Issues in Evolutionary Robotics

I. Harvey, P. Husbands, D. Cliff

CSRP 219, July 1992

Cognitive Science Research Paper

Serial No. CSRP 219

The University of Sussex
School of Cognitive and Computing Sciences
Falmer
Brighton BN1 9QH
England, U.K.

A version of this paper appears in:
Proceedings of SAB92,
the Second International Conference on Simulation of Adaptive Behaviour
J.-A. Meyer, H. Roitblat, and S. Wilson, editors,
MIT Press Bradford Books, Cambridge, MA, 1993.

Issues in Evolutionary Robotics

Inman Harvey[1] **and Philip Husbands**[1] **and Dave Cliff**[1,2]
[1]School of Cognitive and Computing Sciences
[2]IRC, School of Biological Sciences
University of Sussex, Brighton BN1 9QH, U.K.
`inmanh` or `philh` or `davec`, all `@cogs.susx.ac.uk`

Abstract

In this paper we propose and justify a methodology for the development of the control systems, or 'cognitive architectures', of autonomous mobile robots. We argue that the design by hand of such control systems becomes prohibitively difficult as complexity increases.

We discuss an alternative approach, involving artificial evolution, where the basic building blocks for cognitive architectures are adaptive noise-tolerant dynamical neural networks, rather than programs. These networks may be recurrent, and should operate in real time. Evolution should be incremental, using an extended and modified version of genetic algorithms. We finally propose that, sooner rather than later, visual processing will be required in order for robots to engage in non-trivial navigation behaviours.

Time constraints suggest that initial architecture evaluations should be largely done in simulation. The pitfalls of simulations compared with reality are discussed, together with the importance of incorporating noise. To support our claims and proposals, we present results from some preliminary experiments where robots which roam office-like environments are evolved.

1 Introduction

This paper firstly gives an analysis which proposes that an evolutionary approach to the design of robots can be expected to supercede design by hand; it then explores issues arising from this, and presents results from some preliminary experiments: using an extended genetic algorithm, we have evolved control architectures for a simulated version of a physical robot constructed at Sussex.

An evolutionary approach to real robotics was discussed at a 1987 workshop (3), and in the context of subsumption architecture by a student of Brooks (27); but no practical results have been reported. A number of researchers have shown the method to be viable for simulated robots in highly simplified simulated worlds (1), but have not had to face the exponential increase in complexity that follows with progress from toy worlds into the real world.

Independently in 1991, we (as members of the Prance consortium) made a research proposal to use an evolutionary approach in developing real autonomous robots (24); and Brooks proposed at Ecal-91 in Paris a different evolutionary approach, using genetic programming (9).

Straightforward Genetic Algorithms (GAs) use evolutionary ideas for function optimisation, and are not immediately applicable to robotics. Necessary adaptations to GAs are discussed in (14). This paper concentrates on other issues, in particular whether the cognitive architecture of a robot should be evolved in the form of a Behavioural Language, as advocated by Brooks, or in the form of artificial neural networks. We argue that there are good reasons for the latter approach.

After setting the stage with these theoretical considerations we go on to report on preliminary simulation experiments in evolving control networks for simple robots equipped with a few touch sensors. The simulations are not naive – they are based on observations of a real robot and attempt to model the physics of its interactions with the world.

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. This is largely because of the traditional implicit assumption of functional decomposition — the assumption that perception, planning and action can be analysed independently of each other.

In contrast, recent work at MIT bases robot control architectures around *behavioural decomposition* (6, 8). In theory, this involves analysing independent behaviours of a robot or *animat*,[1] 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 suppression or inhibition mechanisms.

It is extremely difficult to foresee all possible interac-

[1] *Animat*: simulated animal or autonomous robot (28).

tions with the environment, and between separate parts of the robot itself (8, 22). Designing appropriate cognitive architectures is a task with inherently explosive complexity. Complexity is likely to scale much faster than the number of layers or modules within the architecture — it can scale with the number of possible interactions between modules.

To design cognitive architectures for robots with emergent behaviours hence requires either (a) a computationally intractable planning problem (10) 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 for real robots doing useful things are currently being reached.

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. Genetic Algorithms (GAs) (12) 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.

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 either type of decomposition, the evolutionary process will determine whether in practice either one, or neither, should characterise the robots' cognitive architecture.

