Temporally adaptive networks: Analysis of GasNet robot controllers

Tom Smith[*1], **Phil Husbands**[2] **and Michael O'Shea**[1]
Centre for Computational Neuroscience and Robotics (CCNR)
[1]School of Biological Sciences, [2]School of Cognitive and Computing Sciences
University of Sussex, Brighton, UK
[*] toms@cogs.susx.ac.uk

A good performance, like a human life, is a temporal affair: a process in time.
- Mortimer J. Adler

Abstract

There are immense problems in developing artificial nervous systems for autonomous machines operating in non-trivial environments. In particular, no principled methodology is in place to decide between solution classes and representations, and between methods by which solutions might be developed using hand-design or search techniques. In this paper we apply the techniques of dynamical systems theory to the analysis of successfully evolved robot control systems, in order to identify useful properties of the underlying control architecture. We investigate the suitability of two different neural network classes for a robotic visual discrimination task, through analysis of both successful controller behaviour and continued evolution of successful solutions in environments with modified characteristics. We argue that the temporally adaptable properties of the GasNet class identified through dynamical systems analysis, and found to be useful in order to re-evolve in modified environments, are crucial to the evolution of successful controllers for the original environment.

1 Introduction

Identification of control system classes capable of generating adaptive behaviour over time is a black art. Many practitioners rely on systems that have "always worked in the past", others may use trial-and-error until success, but carry out no subsequent analysis of why that particular system actually worked. A major problem with such approaches is that it is extremely difficult to develop a more general understanding of the properties necessary for generating adaptive behaviour. In particular, is a particular solution class appropriate for a particular problem? Addressing this issue is crucial if we are to successfully apply techniques such as evolutionary computation to more complex adaptive behaviour problems than at present.

In this paper we develop an approach based on analysis of successfully evolved solutions. This allows us to identify properties of network classes that are potentially useful over a wider class of problems than the original task. We then develop a methodology for testing these properties, through analysis of the evolved solutions in modified environments.

We overview two classes of neural network, the "GasNet" and "NoGas", used as controllers in a visual shape discrimination problem, and give evidence that the GasNet class is more amenable to evolutionary search than the NoGas class. We then use the techniques of dynamical systems analysis to identify possible reasons for this increased evolutionary speed, and frame a number of hypotheses for the suitability of the GasNet class to robot control. In particular, we show how the properties of gas diffusion can be used to filter out sensor input noise, produce simple pattern generation networks, and switch networks from one stable state to another. We hypothesise that these properties lead to GasNet solution spaces in which it is easier to find good controllers than the corresponding NoGas solution spaces.

We go on to compare the operation of two controllers, one GasNet solution and one NoGas solution, which utilise the same visual shape discrimination strategy. We argue that the GasNet controller is easier to tune to the particular characteristics of the environment than the functionally equivalent NoGas controller, and find evidence to support such an argument through re-evolution of the functionally equivalent controllers in modified environments. We then extend the re-evolution analysis to a larger sample of previously evolved GasNet and NoGas controllers, showing that GasNet controllers are faster to re-evolve in modified environments. Finally, we evolve GasNet and NoGas controllers for a simple pattern generation task, backing up the hypothesis that GasNet controllers are easier to tune to the particular characteristics of the environment, or in this case tuning to the particular desired output pattern.

We argue that the key feature of the GasNets seen to be useful on this task is the ability to smoothly adapt to the temporal characteristics of the environment. We further argue that it is this *temporal adaptivity* of the GasNet class that enables the successful evolution of networks capable of generating adaptive behaviour. Finally, we propose that if we are to further develop evolvable artificial neural network classes for adaptive control, the starting point must be from within the class of temporally adaptive networks of which the GasNet is a member.

The paper proceeds as follows. Sections 2 to 5 overview the GasNet and NoGas neural network classes, and the evidence for faster evolutionary search. Readers familiar with the GasNet class of network controllers may skip these sections. Section 6 addresses the question of what might lead to differences in evolutionary search time, outlining a number of possibilities. In section 7 we introduce the methods of dynamical systems analysis, illustrating the techniques through analysis of both a predator-prey population model, and the operation of a GasNet controller pattern generation subnetwork. In section 8 we analyse in detail a single GasNet robot controller, with section 9 comparing two functionally equivalent controllers. Section 10 details the re-evolution of evolved controllers in modified environments, while section 11 describes evolution of simple pattern generation networks. The paper closes with discussion.

2 The GasNet and NoGas classes

The "GasNet" class of neural networks (Husbands, 1998; Husbands et al., 1998) incorporates an abstract model of a gaseous diffusing neuromodulator into a more standard artificial neural network. In previous work the networks have been used in a variety of evolutionary robotics tasks, comparing the speeds of evolution for networks with and without (the "NoGas") the gas signalling mechanism active. In a variety of robotics tasks, GasNet controllers evolve significantly faster than networks without the gas signalling mechanism (see e.g. Husbands, 1998; Husbands et al., 1998). Initial work aimed at identifying the reasons for this faster search has focused on the search spaces underlying the GasNet control class, investigating the ruggedness and modality of the spaces (Smith et al., 2001b), non-adaptive phases of evolution (Smith et al., 2001a), and the local landscape evolvability surrounding solutions (Smith et al., 2002b). In this paper we analyse successfully evolved controllers in order to highlight the properties of GasNets leading to faster evolutionary search.

In the GasNet class, the instantaneous activation of a node is a function of both the inputs from connected nodes and the current concentration of gas(es) at the node. The basic network model consists of connected sigmoid transfer function nodes overlaid with a model of gas concentration; the gas does not alter the electrical activity (from here on, we shall refer to activity propagated across synapse connections as electrical activity) in the network directly but rather acts by changing the gain of transfer function mapping between node input and output, in other words modulating the node properties.

The networks used in the experiments described in this paper consist of sigmoid transfer function nodes connected by weighted links. The networks operate in discrete time; on each "tick of the clock", for all nodes i, the output O_i^t at time-step t is calculated as a function of the sum of the inputs to that node, as described by equation 1. This defines the basic NoGas class:

$$O_i^t \quad = \quad \tanh\left[K_i^t\left(\sum_{j\in C_i} w_{ji}O_j^{t-1} + I_i^t\right) + b_i\right] \tag{1}$$

where C_i is the set of nodes with connections to node i with connection weights w_{ji}, O_j^{t-1} the output of node j on the previous time-step, I_i^t the external (sensory) input to node i at time t, and b_i a genetically set bias. Each node has a genetically set default transfer function parameter K_i^0, and for the NoGas class this transfer parameter is fixed over the operation of the network: $K_i^t = K_i^0 \forall t$.

In the GasNet control system, in addition to this underlying network in which electrical signals flow between units, an abstract process loosely analogous to the gaseous diffusing modulators described above is at play. Some units can emit gases which diffuse and are capable of modulating the behaviour of other units through altering their transfer functions. As described below, this modulation changes the transfer parameter K_i^t as the network runs, thus the actual shape of the node's transfer function is altered via the gas modulation mechanism. This form of modulation allows a kind of plasticity in the network in which the intrinsic properties of units are changing during the operation of the network, that is during the robot controller lifetime. In the next sections we describe the gas diffusion and modulation mechanisms.

2.1 Gas diffusion in the networks

Two gases are used in the GasNet model, gas 1 and gas 2. Gas 1 increases the transfer function parameter K in a concentration dependent fashion, while gas 2 similarly decreases K. It is genetically determined, in other words specified in the controller genotype, which gas each node will emit. This is one of the following: gas 1; gas 2; or neither. It is also genetically determined under what conditions emission will occur, one of the following: when the electrical activation, or output activity, of the node exceeds some threshold; when the concentration of gas 1 in the vicinity of the node exceeds some threshold; or when the concentration of gas 2 in the vicinity of the node exceeds some threshold. In the experiments described later in this paper, we typically use an electrical threshold of 0.5, and a gas concentration threshold of 0.1.

In order to incorporate the gas concentration model, the network is placed in a 2D plane, with node positions specified genetically. The network architecture is detailed in section 3, but for now all that must be borne in mind is that each node has some position on this plane. A very abstract model of gas diffusion is used in order to allow the required computations to be carried out in real time for robot control. For an emitting node, the concentration of gas $C(d,t)$ at distance d from the node and time t is given by equations 2 to 4[1]:

$$C(d,t) \quad = \quad \begin{cases} C_0 \times e^{-(d/r)^2} \times T(t) & d < r \\ 0 & \text{else} \end{cases} \tag{2}$$

$$T(t) \quad = \quad \begin{cases} H(\frac{t-t_e}{s}) & \text{emitting} \\ H(H(\frac{t_s-t_e}{s}) - H(\frac{t-t_s}{s})) & \text{not emitting} \end{cases} \tag{3}$$

$$H(x) \quad = \quad \begin{cases} 0 & x \leq 0 \\ x & 0 < x < 1 \\ 1 & \text{else} \end{cases} \tag{4}$$

where r is the genetically determined radius of influence of the node (concentration falls to zero for $d > r$, which is loosely analogous to the length constant of the natural diffusion of NO, related to its rate of decay

[1]In the early GasNet work (Husbands et al., 1998), the decay with length is given as the exponential $e^{-2d/r}$. Here we use the Gaussian $e^{-(d/r)^2}$; experiments show that this difference is not significant.

through oxidation), C_0 is a global constant, t_e is the last time at which emission was initiated, t_s is the last time at which emission ceased, and s is a genetically determined constant. It should be emphasised that r, C_0, s are determined uniquely for each node by the network genotype, and t_e, t_s will typically be different for each node in the network during operation, so the GasNet network class is heterogeneous in the sense that node properties are not the same across the network.

In other words, the gas concentration varies spatially as a Gaussian centred on the emitting node. The height of the Gaussian at any point within the circle of influence of the node linearly increases or decreases over time depending on whether or not the node is emitting gas, according to the function $T(t)$ which saturates at a maximum of 1 and a minimum of 0. The total concentration at any point in the network is found by summing the concentrations from all emitting nodes. Figure 1 shows a possible GasNet, with node 3 emitting gas. Increased gas concentration is shown centred on the emitting node, extending out as far as node 2; thus node 3 can affect node 2 via the modulatory gas effect despite there being no direct synaptic connections.

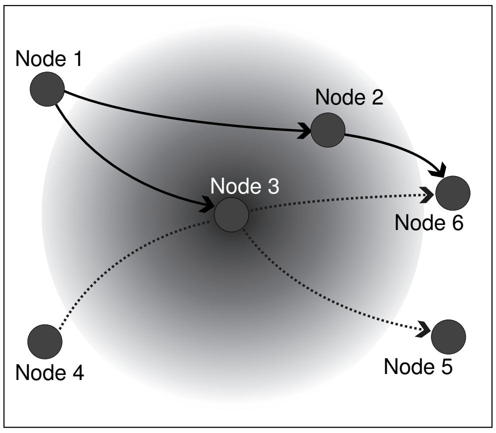


Figure 1: A hypothetical GasNet, with node 3 emitting gas. Increased gas concentration is shown centred on the emitting node, extending out as far as node 2; thus node 3 can affect node 2 via the modulatory gas effect despite there being no direct synaptic connections. All other nodes lie outside the radius of gas concentration, so are not affected by the gas concentration. However, nodes 5 and 6 are affected by synaptic output from node 3. Nodes 1 and 4 similarly affect node 3 through synaptic output.

It should be stressed that the model is a greatly simplified form of the real diffusion process, and certainly does not model the physics of diffusion accurately. Diffusion occurs instantaneously, with no spread of gas concentration over time. In other words nodes close to the emitting node are affected at the same time as nodes far from the emitting node. Also, individual node contributions are summed independently without regard to the local concentration gradient, thus concentration can flow from areas of low concentration to areas of high concentration. Finally, the two gases do not interact, so concentration of one does not affect the diffusion of the other. However, even such an abstract model can capture some of the properties of real diffusing gaseous modulation, and in the next section we describe the modulatory effects of gas concentration.

2.2 Modulation by the gases

As outlined above, the GasNet transfer function parameter K_i^t is modulated during the operation of the network through the concentrations of gas 1 and gas 2, in effect changing the gain of the transfer function given by equation 1. This modulation can occur on any time-step over the lifetime of the network, allowing a form of plasticity very different from that found in most traditional artificial neural networks. The transfer parameter K_i^t for node i on time-step t is described by equations 5 to 8:

$$K_i^t = \mathbf{P}[D_i^t] \tag{5}$$

$$\mathbf{P} = \{-4.0, -2.0, -1.0, -0.5, -0.25, -0.125, 0.0, 0.125, 0.25, 0.5, 1.0, 2.0, 4.0\} \tag{6}$$

$$D_i^t = f\left(D_i^0 + \frac{C_1^t}{C_0 \times K}(N - D_i^0) - \frac{C_2^t}{C_0 \times K} D_i^0\right) \tag{7}$$

$$f(x) = \begin{cases} 0 & x \leq 0 \\ \lfloor x \rfloor & 0 < x < N \\ N-1 & \text{else} \end{cases} \tag{8}$$

where $\mathbf{P}[i]$ refers to the ith element of set $\mathbf{P}$, D_i^t is the node i's pointer into the set $\mathbf{P}$ of possible discrete values that K_i^t can assume, N is the number of elements in $\mathbf{P}$ (13 are shown in equation 6), D_i^0 is the genetically set default value for D_i^t (note that in section 2 we stated that the values for K_i^0 were genetically determined, however more precisely D_i^0 is genetically determined), C_1^t is the concentration of gas 1 at node i on time-step t, C_2^t is the concentration of gas 2 at node i on time-step t, and C_0 and K are global constants (both set to 1 in the experiments reported in this paper).

In other words, the presence of gas 1 increases D_i^t, while the presence of gas 2 decreases D_i^t. This modulation is dependent on both the concentrations of gases 1 and 2, and on the genetically specified value of D_i^0, so the same concentration of the same gas at different nodes will not in general produce the same effect. Figure 2 shows the family of node input-output transfer function curves, for a variety of transfer parameters K.

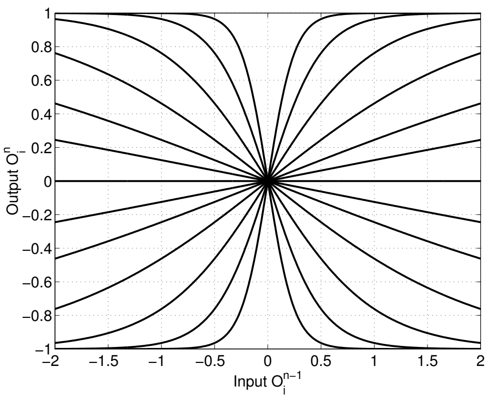


Figure 2: Family of curves defined by $y = \tanh(kx)$ transfer function for a range of values of K. Each curve shows the relationship between x (over the range [-5,5]) and y for a different value of K. The following set of K values are illustrated: $\{-4, -2, -1, -0.5, -0.25, -0.125, 0, 0.125, 0.25, 0.5, 1, 2, 4\}$.

Thus over time, electrical activity flows around the network as in more standard ANNs, alongside a changing pattern of gas concentration. Gas is emitted under certain conditions, with concentration increasing linearly during emission, and decreasing as a Gaussian function of distance from the node. Electrical activity can indirectly affect gas concentration through providing the conditions for nodes to start or stop emitting gas, while gases indirectly affect electrical activity through modulation of the input-output functions of nodes where gas concentration is non-zero. Thus the GasNet model consists of two interacting dynamical processes operating over different temporal and spatial scales. In the next section we detail the genotype-to-network mapping used in the experiments.

3 GasNet neural networks

The GasNet genotypes specify an arbitrary recurrent ANN, with a large number of the network parameters under evolutionary control. In particular, the network genotype specified the size of the network, the network connectivity, node properties, and the input morphology (in this case the input pixel positions). Section 3.1 specifies the individual node encoding scheme, while section 3.2 describes how the network connectivity is formed.

3.1 The node encoding scheme: genotype to phenotype mapping

Each GasNet was encoded on a variable sized genotype coding for a variable number of nodes. A genotype consisted of an array of integer variables, each lying in the range $[0, 99]$. For continuous variables, the phenotype value was obtained by normalising the genotype value to lie in the range $[0.0, 1.0]$[2] and multiplying by the relevant variable range. For nominal values, such as whether the node has a visual input or not, the phenotype value was calculated through the binary modulo division operator:

$$p = \begin{cases} g/99 & \text{continuous } p \\ g \bmod N_{nom} & \text{nominal } p \end{cases} \tag{9}$$

where p is the phenotype value, g the genotype value, N_{nom} the number of possible nominal phenotype values, and mod the binary modulo division operator, that is the remainder when g is integer divided by N_{nom}.

