Evolution of a Circuit of Spiking Neurons for Phototaxis in a Braitenberg Vehicle

R. L. B. French **R. I. Damper**
Image, Speech and Intelligent Systems (ISIS) Research Group
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK.
{rlbf98r|rid}@ecs.soton.ac.uk

Abstract

Animal nervous systems have evolved to use spiking neurons but the 'artificial nervous systems' of animats typically are designed, not evolved, and use networks of formal, artificial neurons. We describe the evolution of circuits of spiking neurons for a robot, motivated by the desire to study links between neurophysiology and behaviour in artificial and (ultimately) natural animals. Spiking neurons have computational capabilities additional to those of artificial neurons based on activation functions. In particular, they should be better suited to processing temporal sequences. Thus, we describe early work aimed at evolution of neural circuitry which, when implanted in a Braitenberg type 2b vehicle, promotes phototaxis behaviour in the form of movement towards flashing lights of a particular frequency. The longer-term aim is to evolve natural taxis behaviours such as that observed in the cricket.

1 Introduction

Animal nervous systems use spiking neurons but the 'artificial nervous systems' (ANSs) of animats usually do not. The latter typically use a gross simplification in the form of artificial, or 'PDP-type', neurons based on activation functions, so that significant characteristics of real neurons are being ignored. It is likely that spikes evolved mainly as a means of regenerative signaling along the relatively long propagation paths of animal nervous systems (Levitan and Kaczmarek 1997). Because of active regenerative processes, the spike is propagated without loss of amplitude; hence, no information is carried by signal amplitude. Accordingly, detailed timing information for individual spikes, and relative timing between spikes, offers an extra dimension to the neural code (as does the stochastic aspect). Hence, spiking neurons ought to have additional computational capabilities relative to PDP-type neurons (Bugmann 1997; Maass and Ruf 1999).

In previous work (e.g., Damper, French, and Scutt 2000), we have used an autonomous robot to study the links between the 'neurophysiology' of the robot and its external behaviour in the belief that this can inform us about the links between neurophysiology and behaviour in real animals. This dictates that we use spiking neurons in the robot controller. Since real nervous systems are evolved rather than designed, we have more recently studied the evolution of spiking neuron circuitry (or ANSs) for our robot (French and Damper, submitted). A motivation for this work was to see if evolution could profit from the additional computational abilities of spiking neurons. However, the simple obstacle-avoidance task that we set the robot meant that there was no real scope for exploiting temporal-sequence processing capabilities. In this new work, we study a task specifically designed to promote sequence-discrimination behaviour. Specifically, we require the robot to move towards a flashing light of one frequency whilst ignoring a light of another frequency. This sort of discrimination is often exhibited in nature. For instance, female crickets move towards the 'song' of a male conspecific but ignore signals of similar (but different) acoustic structure. This is the phenomenon of *phonotaxis*, and it has been modelled in robot studies by Webb and Scutt (2000). Since our robot is not currently equipped with acoustic sensors, we have switched modalities to study the evolution of *phototaxis* behaviour.

The remainder of this paper proceeds as follows. In Section 2, we briefly review relevant work on spiking neuron models including the Hi-NOON simulator used here. Section 3 then describes the general scheme for evolution of an artificial nervous system (ANS) by genetic programming. Next, Section 4 describes the particular case of the evolution of a flashing-light frequency discriminator, which is then analysed in Section 5. Section 6 details the use of the evolved circuitry to implement phototaxis in a Braitenberg type 2b vehicle.

Section 7 describes plans for extending this work to the simulation of cricket phonotaxis and Section 8 concludes. A general description of the mathematical models used in Hi-NOON is included as Appendix A.

2 Spiking Neuron Models

Spiking neuron models have been implemented both as software (Raymond, Baxter, Buonomano, and Byrne 1992; Maass and Natschlager 1998) and hardware (Maris and Mahowald 1998; Rasche, Douglas, and Mahowald 1998). Generally speaking, such models operate at two different levels of detail: that of individual neurons and 'pools' of many neurons. For example, the model of Raymond et al. was designed to explore the contribution of individual neurons to the behaviour of a simulated nervous system, and the model of Maass and Natschlager was intended to look at the use of pools of neurons. In terms of application, Raymond et al. have investigated a possible circuit for reinforcement learning of head-waving behaviour in *Aplysia*, while Maass and Natschlager have constructed an associative memory of spiking neurons based upon a Hopfield network.

Modelling neurons with software contrasts with the hardware approach adopted by Maris and Mahowald and by Rasche et al. However, instead of building digital circuitry dedicated to modelling spiking neurons, these authors work with low-power analogue circuits. Maris and Mahowald have built a line-following robot, that uses a single analogue spiking neuron to control the steering. The work of Rasche et al., on the other hand, uses complex analogue spiking neurons (each have 25 parameters) to model neocortical pyramidal cells.

Although hardware implementations give the benefit of high-speed execution of the model, committing a neural model to silicon does not allow the model's algorithm to be modified afterwards (the analogue circuitry has to be changed to do this). Hence, the advantage of computer simulation is threefold: (1) Neural algorithms are not fixed and so can be modified as needed during development; (2) Data can be conveniently logged to mass storage from any point (any neural object) in the ANS, and (3) Keeping the neural model as a software construct enables rapid prototyping (even evolution) of circuitry.

