Solving Mazes using an Artificial Developmental Neuron

Gul Muhammad Khan[1] and Julian F. Miller[2]

[1]NWFP UET Peshawar, Pakistan, gk502@nwfpuet.edu.pk
[2]University of York, UK
jfm7@ohm.york.ac.uk

Abstract

An agent controlled by a single computational neuron is used to solve maze problems. The neuron has activity and time-dependent computational and topological structure. The behaviour of a neuron is controlled by a collection of seven evolved programs that are loosely analogous to aspects of biological neuron (dendrites, soma, axons, synapses, electrical and developmental behaviour). The programs are represented using Cartesian Genetic Programming. Our aim is to show that it is possible to evolve programs that develop a *single* neuron so that it is able to learn how to solve maze problems purely by experience.

Introduction

Although many techniques have been introduced to develop Artificial Neural Networks (ANNs) using genetic programming, we found no evidence that an attempt has been made to develop the *functional* model of real neurons with biological morphology. We have attempted to do this by devising an abstraction of real neurons which captures many important features. Various studies have shown that "dendritic trees enhance computational power" (Koch and Segev (2000)). Neurons communicate through synapses which are not merely the point of connection between neurons (Kandel et al. (2000)). They can change the strength and shape of the signal over various time scales. We have taken the view that the time dependent and environmentally sensitive variation of morphology and many other processes of real neurons is very important and richer models are required that incorporate these features. In our model a neuron consists of a soma, dendrites, axons with branches and dynamic synapses and synaptic communication. Neurite branches can grow, shrink, self-prune, or produce new branches. This allows it to arrive at a network whose structure and complexity is related to properties of the learning problem.

Our aim is to find a set of computational functions that encode neural structures with an ability to learn through experience. Such neural structure would be very different from conventional ANN models as they are self-training and constantly adjust themselves over time in response to external environmental signals. In addition they could grow new networks of connections when the problem domain required it.

From our studies of neuroscience, we have identified seven essential computational functions that need to be included in a model of a neuron and its communication mechanisms. From this analysis we decided what kind of data these functions should work with and how they should interact, however we cannot design the functions themselves. So we turned to a well established and efficient form of Genetic Programming called Cartesian Genetic Programming (CGP) (Miller and Thomson (2000)).

We have tested the learning capability of this developmental system on maze problems. A maze is a complex tour puzzle with a number of passages and obstacles (impenetrable barriers). It has a starting point and an end point. The job of the agent is to find a route from starting point to the end point. The agent starts with a limited energy that increases and decreases as a result of interaction with the paths and the obstacles in the maze environment. We show that the agent is able to solve the maze a number of times in a single life cycle. The agents start a maze with a single neuron having random structure. However, the branching structure of the neuron can grow and shrink during the game environment.

In previously work, we evaluated the effectiveness of this approach on a classic AI problem called wumpus world (Khan et al. (2007)). There we used a number of neurons to solve the wumpus world. We have also tested the network of CGP neurons for playing Checkers (Khan and Miller (2009)). We found that the agents improved with experience and exhibited a range of intelligent behaviours. In this paper we have turned our attention toward a single neuron. The motivation for this was to explore the capability of a single neuron in this model.

Biology of Neuron

Neurons are the main cells responsible for information processing in the brain. They are different from other cells in the body not only in term of functionality, but also in biophysical structure (Kandel et al. (2000)). They have different shapes and structures depending on their location in the

brain, but the basic structure of neurons is always the same. They have three main parts.

- Dendrites (Inputs): Receive information from other neurons and transfer it to the cell body. They have the form of a tree structure, with branches close to the cell body.

- Axons (Outputs): Transfer the information to other neurons by the propagation of a spike or action potential. Axons usually branch away from the cell body and make synapses (connections) onto the dendrites and cell bodies of other neurons.

- Cell body (Processing area or Function): This is the main processing part of neuron. It receives all the information from dendrite branches connected to it in the form of electrical disturbances and converts it into action potentials, which are then transferred through axon to other neurons. It also controls the development of neurons and branches.

