Incremental Evolution of Neural Network Architectures for Adaptive Behaviour

D. Cliff, I. Harvey, P. Husbands
CSRP 256, December 1992

Cognitive Science Research Paper

Serial No. CSRP 256

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

Incremental Evolution of Neural Network Architectures for Adaptive Behaviour

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

Abstract

This paper describes aspects of our ongoing work in evolving recurrent dynamical artificial neural networks which act as sensory-motor controllers, generating adaptive behaviour in artificial agents. We start with a discussion of the rationale for our approach. Our approach involves the use of recurrent networks of artificial neurons with rich dynamics, resilience to noise (both internal and external); and separate excitation and inhibition channels. The networks allow artificial agents (simulated or robotic) to exhibit adaptive behaviour. The complexity of *designing* networks built from such units leads us to use our own extended form of genetic algorithm, which allows for incremental automatic evolution of controller-networks. Finally, we review some of our recent results, applying our methods to work with simple visually-guided robots. The genetic algorithm generates useful network architectures from an initial set of randomly-connected networks. During evolution, uniform noise was added to the activation of each neuron. After evolution, we studied two evolved networks, to see how their performance varied when the noise range was altered. Significantly, we discovered that when the noise was eliminated, the performance of the networks degraded: the networks *use* noise to operate efficiently.

1 Introduction and Rationale

Increasingly, practitioners of artificial neural network research are realising that both the complexity of model neurons, and also the styles of network architecture, need to be extended beyond those employed in the much-cited work of the early 1980's. Certainly, models such as Hopfield networks, or back-propagating multi-layer perceptrons, played an important historical role in making parallel distributed processing an acceptable paradigm of study; but if we are to succeed in either understanding biological nervous systems, or in building artificial neural networks which exhibit intelligent behaviour, it is likely that we will have to move to more complex models.

But what form should this complexity take? The notion of 'complexity' is often highly subjective, and hence problematic. We should definitely avoid introducing unnecessary complications, but (more importantly) we should not be deceived by our own simplifications. In artificial neural network (ANN) modelling, simplifications are made for various reasons. Often, there are issues of mathematical tractability: certain model neurons or network architectures are easier to formally analyse than others. In other cases, the ease

with which the models can be simulated or built in available hardware is an important factor, and appropriate simplifications are made. In either case, it is important to note that the 'simplification' is made for *our* convenience: the ANN is easier to construct or understand. The problem with this approach is that in using simplified models, we may actually be making life harder for ourselves as scientists; because the tasks we try to make our models perform may, by their very nature, require greater complexity than is possible without using clever 'trick' techniques, or large and unwieldy modular assemblies of simple networks.

There are two simplifications which are very common in ANN models: most models in the literature have very simple (or non-existent) dynamics; and arbitrary connectivity is often avoided. It is manifestly clear that networks with many feedback connections and delays between units are much more challenging to either analyse, simulate, or design, than are networks such as the common three-layer back-propagation network. Yet for many interesting and important problems, feedback and intrinsic dynamics are almost definitely what is required. Furthermore, there is ample evidence in the neuroscience literature from most branches of the animal kingdom, that biological neural networks exhibit rich dynamical behaviour and exploit feed-back connections to great effect.

Additionally, many ANN's are developed purely to transform between representations or encodings which have been formulated by their designers. Such networks may be worthwhile engineering artefacts, performing useful computations; but it is important to remember that the primary evolutionary pressure on the development of biological nervous systems (which we seek to understand or draw inspiration from) was whether a particular nervous system helped an animal survive in environments which were dynamic, uncertain, and often hostile. That is to say, nervous systems evolved where they generated *adaptive behaviours* (i.e. behaviours which are likely to increase the chances that the individual animal survives to reproduce). We, in common with a growing number of other researchers, believe that the generation of adaptive behaviours should form the primary focus for research into cognitive systems, and that issues of purely transforming between representations or encodings are, at best, secondary.

It is the above factors that have influenced our recent work, discussed in the remainder of this paper. We have created ANN's which generate adaptive behaviours in artificial "animals" (i.e. robotic or simulated agents). Our agents have tactile sensors and minimal visual systems (two oriented photoreceptors). The ANN's use highly recurrent networks of artificial neurons (called "units"), with propagation delays as signals pass across links between units. The units have separate excitation and inhibition channels, and operate in the presence of noise introduced both internally (i.e. within each unit) and also externally (i.e. in sensory-motor transduction). The transfer functions for excitation and inhibition in each unit are nonlinear with discontinuities in the first derivative.

Naturally, either analysing or designing networks composed of such units is a challenging and difficult task. Nevertheless, we believe that units of the sort used in our work are closer to the minimum complexity acceptable for generating adaptive behaviours than are the simpler units of prior work. For this reason, the problems of design and analysis have to be tackled, rather than avoided by introducing simplifications. Our approach has been to, as far as is possible, *automate* the design of the networks by employing our own

extended form of genetic algorithm, known as SAGA. Whereas most genetic algorithms are essentially performing optimisation in a fixed parameter space, SAGA allows for the *dimensionality* of the parameter space to be under evolutionary control, by employing variable-length genotypes. In terms of the networks, this means we are able to start with a population of agents each of which has a minimal number of units: extra units may be introduced by mutation, and will only be retained if they increase the evolutionary success of the mutated agent: our automatic network generation is *truly incremental*.

