Evolving Neural Network to Play Game 2048

Tuponja Boris, and Šuković Goran, *Member*, *IEEE*

Abstract — **This paper introduces the Neuroevolution of Augmenting Topologies method for evolving artificial neural networks to play game 2048. Over the course of several hundred generations, the network evolves and learns how to play game.**

***Keywords* — evolutionary algorithms, NEAT, neural networks, neuroevolution.**

I. Introduction

The field of computational computational intelligence in games is well established research field, but still rapidly developing. Neuroevolution [1]-[2] is one of the common methods applicable to a wide range of problem. Neuroevolution refers to the generation of artificial neural networks (their connection weights and/or topology) using evolutionary algorithms. One of the most popular neuroevolution algorithms is NEAT (Neuroevolution of Augmenting Topologies) [3]. This paper presents the series of experiments with NEAT method in order to evolve neural network which is capable to learn how to play game 2048. Game 2048 [4] is very popular puzzle game, and humans after some time of playing, would be able to gain tile 2048 without difficulty. However, higher tile values are hard to reach. A main motivation for performing these experiments is that neuroevolution is an important method that has seen continued popularity since its inception, and that there are successful application of artificial neural network in playing game SuperMario [1].

II. NEAT and 2048

Experiments were implemented using modified ANJI library [5] . In total 15 experiments were performed, divided in 5 groups. Every member of a population (i.e. network) has been tested on 10 different starting states of the board. The fitness function was average score from 10 played games. Also, every experiment has been run 10 times, because of stochastic nature of the game. Experiments were performed with predefined number of generations, using following hardware configuration: Intel i7 4790 3.6 GHz processor with 4 cores and 32 GB DDRAM III. Average execution time of experiment was 35h, one run at the time. Parts of experiment runs, where it was possible, were executed in parallel. This lead to 70% CPU and 40% memory load.

In all the experiments neural network had 16 input neurons, one for every space on the board. Number of output neurons was 4, one for 4 possible moves. Initial neural networks did not have hidden nodes and were fully connected, i.e. all input neurons were connected to all output neurons.

Beside algorithm implementation, game GUI (Figure 1) was created which allows us to follow computer's moves.

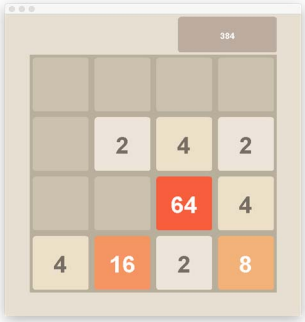

Figure 1 – Game board after few played moves

A. First group of experiments: Different board mappings

In this group focus was on mappings of the game board to neural network inputs. For every input, next move was decided by a neuron that had the greatest output value. If multiple neurons had same maximal output, then move was chosen randomly between those neurons. If chosen move was not possible, game was ended. Our aim was to push program towards making only correct moves.

Algorithm parameters for these experiments are shown in Table 1.

TABLE 1: First group experiment parameters.

Parameter	Value
Number of generations	5000
Population size	200
Add edge mutation probability	0.002
Add neuron mutation probability	0.001
Weight mutation probability	0.8
Coefficient of excess neurons	1
Coefficient of disjoint neurons	1
Coefficient of common neurons	0.4
Speciation threshold	0.8
Weight of edges	[-500, 500]

Input neurons use linear activation function, and all other neurons use sigmoid activation function.

For the first experiments, the board was mapped to

Tuponja Boris, University of Montenegro, Faculty of Mathematics and Natural Sciences, Podgorica, George Washington st. bb, Podgorica 20000, Montenegro (tel: +382-69-528-960, e-mail: boris@mnet.me)

Šuković Goran, University of Montenegro, Faculty of Mathematics and Natural Sciences, Podgorica, George Washington st. bb, Podgorica 20000, Montenegro (tel: 382-69-376669, e-mail: goran.sukovic@gmail.com).

978-1-5090-4086-5/16/$31.00 ©2016 IEEE

neural network inputs as it was and no changes were made on tile values.

For the second experiments we used the fact that all values are powers of 2. For every tile on the board, $\log_2$ was calculated and provided as an input to the neural network. Using this transformation, we got a linear increase of values instead of exponential increase in the first experiment.

Min-max normalization within segment [0, 1] was used for the third experiment.

Fourth and fifth experiments both used z-score normalization, but fifth experiment also used empty tiles as an attempt to check if number of empty tiles could have positive impact on evolution.

In the sixth experiment, we used transformation similar to the one from second experiment, but the values are normalized in segment [0, 1].

Best approximate registered score for every experiment are shown in Table 2.

TABLE 2: FIRST GROUP EXPERIMENT SCORES.