Neural modeling

A number of techniques are used for simulation of neural development either in the form of construction algorithms or biologically-inspired growth processes. One approach aims to reproduce the geometrical properties of real neurons and does not consider the actual biological processes responsible for neural growth that could be used in an electrophysiology simulator (Stiefel and Sejnowski (2007)). Lindenmayer-System have been used to invent the procedure for modeling plant branching structures (Lindenmayer (1968)) and later has been successfully applied to develop neural morphologies (Ascoli et al. (2001)). A number of other methods such as probabilistic branching models (Kliemann (1987)), Markov models (Samsonovich and Ascoli (2005)) and Monte Carlo processes (da Fontoura Costa and Coelho (2005)) are also proposed as construction algorithm for neural development. Although these methods produce interesting neuronal shapes, they do not provide any insight into the fundamental growth mechanisms for neuronal growth. Growth models on the other hand provide the biological mechanisms responsible for generation of neuronal morphology. A number of interesting agent-based simulations are produced that highlights various aspects of biological development, such as cell proliferation (Al-Musa et al. (1999)), polarization (Samuels et al. (1996)), neurite extension (Kiddie et al. (2005)), growth cone steering (Krottje and van Ooyen (2007)) synapse formation (Stepanyants et al. (2008)) and axon guidance and map formation (de Gennes (2007)).

Although these methods introduce various interesting techniques to model the neuronal growth which is the early stage of development of brain, they have not consider the signal processing aspects and its effect on the growth during interaction with the world via sensory mechanisms. We introduce the method of evolving the functions that are responsiple for neuronal growth, signalling and synapse formation during the lifetime of the agent as explained in later sections.

Computational Development

In biology, multicellular organisms are built through developmental process from 'relatively simple' gene structures. The same technique could be used in computational development to produce complex systems from simpler systems that are capable of learning and adapting (Stanley and Miikkulainen (2003)).

Quartz and Sejnowski proposed a powerful manifesto for the importance of dynamic neural growth mechanisms in cognitive development (Quartz and Sejnowski (1997)). Marcus emphasized the importance of growing neural structures using a developmental approach (Marcus (2001)).

Parisi and Nolfi suggested that if neural networks are viewed in the biological context of artificial life, they should be accompanied by genotypes which are part of a population and inherited from parents to offspring (Parisi and Nolfi (2001)). They have used a growing encoding scheme to evolve the architecture and the connection strengths of neural networks. The network consists of a collection of artificial neurons distributed in 2D space with growing and branching axons. The genetic code inside them specifies the instructions for axonal growth and branching in neurons.

Cangelosi proposed a neural development model, which starts with a single cell that undergoes a process of cell division and migration until a collection of neurons arranged in 2D space is developed (Cangelosi et al. (1994)). At the end, neurons grow their axons to produce connection among each other until a neural network is developed. The rules for cell division and migration are stored in genotype, for a related approach see (Dalaert and Beer (1994)). Gruau also proposed a similar method (Gruau (1994)). The genotype used in Gruau's model is in the form of a binary tree structure as in GP (Koza (1992)).

Rust and Adams have used a developmental model coupled with a genetic algorithm to evolve parameters that grow into artificial neurons with biologically-realistic morphologies (Rust et al. (2000)). Jakobi created an impressive artificial genome regulatory network, where genes code for proteins and proteins activate (or suppress) genes (Jakobi (1995)). The proteins define neurons with excitatory or inhibitory dendrites. The individual cell divides and moves due to protein interactions causing a complete multicellular network to develop. Federici presented an indirect encoding scheme for development of a neuro-controller and compared it with a direct scheme (Federici (2005)). He implemented the system on a Khepera robot and tested it using direct and indirect encoding schemes, finding that the latter reached high fitness faster.

Downing favors a higher abstraction level in neural de-

velopment to avoid the complexities of axonal and dendritic growth while maintaining key aspects of cell signaling, competition and cooperation of neural topologies in nature (Downing (2007)). He tested it on a simple movement control problem known as *starfish*. The task for the k-limbed animate is to move away from its starting point as far as possible in a limited time, producing encouraging preliminary results.

One of the major difficulties in abstracting neuroscience is that one can lose the essential aspects required to make a powerful learning system. However the evidence of importance of time-dependent morphological processes in learning is highly compelling and we have thus included many of these aspects in a model of an artificial neuron.