The rest of this paper discusses the neuron model and network simulations in more detail. Following this, details of the how the networks are encoded as genes suitable for use with SAGA are given. Next, we discuss the adaptive behaviour evolved in our simulated agents, and present brief analysis of how the performance of the final evolved networks alters as the internal noise level is varied. For further details of our rationale, see [2, 9], and for full details of the visual sensing employed, see [4].

2 The Model Networks

Because our networks are recurrent, there is no clear divide between different 'layers' (c.f. input, hidden, and output layers found in back-propagation networks). Nevertheless, for the purposes of generating adaptive behaviour, it is necessary to designate some units as receiving input from sensors, and others as producing outputs to actuators (such as motors). As is discussed in [3], this designation may be distorted by the evolutionary processes. The remainder of this section discusses details of the neuron model, and how the networks architectures are encoded as 'genes' which can be operated on by the SAGA genetic algorithm.

2.1 The Neuron Model

The neuron model we have employed in our work to date has separate channels for excitation and inhibition. Values propagate along links between units, and are all real numbers in the range $[0, 1]$. All links are subject to a delay Δt. Figure 1 shows a schematic block diagram of the operations within a single model neuron. Unusually, the inhibition channels operate as a 'veto' or 'grounding' mechanism: if a unit receives *any* inhibitory input, its excitatory output is reduced to zero (but it can still inhibit other units). Excitatory input from sensors or other units is summed: if this sum exceeds a specified veto threshold t_v, the unit produces an inhibitory output. Independently, the sum of excitatory inputs has uniform noise (distribution: $[-n, +n] \in \mathbf{R}$) added internally, and is then passed through an excitation transfer function, the result of which forms the excitatory output for that unit, so long as the unit has not been inhibited.

More formally, the excitation transfer function $\mathcal{T}$ takes the form:

$$\mathcal{T}(x) = \begin{cases} 1 & \text{if } x \geq t_u \\ 0 & \text{if } x \leq t_l \\ (x/(t_u - t_l)) - (t_l/(t_u - tl)) & \text{otherwise} \end{cases}$$

where t_l and t_u are lower and upper threshold levels. The veto output function $\mathcal{U}$ takes

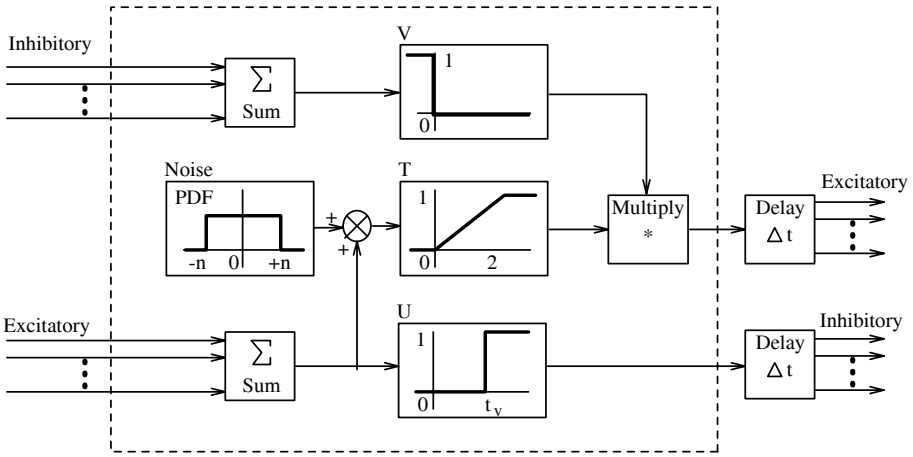

Figure 1: Schematic block diagram showing operations within a single model neuron. See text for further explanation.

the form:

$$\mathcal{U}(x) = \begin{cases} 1 & \text{if } x \geq t_v \\ 0 & \text{otherwise} \end{cases}$$

where t_v denotes the veto threshold, and the veto input function $\mathcal{V}$ is:

$$\mathcal{V}(x) = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{otherwise} \end{cases}$$

Because there are separate excitation and inhibition channels, two connectivity matrices are required: a veto matrix V (where each element $v_{i,j}$ indicates whether unit i vetoes units j), and an excitatory matrix W, with elements w_{ij}. Then, if $o_{ej}(t)$ and $o_{vj}(t)$ respectively denote the excitatory output and veto output from unit j at time t, and $N_j(t)$ denotes the internal noise injected to unit j at time t, the output channels from unit j can be expressed as:

$$o_{vj}(t) = \mathcal{U}(\sum_{\forall i} w_{ij} e_{vi}(t - \Delta t))$$

$$o_{ej}(t) = \mathcal{V}(\sum_{\forall i} v_{ij} o_{vi}(t - \Delta t)) \cdot \mathcal{T}(N_j(t) + \sum_{\forall i} w_{ij} o_{ei}(t - \Delta t))$$

In most of our work, we have used: $t_l = 0.0$; $t_u = 2.0$; $t_{io} = 0.75$; and noise $n = 0.1$; for all units. The dynamic properties of these units are simulated using asynchrous fine time-slice approximation techniques, with random variations in time-cycling to counteract periodic effects. Significantly, we have found that this neuron model is sufficiently sophisticated that there has been no need to introduce variable weights on the links between units, or variable delays: in all our work, weights and delays are all set at one, for all links in the network. Nevertheless, we are actively investigating the use of placing such parameters within evolutionary control.

