989] C. G. Langton, editor. *Artificial Life: Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems, Los Alamos, NM, Sept, 1987*. Addison-Wesley: volume VI in the series of the Santa Fe Institute Studies in the Sciences of Complexity, 1989.

[Langton 1993] C. Langton, editor. *Artificial Life III*. Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVI, Addison Wesley., 1993.

[Lewontin 1983] R. C. Lewontin. The organism as the subject and object of evolution. *Scientia*, 118:63–82, 1983.

[Lindenmayer 1971] A. Lindenmayer. Mathematical models for cellular interactions in development. *Journal of Theoretical Biology*, 30:455–484, 1971.

[Lindgren 1990] K. Lindgren. Evolution in a population of mutating strategies. Technical report, Nordita, Copenhagen, 1990.

[Lindgren 1991] K. Lindgren. Evolutionary phenomena in simple dynamics. In C. G. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor, editors, *Artificial Life II: Proceedings Volume of Santa Fe Conference Feb. 1990.* Addison Wesley: volume XI in the series of the Santa Fe Institute Studies in the Sciences of Complexity, 1991.

[Lopez and Caulfield 1991] L. R. Lopez and H. J. Caulfield. A principle of minimum complexity in evolution. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature.* Springer-Verlag: Lecture Notes in Computer Science, 1991.

[Malsburg and Bienenstock 1986] C. von der Malsburg and E. Bienenstock. Statistical coding and short-term synaptic plasticity: A scheme for knowledge representation in the brain. In E. Bienenstock, F. Fogelman Soulie, and G. Weisbuch, editors, *Disordered Systems and Biological Organization.* Springer-Verlag, 1986.

[Männer and Manderick 1992] R. Männer and B. Manderick, editors. *Parallel Problem Solving from Nature, 2*, Amsterdam, 1992. North-Holland.

[Marr 1977] D.C. Marr. Artificial intelligence: A personal view. *Artificial Intelligence*, 9:37–48, 1977.

[Maynard Smith 1970] John Maynard Smith. Natural selection and the concept of a protein space. *Nature*, 225:563–564, 1970.

[Maynard Smith 1978] John Maynard Smith. *The Evolution of Sex.* Cambridge University Press, 1978.

[McClelland and Rumelhart 1986] J. L. McClelland and D. E. Rumelhart, editors. *Explorations in Parallel Distributed Processing.* MIT Press/Bradford Books, Cambridge Massachusetts, 1986.

[Meeden *et al.* 1993] L. Meeden, G. McGraw, and D. Blank. Emergence of control and planning in an autonomous vehicle. In *Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society*, pages 735–740, Hillsdale, NJ, 1993. Lawrence Erlbaum Associates.

[Meyer and Guillot 1991] Jean-Arcady Meyer and Agnès Guillot. Simulation of adaptive behavior in animats: Review and prospect. In J.-A. Meyer and S.W. Wilson, editors,

From Animals to Animats: Proceedings of The First International Conference on Simulation of Adaptive Behavior, pages 2–14. MIT Press/Bradford Books, Cambridge, MA, 1991.

[Meyer and Wilson 1991] J.-A. Meyer and S.W. Wilson, editors. *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. MIT Press/Bradford Books, Cambridge, MA, 1991.

[Meyer *et al.* 1993] J.-A. Meyer, H. Roitblat, and S. Wilson, editors. *Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92)*, Cambridge, MA, 1993. MIT Press/Bradford Books.

[Miller and Forrest 1989] John H. Miller and Stephanie Forrest. The dynamical behavior of classifier systems. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo CA, 1989. Morgan Kaufmann.

[Miller and Todd 1993] G. F. Miller and P. M. Todd. Evolutionary wanderlust: Sexual selection with directional mate preferences. In J.-A. Meyer, H. Roitblat, and S. Wilson, editors, *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92)*, pages 21–30. MIT Press/Bradford Books, Cambridge MA, 1993.

[Miller *et al.* 1989] Geoffrey F. Miller, P. M. Todd, and S. U. Hegde. Designing neural networks using genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 379–384, San Mateo CA, 1989. Morgan Kaufmann.

[Minsky 1967] Marvin Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[Mjolsness *et al.* 1987] E. Mjolsness, D. H. Sharp, and B. K. Alpert. Recursively generated neural networks. *IEEE Intl. Conf. on Neural Networks*, III:165–171, 1987.

[Montana and Davis 1989] David J. Montana and Lawrence Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pages 762–767, 1989.

[Moravec 1983] H.P. Moravec. The Stanford Cart and the CMU Rover. In *Proc. of IEEE*, volume 71, pages 872–884, 1983.

[Muhlenbein and Kindermann 1989] H. Muhlenbein and J. Kindermann. The dynamics of evolution and learning - towards genetic neural networks. In R. Pfeifer, Z. Schreter, F. Fogelman-Soulie, and L. Steels, editors, *Connectionism in Perspective*, pages 173–197. Elsevier Science Publishers B.V. (North-Holland), 1989.

[Needleman and Wunsch 1970] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[Nowak and Schuster 1989] Martin Nowak and Peter Schuster. Error thresholds of replication in finite populations, mutation frequencies and the onset of Muller's ratchet. *Journal of Theoretical Biology*, 137:375–395, 1989.

[Pomerleau 1991] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3:88–97, 1991.

[PRANCE 1991] PRANCE. Perceptive robots: Autonomous navigation and cooperation through evolution. Unpublished research proposal, PRANCE consortium: Univ. of Sussex, Cap Gemini Innovation, École Normale Supérieure (Paris) and Université Libre de Bruxelles, 1991.