The Neuron Model

This section describes the Cartesian Genetic Programming (CGP) and details the structure and processing inside the CGP Neuron and the way inputs and outputs are interfaced with it.

Cartesian Genetic Programming (CGP)

CGP is a well established and effective form of Genetic Programming. It represents programs by directed acyclic graphs (Miller and Thomson (2000)). The genotype is a fixed length list of integers, which encode the function of nodes and the connections of a directed graph. Nodes can take their inputs from either the output of any previous node or from a program input (terminal). The phenotype is obtained by following the connected nodes from the program outputs to the inputs. The function nodes used here are variants of binary if-statements known as 2 to 1 multiplexers (Miller et al. (2000)).

In CGP an evolutionary strategy of the form $1 + \lambda$, with λ set to 4 is often used (Miller et al. (2000)). The parent, or elite, is preserved unaltered, whilst the offspring are generated by mutation of the parent. If two or more chromosomes achieve the highest fitness then *newest* (genetically) is always chosen. We have used this algorithm in the work we report here.

Health, Resistance, Weight and Statefactor

Four variables are incorporated into the CGP Neuron, representing either fundamental properties of the neuron (*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 dendritic and axonal 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 neuron and branches inactive for a number of cycles. Only when the *statefactor* is zero are the neuron and branches are considered to be active and their corresponding program is run. *Statefactor* is affected indirectly by CGP programs.

Inputs, Outputs and Information Processing inside CGP Neuron

The signal is transferred to and taken from this neuron using virtual axon and dendrite branches by making synaptic connections.

The signal from the environment is applied to CGP neuron using five virtual input axo-synaptic connections. Five virtual output dendrite branches are used to decide the movement of the agent. The virtual axo-synaptic branches are allowed to not only transfer signals to the dendrite branches of processing neuron (CGP Neuron) but also to the output virtual dendrite branches which decide the movement of the agent. The CGP Neuron transfers signals to the virtual output dendrite branches using the program encoded in the axo-synaptic chromosome.

Information processing in the CGP Neuron starts by selecting the list of dendrites and running the electrical dendrite branch program. The updated 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 in forward direction using axo-synaptic branch programs.

Functionality of CGP Neuron

The CGP Neuron is placed at a random location in a two dimensional spatial neural grid (as shown in figure 1). It 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 dendrite branches. The branches may grow or shrink and move from one neural grid location to another. They can produce new branches and can disappear. Axon branches transfer information only to dendrite branches in their proximity. Electrical potential is used for internal processing of neurons and communication between neuron and is represented by an integer (32 bit).

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 neuron and communication between neurons. It consists of dendrite branch, soma, and axo-synaptic branch electrical chromosomes.

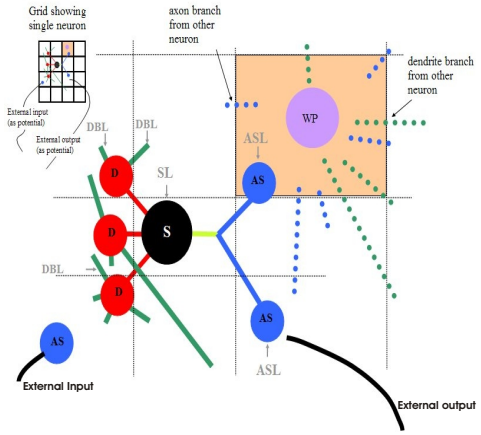

Figure 1: On the top left a neural grid is shown containing a single neuron. The rest of the figure is an exploded view of the neuron is given. Electrical processing parts: dendrite (D), soma (S) and axo-synapse branch (AS) are shown as part of neuron. Developmental programs responsible for the *life-cycle* of neural components are also shown (shown in grey). These are dendrite branch life (DBL), soma life (SL) and axo-synaptic branch life (ASL). The weight processing program (WP) is used to adjusts synaptic and dendritic weights.

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 updates their values. The *Statefactor* is decreased if the update in potential is large and vice versa.

If any of the branches are active (statefactor equal to zero), their life cycle program (DBL) is run, otherwise D 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) 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 neural grid square) branches.

