Emergent Instinctive Behaviour in Evolved Neuro-inspired Agents

Gul Muhammad Khan, Julian F.Miller, and David M.Halliday

Electronics Department, University of York, York, YO10 5DD,UK
{gk502,jfm,dh20}@ohm.york.ac.uk
http://www.york.ac.uk

Abstract. We investigate the emergence of intelligent behaviour of an agent in the classic AI learning environment known as Wumpus World. The agent is given a 'brain' that is a new type of developmental neuro-inspired computational network. The neurons are defined by a genotype consisting of seven computational functions that represent electrical and developmental processes inside neuron. The computational functions are evolved using a form of Genetic Programming known as Cartesian Genetic Programming. The 'brain' that occurs by running the genetic programs has a highly dynamic morphology in which neurons grow, and die, and neurite branches together with synaptic connections form and change in response to situations encountered in the external environment. We present results and analyse characteristics of the model and find that the agents exhibit 'instinctive' behaviour and develop memory.

Key words: Generative and developmental approaches, Learning and memory, Artificial Neural Networks

1 Introduction

In this paper we propose a computational system capable of cognitive and adaptive behaviour inspired by neuroscience. In our view, emergent behaviour in a cognitive system arises as a consequence of biological development of the brain interacting with an external environment. This process is responsible for discovering the meaning of environmental signals and the creation of mental symbols. We have created a computational brain model inspired by neuroscience in which the brain of an agent develops during environmental interaction in a classic AI problem known as Wumpus world. In this world there are signals that indicate the presence of hazards and rewards. The agent acquires an understanding of the meaning of these signals and their relevance during its lifetime, while its brain is developing.

Recently there have been a number of grand challenges proposed in Computer Science both in the US and the UK. In the challenge The Architecture of Brain and Mind a number of sub-tasks were identified that would be needed to achieve the higher aims of a building a robot with the cognitive capabilities of

a human child. One of these is concerned with 'Bottom-up specification, design, and construction of a succession of computational models of brain function'. Our research is concerned with just such a computational model.

Artificial Neural Networks (ANNs), have long been seen as a computational equivalent of the brain, however they have largely ignored many aspects of biological neural systems, particularly neural development. It has been observed that "Mechanisms that build brains are just extensions of those that build the body" [6]. Memory is not a static process and the location and mechanisms responsible for remembered information is in constant (though, largely gradual) change [9]. The physical topology of the neural structures in the brain is constantly changing and is an integral part of its learning capability. Dendrites themselves should no longer be regarded as passive entities that simply collect and pass synaptic inputs to the soma, indeed Koch argues that "Dendritic trees enhance computational power" [4]. In most cases they shape and integrate these signals in complex ways [12]. Neurons communicate through synapses. Synapses are not simply the point of connection, they change the strength and shape of the signal over various time scales [2].

In this work we are attempting to obtain a computational equivalent of the biological *developmental* neuron. We have taken the view that, despite the underlying subtlety and complexity of neurons, their gross morphology and connectivity is sufficiently well understood to allow us to identify essential sub-systems and their inputs and outputs.

Since it appears, at present, impossible to design by hand, a computational equivalent of a neuron, we have chosen to use a method of automatic program evolution called Genetic Programming (GP) [5]. We use a well established and effective form of GP, known as Cartesian Genetic Programming (CGP)[14].

In the model we have idealised seven neural components which we have represented as CGP chromosomes encoding combinational digital circuits [3]. While this model in undeniably quite complex and involves many variables and parameters we feel this is justified by the evident enormous complexity of the brain. Our collection of chromosomes represent the essential aspects of the neuron: soma, dendrites and axon branches, and synaptic connections. The computational network that forms when the seven chromosomes are run grows a network of neurons, neurites and synapses that reflect its own internal dynamics and environmental interaction. We have tested and evaluated the capability of the system on the well known artificial intelligence problem known as wumpus world[10].

Unfortunately due to the limited number of pages (in LNCS format) it is impossible to describe all details of our proposed model and results.

