Training a Lunar Lander agent with NEAT

line 1: 1st Bora Ersoy
line 2: *Izmir Instute of Technology*

line 4: Izmir, Turkiye

line 1: Emir Kaldırımcı
line 2: *Izmir Instute of Technology*

line 4: Izmir, Turkiye

***Abstract*—This project explores the application of NeuroEvolution of Augmenting Topologies (NEAT) to train a lunar lander in a simulated environment. The objective is to develop an AI capable of successfully landing a spacecraft on the lunar surface, optimizing precision, and safety. The NEAT algorithm evolves neural networks by optimizing topologies and weights, enabling the system to learn complex control behaviors without predefined structures. Through iterative simulations, the project demonstrates how NEAT enables adaptive learning, allowing the lunar lander to handle diverse landing scenarios. Key findings include the algorithm's ability to balance exploration and exploitation, achieve convergence on effective strategies, and generalize across varying initial conditions. This study highlights NEAT's potential in real-time control systems and autonomous navigation for space exploration.**

Keywords—NeuroEvolution of Augmenting Topologies (NEAT), lunar lander, artificial intelligence, autonomous navigation, reinforcement learning, neural networks, simulation, real-time control, space exploration, adaptive learning, fuel efficiency, precision landing.

I. Introduction

The challenge of landing a spacecraft on the lunar surface is a complex and high-stakes problem in the field of space exploration. Achieving a safe and accurate landing requires precise control systems that can adapt to a variety of uncertain and dynamic conditions, such as surface irregularities and gravitational forces. Traditionally, spacecraft landings rely on meticulously designed control algorithms and predefined behaviours, which may not easily account for unforeseen conditions or variations in terrain. As space missions become more ambitious, the need for flexible, adaptive systems capable of learning optimal behaviours becomes more pronounced.

This project explores the use of **NeuroEvolution of Augmenting Topologies (NEAT)**, an evolutionary algorithm that evolves neural networks, as a solution for autonomous lunar lander control. NEAT allows for the evolution of both the structure (topology) and the weights of neural networks, providing the flexibility needed to learn complex tasks without being limited to a fixed architecture. Unlike traditional methods, NEAT offers the advantage of dynamic adaptation, enabling the lander to learn from experience and adjust to new, previously unseen landing conditions.

The primary motivation behind this work is to investigate the potential of NEAT to autonomously control the lunar lander in a simulated environment, optimizing for landing precision and safety. By leveraging NEAT's ability to evolve increasingly effective control policies, the project aims to explore the feasibility of creating adaptable AI systems that can perform high-stakes tasks in unpredictable environments.

This study tackles the problem of autonomous spacecraft landing, focusing on optimizing key parameters such as landing accuracy and resilience to disturbances. By training a neural network using NEAT, the lunar lander is tasked with learning how to land successfully under varying conditions, all while avoiding crashes. This work presents a novel approach to spacecraft landing and lays the groundwork for further exploration of AI-driven systems in space missions.

II. METHODOLOGY (Algorithm Design)

This section describes the evolutionary algorithm (EA) implemented for the Lunar Lander project using NEAT (NeuroEvolution of Augmenting Topologies). The algorithm evolves neural network architectures to optimize the control strategy for the Lunar Lander.

1. Representation

The NEAT algorithm represents solutions as neural networks encoded by genomes. Each genome consists of nodes and connections:

Nodes: Represent neurons and are categorized as input, hidden, or output nodes.

Connections: Represent weighted edges between nodes, determining the signal's flow and strength.

In this project:

Inputs: The state is an 8-dimensional vector: the coordinates of the lander in x & y, its linear velocities in x & y, its angle, its angular velocity, and two booleans that represent whether each leg is in contact with the ground or not.

Outputs: 4 outputs corresponding to the possible actions.

- 0: do nothing
- 1: fire left orientation engine
- 2: fire main engine
- 3: fire right orientation engine

Initial connections: Fully connected, feed-forward networks, as specified in the configuration (initial_connection = full and feed_forward = True).

2. Initialization

The population is initialized with 150 genomes Each genome starts as a simple neural network with minimal complexity, enabling gradual evolution.

Key initial parameter settings:

Node biases: Randomly initialized with a mean of 0.0 and standard deviation of 1.0.

Connection weights: Initialized with a mean of 0.0 and standard deviation of 1.0.

3. Fitness Evaluation

After every step a reward is granted. The total reward (fitness of a genome) of an episode is the sum of the rewards for all the steps within that episode.

For each step, the reward:

- is increased/decreased the closer/further the lander is to the landing pad.
- is increased/decreased the slower/faster the lander is moving.
- is decreased the more the lander is tilted (angle not horizontal).
- is increased by 10 points for each leg that is in contact with the ground.

The episode receives an additional reward of -100 or +100 points for crashing or landing safely respectively.

An episode is considered a solution if it scores at least 200 points.

Criterion: The maximum fitness achieved by a genome in the population.

Threshold: The algorithm terminates when a genome reaches a fitness of 300 or when it reaches 70th generation.

4. Selection

Selection is based on fitness, employing a species-based approach. The NEAT algorithm maintains a diverse population by grouping genomes into species based on compatibility distance. Parameters:

Species compatibility threshold: 3.0, ensuring meaningful grouping of genomes.

