# Neuroevolution of Agents Capable of Reactive and Deliberative Behaviours in Novel and Dynamic Environments

Edward Robinson, Timothy Ellis and Alastair Channon

School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK
edd@edd-robinson.net, t.s.ellis@cs.bham.ac.uk, alastair@channon.net

**Abstract.** Both reactive and deliberative qualities are essential for a good action selection mechanism. We present a model that embodies a hybrid of two very different neural network architectures inside an animat: one that controls their high level deliberative behaviours, such as the selection of sub-goals, and one that provides reactive and navigational capabilities. Animats using this model are evolved in novel and dynamic environments, on complex tasks requiring deliberative behaviours: tasks that cannot be solved by reactive mechanisms alone and which would traditionally have their solutions formulated in terms of search-based planning. Significantly, no a priori information is given to the animats, making explicit forward search through state transitions impossible. The complexity of the problem means that animats must first learn to solve sub-goals without receiving any reward. Animats are shown increasingly complex versions of the task, with the results demonstrating, for the first time, incremental neuro-evolutionary learning on such tasks.

**Keywords:** Artificial Life, Neural Networks, Incremental Evolution, Reactive and Deliberative Systems, Novel and Dynamic Environments.

## 1 Introduction

In this paper we present work showing animats that use neural networks to display high level deliberative decision making whilst retaining reactive qualities. Deliberative planning has traditional roots in "Good Old Fashioned Artificial Intelligence" (GOFAI) as a search-based method for the design of behaviour systems. There are issues however with its application in dynamic and novel environments. Reactive models of action selection on the other hand can be very successful in dealing with unpredictable and dynamic environments. However, since these systems generally have only a short look-ahead the individual complexity of behaviour that can emerge is limited. Both reactive and deliberative qualities are essential for a good action selection mechanism: deliberative mechanisms for long term goal seeking and reactive capabilities for dealing with unforeseen events [1, 2].

A complex problem has been designed to demonstrate our model, which we have called the 'river-crossing task' or RC task. In this problem an animat must

cross a river by building a bridge made out of stones collected from locations in a 2D grid-world environment. Importantly, animats are evolved to solve this problem without any task-specific information. Animats are embodied with two very different neural networks. The first acts as a deliberative style decision network: it makes high level choices about the sub-goals that need to be achieved, given current internal and external states. The actions that the animats choose may for example be 'head to the nearest stone', 'avoid the traps' or 'head to the resource'. Once an animat has made a decision, the second (reactive) neural network acts as a navigation tool, taking care of low level actions, such as which direction to move in next. In the RC environment there are several classes of objects that the animats can interact with. *Grass* objects take up most environmental space; the animat can place other objects onto them. *Stones* are movable objects: they can be picked up and dropped on grass or water. If one is dropped on water then the water object is converted into a grass object. *Water* objects are dangerous. If an animat moves onto one and does not place a stone down then the animat drowns. *Traps* are lethal to animats, which die if they move onto one. *Resource* objects offer rewards to animats, if they can reach them. *None of this information is given a priori to the animats, ruling out the possibility of explicit forward search through state transitions.*

Payton [3] used gradient fields to represent the state-space of a problem and as an internalised plan. Unlike more traditional search based models, gradient fields can be generated efficiently, and do not suffer from the same local-minima problems as other wave based mechanisms such as potential fields [4]. However, the gradient fields approach does not deal well with changing environments and so is often coupled with a Brooks inspired [5] subsumption architecture [3]. Another issue with gradient fields is that they have to be laboriously constructed.

We describe in the next section a biologically inspired gradient based model which does not suffer from local minima nor any of the other problems associated with other gradient based models. It is computationally efficient and simple to initialise. We also describe a decision network which is designed to allow animats to manipulate the navigation model. We show experimental results in section 3 and conclude in Section 4.

## 2 The Model

The movements of the animats in the environment are dictated by a shunting model introduced in [6, 7]. Yang and Meng were interested in motion planning models that could react quickly in real-time, allowing a robot or robot-manipulator to perform collision-free motion planning.