Section 2 gives a description of CGP. Section 3 discusses important aspects of neuroscience and how we have incorporated them into our model. Section 4 describes the new computational model in detail. In section 5 we describe how we applied our computational model to the Wumpus world problem and presents our results and analysis. Finally, in section 6 we conclude the paper and describe future work.

2 Cartesian Genetic Programming (CGP)

CGP represents programs by directed acyclic graphs [8]. The genotype is a fixed length list of integers, which encodes the function of nodes and the connections of the directed graph. The CGP function nodes we have used are variants of multiplexers in which data inputs are either inverted or not. Typically CGP uses an evolutionary strategy of $1 + \lambda$, with λ set to 4. The parent, or elite, is preserved unaltered, whilst the offspring are generated by mutation of the parent. It is important to note that if two or more chromosomes achieve the highest fitness then *newest* (genetically) is always chosen [7].

3 Key features and biological basis for the CGPCN model

Features of biological neural systems that we think are important to include in our model are synaptic transmission, and synaptic and developmental plasticity. Signalling between biological neurons happens largely through synaptic transmission, where an action potential in the pre-synaptic neuron triggers a short lasting response in the post-synaptic neuron [11]. In our model signals received by a neuron through its dendrites are processed and a decision is taken whether to fire an action potential or not.

Synaptic plasticity refers to changes in synaptic transmission. It can involve changes in the postsynaptic excitability, which depends on the previous pattern of input [1]. This is incorporated in the CGPCN by introducing weights in neurite branches, and the soma. These can be adjusted by genetic processes during development of the network. Changes in the dendrite branch weight are analogous to the amplifications of a signal along the dendrite branch, whereas changes in the axon branch (or axo-synaptic) weight are analogous to changes at the pre-synaptic level and postsynaptic level (at synapse). Inclusion of a soma weight is justified by the observation that a fixed stimulus generates different responses in different neurones.

Through the introduction of a 'life cycle' chromosome, we have also incorporated developmental plasticity in our model. The branches can self-prune and can produce new branches to evolve an optimized network that depends on the complexity of the problem [13].

4 The CGP Computational Network (CGPCN)

This section describes in detail the structure of the CGPCN, along with the rules and evolutionary strategy used to run the system.

In the CGPCN neurons are placed randomly in a two dimensional spatial grid so that they are only aware of their spatial neighbours (as shown in figure 1). Each neuron is initially allocated a random number of dendrites, dendrite branches, one axon and a random number of axon branches. Neurons receive

information through dendrite branches, and transfer information through axon branches to neighbouring neurons. The dynamics of the network also change since branches may grow or shrink and move from one CGPCN grid point to another. They can produce new branches and can disappear, and neurons may die or produce new neurons. Axon branches transfer information only to dendrite branches in their proximity. Electrical potential is used for internal processing of neurons and communication between neurons and we represent it as an integer.

