Evolution Strategies for Evolving Artificial Neural Networks in an Arcade Game

Tse Guan Tan [1], Jason Teo [2], Patricia Anthony [3]

[1,2] Evolutionary Computing Laboratory
Universiti Malaysia Sabah, MALAYSIA
[1] tseguantan@gmail.com, [2] jtwteo@ums.edu.my

[3] Center of Excellence in Semantic Agents,
Universiti Malaysia Sabah, MALAYSIA
panthony@ums.edu.my

ABSTRACT

The aim of this paper is to use a simple but powerful evolutionary algorithm called Evolution Strategies (ES) to evolve the connection weights and biases of feed-forward artificial neural networks (ANN) and to examine its learning ability through computational experiments in a non-deterministic and dynamic environment, which is the well-known arcade game called Ms. Pac-man. The resulting algorithm is referred to as an Evolution Strategies Neural Network or ESNet. This study is an attempt to create an autonomous intelligent controller to play the game. The comparison of ESNet with two random systems, Random Direction (RandDir) and Random Neural Network (RandNet) yields promising results.

Keywords
Evolution Strategies, Evolutionary Artificial Neural Networks, Ms. Pac-man

1.0 INTRODUCTION

Evolutionary algorithms include four main divisions, which are genetic algorithms, evolution strategies, evolutionary programming and genetic programming (Yao, 2002). An evolutionary algorithm naturally initializes its population randomly where a set of individuals are created. Subsequently, fitness evaluation is a process to measure the fitness of each individual in the population based on a fitness function. Then, selection is a natural method, whereby individuals which can generate new offspring are selected. Normally, this selection can be divided into two approaches, parent selection and survivor selection. Parent selection is a method that is used to choose individuals as parents in the population. New offspring are produced through crossover, which inherits information from the parent. The newly created offspring can then be mutated. The mutation operation changes a small part of the offspring’s genetic information. Finally, survivor selection is a routine to decide the best individuals for the next generation. The evolutionary algorithm process is finished once it has achieved the terminating conditions. An EA can be used successfully in complex problems, involving features such as discontinuities, multimodality, disjoint feasible spaces and noisy function evaluations (Fonseca & Fleming, 1995). Additionally, EAs can be hybridized with traditional optimization techniques (e.g. linear programming, non-linear programming, and dynamic programming) and non-traditional optimization techniques (e.g. fuzzy logic and ant colony algorithm) to improve the efficiency of the algorithms in order to solve the real-world problems.

In this study, the feed-forward artificial neural network (ANN) is evolved with the Evolution Strategies (ES) for the computer player to automatically learn and optimally play the game of Ms. Pac-man. The proposed algorithm will be referred to as an Evolution Strategies Neural Network or ESNet throughout the paper. The ESNet is benchmarked against the Random Direction (RandDir) and Random Neural Network (RandNet) in the same domain. The importance of the proposed algorithm for decision-making in a dynamic environment is that the agent will not only be able to make an intelligent decision like a human player in the computer or video game, but also that the successful application of these techniques will be highly beneficial to the real-world problems, such as in the application of robotics and other complex systems.

The organization of this paper is as follows. In Section 2, the Ms. Pac-man game as an application domain will be discussed. The structures of RandDir, RandNet and ESNet will be explained in Section 3. In Section 4, the benchmarking results and discussions are given. Finally, conclusions are shown in Section 5.

2.0 MS. PAC-MAN

Ms. Pac-man is a famous maze-based game from Midway Manufacturing in 1981 as shown in Figure 1. The objective of Ms. Pac-man is simply to survive as long as possible, while achieving the highest possible score. Ms. Pac-man was inherited from Pac-man, therefore the basic game-play of Ms. Pac-man still mostly remains the same as the original game. The agent (yellow circle with a ribbon) is controlled by the player to be moved around the maze in any of the four directions (up, down, left or right) attempting to eat the pills and fruits, and to evade the ghosts. There are four ghosts named Blinky (red colored), Pinky (pink colored), Inky (blue colored) and Sue (orange colored) in the game that chase the agent and try to kill it in the maze. The maze is filled with hundreds of pills (small dots) and four power pills (large dots. When the agent eats a power pill, the ghosts will change to become edible ghosts (dark blue colored), so that the agent can then chase and gobble down the ghosts to earn some extra points for a limited period of time. Another good way to collect extra points is by eating the fruits that appear twice in every level that bounce around the maze. The first fruit appears after agent has eaten 64 pills (including power pills) whereas the second fruit appears after agent has eaten 176 pills. By default, Ms. Pac-man initially has three lives, however, if the total score reaches 10000 points, an extra life will be awarded, while the player loses a life when a ghost kills the agent. The game will proceed to the next level when all the pills on the maze had been eaten. As the levels increase, the degree of speed and difficulty increase as well, thus making the game more challenging. The game is considered ended when the player loses all lives or completes all levels.