Each node in the network had either 19 or 25 variables associated with it, depending on which of two possible connectivity encoding schemes were used (section 3.2). All variables were under evolutionary control, see figure 3. A single genotype thus consists of a string with length as some multiple of 19 or 25, coding for a variable number of network nodes.

	$<genotype>$	::	$(<gene>)^*$
where	$<gene>$	::	$<x><y>(<Connection>)^*<I_{on}><I_r><I_\theta><I_{thr}>$ $<rec><TE><CE><s><R_e><D^0><bias>$
Arcs:	$<Connection>$	::	$<R_p><\Theta_{1p}><\Theta_{2p}><R_n><\Theta_{1n}><\Theta_{2n}>$
Points:	$<Connection>$	::	$(<Pt_x><Pt_y><Pt_w>)^4$

Figure 3: The genotype-to-phenotype mapping for the arcs and connections network schemes described in section 3.2.

The encoding shown in figure 3 was used to generate networks conceptualised to exist on a 2D Euclidean plane. $<x>$ and $<y>$ give the position of a network node on the plane. The next 6 or 12 numbers define the synaptic connectivity of the network; section 3.2 gives details of the *arc* and *point* schemes used to derive the connectivity. The rest of a gene is interpreted as follows. $<I_{on}>$ is a binary switch that determines whether or not a node has visual input. If it does, the following three variables encode the polar coordinates of a pixel in the camera image that the node will take input from, and a threshold below which input from that pixel is ignored (visual input is normalised to lie in the range $[0.0, 1.0]$, this is the range of the threshold). See section 3.3 for details of the visual input to the network.

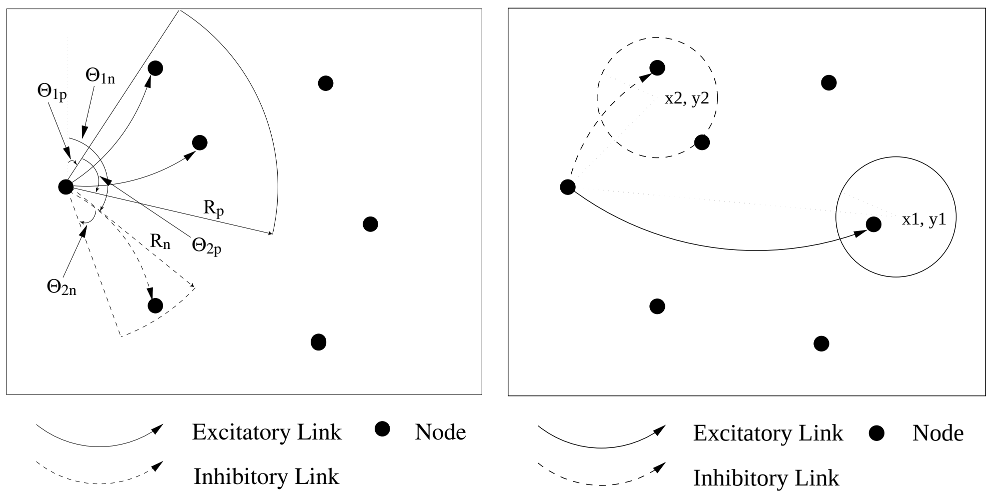
The value of $<rec>$ determines whether the node has an excitatory recurrent connection, an inhibitory recurrent connection or no recurrent connection to itself. $<TE>$ provides the circumstances under which the node will emit a gas. These are one of either: not at all; if the node electrical activity exceeds some threshold; if the concentration of gas 1 at the node site exceeds some threshold; or if the concentration of gas 2 at the node site exceeds some threshold (the electrical and gas thresholds are set at 0.5 and 0.1 as described in section 2.1). $<CE>$ specifies the gas that the node can emit under the correct condition, either gas 1 or gas 2. $<s>$ is used to control the rate of gas build up/decay as described earlier by equation 3, its value ranges from 1 to 11. $<R_e>$ is the maximum radius of gas emission, this ranges from $10\% - 50\%$ of the plane dimension. $<D^0>$ is the default value for the index used in equation 7 to determine the transfer parameter value K_i^t for each node. Finally, $<bias>$ is the bias term b_i in the node transfer function (equation 1), restricted to the range $[-1.0, 1.0]$.

The encoding scheme used was the same for both the GasNet and NoGas classes, with the NoGas genotypes effectively encoded with a number of introns. For the NoGas controllers, certain of the genotype parameters were ignored completely. These parameters were $<TE>, <CE>, <s>, <R_e>$, encoding for the parameters of gas diffusion at each node.

[2] This can be regarded as an approximation to a continuous $[0, 1]$ range; experiments show no significant difference between the two setups.

3.2 The network connectivity: Arcs and points

Two different schemes were used to calculate the synaptic connections between the nodes, see figure 4. The first, the *arc* scheme, connects the start node to any nodes lying in either the positive or negative connection arcs surrounding the node position in the 2D node plane. The radius and angles of the two arc segments are specified genetically ($< R_p >< \Theta_{1p} >< \Theta_{2p} >< R_n >< \Theta_{1n} >< \Theta_{2n} >$), while connection weights are set at ± 1, depending on whether the connected node lies in the positive or negative connection arc. The second *point* scheme connects the start node to the nodes that lie closest to each of a set of four connection points. The x, y node plane positions of each of the connection points, and the connection weights in this scheme are set genetically ($(< Pt_x >< Pt_y >< Pt_w >)^4$). See figure 4 for details.



(a) Connection arcs (b) Connection points

Figure 4: Connectivity of the network is defined by either positive and negative arcs, or circles centred on x, y coordinates. Networks develop and function on a 2D plane.

3.2.1 Connection arcs

R_p gives the radius of the 'positive' arc, Θ_{1p} its angular extent and Θ_{2p} its orientation. R_n, Θ_{1n} and Θ_{2n} similarly define a 'negative' arc. The radii range from zero to half the plane dimension, the angles range from zero to 2π. The arcs are illustrated in figure 4(a). Any node that falls within a positive arc has an excitatory (+1 weighting) link made to it from the arc's parent node. Any node that falls within a negative arc has an inhibitory (-1 weighting) link made to it from the arc's parent node. If the arcs intersect, nodes lying in the intersection will have both excitatory and inhibitory links made to them.

3.2.2 Connection points

x, y coordinate points are used to define node connectivity. Each node has four outgoing connections. Two variables $< Pt_x >, < Pt_y >$, specify each link, defining the centre of a circle on the network plane. The nearest node to this centre within a threshold radius (10% of the plane) has a connection made to it. If no node lies within the threshold radius, no link is made. The connection weight is specified through the third link variable $< Pt_w >$. When this connectivity scheme is used, the six variables of the arcs scheme are replaced with the twelve needed to encode the four circle centres and weights. An alternative was also investigated whereby two of the points encoded +1 weighted connections, and two −1 weighted connections, but no significant differences were seen.

3.3 The network visual input and motor output

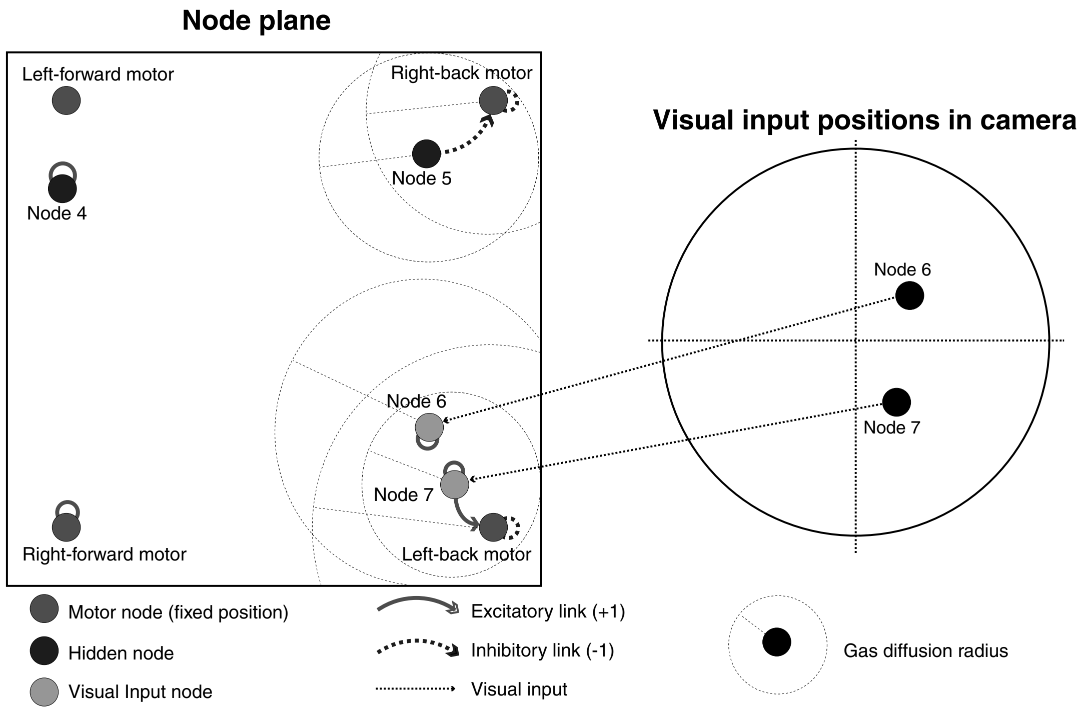


Figure 5: An example GasNet evolved network. The node positions, connections and gas radii are shown in the node plane on the left (gas radii are shown only where the node emits gas during operation). Note the four motor nodes in the four corners of the plane: it was found to be significantly more difficult to evolve successful controllers when these nodes were allowed to change position. The positions of visual input pixels are shown in the right half of the diagram, showing the nodes which receive sensory input. This example network is analysed in section 8.

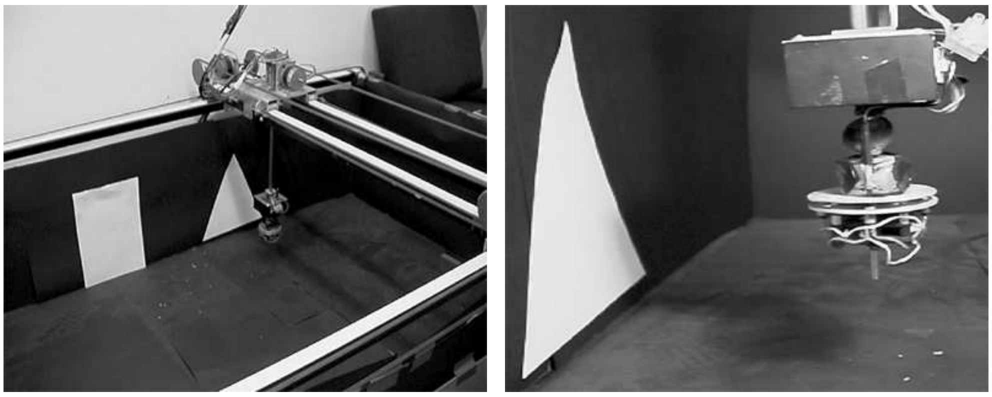
The first four nodes on the genotype (the genotype was required to code for at least five nodes) were taken to be the motor nodes, differing from other nodes in that their position on the plane was fixed to the four corners. Visual input was not permitted to the motor nodes. The four motor nodes were used to drive the two wheels as follows. Each wheel had associated 'forward' and 'back' motor nodes, with each of the nodes considered to be 'on' (+1.0) if the node activity was greater than zero, and 'off' (0.0) otherwise. The actual wheel speeds are set proportional to the output of the relevant forward node minus the output of the relevant backward node. In other words, the output of the network for each wheel can be one of three values: If both the forward and back motor nodes for that wheel are either on or off, the output is 0. If the forward node is on while the back node is off, the output is 1. If the back node is on while the forward node is off, the output is −1. This may seem a crude model of motor control, but it allows evolved control systems to operate successfully on transfer to the real world.

External sensory information can be input to the network at any of the network nodes, except for the motor nodes. As described above, each node encodes whether or not it receives visual input, and the polar coordinates of the input in the visual field. At each time-step, the node receives the intensity level at the pixel position in the visual field specified by the node input coordinates. This intensity is scaled to the range $[0, 1]$, with only visual input intensity higher than the node visual input threshold actually input to the node.

This completes the definition of the GasNet class; a set of sigmoid transfer function neurons with weighted connections, overlaid with a model of diffusing gaseous neuromodulation. Figure 5 shows an example of a successfully evolved network. In the next section we describe the evolutionary robotics visual shape discrimination experiment used in this paper.

4 The robot task

4.1 The Gantry robot



(a) The gantry robot and arena

(b) The gantry robot camera

Figure 6: The Gantry robot. **(a)** The horizontal girder moves along the side rails of the arena, and the robot is suspended from a platform which moves along this girder. **(b)** The camera inside the top box points down at the inclined mirror, which can be turned by the stepper-motor beneath, to give the illusion of rotation. The gantry is best thought of as a two-wheeled robot with fixed camera pointing straight-forward; dedicated hardware and software is used to translate motor commands to the relevant girder, platform and mirror movements, and the camera input is transformed to appear as if received from an onboard camera pointing straight-forward.

The robot task made use of a minimal simulation of the Sussex Gantry Robot (Jakobi, 1998a). Figure 6 shows the real gantry robot.

The robot consists of a CCD camera suspended from the gantry frame, with the camera pointing straight down at a mirror angled at 45^o to the vertical (see figure 6(b)). The gantry frame can move freely along the x, y, z axes, and the mirror can rotate around the vertical axis, in order to simulate the effect of the robot rotating.

Dedicated software takes input as left and right motor commands, and converts the movement into x, y, θ translations for the gantry frame and camera (note that in the work described in this paper, movement along the z direction is not used). The visual input is also transformed, in order to appear as if to come from a robot at the x, y, z position of the gantry head, and looking along a direction specified by the θ angle of the mirror. This transformed visual input appears as a camera view of 20 pixels radius with an acceptance angle of 39^o, giving approximately 1250 pixels in total that can potentially be used as input by the evolutionary process.

The gantry is overall best thought of as a two-wheeled robot with a camera pointing straight-forward, in which the experimenter has access to certain global position information that is not passed directly to the evolutionary process. This position information may be used in the fitness function, for example to determine how close the robot approached to the triangle.

The minimal simulation of the gantry was developed by Jakobi (1998a,b). The base set of robot-environment interactions upon which behaviour could be reliably based, consisted of only two members. First, the way in which pixels of the camera image that are sampled from the walls of the arena (but not from the floor or above the walls) return grey-scale values within certain intervals: over the range $[14, 15]$ for pixels that project onto either the triangle or the square, and over the range $[0, 13]$ for pixels that project onto the walls of the arena, but not onto either the triangle or the square. Second, the way in which the robot moves in response to motor signals.

To ensure that controllers were both base set robust and base set exclusive, in other words that controllers relied only on base set interactions and not on implementation aspects of the model, all other parameters were modelled unreliably. Over the possible ranges of pixel inputs, $[14, 15]$ for pixels that project onto either the triangle or the square, $[0, 13]$ for pixels that project onto the walls of the arena, and the entire $[0, 15]$ range for other cases, values were returned unreliably (remember that the base set aspect is the range over which the pixel values are returned, *not* the way in which they are set over this range). This unreliability was set at the start of each trial, with possible effects varying pixel inputs as a function of time, or as a function of the orientation of the robot, or fixed for the entire evaluation at a random level set before each trial. The momentum of the robot was also made unreliable, with the momentum being fixed at the start of each trial. Similarly, small offsets were added to the wheel speeds, camera horizontal and vertical angles, and the positions of the shape vertices, with the offsets set randomly at the start of each trial. For further details see Jakobi (1998a,b).