In addition to our own work (French and Damper, submitted), Floreano and Mattiussi (2001) have also successfully evolved spiking neuron controllers for a mobile robot. The latter authors claim to have shown that evolved circuits of spiking neurons are superior to evolved circuits of sigmoidal activation function neurons, but this may be because the latter have fixed connection weights (all equal to 1).

Here, as previously, we have used our Hi-NOON software (Scutt and Damper 1992; Damper, French, and Scutt 2001) to simulate the evolved ANSs. The reader is referred to Appendix A for details of the mathematical models used. A brief overview of Hi-NOON's data structures is given as background here, to help understand the evolutionary process. Hi-NOON is a simulator for a Hierarchical Network of Object-Oriented Neurons. For computational efficiency, it attempts to model neuronal function at a relatively high level, using a simple state system to describe trans-membrane potential and its changes. As the name suggests, synapses, neurons and the spiking neuron network itself are all represented as dynamically-created objects (equipped with their own data members and member functions) within an object-oriented hierarchy (Coad and Yourdon 1991; Eliëns 1994). Critically, Hi-NOON was chosen for this work over other simulators because we need intimate knowledge of the simulator and full access to its source code to be able to manipulate ANSs during evolutionary computation. The neural network is held as a list of objects, where each such object corresponds to a single neuron and holds all the information about its state and references synapse objects.

The information held in a neuron object comprises:

- a set of parameters which defines the neuron;
- a set of data structures which defines the 'axon terminals' for the neuron, each of which is itself an object and has its own parameters;
- a set of methods (pointers to functions) which access and alter parameter values and so determine exactly how the neuron functions.

Neurons (together with their synapses) are in turn connected to a linked list of `node` objects as shown in Figure 1. The latter objects form the skeleton of a neural network and are used to manage the network during its dynamic creation, modification, and deletion. (In conventional neural network terminology, *node* and *neuron* are synonyms: Here, they are distinct objects so as to separate structure and function and allow each to be flexibly modified.) The head of the node list is referenced by a `list` object. This is used to maintain the nodes and to allow random access to the neurons via a table, which is used to dereference a postsynaptic target held as a data member of a synapse.

3 Evolving a Nervous System

Traditional genetic programming (GP) does not make a distinction between genotype and phenotype (Koza 1992), unlike genetic algorithms (Goldberg 1989). Even with GP, however, the distinction between genotype and phenotype can usefully be made. Such work is pursued by Banzhaf (1994) where the genotype is a bit-string and the phenotype is source code for languages such as C and FORTRAN (which is then compiled and

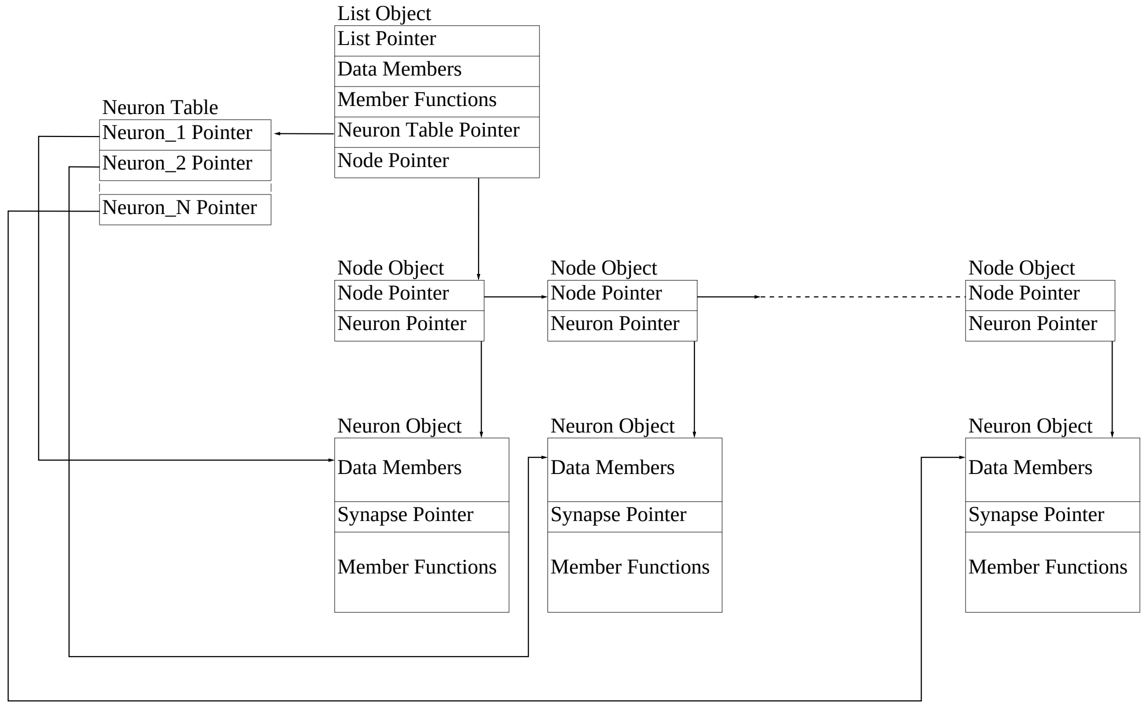

Figure 1: The data structure describing an ANS is a linked list of `node` objects. Each `node` object references a corresponding `neuron` object, which includes a synapse pointer referencing an array of `synapse` objects. The `node` objects are used to manage the ANS during its dynamic creation, modification, and deletion. The head of the linked list is referenced by a `list` object, corresponding to the ANS itself.

evaluated as native code). This distinction between genotype and phenotype allows neutral variants (i.e., different genotypes mapping to the same phenotype) which can improve the search for fit individuals, but has the disadvantage that it may lead to syntactically-invalid offspring (a point taken up in Section 3.1.2). However, such errors from Banzahf's bit-string to source code translation can be dealt with using a correction mechanism. Citing Banzhaf (1994, p. 325):

> "The main ingredients to this mapping are, first a coding of pieces of the bit-string into the nodes of a program tree ... and, second, a correction mechanism that is able to check statements and to transform them into the nearest correct statement if an error was detected."

