Evolving Neural Networks through Augmenting Topologies

Patrik Medur[1], Luka Šćulac[2] and Anthea Zubčić-Đurđević[3]

***Abstract*— This paper is devoted to the use of the Neural Evolution of Augmenting Topologies (NEAT) algorithm to train agents in the "LunarLander-v2" environment from OpenAI Gymnasium API. The goal is to develop neural networks to achieve a soft landing of a spacecraft on the lunar surface in the designated area with the least amount of fuel used. NEAT employs selection, recombination, and mutation, with the objective of enhancing performance in dynamic simulations. The criteria for evaluation are distance to the landing pad, speed, angle, and the use of engines. The outcomes demonstrate the evolutionary advancement with the progression of fitness scores across generations. Several parameter adjustments were made to get the best model, which proved that NEAT is useful in training reinforced neural networks.**

I. INTRODUCTION

This paper focuses on the creation of intelligent agents for specific simulation from the OpenAI Gymnasium environment: "LunarLander-v2", using the NEAT algorithm [1]. The aim is to understand how the architectures of the neural networks can be evolved to optimize their performance in simulated dynamic systems. The goal of the environment is to place a spacecraft on the lunar surface within the given area with the least amount of fuel and with maximum control. This task entails the ability of the agent to learn the best policies through its interaction with the environment and guided by rewards. The NEAT framework used in this paper is adopted using the NEAT-Python [2] library.

II. METHODOLOGY

In the OpenAI environment, NEAT uses evolutionary algorithms to optimize neural network structures. The initial stage involves creating inputs for these networks, with each genome representing a specific neural network. NEAT evolves genomes by assigning fitness values to each genome based on these inputs. The best-performing genomes are selected and recombined, occasionally mutated, depending on defined probability. This process continues iteratively until an individual with the desired fitness level is achieved or surpasses the maximal number of generations. Through this method, NEAT optimizes neural network structures by using evolutionary algorithms. Each cycle of evolution allows for the exploration of new potential solutions, enabling the emergence of highly fit individuals. The combination of selection, recombination, and mutation causes progressive population improvement, enabling the development of highly optimized genomes that meet specific fitness criteria.

*This work was not supported by any organization

[1]P. Medur, University of Rijeka Faculty of Engineering, Department of Computer Science, Croatia

[2]L. Ščulac, University of Rijeka Faculty of Engineering, Department of Computer Science, Croatia

[3]A. Zubčić-Đurđević University of Rijeka Faculty of Engineering, Department of Computer Science, Croatia

III. CASE STUDY

A. Environment

As mentioned before we use the "LunarLander-v2" environment. This environment is a dynamic simulation that generates data by simulating random starting conditions for each episode. It is composed of 8 input values:

- coordinates of the lander in the x and y dimensions
- linear velocities in the x and y dimensions
- lander angle
- angular velocity
- two boolean values indicating whether each leg is in contact with the ground

Also, there are 4 discrete output values:

- doing nothing
- firing the left orientation engine
- firing the main engine
- firing the right orientation engine

Only one output can be active at any time, making this a discrete environment.

B. Evaluation

Each genome in the current population undergoes evaluation to determine its fitness. The evaluation process involves running each genome through 5 iterations, during which the sum of rewards is calculated and averaged to give the genome its fitness score for that generation. The initial setup starts with zero hidden neurons to ensure searching for optimal performance in minimal complexity before moving to higher dimensions.

The reward system for genome fitness is based on several factors: increased/decreased by proximity to the designated landing area, increased/decreased the slower/faster the lander is moving, decreased the more the lander is tilted, increased by 10 points for each leg that is in contact with the ground, decreased by 0.03 points each frame a side engine is firing, decreased by 0.3 points each frame the main engine is firing. Landing safely awards +100 points while crashing results in a deduction of -100 points. The ultimate goal is for the genome to surpass a fitness threshold of 250 points.

Key parameters used in NEAT-Python algorithm for the training of genomes include:

- **Genome Population size:** 150
- **Fitness threshold:** 250
- **Number of initial hidden neurons:** 0
- **Connection add probability:** 0.15

- **Connection delete probability:** 0.1
- **Node add probability:** 0.15
- **Node delete probability:** 0.1
- **Activation function:** tanh
- **Maximum generations** 350

IV. RESULTS

In the final 25th generation, the population's average fitness was -127.97148 with a standard deviation of 132.95120. The best fitness achieved was 265.82874, obtained by an individual with a genome size of (8, 21), where 8 represents the number of hidden and output neurons, and 21 represents the number of connections. The genome population started with 6 species, and by the end, 8 species survived while 2 species were extinguished for not having any innovation in 15 generations.

