Evolving Neural Network Controllers for Task Defined Robots

Kyran Dale
kyrand@cogs.susx.ac.uk

Degree: MSc in Knowledge-Based Systems 1993/94
Supervisor: Inman Harvey
Submitted: 1 September 1994

Abstract

Some recent attention in Artificial Intelligence (AI) research (specifically the sub-discipline known as Artificial Life) has been focussed on the possibility of using genetic algorithms to evolve neural network controllers for task-defined robots. Employing techniques formalised by Holland (1975), the hope is that by using various encoding methods for representing a neural network on a 'genome' -commonly a binary string- and then manipulating a population of these genomes using, primarily, cross-over and mutation operators according to fitness-preferential dictates, one may efficiently search a large parametric state-space for useful networks.

This paper deals with my attempt to evolve a neural network that, by mediating between a simulated robot's actions and its environmental input leads to a 'guard-dog' behaviour.

KEYWORDS: Genetic Algorithms, Neural Network, Task-defined Behaviour, Simulated Environment, Encoding Method.

Acknowledgements
Thanks to my supervisor Inman Harvey for some very useful comments.

Contents

1 Introduction

1.1 A Short Historical Perspective

1.1.1 Genetic Algorithms

The Darwinian Theory of Natural Selection is one of the profound insights in the history of scientific endeavour. At its heart is the idea that purely randomate, stochastic changes coupled with environmental pressures can lead to the development of creatures which are 'tailored for their environment'. Different options are thrown up randomly and those which do well - are fitter than their rivals - prosper. Important to note is that this is not an argument for design; the environment can be seen as a test for which nature throws up random attempts at solution; those attempts that score well get to multiply. In modern biological parlance, their genes are passed down to the next generation.

Both in terms of its notoriety and seeming success, natural selection has made a big impact. It is not, therefore, surprising that many in the fields of Artificial intelligence and Mathematics began to wonder whether the power of genetic algorithms (GAs) could be harnessed for use as a state-space [1] search algorithm. A breakthrough came in 1975 with Holland's ground-breaking paper [1] which showed that a genetic algorithm employing crossover and mutation operations (see below) on a fitness preferential breeding basis (the fittest or most successful attempts -represented by an n-parameter genome - having a proportionately higher chance of passing some part of their genome into the next generation) represented a very efficient means of searching through a large number of possibilities and tended to maximise the number of short, useful blocks of genome (referred to as 'schemas' by Holland) averaged through the population -the algorithm would inevitably optimise for a function that could be represented in terms of these schemas; it would produce progressively better attempts at solving the problem.

2 The Task at Hand

The aim of this project was to produce a guard-dog like behaviour in a simulated robot by using a genetic algorithm to search through a state space of possible neural network controllers. Operating in a four-walled, square arena the task of the robot is, in a limited time (typically four hundred time units) to keep a simulated automaton as far as possible from the centre of the arena. The rewards, in terms of fitness points, to the guard-robot increase progressively the further from the centre the automaton can be kept; being allocated according to a Gaussian function (see section below on the fitness function).

The biological analogy with a guard-dog should be taken loosely. It refers only to the behaviour exhibited; there is no claim being made that the means by which the creature abstracts from its environment or the strategies it employs in defending the centre resemble, in any way, those of a real guard-dog; although the possibility of strategic similarity is not discounted.

2.1 The Difficulty of Hand Wiring Robots

The task was chosen because it seemed suitably complex so as to present a challenge to anyone trying to hand design or hard-wire a robot to perform it; imagine trying to dictate a suitable left and right motor response for the robot based only on the ten intensity levels

[1] n-dimensional where n is the number of possible parameters being explored.

(from ten light sensors spaced evenly around the robot (see below)) presented at each time phase.

Indeed, much of the impetus for research in this area comes from the firmly held belief that, as the tasks required of robots become increasingly more complex, it will be increasingly infeasible to hand-design robots to perform them [2] . In as much as, coupled with suitably chosen selective pressures on breeding (represented as a fitness function, often over time), a GA allows the problem to seek its own solution (the solution is, in this sense, emergent from the system created) the human programmer is freed from the task of designing the solution and can turn his attention to creating the right environmental pressures (finding the right fitness function) and the right encoding system (the way in which, for example in this case, the neural network is represented on the genome (see section below on encoding the network)) in which progressively better solutions will be found. This is, in itself, a difficult task, but is orders of magnitude simpler than having to hard-wire the robots.

Part I
Genetic Algorithms

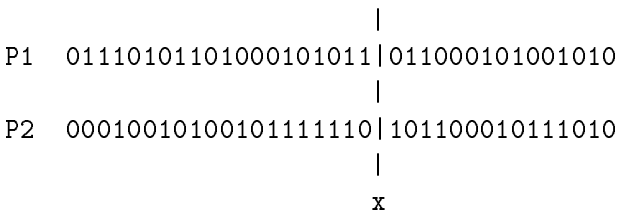
3 Standard Methods Employed by Genetic Algorithms

3.1 Common Genetic Operators

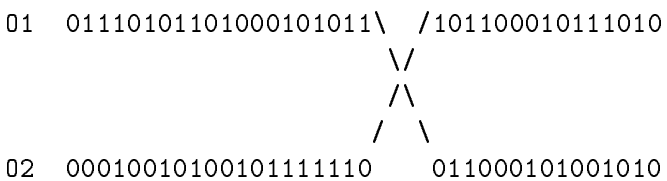
3.1.1 Crossover

The most common form employed, crossover consists of combining elements from one genome with those from another to produce an offspring. The most common method is single-point crossover though multiple-point crossovers are sometimes employed. The method is as follows:

1. Take two parent genomes P1 and P2 of fixed length l and divide them at point x along the genome where x is a random number between 1 and l-1



2. Crossover the two genomes at this point to produce two offspring O1 and O2.



[2] see Harvey, Husbands et al and their provisional manifesto outlining the difficulties faced [2].

3. Choose one of O1 and O2 to be the offspring of P1 and P2; in the GA I employ (see below) this offspring replaces a stochastically chosen weak member of a sub-population.

Single point crossover was used in this GA. The genome of the two chosen parents [3] was cut at the bit level

3.1.2 Mutation

In Darwinian natural selection it is mutation that plays the major role in providing the impetus for change; it is the vehicle for random search of the state space. However in GAs up till recently it has been a poor cousin to the crossover operation. This is because GAs have been commonly perceived as function optimisers whose job stops once the population has converged -this being the case when the members are approximately identical. For real-world evolution this is the starting point; a species has, by definition, achieved a high degree of convergence. Recently, as the limitations of GAs as fixed length function optimisers has become apparent (see [3,4]) for an analysis), more interest has begun to focus on the mutation operator (see [10]) and its subtleties. For a single, bit-wise mutation:

1. Take one of the offspring of parents P1 and P2 of length l. Choose a number x between 1 and l - 1 and invert the bit found there on the offspring's genome

```
          01  01110101101000101011011000101110101
                                    |
                                    x
                                    |
Mutated   01  01110101101000101011111000101110101
```

Rate of mutation has typically been of the low order of one bit per thousand in the genome, though recent research has advocated a far higher rate (see [3,4]).

In the GA employed in this project, mutation was initially set to a rate of one bit per genome.

3.2 Rank Based Selection

Rank Based Selection(RBS) is a means of sustaining useful evolutionary pressures in the face of either very large or very small discrepancies between the fitness of individual members. An example will show why the mediation of a rank-based selection system is necessary:

Consider the example of a population of genomes -artificially small but adequate for the purpose of explanation- g1,g2,g3 and g4 with fitnesses g1=100, g2=20, g3=5, g4=4. A method of choosing which of these members will get to breed might involve totalling their fitnesses and then employing a 'roulette wheel' method to choose which member is to breed:

[3] the genome is an array of n signed characters in 'C'. Dependent on the encoding method employed these signed characters were sometimes cast as unsigned integers.

METHOD 1:

1. take x, a random number between 0 and the fitness total of the group (in this case 129).

2. add the fitnesses of the population members sequentially until this total exceeds x.

3. the member chosen to breed is the last member whose fitness was added to the accumulator.

The probability for each member reproducing is *members fitness / total fitness* and, based on the sample population given, this would give g1 an overwhelming breeding advantage. Based on this stochastic method, g1's genes would very quickly predominate. But an important principle in the theory of GAs maintains the importance of diversity; it is not desirable that one solution within the search space is pin-pointed quickly since it is likely to be sub-optimal [4]. With this in mind and taking into account the fact that at the beginning of a GA there may well be large differences in the fitnesses of the population -such as to lead to one -or a small number in a distributed GA- member predominating , one can already call into question the efficacy of *method 1.*

But another reason exists that mitigates against *method 1.* Most GAs converge at some point or another; the fitnesses in the population are, at this point, similar. *Method 1*, applied to a converged or semi-converged population seems to lead to a rather aimless search of the state-space. If we had a population, for example,with fitnesses of 1001,1020,1100,965 then each would seem to have a roughly equal chance of breeding, by *method 1.* The stochastic search becomes unfocussed and there is less chance of serendipitous discovery.

To counteract the ineffectiveness of *method 1* at these points in a GA's operation RBS is employed. Here the fitnesses of members of the population are used to rank them; for example the least fit is given a rank of N (where N is the size of the population) and the most fit given a rank of 1. From now on it is these ranks that are used to select breeding members, not the fitnesses; these are discarded after ranking has taken place and play no further part in the selection procedure.

RBS means that we can expect relatively similar selection pressures at different stages in the operation of the GA; typically the fittest member having approximately twice the chance of selection as the median fittest. Disparities in the individual fitnesses are only of issue in placing the genomes in rank; they cannot produce the detrimental effects that have been mentioned. The selection procedure has been homogenised.

In the GA employed for this project, RBS was used at all times in preference over *method 1.*

4 The Choice of a Genetic Algorithm

The study of GAs is a discipline in its infancy. Holland's schema processing analysis (see [1] and [5]) has provided a mathematical foundation for GAs which shows them to be an extremely efficient search algorithm; in a population of size P the number of schemata - concurrent genes or 'building blocks' being processed is of the order P cubed. The operation of cross-over and mutation have been shown, in a population of fixed-length genomes with fitness-preferential reproduction, to encourage the formation of small blocks of genes -as averaged from the genome population- which are advantageous to function

[4] the larger the search space the higher this probability.

optimisation. This proof was important because it showed, for these blocks, a remarkably large amount of search space could be combed in relatively few genetic operations on the genome population.