Species fitness function: Maximum fitness within a species.

Stagnation criteria: A species is removed if no improvement is observed for 20 generations

Survival Threshold: 20 percent of the generation survives through next generation.

5. Crossover

Crossover is performed between genomes within the same species. Elitism ensures the top two genomes in each species are preserved unaltered (elitism = 2).

6. Mutation

Mutation introduces diversity by modifying genomes. Key mutation operations include:

Node addition/removal: Nodes are added with a probability of 0.2 and removed with the same probability.

Connection addition/removal: Connections are added with a probability of 0.5 and removed with the same probability.

Weight mutation: Connection weights are perturbed with a probability of 0.8, using a mutation power of 0.5.

Bias mutation: Node biases are mutated with a probability of 0.7.

Activation functions: Only the sigmoid function is used as an activation function.

7. Termination Criteria

The algorithm terminates when:

1. A genome achieves the fitness threshold of 350.
2. A predefined number of generations is reached (if applicable).

8. Implementation Details

The algorithm was implemented using the NEAT-Python library, configured as shown in the configuration file. The environment used was the OpenAI Gym Lunar Lander simulation, providing an episodic reward system as the fitness metric. The NEAT algorithm's inherent mechanisms, such as speciation and incremental complexity, were leveraged to evolve effective control strategies for the Lunar Lander.

.

III. Experimental work

The hyperparameters for the NEAT algorithm were configured based on the default settings provided by the NEAT-Python library. Hyperparameter optimization was not conducted, as automating this task proved challenging. The NEAT framework relies on a configuration file in text format for parameter specification, making dynamic adjustments cumbersome. Additionally, the Gym API and the NEAT framework were not inherently designed to support seamless integration for automated hyperparameter tuning. As a result, the experiments were carried out using a static configuration that demonstrated satisfactory performance. However here are the plot of the evolving fitness and most fit neural network configuration below.

Figure 1.

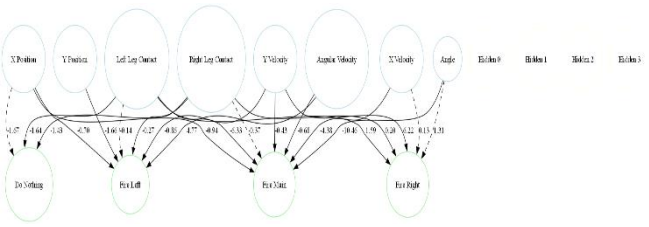

Figure 2.

In figure 1 it is seen that fitness is successfully exceeded 200 (minimum fitness that agent must have to be considered successful) and reached the fitness score of 277.
In figure 2, the neural topology of the fittest genome is shown. Weights and connections can be seen in this plot.

IV. CONCLUSION AND DISCUSSION

This study demonstrated the effectiveness of the NEAT algorithm in solving the Lunar Lander problem, achieving the predefined fitness threshold and showing promise for evolving neural networks in control tasks. However, automating hyperparameter optimization was difficult due to the reliance on a text-based configuration file and the lack of seamless integration between the NEAT framework and the Gym API.

The findings highlight NEAT's potential for reinforcement learning tasks but also suggest areas for improvement, particularly in framework compatibility and ease of configuration adjustments. Future work could focus on automating the configuration process and exploring enhancements like multi-objective optimization or hybrid approaches to improve performance and scalability.

V. COLLABARATION IN GROUP

This project was a joint effort between myself (Bora) and Emir. I took the lead in implementing the NEAT algorithm for the Lunar Lander problem, handling the configuration, integration with the Gym API, and conducting the experiments. I was responsible for the coding, analysis, and the majority of the report's content.

Emir contributed to the initial stages of the project, including discussions around the overall approach and assisting with the setup. His feedback during the project, especially during the experimental phase, was valuable in refining the direction of the work. Emir also played a role in reviewing the final report, ensuring clarity and consistency in the presentation of the findings.

Our collaboration allowed us to effectively tackle the problem, combining my technical implementation with Emir's input and review, ensuring a comprehensive and cohesive outcome.

REFERENCES

[1] J. W. Hoagg and D. S. Bernstein, "Retrospective cost model reference adaptive control for nonlinear systems," IEEE Transactions on Control Systems Technology, vol. 14, no. 4, pp. 730-740, 2006.

[2] K. T. Alligood, T. D. Sauer, and J. A. Yorke, Chaos: An Introduction to Dynamical Systems. New York: Springer, 1996.

[3] OpenAI Gymnasium Contributors, "Gymnasium Documentation," OpenAI Gymnasium, 2024. [Online]. Available: https://gymnasium.farama.org.

[4] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529-533, 2015.

[5] N. Hansen and A. Ostermeier, "Completely derandomized self-fitt in evolution strategies," Evolutionary Computation, vol. 9, no. 2, pp. 159-195, 2001.

[6] J. H. Holland, Adaptation in Natural and Artificial Systems. Cambridge, MA: MIT Press, 1992.

[7] K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," Evolutionary Computation, vol. 10, no. 2, pp. 99-127, 2002.

[8] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the NERO video game," IEEE Transactions on Evolutionary Computation, vol. 9, no. 6, pp. 653-668, 2005.

□