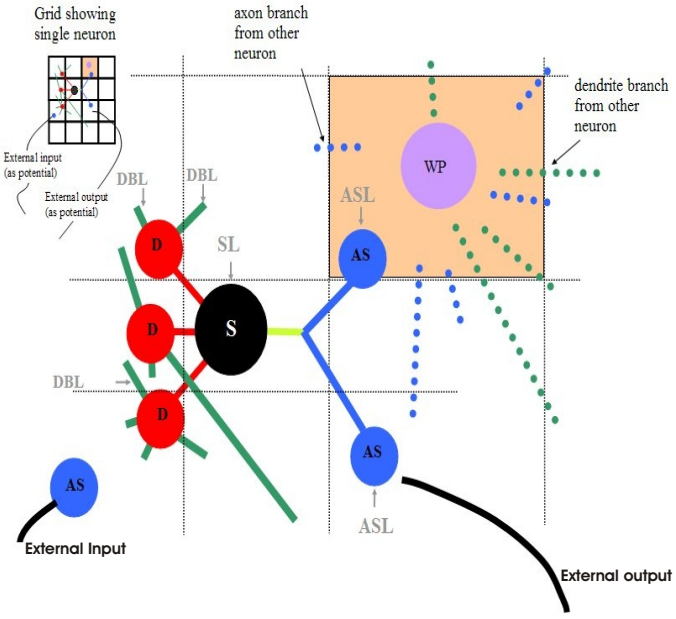

Fig. 1. On the top left a grid is shown containing a single neuron. The rest of the figure is an exploded view of the neuron is given. The neuron consists of seven evolved computational functions. Three are electrical and process a simulated potential in the dendrite (D), soma (S) and axo-synapse branch (AS). Three more are developmental in nature and are responsible for the life-cycle of neural components (shown in grey). They decide whether dendrite branches (DBL), soma (SL) and axo-synaptic branches (ASL) should die, change, or replicate. The remaining evolved computational function (WP) adjusts synaptic and dendritic weights and is used to decide the transfer of potential from a firing neuron (dashed line emanating from soma) to a neighbouring neuron

Health, Resistance, Weight and Statefactor

Four variables are incorporated into the CGPCN, representing either fundamental properties of the neurons (*health*, *resistance*, *weight*) or as an aid to computational efficiency (*statefactor*). The values of these variables are adjusted by the CGP programs. The *health* variable is used to govern replication and/or

death of dendrites and connections. The *resistance* variable controls growth and/or shrinkage of dendrites and axons. The *weight* is used in calculating the potentials in the network. Each soma has only two variables: *health* and *weight*. The *statefactor* is used as a parameter to reduce computational burden, by keeping some of the neurons and branches inactive for a number of cycles. Only when the *statefactor* is zero the neurons and branches are considered to be active and their corresponding program is run. The value of the *statefactor* is affected indirectly by CGP programs. The bio-inspiration for the *statefactor* is the fact that not all neurons and/or dendrites branches in the brain are actively involved in each process.

4.1 Inputs, Outputs and Information Processing in the Network

The external inputs (encoding a simulated potential) are applied to the CGPCN and processed by the axo-synapse program, AS. These are distributed in the network in a similar way to the axon branches of neurons. When AS is executed, it modifies the potentials of neighbouring active dendrite branches. We obtain output from the CGPCN via dendrite branches. These branches are updated by the AS program of neurons and after five cycles the potentials produced are averaged and this value (Fig 1) is used as the external output.

Information processing in the network starts by selecting the list of active neurons in the network and processing them in a random sequence. Each neuron take the signal from the dendrites by running the electrical dendrite branch program. The signals from dendrites are averaged and applied to the soma program along with the soma potential. The soma program is executed to get the final value of soma potential, which decides whether a neuron should fire an action potential or not. If soma fires, an action potential is transferred to other neurons through axosynaptic branches. The same process is repeated in all neurons.

4.2 CGP Model of Neuron

In our model neural functionality is divided into three major categories: electrical processing, life cycle and weight processing. These categories are described in detail below.

Electrical Processing

The electrical processing part is responsible for signal processing inside neurons and communication between neurons. It consists of dendrite branch, soma, and axo-synaptic branch electrical chromosomes.

The dendrite program D handles the interaction of dendrite branches belonging to a dendrite. It take active dendrite branch potentials and soma potential as input and the updates their values. The *Statefactor* is decreased if the update in potential is large and vice versa. If any of the branches are active (has its statefactor equal to zero), their life cycle program (DBL) is run, otherwise it continues processing the other dendrites.

The soma program S determines the final value of soma potential after receiving signals from all the dendrites. The processed potential of the soma is

then compared with the threshold potential of the soma, and a decision is made whether to fire an action potential or not. If it fires, it is kept inactive (refractory period) for a few cycles by changing its *statefactor*, the soma life cycle chromosome (SL) is run, and the firing potential is sent to the other neurons by running the AS programs in axon branches.

AS updates neighbouring dendrite branch potentials and the axo-synaptic potential. The *statefactor* of the axosynaptic branch is also updated. If the axo-synaptic branch is active its life cycle program (ASL) is executed.

After this the weight processing program (WP) is run which updates the *Weights* of neighbouring (branches sharing same grid square) branches. The processed axo-synaptic potential is assigned to the dendrite branch having the *largest* updated *Weight*.