Mapping	Score
Values from board	650
$\log_2$ values	800
min-max normalization	650
z-score without empty tiles	300
z-score with empty tiles	650
$\log_2$ values with normalization in [0,1]	1000

First and third experiments gave similar results, as we expected, because min-max normalization kept features of original data.

Second and sixth experiments gave best results due to a linear increment. Figure 2 shows average score per generation for sixth experiment. Red lines shows minimal value, blue lines is average value and green line presents maximal value. Graphs for other experiments look similar to this one, with lower scores.

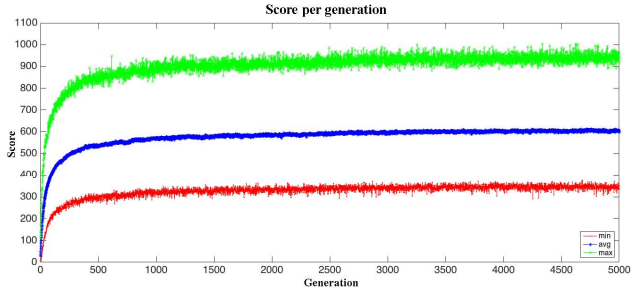

Figure 2 – Graph of scores per generation for experiment 6 from first group

Because of small mutation probability of network topology, size of neural networks increased very slowly (one edge/neuron per 1500 generations). Furthermore, these parameters lead to great number of species. We started with 1 specie, but after 2-3 generations number of species was increased to 198-200 species, i.e. almost every neural network has own specie. This is equivalent to standard evolutionary algorithm, and actually we don't need to use any specific feature of NEAT.

B. Second group of experiments: Multiple tries

Experiments shares the same setup with previous group with one change - game was not ended after first wrong move. Instead, other neurons with maximum output are tried.

As expected, average score was higher. Main idea behind these experiments was to check how much higher scores would be. Results are presented in Table 3.

TABLE 3: SECOND GROUP EXPERIMENT SCORES.

Mapping	Score
Values from board	2200
$\log_2$ values	2500
min-max normalization	2000
z-score without empty tiles	2000
z-score with empty tiles	1800
$\log_2$ values with normalization in [0,1]	2500

Once again, experiments 2 and 6 gave best results. The scores ar reached after just 20 generations and it stay in that score region till the end, with very mild uptrend. Still, these mappings did not create too significant deviation from other mappings.

Unlike the experiments with z-score normalization in the first group, this time mapping that used empty tiles recorded lower score. However, that experiment reached his maximum after 50 generations and the experiment that did not use empty tiles reached its maximum after 100 generations.

Size of neural networks and number of species in population were same as in the first group.

C. Third group of experiments: Reducing species count

This group contains only one experiment. Rules for stopping game were same as in the first group of experiments, i.e. game stops after first wrong movement. Also, because normalized values of $\log_2$ had shown best results, only that mapping was used for following experiments.

Focus of this experiment was to use speciation feature of NEAT that could not come to the fore before. Most parameters are same as in Table 1, with following changes in parameters: coefficients of excess and disjoint neurons were decreased to 0.5, coefficient of common neurons was set to 0.2, and speciation threshold was set to 0.3

Number of species reached 100 after 40 generations where that number vary in range of [-5, 5]. This is shown in Figure 3 (for clarity, only first 200 generation are shown, because latter have same values).

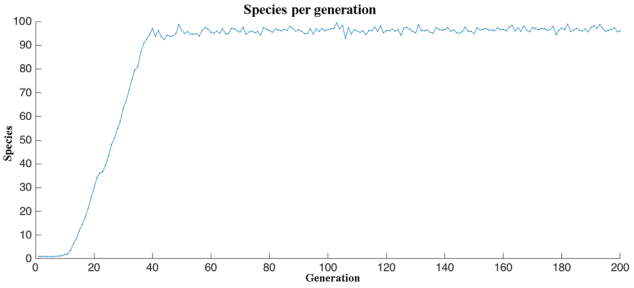

Figure 3 – Graph of number of species per generation in experiment from the third group

Unfortunately, using only corrections regarding species did not improve average network score. Score was little lower comparing to the previous group, as shown in Figure 4. Still, there was other parameter that should be considered such as complexity of neural networks. Because the experiment did not include change of topology related parameters, network complexity was too low and stay at the same level as in the previous experiments. Therefore, this experiment was extended to include more topology changes.

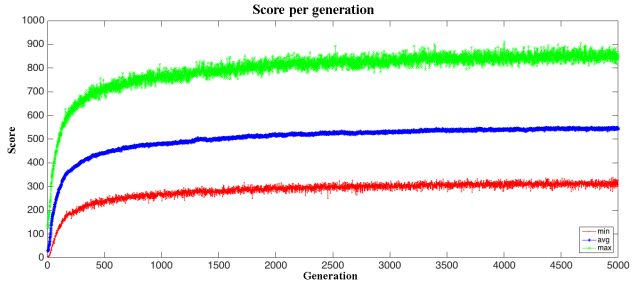

Figure 4 – Graph of scores per generation for the experiment from the third group

D. *Fourth group of experiments: Increasing complexity of neural networks*

This group also presents only one experiment. Experiment setup use same parameters as previous experiment. Node mutation probabilitiy was set to 0.05 and add edge mutation probability was set to 0.08.