2.2 The Genetic Encoding

To enable the use of SAGA, the network architecture has to be encoded as a 'gene'. In our work to date, we have used a genetic encoding scheme which stores the "wiring diagram" (connectivity data) for the network as a string of alphanumeric characters. For further details of the encoding, see [9]. The encoding has been developed to be robust with respect to the mutation and crossover operators, where 'robustness' indicates that if a new gene is formed via crossover and mutation from two "parent" genes, and the parent genes encode valid networks, then the resultant new network will also be valid. A network is valid insofar as all the links in the network connect one unit to another: for each individual agent, the control network is initially randomly connected (but valid). It is our evolutionary learning algorithm, SAGA that develops these random networks into useful control architectures.

Briefly, SAGA differs from other genetic algorithms in that it allows for variable-length genotypes, which allow for the *dimensionality* of the search space to be varied under evolutionary control. Other authors have explored the use of genetic algorithms in creating sensory-motor controllers for adaptive behaviour (e.g. [1, 5]), but (as far as we are aware) all such work has involved using genetic search in a parameter space of *fixed* dimensionality: a relatively constrained optimisation task. For example, in [1], Beer and Gallagher use a genetic algorithm to find parameter values for a dynamical neural network controlling hexapod (six-legged) locomotion. For the subnetwork controlling each leg, there are 40 free parameters (thresholds, time constants, and weights) [1, p.107], the values of which are found by genetic search. Constraints are introduced so that, for the full six-legged network, only 50 free parameters need be searched, rather than the 990 that would be required if the six leg-controller subnetworks were fully interconnected with each other [1, pp.107–108]. The important factor here is that the researchers have introduced *a priori* constraints to reduce the size of the fixed space searched by their genetic algorithm. In contrast, we use SAGA to search a parameter space whose dimensionality is, for practical purposes, *not predefined*: the number of connections and internal "hidden" units in a network specifies a search-space of a particular number of free parameters, but by allowing variable-length genotypes, extra connections or hidden units may be introduced, which has the effect of increasing the dimensionality of the search space. However, search will only be maintained in the higher-dimensional space if it leads to fitter network architectures: it is perfectly feasible that, under particular fitness evaluation functions, fitter solutions may lie in lower-dimensioned search spaces. Thus, we can start with relatively minimal randomly-specified valid network architectures, and rely on the use of SAGA to establish whether larger networks with more free parameters are required. It is in this sense that we refer to SAGA as *truly incremental*.

One notable feature of the SAGA approach is that the initially random population of individual genotypes converges, over evolutionary time, to a situation where the population is evolving as a *species*. See [8, 6, 7] for full details.

3 Evolving a Visually Guided Robot

Here we briefly present some recent results. We attempted to evolve networks for a simple adaptive behaviour, which was for a simulated[1] visually guided robot to spend as much time as possible in the centre of a circular arena. The robots have two independent drive-wheels and a third free-wheel. The drive wheels may go at either full or half speed, either forwards or reverse, so the robot is capable of rotating on the spot, or travelling in wider-radius circles, or in straight lines, or stopping still.

Each robot had six tactile sensors: two 'bumpers' (at front and back), and four radially symmetric wire 'whiskers'. The tactile sensors are primarily of use in detecting collisions with walls of the arena, and appropriately reorienting. The robot also has two directionally-sensitive photoreceptors, which allowed it to visually sense its environment (the walls of the simulated arena are dark, while the floor and ceiling are light).

Each individual robot was positioned at a randomly chosen point near the edge of the arena, in a random orientation. The robot then had a fixed finite 'lifetime', in which it had to get as close to the centre of the arena as it could, and then stay there. The robot's performance was evaluated by taking the gaussian function of a discrete temporal integral of its distance from the arena-centre during its lifetime: the higher the evaluation function, the more time the robot spent at or near the centre. As is demonstrated in [4], this is sufficient to evolve controllers for visually-guided behaviours: no explicit specification of visual processing is required.

We created a population of 60 robots with initially random genes, and evaluated each one over eight 'lifetimes'. At the end of the evaluation, we took the robot's *worst* score as a measure of its performance (best and average scores are too often deceptively high). When all 60 robots had been evaluated 8 times, the genes of the higher-scoring robots were 'inter-bred' using SAGA principles to create a new generation of 60 individuals. We repeated this process for 100 generations.

The typical behaviour of a robot controlled by an evolved network is that it finds its way to the centre of the circular arena, and then stays there by spinning on the spot. This is a perfectly acceptable strategy, given that the robots were evaluated only on the basis of how much time they spent at the centre, and not on the basis of how much energy they used. In this paper, we will consider two of the best evolved robot controller networks, referred to as C1 and C2. The architectures of the two networks are shown in Figures 2 and 3. Typical behaviours exhibited by the robots controlled by these networks are shown in Figures 4, 5, and corresponding time-plots of sensor inputs, internal activations, and motor outputs, are shown in Figures 6 and 7.