However, many areas of GA research defy mathematical formalism. There is much that is unpredictable and often heuristic, rules-of-thumb are all that is available to guide the development of new GAs.

The upshot of this is that much of GA research involves trial and error, a natural selectionist approach true to its subject matter. This trial and error is not, of course, totally uninformed; at the heart of most approaches is a putative gem of wisdom:

One approach currently gaining credibility is motivated by the belief that in restricting evolutionary forces to small pools or sub-populations within the main population of genomes, one may encourage diverse approaches to specific problems to flourish within the main population; this approach has been developed in response to a problem that consistently dogs the operation of GAs:

It is entirely possible that a genome which is initially relatively unfit may possess the key, or springboard, to a very fit creature -a good position in the n-dimensional search space- of the future. Equally, a genome which is initially fit may well prove redundant in the future, being further away in the search space from the optimum than the previous ailing genome. If both these creatures are encountered early on in the evolution of the main population and both are subjected to the full battery of evolutionary pressures -if both are in direct competition with one another- then it is probable that the latter will triumph at the expense of the former and a potentially useful future approach will be lost. If, however, we allow the less fit genome to breed away from the full rigours of the main population in a small pool then it may lead to the development of a better solution than the previously fitter genome. In short, it is recognised that encouraging diverse approaches to a problem is advantageous but that in GAs a fitter solution may well come to predominate too quickly, thus preventing any of the weaker but potentially useful approaches from reaping fruit. What is needed is an approach which compromises between a desire to see the fittest triumph and a belief that initially weak members of the main population should be protected from the full application of the Darwinian ethic; a time buying strategy to allow the different approaches to show their worth.

One way in which this can be achieved is by employing a spatialised GA. The algorithm is as follows:

1. Arrange initial n*n members on a toroidal (wrap around) array of dimension nxn.
2. To breed two creatures:
   (a) Choose a member of the main population (*parent1*) on a group-fitness preferential basis - where group-fitness is the average fitness of a member of the main population and his immediate neighbours- from a sample of n chosen at random; use rank-based-selection(RBS).
   (b) Restrict breeding to within the sub-population defined by *parent1* and its immediate neighbours.
   (c) Using RBS select a member of this subpopulation (*parent2*) to mate with *parent1*.
   (d) Crossbreed *parent1* and *parent2* and choose one of the crossed genomes to be the the *offspring*. Mutate this offspring.

(e) Choose a member of the sub-population using RBS biased in favour of the weaker members [5]. Replace this member with the *offspring*.

Breeding, within this sub-population is governed by rank-based-selection (RBS) (see section above) which is used to choose the breeding partners and the member of this sub-population to be replaced by their off-spring.

It should be recognised that these sub-populations merge together and that via this route a very fit solution may eventually colonise the whole population. However, the time it will take to achieve this has increased and in the meantime any one of these semi-isolated colonies may produce a solution to combat invasion.

A spatialised GA appears to me to be a good compromise and I chose to employ one in my project.

5 Encoding the Neural Network on the Genome

The genome is, at its most basic, a long binary string. Moving up a level of interpretation, this string can be viewed as encoding numbers (signed or unsigned) or characters (letters of the alphabet encoded in ASCII). Somewhere on this string is the information needed to build the neural network controller for the robot. There is much debate, within Alife, as to which encoding should be employed. A number of options present themselves:

5.1 Direct Encoding

We can pre-establish the structure of the network (number of input, hidden, and output nodes and connections to be made between them) and then use the genome to represent network parameters (the strength of the weight connections -whether excitatory or inhibitory, and the value of the thresholds on the weights -their bias). This method is comparatively simple to employ but, by pre-establishing the structure of the net we have already imposed what is probably a sub-optimal form of network for the task; the structure of the network forms no part of the search space. In this form parts of the genome encode all the weight and threshold parameters for a single unit [6]. In this paper I refer to this method as *strong direct-encoding*[7] (see Fig 1). Fig 5 shows how the genome, an array of signed characters (in the range [-127,127] $\subset \mathbf{R}$) encodes network parameters.

Another more profound problem exists which calls into question the whole ethos of *strong direct-encoding*. As has been mentioned (see section above) the spatialised GA employed in this project encourages diverse approaches to the problem posed - in this case evolving a guard-dog behaviour in robots. The distributive nature of connectionist (or neural) networks means that an individual robot's solution to the problem posed is represented as a generalised hypothesis spread through the network parameters (weight, thresholds etc.). It can be seen as a vector in the n-dimensional space formed by these parameters. The efficacy of an individual weight connection or threshold may well depend on the context in which it finds itself (the generalised hypothesis or vector). Since the

[5] The weakest is now ranked 1, the fittest n, where n is the size of the subpopulation.

[6] Holland's schema processing advocates that the details for a single unit should be placed close together on the genome (see [1],[5], and especially [6])

[7] commonly the term direct-encoding is applied to any situation where there is a one to one mapping between the genotype and the phenotype it encodes (in this case a neural network). The encoding method is termed 'strong' because the structure of the network is pre-established, as opposed to its being defined on the genome.

crossover operation radically disrupts this context we might expect many of the parameters, post-crossover, to be distinctly sub-optimal; it is possible that two very fit robots, when bred, might lead to a very weak offspring.

This should not, however, provoke despair in the method of *strong direct-encoding*. Many weight connections and thresholds will probably be of some use in most networks. The weights and thresholds run in the range from -127 to +127. Negative values can be viewed as inhibitory (the node receiving input along a negatively weighted connection being less likely to register 1 as its activation value) while positive are excitatory (the node being more likely to register 1 as its activation value). Since the relative excitatory or inhibitory capacity of a connection or threshold is preserved through the genetic operation of crossover, and taking into account the fact that we might expect certain relative weights to be generally useful for producing a fit network, it is only necessary for the genetic algorithm to encourage these generally useful parameters for it to be successful; Holland's schema processing has shown that GAs do perform this function (see [1]) [8].

5.2 The Hybrid Encoding

While *strong direct-encoding* can be shown to work (see Results section below) its efficiency must be called into question. As has been mentioned, by pre-establishing the structure of the network we are already imposing what is almost certainly a sub-optimal state-space in which the GA is required to work; better structures for the task exist. Recent interest in evolving connectionist architectures [9] has focussed on using the GA to design structural features of the network, either with fixed weight and threshold parameters (typically binary values) or in conjunction with defining these parameters also [10].

As well as employing *strong direct-encoding* (see Fig 5) with my GA, I also designed and tested what I refer to as a *hybrid encoding* system [11]; the number of input, hidden and output nodes (see section on the neural network) were fixed and a portion of the genome (invariable) was allocated to each. Within this portion the units are free to set their own threshold parameter and choose which other units they receive input from and the strength of the connections from these units. Depending on its status, hidden or output, the unit is allowed a set number of connections. As opposed to the *strong direct encoding* method, this *hybrid* method allows a fair amount of flexibility for the structure of the network ; a unit can, for example, make a multiple number of recurrent connections to itself or other units; it can also ignore certain units completely.

The difference between the two encoding systems is shown in Fig 1. In the *strong direct-encoding* system, any number on the genome represents, dependent on its position, the value of a pre-established weight connection or threshold; the weight-link is already fixed, requiring only a weight. In the *hybrid encoding* system the weight-links themselves are variable. After establishing the threshold of a unit (as in the *strong direct encoding*) two numbers are required to establish each link and its weight. For the first the signed character (in the range [-127,127] $\subset \mathbf{R}$) is recast as an unsigned integer (in the range [0,255]$\subset \mathbf{R}$) and its value modulus 20 (the number of units in the network) establishes the

[8] many examples exist of successfully employed *strong direct-encoding*. See [7] for an example.

[9] see [2],[6], and [8] for some recent examples of incorporating structural details of the connectionist network on the genome.

[10] see [8] for an example of such inclusive coding.

[11] the name *'hybrid'* used is solely for the purpose of distinguishing between the two encoding methods employed but it is hoped to encompass the fact that, though pre-established segments of the genome are still allotted to individual units, within these segments freedom exists for the units to define structural features of the neural network.

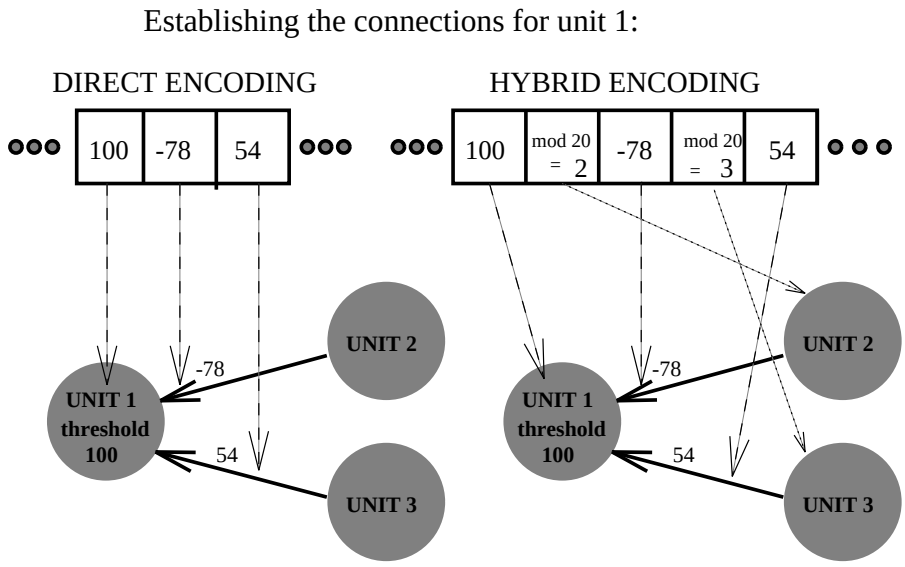


Figure 1: The *hybrid encoding* method establishes structural details of the network which can be transferred during the crossover operation. Under the *strong direct-encoding* these details are pre-established.

unit that the weight is to come from [12]. The next number specifies the weight of the link in the normal way.