Thus the correction mechanism ensures syntactically-valid source code.

So, at the outset, there would appear to be two rather different options available for evolving a nervous system: We can either make a strong distinction between genotype and phenotype, or not. In the first case, a population of genotypes could be 'grown' into a phenotype (i.e., an ANS) by a relatively complex process of neurogenesis (e.g., Michel 1997; Mautner and Belew 1999). In the second case, which is the approach adopted here, genotypes are sought which are as close as possible to the required phenotypes. A solution is to use hierarchical objects as the genotype, which are directly manipulated during evolution. Since these are executable in Hi-NOON, the genotype is identical to the phenotype, remaining close to traditional GP.

There are a few reasons for adopting this alternative. First, to save unnecessary effort, we wish to (re)use as much of the existing Hi-NOON code as possible. Second, because the genotype/phenotype distinction is weak (i.e., there is no complex and error-prone translation from genotype to phenotype that needs subsequent correction), the problem of non-viable offspring can be avoided. Finally, direct manipulation of objects avoids a (possibly) protracted growth phase, and so simplifies fitness evaluation during evolution.

3.1 Mutation, crossover and selection

In what follows, let us assume the prior existence of an initial population of candidate artificial nervous systems (ANSs), each of which is held as a list of objects.

3.1.1 Mutation

Mutation is used to add components (i.e., neural subcircuits) to an ANS and/or to delete them. Mutation operators simply create and destroy neurons and synapses. Created objects are given values for their data members which are randomly distributed within a predefined range. The operators are:

- `add_neuron`: May call `add_synapse` for connection to post-synaptic targets.
- `delete_neuron`: Calls delete synapse for a particular neuron as well as any pre-synaptic cell.
- `add_synapse` Called by `add_neuron` for connection to post-synaptic targets. Synapses may be normal, noisy or habituating types.
- `delete_synapse` Also starts a recursive call for deletion of any synapse-on-synapse connections and schedules isolated neurons for deletion.

Mutation operators are randomly selected.

Because the nervous system representation is manipulated directly during the evolutionary process, there is no limit to the size of genotype in this application. Genetic algorithms usually employ a fixed-length genotype (Koza 1992, p. 18) although this is by no means always so (e.g., Goldberg and Deb 1991). This is an additional reason why this work may be viewed as being closer to the paradigm of genetic programming than genetic algorithms, with Hi-NOON viewed as a virtual machine.

3.1.2 Crossover

Defining an appropriate crossover operator confronts a problem common to a range of applications of GP and which was mentioned above. To quote Koza: "Crossover must be performed in a structure-preserving way so as to preserve the syntactic validity of all offspring" (Koza 1994, p. 85). By a careful choice of a genotype as a hierarchical data structure defining a candidate ANS, a way has been found to ensure that any child resulting from crossover applied to its parents is always syntactically valid for execution on Hi-NOON.

Crossover is single-point. Two parents are chosen as follows. Let the first parent have neurons labeled $1, \ldots, N_1$ and let the second parent have neurons labeled $1, \ldots, N_2$. The crossover point is then randomly chosen after position n (in the linked list in Fig. 1), where $n \in \{2, \ldots, (\min[N_1, N_2] - 1)\}$. Two children are then produced, as demonstrated in Figure 2, and any of their synapses that no longer have valid postsynaptic targets are pruned. In the interests of computational efficiency, this method has been chosen in preference to any attempt at analysing the two parents looking for an 'ideal' crossover point. This is similar in spirit to

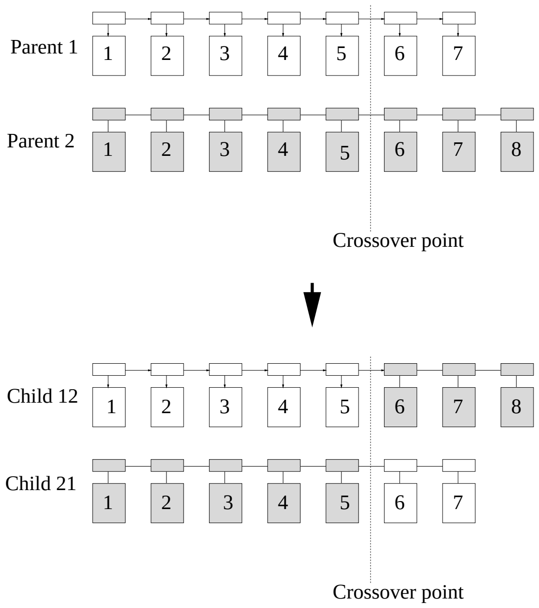

Figure 2: An example of crossover after neuron position 5 applied to the data structures of two parent ANSs.

the crossover operation performed on animat nervous systems by Sims (1994a) in his artificial life research. Here, however, only one crossover point is used whereas Sims uses one or more. Also, he reassigns synapses that no longer have a postsynaptic target and we do not.

