Acquired Intelligence and Adaptive Behaviour 2024
Coursework No 2

Investigation of Genetic Algorithms for Solving the CartPole-v1 Task

Anthony Guerges

Candidate Number – 262809

1 Abstract

This investigation explores the application of different genetic algorithms (GAs) to the CartPole-v1 task. The CartPole task involves balancing a pole on a cart by applying forces to the cart in a manner that prevents the pole from falling over. The goal is to keep the pole balanced for as long as possible. Genetic algorithms are employed to evolve the weights of a neural network agent that controls the cart's movements based on observations from the environment. This report compares the performance of two GAs in terms of their ability to optimise the agent's control policy.

2 Introduction

Genetic algorithms (GAs) are evolutionary strategies used to optimise complex problems by mimicking natural selection[3]. This study investigates the effectiveness of two distinct GAs—Hillclimber and Microbial GA—in optimising a neural network agent for the CartPole-v1 task. The Hillclimber algorithm is a simple evolutionary strategy that iteratively improves a single solution through small mutations. In contrast, the Microbial GA employs a population of solutions and uses tournament selection with replacement to evolve solutions[4].

Our Null hypothesis is that the Microbial GA will demonstrate superior performance and robustness compared to the Hillclimber algorithm due to its population-based approach, which maintains greater genetic diversity and avoids local optima more

effectively[9]. To test this hypothesis, we conducted experiments using the CartPole-v1 environment from OpenAI's Gym, which provides a standardised benchmark for this task.

3 Hypothesis and Methods

3.1 Environment

For this investigation, we utilised the CartPole-v1 environment from OpenAI's Gym library.

CartPole-v1 Environment Details

- State Space: The environment's state space includes four variables: cart position, cart velocity, pole angle, and pole angular velocity.[2]
- Action Space: The agent can choose between two actions: applying a force to the left or to the right of the cart.
- Objective: The objective is to keep the pole balanced by applying appropriate actions to the cart, maximising the cumulative reward over time.

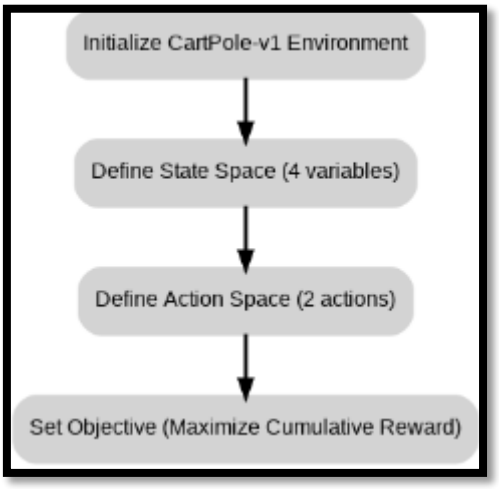


Figure 1 Environment set-up Flowchart

3.2 Agent

We defined a single agent with a neural network architecture to map the environment's state space to the action space.[7] The agent's architecture and parameters were optimised using genetic algorithms.

3.3 Neural Network Structure

- Inputs: Four input neurons corresponding to the state variables.

- Outputs: One output neuron to decide the direction of force applied to the cart.
- Genome Encoding: The agent's genome encodes the weights and biases of the neural network. Specifically, the genome consists of weights connecting the input neurons to the output neuron and the biases for the output neuron.

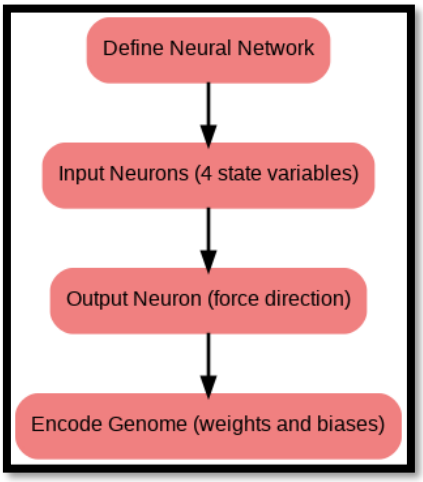


Figure 2 Neural Network Flowchart

3.4 Adaptation Mechanisms

We compared two genetic algorithms: Hillclimber and Microbial GA. Each algorithm's implementation and evaluation involved several steps, outlined below.