4.2 Visual shape discrimination

Starting from an arbitrary position and orientation in a black-walled arena, the robot must navigate under extremely variable lighting conditions to one shape (a white triangle) while ignoring a second shape (a white square). Fitness over a single trial was taken as the fraction of the starting distance moved towards the triangle by the end of the trial period, and the evaluated selective fitness for each controller was returned as the weighted sum of N trials of the controller from different initial conditions:

$$F = \frac{2}{N(N+1)} \sum_{i=1}^{i=N} i(1 - \frac{D_i^F}{D_i^S}) \qquad (10)$$

where D_i^F is the distance to the triangle at the end of the ith trial, D_i^S the distance to the triangle at the start of the trial, and N the number of trials, sorted in descending order of $(1 - \frac{D_i^F}{D_i^S})$. Thus good trials, in which the controller moves some way towards the triangle, receive a smaller weighting than bad trials, encouraging robust behaviour on all trials. In practice we use 16 trials, changing the relative positions of the triangle and square, and the starting orientation and position of the robot, on each trial.

Evaluations are carried out in a *minimal simulation* (Jakobi, 1998a) of the gantry robot, described in section 4.1, with large amounts of noise added to sensor and motor readings, so that controllers will transfer to robots operating in the real environment. Figure 7 shows the fitness distribution of a single genotype evaluated $10,000$ times in the minimal simulation environment, while figure 8 shows a screen shot of a simulated evaluation. As in many problems requiring controllers to provide sensor-to-motor mappings over time, fitnesses are extremely time consuming to evaluate (in the work presented here, evaluating a sample of 10^6 fitnesses takes around 24 hours on a Pentium II 700MHz machine) and inherently extremely noisy. Success in the task was taken as an evaluated fitness of 1.0 over thirty successive generations of the evolutionary algorithm.

GasNet controllers were fully under evolutionary control, with the network connectivity, node properties, and input morphology (in this case the input pixel positions) specified by a network genotype. Full details

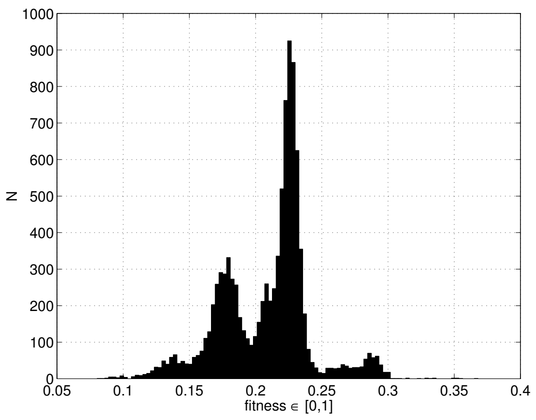
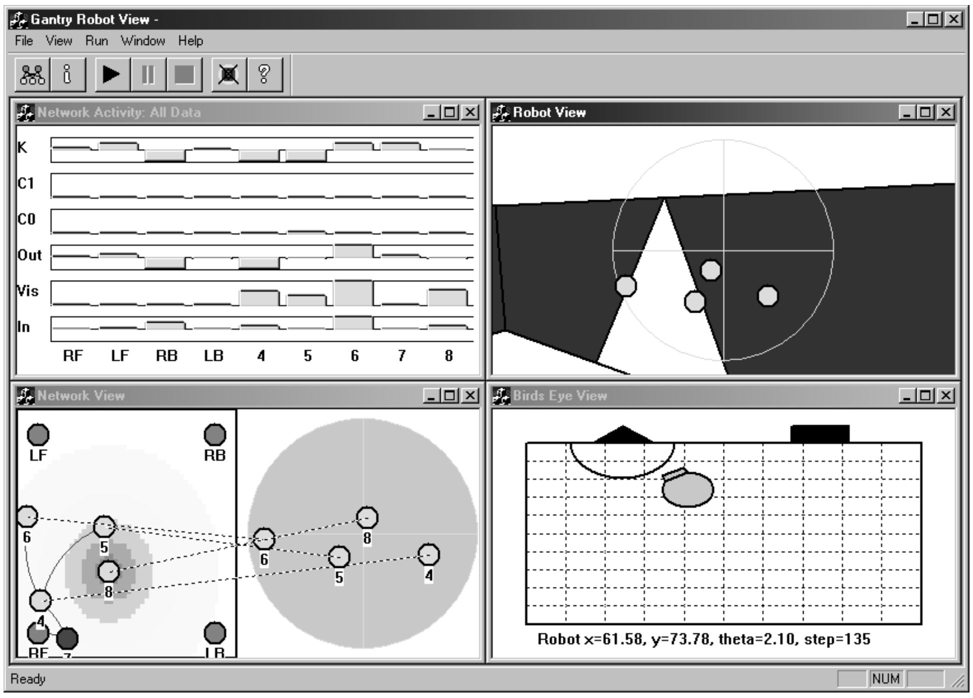


Figure 7: The fitness distribution of a single genotype evaluated 10,000 times in the minimal simulation evaluation environment. 95% of the fitnesses lie in the range $[0.1343, 0.2856]$, with possible controller fitness $\in [0, 1]$.



Robot x=61.58, y=73.78, theta=2.10, step=135

Figure 8: Screen shot of the simulated arena and robot. The bottom-right view shows the robot position in the arena with the triangle and square. Fitness is evaluated on how close the robot approaches the triangle. The top-right view shows what the robot 'sees', along with the pixel positions selected by evolution for visual input. The top-left view shows the current activity of all nodes in the neural network. The bottom-left view shows the robot control neural network: the visual input positions in the camera are shown on the right, with the nodes they connect to placed in the network plane on the left. The motor output nodes *RF*, *LF*, *RB* and *LB* are shown in the four corners of the network plane, and high gas concentrations are shown by shading, such as that surrounding node 8.

of the GasNet controllers used in the gantry visual discrimination task are given in section 3, along with the genotype-to-network mapping.

In the next section we detail the evolutionary algorithms used in the visual shape discrimination experiments.

5 The evolutionary machinery

5.1 Mutation and recombination operators

Three mutation operators were applied to solutions with probability $\mu\%$ during the evolution and re-evolution experiments reported in this paper (for the experiments detailed here, $\mu = 4$). First, each integer in the string had a $\mu\%$ probability of mutation in a Gaussian distribution $N(0, 10)$ centred on its current value (20% of these mutations completely randomised the integer). Second, there was a $\mu\%$ chance *per genotype* of adding one neuron to the network, increasing the genotype length by 19 or 25 depending on the synapse connectivity used (section 3.2). Third, there was a $\mu\%$ chance per genotype of deleting one randomly chosen neuron from the network, decreasing the genotype length by 19 or 25 (note that these two operators checked that the length did not pass certain minimum and maximum bounds; the network was not allowed to contain greater than 50 neurons or fewer than 5 neurons).

It should be noted that the value of $\mu = 4$ used in these experiments is a much larger level of mutation than typically used in artificial evolution optimisation (and certainly much larger than in biological evolution). However, lower levels of mutation produce extremely slow evolution of successful solutions, mainly due to the extremely high level of neutrality seen in the solutions found through evolution (Smith, 2002).

For the NoGas networks, certain parameters coding for the gas diffusion details were simply ignored, with parts of the genotype effectively coded as introns. This does not affect the mutation rate *per locus* as the rate is specified per genotype locus, however it does affect the expected mutation rate *per genotype*. Thus the expected number of mutations for a GasNet controller is greater than the expected number of mutations for the same sized NoGas controller, by a factor of the ratio of their lengths (25/21 for the point encoding scheme, 21/17 for the arc encoding scheme, described in section 3.2). However, experiments in which the mutation rates were varied by much greater factors showed comparable results (Smith, 2002).

Recombination operators were also investigated, with two different types implemented. The first version used a version of two-parent uniform crossover (Mitchell, 1996), however instead of each genotype locus having equal probability of being taken from each parent, each neural network node had equal probability of being taken from either parent. In other words the unit of crossover was the network node rather than the genotype locus. The second version used two-parent one-point crossover (Mitchell, 1996) in which a cut-point was chosen at random; genetic material lying before this point was taken from parent A, while genetic material lying after the cut-point was taken from parent B.

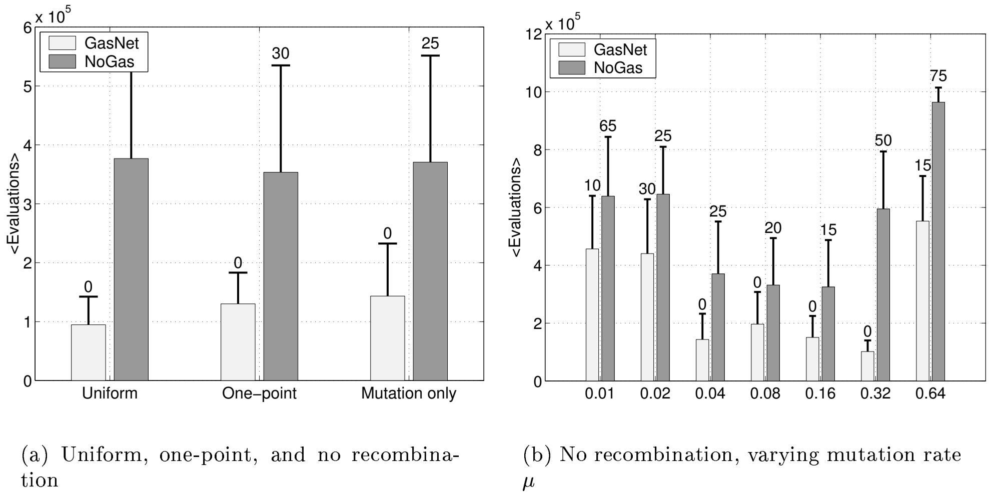
5.2 The evolutionary algorithm

The evolutionary experiments described in this paper used a steady-state distributed evolutionary algorithm. A population of 100, initially randomly generated, solutions were arranged on a 10x10 grid, with all initial fitnesses evaluated. In a breeding event, a single solution was picked at random on the grid, and a mating pool of 9 solutions created, consisting of the randomly picked solution plus the 8 nearest grid neighbours. Rank-based roulette selection was used to select a single parent, and the offspring created through application of the mutation operator. The offspring was placed back into the mating pool, replacing a solution chosen through inverse rank-based roulette selection, that is selection proportional to the solution rank in order of ascending fitness. The parental fitness was probabilistically re-evaluated to avoid over-valued solutions contributing too much genetic material to the next generation. One generation was specified as 100 such breeding events, and the evolutionary algorithm run for a maximum of $10,000$ generations, or until successful solutions had been produced.

In all experiments, the evolutionary populations were initially seeded with networks containing ten nodes, and evolution was continued until the best fitness found did not fall below 1.0 over thirty consecutive generations.

5.3 Speed of evolution results

The evolution of solutions based on the GasNet class consistently produces successful robot control solutions in significantly fewer evaluations than required by the evolution of solutions based on the NoGas class. This result holds over a number of different evolutionary algorithms, with a number of different mutation and recombination rates used, including fixed length genotypes, uniform and one-point crossover recombination, and mutation rates affecting from 1% to 62% of the genotype loci. Two different network connectivity schemes were also investigated, arcs and points, with both seen to show faster GasNet evolution. For full results, see Smith (2002). Figures 9 and 10 show example results.



(a) Uniform, one-point, and no recombination

(b) No recombination, varying mutation rate μ

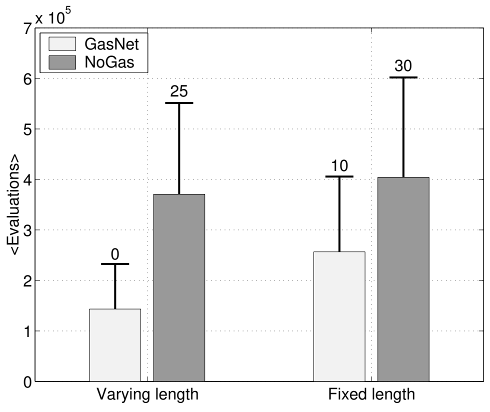
Figure 9: The mean number of evaluations required for **(a)** Uniform, one-point and no recombination, and **(b)** No recombination, varying mutation rate $\mu \in \{0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64\}$. Data averaged over twenty runs of the distributed evolutionary algorithm. The error bars represent 95% confidence limits for the mean, the number above the bar gives the percentage of runs failing to finish in 1, 000, 000 evaluations.

In the next section, we outline possible reasons for this evolutionary speed difference.

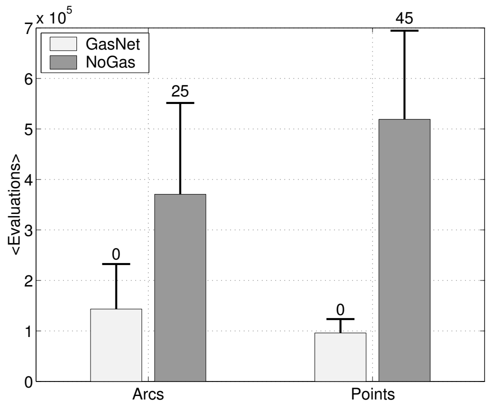
6 Why are GasNets more evolvable?

What are the reasons for the speed of evolution differences seen between the GasNet and NoGas spaces, shown in figures 9 and 10? We can frame many of the possible reasons as properties of the underlying fitness landscapes. There may simply be many more successful GasNet than NoGas solutions, simplifying the search problem. The underlying search spaces may differ in their ruggedness, local modality, degree of neutrality or some other property, making it easier for search processes to find solutions of increasing fitness. More subtle effects may also be important, as properties of the fitness landscapes may not be homogenous across the space; for example the GasNet space may contain smaller regions that are easier to search in some way.

In previous work, Smith et al. (2001b) have shown no evidence for increased numbers of GasNet solutions in the search space; massive random sampling shows very few solutions of either class above 50% fitness, even though this fitness is relatively easy to find through directed search. It is entirely possible that the number of high fitness GasNet solutions is significantly larger than the number of NoGas solutions, but this is extremely difficult to show without exhaustive sampling of the space, which is clearly impractical. Similarly, it was seen that the spaces do not measurably differ in ruggedness and modality at different fitness levels. Smith et al. (2002a) develop the technique of *fitness evolvability portraits*, using the fitness



(a) Variable and fixed length genotypes



(b) Arcs and points connectivity schemes

Figure 10: The mean number of evaluations required for **(a)** variable and fixed length genotypes, and **(b)** Arcs and points connectivity schemes. Data averaged over twenty runs of the distributed evolutionary algorithm. The error bars represent 95% confidence limits for the mean, the number above the bar gives the percentage of runs failing to finish in 1, 000, 000 evaluations.

distribution of the search space surrounding solutions to build up a description of the fitness landscape. However, applying such measures to the GasNet and NoGas search spaces shows no measurable differences in ruggedness, modality and neutrality between the landscapes underlying the two classes (Smith et al., 2002b).

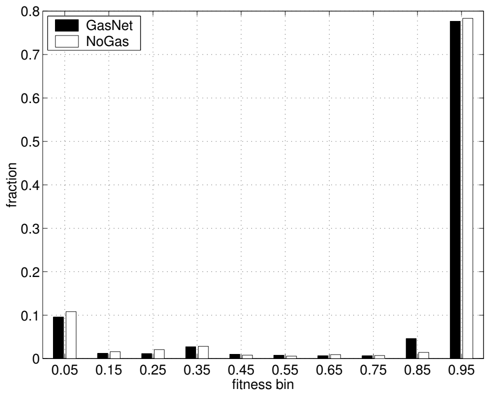


Figure 11: Robustness of GasNet and NoGas solutions; Fitness of the one-point mutations from sample of successfully evolved controllers. Fitness evaluated in non-noisy simulation.

The control classes are also of similar robustness to mutation; figure 11 shows the fitness distribution of all one-point mutations from the sample of successfully evolved controllers used in section 10. The robustness of the GasNet and NoGas controllers are extremely similar; in both cases roughly 80% of mutations are neutral, with 10% catastrophic (it should be noted that the mutation operators used during optimisation typically mutate more than one loci value, so the observed degree of neutrality will be significantly smaller than this 80%).

However, the understanding of search difficulty in terms of the fitness landscape properties is not simple in such a complex search space. The differences between the spaces may be small enough to be obscured by variation; it may be the case that search processes only 'recognise' these differences when iterated over large numbers of fitness evaluations. It is also possible that the distinctions between the spaces are not measurable by the techniques at hand; some other property of high-dimensional search spaces may be involved. In this paper we develop a different approach, through analysis of successfully evolved controllers, allowing us to identify general properties of the GasNet and NoGas classes that may hold in a wider class of problems than just the visual shape discrimination task used here. In the next section we introduce the techniques of dynamical systems analysis.