3.1.3 Selection

The selection operator used here is the stochastic universal sampling (SUS) method (Baker 1987; Whitley 1993) with elitism. SUS was chosen because it is computationally efficient and maintains diversity among the population.

3.2 Initial population

Initially, the evolutionary process must be seeded with a population of randomly-generated individuals of appropriate size, P. To balance the need for a reasonably large population (to maintain diversity) against the practical necessity for reasonably short run times, a value of $P = 1000$ has been chosen for this work. Each member of the initial population is created by five applications of the mutation operators described in Section 3.1.1.

4 Evolving a Frequency Discriminator

In our work to date, circuits of model spiking neurons have not been evolved to process a sensory signal with a temporal component to it. In this section, such work is addressed by evolving a frequency discriminator. In the longer term, it is intended to evolve circuitry which can discriminate between signals with different

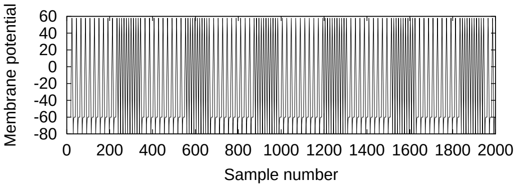

Figure 3: Neuron activity alternating between two firing rates.

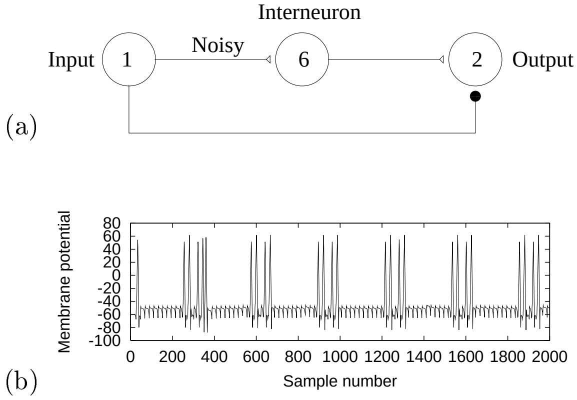

Figure 4: (a) A nervous system that was evolved to select the high firing rate of an input signal (Fig. 3) and (b) its output. Redundant components have been removed for clarity.

temporal structures.

Figure 3 shows the activity of a neuron that alternates between high and low firing rates. This represents the output of a sensory neuron that is being depolarised by a stimulus from an environment and forms the input to our frequency discriminator.

The fitness function of a candidate ANS for this evolutionary task simply increments a counter every time a particular input firing rate (e.g., ‘low’) coincides with firing of a designated output neuron. The fitness counter is decremented for an incorrect output response. Hence, this fitness function attempts to evolve a circuit of spiking neurons that can recognise a particular (and very simple) signal that has a temporal component.

A population of 1000 individuals was used running for 200 generations. The evolutionary process was repeated 10 times for the low-frequency discrimination task and 10 times for the high-frequency discrimination task. This resulted in only 3 of the high-frequency discrimination and 5 of the low-frequency discrimination candidates being viable. This may be an indicator of a very large search space for this problem.

Figure 4 shows a circuit evolved with a fitness function configured to select the high-frequency component of Fig. 3. In the same way, Figure 5 was evolved with a fitness function configured to select the low-frequency component of Fig. 3.

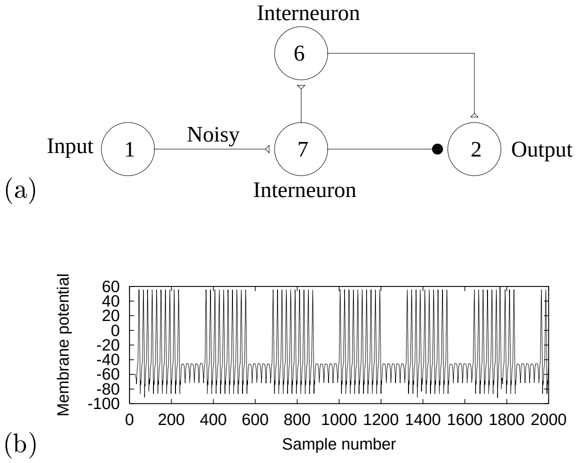

Figure 5: (a) A nervous system that was evolved to select the low firing rate of an input signal (Fig. 3) and (b) its output. Redundant components have been removed for clarity.

5 Analysis of the Evolved Circuits

In his artificial life research, Sims (1994b, p. 34) states “ . . . it can be difficult to analyze exactly how a control system such as this works . . . a primary benefit of using artificial evolution is that understanding these representations is not necessary”. We believe this view is unnecessarily pessimistic; in our opinion, it is well worth trying to understand the evolved circuits, as we now demonstrate.

In the high-frequency discriminator of Fig. 4, the spikes of neurons 1 and 6 are almost in-phase during the low-frequency input. This is because the delay line effect of neuron 6 has very little influence during low-frequency input. Hence, their excitatory and inhibitory post-synaptic potentials cancel in the membrane of neuron 2. However, during high-frequency input this synchronisation is lost as the delay introduced by neuron 2 effectively stores a spike whose excitatory post-synaptic potentials (EPSPs) in neuron 2 cannot by cancelled by inhibitory post-synaptic potentials (IPSPs) from neuron 1. Hence, these excitatory potentials fire neuron 2 during high-frequency input.