Life Cycle of Neuron This part is responsible for replication, death, growth and migration of neurite branches. It consists of three life cycle chromosomes responsible for the 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 neural grid square 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.

Maze

A maze is a term used for complex and confusing series of pathways. It is an important subject for autonomous robot navigation and route optimization (Tani (1996); Blynel and Floreano (2003)). The idea is to teach an agent to navigate through an unknown environment and find the optimal route without having prior knowledge. A simplified version of this problem can be simulated by using a random two-dimensional synthetic maze. The pathways and obstacles in a maze are fixed.

Experimental Setup

In our experiments an agent is provided with CGP Neuron as its computational network. The job of the agent is to find routes from a starting point toward an end point of a maze as many times as it can in a single life cycle. We have used a 2D maze representation for this experiment as shown in figure 2. The 2D Maze representation is explored in a number of scenarios (Werbos and Pang (1996); Ilin et al. (2007)). We have represented the maze as a rectangular array of squares with obstacles and pathways (As shown in the figure 2). A square containing an obstacle cannot be occupied. Movement is possible up or down on squares on the outside columns. Movement is either left or right on rows, unless there is a pathway, in which case downward motion is possible. This is inspired by the clustering approach used to improve learning capabilities of an agent (Mannor et al. (2004)). We used different sizes of mazes to test the ability of the agent. The location of the obstacles, pathways and exit are chosen randomly for different experimental scenarios.

Energy of Agent The agent is assigned a quantity called energy, which has an initial value of 50 units. If an agent attempts to penetrate an obstacle its energy level is reduced by 5 units. If it encounters a pathway and moves to a row closer to the exit, its energy level is increased by 10 units. If it moves a row further away from the maze exit, its energy is reduced by 10 units. This is done to enhance the learning capability of agent by giving it a reward signal. If the agent

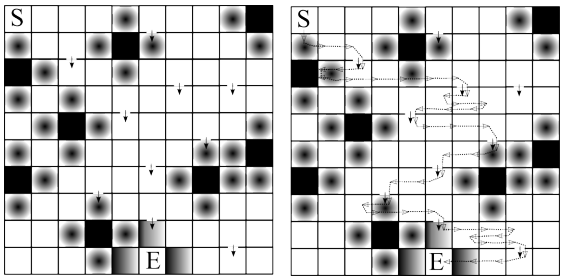

Figure 2: The left figure shows a 10x10 maze with impenetrable obstacles (black), downward pathways (arrows), start (S) and exit point (E), and their corresponding signals. On the neighbouring squares of an obstacle (north, south, east and west) and the exit there is a signal detectable by the agent indicating whether the agent is on a square neigbouring an obstacle (radial shading) or exit(linear shading). The figure on the right shows the path of an evolved agent.

reaches the exit, its energy level is increased by 50 units and it is placed back at the starting point and allowed to solve the maze again. Finally, if the agent arrives home, without having reached the exit, the agent is terminated. 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.

Fitness Calculation 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, increase fitness by one. This is done, to encourage the agents to have 'brain' that remains active and does not die.
- Each time the agent reaches the exit, its fitness is increased by 100 units.

Inputs to neuron The maximum allowed neural potential is $M = 2^{32}-1$. The agent's input axo-synapses can perceive input potentials, I, depending on the circumstances in the following way. Note that the agent can perceives *only one* signal on a maze square, even if there are more than one.

- $I = 0$ default.
- $I = M/60$ finds a pathway to a row closer to exit.
- $I = M/120$ tries to land on obstacle.
- $I = M/200$ on exit square.
- $I = M/100$ adjoining square north of an obstacle.
- $I = M/110$ adjoining square east of an obstacle.
- $I = M/130$ adjoining square south of an obstacle.
- $I = M/140$ adjoining square west of an obstacle.
- $I = M/180$ approaches exit from north direction
- $I = M/190$ approaches exit from east direction
- $I = M/210$ approaches exit from south direction
- $I = M/220$ approaches exit from west direction
- $I = M/255$ home square (starting point)

Agent movement and termination When the experiment starts, the agent takes its input from the starting point (on the top left corner as shown in figure 2). This input is applied to the computational network (CGP Neuron) of the agent using input axo-synapses. The network is then run for five cycles (one step). During this process it updates the potentials of the output dendrite branches. 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 $(M/2)$ and the other if greater. The same process is then repeated for the next maze square. The agent is terminated if either its energy level becomes zero or if it returns home.