Each hidden and output unit was allowed to establish ten incoming weight-links. This meant that the *hybrid encoded* network used 100 links, as opposed to the 130 in the *strong direct-encoded* network. The number of weight-links allowed provided for a fair comparison between the *strong direct-encoding* and the *hybrid encoding*, [13] the extra links to the output units , compared with the *strong direct-encoding*, reflecting their primary role in establishing the motor responses of the robot (see Fig 4).

The relative success of this system (see Results below) shows the efficacy of allowing at least some structural elements of the network to be decided by the GA [14].

[12] Allowing units only to receive input from other units and not to send out links simplifies the processing of the genome into the neural network. This speeds things up considerably and also limits the search space since specifying a link as 'to' or 'from' would require extra indicators on the genome.

[13] the increased search space of the *hybrid* method means that relatively fewer potential phenotypic options can be processed in the number of breeding cycles available, when compared with the *strong direct-encoded* network.

[14] An even more radical approach is adopted by Moriarty, D., and Miikkulainen, R., [8] in their attempt to breed an Othello playing program. This involves using a marker-based encoding schema whereby the individual network units are not allocated fixed portions of the genome but are allowed to mark out their own (for example any number modulated by 50 returning a 1 is assumed to mark the beginning of a portion of genome fixing the structural parameters for a unit. Any number modulated by 50 returning 0 marks the end of these parameters).

5.3 Search Space Consideration

The total size of the *strong direct-encoded* genome is 130 signed integers [15]. Parametrically this makes for a 130 dimensional state space.

In the *hybrid encoding* each output and hidden unit was allowed to define ten weighted connections. Although the total number of weighted connections was less (100 compared with 120) than the *strong direct-encoding*, because each connection required two numbers, (one to establish a link, the next to weight it) the search space, 210 dimensional, was larger than the *strong direct-encoding* method.

It was hoped that the general efficiency of genetic algorithms (see above), coupled with the freedom of the neural network to establish novel and better task-oriented network structures, would compensate for this increased search space and still allow the *hybrid system*, in the allotted 2000 breeding cycles, to outperform the *strong direct-encoded* method (see Results below for confirmation of this belief).

Part II
The Simulation

6 Practical Considerations

6.1 Availability and Speed of Processor

As a member of a networked system, processor time is shared out among a group and is as such fairly limited. The amount available affects considerations such as the complexity of the environment.

6.2 Designing the Environment

The environment in a simulated robotics project can be made arbitrarily complex. Processor intensive applications such as recursive ray-tracing (see section below) or complex kinematics can almost always be made more accurate. However, the availability of processor time meant that my environment had to be necessarily simplistic. The addition of generous quantities of noise should, however, alleviate to a certain extent the problem of unrealism (see section below on the importance of noise).

6.3 Choice of a Programming Language

Having been impressed by the speed of 'C' coded neural networks, I decided to teach myself this language and coded my programming project in it. It is not unusual for GAs to take days to achieve results -dependent, of course, on the problem posed- and the speed and efficiency of implementation is an extremely important factor.

7 The Environment

The environment in which the guard-robot finds itself is a two dimensional four-walled room (dimensions 400x400 units). The walls radiate light with an intensity of 1, whilst

[15] (input-hidden) 10x6 + (hidden-hidden) 6x6 + (hidden-output) 4x6 + (hidden and output thresholds) 10.

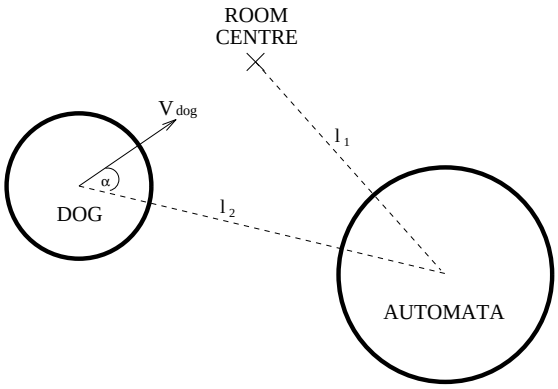


Figure 2: The Automaton's Movement

the automaton is black (intensity 0) (see section below on ray-tracing).

7.1 The Kinematics

The robot and automaton do not physically interact. This transparency saves processor time computing the collision dynamics and acknowledges the precedence given in the experiment to the central motion vectors of the two [16].

The environment, as mentioned, is non-toroidal and collisions occur between the robots and the walls. These collisions are elastic and no attempt is made to simulate complex, friction-dependent interactions - once again frugality rules.

The robot's movement is a product of disparities between the forces generated by its left and right wheels; unless the impulses are equal - in which case it moves a set distance in the direction it is facing- this motion is then resolved into a rotation and translation. This allows for quite a degree of freedom in the robot's movement.

7.2 The Movement of the Automaton

Two influences effect the movement of the automaton in the simulation. These are attraction to the centre and repulsion from the guard-robot. Of these two, it is the latter which takes precedence. If the automaton is within 100 distance units of the robot, it will either be immobilised or repelled by the robot. Referring to the diagram (Fig 2):

1. If the length of l_2 is greater than 100 then the automaton will be attracted towards the centre, along l_1 with a constant velocity of 60 units per time phase. This is double the maximum guard robots speed and is intended to severely punish any robot leaving the centre unguarded.

2. If the length of l_2 is less than 100 then the automaton's reaction is dependent on whether the robot is advancing towards it or not (whether $\cos(\alpha)$ is positive):

   (a) If $\cos(\alpha)$ is negative then the automaton is immobilised.

[16] for the early simulations, the repulsion of the automaton from the robot was inversely proportional to the distance between the two centres (however, the periphery of the automaton is taken into account in the ray-tracing procedure). By allowing the centres to come close together, greater repulsion could be generated.

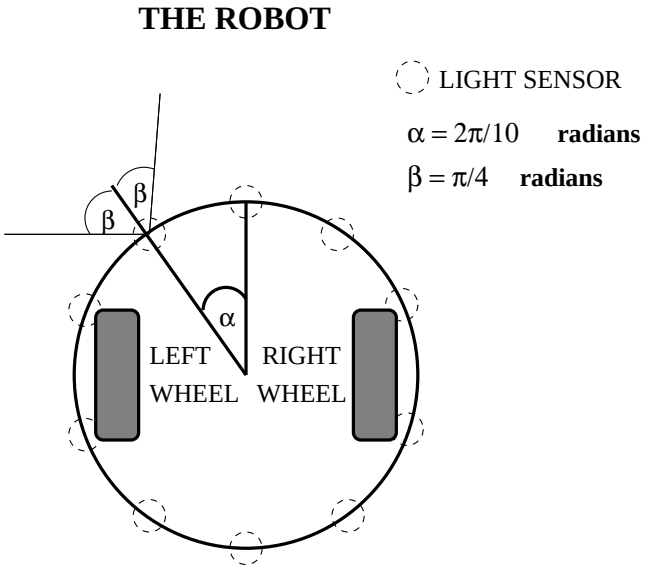


Figure 3: The Simulated Robot

(b) If $\cos(\alpha)$ is positive then the automaton is repelled along the line l_2 either by a force proportional to the robot's velocity times cos (α) divided by the length of l_2 or, in the case of the normalised automaton response (see below), at a speed of 60 units per time phase.

8 The Robot

The robot is circular (the environment being two dimensional) with ten light sensors arranged uniformly (every $(2\pi/10)$ radians) around its periphery. It's movement is effected by left and right wheels powered by separate motors (see Fig 3).

Information from the environment is obtained by the light sensors which return an average intensity rating over a defined arc-spread 2β (see Fig 3). Referring to Fig 4, normalised input from these (in the range [0,1] $\subset \mathbf{R}$) is sent directly to the ten input nodes of the robot's neural network [17] where the values are forward propagated through the network, according to weighted connections and node thresholds whose parameters are encoded on the robot's genome (see section on encoding the genome). This propagation establishes the activation values of four output units (in the range [0,1] $\subset \mathbf{R}$) O_1, O_2, O_3 and O_4. The left motor response is obtained by $O_1 - O_2$, the right by $O_3 - O_4$; this gives a possible range for each motor of between -1 and +1 (allowing forward and backward motion, and any discrepancy between the left and right motor outputs providing for a rotation motion) .

For the simulations run the left and right motor impulses at each time period were multiplied by the robot's maximum forwards and backwards velocity of 30 distance units per time period to give a distance over which each wheel travels; the robot's movement is resolved into a translation and rotation dependent upon the distance travelled by each

[17] analogous to its sensory nervous system, being employed to mediate between received sense data subsequent action.

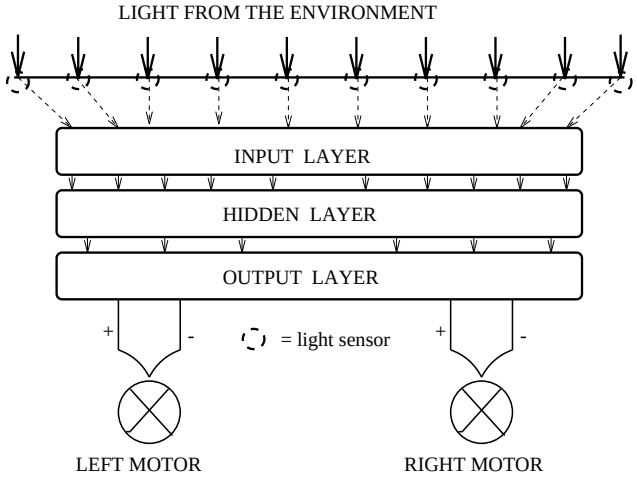


Figure 4: Converting Light Intensities to the Robot's Movement

wheel.

Uniform noise at $\pm 10\%$ is added at all stages of the network's propagation and the resulting motor response causes the robot to move position (effecting a rotation/translation resolution according to standard dynamics). This change in its position causes the robot's environmental input to change. This new information is sampled by the light sensors and the cycle begins again. In this way the system can be said to be closed [18].

9 Discriminating the Environment

The guard-robot discriminates its environment with the use of a neural network; its behaviour being moderated by this discrimination.