4 An incremental, species approach

An animal should not be considered as a solution to a problem posed 4 billion years ago. Nevertheless, 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 of an existing *animat*, not as a search through the complete space of *animats*. The basis for extending standard GAs to cope with this has been worked out in (14). The implications are that the population being evolved is always a genetically-converged *species*; and that increases in genotype length, associated with increases in complexity, can only happen very gradually.

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 we advocate is that of substituting evolution for design.

5 The use of simulation

Artificial evolution requires that the members of a sizeable population must be evaluated over the course of many generations. In the case of the evolution of autonomous robot control systems, to date it has been assumed it would take far too long to do all of these evaluation in the real world (15, 9, 24, 3). Instead it is suggested that most evaluations should be done in simulation. In the short to medium term this seems a sensible strategy but we have strong doubts about its long term viability.

Assuming the use of simulation for the time being, it is crucial that it is kept as closely in step with reality as possible. A number of techniques can be used to this end. Firstly, the simulation can be calibrated at regular intervals by carefully testing the architectures evolved in the real robot. Serious discrepancies should be ironed out. Secondly, accurate simulations of the inputs to the robot sensors and the reactions of the actuators should be based on carefully collected empirical data. Thirdly, and above all, noise must be taken into account at all levels. In order to acquire the desired level of accuracy it may be necessary to use a mixed hardware/software simulation in which simulated signals are fed into hardware sensors or actuators and the response is read directly. The use of low resolution sensing makes this approach feasible. It is important to remember that it is not our world that is being simulated, but the robot's.

A range of unstructured dynamic environments should be used in the simulation. A cognitive architecture that has evolved to cope with a range of such environments is much more likely to be robust than one evolved to operate in a single well structured world.

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 preprocessing is painfully slow on today's hardware. Techniques to test many generations of control systems in real worlds will have to be developed. We are currently pursuing the development of one such technique: see (11) for further details.

6 What should we evolve?

So far we have not addressed the question of what exactly it is that is being evolved. There are at least three useful ways to implement the control system of an autonomous robot:

- An explicit control program, in some high level language;
- A mathematical expression mapping inputs to outputs, e.g. a polynomial transfer function;
- A blue-print for a *processing structure*, a network of simple processing elements.

6.1 High Level Programs

In (9), following a suggestion by Langton, Brooks proposes using an extension of Koza's genetic programming techniques (18) as the method for evolving a physical or simulated robot.

One potential problem with evolving a programming language is that, if it supports partial recursion, programs to be evaluated may never halt, unless some arbitrary 'time-out' is imposed. Brooks' Behaviour Language (7) does not use partial recursion, and hence can be evolved without this problem. Subject to the qualification that Genetic Programming should have genotype length changes restricted to small steps his approach at first sight seems reasonable, but we have two broad objections.

The first is that any such programming approach treats the 'brain' as a computational system, producing a set of motor outputs for any given set of sensor inputs. This snapshot view of cognition has been the main paradigm in AI, but we support an alternative view of agents as dynamical systems rather than computational systems, which are perturbed by their interactions with the environment, which is also a dynamical system. This view is expressed in (19, 5, 26), and will not be developed further here.

The second objection, which is supported by our simulation results, is that the primitives manipulated in the evolutionary process should be at the lowest level possible, and this is in contrast to Brooks' use of higher level languages. The Behavior Language, BL, is in effect a blueprint for a network of Finite State Automata, and the target language Brooks proposes for Genetic Programming is an even higher level language, GEN, which can be compiled into BL.

Our intuitions are based on the notion that any high level semantic groupings necessarily restricts the possibilities available to the evolutionary process, compared to the alternative of letting the lowest level of primitives be manipulated by genetic operators. The human designer's prejudices are incorporated within their choice of high-level semantics, and these restrictions give rise to a much more coarse-grained fitness landscape, with steeper precipices. It might be thought that the use of low-level primitives necessitates enormously many generations of evolution with vast populations before any interesting high-level behaviour emerges, but our simulations show that this is not the case at all.

A further factor concerning high-level languages is that the injection of noise into anything other than the lowest levels becomes difficult to justify. For a network considered to be modelled at a physical level it is easier to justify the insertion of noise at many points within the system, and as will be seen this appears to have valuable effects, not least in making the fitness landscape more blurred and hence less rugged for evolution.

6.2 Polynomial Transfer Functions

There are a number of close relationships, in this context, between polynomial transfer functions and artificial neural networks, not least that the input-output associations of most neural networks can be arbitrarily closely approximated by a polynomial function and vice versa. Clearly the problem of brittleness and halting is not an issue, yet neither scheme is computationally restricted.

