Evolution of Recurrent Neural Networks to Control Autonomous Life Agents

Tony Abou-Assaleh
Dept. Computer Science
University of Waterloo
Waterloo, ON, Canada
N2L 3G1
taa@acm.org

Dr. Jianna Zhang
Dept. Computer Science
Brock University
St. Catharines, ON, Canada
L2S 3A1
jianna@cosc.brocku.ca

Dr. Nick Cercone
Dept. Computer Science
University of Waterloo
Waterloo, ON, Canada
N2L 3G1
ncercone@math.uwaterloo.ca

Abstract

Studies of artificial life (alife) attempt to simulate simple living beings. On the other hand, autonomous agents researchers are interested in building agents that are able to complete a particular task without supervision. In this research, these two areas of artificial intelligence are combined into what we call "Autonomous Life Agent" (ALA). ALA is an artificial agent that is sent to some environment in which to live without any supervision or any predefined behaviour rules. The primary goal of the agent is to learn how to survive in its artificial environment. We utilize a recurrent neural network (RNN) to determine the agent's actions. A novel ALA Training System is developed that evolves recurrent neural networks using the genetic algorithms (GAs) approach. The resulting agents are capable of living in multiple similar worlds from random initial positions as well as in worlds that are unseen during training.

1 INTRODUCTION

In our previous work [1], we developed an ALA architecture and a basic training system. The system supported the ALA architecture with two internal variables: energy level and maintenance level. The world consisted of only three types of cells: empty cells, energy cells, and home cells. The basic training system evolved the link weights for a three-layered RNN with a predefined topology.

The objective of this research is to further explore the power of using RNNs to control ALAs. A new ALA Training System is implemented with the following specifications:

- simultaneous evolution of RNN topology and weights, optionally favouring RNNs with smaller topology
- user-defined internal variables and active cells
- agents start their life span from random initial positions and are capable of living in multiple similar world as well as in worlds unseen during training

2 GENERAL DESCRIPTION

ALA has a number of internal variables such as energy and maintenance levels. Each variable is reduced as the agent acts in the environment by a user-controlled rate. The agent dies when one of the variables becomes negative. Therefore, to remain alive, the agent has to periodically increment the values of the internal variables by visiting specific cells in the world that correspond to each variable. This increment is analogous to acquiring energy from energy cells and receiving maintenance in home cells.

The artificial environment in which the agent lives is a grid of cells. Each cell can be an empty cell, a wall cell, or a user-defined active cell that modifies an internal variable. Typically, there are at least two types of active cells: energy cells and home cells. Empty cells are passive cells. The agent can freely move over empty cells without any reward or penalty. On the other hand, moving into wall cells is disallowed. Any attempt by the agent to do so causes the agent's death. Staying over an active cell results in incrementing the corresponding ALAs internal variable by the amount provided by the cell up to a maximum limit of 1.0. After that, there is a time delay before the agent can re-consume the active cell.

One of the features of the new system is that the agent's world is surrounded by wall cells. This serves two purposes: (1) the agent must learn to stay within the boundaries of its world and (2) the boundary cases do not require special treatment in the implementation. Wall cells can also be placed at various locations to act as mines that the agent should avoid.

This task involves hidden system states. Some information about the world is not available to the agent as input. For instance, at any time, the agent can sense only the immediately surrounding cells and the current cell. Thus, the agent has to explore the world and encode its information in the agent's brain. Furthermore, some data is not available to the agent at all such as the time

delay of the active cells. The agent must automatically discover this information by observing the visible system states.

3 APPROACH

GA has been used to evolve the topology [5, 6, 8] and the weights [10] of neural networks since the 80's [9], as well as evolving network training parameters [2, 5]. Although evolving complete (both weights and topology) feedforward networks has been successfully achieved [7], less research has been conducted to apply this approach to RNNs.

The ALA Training System consists of five components: (1) agent, (2) genome, (3) population, (4) world, and (5) genetic engine. Each of these components is introduced in the following subsections.

3.1 AGENT STRUCTURE

The internal structure of the agent is a three-layer RNN where each layer is fully connected with the next layer. Some of the nodes in the hidden layer have recurrent links. The input vector consists of x and y coordinates, the types of the eight surrounding cells and the current cell, the values of the internal variables, and the previous action. The output layer contains nine outputs that represent a movement to one of the eight surrounding cells or staying at the current cell.

3.2 GENOME STRUCTURE

The genome completely defines an agent. Thus, it consists of two parts: (1) topology definition and (2) link weights. The topology definition includes the number of nodes in the hidden layer and the number of recurrent links in that layer. The link weights are serialized sets of two-dimensional arrays where each array `Ai` represents the weights of links from layer `i` to layer `i+1`. In addition, the initial outputs of the recurrent links are also represented in the genome.

3.3 POPULATION

The role of the population is to manage genomes and make them available for the genetic engine for reproduction. The selection of individuals is performed using tournament selection with a user defined size.