The circuit of Fig. 5 suppresses a high-frequency output by cancellation of EPSPs and IPSPs in neuron 2. However, the mechanism used to detect the high-frequency appears to be based upon the noisy nature of the excitatory synapse between neurons 1 and 7: Editing the evolved circuit to remove this noise destroys the ability to ignore the high-frequency signal. This noisy synapse forms a frequency inverter—the inter-spike interval of neuron 7 increases by a small amount during the high-frequency activity of neuron 1. This is counter-intuitive because we expect an increase in parent neuron firing rate to result in an increase of synapse firing also. However, no bugs have been found in the program that

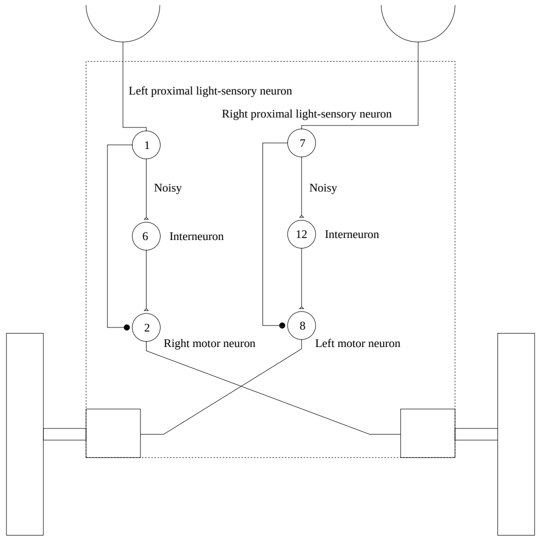

Figure 6: A Braitenberg type 2b vehicle architecture with spiking neurons that seeks high-frequency signals.

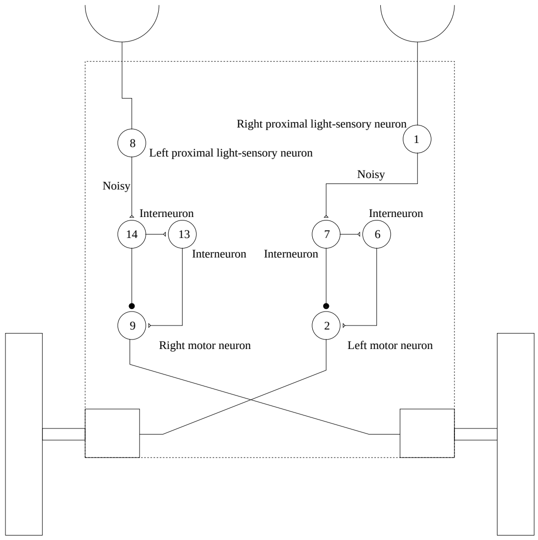

Figure 7: A Braitenberg type 2b vehicle architecture with spiking neurons that seeks low-frequency signals.

could be exploited by the evolutionary process in this way. As in Fig. 4, a delay line formed from neuron 6 (with respect to neuron 7) results in EPSPs and IPSPs cancelling in the membrane of neuron 2 during low-frequency firing of neuron 7 (during high-frequency firing of neuron 1). During low-frequency input (causing a higher firing rate in neuron 7 with respect to its rate during high-frequency input), potentials do not cancel in the membrane of neuron 2. Thus, its membrane reaches threshold and fires, giving an output.

6 Phototaxis in a Braitenberg Vehicle

Now that we have evolved neural circuits that can discriminate between flashing lights of different frequency, they were incorporated in a Braitenberg type 2b vehicle (Braitenberg 1984, p. 7). This should result in a vehicle that can seek out the source of a stimulus, e.g., a light source flashing at a particular rate, so exhibiting phototaxis.

An ANS describing a type 2b vehicle architecture was created using two of the networks in Fig. 4 to connect contralateral *proximal* sensory-motor pairs (i.e., the sensors respond to changes in light level). This is shown in Figure 6. The same was done using two of the networks in Fig. 5 to produce the vehicle with ANS shown in Fig. 7. Proximal light sensory neurons were selected to reduce the risk of spurious sensory input. This is because ordinary light-sensor neuron firing is proportional to light intensity.

No changes to the evolved nervous system circuitry were made, with the exception of adding sensory and motor neurons in place of input and output neurons. Experiments were carried out to test the expected

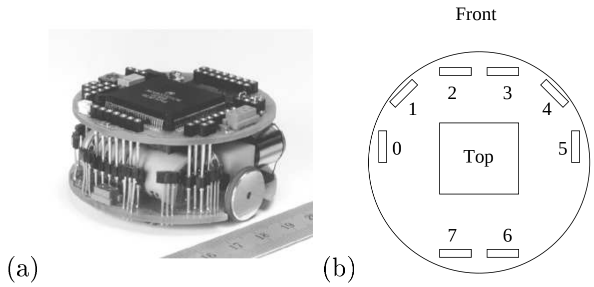

Figure 8: (a) The Khepera robot. (b) Sensor positions on Khepera.

phototaxis behaviour.

The experimental setup was as follows. A Khepera robot (Figure 8(a)) was placed in a test environment facing the mid-point between two 2.5 V filament lamps, 15 cm away. The lamps were positioned 15 cm apart. An electronic circuit switched the lamps at approximately 1 Hz and 10 Hz respectively. (These frequencies were selected to match the spiking input used to evolve the circuits.) A toggle-switch allowed the frequency selection for the lamps to be swapped over (i.e., the low-frequency signal can appear on the left lamp, with the high frequency signal on the right or vice versa). The nervous system models executed on a 80486 Linux PC.