7 Dynamical systems analysis

The dynamical systems approach analyses how a system behaves over time, in particular investigating the future behaviour of the system given its current state. In this section we outline the basic theory applicable to dynamical systems in general, and apply the theory to a simple example system from the biological literature; the predator-prey population model. We then investigate the behaviour of an example subnetwork from the robot controller analysed more fully in section 8, showing how the dynamical systems method can fully explain the pattern generation properties of this subnetwork.

A full description of dynamical systems theory and analysis is well beyond the scope of this paper; there is an extremely large body of literature dealing with a variety of dynamical systems, including physical (see for example Goldstein, 1980), biological (see for example Rosen, 1970; Rubinow, 1975) and chemical systems (see for example Gavalas, 1968). However for the sake of completeness, in this section we give a brief overview; for further information the reader should consult a standard text.

7.1 Introduction to dynamical systems

A dynamical system is defined by some finite set of *state variables*, and some *dynamical law* by which the state variables change over time. If the state variables are sufficient to describe the system fully, the system is said to be *autonomous*. If other variables need to be taken into account to describe the full behaviour, the system is said to be *non-autonomous*. A non-autonomous system can generally be transformed into an autonomous system by simply incorporating the external variables into the description. However this is not always useful or practical, for example in systems in which the dynamical law changes over time.

Given the initial conditions for the state variables, the dynamical law describes the future behaviour of the system. Thus we can represent the system as a single point in the phase space of state variables (both the *phase* and *state* space refer to the space produced when mapping one dimension for each state variable); the phase space portrait of the system shows the movement over time of this point, and hence the evolution of the system. Typically the number of state variables is greater than three, so this state space cannot be viewed directly, and only two or three variables of interest are mapped in such a way. For example, even a single atomic particle requires six state variables for its position and velocity in space: three variables for the x, y, z coordinates, and three for the P_x, P_y, P_z momenta (in this case, the dynamical law would typically take the form of the force exerted on the particle through an external field). Thus in most systems of interest, we will have to focus on particular measurements of the system.

In general, we are unlikely to be able to solve the dynamical equations directly. However, we may be able to find the *equilibrium states* (or fixed points) of the system, at which the system remains unchanged over time. These will correspond to points in the phase space at which the derivatives of all the state variables are zero, and can be *stable* (if perturbed, the system will return to the fixed point) or *unstable* (if perturbed, the system will not return to the fixed point). A simple example of a system with two such fixed points is that of a needle balanced on a table. The first fixed point is the needle in a perfect upright position, which is clearly unstable since any small movement will inevitably topple the needle. The second fixed point is the needle lying on its side - the stable equilibrium it will fall into from nearly every other initial position

Over time the system is likely to fall into these stable attractor states, or oscillate with fixed period in some limit cycle behaviour (although chaotic behaviour is also possible). Generally, the goal of dynamical systems analysis is to find this long-term behaviour given the dynamical law and initial states of the system. A related goal is to describe the conditions under which the system may fall into another behaviour, through approaching a different attractor state. Such analysis has been carried out to understand the behaviour of a variety of evolved robot controllers, most especially in the work by Beer and co-workers (see for example Beer, 1990; Beer and Gallagher, 1992; Beer, 1995; Chiel et al., 1999; Calvatti and Beer, 2001). However, in this paper we are interested in analysing the controllers operation in terms of how easy or difficult such controllers may be to evolve, especially when using different robot control classes. In the next two sections, we apply the basic techniques to two example dynamical systems; a predator-prey population model, and an evolved pattern generation GasNet controller.

7.2 An example dynamical system: Predator-prey populations

The classic biological dynamical system is the predator-prey (or host-parasite) population model famously studied by both Lotka (1925) and Volterra (1926). In this model we are interested in how the populations of the two species vary over time, especially with respect to initial conditions of the state variables.

Consider a population x. Over time, the population increases exponentially in size through breeding, but with a self-limiting factor dependent on the current population size, for example due to overcrowding or limited food resources. Thus we derive the differential equation for the rate of change of the population:

$$\frac{dx}{dt} = a.x(1 - b.x) \tag{11}$$

Now consider a prey population x, and a predator population y. Both populations change over time as given by equation 11, although we neglect the self-limiting factor for the prey population as we assume the predators never let the prey population reach such a level. However, there is an additional population interaction term when both predator and prey are present: the probability of a predator-prey encounter is proportional to the product of *both* the predator and prey populations. Thus the predator population increases, and the prey population decreases, with rate proportional to the product of the two populations. From these premises we derive the differential equations, or dynamical laws, governing the rate of change of the two populations:

$$\frac{dx}{dt} = a.x(1 - y) \tag{12}$$

$$\frac{dy}{dt} = b.y(1 - c.y + x) \tag{13}$$

the Volterra equations for the predator-prey system. In the rest of this paper, we consider discrete time-step neural networks in which the activity of a network node is derived directly, rather than through differential equations. Thus we use the discrete form of the above equations (Sandefur, 1990, gives a good introduction to discrete dynamical systems):

$$\frac{x^{t+1} - x^t}{\Delta t} = a.x^t(1 - y^t) \tag{14}$$

$$\frac{y^{t+1} - y^t}{\Delta t} = b.y^t(1 - c.y^t + x^t) \tag{15}$$

the change in x and y from time-step $t \to t + 1$. Without loss of generality, we can take the time-step $\Delta t = 1$. Thus we have our dynamical equations for the predator-prey system $\mathbf{A}^{t+1} = \mathbf{F}(\mathbf{A}^t)$:

$$\begin{pmatrix} x^{t+1} \\ y^{t+1} \end{pmatrix} = \begin{pmatrix} a.x^t(1 + 1/a - y^t) \\ b.y^t(1 + 1/b - c.y^t + x^t) \end{pmatrix} \tag{16}$$

Now, we are interested in the behaviour of the system over time, so we need to find the equilibrium values. When the system is in equilibrium, the state variables do not change over time: $d\mathbf{A}/dt = 0$, or $\mathbf{A}^{t+1} = \mathbf{A}^t$. Solving for our system, we find three equilibrium values at $(0, 0), (0, 1/c)$ and $(c - 1, 1)$.

The next question is whether these equilibrium states are stable or unstable to small perturbations, and is the system likely to converge to one of the states? In general, the stability of a system with m state variables is found through the partial differential matrix $\mathbf{R}$ of the m-vector valued function $\mathbf{F}$ at the fixed points $\mathbf{A}$ of the system:

$$\mathbf{A} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} f_1(\mathbf{A}) \\ f_2(\mathbf{A}) \\ \vdots \\ f_m(\mathbf{A}) \end{pmatrix} \tag{17}$$

$$\mathbf{R} \equiv \begin{pmatrix} \frac{\delta f_1(\mathbf{A})}{\delta a_1} & \frac{\delta f_1(\mathbf{A})}{\delta a_2} & \cdots & \frac{\delta f_1(\mathbf{A})}{\delta a_m} \\ \frac{\delta f_2(\mathbf{A})}{\delta a_1} & \frac{\delta f_2(\mathbf{A})}{\delta a_2} & \cdots & \frac{\delta f_2(\mathbf{A})}{\delta a_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta f_m(\mathbf{A})}{\delta a_1} & \frac{\delta f_m(\mathbf{A})}{\delta a_2} & \cdots & \frac{\delta f_m(\mathbf{A})}{\delta a_m} \end{pmatrix} \tag{18}$$

We find the stability of the equilibrium values from the norm of the partial differential matrix $\mathbf{R}$, that is the absolute value of the largest eigenvalue r:

$$||\mathbf{R}|| = |\text{largest } r| \tag{19}$$

$$\text{where } (\mathbf{R} - r\mathbf{I}) = 0 \tag{20}$$

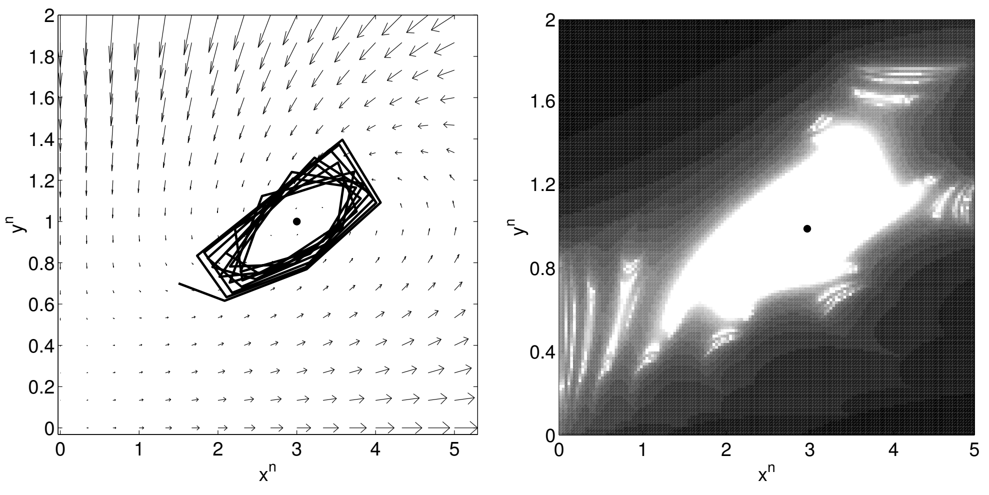
$$||\mathbf{R}|| = \begin{cases} < 1 & \text{stable equilibrium value} \\ > 1 & \text{unstable equilibrium value} \\ = 1 & \text{inconclusive; need to look at higher derivatives} \end{cases} \tag{21}$$

Let us suppose our particular predator-prey system has values $a = 1.3, b = 0.4, c = 4.0$. Now the equilibrium values are $(0,0), (0,0.25)$ and $(3,1)$, and we have the partial differential matrix $\mathbf{R}$:

$$\mathbf{R} = \begin{pmatrix} 2.3 - 1.3y^t & -1.3x^t \\ 0.4y^t & 1.4 - 3.2y^t + 0.4x^t \end{pmatrix} \tag{22}$$

Solving for our equilibrium values shows that the points $(0,0)$ and $(0,0.25)$ are unstable ($||\mathbf{R}|| > 1$), but that $(3,1)$ is stable ($||\mathbf{R}|| < 1$). We also have complex eigenvalues r for the point $(3,1)$, which suggests that the system will oscillate around this point (Sandefur, 1990). Figure 12(a) shows the phase plane analysis of the system around this point. The prey and predator populations are shown on respectively the x and y axes. The vector flow field (shown by the set of background arrows) represents the size and direction of the movement of the system from any initial starting state. The heavy line shows the evolution of the system over time, when started in the neighbourhood of the point $(3,1)$. The equilibrium state is shown by a filled black circle; as predicted, the fixed point is stable, with the system oscillating around this equilibrium state.

In figure 12(b) we show the basins of attraction of the fixed point; from which starting points does the system fall into the stable equilibrium state? As before, the stable equilibrium state is shown by a filled black circle. From each initial starting point, the system was iterated for a fixed number of time-steps, stopping if any state variable fell outside some given range. The plot shows the number of time-steps for which the system remained stable from each initial state. White areas show initial population values for which the system falls towards the stable fixed point. Black areas fall quickly away from the fixed point, with intermediate shades representing initial values which take some time to fall away from the fixed state, that is systems with initial parameters that nearly fall into the limit cycle. We can see that the plot shows complex features - it is conceivable that the system shows fractal features for this stability (the well-known Mandelbrot set is generated through the same stability analysis of an iterated mapping in the complex plane, see for example Devaney, 1989).



(a) System movement towards the fixed point (b) Fixed point basins of attraction

Figure 12: Figure 12(a) shows the phase plane analysis of the system around the fixed point. The prey and predator populations are shown on respectively the x and y axes. The vector flow field (shown by the set of background arrows) represents the size and direction of the movement of the system from any initial starting state. The heavy line shows the evolution of the system over time, when started in the neighbourhood of the point $(3, 1)$. The equilibrium state is shown by a filled black circle; as predicted, the fixed point is stable, with the system oscillating around this equilibrium state. Figure 12(b) shows the basins of attraction for the fixed point; from which initial populations does the system fall into the fixed point? White areas show initial population values for which the system falls towards the stable fixed point. Black areas fall quickly away from the fixed point, with intermediate shades representing initial values which take some time to fall away from the fixed state.

Thus we can characterise the behaviour of the system over time. There are three fixed equilibrium points of the system, two of which are unstable, one stable. If the system is initially placed *exactly* on either unstable equilibrium, it will remain there indefinitely. However, any small perturbation will result in the system moving quickly away from these points. If placed in the vicinity of the stable equilibrium state, the system will oscillate around this fixed point indefinitely. The initial values from which the system will reach this equilibrium value tend to cluster around the fixed point, but show complex structure further out from the point. Thus for a wide range of initial parameters the system will reach stability, with both the predator and prey populations oscillating over time. In the next section, we apply the techniques to a more relevant example, an evolved robot control subnetwork.

7.3 A discrete dynamical pattern generation network

In this section we consider a two node pattern generation network, part of the evolved robot controller analysed more fully in section 8, and shown in figure 13. In particular, we want to see how the properties of both the individual nodes and the gas diffusion mechanism lead to pattern generation.

In the robot controller shown in figure 13, the two node subnetwork in the top-right of the node plane acts as a pattern generator, in which the output of the right-back motor node 'spikes' once every eight time-steps. Figure 14 gives the behaviour of the two nodes over 100 time-steps, showing node output (the bottom two graphs Y_2, Y_5), node transfer parameter (graphs K_2, K_5), and the concentrations of the two gases (graphs $C_1Y_2, C_1Y_5, C_2Y_2, C_2Y_5$). Note the spiking behaviour shown in the motor node Y_2 graph; once in every eight time-steps, the output of this motor node is positive.

As we shall see in section 8.2, this spiking behaviour is crucial to the final fitness of the solution; with

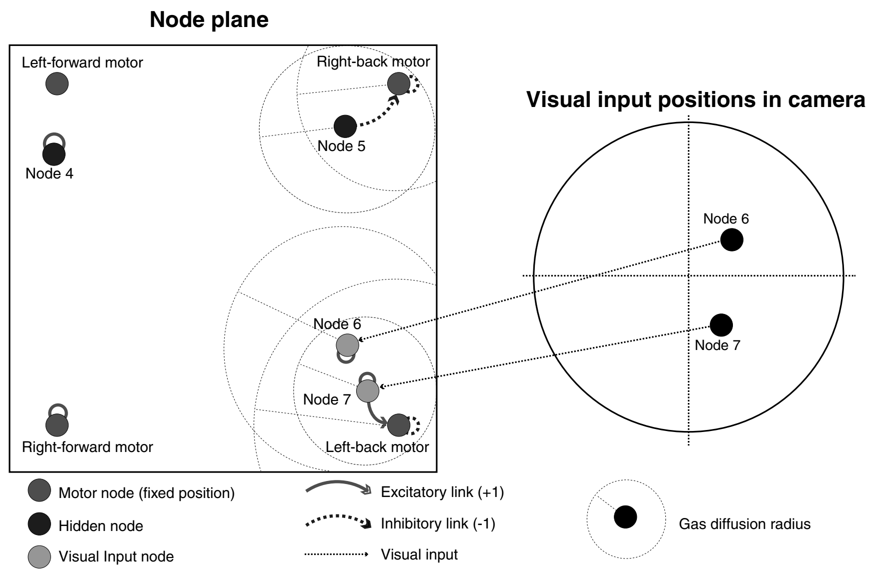


Figure 13: *Open-loop* GasNet visual discrimination network. Gas diffusion radii are shown only where diffusion occurs. The node plane is shown with x, y positions of each node, the connections between each node (indicating whether excitatory/inhibitory) including recurrent connections, and the position in the visual field of any external inputs.

both motors on, the robot will move straight-forwards. However, the right-back motor node turning on once in every eight time steps produces a slow clockwise turn in the robot, which results in the robot arcing back towards the triangle. So, how is this spiking behaviour generated?