Even for modest numbers of inputs and outputs, the most useful transfer function may be a highly complex non-linear expression with many terms; the search space is potentially very large. Simulation results suggest that the search space, except for low dimensions, lacks structure, resulting in the GA degenerating into random search. Similar results for the related problem of system identification have also been reported (16).

The robustness required for useful behaviour in the real world is almost certainly going to demand either some degree of adaptation or an expression complicated enough to cover a wide enough range of situations. The latter leads to the problems described above. The former will require auxiliary systems identification algorithms which will seriously complicate matters by being computationally expensive (25) and requiring error measures.

6.3 Neural Nets

Evolutionary approaches to designing connectionist network architectures are manifold, e.g. (17, 13, 21, 23); All these have used some form of genetic algorithm to search through a pre-defined finite space of possible network architectures. In other words, at a more or less sophisticated level, the basic architecture has been defined with some parameters left as variables, and the GA has been used to tweak the parameters to optimal values. It is argued in (14) that for the equivalent of robot evolution it will be necessary to extend this to open-ended evolution instead, with significant implications. Nevertheless, the evolvability of connectionist networks in general is clearly established.

It might be argued that in practice connectionist networks are simulated on a serial computer; and in turn that a serial Turing machine can be simulated with a connectionist network. This does not mean that their evolvability is the same. 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.

Concise specification on the genotype of sub-networks or modules which may be repeatedly used is possible, provided that there is a mechanism to interpret such specifications several times analogously to the way subroutines are called within a program. The desirability of adaptation has been mentioned above, and obviously artificial neural networks allow for this. The massively parallel nature of neural nets enables very fast implementation in the appropriate hardware, in contrast to the necessarily (locally) serial interpretation of a behaviour language.

What sort of network?

There are good grounds for thinking that a generalised form of connectionist network could be one very appropriate class. Let us start with three basic axioms:

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 axioms apply to any physical subvolume of the whole brain.

A justification for the third axiom is that of the incremental development of the whole by alterations and additions over evolutionary timescales. The consequence of these axioms, 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 means that this is not a field theory. It rules out of this model such more general methods of physical interaction as might be assumed to be involved with, e.g. diffuse 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 (20) is that timelags between nodes need to be specified. But a whole new universe of possible dynamical behaviours is opened up by this extension.

Such networks are more difficult to analyse than standard feedforward ones. However 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.

A particular type of network falling under this general classification, and used in the experiments described in this paper, is described in more detail in section 9.1.

7 Timing issues

The practical problems of timing should be taken note of. The robot will have timing circuitry to synchronise sensing, control and motor activities. This should not cause any undue problems for the implementation of evolved neural networks. As long as they operate using discrete time intervals, then even complex recurrent networks can be handled in a straightforward manner. The more general and possibly more powerful class of asynchronous continuous time networks are a little more difficult but create no significant problems. Arbitrarily complex polynomial transfer functions, which may involve a lengthy computation, are certainly more difficult to handle than discrete time networks. Potentially non-halting high level programs with many conditional

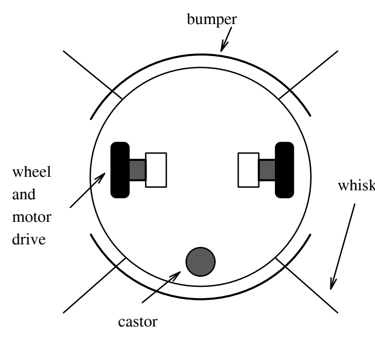


Figure 1: Plan view of simple six-sensor robot.

branches are harder still.

8 Robots need vision

In autonomous robot navigation, a number of sensor technologies are commonly employed to enable the task of navigation. For the purposes of the discussion that follows, it is useful to employ the distinction found in the biology literature between *exteroceptors* and *interoceptors.* An exteroceptor is a receptor (i.e. sensor) which detects stimuli external to an animal, e.g. light or sound; an interoceptor senses or detects stimuli that arise inside the animal, e.g. blood pressure.

The development of a successful navigating robot depends on finding a satisfactory combination of exteroceptors and interoceptors. Here the discussion focusses on the exteroceptors we envisage necessary for our robot.