CGP Neuron Setup The various parameters of CGP neuron are chosen as follows. The neuron's branches are confined to 3x3 CGPN neural grid. Inputs and outputs to the network are located at five different random squares. The maximum number of dendrites is 5. The maximum branch *statefactor* is 7. The maximum soma *statefactor* is 3. The mutation rate is 2%. The maximum number of nodes per chromosome is 100. Maximum number of dendrite and axon branches are hundred and twenty respectively. These parameters have not been optimized and have largely been chosen as they work reasonably well and do not incur a prohibitive computational cost.

Difficulty of the problem

It is important to appreciate how difficult this problem is. The agents starts with a single neuron with random connections. Evolution must find a series of programs that build a computational neural structure that is stable (not lose all branches etc.). Secondly, it must find a way of processing infrequent environmental signals (pathway, blocks, exit, home etc) and understand their meaning (beneficial and deleterious). Thirdly, it must navigate in this environment using some form of memory. Fourthly, it must confer goal-driven behaviour on the agent. The agent performance is determined by its capability to solve the maze as many times as it can during a single life cycle.

The maze environment we produced is much more complex than the traditional mazes, as the agent in this environment can only sense the signal from the maze square it is occupying, not from neighbouring squares. So in order to solve the maze the agent must develop a memory of each step it makes and the direction of movement, and use this memory to find a route toward the exit. As the structure and weights of branches changes at runtime while solving the maze, the learned information is stored both in weights and the structure of the neuron. The *capability to learn* and transformation of learned information into memory in the form of update in weights and structure is stored in genotype.

Results and Analysis

Figure 3 shows a number of mazes in first column. Fitness improvement during evolution is shown in the second column. The third column in figure 3 shows the energy variation of the best maze solving agent. The small continuous drop in energy is due to an agent losing its energy after every step. Large decreases occur through encounters with an obstacle or going away from the exit by following the pathway in opposite direction. Small increases shows the result of following the pathway and moving toward the exit and large increases happen when the agent finds the exit. The fourth and the last column shows the variation in neuron branching structure over the agent lifetime, while it is solving the maze.

The agent is able to solve the maze four to five times during a single life cycle in all the cases as shown in the second column of figure 3. During this process the structure of the neuron also changes in terms of the number of dendrite and axon branches. The fourth column of the figure 3 shows that although agents start with a minimal structure they soon achieve a structure that is most advantageous.

In traditional methods that train an agent to solve the maze and find a path, the network characteristics are fixed once it is trained to solve the maze. So if they are allowed to start the maze again they would always follow the same path. As the CGP Neuron continues to change its architecture and parameter values it also continues to explore different paths on future runs. This makes it possible for it to obtain (or forget!) a global optimum route. The networks is not trained to stabilize on a fixed structure, that it does so, seems to be because it has found a suitable structure for the desired task. The best architecture does not necessarily have to have the most neurite branches. This is evident from the varied characteristics in the last column of figure 3.

It is interesting to note that as the task become bigger and bigger the structure of the neuron grows in response to it. This is evident from the last column of the figure 3. For an 8x8 maze (first and second maze) the agent structure grows and stabilizes on a fairly small structure whereas for a 10x10 maze (3rd, 4th and 5th mazes) the number of dendrite and axon branches grows into a fairly large structure (the maximum allowed value is 100 in this case). Further investigation reveals that as the route toward the exit becomes more and more complex, the network structure become richer in terms of branches. This is evident from the second 10x10 maze (4th row) where the number of blocking paths are 10 (with each obstacle providing four walls in all the four directions, 40 walls), and number of pathways are 20. Ten on the sides (first and last column) with possibility to move in both upward and downward directions and ten that are only open toward the exit in downward direction). In this case the agent was able to solve the maze three times, as is evident from the rises in the energy level diagram. However, it dies on the fourth run when it tried to escape through the starting point. In next case, when we have reduced the number of obstacles to six (24 walls) while keeping the number of pathways the same as shown in the in fourth row of figure 3. This time the agent was able to solve the maze four times and its axon branch structure is improved during its run but the dendrite structure is stabilized on a low value. The final maze is a variant of 10x10 maze in third row with similar characteristics. In 8x8 mazes when the environment is simple, the agent was able to solve the maze a number of times even though it stabilized on a fairly small branch structure. This strongly suggests that the complexity of the CGP Neuron structure increases with increase in the complexity of the task environment.