As can be seen, the networks do not resemble the sort of networks which are traditionally published in the literature, but then this network was *not* designed by a human: it evolved according to Darwinian principles. Yet the plots of the activity of the networks clearly indicate that the robot is approaching the centre of the arena and staying there: they achieve their specified task. For further analysis of these networks, see [3]. The next

[1] The simulations involve accurate physical and kinematic models of a real robot constructed at Sussex. Vision was simulated using ray-tracing with anti-aliasing via 16-fold super-sampling. See [4] for further details.

section discusses our findings from studies of varying the amount of internal noise in the neuron model.

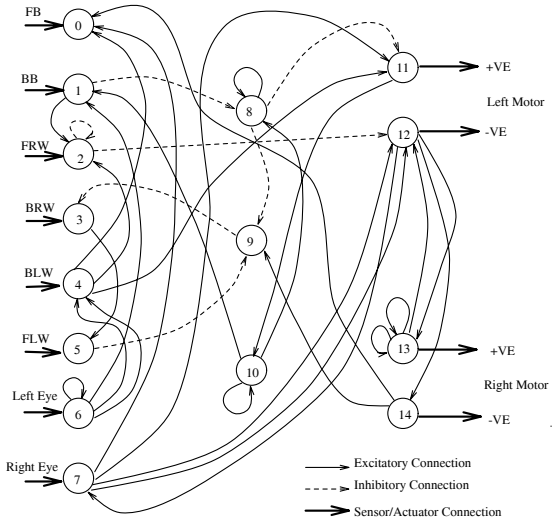

Figure 2: The network architecture for C1

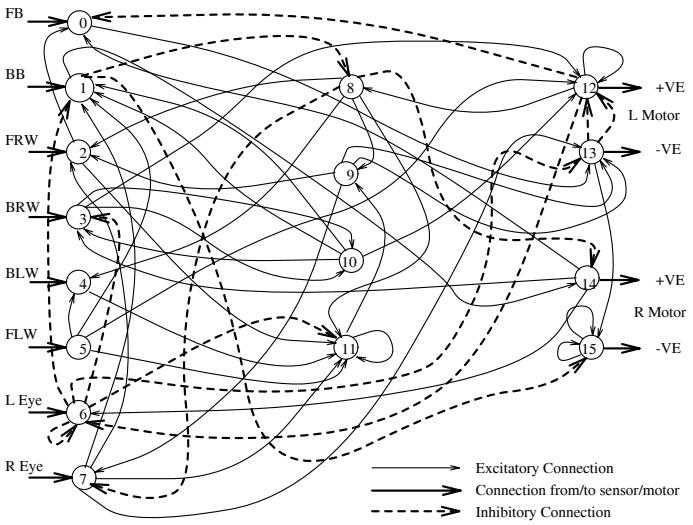

Figure 3: The network architecture for C2.

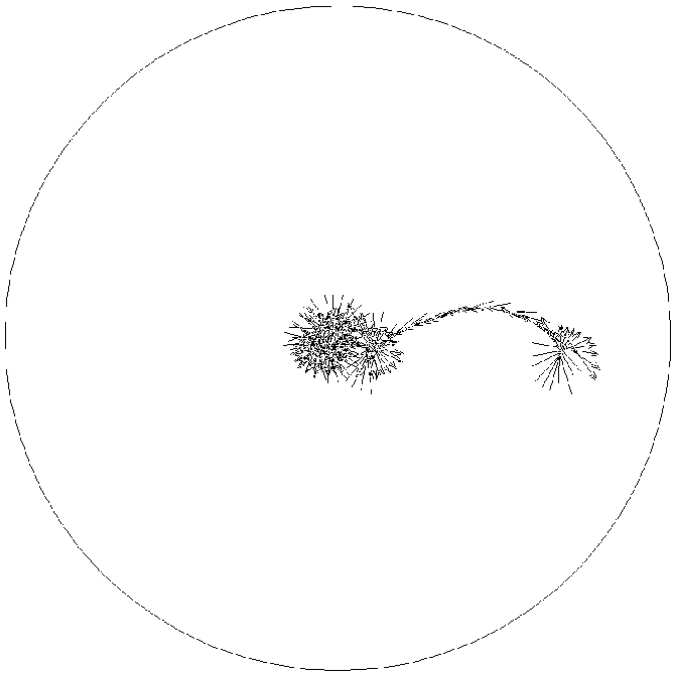

Figure 4: Typical behaviours of the robot controlled by evolved networks. The figure shows a top-down view of the circular arena; the robot's position is marked by an arrow: the length of the arrow-shaft represents the diameter of the robot; the direction of the arrow shows the orientation of the robot (not necessarily the same as the direction of travel).