Hillclimber algorithm: A simple evolutionary strategy that iteratively improves a single solution by small mutations. The process is shown in the flowchart below:

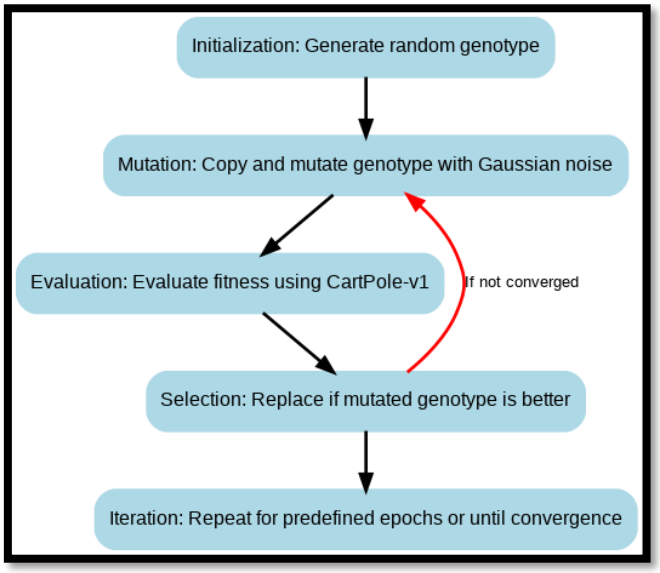


Figure 3 Hillclimber Implementation and Evaluation

Microbial GA: involves a population of multiple individuals and employs tournament selection with replacement.[4] The process is shown in the flowchart below:

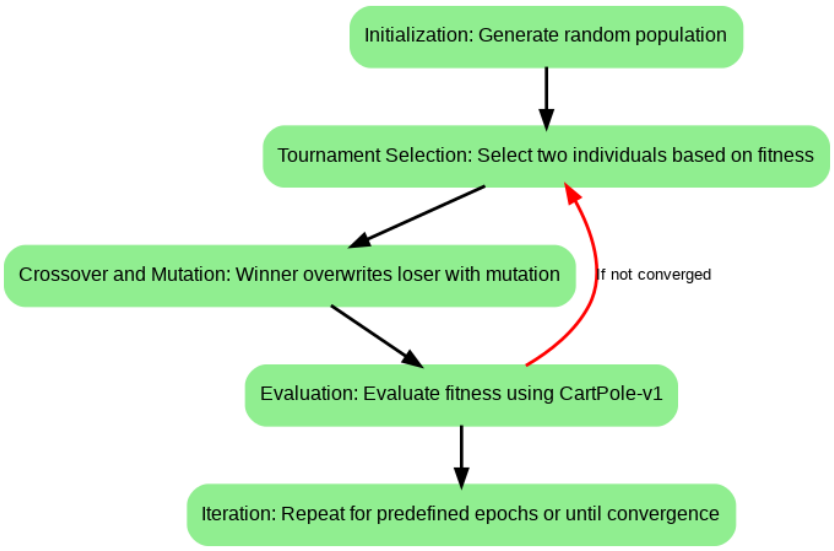


Figure 4 Microbial GA Implementation and Evaluation

3.5 Parameters

- Population Size: For the Hillclimber, a single genotype is used, while for the Microbial GA, a population of 15 genotypes is used.
- Mutation Rates: Various mutation rates (0.01, 0.05, 0.1, 0.2) were tested to assess their impact on the performance of each algorithm.