The majority of recent projects in autonomous behavioural[2] robotics have not employed vision as a primary exteroception mechanism. Most commonly, mechanosensory "whiskers" and "bumpers", or active ranging devices (such as ultrasound depth sensors or laser light-stripers) have been employed. Such sensors are essentially *proximal* sensing devices. That is, they only provide reliable data for the immediate surroundings of the robot. Such robots are thus forced to employ primitive navigation strategies. The most common such strategy is "wall following", where the robot must always maintain sensory contact with a sizeable static external surface, such as a wall of the robotics lab. Wall-following robots that lose sensory contact with all external surfaces often suffer from sensor-blindness, the chief symptom of which is a significant degradation, or total loss, of navigation ability.

In some restricted behavioural or ecological niches, wall-following is a satisfactory navigation strategy, and sensor-blindness can be overcome by wandering until sensory contact is re-established. The need for more sophisticated navigation competences, which we take as manifest, is only likely to be overcome fully by an increased reliance on *distal* sensors — in particular, vision.

[2]I.e., subsumption-based or reactive-systems robotics.

There is growing research in the field of mounting computer vision systems on mobile robotic platforms – an approach referred to as *animate vision* (2). While many projects are underway in developing animate vision systems, we are not aware of any where evolution is employed in preference to design.

Although using vision does not *eliminate* problems such as sensor-blindness, it does provide a rich source of information concerning an agent's external environment. Whether designing or evolving a visual system, a number of factors have to be taken into account. Significant factors include:

- The discretization of the sampling of the optic array, i.e. how many pixels do we want in the images our robot samples, and what sort of geometry should the image have (a square raster is not necessarily convenient). The number of pixels in the image has a manifest effect on the bandwidth of the visual processing channels.
- The angular extent of the vision system's field of view – should the robot be equipped with 360^o vision or will a more restricted field of view suffice?
- The visual angular resolution of the robot's optics. Should the vision system employ a uniform resolution, or have some sort of spatially variant "foveal" (nonuniform) vision system? Many animals have resolution which varies across the visual field. Typically this is a result of the need for high-resolution vision for certain tasks (predation or identifying mates) coupled with a need for a wide field of view, sampled at a lower resolution.

That many animals, particularly insects, successfully occupy their ecological niches using low-resolution low-bandwidth vision as a primary source of exteroception information indicates that such an approach (as opposed to high resolution and bandwidth) is worth exploring within an evolutionary robotics context, in the first instance at least.

Simulation vs. Reality in Vision

For the evolutionary approach to be successful, methods of varying the details of the visual sampling and the subsequent processing of the visual signal are imperative.

Evolutionary learning can be accelerated if the populations undergoing evolution exist within a simulated system. The problem here is in ensuring that the simulated visual systems correspond in a useful manner to the physical visual systems with which the robot will be equipped. While such simulations are possible in principle, the computational demands soon become considerable and, unless the necessary processing hardware (e.g.

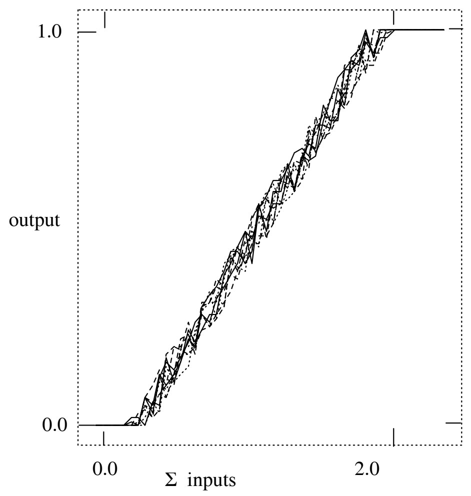


Figure 2: Noisy neuron transfer function.

specialised graphics pipeline processors) is readily available, it is envisaged that physical instantiation of the visual systems would be required at an early stage.

For a physical robot to be equipped with a vision system that has variable sampling bandwidth, geometry, visual extent, and angular resolution, requires that the camera(s) mounted on the robot are capable of offering performance at the upper limits of what is envisaged necessary. For example, a high-resolution image can be subsampled (with averaging or smoothing) to provide a lower-resolution image. This could be done under genetic control, which is a very crude first step towards evolving sensors.

Whether the image-capture mechanism is real or virtual, the image-processing scheme can be simulated (i.e. the parallelism can be simulated rather than embodied in truly parallel hardware). But as the imaging bandwidth increases, or if the robot's speed of reaction is critical, specialised image processing hardware (with suitably adjustable parameters) would be required.