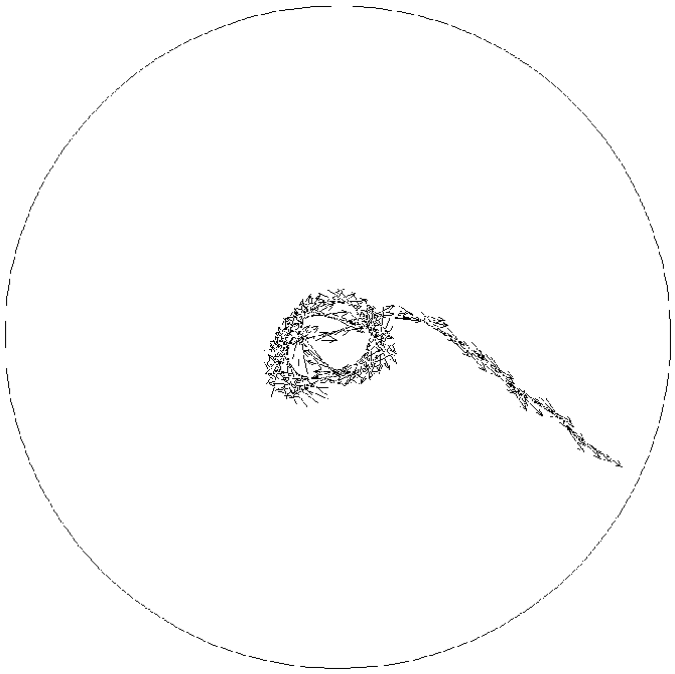

Figure 5: Behaviour of the C2 controller; note that C2 drives the robot backwards (i.e. in reverse).

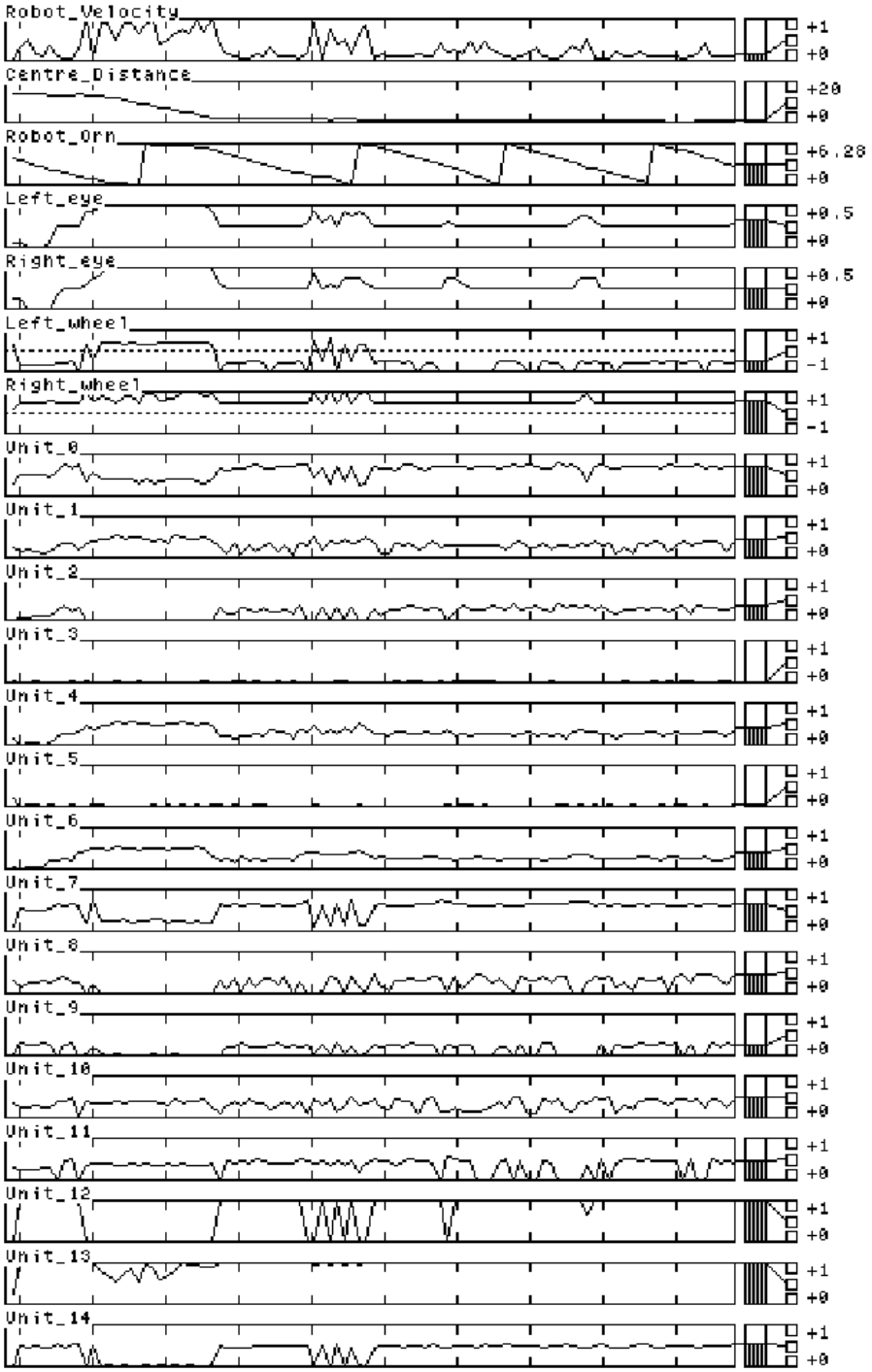

Figure 6: Time-plots of sensor, motor, and internal activation values for the C1 behaviour plotted in Figure 4. From top, graphs show: robot's velocity; distance of robot from centre of arena; visual input to left eye; visual input to right eye; output of motor for left wheel; output of motor for right wheel; excitatory output of the model neurons ("units") in the network.