Life Cycle of Neuron

This part is responsible for replication or death of neurons and neurite branches and also the growth and migration of neurite branches. It consists of three life cycle chromosomes responsible for the neuron and neurites development.

The two branch chromosomes update *Resistance* and *Health* of the branch. Change in *Resistance* of a neurite branch is used to decide whether it will grow, shrink, or stay at its current location. The updated value of neurite branch *Health* decides whether to produce offspring, to die, or remain as it was with an updated *Health* value. If the updated *Health* is above a certain threshold it is allowed to produce offspring and if below certain threshold, it is removed from the neurite. Producing offspring results in a new branch at the same CGPCN grid point connected to the same neurite (axon or dendrite).

The soma life cycle chromosome produces updated values of *Health* and *Weight* of the soma as output. The updated value of the soma *Health* decides whether the soma should produce offspring, should die or continue as it is. If the updated *Health* is above certain threshold it is allowed to produce offspring and if below a certain threshold it is removed from the network along with its neurites. If it produces offspring, then a new neuron is introduced into the network with a random number of neurites at a different random location.

5 Wumpus World

Wumpus world, is a variant of Gregory Yobs "Hunt the Wumpus" game [15]. It is an agent-based learning problem used as a testbed for various learning techniques in Artificial Intelligence [10]. The Wumpus world was originally presented by Michael Genesereth [10]. It consists of a two dimensional grid containing a number of pits, a wumpus (monster) and an agent [10]

5.1 Experimental setup

In our experiments we have slightly adapted the rules of the wumpus world. Namely, the agent encountering pits or wumpus is only weakened (thus reducing

its life), it is not killed and the agent does not have any arow to shoot the wumpus. Also glitter is on the side squares of the gold. These changes are introduced to facilitate the capacity of the agent to learn. The CGPCN learns everything from scratch and builds its own memory (including the meaning of signals, pits and the wumpus).

It is important to appreciate how difficult this problem is. The agents starts with a few neurons with random connections. Evolution must find a series of programs that build a computational network that is stable (doesn't lose all neurons or branches etc.). Secondly, it must find a way of processing infrequent environmental signals. Thirdly, it must navigate in this environment using some form of memory. Fourthly, it must confer goal-driven behaviour on the agent. This makes the wumpus world problem a challenging problem.

The wumpus world we used is a two dimensional grid (10x10), having ten pits, one wumpus and the gold. The location of the wumpus, gold and pits is chosen randomly. In the square containing the wumpus, gold, pits, and in a directly (not diagonally) adjacent square the agent can perceive stench, glitter and breeze, respectively. The input to the CGPCN is a potential of zero, if there is nothing on the square, a potential of 60 if encounters a pit, 120 if caught by wumpus and 200 if arrives at the square containing the gold. The system is arranged in such a way that the agent will receive input signals of different magnitudes depending on the direction that the agent perceives the signal. The agent's CGPCN can perceive the direction of a breeze through the magnitude of the breeze potential. We chose the following values to represent these directions: north of the pit is 40, east 50, south 70, and 80 for west. Similarly the stench signal has a direction which is represented by different magnitudes of potentials as follows: for north 100, for east 110, for south 130 and finally 140 for west of wumpus. Also it receives 180, 190, 210 and 220 from north, east, south and west of gold on for glitter. An agent detects that it is on the home square via a special input signal (255 is the maximum value of potential). In all the other locations which are safe the input potential is zero. The agent can always perceive *only one* signal on a grid square, even if there are more than one. The priority of the signals is in the following order: wumpus, gold, pit, stench, glitter, breeze. The agent is assigned a quantity called energy, which has an initial value of 200 units. If an agent is caught by the wumpus its energy level is reduced by 60%, if it encounters a pit its energy level is reduced by 10 units, if it gets the gold its energy level is increased by 50 units, and on arriving home the agents ceases to exist. For each single move the agent's energy level is reduced by 1 unit, so if the agent just oscillates in the environment and does not move around and acquire energy through solving tasks it will run out of energy and die.