The two systems are described by the dynamical equations for the nodes, governed by the input-output transfer function (equation 1). For our two-node system, the right-back motor node (Y_2) receives an inhibitory recurrent input, and an excitatory input from the second node in the network (Y_5). The second node does not receive any input, and neither node receives external sensor input. Both nodes are potential gas diffusion emitters, with the motor node emitting gas 1 (thus increasing K on nearby nodes) when output activity is high, and node 5 emitting gas 2 (decreasing K on nearby nodes) when the concentration of gas 1 at node 5 is high. Thus we can write down our dynamical equations for the nodes:

$$Y_2^t = \tanh\left(K_2^t(-Y_2^{t-1} - Y_5^{t-1}) - 0.66\right) \qquad (23)$$
$$Y_5^t = \tanh\left(K_5^t.(0) + 0.48\right) \approx 0.44 \qquad (24)$$

Note that because node 5 receives no input, recurrent or otherwise, it has a constant output over time of approximately 0.44 for all values of the transfer parameter K_5.

The transfer parameter K is dependent on the current concentrations of gas at the node (section 2.2), and as shown in figure 14 changes during the operation of the network. To consider the dynamical system as autonomous, we would need to incorporate the change of K over time into the equations. However, we can instead regard the dynamical equations as non-autonomous, changing over time through responding to external input, and solve for the values of K that the system is likely to encounter. By default (the genetically determined value) $K_2 = 4$, thus our motor node system is simply the one-dimensional:

$$y = \tanh\left(4(x - 0.44) - 0.66\right) \qquad (25)$$

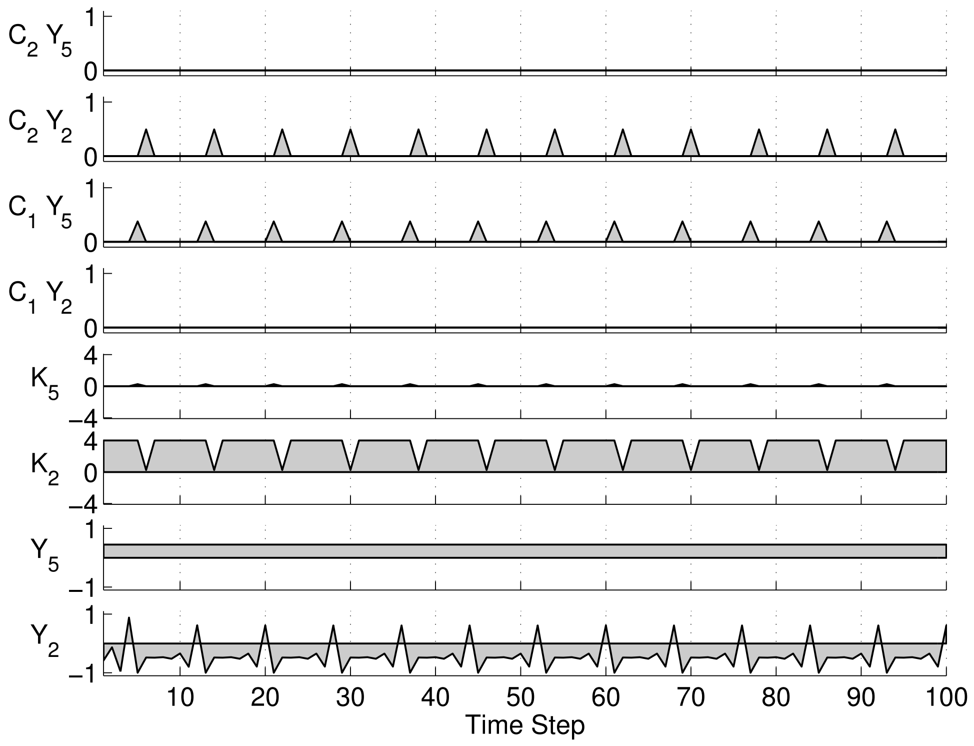


Figure 14: For nodes 2 (the right-back motor node) and 5, involved in the 'spiking' subnetwork, the figure shows data over a run of 100 time-steps for node output $Y \in [-1, 1]$, node transfer parameter $K \in [-4, 4]$, positive and negative gas concentrations $C_1, C_2 \in [0, 1]$ at the node site. Area between the output and time axis is shaded for clarity.

with Y_2^t rewritten as y, and Y_2^{t-1} rewritten as x. By comparison with the M-dimensional case, we find the fixed point(s) by setting $y = x$ and solving, obtaining a single fixed point for the motor node at $y = x = a = Y_2 \approx -0.48$. Now we want to know the stability of this point a. For the one-dimensional case we can use the first differential directly (Sandefur, 1990):

$$\left|\frac{dy}{dx}\right|_{y=x=a} \begin{cases} < 1 & \text{stable equilibrium value} \\ > 1 & \text{unstable equilibrium value} \\ = 1 & \text{inconclusive; need to look at higher derivatives} \end{cases} \tag{26}$$

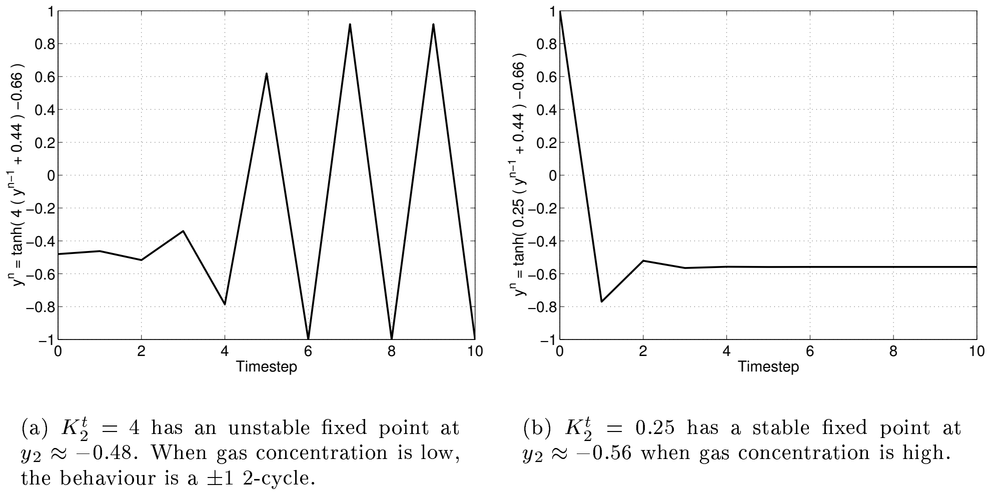
Now, for the tanh transfer function used in this paper:

$$y = \tanh(kx + b) \tag{27}$$

$$\left|\frac{dy}{dx}\right|_{y=x=a} = \left|K(1 - a^2)\right| \tag{28}$$

which gives us the fixed point at $Y_2 \approx -0.48$ as unstable. Intuitively, we can understand this instability as a result of the high level of inhibitory feedback on the motor node. The output over time of the node, in the absence of high gas concentrations, will thus be an oscillating ± 1 2-cycle. In other words the node behaviour is a limit cycle of period 2, in which node output alternates between $+1$ and -1, see figure 15(a).

However, when the activity of the motor node is high, which will happen on every other time-step once transients have died down, the motor node will emit gas 1. The distance between the nodes in the subnetwork is such that the concentration of gas 1 near node 5 will cause node 5 in turn to emit negative gas 2. Consideration of the concentration of this negative gas in the region of the motor node shows that K_2 will decrease from 4 to 0.25, seen in figure 14). Application of fixed point stability analysis shows that the new one-dimensional system for the motor node possesses a stable equilibrium point at $Y_2 \approx -0.55$, understood intuitively through the much smaller inhibitory recurrency on the motor node. Figure 15 shows the output over time of the motor node under the two gas concentrations.



(a) $K_2^t = 4$ has an unstable fixed point at $y_2 \approx -0.48$. When gas concentration is low, the behaviour is a ± 1 2-cycle.

(b) $K_2^t = 0.25$ has a stable fixed point at $y_2 \approx -0.56$ when gas concentration is high.

Figure 15: $y_2^t = \tanh\left(-K_2^t(y_2^{t-1} + 0.44) - 0.66\right)$ behaviour over time for different K_2^t values (modulated by the concentration of gas 2 at the node). The gas concentration mediated switch between these two dynamical states is the basis for pattern generation.

Now, this stable fixed point will result in the motor node ceasing to emit gas due to low output activity, which in turn results in node 5 ceasing to emit gas, as emission was stimulated by the presence of gas 1. The fall in gas concentration will then increase K_2, destabilising the fixed point. The motor node will return to the ± 1 2-cycle, and the pattern repeats.

Thus we have explained the pattern generation subnetwork. The base behaviour of the motor node is a ± 1 2-cycle oscillation, providing the single spike seen in figure 14. This spike stimulates gas emission from both nodes, resulting in the creation of a stable fixed point to which the motor node returns. This equilibrium state is destabilised by the subsequent decay of gas concentration, and the pattern repeats. Thus it is the interaction between the gas and electrical mechanisms in the network that produces the fixed limit cycle spiking behaviour; high electrical activity of node 2 stimulates gas emission, which in turn inhibits node 2, in turn stopping the emission of gas which in turn finally allows node 2 to return to high electrical activity.

In a number of other successfully evolved GasNet controllers we have observed similar subnetworks; it appears that the properties of the GasNet class lend themselves readily to pattern generation. In the next section, we describe the operation of an entire robot control network used for visual discrimination in a noisy environment, using the techniques developed in this section.

8 Open-loop GasNet controller analysis

In this section we analyse in detail the operation of the GasNet controller shown in figure 13. It is seen that the mechanism underlying successful triangle discrimination is a permanent *open-loop* switch from one dynamical system to another, regulated by gas modulation of node properties. The open-loop nature of the switch ignores further external input. We show that the dynamical systems approach can be used to identify a number of possible reasons for the evolvability of the GasNet class, and also analyse the failure modes of the controller.

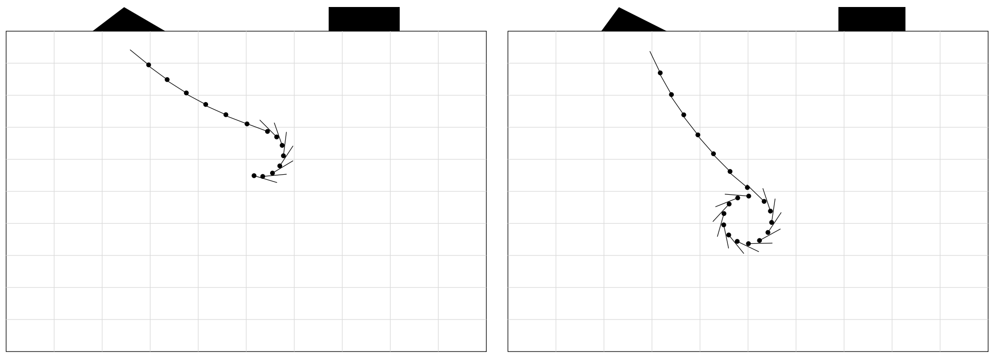


Figure 16: Two evaluations of the open-loop controller analysed in section 8, showing the arena with the triangle and square shapes on the wall. The robot is represented by the black circle, with the line showing the forward direction. Note how the robot curves back in towards the triangle once it starts moving forwards, due to the pattern generation subnetwork analysed in section 7.3.

Figure 13 shows the network layout for the open-loop GasNet controller, while figure 16 shows two evaluations of the controller. In both evaluations, the robot rotates counter-clockwise until *after* it has rotated past the triangle, at which point it moves forwards with a slow clockwise arcing turn which brings it back to the triangle.

The controller behaviour is based on the two subnetworks in the right-hand corners of the node plane (figure 13); both are required for accurate triangle finding behaviour, despite the lack of explicit interaction between the two networks. The first pattern generation subnetwork (consisting of nodes 2 and 5) was described fully in section 7.3. This subnetwork produces a periodic output of the right-back motor node, in which the motor node is on for one time-step in every eight, producing the slow clockwise turn once the triangle has been rotated past. The second subnetwork (consisting of nodes 3, 6 and 7) is described in section 8.1, and produces a fixed behaviour in which the subnetwork permanently switches from one stable state to another, again once the triangle has been rotated past. Both networks rely heavily on gas diffusion effects (disabling gas diffusion results in the failure of both networks), and both are required for the overall triangle discrimination behaviour.

The network is described by the node transfer functions:

$$\begin{aligned}
y_0^t &= \tanh\left(-0.5y_0^{t-1} + 0.48\right) \approx 0.31 && (29)\\
y_1^t &= \tanh\left(0.16\right) \approx 0.16 && (30)\\
y_2^t &= \tanh\left(K_2^t(-y_2^{t-1} - y_5^{t-1}) - 0.66\right) && (31)\\
y_3^t &= \tanh\left(K_3^t(-y_3^{t-1} + y_7^{t-1}) + 0.62\right) && (32)\\
y_4^t &= \tanh\left(0.25y_4^{t-1} - 0.28\right) \approx -0.35 && (33)\\
y_5^t &= \tanh\left(0.48\right) \approx 0.44 && (34)\\
y_6^t &= \tanh\left(K_6^t(y_6^{t-1} + I_6^t) - 0.38\right) && (35)\\
y_7^t &= \tanh\left(K_7^t(y_7^{t-1} + I_7^t) - 0.32\right) && (36)
\end{aligned}$$

where fixed values for the transfer function parameters K_i^t are shown only for nodes where no increase in gas concentration occurs during evaluation.

None of the three nodes *not* involved in the subnetworks receive any external input; nodes 0 and 1 (respectively the right and left-forward motor nodes) stabilise at constant positive values given by $y_0 =$

$\tanh(-0.5y + 0.48)$ and $y_1 = \tanh(0.16)$ respectively, while node 4 stabilises at a constant negative value, $y_4 = \tanh(0.25y - 0.28)$ but is unused by the network. Thus both forward motor nodes are continually on, and behaviour is governed by the two subnetworks acting on the back motor nodes. In the next section we analyse the switching subnetwork.

8.1 Stable state switching

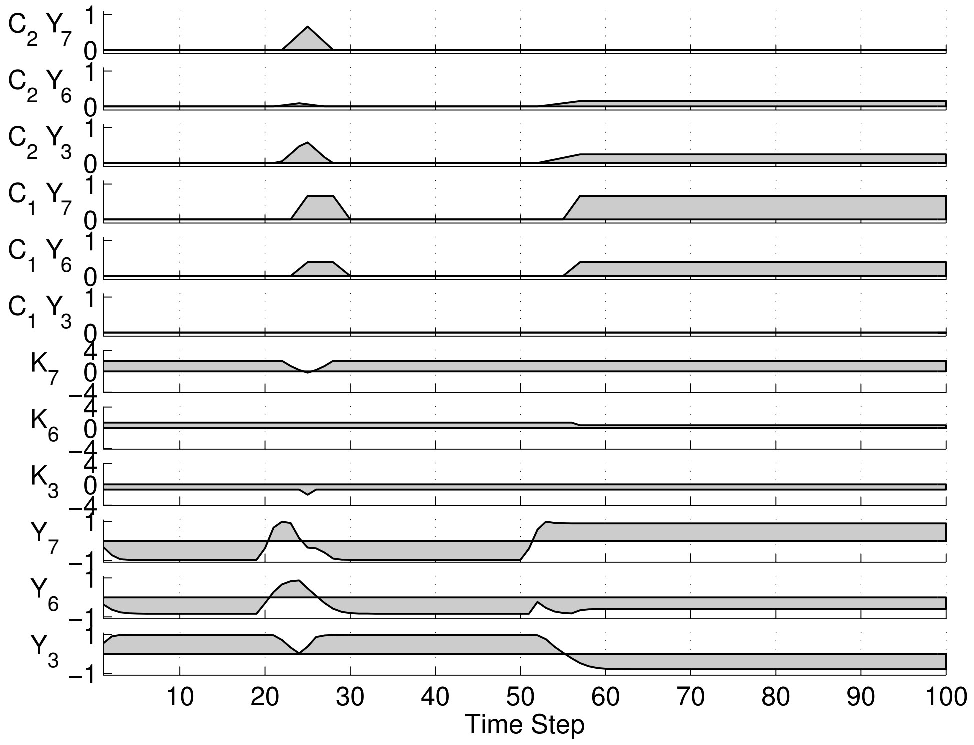


Figure 17: For nodes 3 (the left-back motor node), 6 and 7, involved in the 'switch' subnetwork described fully in section 8.1, the figure shows data over a run of 100 time-steps for node output $Y \in [-1, 1]$, node transfer parameter $K \in [-4, 4]$, positive and negative gas concentrations $C_1, C_2 \in [0, 1]$ at the node site. Area between the output and time axis is shaded for clarity.