Neural networks have been used extensively and successfully for robot control problems. Often controllers specify a robot's behaviour based upon sensory input from the environment; this makes them good for dynamic environments, which are likely to change continuously. The model in [6, 7] uses neural networks in a very different way. Instead of using the network to specify behaviour by, for example, mapping the actuators of the robot to the outputs of the network, the

network's activation landscape itself directly specifies the robot's movements through the environment.

Their model consists of a neural network composed of an n-dimensional lattice of neurons $\ell$, where each neuron represents a possible state of the system. Therefore any system that can be fully described by a set of discrete states can be represented. This is referred to as the 'configuration space' of the robot [8]. In the case of the simulated robot in their studies, the configuration space was the discretised 2D Cartesian workspace. Each neuron is connected to a subset of the lattice—$\Re_i \subset \ell$. This subset is called the receptive field, and represents all the states that are reachable from the current state. It is useful to note that if, as in Yang and Meng's simulated robot example, the state space is simply the 2D coordinates of the environment that we wish the agent to navigate, there is always a simple one-to-one relationship between neurons and locations.

The transition function used to specify inter-neuron dynamics is based on the 'shunting equation', inspired by Hodgkin and Huxley [9] and Grossberg [10]. Yang and Meng designed two versions of this transition function: one which helped to control activity saturation in the network, and a simpler one which did not. In our study we found that the more elaborate transition function was not necessary, since out model did not develop saturation problems; the function is shown in equation 1.

$$\frac{dx_i}{dt} = -\mathrm{A}x_i + \mathrm{I}_i + \sum_{j=1}^{k} w_{ij}[x_j]^+ \ . \tag{1}$$

Alpha (A) represents the passive decay rate, which determines the degree to which each neuron's activity diminishes towards an idle state. The function $[x]^+$ can be described as $max(0, x)$. The connection weight (or synapse strength) $w_{i,j}$ between neurons is simply specified as the Euclidean distance between the cell and its neighbour within the receptive field. $k$ is the receptive field size and is set to 8 to represent the direct neighbours in a 2d grid-world environment. Iota (I) is equal to $E$ in the case of a target, and $-E$ for an obstacle, where $E$ is a large integer.

Once the network is configured, and the targets and obstacles established, neural activity can be used to navigate a robot by gradient ascent. At each time-step the robot looks at the level of activity in each grid-cell that it is connected to (its neighbours), because they are all the states that it can reach, and picks the one with the highest value. As a result of the network's dynamics, positive activity entered at neurons that map to targets propagates through the network, whilst negative activity that is inputted into neurons mapping to obstacles cannot propagate globally. Due to the nature of the leaky integrator function in the model, the activity contribution from a target decreases with distance from that target source, leaving a trail of activity back to the target.

This system therefore allows a robot to navigate a dynamic environment, avoiding obstacles and heading towards targets. Using the shunting network to control movement throughout the environment means that high level actions

such as 'head to the target whilst avoiding obstacles' can be carried out flawlessly and quickly. The highly dynamic nature of the network means that when the environment changes (e.g. an obstacle moves), a new activity path can be generated quickly. In the RC task there are four different classes of objects (Resource, Stone, Water and Trap) that have Iota values associated with them. Grass states are considered empty states to allow activity to flow through the environment. For target acquisition and obstacle avoidance there are two types of value: positive and negative. In our implementation of the model however, we allow a class to be specified with no value at all. Setting Iota to 0 for a class means that no external activity will be inputted into any of the neurons in that class. The animat will ignore the object present at that location, and may or may not pass over it while moving through the environment.

### 2.1 The Decision Network

The outputs of the decision network are used to set the Iota values for the object classes. Using this network, the animat can manipulate the activity landscape in the shunting network in a way that allows it to string together multiple actions in parallel to create more complex behaviours.