During tests aimed at setting up the intensity of the lamps, it became clear that Khepera's left-hand light sensor (Khepera position 1, see Figure 8(b)) was not as sensitive as that on the right (Khepera position 4). Also, ambient light influenced the robot's phototaxis performance: Bright background light disrupted the ability to discriminate between high- and low-frequency

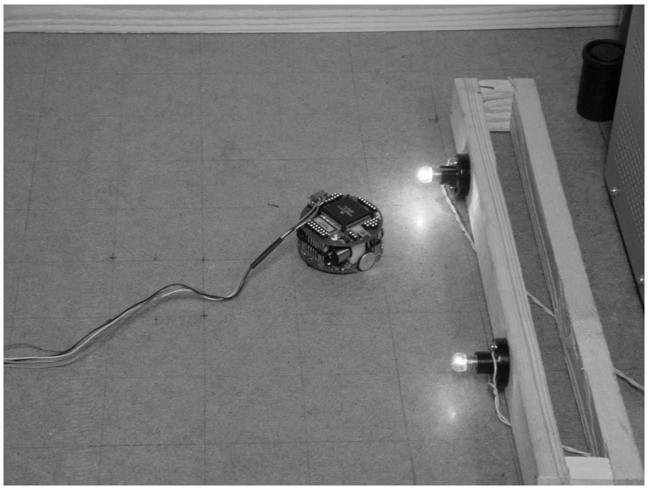

Figure 9: Khepera in the test environment used for phototaxis experiments. The two lamps have different flash frequencies and are shown here at different stages of illumination.

signals, causing the robot to come to rest between the two lamps. Hence, final runs took place in low levels of lighting—low enough not to disrupt phototaxis but sufficient to film the vehicle's activity with a digital camcorder. The inability to cope robustly with ambient light is probably because the discrimination circuits of Figs. 4 and 5 were evolved in isolation, without the input noise generated by the Khepera simulator sensor models.

The experiments started with a Braitenberg type 2b vehicle that favoured the high-frequency signal (Fig. 6). Twenty trials were carried out to observe the robot's behaviour with both settings of the toggle switch mentioned above. For the first 10 trials the high-frequency signal was on the robot's left; For the second group of 10 trials, the high-frequency signal was moved to the robot's right. This was then repeated with a vehicle that favoured the low-frequency signal (Fig. 7).

Figure 9 shows the vehicle (Khepera) in the test environment. It was noticed that the response of the high-frequency circuit was not as vigorous as its low-frequency counterpart, resulting in slower movement. (This is undoubtedly because the period of high frequency spiking is shorter in the input signal, see Fig. 3.) In all trials, the vehicles performed their tasks without error, halting in front of the appropriate lamp. For the high-frequency case, the vehicle took approximately 80-90 seconds to reach the lamp and halt. For the low-frequency case, the corresponding time was 25-30 seconds.

7 Towards the Evolution of Phonotaxis

In the Introduction, we mentioned phonotaxis in the cricket, whereby a female insect seeks the calling song of male conspecifics in a noisy environment. This has been extensively studied in the animat research of Webb and colleagues (Lund, Webb, and Hallam 1997; Webb and Scutt 2000; Reeve and Webb 2001). However, throughout their work, the circuitry underlying the behaviour of the animats has been manually designed. This prompts the question: "Can we evolve an ANS that allows a robot to perform phonotaxis behaviour?" This is interesting because an ANS that is generated through evolutionary computation may operate using principles that had previously not been considered for manual designs.

Perhaps the most basic of song characteristics is carrier frequency. For example, the cricket *Gryllus bimaculatus* has a calling song with a carrier frequency of some 4.7 kHz. The song is generated by stridulation—a very common mechanism for sound production in arthropods. The complete song consists of chirps with three syllables, each of about 20 ms in duration and separated by 20 ms silence. Each chirp is separated with a 200 ms break (Lund, Webb, and Hallam 1997).

The early work presented here on phototaxis (which relies on the temporal processing capabilities of evolved circuits of spiking neurons) is intended as an early stage in the evolution of circuitry to drive cricket phonotaxis. Clearly, the cricket song has more complex temporal structure than mere switching between low and high frequencies. Nonetheless, a useful start has been made.

8 Conclusions and Discussion

We have described early work to evolve spiking neural circuits that can discriminate between high- and low-frequency signals. Light-seeking Braitenberg vehicles were constructed that used these circuits as the basis of phototaxis. Although the Braitenberg vehicles were unable to deal with strong ambient light, this is probably because their circuitry was evolved in isolation (for computational convenience) and not onboard a vehicle situated in a real or simulated environment. We intend to extend this work to simulate phonotaxis in the cricket, where the temporal patterning of the relevant signals is considerably more complex.

A motivation for this work was to see if evolution could profit from the additional computational abilities of spiking neurons. At this stage, the question remains open. It seems highly likely, for instance, that a robot equipped with non-spiking recurrent neural networks of the kind evolved by Yamauchi and Beer (1994) would be able to behave appropriately in the simple task described here. Perhaps, if there really are significant differences in computational abilities between spiking and non-spiking neurons, they will only be revealed when we are able to achieve much more complex patterns of behaviour.

Appendix A: Neurons and Synapses

In this section, we present more detailed descriptions of neurons and synapses within Hi-NOON. Since Hi-NOON is intended for (amongst other things) applications in situated robotics studies, there is provision for sensory and motor neurons which connect to the environment, as well as for more prosaic 'basic' (information processing) neurons.