In the open-loop controller, the only nodes receiving external visual input are in the subnetwork involved in the triangle discrimination network, consisting of the left-back motor node 3, and nodes 6 and 7. The subnetwork, shown in the bottom-right corner of the node plane (figure 13), regulates the left-back motor node through electrical synapse and gas diffusion effects. Both nodes 6 and 7 receive recurrent and visual input, while the motor node 3 receives recurrent input, plus an input from node 7. Figure 17 shows the output Y, transfer function parameter K, and gas concentrations C_1, C_2 for the three nodes. The three node subnetwork produces a dynamic system which can produce a permanent switch from one stable state to another, when a specific combination of high external sensory input is received. Note that we are treating the subnetwork as a non-autonomous dynamical system, receiving external input from the environment. External visual input is received by nodes 6 and 7, but only when the visual input level is above the genetically specified node input thresholds. We analyse the subnetwork behaviour for the cases when inputs are below threshold, and when inputs are above visual input threshold levels.

8.1.1 Inputs below threshold

Both nodes 6 and 7 have the same high visual input threshold, with only intensities above 0.84 having any effect. So we can investigate the case when input is below this, where the equations simplify to $y_6^t = \tanh(y_6^{t-1} - 0.38)$ and $y_7^t = \tanh(2y_7^{t-1} - 0.32)$ (in the absence of gas, $K_3^t = -1, K_6^t = 1$ and $K_7^t = 2$). The stable solution to these equations is $y_3 \approx 1.0; y_6 \approx -0.8; y_7 \approx -1.0$. Note that y_7 has 3 stable fixed points, but applying fixed point stability analysis (section 7) shows that from an initial position of $y_7 = 0.0$, the $y_7 \approx -1.0$ solution is reached. However, the other y_7 solutions are crucial when input is above threshold, a situation that is analysed in the sections below. Both nodes 6 and 7 emit negative gas when output activity is high, but this is not the case for the stable point. In the presence of negative gas, node 3 emits positive gas - again this is not the case for the stable point.

Thus we have the general picture when no visual input is received above the threshold level of 0.84. Both visual input nodes 6 and 7 are highly inhibited, and the left-back motor node 3 is highly excited. No nodes are emitting gas, and gas concentrations are zero in the neighbourhood of each node. Thus the left motor is inhibited, and the robot circles counter-clockwise, due to the right motor being on for seven in eight time-steps (remember that the spiking subnetwork on the left-back motor node only turns off the motor one in eight time-steps, see section 7.3). So what happens when inputs are above threshold?

8.1.2 Inputs above threshold

The following analysis assumes inputs take their maximum value of 1.0, but is qualitatively the same for all values above the visual input thresholds of 0.84. In the presence of high input to both nodes (again in the absence of high gas concentrations), the equations simplify to $y_6^t = \tanh(y_6^{t-1} + 0.62)$ and $y_7^t = \tanh(2y_7^{t-1} + 1.68)$; the stable solution is $y_3 \approx -0.8; y_6 \approx 0.9; y_7 \approx 1.0$. Note how all the node output activities have reversed; the previously inhibited nodes 6 and 7 are now excited, while the previously excited left-back motor node is now inhibited. The input threshold has produced an on/off 'switch'. The immediate effect of high visual input to node 7 is to turn off the left-back motor node through the inhibitory connection, thus turning on the left motor, so the robot goes in a straight line.

However, the picture is complicated by the emission of gas from the subnetwork nodes. Both nodes 6 and 7 emit negative gas when highly active, and node 3 emits positive gas in the presence of high negative gas concentrations. Three different scenarios are investigated: where both inputs go high at the same time, and where either input goes high first.

In the model of gas diffusion used, gas concentration builds up according to equations 2 to 4, reaching a maximum concentration $C = C_0 e^{-(d/r)^2}$. The node 6 characteristics ensure negative gas spreads out very quickly over a large area: the concentration of negative gas at node 7 due to node 6 emission quickly affects the transfer function (on the very next time-step). The small distance between nodes 6 and 7, and the high value of the radius of gas emission r for node 6, produce a gas concentration that drops K_7 from 2 to -0.25. Now $y_7^t = \tanh(-0.25y_7^{t-1} - 0.57)$ has a stable negative solution (-0.43) even with high positive sensory input. Thus, if node 6 receives bright input before node 7, node 7 is inhibited despite receiving bright input, so does not inhibit the left-back motor node, and the robot continues rotating.

The case where both inputs are bright at the same time is very similar to the case where node 6 receives high input before input 7, due to the faster diffusion of gas from node 6. Node 7 will inhibit the left-back motor node briefly (even more so as the combined negative gas from nodes 6 and 7 is concentrated enough to affect the left-back motor node transfer function parameter) but the negative gas build up due to node 6 quickly inhibits node 7, and the left-back motor node will return to its previous excited state where $y_3 \approx 1.0$.

Finally we turn to the case where node 7 receives bright input before node 6. The immediate effect is for node 7 to inhibit the left-back motor node. The secondary effect is for node 7 to emit negative gas. Despite the slower diffusion of gas from node 7 gas than from node 6, it is still enough to inhibit node 6 so long as node 7 receives bright input for *four or more time-steps* before node 6; this time period is crucial to the behaviour. Even with high input, node 6 cannot now produce output sufficient to emit gas so cannot inhibit node 7. Now, the three solutions to the node 7 equation with no input ($y_7^t = \tanh(2y_7^{t-1} - 0.32)$) mentioned previously come into play. From an initial condition of $y_7 \approx 1.0$, even with no external input, there is a stable solution at $y_7 \approx 0.9$. Thus the network is now in a highly stable state with node 7 output at near maximum with or without external input, node 6 inhibited due to negative gas emitted by node 7, and the left-back motor inhibited due to node 7 synapse output. The overall effect is to switch the network into a permanent open-loop behaviour where further external input is irrelevant. Due to the inhibition of the left-back motor node, the left motor is on and the robot continues straight-ahead. So under what conditions does node 7 receive bright visual input four or more time-steps before node 6?

8.1.3 Visual input positions, success and failure modes

Figure 13 shows that the visual inputs to nodes 6 and 7 are vertically aligned in the visual field, with 7 directly below 6. Scanning across the square will cause both nodes to receive bright input at roughly the same time, thus node 7 will be inhibited by node 6, and the robot will continue rotating. However, scanning across the triangle will cause node 7 to receive bright visual input significantly before node 6, thus inhibiting node 6. This in turn will cause the network to switch into the permanent open-loop state, and the robot will continue straight-ahead. Table 1 summarises the behaviour of the robot as determined by the switching subnetwork.

Node 6 input	Node 7 input	Node receiving first bright input	Left-back motor node	Robot motion
dark	dark	-	excited	ccw rotation
bright	dark	6	excited	ccw rotation
bright	bright	6	excited	ccw rotation
bright	bright	same time	excited	ccw rotation
dark	bright	7	inhibited	straight-forward
bright	bright	7	inhibited	straight-forward

Table 1: Summary of ‘switch’ subnetwork behaviour, showing the robot motion based on the visual input to nodes 6 and 7. The robot rotates counter-clockwise (ccw) for the cases where no bright visual input is received, and for the cases where node 6 receives bright input before or at the same time as node 7. The robot moves straight-forward only when node 7 receives bright input significantly before node 6. Due to the visual input to node 6 being higher in the visual field than the input to node 7, node 7 will only reliably receive bright visual input before node 6 when the triangle is scanned across. Thus the robot will rotate past the square, but move straight-forwards when the triangle is encountered.

We can also see the failure modes from this analysis. It is the four (or more) required time-steps of bright input to node 7, without bright input to node 6, that produces visual input noise filtration. However, an extremely noisy environment may ‘fool’ the controller into the permanent dynamical system switch, through only node 7 receiving bright input. Such a situation may occur with flashes of light aimed at only certain parts of the arena, which might be erroneously identified as triangles by the controller. Similarly, bright shapes with angled edges leading to node 7 receiving input before node 6 will be identified by the controller as triangles, and approached. Finally, triangles with edges insufficiently angled to allow input 7 to be bright for 4 time-steps before input 6 will not be approached by the controller. Among these unidentified triangles will be upside-down triangles and right-angled triangles.

In the next section, we summarise the overall behaviour of the controller, and draw some general principles of GasNet robot controller operation.

8.2 Open-loop GasNet controller summary

The overall behaviour of the robot controller can be summarised as follows. In the absence of bright visual input, the robot rotates counter-clockwise, with the right motor permanently excited, and the left motor inhibited by the switching subnetwork. This behaviour continues, until the robot scans across a bright object, such that the lower half of the visual field receives bright input significantly before the upper half. This permanently switches off the left-back motor node, exciting the left motor and causing the robot to move straight-forward. Now the effect of the spiking subnetwork is seen; once every eight time-steps the right motor is turned off, thus the robot moves in a slow clockwise arc back towards the triangle, which it has rotated past. So we have explained in full the behaviour seen in the two example evaluations, shown in figure 16.

The two subnetworks analysed are crucial to the understanding of the robot controller triangle discrimination, in conjunction with the robot-environment coupling. The primary robot-environment coupling is the permanent switch mechanism; scanning across the square will produce no change in the robot motion beyond a slight slowing of the turn. By contrast, scanning across the triangle will lock the robot into a fixed behaviour in which no subsequent external input affects the network, and with both motors full on the robot goes forward towards the bright object. The second subnetwork is used by the controller to compensate for both the relatively slow time-scale of the permanent switch mediated by the gas diffusion, and the momentum of the robot. While rotating past the triangle the permanent switch behaviour will come into play, but the robot motors will take several time-steps to overcome momentum and friction to actually produce the straight motion. It is the right motor turning off once in every eight time-steps that adjusts for this, turning the robot back towards the triangle. Without this spiking behaviour, the robot would overshoot and run into the wall past the triangle; it is the lack of active 'closed-loop' tracking that produces the need for this compensation.

In the next section we hypothesise why the GasNets network class is more evolvable than the NoGas network class.

8.3 Why are GasNets good for evolution?

From this detailed analysis of the open-loop controller, we can frame some preliminary conclusions on the usefulness of the mechanisms utilised in GasNet controllers for the generation of adaptive behaviour over time.

First, tunable pattern generation is extremely easy to produce using GasNet controllers. In general, pattern generation is based on limit cycle behaviour, with the system cycling through some set of states (Beer et al., 1999; Chiel et al., 1999). As we have seen from the analysis in section 7.3, the spiking subnetwork used by the open-loop controller operated in exactly such a fashion; the high fitness of the controller is due to this subnetwork slowly turning the robot back towards the triangle. This leads to our first hypothesis for why the GasNet class is more evolvable than the NoGas class; the GasNets are more amenable to being 'tuned' to the specific characteristics of the environment. The pattern produced in which the right-back motor node spiked once in every eight time-steps was perfectly tuned to the speed and size of the robot wheels, the size of the triangle, and the size of the arena in which the robot operated. A different pattern would not have produced such high fitness in this environment, and the same pattern would not have produced such high fitness in a different environment. We hypothesise that the same kind of environmental tuning is more difficult with the NoGas class.

The tuning of generated patterns is closely related to our second hypothesis regarding useful properties in the GasNet class for adaptive behaviour; the ability to switch between stable states, in other words a discontinuous change of behaviour determined by external input, and the ability to mediate such switching. This is clearly possible to achieve without gas modulation, but the features of the gas diffusion mechanism allow such a switch to take place over several time-steps, through the build-up of gas concentration levels. This was seen in the functionally equivalent mechanisms of section 8.1, where the switch from rotation to straight-forwards motion was inhibited by bright input, *only when such bright input was received over several time-steps.* Thus the switching can be based on input patterns received over time, not just at a single time-point.

Finally, the ability to filter out noisy input is straight-forward to produce when using the GasNet controller class, similarly through requiring that input be consistent over several time-steps. This was seen in the requirement that bright input was received several time-steps earlier by visual input nodes lower in the visual field, before the controller responded (section 8.1). Thus bright flashes and other noisy environment effects were efficiently excluded by the robot controller.

In the next two sections, we investigate these hypotheses in two ways. First, we analyse and compare two solutions utilising the same shape discrimination strategy, one GasNet controller and one NoGas controller, in order to compare the underlying mechanisms. Second, we re-evolve previously evolved controllers in an environment with different characteristics to the environment in which they were originally evolved, in order to compare the tunability of the mechanisms used by the evolved controllers.

9 Functionally equivalent GasNet and NoGas controllers

Two controllers, one evolved using the GasNet class and one evolved using the NoGas class, were analysed using the dynamical system methods of the previous sections. It was found that both employed the same strategy for the triangle-square discrimination task, based on a method of timing the duration for which bright visual input was received in the upper half of the visual field. Due to triangles being narrower at the top than squares, this allows the controllers to successfully discriminate between the two shapes. In this section we investigate the GasNet and NoGas mechanisms for timing the duration over which bright input is received, and argue that the GasNet mechanism is simpler to tune to the characteristics of the environment.

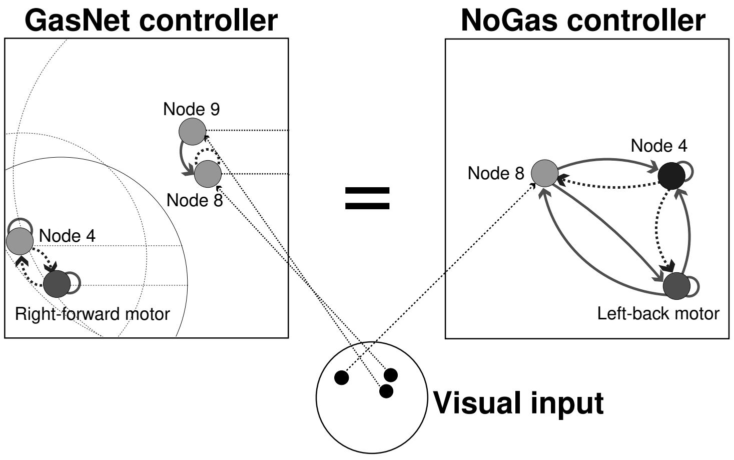


Figure 18: The two functionally equivalent subnetworks analysed in section 9. Both employ the same strategy for triangle-square discrimination; timing the duration of receiving bright input in the upper half of the visual field. Due to triangles being narrower than squares at the top, this allows the shapes to be successfully discriminated.

Figure 18 shows the two functionally equivalent subnetworks; both controllers time the duration over which bright input is received from visual inputs in the upper half of the visual field. A second visual input mechanism (not shown) acts simply as a "bright object finding detector". This bright object finding mechanism is 'later' in the visual field than the timing mechanism, in the sense that the position of visual input and direction of robot rotation is such that the bright object finding mechanism will 'see' things after the timing mechanism. The bright object behaviour is inhibited if the duration of bright input to the timing mechanism is sufficiently long, which is the case when scanning across the square, but not when scanning across the triangle. Thus the controller approaches the triangle, but rotates past the square. In the next sections we describe the methods by which the GasNet and NoGas solutions produce such a timing mechanism.

9.1 The GasNet "timer"

With the GasNet class, it is simple to produce a timing mechanism that retains activity for some time after the initial input has been received. A single node receiving visual input, and with the property that gas emission occurs when the node output activity is high, will start emitting gas when bright input is received. The gas concentration built-up during emission will take some time to decay once bright input is no longer received, with this decay time being a function of the genetically specified rate of gas concentration build-up. Remember from equations 2 to 4 (section 2.1) that gas concentration decays in a Gaussian fashion with distance from the emitting node, but increases linearly over time during emission, and decreases linearly over time once emission stops. As the time taken for gas concentration to build up to a maximum is specified by the genotype, the mechanism is simple to tune to the characteristics of the environment.