Conclusion

We have described a neuron-inspired developmental approach to construct a new kind of computational neural architectures which has the potential to learn through experience. We found that the neural structure controlling the agents grows and changes in response to their behaviour, interactions with the environment, and allow them to learn and exhibit intelligent behaviour. We found that the network complexifies itself in response to the environmental complexity. The eventual aim is to see if it is possible to evolve a network that can learn by experience.

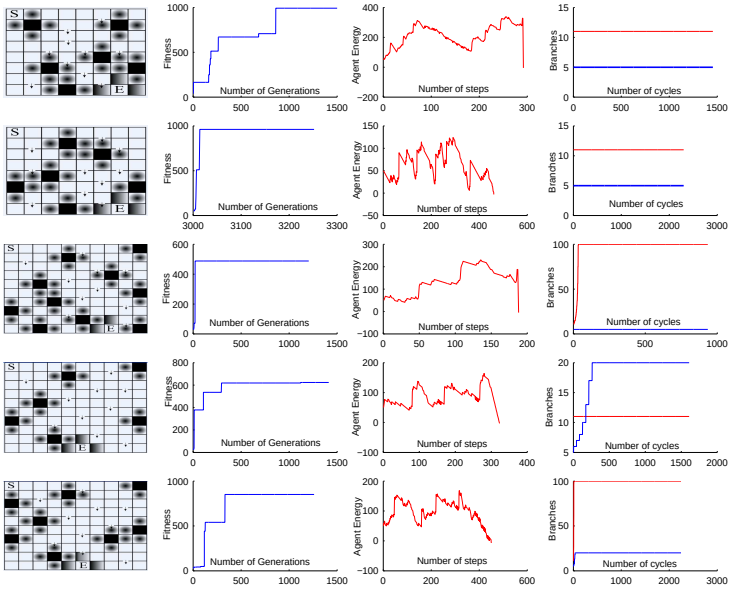

Figure 3: Mazes, Fitness, Best Run and Variation in Branch Structure

References

Al-Musa, S., Abu Fara, D., Badwan, A., Ryder, E., Bullard, L., Hone, J., Olmstead, J., and Ward, M. (1999). Graphical simulation of early development of the cerebral cortex. *Computer Methods and Programs in Biomedicine*, 59(2).

Ascoli, G. A., Krichmar, J. L., Scorcioni, R., Nasuto, S. J., and Senft, S. L. (2001). Computer generation and quantitative morphometric analysis of virtual neurons. *Anat. Embryol.*, 204.

Blynel, J. and Floreano, D. (2003). Exploring the t-maze: Evolving learning-like robot behaviors using ctrnns. In *EvoWorkshops*, pages 593–604. Springer Berlin / Heidelberg.

Cangelosi, A., Nolfi, S., and Parisi, D. (1994). Cell division and migration in a 'genotype' for neural networks. *Network-Computation in Neural Systems*, 5:497–515.

da Fontoura Costa, L. and Coelho, R. C. (2005). Growth-driven percolations: the dynamics of connectivity in neuronal systems. *Eur. Phys. J. B Condens Matter Complex Syst.*, 47.

Dalaert, F. and Beer, R. (1994). Towards an evolvable model of development for autonomous agent synthesis. In *Brooks, R. and Maes, P. eds. Proceedings of the Fourth Conference on Artificial Life*. MIT Press.

de Gennes, P.-G. (2007). Collective neuronal growth and self organization of axons. In *Proc. Natl. Acad. Sci. U.S.A.*, page 49044906.

Downing, K. L. (2007). Supplementing evolutionary developmental systems with abstract models of neurogenesis. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 990–996.

Federici, D. (2005). Evolving developing spiking neural net-

works. In *Proceedings of CEC 2005 IEEE Congress on Evolutionary Computation*, pages 543–550.