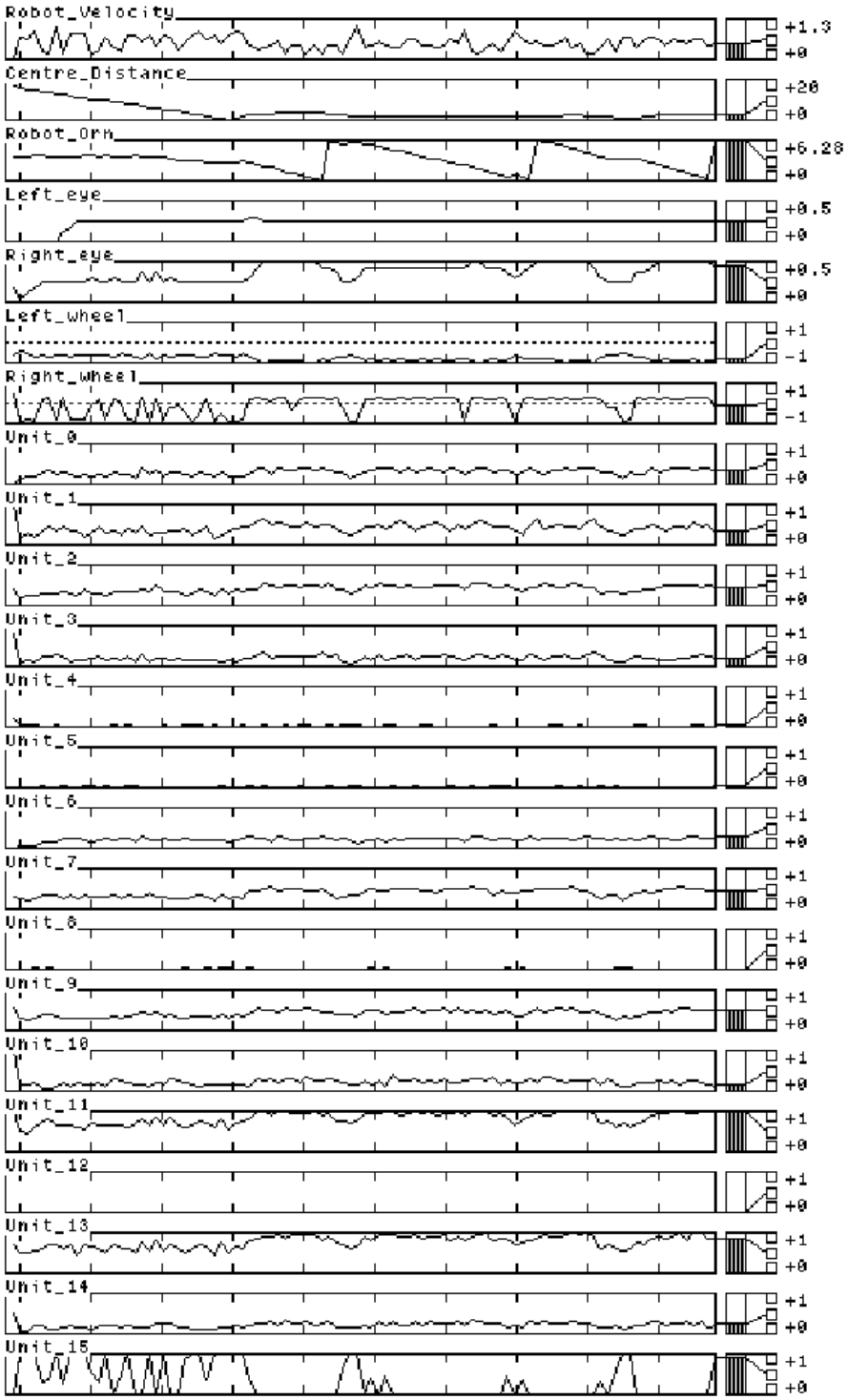

Figure 7: Time-plots of sensor, motor, and internal activation values for the C2 behaviour plotted in Figure 5. Display format as for Figure 6.

4 Varying the Noise

As was mentioned above, each robot is monitored for the same fixed amount of time, during which its fitness value is calculated. For further details of this process, see [3]. For the purposes of this discussion, it is sufficient to note that, if the robot spent all its time at the centre, it would receive a score of 100. But, because each robot's randomly chosen initial position is always some distance from the centre, this maximum score can never be reached: an optimum controller would score about 85 points. A robot which never moved would score less than one.

For the C1 and C2 controllers, after 100 generations of evolution, both networks managed an *average* score of around 65 (peak scores were nearer 80). These are the scores obtained with internal noise $n = 0.1$ injected in the model neurons. Both networks were then tested with different values of n, varying from $n = 0$ (i.e. no noise) to $n = 1.0$ (i.e. noise uniformly distributed in the range $[-1.0, 1.0]$). For each value of n, the network was evaluated 80 times, and the average score taken. Results from these tests are shown in Figure 8.