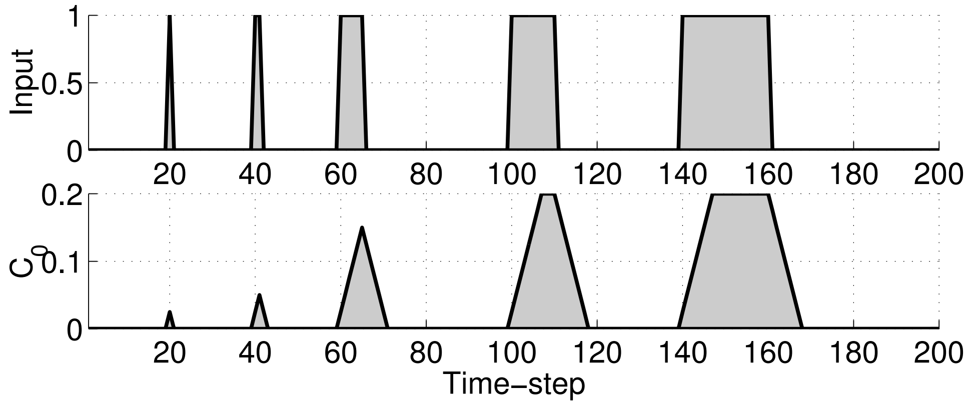


Figure 19: Gas concentration C_0 over 200 time-steps. A square wave external visual input of increasing width is applied as input, to illustrate the differences between the output seen for the triangle and for the square. Area between the output and time axis is shaded for clarity.

For this timing mechanism to affect the controller operation, we require the built-up gas concentration to affect the motor node activity. Again, this is relatively simple to effect, as the gas concentration can modulate either the motor node, or as in the controller analysed here, another visual input node which has an output connection to the motor node. The left-hand side of figure 18 shows the subnetwork responsible for the GasNet timing mechanism; bright input to nodes 8 and 9 produces concentration of gas 1 at node 4. If gas concentration is high enough, the increase in the node 4 transfer parameter K increases output sufficiently to inhibit the right-forward motor node. Thus the robot rotates clockwise past the square, but moves straight-towards the triangle. In the next section we describe the intuitively less-obvious operation of the NoGas timing mechanism.

9.2 The NoGas "timer"

A fully connected three-node subnetwork based around a motor node (see the right-hand side of figure 18) allows the NoGas solution to create exactly the same timing mechanism seen in the previous section. Dynamical systems analysis of the subnetwork shows a single stable equilibrium point for the system when visual input is below the input threshold, and a different single stable equilibrium when visual input is above threshold.

The key to the timing mechanism is how the system moves between these fixed points when the visual input changes. The fully connected feedback nature of this three node system makes it impossible to give a full quantitative description of the behaviour, but qualitative features can be outlined. With no bright input to node 8, the system settles into the first stable fixed point described above, while with bright input the system moves towards the second stable point. Once bright input is no longer received, the system slowly decays back to the first stable fixed point.

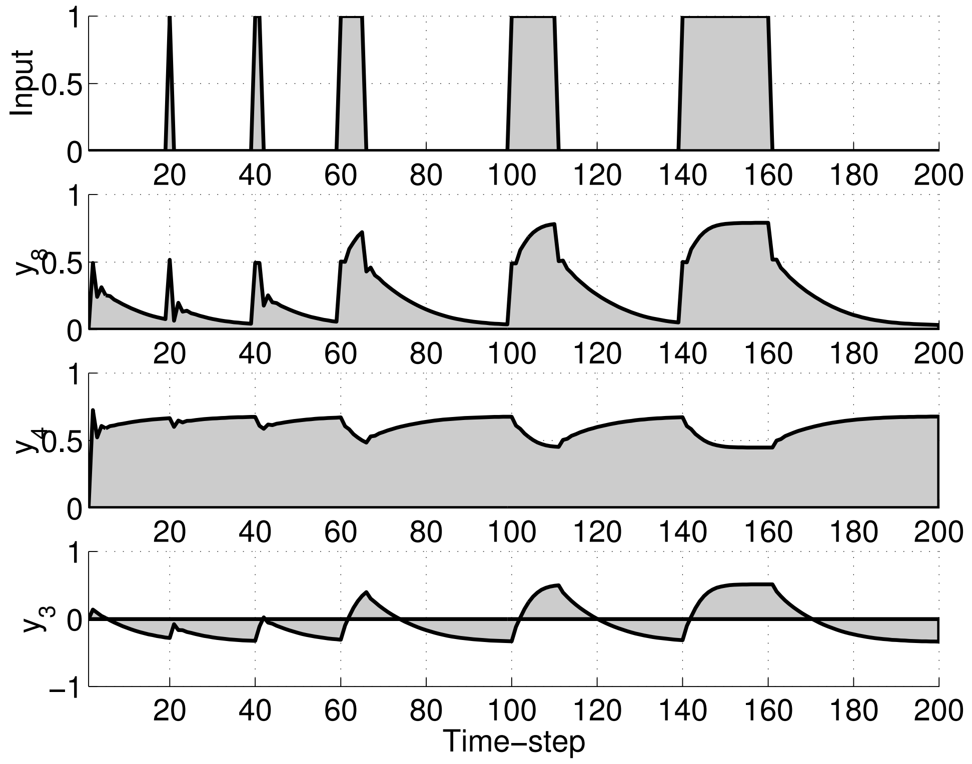


Figure 20: Node output data (ranging from ±1) for nodes 3, 4 and 8 over 200 time-steps. A square wave external visual input is applied to node 8 to illustrate the two fixed points of the system, and the slow decay from one state to the other. Area between the output and time axis is shaded for clarity.

The feedback between the nodes ensures that the decay between stable states is fairly slow, producing an effect which can build-up and decay over time, in a similar fashion to that of gas concentration. The longer that bright input is received for, the nearer to the high visual input stable state the system reaches, and the longer it takes to decay back to the low visual input stable state. Figure 20 shows the outputs for the three nodes in the system when a square wave visual input is applied to node 8, and clearly shows the slow change from one state to another when visual input changes from dark to bright, and vice versa. It takes roughly ten time-steps for node 3 to reach the bright input stable state, and roughly thirty steps to decay back to the dark visual input regime. The motor node activity y_3 is the crucial value; as this goes from negative to positive, the left motor is inhibited, and the robot does not approach the bright object. Only when sufficient bright input has been received will this occur, for instance when the square has been scanned across.

Thus we have our NoGas timing mechanism. Instead of using the GasNet build-up and decay of gas concentration to modulate motor node properties, the NoGas version uses a fully connected subnetwork which decays from one stable equilibrium state to another, depending on the level of visual input received.

As proposed in section 8.3, we hypothesise that the GasNet version is easier to tune to the specific characteristics of the environment than the NoGas version. In the next section, we test this hypothesis for the two functionally equivalent controllers through re-evolving the controllers in environments with modified properties.

10 Re-evolution of controllers in modified environments

The hypothesis that GasNet controllers are easier to tune to the specific properties of the environment than NoGas controllers can be investigated through the behaviour of the controllers in environments with modified properties. In this section, we analyse controllers when evaluated in two separate environments, where the robot motor speeds are respectively set to double and quarter the usual motor speeds. This has the effect of making the robot move at a different speed in the arena, in particular spinning past the two shapes at very different rates to the speeds encountered during the original evolutionary phase. Note that the environments could similarly be modified through altering the size and properties of the shapes, and/or the size of the arena. Other modifications could also investigate the effect of re-evolving from lesioned or similarly modified control networks. However, in this work we focus on modification of the robot motor speeds.

10.1 Re-evolution of the functionally equivalent controllers

We would expect the crucial timing mechanisms described in section 9 to be affected by evaluation in environments with modified robot speeds, with the time spent spinning past the triangle and square much shorter in the double speed environment, and much longer in the quarter speed environment. However, the hypothesis that the GasNet mechanism is in some sense easier to tune to the particular properties of the environment can be tested through seeding the controllers back into the evolutionary process, with fitness based on evaluation in the modified environments. We can then re-run the evolutionary process from the controller seeds, assessing how long before controllers of 100% fitness are again achieved. Although this will not tell us directly how easy the controller was to originally tune to the environment, we argue that the evolutionary tuning processes involved are similar. In other words, if it is much easier to tune the evolved GasNet controller to the specific characteristics of the modified environment, it would also have been much easier to tune the GasNet controller to the original environment.

The two controllers were used to seed the initial populations for the distributed evolutionary algorithm (section 5.2), and evolution repeated twenty times for each controller in each modified environment until controllers of 100% fitness were observed. In this re-evolution, we allow only the parts of the genotype involved in the timing mechanism to be affected by the evolutionary process; we are assessing how easy it is to modify the actual mechanism itself, not the rest of the network.

Results for re-evolution studies of the two controllers are given in table 2. In the double speed environment, both controllers drop in fitness to well under 20%, with no significant differences seen between the fitness of the two controllers. However, there is massive difference in the number of generations required to re-evolve controllers of 100% fitness; the GasNet controller is much easier to tune to the modified properties of the environment (10 generations on average compared with 409 generations). In the quarter speed environment, the GasNet controller achieves significantly higher fitness than the NoGas controller, but the difference in the number of generations required to reach 100% fitness controllers is much larger than might be predicted by this fitness difference (30 generations on average compared with 591 generations). So in both modified environments, the GasNet controllers are much easier to evolve successful controllers than would be predicted from their fitnesses; the GasNets are more tunable.

From the results presented in this section, we can support the hypothesis of the previous section; namely that the GasNet controllers are easier to tune to the specific properties of the environment than the corresponding NoGas controllers. In the next section we extend this to a sample of previously evolved

	Double speed		Quarter speed	
	GasNet	NoGas	GasNet	NoGas
Number of runs	20	20	20	20
Mean evaluated fitness (σ)	0.17 (0.074)	0.15 (0.066)	0.36* (0.10)	0.21 (0.016)
Mean re-evolution generations (σ)	10 (5)**	409 (336)	30 (31)**	591 (346)
Median re-evolution generations	10**	360	19**	608

Table 2: Data for the two functionally equivalent networks shown in figure 18, re-evolved in two modified environments. The robot motors are set to double-speed and quarter-speeds respectively, and the two controllers evaluated 100 times for fitness, then used to seed the initial populations for the evolutionary algorithm until 100% fitness controllers were produced (20 runs were performed for each controller on each condition). The evaluated fitnesses, and mean, median and standard deviation of the number of generations of re-evolution required to reach 100% fitness controllers are shown, with significant differences between the GasNet and NoGas controllers highlighted (both parametric T-tests and non-parametric Mann-Whitney U tests were performed $^{*}p < 0.05,^{**} p < 0.01$).

controllers.

10.2 Re-evolution of a sample of controllers

It may be argued that the two functionally equivalent controllers investigated are based on a mechanism which is in principle easier to both produce and tune using the GasNet control class. Thus re-evolving the timing mechanism in modified environments will unfairly favour the GasNet controller. By contrast other mechanisms may favour NoGas classes; here we counter this argument through extending the re-evolution analysis to a random sample of forty previously evolved GasNet and NoGas controllers of 100% fitness.

	Double speed		Quarter speed	
	GasNet	NoGas	GasNet	NoGas
Number of runs	200	200	200	200
Mean evaluated fitness (σ)	0.27 (0.13)	0.26 (0.18)	0.35 (0.27)	0.29 (0.19)
Mean re-evolution generations (σ)	107 (190)**	240 (363)	108 (229)	116 (252)
Median re-evolution generations	36*	49	13**	21

Table 3: Data for a sample of twenty GasNet and twenty NoGas controllers, re-evolved in two modified environments. The robot motors are set to double-speed and quarter-speeds respectively, and the two controllers evaluated 100 times for fitness, then used to seed the initial populations for the evolutionary algorithm until 100% fitness controllers were produced (10 runs were performed for each controller on each condition). The evaluated fitnesses, and mean, median and standard deviation of the number of generations of re-evolution required to reach 100% fitness controllers are shown, with significant differences between the GasNet and NoGas controllers highlighted (both parametric T-tests and non-parametric Mann-Whitney U tests were performed $^{*}p < 0.05,^{**} p < 0.01$).

The forty controllers were used to seed the initial populations for the distributed evolutionary algorithm, which was run until controllers once more showed 100% fitness, with fitness evaluated in the same double- and quarter-speed environments described in the previous section. Table 3 shows the results for the two conditions, averaged over ten evolutionary runs of each of the forty controllers. The results are not as striking as those from the functionally equivalent controllers, lending some weight to the hypothesis that the previous analysis unfairly favoured the GasNet mechanism. However, the GasNet controllers still showed significantly faster re-evolution than the NoGas controllers. In the double speed environment, both samples of controllers fell to average fitnesses of 0.26, but the GasNet controllers on average re-evolved in 107 generations compared with 240 generations for the NoGas controllers. In the quarter speed environment, the differences are much smaller, with comparable mean numbers of generations for re-evolution, however there is evidence of faster evolution from the median numbers of generations. Thus from our sample of GasNet controllers, we also see evidence of significantly faster re-evolution to modified environments; the GasNets are more tunable.

In the next section, we investigate the hypothesis further, using a more abstract fitness evaluation. We evolve GasNet and NoGas networks for a central pattern generation problem, in which the output of a single node is evaluated against a test pattern. We then use successfully evolved networks to seed the initial populations for a re-evolution environment where fitness is evaluated against a different test pattern.

11 Evolving central pattern generator networks

In this section, we test the GasNet and NoGas classes further in a central pattern generation (CPG) experiment. We evolve fully connected GasNet and NoGas networks, with output tested against some required test pattern, then re-evolve successful controllers against different test patterns. We argue that the increased evolutionary speed seen for the GasNet class over the NoGas class on both the original evolution and the re-evolution experiments lends support to the hypothesis that the GasNet class is more tunable to the characteristics of the environment, which in this case correspond to the desired pattern output.

11.1 The central pattern generation network

The networks used in the CPG experiment consisted of four fully-connected nodes, including recurrent connections (other size networks were also investigated, with comparable results). Connection weights between the four nodes were genetically specified, and were constrained to lie in the range $\{-1, 1\}$. Each node received a genetically specified fixed bias input, and the same tanh input-output transfer function was used as in previous experiments:

$$O_i^t = \tanh\left[K_i^t\left(\sum_{j\in C_i} w_{ji} O_j^{t-1}\right) + b_i\right] \tag{37}$$

where O_i^t is the ith node activity at time-point t, K_i^t the transfer function parameter, fixed for the NoGas networks, but able to vary for the GasNet networks, w_{ji} the connection weight from node j to node i, and b_i the bias input to node i.

Gas diffusion and modulation occurred exactly as described for the robot visual discrimination problem in section 2. Each node had a set x, y position in the gas diffusion plane, and was able to emit one of two gases, respectively increasing or decreasing the transfer function parameter K_i^t of nearby nodes. However, it should be emphasised that in the CPG networks the electrical architecture, in other words the pattern of synaptic connections between the nodes, was not specified arbitrarily in the gas diffusion plane, but specified directly on the genotype in terms of the weights between nodes. Figure 21 shows the network setup.

The NoGas genotype consisted of 18 integers in the range $[0, 99]$, encoding the 4 node biases b_i, 4 node transfer parameters K_i, and 10 connection weights w_{ji}. The GasNet genotype consisted of the NoGas genotype plus an extra six parameters per node for the gas diffusion parameters (the type of gas emitted $< CE >$, the conditions under which gas emission occurs $< TE >$, the radius of gas emission r, the gas build-up parameter s, and the x, y co-ordinates of the node in the 2D gas plane), thus the GasNet genotype consists of 42 integers in the range $[0, 99]$.

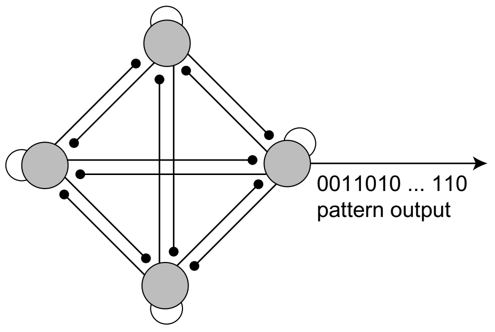


Figure 21: The fully-connected central pattern generator experiment networks. Each node provides output to, and receives input from, every other node in the network (including itself). Weights are specified by the genotype over the range $\{-1, 1\}$, and the final output pattern is specified as the output of the first node in the network.

11.2 Fitness evaluation

Four simple test-patterns were used, all consisting of a number of ones followed by a number of zeros, repeating for the entire 200 time-steps of fitness evaluation. The **Ten:Four** pattern consisted of a repeated block of ten ones followed by four zeros. Similarly **Eleven:Five**, **Eleven:Seven**, and **Seven:Five** consisted of repeated blocks of the relevant numbers of ones and zeros. Other patterns showed comparable results.