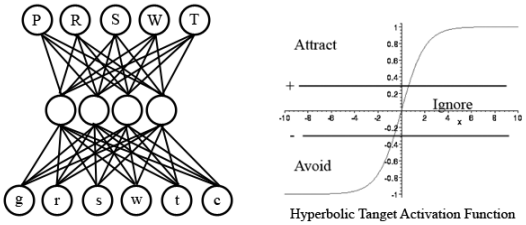


**Fig. 1.** The decision network controller. The output neurons are P = pick up/put down; R = resource; S = stone; W = water; T = trap. The input neurons are g = grass; r = resource; s = stone; w = water; t = trap.

The decision network is a feed-forward neural network with a single hidden layer of four neurons (figure 1). The input layer represents the current state of the animat, or more precisely, the object class situated on and the carrying status of the animat. The inputs are single values of 1 or 0 and they feed through weighted links into the hidden layer neurons, where tan activation functions are applied to the summed input values. The neurons in the output layer represent Iota values for the object classes needing them (four). Output neurons have tan activation functions and two fixed thresholds. Neurons with activations over 0.3 or under -0.3, after being processed through the activation function, output 1 and -1 respectively. Any activation values in the range $[-0.3; 0.3]$ resolve to 0.

Output neurons then, have three possible outputs: -1, 0 or 1. The Iota values of all the objects (except grass) in the environment are set based upon the

output of the decision network neuron associated with that class. If a neuron has a negative output then all of the objects in that class will have a negative Iota value (-15 in our simulations). Similarly, a positive neuron output sets all the objects in that class with positive Iota values (+15 in our simulations). Finally, a neuron with an output of zero sets the objects in that class with Iota values of 0. Having an Iota value of 0 means that the objects have no external activity inputted into them: their activation values in the shunting network will be solely based upon those of their neighbours. To get a clear understanding of the purpose and ability of the decision network, two examples of resulting shunting network activity landscapes are shown in figure 2. Each landscape is different because of the Iota values of object classes. In the first landscape for example, the Iota values result in positive activity propagating from the resource neuron, through the rest of the network attracting the animats, while negative activity repels animats from the traps.

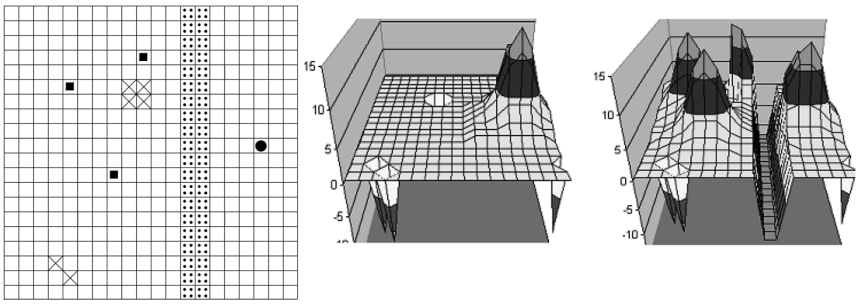


**Fig. 2.** A typical environment (left) and two activity landscapes (middle, right). Iota values in the first (middle) landscape a are: resource = 15; stone = 0; water = 0; trap = -15. In the right landscape b, resource = 15; stone = 15; water = -15; trap = -15. Environment legend: stone = small square; resource = circle; trap = cross; water = four dots.

Landscape b represents the same environment, but with different Iota values for object classes, and so the animat acts differently. Activity from the resource would no longer be able to propagate through the river, but since the stones have positive activities, the animat moves to the nearest one (still avoiding traps). One of the output neurons on the decision network is not used to provide an Iota value for motion. Instead, its output is used to make a decision about whether or not to pick up or drop stones: the other actions an animat can take in our system. If the output is positive then the animat will attempt to pick up whatever object it is currently situated on in the grid-world. If negative then the animat will attempt to drop an object.

### 2.2 Evolution of Decision Networks

We used a steady-state genetic algorithm (GA) to search the weight-space, with fitness based upon animat performance in evaluation. Tournament selection was used for each iteration, with three animats evaluated and the worst performer replaced by a new offspring created from a combination of the other two.