9.1 Why a Neural Network

For the guard-robot to function properly it must have knowledge about its environment. In this case knowledge is exhibited in the robot's capacity to make the right move given certain information about its environment, received as intensities registered by its light sensors. It must discriminate the information received and act according to that discrimination; a simplistic example might be "if the sensors in front register light, dark, light -indicating the presence of the automaton silhouetted against a wall - then cause the motors to advance the robot"; this action is likely to have the desirable consequence of forcing (via repulsion) the automaton further away from the centre of the room, thus improving the robot's fitness.

A neural network is a very efficient way of representing, through its weighted connections and thresholds, a generalised hypothesis about the environment in which the guard-robot finds itself. If one imagines the total number of possible inputs to the network (in the form of normalised activations from the ten light sensors spaced evenly around the robot) as representing an 'input space', each n-dimensional (in this case n = 10) input vector finding a place here, then the combination of hidden and output units allied to the

[18] see [11] for an advocation of closed-environment simulation systems.

weights and thresholds, can be said to partition this input space into discrete areas. Members of these areas provoke a similar output response which, applied to the motors, cause the robot to move in a set way; in this way, the robot can be said to possess a number of strategies. As the robot gets fitter, the partitioning of the input space into set output response comes to represent progressively better strategies for repelling the automaton from the centre; the guard-robot begins to develop a strategic understanding of the problem at hand. The distributive nature of the neural network; each unit participating in a large number of abstractions from the input space (for use in fulfilling useful discriminations of the environment) makes it an efficient way of storing the large amounts of knowledge the robot must employ if it is to develop a successful general approach [19].

The optimal solution to the task can be viewed as a vector in the parametric weight and threshold space. Successful solutions will tend towards this vector and the GA can be seen as the motive force for this tendency. In the case of the *hybrid encoding* scheme (see above) the state space is extended to include structural details about the net; the individual units are allowed to search for useful weighted links with other units, rather than having them pre-established.

9.2 The Neural Network

The neural network in the guard-robot is analogous to a central nervous system. It acts as an intermediary deciding, from a given sensory input -provided by the ten light sensors arranged around the robot- what action is to be taken by the robot -achieved through impulses sent to its left and right motor-driven wheels. The structure of the neural network is dependent on the encoding system used (see section on encoding the network).

Its most general form is a three layer network with ten input units (one for each light sensor on the robot), six hidden units and four output units (each motor using two outputs to establish its impulse). The *strong direct encoding* method (see above) has recurrent connections on its middle, or hidden connections (see Fig 5). The *hybrid method* allows the hidden and output units to establish recurrencies; any input to a unit, the weight connection being defined on the genome, coming from a unit on the same or a higher level of the network (the levels being input, hidden and output) is assumed recurrent, and the old activation of the incoming unit used in establishing the value of the input [20].

The recurrent units should offer some short-term memory capacity for the robot by preserving details of activation at time (t) for use in deciding activations at time (t+1). The decay of the activation units is total per unit time.

I felt this network was large enough to provide the discrimination necessary for achieving what is a fairly complex task whilst at the same time not proving too processor hungry -this consideration must once again be emphasised.

The input nodes receive normalised input (in the range $[0,1] \subset \mathbf{R}$) from the light sensors; a unit's activation is the intensity registered at its light sensor; the ranges being from 1 at maximum possible light intensity (close to and facing a wall in the room) to 0 when no light is being received (close to and facing the automaton) at the sensor.

[19] although most connectionist simulations are, at the present time, virtual, being instantiated on current Von Neumann architecture machines, in the future, with the increasing availability of connectionist hardware, the massively parallel operations of a neural network should reap considerable speed benefits.

[20] if the propagation occurs at time(t) the old activation is defined as activation at time (t-1).

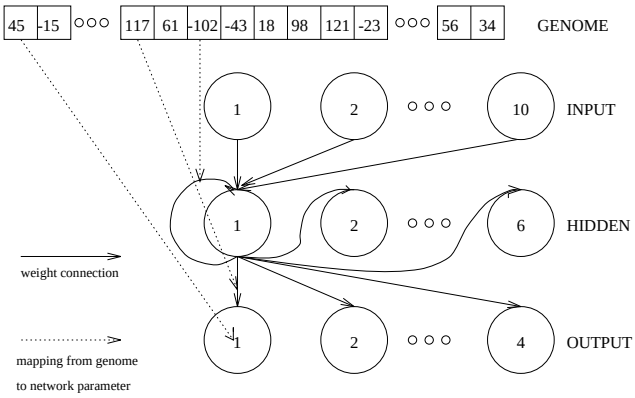


Figure 5: The Standard Neural Network

9.2.1 The Direct Encoded Network

THE PROPAGATION OF THE NEURAL NETWORK WITH DIRECT ENCODING

1. Activation A at input comes directly from light sensors in the range[0,1] $\subset \mathbf{R}$. The output σ of an input unit is its activation A.

2. The activation for a hidden unit i at time t:

$$A_{i(t)} = \sum_i \sigma_{j(t)} W_{ij}\, x + \sum_h \sigma_{h(t-1)} W_{ih}\, x$$

   where j is an input unit, h a hidden unit and W_{ij} represents the weighted connection from unit j to unit i. x is a random multiplier in the range [0.9,1.1] $\subset \mathbf{R}$. The activation passed via the recurrent connections between the hidden units is one time phase behind (t - 1) the current activation. This provides for a limited short-term memory.

3. The output for each hidden unit

$$\sigma_{h(t)} = \frac{1}{1 + e^{-A_h(t)}}$$

   The sigmoidal function approximates a step or threshold function.

4. The activation for each output unit o at time t:

$$A_{o(t)} = \sum_h \sigma_{h(t)} W_{oh}\, x$$

5. The output for each output unit

$$\sigma_{o(t)} = \frac{1}{1 + e^{-A_o / c(t)}}$$

   where c is the temperature or gradient constant for the sigmoidal, defining over which range the function approximates a linear function.

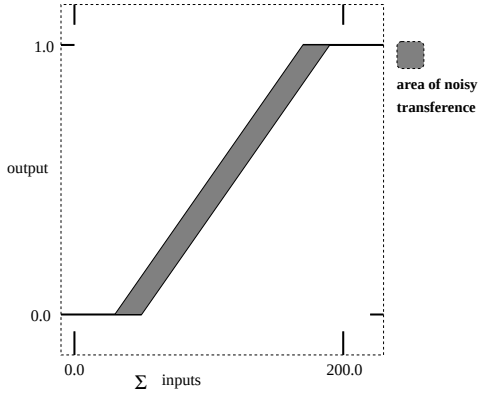


Figure 6: The Noisy transference sigmoidal function

The activation at the input units is propagated to the hidden units via weighted connections whose parameters are established by the evolved genome -as are all weight and threshold parameters in the standard encoding scheme (See Fig 5). Each hidden unit also receives weighted recurrent input from itself and the other hidden units. This weighted input is summed and fed through a sigmoidal function which at a temperature of 1 approximates a step function with a threshold established by the permanently active bias unit or weight. Each hidden unit therefore is registered as 1 or 0 -the return values of the sigmoidal.

Each output unit j receives weighted input from every hidden unit. This is fed through a sigmoidal with a temperature adjusted to give a variable motor response. Within the temperature bound, the sigmoidal approximates a linear function ($y = mx + c$, where c is the established threshold and m is the gradient of the sigmoidal between (-temp to +temp)) and the output value of the function is proportional to the input value (within those bounds). This should offer a fairly smooth range of output values between 0 and 1. Since the motor output for each wheel is obtained by subtracting one of the four output activations from another, this should provide each wheel with a fairly smooth choice within the range maximum reverse to maximum forward. Noise is added at all points in the network giving a noisy transference function for the hidden and output units (see Fig 6).

9.2.2 The Hybrid Encoding

As is shown in Fig 1 the *hybrid encoding* allows for flexibility in the structure of the network. Whilst the number of input, hidden and output units is the same for the *strong direct encoding*, the weight connections (and their weights) are established by the genome. The forward propagation of the network takes into account these connections; hidden to hidden, output to hidden or output to output connections are recurrent; the activations used (the old activations) being one time phase behind the current activations. Noise is also added at all stages of propagation.

9.3 The Importance of Noise

The 'real world' is a noisy environment; information is often corrupted, vision occluded, hazy, unfocussed etc. Environments are constantly changing, unpredictable. A mathe-

matically formulated simulation which failed to take into account this 'fuzziness' in the real world would be of questionable relevance. The ability to cope with noisy input must be seen as fundamental to the useful functioning of a robot.

Harvey, Husbands, Cliff (see [2,9]) have also used noise to practical effect in massively recurrent neural network controllers where the noise is used to sustain 'generator' nodes which act to provide constant intra-network stimulus.

In my uniform noise at ± 10 % is added at all stages of the network's propagation. As has been mentioned it is hoped this will improve the durability of the network-controller. In the case of paltry input, the recurrent nature of the net means that the robot should eventually do something; which will with any luck take it to a less ambiguous position in the room.

9.4 Vision

9.4.1 Why Vision?

Many ways exist with which to sample information about one's environment; for example sound, touch, taste etc. Within Robotics a common method with which robots have sampled from their environment is through the use of touch sensors or distance sensing equipment -a crude sonar for example. Although useful, a method such as the use of whiskers or bumpers -which register contact with an object- is limited in its potential; an object must fall within the range of these tactile sensors if it is to play any part in the robot's behaviour and this range is necessarily limited in practice. Also, the information returned is necessarily crude; discrimination is limited usually to simple proximity detection. A common problem encountered is that of sensory blindness; where no objects fall within the range of these sensors and the robot is left with no information about it environment with which to compute a behaviour (see [9]).

Of all the ways with which to absorb information about one's environment vision seems the most profound and informative; its medium is after all the fastest in the universe. Depending on the complexity of the sensing equipment employed, discrimination can be almost arbitrarily fine. Unlike tactile detectors there are almost unlimited ways of improving the resolution of the robot's immediate environment.

The vision employed by the guard-robot is necessarily crude; since the ray-tracing techniques used are processor-intensive. The sensors are really just intensity indicators, averaging for an area of the environment.

9.4.2 Simulating Vision

Ray tracing is the technique I used to enable the robot to absorb information from its environment via the medium of light. It is without doubt the most popular of all the methods that can be used to simulate light in a computerised environment.

The environment is two dimensional and 2-D vector mathematics was employed to calculate intersections etc. The technique involves extending a ray [21] from the light sensor and calculating the object that it first intersects with (in this case one of the walls of the arena or the automaton) [22].