3.6 Functions

- Fitness Function: This was defined as the total reward accumulated during an episode. An episode ends when the pole falls over or the time limit is reached.
- Complex Mutation Function: For both algorithms, we used a complex mutation function that adjusts the mutation rate dynamically based on the current fitness landscape. The pseudocode for this is shown below.

```
function complex_mutation(gene_pop, fitness, mean = 0,
base_std = 0.1, dynamic_adjustment = True, adjustment_factor = 0.05):
    new_gene_pop = copy(gene_pop)
    std = base_std

    if dynamic_adjustment:
        avg_fitness = mean(fitness)
        max_fitness = max(fitness)
        std = base_std + adjustment_factor * (max_fitness - avg_fitness)

    for each gene in gene_pop:
        mutation = random_normal(mean, std)
        new_gene_pop[gene] = gene + mutation

        # Apply boundary constraints
        if new_gene_pop[gene] > 4:
            new_gene_pop[gene] = 4
        if new_gene_pop[gene] < -4:
            new_gene_pop[gene] = -4

    return new_gene_pop
```

Figure 5 Pseudocode of Functions

3.7 Experimental Procedure

To compare the performance of the algorithms, we followed these steps:

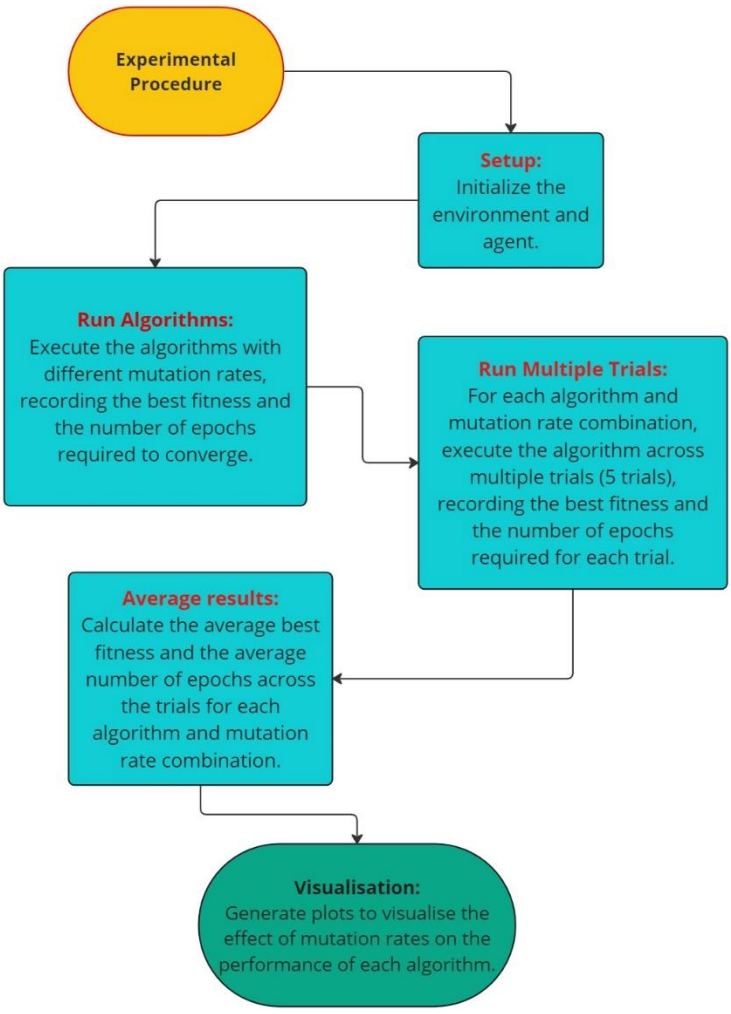


Figure 6 Experimental Procedure Flowchart

3.8 Performance Metrics

To evaluate the algorithms' performance, we used two main metrics:

- Best Fitness: The highest fitness score achieved by the algorithm, indicating the best solution found.
- Number of Epochs: The number of epochs required for the algorithm to converge to the best solution, indicating the efficiency of the algorithm.

By analysing these metrics, we aimed to understand how mutation rates influence the performance and efficiency of the Hillclimber and Microbial GA algorithms, providing a more comprehensive evaluation through multiple trials.

4 Results and Discussion

The Hillclimber and Microbial GA algorithms displayed distinct behaviours when applied to the CartPole-v1 task, highlighting their strengths and weaknesses.