An animat has a set of chromosomes: one for each neuron in its decision network. Each chromosome contains the floating point values for the weights of its neuron's input connections. For each of an offspring's (output and hidden) neurons, there is a probability $P_{whole} = 0.95$ that the corresponding chromosome will be inherited from just one parent; which parent this is copied from is then chosen at random. Otherwise (probability $P_{mix} = 0.05$) the offspring will instead inherit a new chromosome whose genes are a mixture of both parents' versions of the same chromosome, combined by single-point crossover. Finally, each weight has a probability of $P_{mut} = 0.001$ of having a mutation value from $N(0, 0.4)$ added to it. All mutations are bounded to within $[-1; 1]$.

## 3 Experimentation

For each experiment, a population of 250 animats is initialised with random chromosomes, hence with random decision network weights. Animats are evaluated singularly on the RC task in a 20x20 cell grid-world. They are only rewarded when they reach a resource state: if they fail to reach it then their fitness is zero. Similarly, if the animat performs any action that leads it to either drowning in the river or moving onto a trap, the evaluation task ends and the animat's fitness is zero. Animats are placed randomly to the left of the river and stones and traps are distributed randomly inside the environment. The animat's decision network inputs are updated whenever the animat's state changes; then the shunting model updates the activity landscape with new Iota values and the animat moves to the neighbouring cell with the highest activity. If the pick up/put down neuron is activated then the animat will attempt to pick up/put down whatever it is on or carrying. At each iteration of the GA three animats are randomly selected. Each animat is evaluated on the same randomly generated environment to ensure fairness; a new environment is randomly generated in the next iteration of the GA.

First we tested animats by evaluating them on one environment that contained a river two cells wide. Due to the problem complexity, and because the animats had to learn to solve sub-problems before even receiving any fitness, all of the population's individuals had zero fitness and search was random. To overcome this problem we exposed the animats to three trials of increasing difficulty. Figure 3 shows three randomly generated versions of the three maps used. The randomly positioned objects are the traps and stones: the same river and resource locations are always used. The first map that the animats are tested on already has a completed bridge. To solve this task the animats simply have to learn to avoid traps and the river and get to the resource. To solve the second and third maps an animat has to build a bridge, with the bridge needing to be

smaller (one cell wide) for the second map than the third (two cells wide)[1]. This system provides a route for incremental evolutionary learning and can be seen as a simplification of a more complex general world-environment in which animats encounter opportunities for reward, of varying degrees of difficulty.

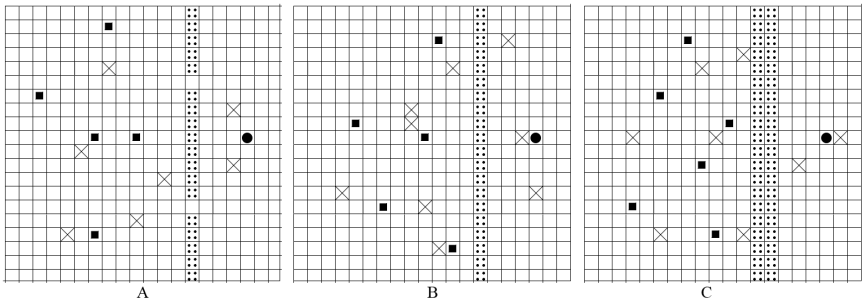


**Fig. 3.** Examples of the three environments that animats are evaluated in.

### 3.1 Experimental Results

For each task the animats solved out of the three they received a score of 100. If they failed a task they were awarded a score of 0 only for the task they failed. In each simulation, data showing how many versions of each task had been solved in the previous 250 tournaments was collected in 250 tournament intervals. Since there were three animats taking part in each tournament, the maximum number of tasks solved in this period, for each map, was 750. Table 1 shows results from 15 simulations. Once 80% (ie. 600) of animats tested per iteration could solve all versions of the task they were shown, the simulation was stopped.