[21] a line defined by an initial point (the coordinate of the light sensor involved) and a direction vector (ten rays are traced from each sensor at different directions and covering a specified arc-angle of, typically, ninety degrees).

[22] Advanced ray tracing would extend the journey of this ray in a recursive procedure by noting its interaction with the object (dependent on indexes of refractivity and other qualities of the wall) and then

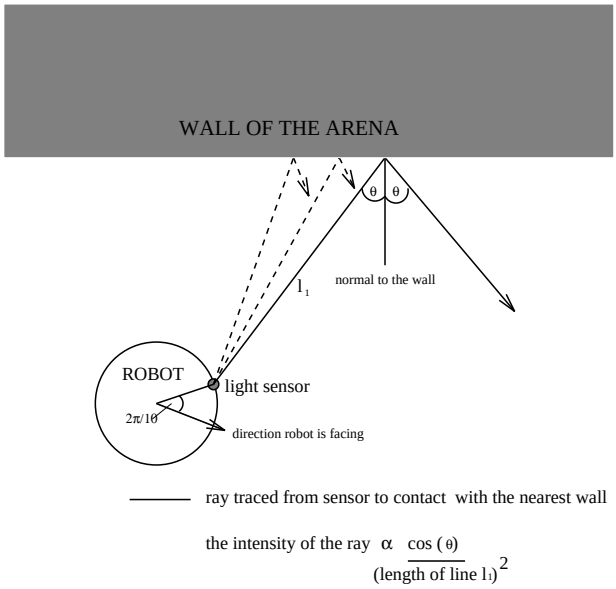


Figure 7: The rays are projected from the light sensor and the intensity registered is a a product of their first encounter with an object (the automaton or a wall).

The simplified procedure I employed assumed that the light in the simulation radiated at an intensity of 1 from the arena walls. Each point on a wall could be assumed to be a point source of light directed at a normal to the wall. The automaton was assumed totally dark (any ray intersecting with its circle returning an intensity of 0) for maximum contrast with walls. The intensity value returned by a ray was calculated by taking into account only its first collision with an object.

In Fig 7 the ray leaves the sensor and intersects with the wall of the arena. The intensity registered is proportional to the cosine of the incident angle (θ) divided by the square of the distance from sensor to the point of intersection (the inverse square law for the radiation of light) times a constant c which regulates the distance from a wall at which a ray perpendicular to that wall will return the maximum intensity of 1. Using vector maths these values are calculated and an intensity for that ray returned. The intensity registered by the sensor is the average of the ten rays traced over the arc-angle.

The ray tracing employed was necessarily simplistic (see comments above on the importance of limiting the processor time necessary) but the general noisiness of the simulation - noise being injected at all phases of the operation of the neural network - does not require much more then a crude intensity indicator, and in this respect the procedure works well. Results obtained (see below) show that it provides enough information for the robot to successfully negotiate a guard-dog behaviour. Within the limits imposed by the available processor time it has been made as veracious as possible a simulation of light interaction.

following the multiple divergent rays this interaction might produce; each of these rays would in turn be followed and their progress recorded until their contribution (in terms of adjusting the eventual light intensity registered) was deemed insignificant, according to a pre-established threshold. The contribution of an individual ray to the final intensity would be ultimately decided by one or many light sources in the simulated environment.

Part III
The Experiment

10 The Interface

A modest X-Windows interface was designed to allow monitoring of the GA's progress. This interface provided for the initiation and subsequent review of a GA. From the interface it is possible to initiate a GA and review its progress or that of an already evolved population (specified in the command line); monitoring of the behaviour of the fittest in the population, a graphical representation of the GA's performance and the spatialised fitnesses of the population.

11 The fitness function

The task at hand requires the robot to guard the centre of the arena from intrusion. The further away from the centre the automaton can be kept, the better the robot is doing its job. I decided to employ a Gaussian function ($\mathcal{G}$) giving an optimal fitness per unit time of one if the automaton is confined to the corner of the arena (as far from the centre as possible) and approximately zero (using a suitable radius of Gaussian) if the automaton is at the centre:

$$\mathcal{G} = e^{\frac{-r^2}{c}}$$

where r is the distance of the automaton from the centre and c is a constant chosen to ensure a fitness return of approximately 0 for a centralised automaton.

Fitness is added at each time cycle giving an optimal fitness, for 400 time units, of 400. However, speed constraints and the general noisiness of the environment mediate against fitnesses of over 200. An optimum fitness give the relative severity of the fitness function (its non linearity means that maintaining the automaton at around half the optimal distance from the centre accrues far less than half the optimal fitness) is around this 200 mark.

12 Preliminary Details

A number of important parameters in the experiment are not under the control of the GA. Possibly the most important is the arc angle over which each light receptor collects light. I chose 90 degrees as a reasonable compromise (in Fig 3 angle β set at 45 degrees) and though this is probably sub-optimal to the task (for which possibly the most important element is discriminating the automaton from the background) it remained constant so that comparisons as to the effectiveness of the different encoding schemes could be made.

The starting positions for the robot and automaton are shown in Fig 8. The automaton starts at the centre with the robot placed such that the automaton falls just outside its radius of influence. At time t with the automaton at the centre, the fitness function (see Section) returns approximately 0, so in the number of time steps allocated (400 was chosen since this meant the 2000 or so breeding cycles could be completed in a realistic time) the robot's centre must enter into the circle of radius 100 (its radius of influence) from the

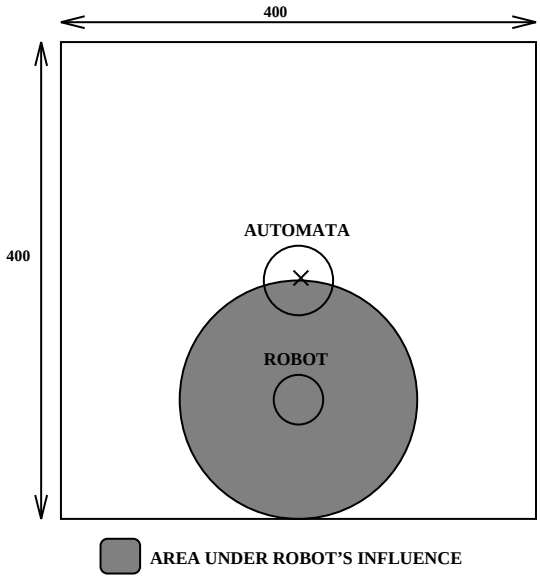


Figure 8: The Initial Positions in the Arena

centre of the arena if it is to effect the automaton and register a significant fitness for the trial. This discriminates heavily in favour of robots attracted to the automaton or with a tendency to centralise themselves in the arena -both good strategies.

13 Calculating the Fitness of an Offspring

An offspring is chosen by applying the algorithm for a spatialised GA (see section above on choosing a GA). The fitness of the offspring is then decided by giving it a number of trial runs in the arena:

13.1 The Trial Based system

The starting position for each trial is as described above but the initial orientation of the robot is random (a value in radians in the range $[0,2*\pi] \subset \mathbf{R}$) to encourage durable guard-dog behaviour; a robot which just moved forwards regardless of input might do well every now and then but could not be said to be guarding anything.

With the aim of a durable guard-behaviour in mind, the robot was run for six trials and its worst performance selected; it is hoped this would discriminate against occasionally lucky but generally ineffective robots and encourage those with a generally positive behaviour. To do well it is necessary for the robot, regardless of its original orientation, to move the automaton from the centre at each trial. Fitness can be accrued at all points in the trial and ineffective periods of behaviour (dalliance in the corner for example) are thus punished.

14 Normalising the Automaton's response

Making the repulsion of the automaton relative to the distance between its and the guard-robot's centre encouraged very fast robots to develop. Using epicycles to remain localised

at the centre of the arena whilst turning very quickly proved a very effective strategy and the fittest of the early simulations all possessed a variant on this strategy, with marginally different sizes of epicycle and speeds (see Results below).

In order to encourage more varying behaviours, the decision was taken to normalise the speed of the automaton; its repulsion being constant regardless of the speed of the robot.

By mediating against the quick epicyclists the hope was to encourage behaviour which focussed more on maintaining contact with the automaton rather than the 'fast and dirty' behaviour witnessed up to this point; where subtleties of approach were possibly being subsumed by messy, but effective speed strategies.

15 Results

Most of the results obtained are from simulations employing a normalised repulsive response from the automaton (see above for a rationale). Time prevented me from running as many simulations as I would have liked but, nevertheless, I feel those obtained do point out some interesting comparisons; between the *strong direct encoding method* and what I refer to as the *hybrid encoding method.* I include representative results from the simulations, in the form of graphs, showing the performance of the simulations over 2000 breeding cycles, the spatialised populations after 2000 breeding cycles and the performance of the best robot from these populations at this point. All these are obtained from the X-windows interface.

There is also an attempt to analyse the behaviour of two robots in more detail, including their performance without noise (the presence of uniform noise being an important element of the experiment); with relevant graphs showing motor impulses over time.

15.1 Results with Repulsion Relative to Distance

As has been mentioned, making the automaton's repulsion inversely proportional to its distance from the guard-robot produced similar strategies in the GAs run:

The form of movement most often chosen was epicyclical (see Fig 9 and Fig 10). This seems to be a good compromise motion between maintaining a high velocity (the repulsion of the automaton being relative to the velocity of the robot) and remaining centralised (colonising the centre of the arena is almost an optimal strategy for the robot). The robot was observed to re-orient its epicycles in the direction of the automaton thus slowly forcing the automaton further from the centre of the arena. At a critical moment, the two would interact violently and the automaton be repelled towards the centre. At this point the robot would quickly recolonise the centre and the cycle begin again.

This form of repulsion was discarded quite early on in favour of a normalised repulsion by the automaton; it was hoped this would encourage more varied strategies from the robots bred. Whilst the epicyclical strategy proved good in both normalised and non-normalised simulations, it was possible that, by mitigating against the speed specialists, more subtle strategies might be encouraged.