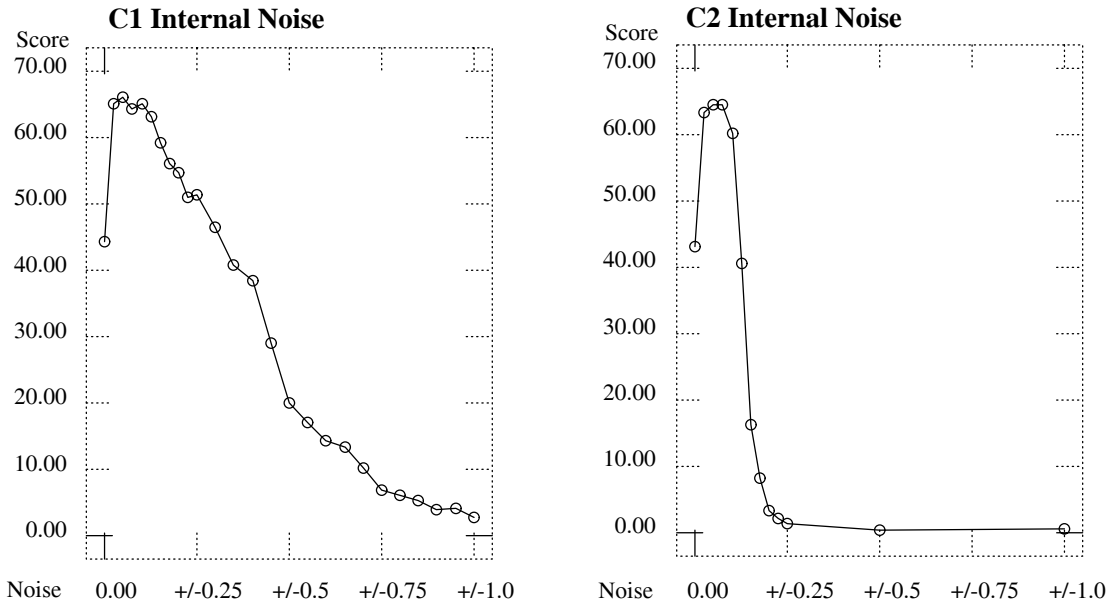

Figure 8: Results from testing with varying noise distributions. Left graph is for the C1 network; right graph is for the C2 network. Ordinate is average performance score over 80 trials. See text for discussion.

As can be seen from the graphs, both controllers show their peak performance close to $n = 0.1$, which was the value used in evolution. The performance of C1 degrades fairly gracefully as n increases, whereas C2 rapidly fails. Figure 9 gives a qualitative impression of C1's responses as noise is varied. However, a much more significant observation is the fact that both controllers exhibit a significant loss of performance when $n = 0$; this is discussed further below.

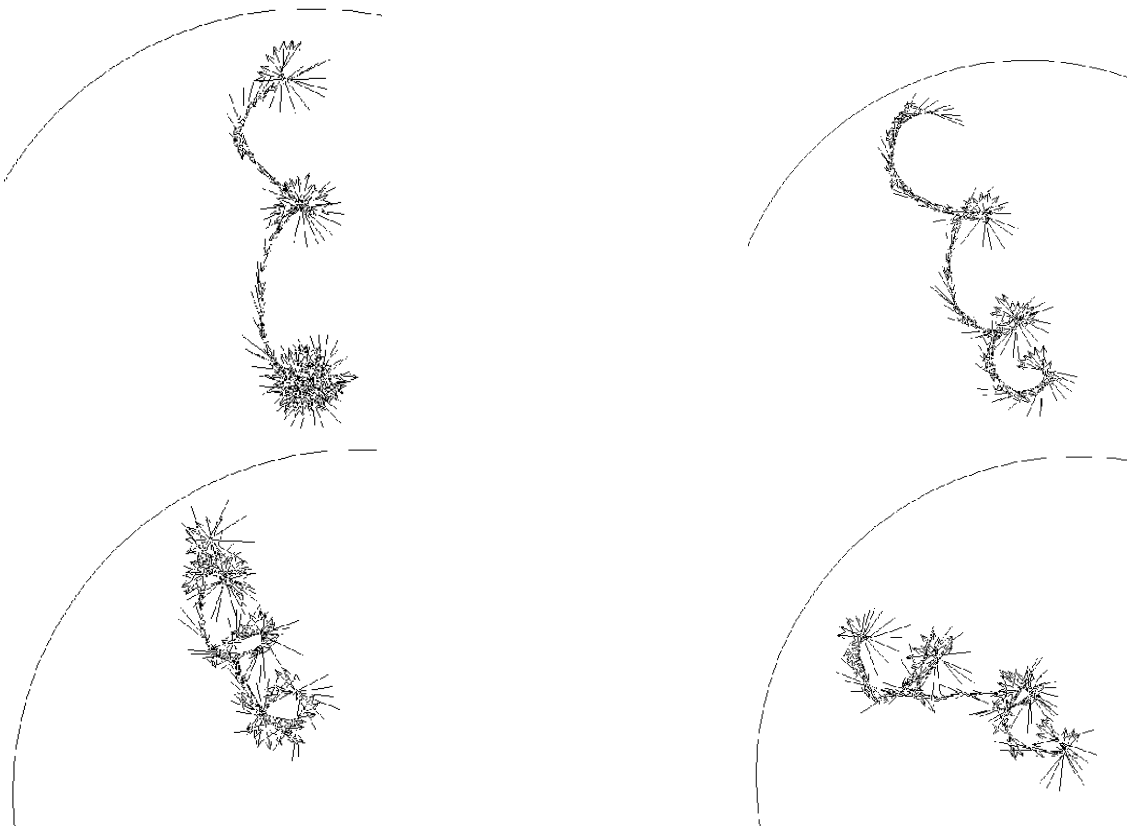

Figure 9: Typical behaviour resulting from using C1 with varying levels of internal noise. Display format as for Figure 4. At the top, on the left noise ± 0.2, on the right ± 0.4. Below, on left ± 0.6, on right ± 0.8.