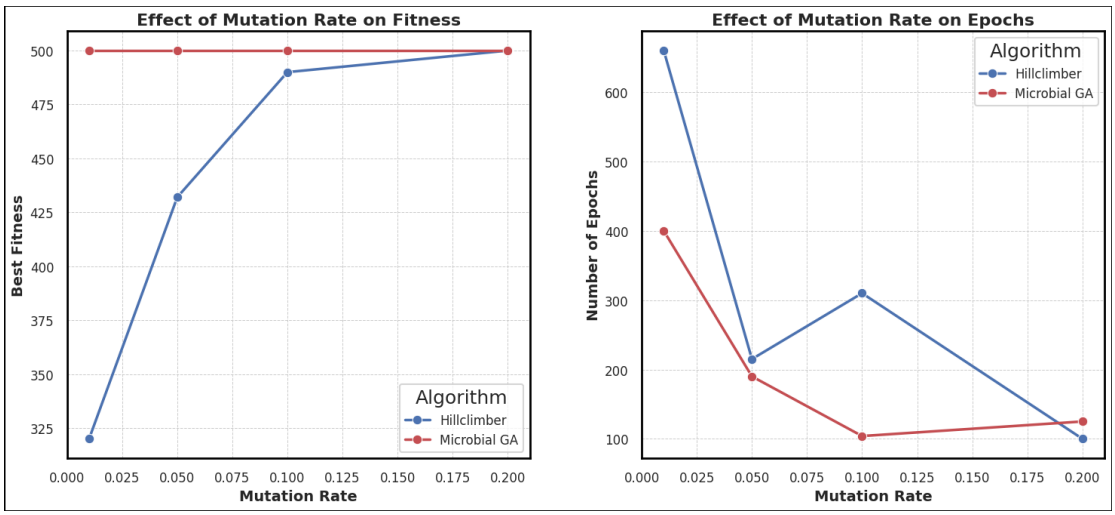


Figure 7 Results of both Algorithms applied to CartPole task

4.1 Hillclimber Algorithm

The Hillclimber algorithm's performance is highly sensitive to mutation rates:

- Low Mutation Rates: Lead to poor performance due to insufficient exploration, causing the algorithm to get stuck in local optima.
- Moderate to High Mutation Rates: Improve performance by balancing exploration and exploitation, allowing the algorithm to escape local optima more effectively.

Best Parameter: A mutation rate of 0.2, offering the best fitness and quickest convergence, suggesting it optimises exploration and exploitation.

4.2 Microbial GA

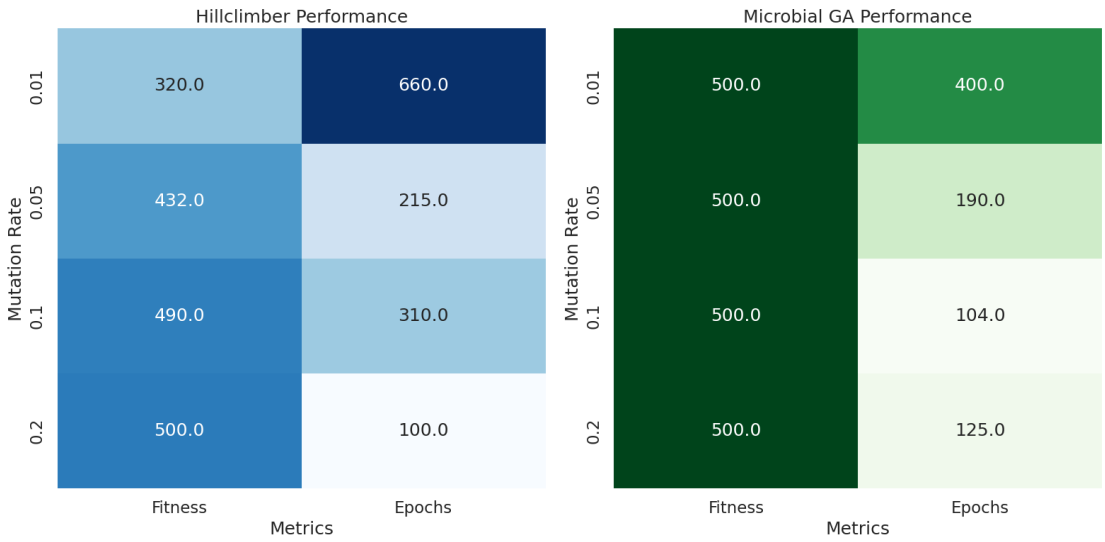
The Microbial GA shows consistent robustness across different mutation rates, achieving optimal fitness with varying convergence speeds[4]:

- Low Mutation Rates: Although it achieves optimal fitness, the convergence is slow due to limited diversity.
- Moderate Mutation Rates: Provides an efficient balance between mutation and selection, resulting in faster convergence.
- High Mutation Rates: Slightly reduced efficiency due to excessive randomness, though it still maintains overall robustness.