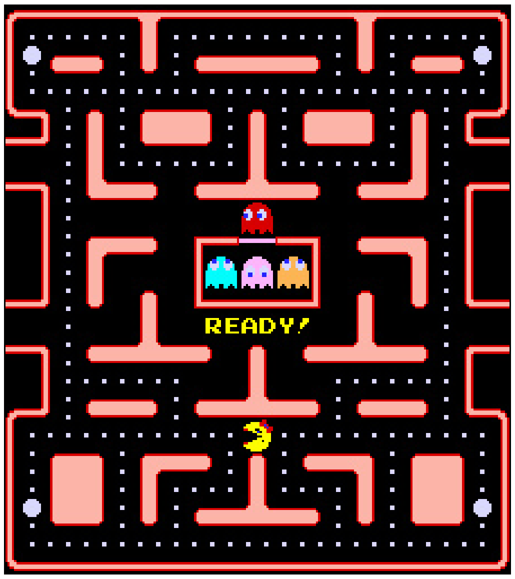

Figure 1: The first maze of Ms. Pac-man

3.0 DEVELOPMENT OF EXPERIMENTAL SYSTEMS

In this research, the feed-forward ANN (Haykin, 2009) with one hidden layer is applied. The architecture can be briefly described as 5-20-1, with 5 inputs, 20 hidden units and 1 output unit. A log-sigmoid activation function is used for network units. The number of evaluations per run is fixed at 500 in the ESNet, whereas the RandDir and RandNet are evaluated only once game in a run since both algorithms did not involve any learning process. The general parameters are summarized in Table 1. Below we give a description of the three algorithms, RandDir, RandNet and ESNet.

2.3 Random Direction (RandDir)

RandDir is a very simple controller, which directs the Ms. Pacman agent to move in random directions in the maze. There are 4 directions available to the agent: up, down, left and right. At each time step or every few time steps, the algorithm applies a random change to the agent's direction.

2.4 Random Neural Network (RandNet)

The structure of the RandNet controller is shown in Figure 2. This system begins with a randomly initialized vector of weights and biases for the ANN from a uniform distribution in the range between -1 and 1, whose output is then used to control the Ms. Pacman agent.

2.5 Evolution Strategies Neural Network (ESNet)

ES is a simple and fast algorithm was envisaged by Rechenberg and Schwefel in 1965 as a numerical optimization technique. The (1+1)-ES for a two membered ES has been applied to train the ANN by evolving the weights and biases, called ESNet. In the initialization phase, the ANN weights and biases are encoded into a chromosome from uniform distribution with range [-1, 1] to act as parent and evaluate its fitness. Subsequently, polynomial mutation operator was used to create an offspring from the parent and evaluate its fitness. After that, the offspring and parent are compared. If the offspring performs better than the parent, then the parent is replaced by the offspring as new parent for the next evaluation (generation). Otherwise the offspring is eliminated and a new mutated offspring is generated. If parent and offspring are incomparable, the offspring is compared with set of previously nondominated individuals in the archive. Figure 3 shows the flowchart of ESNet.

The fitness function F chosen for maximization is based on the score obtained in each evaluation as follows:

$$F = \sum_{n=1}^{N} (Ms.\ Pacman\ Scores) \qquad (1)$$

where n and N represent the number of lives in a full game.

Table 1: Parameters setting

Parameters	Setting
Number of inputs	5
Number of outputs	1
Number of hidden layers	1
Number of hidden neurons	20
Activation function	Log-sigmoid
Mutation operator	Polynomial mutation
Mutation probability	0.9
Distribution index	20.0
Number of runs	10

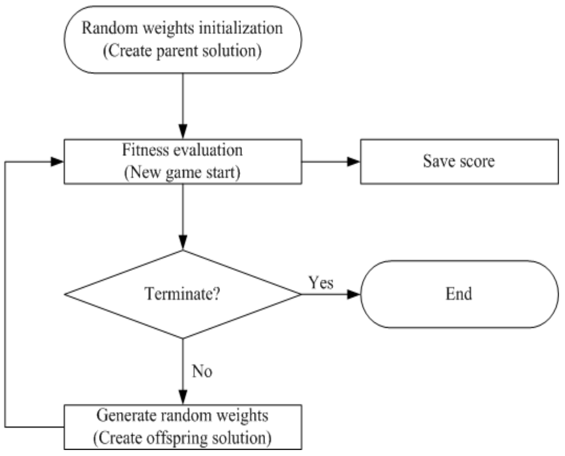

Figure 2: The flowchart of RandNet algorithm

4.0 RESULTS AND DISCUSSIONS

Table 2 presents the experimental data in 10 independent runs to compare the performances of RandDir, RandNet and ESNet.