4.1 Discussion

Further examination of the results indicates that the drop in performance when noise is eliminated is due to the recurrent dynamical nature of the networks: the recurrency implies that the network architectures contain feedback loops at a number of levels. That is, it is common to see a unit with connection(s) to itself, or two mutually excitatory units, or cycles of excitatory links incorporating several units. In such cases, low levels of internal noise may build up over time by a process of accumulation through feedback loops. However, because the noise distribution is centred on zero, it is also possible that these high levels of activity could then be driven downwards by injections of negative noise: therefore, when noise is injected into networks of the type we use, so-called "drunkard's walk" phenomena emerge; where activation values 'wander' between upper and lower bounds; and in certain feedback configurations, circuits with quasi-periodic oscillatory activity can be seen to evolve. For supporting theoretical analysis, see [10].

So, when the noise is eliminated, any parts of the neural circuitry which act as accumulators or oscillators in the presence of noise will be rendered ineffective, and the performance of the controllers is consequently impaired. In both the C1 and C2 net-

works, the drop in *average* performance was due to an increase in the number of near-zero scores: peak scores were still high, but under certain conditions the absence of noise allowed the activity in the network to fall to such an extent that the robot was rendered immobile. Put more formally, the noise helps the state trajectory of the controller system from becoming trapped on attractors which correspond to inactivity or unproductive behaviours. In this sense, it is realistic to describe the networks as *using* noise to produce useful behaviours.

This is a significant issue: if networks evolve to take advantage of internal noise, then it is important to ensure that the internal noise used in simulation (i.e. during evolution) closely matches the true noise levels that will be found when the evolved controller is put into use. In our current work, this is something of a non-issue because the real robot can be controlled by evolved networks simulated in the same manner as was employed in evolution, using the robot's on-board microcomputer. However, if our methods are used to develop control networks which will be realised in hardware, with each model neuron implemented as an electronic circuit, then it is essential that a fairly precise characterisation of the tolerances and internal noise distributions of the model neuron circuit should be incorporated in the evolution simulation.

5 Conclusion

Our work is motivated by concerns that prior network models may have been over-simplistic, and have not paid sufficient attention to the generation of adaptive behaviour. We have demonstrated that, using a neuron model with elementary dynamics, recurrent networks can exhibit rich dynamical activity that is not unduly hampered by noise, and can be used for evolving controller networks that generate adaptive behaviour. We have presented results which indicate that the networks use noise to avoid the effects 'unproductive' attractors can have on the state trajectory of the controller network. The evolved networks have a distinctive appearance, in that they do not resemble networks designed by humans. As far as we know, we are the only research group who have successfully employed truly incremental evolution in creating dynamic recurrent networks for the generation of adaptive behaviour. We expect that our techniques will, as time progresses, become standard practice.

References

[1] R. D. Beer and J. C. Gallagher. Evolving dynamical neural networks for adaptive behaviour. *Adaptive Behaviour*, 1(1):91–122, 1991.

[2] D. T. Cliff. Computational neuroethology: A provisional manifesto. In J.-A. Meyer and S. W. Wilson, editors, *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB90)*, pages 29–39, Cambridge MA, 1991. M.I.T. Press — Bradford Books. Also available as University of Sussex School of Cognitive and Computing Sciences Technical Report CSRP162.

[3] D. T. Cliff, P. Husbands, and I. Harvey. Analysis of evolved sensory motor controllers. Technical Report CSRP 264, University of Sussex School of Cognitive and Computing Sciences, 1992.

[4] 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, 1993. In Press. Also available as University of Sussex School of Cognitive and Computing Sciences Technical Report CSRP220.

[5] H. de Garis. Genetic programming: Building artificial nervous systems using genetically programmed neural network modules. In B. W. Porter and R. J. Mooney, editors, *Proceedings of the Seventh International Conference on Machine Learning*, pages 132–139. Morgan Kaufmann, 1990.

[6] 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.

[7] I. Harvey. The SAGA cross: the mechanics of crossover for variable-length genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 269–278. North-Holland, 1992. Also available as University of Sussex School of Cognitive and Computing Sciences Technical Report CSRP223.

[8] I. Harvey. Species adaptation genetic algorithms: A basis for a continuing SAGA. In F.J. Varela and P. Bourgine, editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life (ECAL91)*, pages 346–354. M.I.T. Press — Bradford Books, Cambridge MA, 1992. Also available as University of Sussex School of Cognitive and Computing Sciences Technical Report CSRP221.

[9] I. Harvey, P. Husbands, and D. 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)*. M.I.T. Press — Bradford Books, Cambridge MA, 1993. In Press. Also available as University of Sussex School of Cognitive and Computing Sciences Technical Report CSRP219.

[10] P. Husbands, I. Harvey, and D. T. Cliff. Analysing recurrent dynamical networks evolved for robot control, 1993. Submitted, IEE conference on artificial neural networks (ANN93), Brighton, 1993. Also available as University of Sussex School of Cognitive and Computing Sciences Technical Report CSRP265.