Gruau, F. (1994). Automatic definition of modular neural networks. *Adaptive Behaviour*, 3:151–183.

Ilin, R., Kozma, R., and Werbos, P. (2007). Efficient learning in cellular simultaneous recurrent neural network the case of maze navigation problem. In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 324–329. IEEE Press.

Jakobi, N. (1995). *Harnessing Morphogenesis, Cognitive Science Research Paper 423, COGS*. University of Sussex.

Kandel, E. R., Schwartz, J. H., and Jessell, T. (2000). *Principles of Neural Science, 4rth Edition*. McGraw-Hill.

Khan, G., Miller, J., and Halliday, D. (2007). Coevolution of intelligent agents using cartesian genetic programming. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 269 – 276.

Khan, G. M. and Miller, J. F. (2009). Evolution of cartesian genetic program capable of learning. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'09)*, pages 707–714. ACM.

Kiddie, G., McLean, D., Ooyen, A. V., and Graham, B. (2005). Development,dynamics and pathology of neuronal networks: from molecules to functional circuits, progress in brain research 147. *In Biologically Plausible Models of Neurite Outgrowth*.

Kliemann, W. (1987). A stochastic dynamical model for the characterization of the geometrical structure of dendritic processes. *Bull. Math. Biol.*, 49.

Koch, C. and Segev, I. (2000). The role of single neurons in information processing. *Nature Neuroscience Supplement*, 3:1171–1177.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural selection*. MIT Press.

Krottje, J. K. and van Ooyen, A. (2007). A mathematical framework for modeling axon guidance. *Bull. Math. Biol.*, 69.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development. parts 1 and 2. *J. Theor. Biol.*, 18.

Mannor, S., Menache, I., Hoze, A., and Klein, U. (2004). Dynamic abstraction in reinforcement learning via clustering. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 71.

Marcus, G. F. (2001). Plasticity and nativism: towards a resolution of an apparent paradox. pages 368–382.

Miller, J. F. and Thomson, P. (2000). Cartesian genetic programming. In *Proc. of the 3rd European Conf. on Genetic Programming*, volume 1802, pages 121–132.

Miller, J. F., Vassilev, V. K., and Job, D. (2000). Principles in the evolutionary design of digital circuits-part i. *genetic programming*. volume 1:1/2, pages 7–35.

Parisi, D. and Nolfi, S. (2001). *Development in Neural Networks. In Patel, M., Honovar, V and Balakrishnan, K.eds. Advances in the Evolutionary Synthesis of Intelligent Agents*. MIT Press.

Quartz, S. and Sejnowski, T. (1997). The neural basis of cognitive development: A constructivist manifesto. *Behav. Brain. Sci*, 20:537–556.

Rust, A., Adams, R., and H., B. (2000). Evolutionary neural topiary: Growing and sculpting artificial neurons to order. In *Proc. of the 7th Int. Conf. on the Simulation and synthesis of Living Systems (ALife VII)*, pages 146–150. MIT Press.

Samsonovich, A. V. and Ascoli, G. A. (2005). Statistical determinants of dendritic morphology in hippocampal pyramidal neurons: a hidden markov model. *Hippocampus*, 15.

Samuels, D. C., Hentschel, H. G., and Fine, A. (1996). The origin of neuronal polarization: a model of axon formation. philos. trans. r. soc. lond., b. *Biol. Sci.*, 351.

Stanley, K. O. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.

Stepanyants, A., Hirsch, J. A., Martinez, L. M., Kisvrday, Z. F., Ferecsk, A. S., and Chklovskii, D. B. (2008). Local potential connectivity in cat primary visual cortex. *Cereb. Cortex.*, 18.

Stiefel, K. M. and Sejnowski, T. J. (2007). In biologically plausible models of neurite outgrowth mapping function onto neuronal morphology. *J. Neurophysiol*, 98.

Tani, J. (1996). Model-based learning for mobile robot navigation from the dynamical systems perspective. *IEEE Trans. on Systems, Man, and Cybernetics*, 26:421–436.

Werbos, P. and Pang, X. (1996). Neural network design for j function approximation in dynamic programming. *Math'l Modeling and Scientific Comp.*, 2.