Table 2: Ms. Pac-man obtained scores.

Run	RandDir	RandNet	ESNet
1	450	700	6350
2	410	720	5700
3	470	670	6680
4	430	260	5920
5	420	200	6020
6	460	650	6270
7	420	420	5960
8	470	560	6420
9	460	900	6500
10	400	490	5930
Mean	**439**	**557**	**6175**

There were significant differences in the scores for ESNet (Mean = 6175), RandNet (Mean = 557) and RandDir (Mean = 439). According to the analyzed results, the performance of ESNet is noticeably better than that of RandDir and RandNet. Furthermore, it was shown to be produce relatively higher results compared with previous studies (Lucas, 2005; Handa, 2008; DeLooze & Viner, 2009). Hence, we have shown empirical evidence that ESNet can improve the learning capability of the ANN by evolving their weights and biases in a dynamic video game setting.

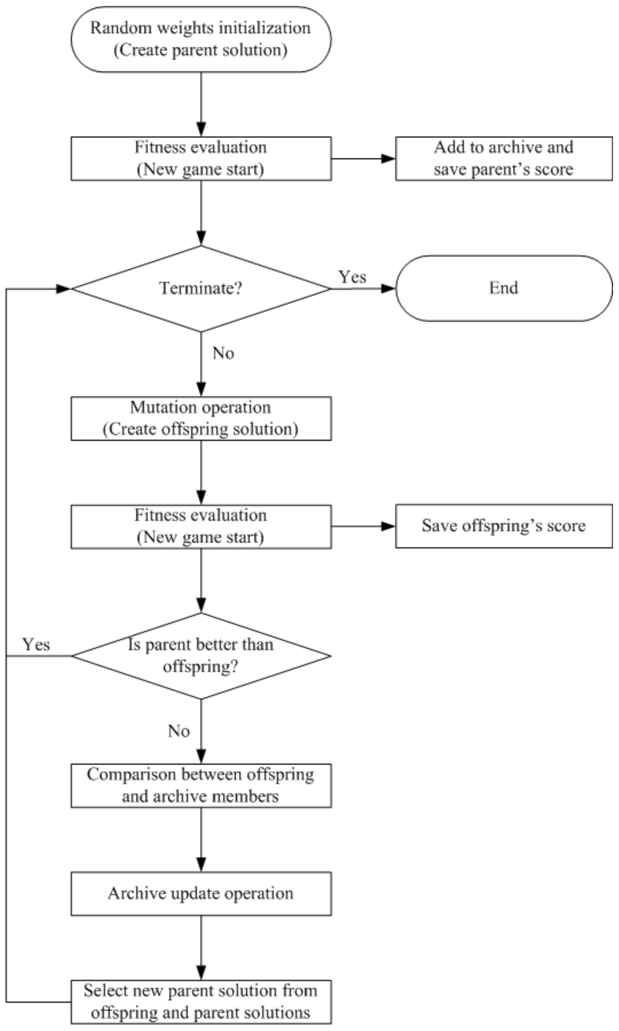

Figure 3: The flowchart of ESNet algorithm

5.0 CONCLUSIONS

ESNet is introduced in this paper to create an intelligent Ms. Pac-Man agent to play the game. This proposed algorithm has been tested against RandDir and RandNet and the results revealed that the ESNet clearly outperforms both random systems. Overall, ESNet has been successfully used to optimize the performance of ANN.

ACKNOWLEDGMENT

This research is funded under the ScienceFund project SCF52-ICT-3/2008 granted by the Ministry of Science, Technology and Innovation, Malaysia.

REFERENCES

DeLooze, L. L., & Viner, W. R. (2009). Fuzzy Q-Learning in a Nondeterministic Environment: Developing an Intelligent Ms. Pac-Man Agent. *In Proc. 2009 IEEE Symposium on Computational Intelligence and Games*, 162-169.

Fonseca, C. M., & Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, *3*(1), 1-16.

Handa, H. (2008). Constitution of Ms. Pacman Player with Critical-Situation Learning Mechanism. *In Proc. 4th International Workshop on Computational Intelligence and Applications*, pp. 48-53.

Haykin, S. (2009). *Neural Networks and Learning Machines* (3rd ed.): Prentice Hall, 1-46.

Knowles, J. D., & Corne, D. W. (1999). The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimization. *In Proc. 1999 Congress on Evolutionary Computation*, pp. 98-105.

Lucas, S. M. (2005). Evolving a Neural Network Location Evaluator to Play Ms. Pac-Man. *In Proc. IEEE Symposium on Computational Intelligence and Games*, 203-210.

Yao, X. (2002). *Evolutionary Optimization*: Kluwer Academic,27