3.4 WORLD SRUCTURE

This component defines the environment in which the agent must live. Its main function is to evaluate agents by measuring how long they are able to survive in the given environment. The internal representation of the environment is a 2-dimensional grid where each cell consists of coordinates, type, increment, and delay. The increment is the amount of change that will affect the corresponding internal variable when the agent selects to 'stay' in this cell. The delay is the number of time intervals that must pass after the last time this cell resulted in a change in an internal variable in order for this cell to become active again. Examples of world configurations are provided in Tables 1.

3.5 GENETIC ENGINE

The genetic engine conducts the evolution process to evolve agents. It uses populations as collections of genomes and uses the world component to determine the fitness of individuals. The evolution terminates when the following termination criteria are met: (1) the agent lives up to the maximum age; and (2) the size of the genome is less than or equal to the desired genome size as specified by the user. N-point crossover and mutation are used for the weights. Topologies can only be mutated. In topology mutation, the number of the neurons in the hidden layer is changed as well as the number of the recurrent links to a random number. Once a new topology is defined, the link weights and the initial outputs of the recurrent links are copied from the original genome if possible or assigned random numbers between -1.0 and +1.0. Thus, in the topology mutation operation, effort is made to preserve the original weight values.

4 EXPERIMENTAL RESULTS

Since GAs produce a good solution and not necessarily the optimal solution, the experiments are directed towards evolving an agent who can primarily survive the environment with preference to solutions with smaller number of weights, rather than determining the absolute minimum number of nodes that is required in the hidden layer. We do not aim to correlate the size and complexity of the environment with the required minimum number of nodes in the hidden layer.

The experiments are conducted in four phases, which are described in the following subsections.

4.1 PHASE I

The experiments were performed on a 10x10 world definition with 4 types of active cells. For each run, the agent started his lifespan from the same initial position throughout the evolution. The results showed that the new system was able to evolve agents in environment that is more complex than in previous research [1]. Surprisingly, the evolved network is a feedforward network with 3 nodes in the hidden layer.

4.2 PHASE II

In this phase, a 7x7 world with 3 types of active cells was used. However, the agent starts from a random position every time it is evaluated. The agents are required to encode the world in their RNN to be able to go to the desired active cell from any position in the world. The evolved RNN was also a feedforward network with 3 nodes in the hidden layer.

4.3 PHASE III

The objective of this phase was to force the agent to employ judgement on current local input as well as generalized global knowledge of the active cells in the world. Two similar but distinct 10x10 worlds with 2 types of active cells are used. The evolved agent is able to live in the both worlds starting its life span from any cell.

Table 1: Phase III Parameters

Parameter	Value
Population size	2000
Number of generations	2000
Maximum age	500
Internal variables (variable, rate of change)	E, -0.02 H, -0.018
Training world 1 definition # = wall cell . = empty cell H = Home E = Energy	########## #........# #.H......# #........# #........# #........# #........# #........# #.......E# ##########
Training world 2 definition # = wall cell . = empty cell H = Home E = Energy	########## #........# #........# #.H......# #........# #........# #........# #........# #.......E# ##########
Testing world definition # = wall cell . = empty cell H = Home E = Energy	########## #........# #..H.....# #........# #........# #........# #........# #........# #.......E# ##########

Table 1 present some of the parameters that are used in this phase. The evolution progress averaged over the ten runs is shown in Figure 1. The number of generations was truncated to 244 generations, which is where the smallest ALA was found. The evolution progress of the run that evolved the smallest ALA is show in Figure 2.

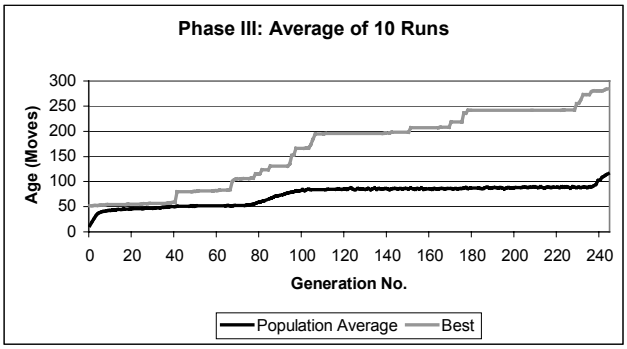

Figure 1: Evolution Progress of Phase III Averaged Over 10 Runs

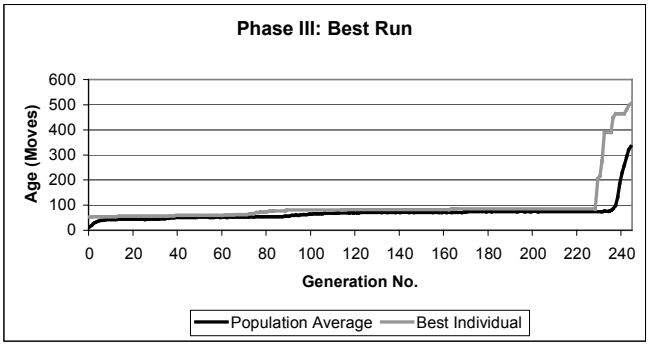

Figure 2: Evolution Progress of Best Run in Phase III

Out of 10 runs, 8 found an agent that can live up to the maximum age. However, 2 of the 8 successful runs evolved genomes with size of 105, which is greater than the desired size of at most 100. The best solution was found in generation 244 with size of 57, which corresponds to 2 neurons in the hidden layer and no recurrent links.