15.2 Results with Normalised Repulsion

With normalised repulsion, the automaton is repelled by a constant speed if it falls within the robot's radius of influence (100 units) and the robot is advancing towards it. Whereas before a robot might interact infrequently with the automaton but, due to its speed and

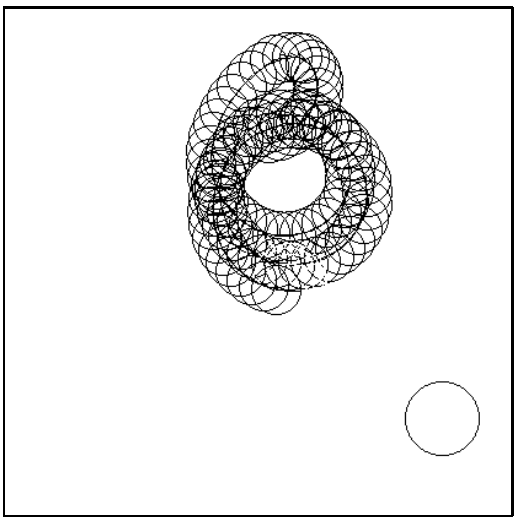


Figure 9: The *epicyclist* makes a sharp turn away from the wall and towards the centre of the arena

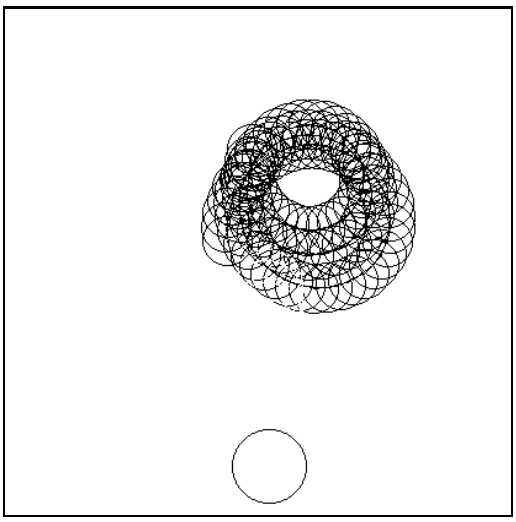


Figure 10: The *epicyclist* begins facing the centre of tre and proceeds to colonise it

proximity, still generate enough repulsion to score a high fitness, now robots maintaining a closer proximity (within 100 distance units) to the automaton are guaranteed more repulsion.

Normalising the repulsion of the automaton proved surprisingly effective at producing varying robot behaviours (see Fig 17 and Fig 19). Optimising speed in a fixed position no longer proves necessarily the best strategy for the robot. While fast robots were observed (see *curly robot* in Fig 9 and Fig 10), speed now appears to be employed more as a means of influencing large areas of the arena in a short space of time [23] whereas before this useful strategic employment of speed was subsumed by the general efficacy of speed as a means of generating proportionate repulsion.

One of the most successful robots bred, I nicknamed the *big dipper* (see below for an analysis). Eschewing the large ellipses employed by *curly robot*, the *big dipper* employs a more direct approach. Its general behaviour is shown in Fig 17. Circling for a few time cycles it then moves rapidly to colonise the centre of the arena. This colonisation is more localised than the previous epicyclical strategies, relying on the robot's continuous presence at the centre, rather than its speed, to score high fitness points (see below for a fuller analysis of the *big dipper*).

15.3 Direct Versus Hybrid Encoding

The availability of processor time prevented a large number of trials being run, but for the purposes of this project three to four trials are assumed representative.

With the parameters fixed such that a comparison could be made [24] simulations were run and relative fitnesses compared.

[23] the larger the epicycles employed by the robot, the more likely that, at some point, the automaton will evade the robot's influence and return to the centre, thus costing the robot valuable fitness points. One way to compensate for this is to increase the speed of the robot.

[24] the light sensors averaging over 90 degrees; the repulsion normalised.

15.3.1 Degrees of Convergence

Of interest in monitoring the performance of the genome populations was the degree of convergence exhibited; the extent to which the fitnesses in a population are similar to one another; a guide to this is how the best performance for the population is reflected in the average performance. In the graphs shown, this can be monitored by seeing how far the best line is from the average. At the end of the 2000 breeding cycles the fitnesses of the spatialised members were also recorded.

In the *strong direct-encoding* the degree of convergence was low (see Fig 11). This was predicted in the section on encoding the GA (see above); since context is radically disrupted by the crossover operation, the possibility that two parents will produce a relatively unfit offspring is far higher than that they will produce a fit one; at any time we would expect to see a fairly high number of weak members in the population. This feature keeps the average of the population relatively low in comparison with the fittest of the group. As can be seen in Fig 12, the best performance achieved by *strong direct-encoding*, even at a fairly late stage, the population still possesses some very poor performers.

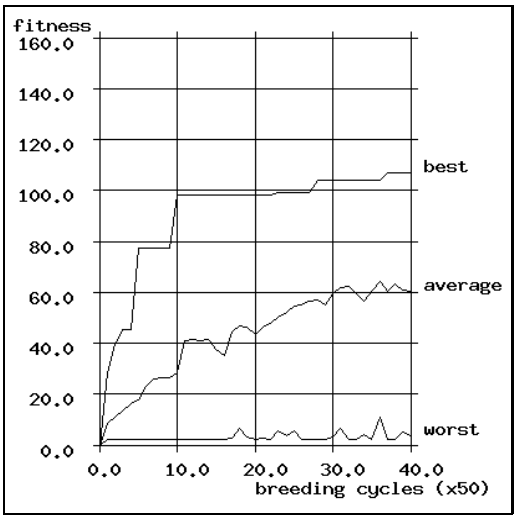


Figure 11: The Best Performance by Direct Encoding

90	55	83	94	81	25	70
52	42	92	98	65	106	79
80	55	104	98	19	64	106
79	27	64	29	38	6	13
77	31	28	79	31	70	98
13	39	91	37	44	41	94
65	73	43	98	39	38	30

Figure 12: The Non-converged Population

In the *hybrid encoding*, where some structural features are preserved through the crossover operation, we would expect to find a higher degree of convergence than that of the *strong direct-encoding*. This was indeed the case as can be seen in Fig 13 and Fig 15 (the related populations are shown in Fig 14 and Fig 16). A dominant approach has emerged through the population. The general ruggedness of the populations in this *hybrid* scheme was impressive when set against that of the *strong direct-encoding*. Whilst the *strong direct* method might occasionally produce a fit or even very fit result it seems that, in the long run, (accommodating the *hybrid method's* larger search space) the fitter population will produce the best results and that the fitter populations result from preserving structural details in the population; the increased search space that this entails [25] is compensated by improved performance; the best the *strong direct-encoding* (see Fig 11)could manage was some way away from the worst of the *hybrid encoded* results.

[25] two numbers are required to establish a weighted link between neural network units, one to establish which unit the link comes from, the other to set its weight. In the *strong direct-encoding* only one number is required since the structural details of the weight-links are pre-established.

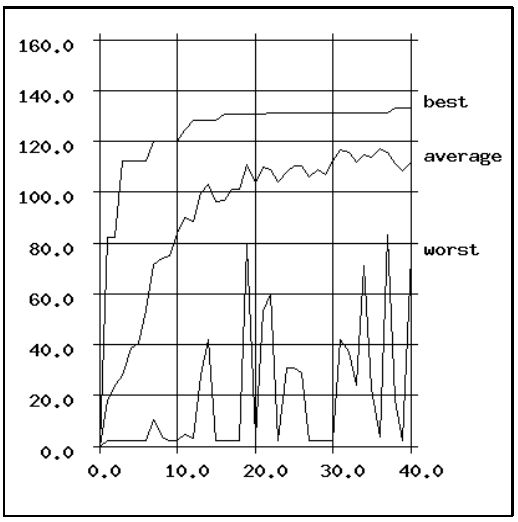


Figure 13: A typical performance by the *hybrid encoding* scheme. This simulation bred the *corkscrew*

126	128	109	126	131	126	124
120	124	108	130	123	127	131
112	118	127	105	111	127	129
116	116	126	104	131	120	129
122	121	133	123	115	109	124
129	113	2	122	114	116	113
125	124	107	112	114	117	123

Figure 14: The Converged Population

15.3.2 Behaviours Evolved

Most of the behaviours evolved in the time available exhibited epicyclical elements (see above and Fig 9 and Fig 10). All those *strong directly-encoded* exhibited this feature. With the *hybrid encoding*, as well as the *big dipper* (see below for an analysis), one of the more interesting behaviours was that of the *corkscrew*. Its behaviour has three elements: a sharp reorientation, whilst moving relatively fast, in the direction of the automaton (shown as an extended corkscrew); occasional large epicycles around the centre of the arena; colonisation of part of the arena by rotating quickly whilst minimising the translation element of motion (see Fig 19, Fig 20 and Fig 21).

15.4 Testing Performance Without Noise

Time available meant that not all the fittest networks evolved could be analysed to ascertain the part noise plays in their performance. However, two of the more interesting robots, nicknamed the *big dipper* and the *corkscrew* were chosen and their motor impulse over the time of a trial monitored.

In order to remove noise from the propagation of the neural network, the NOISE macro was set to 0.0; the noise in the system being multiplied by this factor.

15.4.1 The Performance of the Robots

Part of the theory lying behind the introduction of noise into the system is that it is useful in allowing the robot to escape from ambiguous positions ; where, for example, the external stimulus would cause a noiseless system to vacillate (or maybe be trapped in an oscillatory response). Although the noise is random, the recurrent nature of the network means that noise can build up and provoke a definite response in the robot; by this means the robot can escape from an ambiguous position or be forced over a notional threshold into making, with any luck, a positive response.

The *Big Dipper*

Comparing the performance of the *big dipper* with and without noise, a marked difference can be discerned in its respective performances. The standard action of the *big dipper* is to circle at its starting position for a few time units and then make a quick dart to the centre to adopt a very fit colonisation of this area, circling around indefinitely, though occasionally adjusting its position in response to movements in the automaton (see Fig 17).

The performance of the robot without noise is markedly different (see Fig 18). Its initial circling around the start position is indefinitely prolonged and, in the same number of time units allocated to the noisy trial, it consistently failed to make the very advantageous dart to the centre; although it did progress very slowly in this direction. The noise appears to provide the necessary impetus for the robot to move from its ambiguous circling of the start position to its colonisation of the centre.