Envisaging what is necessary for the robot is likely only to be possible after some experience with it: a circularity which reveals that some iteration is required between the simulation work and the building of real robots – a pluralist approach will be the most fruitful.

9 Preliminary Experiments

A real robot assembled in the Engineering Department at Sussex has been simulated using the methodology established above. The behaviour of the motors propelling the wheels has been modelled for the outputs, as have inputs from whisker and bumper touch sensors. Simulation of a low resolution insect-type visual system has been added and results using that are described in another paper (11). This is part of an ongoing project at Sussex to develop an evolutionary approach to robotics, with increasingly sophisticated tasks leading to navigation using learnt visual landmarks.

The first phase of the work explored the methodology using careful simulations. Results from this are presented here. We are now into the second phase of the work which will directly calibrate the simulations using the real robot. A further phase, described in (11), also just begun, will look at the evolution of visually guided behaviours without using simulations at all.

A plan view of the robot used in the simulation experiments is shown in Figure 1. The robot is cylindrical in shape with two wheels towards the front and a trailing rear castor.

The wheels have independent drives allowing turning on the spot and fairly unrestricted movement across a flat floor. The signals to the motor can be represented as a real value in the range $[-1.0, 1.0]$. This range is divided up into five more or less equal segments, depending on which segment the signal falls into, the wheel will either: remain stationary; rotate full speed forward; full speed backward; half speed forward; or half speed backward.

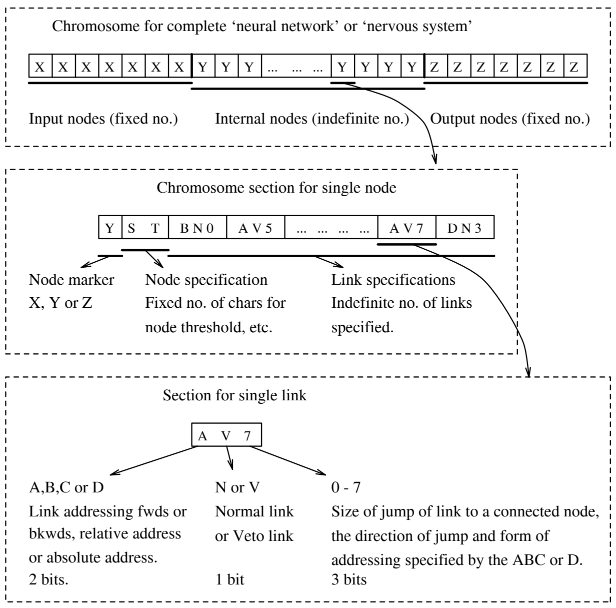
The aim of the experiments was to evolve 'neural-style' networks to control the robot in a variety of environments. Before going on to describe the experiments in detail, the particular type of neural networks used, and their genetic encoding, will be described.

9.1 The neural networks

As explained in Section 6.3 we advocate the use of continuous real-valued networks with unrestricted connections and time delays between units. These can be thought of as something like analogue circuits with real-valued signals continuously propagating through the connections. Our experience, and also that of others (4), is that this sort of network can support a range of behaviours, depending on its exact couplings with the world, and so is highly adaptive without using Hebbian-type weight changes or the like.

The particular networks used in the experiments have a fixed number of input nodes, one for each sensor, and a fixed number of output units, two for each motor. As all the units are linear threshold devices with outputs in the range $[0.0, 1.0] \subset \mathbf{R}$, two units are needed to give the motors a signal in the range $[-1.0, 1.0] \subset \mathbf{R}$. If the output signals from these four output units are labelled S_{o1}, to S_{o4} then, the left motor signal is given by $S_{o1} - S_{o2}$, and the right motor signal is given by $S_{o3} - S_{o4}$.

Each unit is a noisy linear threshold device. Internal noise was added because we felt it would provide further useful and interesting dynamical properties. Any physical implementation of our nets would be likely to include naturally occurring noise anyway. The input-output relationship for such a node is shown in Figure 2, which was generated by plotting the output for a fixed set of inputs ten times and overlaying them.