Networks were evaluated over 200 time-steps, with fitness evaluated on the output of the first node in the network. The pattern output of the network was specified as *one* if this node had positive activity, *zero* otherwise. The full pattern over the 200 time-steps was compared with the test pattern, receiving positive score for correct pattern output on each time-step. The scores allocated for each correct output were weighted to give equal weighting to ones and zeros, with fitness scaled to the range $[0, 1]$. In other words, each output one that matched the required test pattern scored inversely proportional to the total number of ones in the test pattern, while similarly each zero that matched the required test pattern scored inversely proportional to the total number of zeros in the test pattern.

11.3 Evolutionary results

The distributed steady-state evolutionary algorithm described in section 5 was used. Two other evolutionary algorithms showed comparable results (a generational tournament algorithm and simulated annealing algorithm). Similarly, the mutation and recombination operators described in section 5 were used. We report results only from experiments using mutation rate $\mu = 8\%$, with no recombination.

Table 4 and figure 22 show the evolution results for the four test patterns. Results are given for the number of evaluations required before successful controllers of 100% fitness are evolved. For each of the test patterns, fifty evolutionary runs were carried out for the GasNet and NoGas classes. We see clearly that the GasNet class evolves successful pattern generators significantly faster than the corresponding NoGas class, typically in about half the number of evaluations.

In figure 23 we show the results for the re-evolution of successful controllers, when evaluated against a different test pattern from the original evolution. The fifty controllers for each of the four test patterns evolved above were used to seed the initial populations for the evolutionary algorithm, and 10 evolutionary runs carried out for each of the previously evolved controllers against each of the other three test patterns. Again we see that the GasNet class produces significantly faster evolution for each of the re-evolution experiments.

The results for the re-evolution of pattern generation networks show a number of cases where the number of evaluations required to re-evolve to a different test pattern is not significantly smaller than the original evolution (and even some cases where re-evolution is slower than the original evolution). However, in all cases the GasNet class shows significantly faster evolution and re-evolution of CPG networks than the NoGas class.

Pattern	Statistic	GasNet	NoGas
Seven:Five	Mean (σ)	16390 (34060)	33530 (45910)
	Median	2500	3260
Ten:Four	Mean (σ)	18000 (36080)	36000 (46230)
	Median	2120	5010
Eleven:Five	Mean (σ)	22410 (39100)	43830 (48060)
	Median	3070	5470
Eleven:Seven	Mean (σ)	13750 (32130)	31610 (45080)
	Median	1460	2980

Table 4: The number of evaluations required to evolve successful GasNet and NoGas networks, for the four experiments where fitness is evaluated over the four different test patterns: **Ten:Four**, **Eleven:Five**, **Eleven:Seven**, and **Seven:Five**.

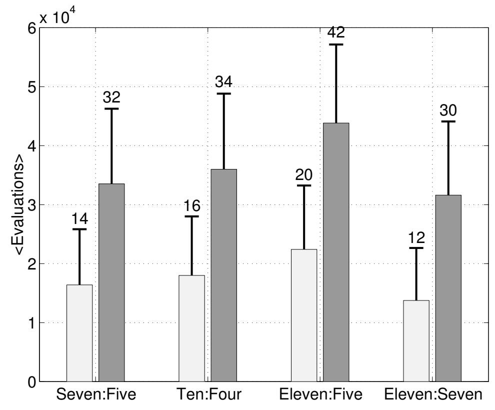


Figure 22: The number of evaluations required to evolve successful GasNet and NoGas networks, for the four experiments where fitness is evaluated over the four different test patterns: **Ten:Four**, **Eleven:Five**, **Eleven:Seven**, and **Seven:Five**. Mean data shown, with the error bars representing 95% confidence intervals. The numbers above the error bars show the number of runs not finishing. GasNet data is given by the light grey bars, NoGas data by the dark grey bars.

So again, the results support our hypothesis that the GasNet class is well suited to being tuned to the specific properties of the environment when compared with the NoGas class. In the next section we draw together the various experiments carried out in this paper.

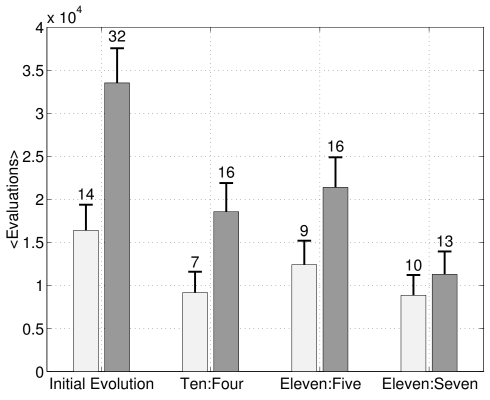
12 Summary

The detailed analysis of a number of GasNet and NoGas controllers allowed us to frame two hypotheses regarding the suitability of the GasNet class to robot control.

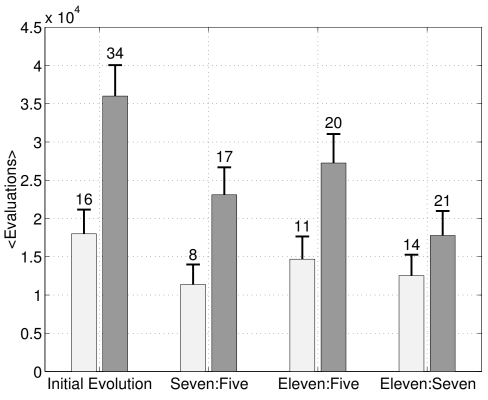
First, the ability to both produce and modify central pattern generation output was seen to be central to a number of evolved control solutions. This seems surprising. We are not investigating such behaviours as walking and swimming gaits, or rhythmic feeding, where behaviour is often based on central pattern generation. Our visual shape discrimination task might not at first sight appear to be related to such pattern generation. However, a number of GasNet controllers were seen to use pattern generation subnetworks in the final evolved behaviour.

Second, the ability to switch between dynamical states dependent on external input, and the ability to mediate this switch over a number of time-steps was seen to be extremely useful both in behaviour generation and filtering environmental noise. From analysis of the functionally equivalent GasNet and NoGas controllers we argued that the kinds of timing mechanisms able to mediate such behaviour switching and noise filtration were much easier using the GasNet class.

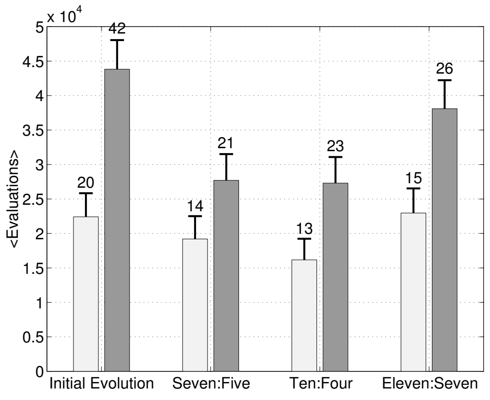
The twin hypotheses that GasNet classes were more amenable to both the development and tuning of pattern generation and the development and tuning of switching mechanisms were supported by the



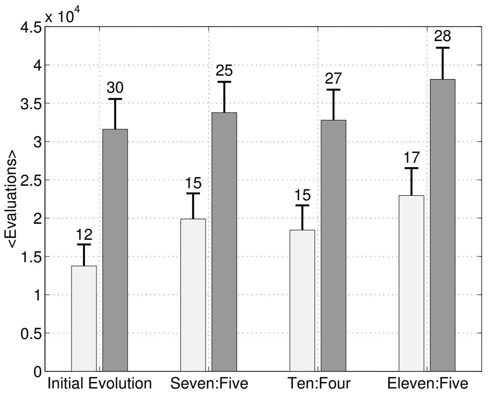
(a) **Seven:Five** pattern



(b) **Ten:Four** pattern



(c) **Eleven:Five** pattern



(d) **Eleven:Seven** pattern

Figure 23: The number of evaluations required to re-evolve successful GasNet and NoGas networks for central pattern generation, when evaluated on different patterns to the initial evolution. Controllers originally evolved on one of the four patterns were re-evolved on the other three patterns. Mean data shown, with the error bars representing 95% confidence intervals. The numbers above the error bars show the number of runs not finishing. GasNet data is given by the light grey bars, NoGas data by the dark grey bars.

re-evolution studies. We saw that the functionally equivalent GasNet controller was much easier to tune to a modified environment than the corresponding NoGas controller. To a lesser extent, although still significant, this same re-evolution tunability was seen over a large sample of previously evolved controllers. We further supported the pattern generation hypothesis through the evolution and re-evolution of GasNet and NoGas controllers on a simplified pattern output task. Again we saw that not only were GasNet solutions much easier to evolve to the original pattern output, but that the GasNet networks were faster to re-evolve to a different pattern than the NoGas networks.

So can we draw any conclusions from this work on what makes an evolvable network class for the visual discrimination problem? The simple answer is yes. The key feature of the GasNets seen to be useful on this task is the ability to smoothly adapt to the temporal characteristics of the environment. This encompasses the initial development and subsequent tuning of the controllers to the detailed properties of the robot and environment in which it finds itself. Included in this ability to smoothly adapt to the temporal characteristics of the environment, is the ability to generate a rich variety of temporal patterns, through the interaction of the gas diffusion mechanism and the electrical synaptic mechanism.

The different time-scales over which these two mechanisms operate was seen to be crucial to this pattern generation.

In the final section, we conclude with discussion of *temporally adaptive* networks.

13 Discussion: Temporally adaptive networks

One feature common to many of the neural network classes used for generating adaptive behaviour, is the incorporation of *time*. Few evolutionary robotics practitioners rely on feedforward networks consisting of nodes that retain no activity over time, with most using network classes that are able to access some form of *memory* through either recurrency or retaining some level of node activity. In simple terms, the current behaviour of the robot is not solely a function of the current sensory input, but a function of the current and previous sensory inputs, and the current and previous internal network activity. In principle this allows a much richer range of dynamical behaviours to be generated; the robot is not merely *reacting* to the environment, but *interacting* with the environment.

From the analysis of the operation of GasNet and NoGas robot controllers in the visual discrimination task, we see that the effect of time is crucial to the development of robot control solutions, *even for a task in which timing might not be thought to play any role.* Although the NoGas controllers potentially have access to previous sensory input and activity through the arbitrarily recurrent network architecture, the ability of the GasNet controllers to adapt to the particular characteristics of the environment was seen to be extremely powerful in generating pattern output, and mediating behaviourial switching. Such temporal adaptation enabled the evolutionary process to more easily tune controllers to the particular environment, switch between dynamical states on the basis of sensory input over extended time periods, and efficiently filter out environmental noise.

With this result we have provided some support for the intuition of many evolutionary robotics practitioners, that robot controllers operating in the real world must incorporate temporal structure, and that the evolutionary process must be able to easily adapt that structure[3]. On this fundamental principle of temporal adaptivity, the GasNet neural network class falls squarely into a much larger class containing among others continuous time recurrent networks (Beer and Gallagher, 1992), pulsed neural networks (Maass and Bishop, 1999), and networks with time-lagged synaptic activity (Harvey, 1993). However, we argue that simple recurrent networks such as the NoGas are not members of this class; although activity is retained over time, it is not straight-forward for the evolutionary process to temporally modify the activity of the network[4]. It seems plausible that if we are to further develop evolvable artificial neural network classes for robot control, the starting point must be from within this class of temporally adaptive networks.

Acknowledgements

The authors would like to thank Andy Philippides and the members of the Centre for Computational Neuroscience and Robotics (http://www.cogs.susx.ac.uk/ccnr/) for constructive comments and discussion. We would also like to thank the Sussex High Performance Computing Initiative (http://www.hpc.sussex.ac.uk/) for computing support. TS is funded by a British Telecom Future Technologies Group (http://innovate.bt.com/projects/ftg.htm) sponsored Biotechnology and Biology Science Research Council Case award.

[3]For example, Harvey (1993) makes the point that "... in environments where physical events have natural timescales, the dimension of time is not an optional extra, but fundamental." Similarly, Gallagher and Beer (1999) state that "... nontrivial behavior requires the integration of experiences across time and the ability to initiate actions independent of an agent's immediate circumstances."

[4]Of course the architecture of the NoGas networks may be arbitrarily modified, however it does not seem an efficient method by which to carry out such temporal adaptation.

References

Beer, R. (1990). *Intelligence as Adaptive Behaviour: An Experiment in Computational Neuroethology.* Academic Press, Cambridge, Massachusetts.

Beer, R. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behaviour*, 3(4):469–509.

Beer, R., Chiel, H., and Gallagher, J. (1999). Evolution and analysis of model CPGs for walking II. general principles and individual variability. *Journal of Computational Neuroscience*, 7(2):119–147.

Beer, R. D. and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behaviour. *Adaptive Behaviour*, 1:94–110.

Calvatti, A. and Beer, R. (2001). Analysis of a distributed model of leg coordination I. individual coordination mechanisms. *Biological Cybernetics*, 82:197–206.

Chiel, H., Beer, R., and Gallagher, J. (1999). Evolution and analysis of model CPGs for walking I. Dynamical Modules. *Journal of Computational Neuroscience*, 7(2):99–118.

Devaney, R. (1989). *Chaos, Fractals and Dynamics: Computer Experiments in Mathematics.* Addison-Wesley, Redwood, California.

Gallagher, J. C. and Beer, R. D. (1999). Evolution and analysis of dynamical neural networks for agents integrating vision, locomotion, and short-term memory. In Banzhaf, W. and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99).* Morgan Kaufmann, San Mateo, California.

Gavalas, G. R. (1968). *Nonlinear Differential Equations of Chemically Reacting Systems.* Springer, Berlin.

Goldstein, H. (1980). *Classical Mechanics.* Addison-Wesley, Redwood, California, 2nd edition.

Harvey, I. (1993). *The Artificial Evolution of Adaptive Behaviour.* PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, UK.

Husbands, P. (1998). Evolving robot behaviours with diffusing gas networks. In Husbands, P. and Meyer, J.-A., editors, *Evolutionary Robotics: First European Workshop, EvoRobot98*, pages 71–86. Springer, Berlin.

Husbands, P., Smith, T., Jakobi, N., and O'Shea, M. (1998). Better living through chemistry: Evolving GasNets for robot control. *Connection Science*, 10(3-4):185–210.

Jakobi, N. (1998a). Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behaviour*, 6:325–368.

Jakobi, N. (1998b). *Minimal Simulations for Evolutionary Robotics.* PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, UK.

Lotka, A. (1925). *Elements of Physical Biology.* Williams and Wilkins. Reprinted as *Elements of Mathematical Biology* (1956), Dover, New York.

Maass, W. and Bishop, C., editors (1999). *Pulsed Neural Networks.* MIT Press, Cambridge, Massachusetts.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, Massachusetts.

Rosen, R. (1970). *Dynamical System Theory in Biology*, volume I. Stability theory and its applications. John Wiley and Sons, New York.

Rubinow, S. (1975). *Introduction to Mathematical Biology.* John Wiley and Sons, New York.

Sandefur, J. (1990). *Discrete Dynamical Systems: Theory and Applications.* Oxford University Press, Oxford, UK.

Smith, T., Husbands, P., Layzell, P., and O'Shea, M. (2002a). Fitness landscapes and evolvability. *Evolutionary Computation*, 10(1):1–34.

Smith, T., Husbands, P., and O'Shea, M. (2001a). Neutral networks and evolvability with complex genotype-phenotype mapping. In Kelemen, J. and Sosík, P., editors, *Advances in Artificial Life, 6th European Conference (ECAL'2001)*, pages 272–281. Springer, Berlin.

Smith, T., Husbands, P., and O'Shea, M. (2001b). Not measuring evolvability: Initial exploration of an evolutionary robotics search space. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC'2001)*, pages 9–16. IEEE Press, Piscataway, New Jersey.

Smith, T., Husbands, P., and O'Shea, M. (2002b). Local evolvability of statistically neutral GasNet robot controllers. *Biosystems.* In press.

Smith, T. M. C. (2002). *The Evolvability of Artificial Neural Networks for Robot Control.* PhD thesis, School of Biological Sciences, University of Sussex, UK. Submitted.

Volterra, V. (1926). Fluctuations in the abundance of a species considered mathematically. *Nature*, 118:558–560. Reprinted in Real, L. and Brown, J.H. (1991). *Foundations of Ecology: Classic Papers With Commentaries.* University of Chicago Press, Chicago, Illinois.