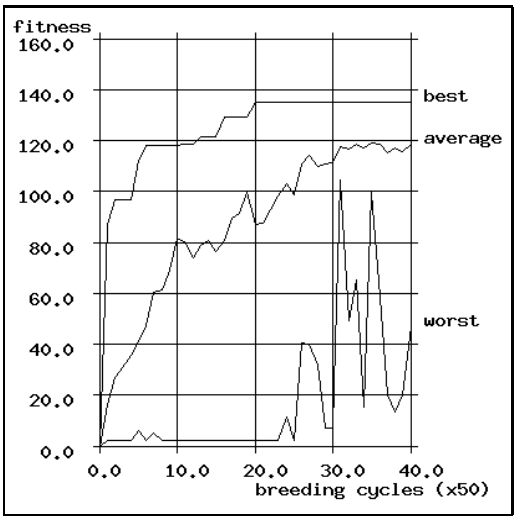


Figure 15: A performance by the *hybrid encoding* scheme. This simulation bred the *big dipper*

120	129	111	113	124	122	128
119	119	127	119	108	121	129
46	126	126	125	117	126	80
116	111	120	103	112	119	126
128	124	109	105	125	118	125
118	129	127	117	115	112	123
126	122	125	122	114	134	117

Figure 16: The Converged Population

The *Corkscrew*

As has been noted (see above), the *corkscrew* exhibits three types of behaviour when noise is added to its neural network controller:

1. sharp corkscrew like turns (see Fig 20)
2. occasional localised rotation (colonisation) (see Fig 21)
3. occasional wide circles (see Fig 19)

The *corkscrew* proved more durable than the *big dipper* when noise was removed from the network. However, some changes were apparent; although behaviour (1) was still exhibited when noise was removed, behaviours (2) and (3) were not; it seems that these were a product of the noisy system (see below for motor analysis).

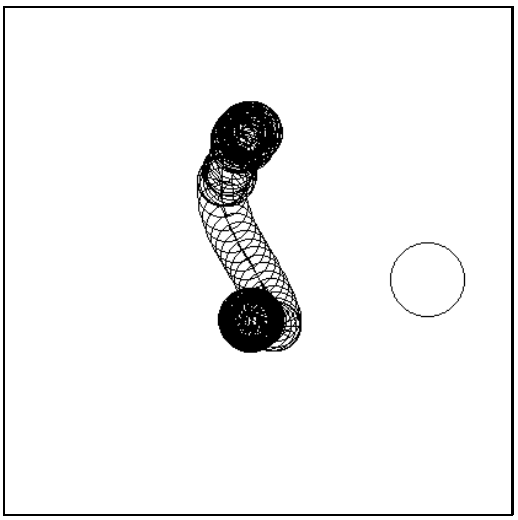


Figure 17: The performance of the *big dipper* with noise injected

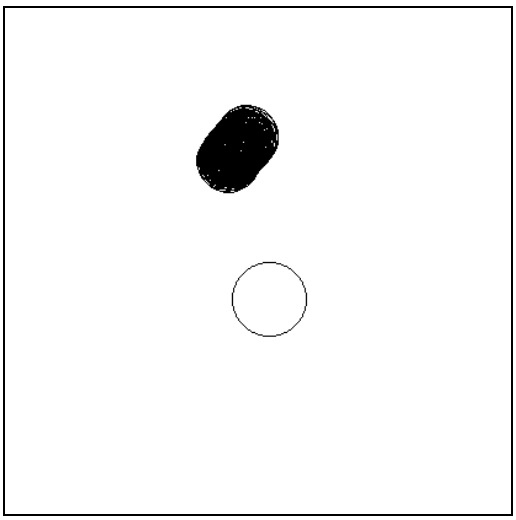


Figure 18: The performance of the *big dipper* with no noise added to the system

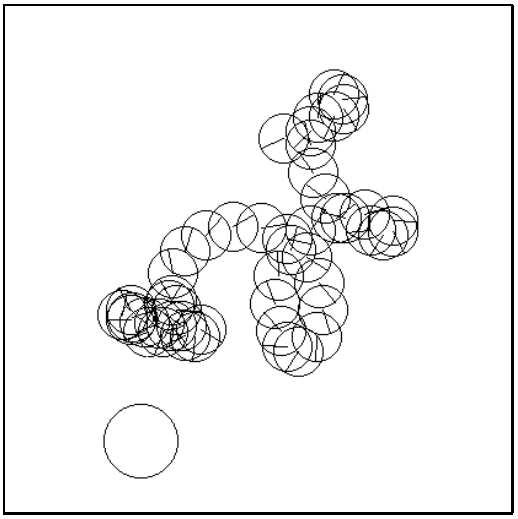


Figure 19: The *corkscrew* traps the automaton in the corner

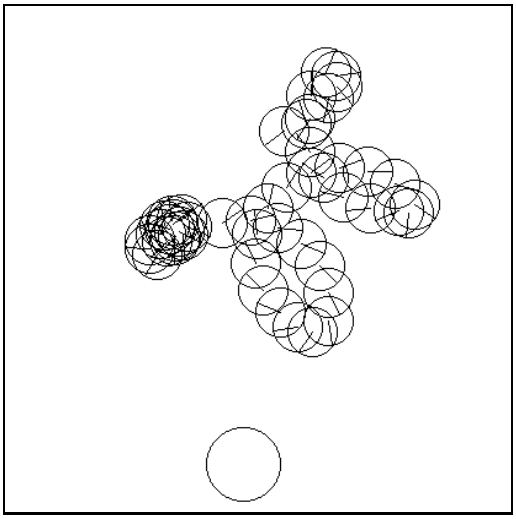


Figure 20: A standard response by the *corkscrew*. Notice the colonising behaviour, shown by the dark circle

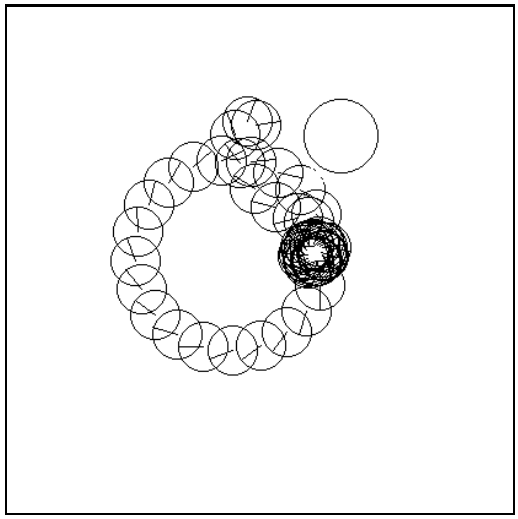


Figure 21: The *corkscrew* colonises the centre

15.4.2 Analysing Motor Response

The *Big Dipper*

In Fig 22 the left and right motor response of the *big dipper*, with noise added, is shown over the first 100 time units of its progression. Its standard motion is characterised above and can be seen in Fig 17. The left motor response remains low throughout the 100 time units, though varying slightly. However, the right motor, normally with a high positive impulse, is characterised by occasional periods of low activity where the graph is seen to dip. Three consecutive trials are superimposed and this dip is a regular feature of all three. During the time when its right motor output is low, the rotational component of the robot's motion is minimised and its translational component increased [26]; in other words it stops spinning and covers a relatively large amount of ground. The sudden spurt towards the centre that characterises the *big dipper's* movement is explained by this sudden diminution in the right motor output.

Fig 23 shows the motor responses of the *big dipper* without noise. The right motor shows a high periodic response which keeps within a fairly small band. The left motor has a small response but it is periodic. This response fits the observed rotational behaviour (see Fig 18). Unlike that in Fig 22 the right motor response does not occasionally dip, allowing for increased translational motion; the noiseless response is that of a robot with a consistently high rotational component of motion and very little translation.

The part noise plays in allowing the *big dipper* to move to the centre seems clear. Recurrencies within the network build up and occasionally are enough to push the robot into making a positive move. This positive action is required to move the robot from an ambiguous position to a more certain one; once it has begun to move towards the centre, interactions with the automaton become increasingly likely and ambiguity diminishes.

Looking at Fig 23 the right motor output oscillates periodically. It is probable that the sudden dip in the noisy motor response (see Fig 22) coincides with the lowest motor impulse in this periodic cycle and that this low point occurs when the robot is facing

[26] the translational component of the robot's movement increases as the left and right motor outputs converge and is at an optimum when they are the same.

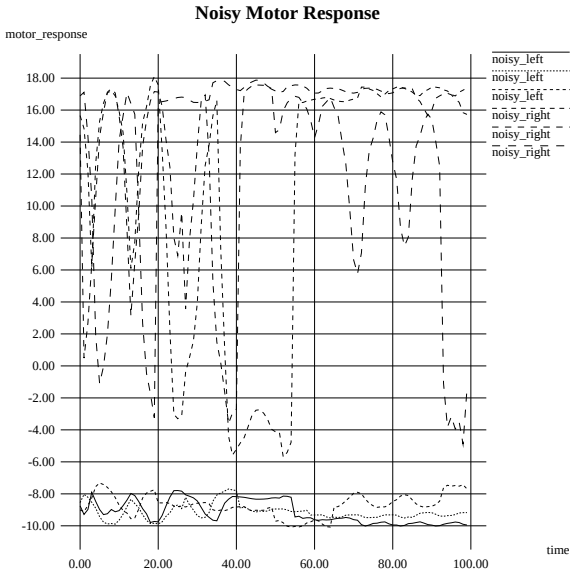


Figure 22: Noisy Motor Response

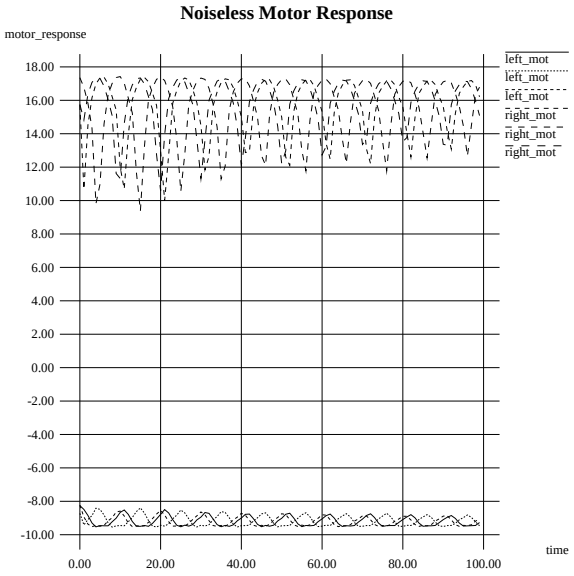


Figure 23: Noiseless Motor Response

towards the centre of the arena; therefore the translation occurs in the direction of the centre of the arena. The noise enables the robot to colonise the centre with a very effective strategy.