A.1 Neurons

The 'basic' neuron type has the state system functionality which is subsequently embedded in all derivatives, such as the sensory and motor cells.

A.1.1 Basic neurons

Updating equations for the membrane potential (MP – in millivolts) for this neuron type are:

state A: $MP(t+1) = MP(t) - \tau + S(t)$
state B: $MP(t+1) = MP(t) - \alpha + S(t)$
state C: $MP(t+1) = h + S(t)$
state D: $MP(t+1) = MP(t) - \mu + S(t)$
state E: $MP(t+1) = l + S(t)$
state F: $MP(t+1) = MP(t) + \frac{BaseMP - MP(t)}{\eta} + S(t)$

where:

$$S(t) = \sum_i w_i \kappa \left(MP_i(t) - BaseMP_i\right)$$

is the synaptic potential (`SynPot`), i is a counter which counts over active pre-synaptic cells, w_i is the synaptic weight from a pre-synaptic neuron, τ is the neuron time constant, $\eta = 1.5$ is the post-undershoot increment rate, $\mu = 25$ is the post-action potential peak-MP decrement, $\kappa = 1/450$ is a heuristically-set constant, $\alpha = 20$ is the post-threshold attack increment, $h = 45$ is the post-threshold maximum MP, and $l = -69$ is the pre-undershoot minimum MP.

Certain of the above parameters (e.g. τ, η) are time-dependent and have been set empirically to suit a range of processor speeds and implementations. However, they may be inappropriate in some circumstances (as when implementing a real-time robotic system using a fast processor).

A.1.2 Sensory and motor neurons

These neuron types are important in the specific case of a robotic system which needs input and output from/to its environment. Since, in this paper, we are principally concerned with more general principles, we omit details of these neuron types here, except to say that the spiking output of the motor neurons was smoothed with a very simple (first-order) low-pass filter before being fed to the motor drives. Hence, the robot moves in a slightly jerky fashion.

A.2 Synapses

The basic synapse (which is noise free) has functionality which is subsequently embedded in all derivatives such as the habituating, sensitising and conditioning types used in our robot work (see below). These allow us to implement a simple, biologically-based form of learning.

$$w(t) = \begin{cases} w(t) - \beta & \text{if } w(t) > w_{\text{base}} \\ w(t) + \beta & \text{if } w(t) \leq w_{\text{base}} \\ w_{\max} & \text{if } w(t) > w_{\max} \\ w_{\min} & \text{if } w(t) < w_{\min} \\ w_{\min} & \text{otherwise} \end{cases}$$

where β is the MP recovery parameter and w_{base} is the base weight (typically 0). These are individually set (together with $w_{\min}$ and $w_{\max}$, typically ± 16) for each neuron.

A.2.1 Noise-free synapse

$$\texttt{fired}(t) = \begin{cases} \text{TRUE} & \text{if state B, C, D} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

A.2.2 Noisy synapse

$$\texttt{fired}(t) = \begin{cases} \text{TRUE} & \text{if } \texttt{cond1} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

where `cond1` is state B, C, D, as for the noise-free synapse, ANDed with:

$$\frac{MP_p - \theta_p}{h - \theta_p} \times 100 \geq \text{rand mod100}$$

and p denotes a parent (pre-synaptic) neuron.

A.2.3 Habituating type

$$w(t+1) = \begin{cases} w(t) - d & \text{if state C} \\ w(t) & \text{otherwise} \end{cases}$$

where d is a constant decrement (typically ~ 1).

A.2.4 Sensitising type

$$w(t+1)_{\text{targ}} = \begin{cases} w(t)_{\text{targ}} + w(t)_{\text{sos}} & \text{if } \texttt{cond2} \\ w(t)_{\text{targ}} & \text{otherwise} \end{cases}$$

where `cond2` is $\texttt{fired}_{\text{targ}} \wedge \texttt{fired}_{\text{sos}}$, 'targ' denotes the target synapse (to be sensitised) and 'sos' denotes the synapse-on-synapse influence.

A.2.5 Conditioning type

$$w(t+1)_{\text{targ}} = \begin{cases} w(t)_{\text{targ}} + kw(t)_{\text{sos}} & \text{if } \texttt{cond2} \\ w(t)_{\text{targ}} & \text{otherwise} \end{cases}$$

where:

$$k = \frac{nT}{\psi} \exp\left(\frac{-nT}{\varsigma}\right)$$

and nT is a count of sample periods initiated by encountering state C for the target neuron, ψ (= 250) is an empirically-set scaling factor and ς (= 500) is a constant chosen to maximise the effect of conditioning when the conditioning stimulus precedes the unconditioned stimulus by 0.5 s.

A.2.6 Autonomous decay of synapses

To prevent synaptic weights growing without limit, `Weight` is bounded during simulation. This models the finite size of stores of neuro-transmitter in the synaptic terminals of real biological neurons. Also, according to Sutton and Barto (1981, p. 161):

> "If it is assumed that synaptic strength slowly decays in the absence of a reinforcement signal, then a bound on weight size is imposed that is a function of reinforcement level and the decay rate. ... In system theoretic terms, the adaptive element has *definite memory*: it cannot remember anything that occurred arbitrarily far in the past."

Sutton and Barto call this *autonomous decay*. This decay mechanism is also implemented here (via `Recovery`).

References

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In J. J. Grenfenstett (Ed.), *Genetic Algorithms and their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 14–21. Hillsdale, NJ: Lawrence Erlbaum Associates.