The fitness of an agent is calculated according to its ability to complete learning tasks. It is accumulated over the period that the agents energy level is greater than zero (or before it returns home). The fitness value, which is used in the evolutionary scheme, is accumulated while the agent's energy is greater than zero as follows:

- For each move, the fitness is increased by one. This is done, to encourage the agents to have 'brain' that remains active and does not die.
- If the agent returns home without the gold, its fitness is increased by 200.
- If the agent obtains the gold, its fitness is increased by 1000.
- If the agent returns home with the gold, its fitness is increased by 2000.

When the experiment starts, the agent takes its input from the wumpus world grid square. This input is applied to the computational network of the agent and is processed by input axosynapses. The network is then run for five cycles (one step). During this process it updates the potentials of the output dendrite branches which act as the output of the CGPCN. After the step is complete the updated potentials of all output dendrite branches are noted and averaged. The value of this average potential decides the direction of movement for the agent. If there is more than one direction the potential is divided into as many ranges as possible movements. For instance if two possible directions of movement exist, then it will take one direction if the potential is less than 128 and the other if greater. The same process is then repeated for the next wumpus world grid square. The agent is terminated if either its energy level becomes zero, or all its neurons die, or all the dendrite or axon branches die, or if the agent return home.

Five randomly generated genotypes are produced and the corresponding agent behaviour is assessed to obtain its fitness. The best agent genotype is selected as the parent for a new population. The CGPCN is arranged in the following manner for this experiment. The neurons are confined to 3x4 CGPCN grid. Inputs and outputs to the network are located at five different random squares. The initial number of neurons is 5. The maximum number of dendrites is 5. The maximum branch *statefactor* is 7. The maximum soma *statefactor* is 3. The mutation rate is 5%. The maximum number of nodes per chromosome is 100.

5.2 Results and Analysis

We carried out twenty independent evolutionary runs (using the same wumpus world) and found the average numbers of generations that the agent took to return home without gold was 4, to find the gold was 13 and to return home with gold was 300.

While solving the wumpus world the CGPCN changes substantially in its structure. Figure 2 shows the variation in neural structures and growth of the CGPCN at different stages.

To test whether agent behaviour was general we took the programs for the best evolved agent on the original wumpus world and tested its performance on other wumpus worlds generated at random. We found that in some of the cases the agent was able to get the gold and bring it to home. In others the agent gets the gold but is unable to find its way to home. Sometimes it cannot find the gold and returns home empty handed. In others, the agent cannot find the gold or get back home. Interestingly, we found that the agent always first looks for the gold

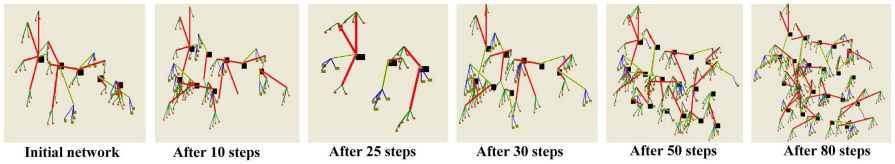

Fig. 2. Structural changes in agent CGPCN network at different stages of wumpus World shown against number of completed steps. The initial network has 5 neurons and 21 neurons after completing 80 steps. Black squares are somas, red thick lines are dendrites, yellowish green lines are axons, green lines are dendrite branches, and blue lines show axon branches.

at the place where the gold was when it was evolved. This is very interesting as it shows that the evolved *genetic codes* are able to build a computational network that holds information about where to find the gold from an initial small random network holding no information. This is reminiscent of instinctive behaviour.