V. DISCUSSION

The results obtained from the final 25th generation show that the fitness is achieved relatively early in the process. Although the average fitness is -127.97148, it is better than the initial -442.92413. This improvement is also emphasized by a standard deviation of 132.95120. A high standard deviation like this one means a wide range of fitness levels, with some individuals struggling while others adjust effectively. In figure [1], three lines are visible: the blue line represents the average fitness, the red line shows the highest individual fitness achieved, and the green dotted line displays the sum of average fitness and one standard deviation. The best individual with a fitness of 265.82874 is an example of successful evolution, showing the evolutionary algorithm's ability to find optimal solutions. The final neural network topology, as shown in figure [2], features green arrows representing positive weights, while red arrows represent negative weights. Dotted arrows indicate that the connections are not active, but were used while training, whereas solid arrows mean active contributions to the final network's behavior. The best solution was achieved with a genome size of 8 neurons (4 hidden and 4 outputs) and 21 connections. This specifies that the best solution was found in a more complex configuration than it started with. The hyperbolic tangent (tanh) activation function was shown to be effective because it enables nonlinear transformations within the neural network. The increase in species from 6 to 8 suggests that speciation helped maintain diversity, preventing premature convergence and enabling the development of various strategies. This diversity is important for exploring different solutions and avoiding local optima. Figure [3] depicts the emergence and extinction of species when innovation is absent over several generations.

VI. CONCLUSION

The application of NEAT in training agents within the "LunarLander-v2" environment has shown the effectiveness of the evolutionary approach for training neural networks through evolutionary techniques. In the 25 generations, the population showed improvements in fitness levels observed from the results. The maximum fitness score was 265.82874 for the best genome, which proves that NEAT is effective in evolving neural networks to find the best solutions. The increase of species number from 6 to 8 proved that the algorithm can help generate new ideas and avoid the issue of premature convergence, this enables the exploration of various strategies. The literature review shows that NEAT is a practical solution to the challenges of reinforcement learning problems. Because of its adaptability and effectiveness, it is ideal for creating new and more sophisticated neural networks to solve different problems.

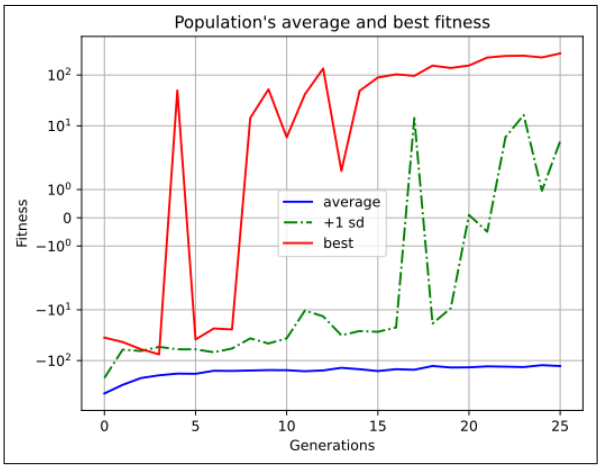

Fig. 1. Fitness through generations

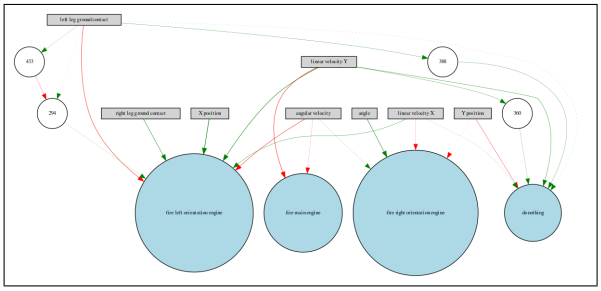

Fig. 2. Topology of best genome

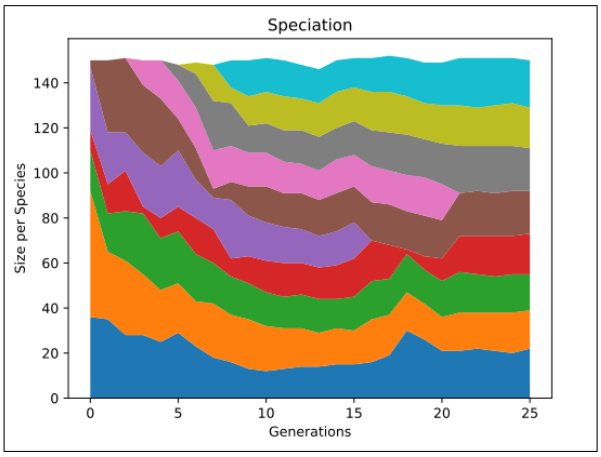

Fig. 3. Number of genomes in species through generations

REFERENCES

[1] K. O. Stanley, R. Miikkulainen, Evolving Neural Networks through Augmenting Topologies, The MIT Press Journals, vol. 10, no. 2

[2] https://neat-python.readthedocs.io/en/latest/