Banzhaf, W. (1994). Genotype-phenotype mapping and neutral variation – a case study in genetic programming. In Y. Davidor, H.-P. Schwefel, and R. Männer (Eds.), *Proceedings of Parallel Problem Solving from Nature, PPSN III*, pp. 322–332. Berlin, Germany: Springer.

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology.* Cambridge, MA: MIT Press.

Bugmann, G. (1997). Biologically plausible neural computation. *BioSystems 40*(1–2), 11–19.

Coad, P. and E. Yourdon (1991). *Object Oriented Analysis* (Second ed.). Englewood Cliffs, NJ: Prentice-Hall.

Damper, R. I., R. L. B. French, and T. W. Scutt (2000). ARBIB: an autonomous robot based on inspirations from biology. *Robotics and Autonomous Systems 31*(4), 247–274.

Damper, R. I., R. L. B. French, and T. W. Scutt (2001). The Hi-NOON neural simulator and its applications. *Microelectronics Reliability 41*(12), 2051–2065.

Eliëns, A. (1994). *Principles of Object-Oriented Software Development.* Wokingham, UK: Addison-Wesley.

Floreano, D. and C. Mattiussi (2001). Evolution of spiking neural controllers for autonomous vision-based robots. In T. Gomi (Ed.), *Evolutionary Robotics IV.* Berlin: Springer-Verlag.

French, R. L. B. and R. I. Damper (2001). Evolving a nervous system of spiking neurons for a behaving robot. In *Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2001*, San Francisco, CA, pp. 1099–1106. Morgan Kaufmann.

French, R. L. B. and R. I. Damper (submitted). Evolution of biologically-inspired nervous systems for a mobile robot. *IEEE Transactions on Evolutionary Computation*.

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

Goldberg, D. E. and K. Deb (1991). Don't worry, be messy. In R. K. Belew and L. B. Booker (Eds.), *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 24–30. La Jolla, CA: Morgan Kaufmann.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: MIT Press/Bradford Books.

Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: MIT Press/Bradford Books. (Pagination refers to 1998 second printing).

Levitan, I. B. and L. K. Kaczmarek (1997). *The Neuron: Cell and Molecular Biology.* New York, NY: Oxford University Press.

Lund, H. H., B. Webb, and J. Hallam (1997). A robot attracted to the cricket species *Gryllus bimaculatus*. In P. Husbands and I. Harvey (Eds.), *Proceedings of Fourth European Conference on Artificial Life, ECAL'97, Brighton, UK*, pp. 246–255. Cambridge, MA: MIT Press/Bradford Books.

Maass, W. and T. Natschlager (1998). Associative memory with networks of spiking neurons in temporal coding. In L. S. Smith and A. Hamilton (Eds.), *Neuromorphic Systems, Engineering Silicon from Neurobiology*, Volume 10, pp. 21–32. Singapore: World Scientific.

Maass, W. and B. Ruf (1999). On computation with pulses. *Information and Computation 148*(2), 202–218.

Maris, M. and M. Mahowald (1998). Neuromorphic sensory-motor mobile robot controller with preattention mechanism. In L. S. Smith and A. Hamilton (Eds.), *Neuromorphic Systems, Engineering Silicon from Neurobiology*, Volume 10, pp. 126–137. Singapore: World Scientific.

Mautner, C. and R. K. Belew (1999). Coupling morphology and control in a simulated robot. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pp. 1350–1357.

Michel, O. (1997). Dynamical genomic network applied to artificial neurogenesis. *Control and Cybernetics 26*(3), 511–531.

Rasche, C., R. J. Douglas, and M. Mahowald (1998). Characterization of a silicon pyramidal neuron. In L. S. Smith and A. Hamilton (Eds.), *Neuromorphic Systems, Engineering Silicon from Neurobiology*, Volume 10, pp. 169–177. Singapore: World Scientific.

Raymond, J. L., D. A. Baxter, D. V. Buonomano, and J. H. Byrne (1992). A learning based on empirically-derived activity-dependent neuromodulation supports operant conditioning in a small neural network. *Neural Networks 5*, 789–803.

Reeve, R. and B. Webb (2001). Neural network control of a cricket robot. In U. Nehmzow and C. Melhuish (Eds.), *Third British Conference on Autonomous Mobile Robotics and Autonomous Systems: TIMR 01—Towards Intelligent Mobile Robots*, Manchester, UK. No pagination.

Scutt, T. W. and R. I. Damper (1992). Object-oriented modelling of small neuronal systems. *Artificial Intelligence and Simulation of Behaviour Quarterly 80*, 24–33.

Sims, K. (1994a). Evolving virtual creatures. In A. Glassner (Ed.), *SIGGRAPH 94 Conference Proceedings*, pp. 15–22. New York, NY: ACM Press.

Sims, K. (1994b). Evolving 3D morphology and behavior by competition. In R. A. Brooks and P. Maes (Eds.), *Artificial Life IV*, pp. 28–39. Cambridge, MA: MIT Press.

Sutton, R. S. and A. G. Barto (1981). Towards a modern theory of adaptive networks: Expectation and prediction. *Psychological Review 88*, 135–170.

Webb, B. and T. Scutt (2000). A simple latency-dependent spiking-neuron model of cricket phonotaxis. *Biological Cybernetics 82*(3), 247–269.

Whitley, D. (1993). A genetic algorithm tutorial. *Statistics and Computing 2*(4), 65–85.

Yamauchi, B. M. and R. D. Beer (1994). Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior 2*(3), 219–246.