Chromosome for complete 'neural network' or 'nervous system'
X X X X X X X Y Y Y ... ... Y Y Y Y Z Z Z Z Z Z Z
Input nodes (fixed no.)
Internal nodes (indefinite no.)
Output nodes (fixed no.)
Chromosome section for single node
Y S T BN0 AV5 ... ... AV7 DN3
Node marker X, Y or Z
Node specification Fixed no. of chars for node threshold, etc.
Link specifications Indefinite no. of links specified.
Section for single link
A V 7
A,B,C or D Link addressing fwds or bkwds, relative address or absolute address. 2 bits.
N or V Normal link or Veto link 1 bit
0 - 7 Size of jump of link to a connected node, the direction of jump and form of addressing specified by the ABC or D. 3 bits

Figure 3: The genetic encoding scheme

Within the networks two types of connection are allowed: normal and veto. 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 connection 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 connection outputs from b are turned off (though in the current implementation, further veto outputs are not affected). The veto threshold is always much higher than the lower threshold for the normal signal. The veto mechanism is a crude but effective model of phenomena found in invertebrate nervous systems.[3]

As well as the input and output units, each network will have some number of 'hidden' units. This number is not prespecified – the genotypes can be a variable length. The genetic encoding specifies properties of the units and the connections and connection-types emanating from them. It is now described in more detail.

9.2 The genetic encoding

The genetic encoding used is illustrated in Figure 3.

The genotype 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

[3]For example, feed-forward inhibition of the locust LGMD visual interneuron acts as a veto to prevent the LGMD from producing transient responses caused by delays in earlier processing. See e.g. (29, pp.77-78).

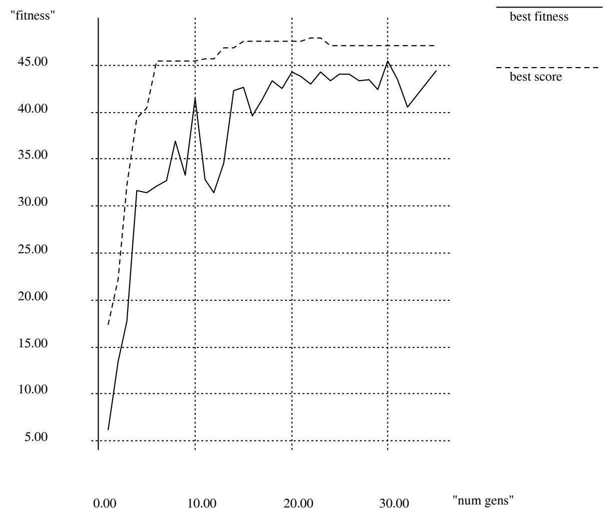


Figure 4: Results of simple experiment. See text for further explanation.

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.

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.

9.3 The physics: simulating movement

In the experiments described next the continuous nature of the system was modelled by using a fine-time-slice simulation. At each time step the sensor readings are fed into the neural network. The continuous nature of the networks is simulated by running them (synchronously updating all units inputs and outputs for a number of iterations (about 100), with a variance to counter distorting periodic effects) and then converting the outputs to motor signals. The new position of the robot is then calculated by using the appropriate kinematic equations. Using the wheel velocities, the motion is resolved into a rotation about one wheel plus a translation parallel to the velocity vector of the other. Standard Newtonian mechanics are used. However, the motion is not modelled as being wholly deterministic: noise is injected into the calculations. Collisions are handled as accurately as possible — using observations of the real system. The

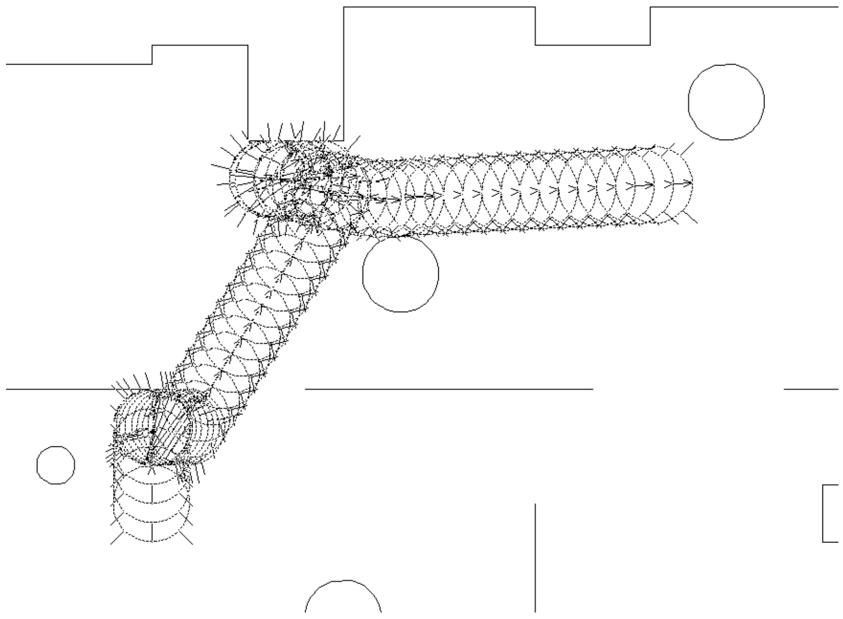


Figure 5: Motion of a single robot controlled by an evolved network.

nature of the collision depends on speed and angle of incidence as well as the shape of the obstacle.