[Ramsey *et al.* 1991] W. Ramsey, S.P. Stich, and D.E. Rumelhart, editors. *Philosophy and Connectionist Theory*. Erlbaum, 1991.

[Ray 1992] Thomas S. Ray. An approach to the synthesis of life. In J.D. Farmer, C.G. Langton, S. Rasmussen, and C. Taylor, editors, *Artificial Life II*, pages 371–408. Addison-Wesley, 1992.

[Reynolds 1993] C. Reynolds. An evolved, vision-based model of obstacle avoidance behavior. In C. Langton, editor, *Artificial Life III, Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. XVI*. Addison Wesley., 1993.

[Rosenschein 1985] S.J. Rosenschein. Formal theories of knowledge in AI and robotics. *New Generation Computing*, 3:345–357, 1985.

[Roughgarden 1979] Jonathan Roughgarden. *Theory of Population Genetics and Evolutionary Ecology: an Introduction*. Macmillan, New York, 1979.

[Rudnick 1990] M. Rudnick. A bibliography of the intersection of genetic search and artificial neural networks. Technical Report CS/E 90-001, Oregon Graduate Institute, Dept. of Computer Science and Engineering, 1990.

[Sankoff 1972] David Sankoff. Matching sequences under deletion/insertion constraints. *Proceedings of the National Academy of Science, USA*, 69(1):4–6, 1972.

[Schaffer *et al.* 1989] J.D. Schaffer, R.A. Caruana, L.J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. D. Schaffer, editor, *Proceedings of the 3rd ICGA*, San Mateo CA, 1989. Morgan Kaufmann.

[Schaffer 1989] J. D. Schaffer, editor. *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo CA, 1989. Morgan Kaufmann.

[Schuster 1992] P. Schuster. How do RNA molecules and viruses explore their worlds? In *Proceedings of Integrative Themes in Complexity Meeting, Santa Fe*, 1992.

[Schwefel and Männer 1990] H.-P. Schwefel and R. Männer, editors. *Parallel Problem Solving from Nature*. Springer-Verlag, 1990.

[Smith and Goldberg 1992] R.E. Smith and D. E. Goldberg. Diploidy and dominance in artificial genetic search. *Complex Systems*, 6(3):251–285, 1992.

[Smith 1980] Stephen F. Smith. *A Learning System based on Genetic Adaptive Algorithms.* PhD thesis, Department of Computer Science, University of Pittsburgh, USA, 1980.

[Starkweather *et al.* 1991] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley. A comparison of genetic sequencing operators. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76, San Mateo CA, 1991. Morgan Kaufmann.

[Stenseth and Smith 1984] N. C. Stenseth and J. Maynard Smith. Coevolution in ecosystems: Red Queen evolution or stasis? *Evolution*, 38:870–880, 1984.

[Thomas 1991] R. Thomas. Regulatory networks seen as asynchronous automata: A logical description. *Journal of Theoretical Biology*, 153:1–23, 1991.

[Thompson In Press 1995] A. Thompson. Evolving electronic robot controllers that exploit hardware resources. In *Proceedings of Third European Conference on Artificial Life*. Springer-Verlag, In Press, 1995.

[Todd and Miller 1991] P. M. Todd and G. F. Miller. On the sympatric origin of species: Mercurial mating in the quicksilver model. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth Intl. Conf. on Genetic Algorithms, ICGA-91*, pages 547–554, San Mateo CA, 1991. Morgan Kaufmann.

[Turing 1950] A. M. Turing. Computing machinery and intelligence. *Mind*, LIX(2236):433–460, October 1950.

[Valen 1973] L. Van Valen. A new evolutionary law. *Evolutionary Theory*, 1:1–30, 1973.

[van Gelder 1992] Tim van Gelder. What might cognition be if not computation. Technical Report 75, Indiana University Cognitive Sciences, 1992.

[Varela and Bourgine 1992] F. J. Varela and P. Bourgine, editors. *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life.* MIT Press/Bradford Books, Cambridge, MA, 1992.

[Varela *et al.* 1991] F. Varela, E. Thompson, and E. Rosch. *The Embodied Mind.* MIT Press, 1991.

[Viola 1988] P. Viola. Mobile robot evolution. Bachelors thesis, M.I.T., 1988.

[Wagner and Fischer 1974] R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the A.C.M.*, 21(1):168–173, 1974.

[Whitley *et al.* 1989] D. Whitley, T. Starkweather, and D'A. Fuquay. Scheduling problems and traveling salesmen: The genetic edge recombination operator. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 133–140, San Mateo CA, 1989. Morgan Kaufmann.

[Whitley 1989a] D. Whitley. Applying genetic algorithms to neural network learning. In *Proceedings of the 7th Conference for the Study of Artificial Intelligence and Simulated Behaviour, Sussex, England*, London, 1989. Pitman Publishing.

[Whitley 1989b] D. Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, San Mateo CA, 1989. Morgan Kaufmann.

[Wiener 1961] N. Wiener. *Cybernetics, or Control and Communication in the Animal and the Machine*. MIT Press, 1961.

[Wilson 1985] S.W. Wilson. Knowledge growth in an artificial animal. In J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their applications*, Hillsdale NJ, 1985. Lawrence Erlbaum Associates.

[Wilson 1987] S. Wilson. The genetic algorithm and biological development. In J. J. Grefenstette, editor, *Genetic Algorithms snd their Applications: Proceedings of the Second Intl. Conf. on Genetic Algorithms*, pages 247–251, Hillsdale NJ, 1987. Lawrence Erlbaum Associates.