A testing world (see table 1), which is similar to but distinct from the two training worlds, is used to measure the performance of the evolved agents in unseen environment. Our goal is to observe the agent's generalization and fault tolerance abilities. These are rather ambitious intentions. The results of this testing phase, both positive and negative, serve as the basis for directing future research.

Only one ALA from the six candidate ALAs was able to live in the testing world. Its RNN had 3 nodes in the hidden layer with one of them having a recurrent link. The agent succeeded starting from all cells except cell (1,8), which is a corner cell.

5 CONCLUSION

This work presents a unique approach to evolving RNNs where a direct numeric representation is used to encode the definition of the hidden layer. The successful results

of the 3 phases of experimenting lead to a number of significant discoveries.

The scalability of the new ALA Training System is confirmed by training agents on much more complex world definitions than the ones used in our previous work. The evolved agents have great robustness, flexibility, and generalization abilities. ALAs are able to survive in environment that is unseen during the training. The initial position of the agent is also irrelevant. The agent is capable of memorizing the approximate positions of the active cells during the training process and applying this knowledge to new environment.

The evolution of the topology of the RNN has led to discovering a number of interesting results. Firstly, the required number of nodes in the hidden layer is much smaller than what is used traditionally. The experimental results show that it is possible to evolve agents with as few as 1 node in the hidden layer and still have good agent performance in non-trivial world definitions. Secondly, The literature review that I carried out in previous and current work indicates that recurrent neural networks have always been used (as opposed to feedforward neural networks) to control autonomous agents. More specifically, recurrent neural networks have always been the only choice whenever the gradient function is not available. However, recurrent links are not required in the neural network in order to successfully control autonomous agents in environment with hidden system states.

6 FUTURE WORK

The findings of this research form the basis for the first author's Masters studies at the University of Waterloo.

The next step is to further test the flexibility and scalability of the system by using more complex and different worlds for training and testing. Ideally, the evolution of ALAs with practical applications would be involved. For instance, using the evolved RNNs to control a physical robot is one of the long-term goals of this research.

We are currently working on an empirical comparison between Gas and conventional RNN training algorithms such as back-propagation through time (BPTT). The training time and the performance of feedforward neural networks vs. recurrent neural networks are also being investigated.

Acknowledgements

The first author is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under the USRA award.

The second author provided the financial support by her NSERC Grant 228142-2000.

The authors thank Communications and Information Technology Ontario (CITO) for its support.

Special thanks and appreciation to Dr. Brian Ross for his valuable remarks.

References

[1] T. Abou-Assaleh and J. Zhang, "Autonomous Life Agent Using Recurrent Neural Networks and Genetic Algorithms", Proc. Late Breaking Papers of AAAI Genetic and Evolutionary Computation Conference (GECCO), pp. 1-5, 2000.

[2] R.K. Belew, J. McInerney, and N.N. Scharaudolph, "*Evolving Networks: Using Genetic Algorithms with Connectionist Learning*", CSE technical report CS90-174, University of California at Dan Diego, La Jolla, CA, 1990.

[3] S.A. Harp, T. Samad, and A. Guha, "*Towards the Genetic Synthesis of Neural Networks*", Proc. Third International Conference on Genetic Algorithms, pp. 360-369, 1989.

[4] J.H. Holland, "*Adaption in Natural and Artificial Systems*", MIT Press, 1992.

[5] J.R. Koza and J.P. Rice, "*Genetic Generation of Both the Weights and Architecture for a Neural Network*", Proc. IEEE International Joint Conference on Neural Networks, pp. II-397 - II-404, 1990.

[6] Jm.M Molina, A. Torresano, I. Galván, P. Isasi, and A. Sanchis, "*Evolution of Context-free Graph Grammers for Designing Optimal NN Architecture*", Proc. Genetic and Evolutionary Computation Conference Workshop Program, pp. 61-63, 2000.

[7] D. Polani and R. Miikkulainen, "*Eugenic Neuro-Evolution for Reinforcement Learning*", Proc. Genetic and Evolutionary Computation Conference, pp. 1041-1046, 2000.

[8] T. Ragg, H. Braun, and H. Landsberg, "*A Comparative Study of Neural Network Optimization Techniques*", Proc. International Conference on Artificial Neural Nets and Genetic Algorithms, pp. 341-345, 1997

[9] J.D. Schaffer, D. Whitley, and L.J Eshelman, "*Combination of Genetic Algorithms and Neural Networks: A Survey of the State of the Art*", Proc. International Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 1-37, 1992.

[10] A.P. Wieland, "*Evolving Neural Network Controllers for Unstable Systems*", Proc. IEEE International Joint Conference on Neural Networks, pp. II-667 - II-673, 1990.