In all simulations the animats quickly evolved behaviours that could be used to solve the simplest map, where they needed only to know how to avoid dangers like water and traps. They didn't need to interact with stones to solve this task, making the process simple. The next two maps were substantially harder to solve, and the time taken to solve them reflects this. Results showed that contrary to previous observations: the third map was not harder to solve than the second, even though it required a specific strategy. The second intermediate map could

[1] Observations of the simulation showed that animats were solving a map with a single celled river by picking up stones and dropping them on the river indiscriminately. They were attracted to both the resource and the river; since the river was closer, they would deposit a stone on the nearest river cell. Although once they had done this they could complete the task, because they were still attracted to the river they would often keep depositing stones there. Eventually they would deposit enough stones to create a gap so large that the activity from the resource attracted them enough for them to reach it. Using a deeper river stopped animats for being rewarded for developing this 'brute force' behaviour.

**Table 1.** The mean, best and worst number of tournaments needed for 80% of the animats evaluated in a 250-iteration period to have solved each map.

| Map | Mean | Best | Worst | Stdev |
|---|---|---|---|---|
| 1 | 5700 | 4000 | 8250 | 1203.4 |
| 2 | 99084.6 | 13000 | 437750 | 139542.4 |
| 3 | 99083.3 | 13000 | 437750 | 139544.4 |

be solved in two ways: the 'brute force' approach (above)[1] or the 'correct' way, described below. Since the third map could only be solved in the correct way, animats that learnt this behaviour also solved the second map in the same way; this accounts for the similar (usually identical) time taken to solve both harder maps. However, the exclusion of either of these maps in the learning process would have lead to sub-optimal behaviour evolving.

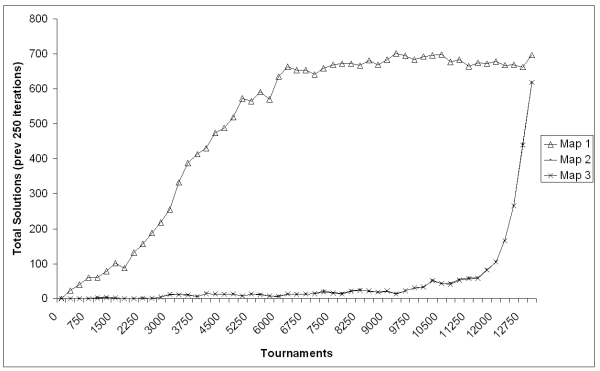


**Fig. 4.** Results from the simulation which solved all the tasks in the fastest time. Map 1 was learnt in around 6000 tournaments, while maps 2 and 3 took 13000 tournaments each.

The observed 'correct' behaviour for solving the harder maps can be described as follows: When the animats are on grass and not carrying anything they head to the nearest stone, whilst avoiding the river and traps. Once they reach a stone they pick it up; they are then situated on grass, but now carrying a stone. Next, they adjust the shunting model so they are attracted to the resource, ignoring the river/other stones and avoiding traps. Once they reach the river they deposit a stone; they are now back to being on grass and not carrying. If activity can propagate from the resource to the animat (because of a completed bridge) they head to the resource. Otherwise they return to a stone and repeat.

The RC problem is particularly difficult because of the lack of reward available leading to long periods of stasis between solving the easy and the harder maps. During this time the animats phenotypic behaviour does not change by

much; they keep solving the easy task and failing the harder ones. It is possible that selection pressures are preventing change from taking place gradually as one might expect. Animats beginning to learn to solve the harder tasks, for example by starting to pick up stones, may disrupt and forget previous behaviours causing them to fail the simple task - most likely due to sharing connections for the different behaviours [11]. Figure 4 shows the results of the simulation with the quickest time to solve all tasks, and although it is the fastest, the nature of the graph is the same for all runs: the simpler task is learnt quickly, then there is a stasis period until by chance animats are born that can solve the harder tasks. This advantageous behaviour then quickly propagates through the environment until the vast majority can solve all tasks.