The *Corkscrew*

The motor response of *corkscrew*, with and without noise, over 100 time units is shown in Fig 24. The right motor is jammed full on for the whole time in both noisy and noiseless simulations. It is the variation in left-motor response that is responsible for the interesting behaviour noted above.

Looking at the noiseless left motor impulse, it hovers around 17 with occasional large dips to the -15 mark. This is interpretable, assuming the right motor is fixed at 30, as a robot which is moving quite fast (its translation component) whilst turning to the left (its rotation component) and which suddenly takes a sharp left turn (maximising its rotation component with a sharp increase in motor disparities) before continuing on its way. Further analysis is needed, but observation over an extended time shows the sharp turns to be a product of interaction with the automaton [27] tending to force the automaton away from the centre, explaining the high fitness of this strategy.

As has been mentioned above, the noiseless *corkscrew* exhibited the corkscrew behaviour but not the other two elements witnessed in the noisy version.

Looking at Fig 24, the noisy left-motor response shows occasional 'basin' form. Here the robot suddenly executes a sharp turn (as in the noiseless robot) and remains caught in this response, rotating quickly on the spot. This conforms with the colonisation behaviour seen is Fig 21. This colonisation often traps the automaton in a position away from the centre (see Fig 19) and is a way of occasionally toting up considerable fitness points, especially if the automaton is maintained in a corner. The noise in the system must be extending the life of a sharp turn into a colonisation tactic by the built up recurrencies.

The occasional circular sequences observed (see Fig 21) in the robot are concomitant with repression of the sharp left turns (seen as the dip in left-motor response from 17 to

[27] presumably a sudden dip in the light intensity registered by one or more light sensors.

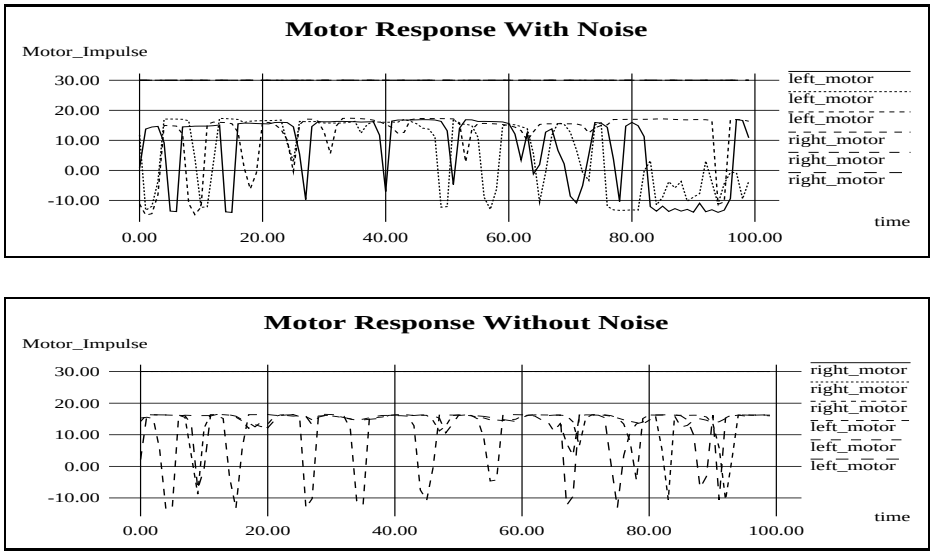


Figure 24: The Motor Response of the robot *corkscrew* with and without noise; three consecutive performances over 100 time units are superimposed

-15). Looking at the third (noisy) left-motor response in Fig 24, this appears to be what is happening. The dips observed are much shallower then normal (down to 7 - 10 as opposed to -15) and fail to provide the impetus for a sharp turn. The extended left turn (with occasional adjustments) leads to a circular behaviour.

The complexity of the *corkscrew's* behaviour is a product of noise in the neural network, the recurrencies building up to provide impulses crucial to both the colonisation and circular behaviour. In this way noise is seen as a potentially useful tool for subverting standard behaviour types into more successful but more complex responses.

16 Areas for Improvement

As has been mentioned , part of the motivation for a project of this nature lies in the belief that, in certain areas of exploration (particularly the field of Alife) human intuition is a poor guide. For my own part, I do not feel qualified to decide the most efficient number of light sensors to be employed in a project such as this or where they should be positioned on a robot. One way that insight might be gained is in looking to the natural world; a heuristic might be derived, along the lines of; "predators have their eyes to the front for optimal focus whilst prey tend to position their eyes on the sides, thus gaining maximum coverage". But though this insight may be useful in the development of certain robots, for a task-specific robot it is too probably crude a guide for developing optimal performance; whilst observations from nature may allow one to establish certain boundary-limits to robotic operational parameters, within these limits it is probably preferable to allow the GA some search capacity.

However, caution should be advocated here. Every new parameter the genome specifies increases the search space and, by implication, increases the time required to establish fit robots. All increases to the genome length should be thought through thoroughly.

Nevertheless, a number of parameters in the simulation were under my control that I feel it would have been better to submit to environmental pressures. For example, in the *hybrid encoding* system the number of connections each unit could receive was pre-established. It would doubtless be far more efficient, and in the long run effective, to allow the GA to establish these parameters [28]. I feel now that I allowed too many weight connections; the larger weight space this involves required more breeding cycles than time allowed for to do it justice.

Though the results achieved with the *hybrid method* were superior to those managed by the *strong direct-encoding* (see Results above), areas such as the number of weight connections allowed to each unit, the possibility of allowing a unit to send impulses to another rather than just receive them (though this would require an extra indicator on the genome), the number of cycles the neural network propagates during each time unit etc., suggest improvements for the future.

17 Conclusion

The genetic algorithm employed proved successful in breeding network controllers that simulated a guard-dog behaviour in the robot. The fitness of the population increased progressively for around 1500 breeding cycles and then was then seen to fall off as further progress became increasingly unlikely.

Though the environment simulated was necessarily simplistic, this does not invalidate the results achieved. Within the large search space defined, the GA was able to breed neural network controllers that enabled the robot to perform its specified guard-behaviour with some efficacy. The task is of suitable complexity that I feel a human trying to design an intermediary, between the information from the robot's light sensors and its subsequent response, that performed the task as well would be faced with a considerable challenge; certainly one greater than designing a spatialised GA to breed an attempted solution.

While this project has shown that the approach of breeding network controllers has some efficacy, the success of the *hybrid encoding* scheme (see Results) over the conventional *strong direct-encoding* approach highlights the ever-present possibilities for innovation. The more parameters left under the effective pressures of natural selection, the better the results seem to be. This seems in validation of the belief that, in some areas, natural selection is a more effective tool than human intuition. An obvious next step would be to place as many parameters as possible under the control of the GA, towards the aim of limiting, as far as possible, any rigid structural assumptions imposed from above [29] ; my intuition is that the freer the GA to explore the more impressive will be its eventual solution.

The time available meant that a control experiment, using a more conventional GA (in this context, non-spatialised) could not be run, and, as such, no figures are available for comparing relative performance. It is only speculation, but the fact that diverse approaches to the problem (see Results) did surface leads me to believe that the spatialised GA fulfilled at least this aspect of its remit.

[28] in [4] an example is given of a variable length genome; the genome being extendible allows it to increase its own search space. This makes regulating genetic operations a more complex operation, but the advantages in terms of flexibility seem to outweigh this aspect.

[29] However, as mentioned, care must be taken not to expand the search space too much. The incremental approach favoured by Harvey, Husbands and Cliff (see [2],[4] and [9]) where the genome is of variable length and able to increase its own search space points towards a possible solution to the problem of initially overloading the genome.

REFERENCES

[1] Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* Ann Arbor, MI: University of Michigan Press, 1975.

[2] I. Harvey, P.Husbands and D.T. Cliff. Issues in evolutionary robotics. In J.A.Meyer, H. Roitblat, and S. Wilson, editors,*Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92).* MIT Press Bradford Books, Cambridge, MA, 1992. In Press.

[3] M. Eigen, P. Schuster.*The Hypercycle: A Principal of Natural Self-Organization.* Springer-Verlag, 1979.

[4] I. Harvey. Evolutionary robotics and SAGA: the case for hill crawling and tournament selection. Technical Report CSRP 222, University of Sussex School of Cognitive and Computing Sciences, 1992.

[5] Goldberg, D.E. *Genetic Algorithms in Search, Optimization and machine learning.* Reading, MA: Addison-Wesley, 1988.

[6] Belew, R.K. McInerney, J., and Schraudolph, N.N. Evolving networks: Using genetic algorithms with connectionist learning. Technical Report CS90-174, The University of California at San Diego, La Jolla, California 92093, 1991.

[7] Werner, G.M., and Dyer, M.G. Evolution of communication in artificial organisms. In Farmer, J.D., Langton, C., Rasmussen, S.,and Taylor, C., editors, *Artificial Life II.* Reading, MA: Addison-Wesley,1991.

[8] Moriarty, D., and Miikkulainen. Evolving complex othello strategies using marker-based genetic encoding of neural networks. Technical Report AI93-206, The University of Texas at Austin, Austin, TX 78712-1188, 1993.

[9] D.T. Cliff, P.Husbands, and I. Harvey. Evolving visually guided robots.In J.A.Meyer, H.Roitblat, and S.Wilson, editors,*Proceedings of the Second International Conference on Simulation of Adaptive Behaviour (SAB92).* MIT Press Bradford Books, Cambridge, MA, 1992. In Press.

[10] I. Harvey. The puzzle of persistent question marks: a case study of genetic drift. Technical Report CSRP 278, University of Sussex School of Cognitive and Computing Sciences, 1993.

[11] D.T.Cliff Computational Neuroethology: a provisional manifesto. Technical Report CSRP 162, University of Sussex School of Cognitive and Computing Sciences, 1990.