This type of simulation is not perfect, it can and will be made more accurate, but we feel it is realistic enough to take our results seriously.

9.4 The experiments

Each of the following experiments was run for 50 generations, each with a population size of 40. The crossover rate was set at 1.0, while the mutation rate was of the order of 1 bit per genotype.

Figure 4 shows the results for an experiment in which a control network was evolved using an evaluation function which encouraged wandering in a cluttered office-type environment containing walls, pillars and doorways (see Figure 5). Robust control networks were favoured by scoring each genotype several times for a single fitness evaluation. The robot was always started from the same position in the same orientation, and was scored on how far away from its starting position it moved in a fixed time period. On each scoring run the robot faced a different set of situations because of the noise in the system. The *minimum* of the scores achieved was taken as the fitness value for the genotype. The robots were started from rest with no initialising signals; internal noise was sufficient to allow fitter nets to settle into useful initial states. The bottom line on the graph shows the fitness of the best individual in each generation. The top line shows the best score achieved by any member of the population for any of the runs making up its evaluation set. The fact that these two lines converge indicates that more and more robust networks began to appear. Clearly very good control networks have evolved for this simple specific task. Figure 5 shows a short run by a robot controlled by one of these networks. As a matter of convenience, the robot's whiskers are shown moving through objects. It can be seen that the network generates a 'move in a straight line at full speed' behaviour when in free space, and various rotational movements when presented with obstacles. Members of earlier generations had far more random behaviours, spending most of their time in messy collisions or just sitting still.

Figure 6 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. This time the robot was always started at random locations with a random orientation. Note the robot turns fairly smoothly on encountering obstacles. In earlier generations collisions were much more messy.

Figure 7 gives interesting comparative results for different fitness functions based on the above evaluation function. It shows the best, average and worst scores of the best individuals per generation scored over its evaluation set as in the previous experiment.

The upper graph was obtained by taking the fitness to be the *average* of the small number of runs in the evaluation set; the lower graph was obtained by taking the fitness to be the *worst* of the runs. Each run started from a random position with a random orientation. The results clearly 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

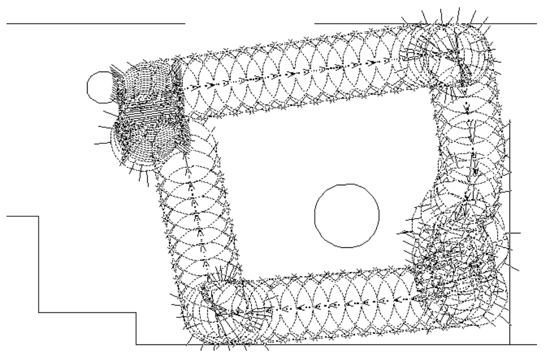


Figure 6: Motion of a robot evolved to maximise the area of the bounding polygon of its path over a limited time period.

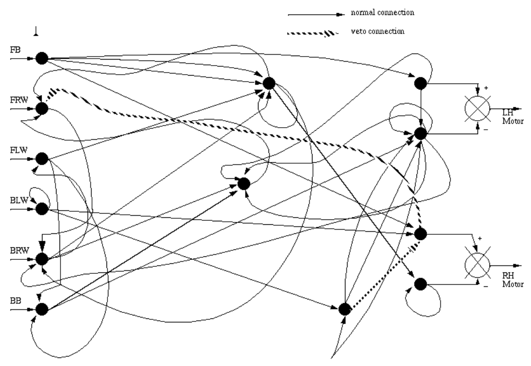


Figure 8: An evolved network: no work of art, but a working robot controller.

and average much closer together, providing a far more robust solution.