We also examined the learning behaviour of a number of highly evolved agents when they were allowed to live after returning home. Figure 3 shows the original

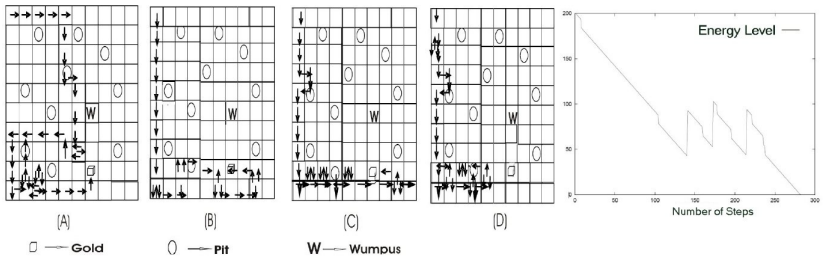

Fig. 3. Different paths followed by the agent in four consecutive trips from the home square towards the gold. The first path (far left) towards the gold starts with an initial random CGPCN. In the last path (far right) the agent wastes energy resulting in death. On the right it shows the variation in energy level of the agent during all these journeys

path that the agent took to the gold, when evolved (on left) and the three other paths to the gold on subsequent journeys. The agent takes 135 steps to find the gold in the first of these journeys. On the second journey (Fig 4b) the agent took an almost straight path towards the gold and encounters a pit once. On the third journey (Fig 4c) the agent takes almost the same path as the second journey, but with more energy wasting oscillations. It encounters two pits, but finally gets the gold. In the fourth and final journey the agent follows a similar path to that of third, but with even more oscillations and when it reaches a corner it gets stuck until its energy level decays to zero.

It is worth noting that an agent that moves in a straight line (even at edges) is highly non-random. A random move at an edge would have 33% probability.

Thus a straight path of 8-steps (as in figure 3b) returning home would have a probability of 0.0152% which is very unlikely to be random.

6 Conclusion

We have presented a brain-inspired computational system tested in a classic AI learning problem. We found that agents could acquire an understanding of the meaning of environmental signals and act intelligently based on them. In addition, the agents exhibited a form of memory that arose as a consquence of the interaction of their internal developmental processes and external environmental signals. We also tested the system without development (by ignoring the life cycle chromosome), and found that it took much longer to evolve networks with the same performance as the original system. This highlights the importance of development. In future work, we plan to evaluate this approach in richer and more complex environments.

References

1. J. Gaiarsa, O. Caillard, and Y. Ben-Ari. Long-term plasticity at gabaergic and glycinergic synapses: mechanisms and functional significance. *Trends in Neurosciences*, 25(11):564–570, 2002.
2. E. R. Kandel, J. H. Schwartz, and T. Jessell. Principles of neural science. *McGraw-Hill,4rth Edition*, pages 67–70, 2000.
3. G. Khan, J. Miller, and D. Halliday. Coevolution of intelligent agents using cartesian genetic programming. In *Proc. GECCO*, pages 269 – 276, 2007.
4. C. Koch and I. Segev. The role of single neurons in information processing. *Nature Neuroscience Supplement*, 3:1171–1177, 2000.
5. J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural selection*. MIT Press, 1992.
6. G. Marcus. *The Birth of the Mind*. Basic Books, 2004.
7. J. Miller, D. Job, and V. Vassilev. Principles in the evolutionary design of digital circuits-i. *Genetic Programming and Evolvable Machines*, 1(2):259–288, 2000.
8. J. Miller and P. Thomson. Cartesian genetic programming. volume 1802 of *Proc. EuroGP*, pages 121–132, 2000.
9. S. Rose. *The Making of Memory: From Molecules to Mind*. Vintage, 2003.
10. S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, 1995.
11. G. Shepherd. *The synaptic organization of the brain*. Oxford Press, 1990.
12. G. Stuart, N. Spruston, and M. e. Hausser. *Iterative Broadening: Dendrites*. Oxford University Press, 2001.
13. A. Van Ooyen and J. Pelt. Activity-dependent outgrowth of neurons and overshoot phenomena in developing neural networks. *Journal of Theoretical Biology*, 167:27–43, 1994.
14. J. A. Walker and J. Miller. The automatic acquisition, evolution and reuse of modules in cartesian genetic programming. *IEEE Trans. Evol. Comp.*, 12:in press, 2008.
15. G. Yob. Hunt the wumpus. *Creative Computing*, pages 51–54, 1975.