Best Parameter: A mutation rate of 0.1, ensuring the fastest convergence with an optimal balance of exploration and exploitation.

4.3 Comparative Analysis

Both algorithms demonstrated the critical role of mutation rates in evolutionary algorithms[1]. The Hillclimber algorithm showed a clear improvement in performance with increasing mutation rates, highlighting its reliance on sufficient exploration to escape local optima. In contrast, the Microbial GA achieved consistent optimal fitness across all mutation rates, indicating its robustness and effectiveness in maintaining diversity and avoiding premature convergence. Furthermore, it consistently held these higher fitness levels with a lower number of epochs, suggesting that it is more efficient. A heat map is shown below highlighting the difference in performance.



4.4 Hypothesis Inference from Results

Based on the experimental results, we accepted the null hypothesis. The Microbial GA consistently achieved optimal fitness across various mutation rates compared to the Hillclimber algorithm. This indicates a significant difference in performance between the two algorithms, with the Microbial GA demonstrating superior robustness and efficiency[5].

4.5 Implications and Future Work

The Hillclimber algorithm also demonstrated the capability to achieve high performance, particularly with finely tuned higher mutation rates. This suggests that while Microbial GA is robust across different conditions, the Hillclimber can still be highly effective with proper parameter adjustments.

Future research could focus on hybrid approaches that leverage the strengths of both algorithms. For instance, combining the robustness of the Microbial GA with the fine-tuning capabilities of the Hillclimber could potentially yield an even more efficient algorithm. This hybrid approach could balance the broad search capabilities of the Microbial GA with the detailed local search enhancements of the Hillclimber, leading to faster convergence and improved performance.

Additionally, it would be beneficial to test these algorithms in more complex environments with varied fitness landscapes. Such testing could provide further insights into their adaptability. More intricate environments would challenge the algorithms' ability to handle diverse and unpredictable conditions, thus validating their robustness and efficiency further[6].

5 Conclusion

This investigation highlighted the comparative performance of the Hillclimber and Microbial GA algorithms in the CartPole-v1 environment under varying mutation rates. We accepted the null hypothesis as the results demonstrated that the Microbial GA consistently outperformed the Hillclimber, achieving optimal fitness more reliably and converging faster.

7 References

[1] Thomas B¨ack, David B Fogel, and Zbigniew Michalewicz. Handbook of evolutionary computation. Release, 97(1):B1, 1997.

[2] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics, SMC-13(5):834–846, 1983.

[3] DE Goldberg. Genetic algorithms in search, optimization and machine learning. addison-wesley longman publishing co., inc. 1989.

[4] Inman Harvey. The microbial genetic algorithm. In Advances in Artificial Life. Darwin Meets von Neumann: 10th European Conference, ECAL 2009, Budapest, Hungary, September 13-16, 2009, Revised Selected Papers, Part II 10, pages 126–133. Springer, 2011.

[5] Yohannes Kassahun and Gerald Sommer. Efficient reinforcement learning through evolutionary acquisition of neural topologies. In ESANN, pages 259–266, 2005.

[6] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. science, 220(4598):671–680, 1983.

[7] Geoffrey F Miller, Peter M Todd, and Shailesh U Hegde. Designing neural networks using genetic algorithms. In ICGA, volume 89, pages 379–384, 1989.

[8] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. nature, 323(6088):533–536, 1986.

[9] Melanie Mitchell. An introduction to genetic algorithms. MIT press, 1998.

8 Appendix

See AIAB_Coursework_No2.ipynb for the code used in this report.