Figure 8 shows a network evolved in this second experiment. It is fairly complex with many feedback loops, but it is interpretable in terms of generated behaviours. If it reminds you of a bowl of spaghetti without the bolognese sauce and chianti, this is probably partly due to the fact that there is no term in the evaluation functions that penalises unnecessary links. However, initial populations are started with individuals having (randomly) one or zero internal nodes; the number can only grow gradually if that promotes greater fitness. We expect that more concise networks will result if we introduce a cost for link creation in the evaluation function, and allow for the possibility of non-unity time delays and/or weights on connections.

These early experiments with primitive behaviours have clearly been successful: we have built on them by evolving networks for sighted robots; further details of the work involving vision are given in (11).

10 Conclusions

There is no evidence to suggest that humans are good at designing systems which involve many emergent interactions between many constituent parts. But robust control systems for robots may well fall under this classification. Artificial evolution seems a good way forward, and it has been advocated in this paper.

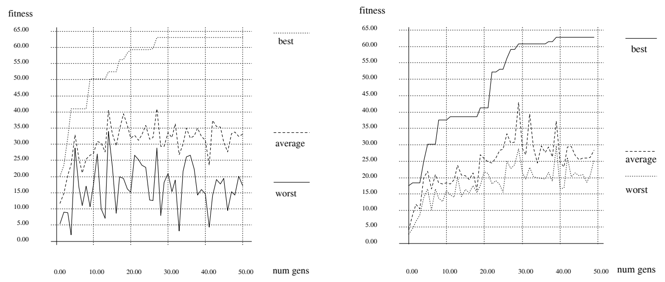


Figure 7: Comparative results for different fitness functions. Left-hand graph is where fitness is measured by the *average* of a series of tests; right-hand graph where fitness is measured as the *worst* performance in a series of tests. Abscissa is generation number; ordinate is fitness value. See text for further explanation.

Results from realistic simulation experiments have been presented. They lend weight to our claim that an incremental artificial evolution is a viable methodology.

Acknowledgements

Many thanks to Tony Simpson for the engineering design work on the robot, and to Linda Thompson for her help in the preparation of this paper. Inman Harvey has been supported by the SERC. This work is currently supported by a University of Sussex research development grant.

References

[1] 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.* Addison Wesley: volume XI in the series of the Santa Fe Institute Studies in the Sciences of Complexity, 1991.

[2] D. H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–86, 1991.

[3] 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, 1987.

[4] R. D. Beer and J. C. Gallagher. Evolving dynamical neural networks for adaptive behaviour. Technical Report CES 91–17, Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, Ohio, 1991.

[5] R.D. Beer. A dynamical systems perspective on autonomous agents. Technical Report CES-92-11, Case Western Reserve University, Cleveland, Ohio, 1992.

[6] R. A. Brooks. Achieving artificial intelligence through building robots. A.I. Memo 899, M.I.T. A.I. Lab, May 1986.

[7] R. A. Brooks. The behavior language. A.I. Memo 1227, MIT AI Lab, 1990.

[8] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[9] 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.

[10] David Chapman. Planning for conjuctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.

[11] D. T. Cliff, P. Husbands, and I. Harvey. Evolving visually guided robots, 1992.

[12] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Massachusetts, USA, 1989.

[13] 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, California, 1989. Morgan Kaufmann.

[14] 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.

[15] 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.

[16] Timothy Johnson and Philip Husbands. System identification using 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.

[17] 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.

[18] 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.

[19] H.R. Maturana and F.J. Varela. *The Tree of Knowledge: The Biological Roots of Human Understanding*. Shambhala Press, Boston, 1987.

[20] J. L. McClelland and D. E. Rumelhart, editors. *Explorations in Parallel Distributed Processing*. MIT Press/Bradford Books, Cambridge Massachusetts, 1986.

[21] 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*, San Mateo, California, 1989. Morgan Kaufmann.

[22] H.P. Moravec. The Stanford Cart and the CMU Rover. In *Proc. of IEEE*, volume 71, pages 872–884, 1983.

[23] 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.

[24] 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.

[25] T. Soderstrom and P. Stoica. *System Identification*. Prentice Hall, 1989.

[26] Tim van Gelder. What might cognition be if not computation. Technical Report 75, Indiana University Cognitive Sciences, 1992.

[27] P. Viola. Mobile robot evolution. Bachelors thesis, M.I.T., 1988.

[28] 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*. Lawrence Erlbaum Assoc., 1985.

[29] D. Young. *Nerve Cells and Animal Behaviour*. Cambridge University Press, Cambridge, 1989.