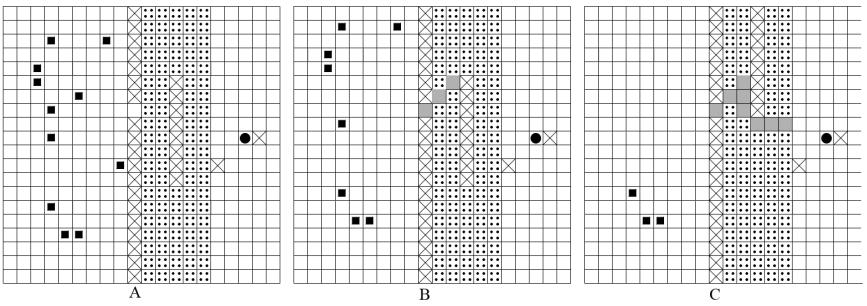


**Fig. 5.** A dynamic environment: In Part B the wall is about to move and block the animats path; instantly the animat starts building in a new direction and crosses successfully (Part C)

One advantage of the shunting model as shown in [6] was that it was highly reactive to a dynamic environment. We tested this ability in our system by evaluating animats that had been evolved to solve the three maps. Animats were shown the map in part A of Figure 5. The map has a larger river than they have been shown during evolution. Also, it has a movable wall of traps. The animat begins building a bridge through the shortest route of the river (part B); once the animat gets halfway across however the wall is moved, blocking the animats path (part C). Without hesitation the animat continues to build its bridge in the new required direction and navigates across the bridge.

## 4 Conclusions and Future Work

We have developed and presented a model that allows animats to develop complex behaviours such as building a bridge in the RC task, by using an incremental approach. Through the use of an adapted version of Yang and Meng's shunting model animats manipulate an activity landscape by utilising a decision network. This approach allows high level behaviours such as 'find a stone without drowning or falling into traps' to be carried out without any further requirements from

the animat other than the evolved desire to do so. Further, animats that are only shown the simpler three maps used in the evolutionary process can solve a novel and dynamic version of the task. Due to the lack of hard-coded constraints, this model could be used with many different environments without needing to make changes.

One draw-back of the current model is that it requires incremental versions of the task to be *shown* to the animat. Using a larger environment could allow these tasks to be situated in the same realm, but we chose to implement them as separate worlds. This approach leads to problems as more complex environments are constructed. To address this issue, future work will include methods for allowing the animats to generate intrinsic motivations, which has been shown to be imperative in mental development [12]. The intrinsic motivation will encourage an animat to solve sub-components of complex problems without the need of an outside critic guiding them.

## References


1. Tyrrell, T.: Computational Mechanisms for Action Selection. PhD thesis, University of Edinburgh (1993)
2. Benjamin, M.R.: Virtues and limitations of multifusion based action selection. In: Agents '00: The Fourth International Conference on Intelligent Agents. (2000) 23–24
3. Payton, D.W., Rosenblatt, J.K., Keirsey, D.M.: Plan guided reaction. IEEE Trans. on Systems, Man, and Cybernetics **20**(6) (1990) 1370–1382
4. Koren, Y., Borenstein, J.: Potential field methods and their inherent limitations for mobile robot navigation. In: IEEE Int. Conf. on Robotics and Automation. (1991) 1398–1404
5. Brooks, R.A.: A robust layered control system for a mobile robot. IEEE J. Robot. and Auto. **2**(3) (1986) 14–23
6. Yang, S.X., Meng, M.: An efficient neural network approach to dynamic robot motion planning. Neural Networks **13**(2) (2000) 143–148
7. Yang, S.X., Meng, M.: An efficient neural network method for real-time motion planning with safety consideration. Robotics and Autonomous Systems **32**(2-3) (2000) 115–128
8. Schultz, A.C.: Adapting the evaluation space to improve global learning. In Belew, R., Booker, L., eds.: Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kaufman (1991) 158–164
9. Hodgkin, A.L., Huxley, A.F.: A quantitative description of membrane current and its application to conduction and excitation in nerve. Journal of Physiology. **116** (1952) 500–544
10. Grossberg, S.: Nonlinear neural networks: Principles, mechanisms, and architectures. Neural Networks **1** (1988) 17–61
11. Seipone, T., Bullinaria, J.: The evolution of minimal catastrophic forgetting in neural systems. In Mahwah, N., ed.: Twenty-Seventh Annual Conference of the Cognitive Science Society, 1991-1996, Lawrence Erlbaum Associates (2005)
12. White, R.W.: Motivation reconsidered: The concept of competence. Psychological Review **66**(5) (1959) 297–333