During this experiment average complexity of neural networks incremented linearly as shown in Figure 5. However, score did improve slightly, but not as much as we expected. It remained on area of 900, but unlike the experiment from third group, maximal score stayed above 900 (Figure 6.).

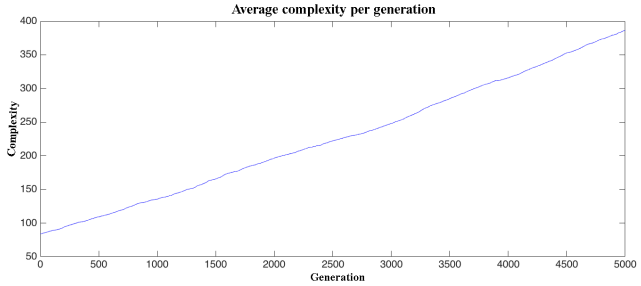

Figure 5 – Complexity of neural networks for the experiment from the fourth group

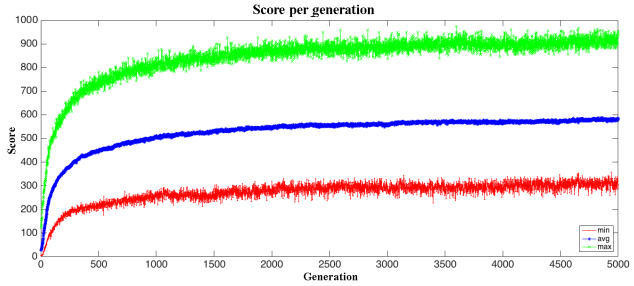

Figure 6 – Graph of scores per generation for the experiment from the fourth group

All the previous experiments observe every board configuration as independent state. However, for systems that predict a next state it is important to have information about previous states. With this in mind, an experiment, which was also only experiment in this group, used recurrent edges in neural networks in order to memorize previous board configurations.

Parameters for the experiment were same as for the experiment in fourth group and number of cycles in recurrent edges was 1, i.e. neural network memorized only one previous board configuration.

With these changes on neural networks, there were great improvements in average scores. The experiment recorded score of 1000 in first 500 generations and score was increased to 1500 till 5000th generation with slight uptrend. However, based on the previous observations this trend probably would not increment score drastically. Score graph for this experiment is presented in Figure 7.

Increasing number of cycles in neural networks did not lead to further improvements.

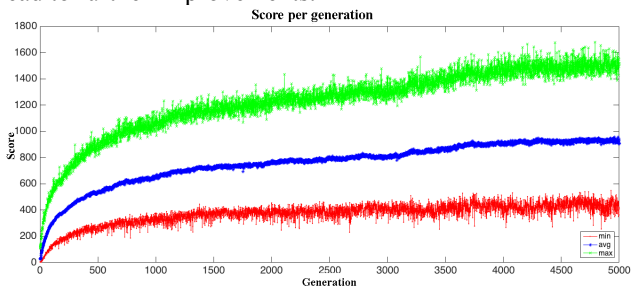

Figure 7 - Graph of scores per generation for the experiment from the fifth group

III. Conclusion

Game 2048 was a real challenge for NEAT method. A lot of randomness that were involved in game, including initial board configuration and random selection of movement when multiple neurons had maximal value, drastically complicates evolution. We develop number of board mapping trying to find most suitable encoding for NEAT. Experiments show that all mappings give similar scores. Parameter changes did not have high impact as expected, but introduction of recurrent edges looks promising.

Still, there is a lot of space for improvement and first step towards agent that could excel this game. Neuroevolution is already a technique that can be applied more or less out of the box for some problems. There are also various NEAT approaches that have been only superficially explored.

References

[1] Togelius, J.; Shaker, N.; Karakovskiy, S.; Yannakakis N.G. The Mario AI Championship 2009-2012. AI Magazine, 34(3), 89-92, 2013

[2] Togelius, J.; Risi, S. Neuroevolution in Games: State of the Art and Open Challenges, arXiv:1410.7326

[3] Stanley, Kenneth O.; Miikkulaien, Risto: *Evolving Neural Networks through Augmenting Topologies,* MIT Press, 2002

[4] Game 2048: https://en.wikipedia.org/wiki/2048_(video_game)

[5] ANJI library. Available: http://anji.sourceforge.net/