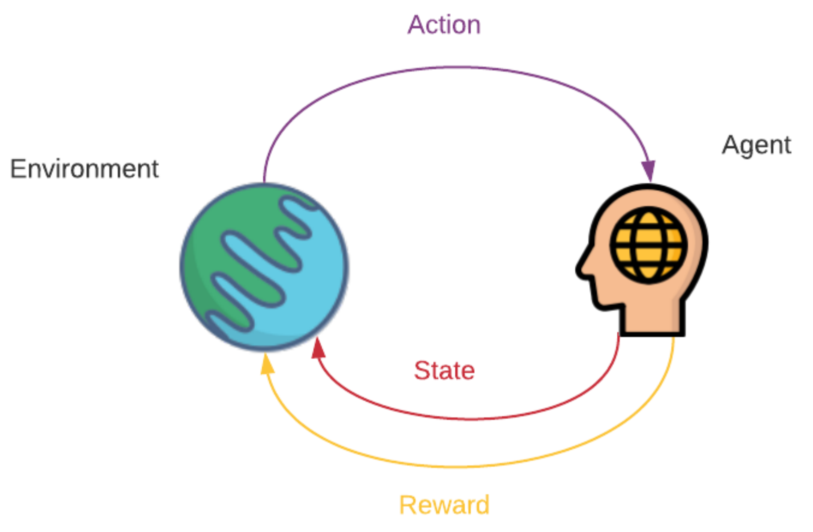
Exploring evolutionary optimization of an RL agent on the Cartpole-v1 task

Abstract

The Cartpole-v1 problem is an OpenAI gym [2] task designed by Sutton and Barto in order to test reinforcement learning (RL). RL involves learning through the interaction between an environment, and an agent (see Figure 1). The agent we have used to solve the cartpole task is a single layer perceptron optimized by a microbial genetic algorithm (GA). The GA tunes the perceptron's weights and biases to optimize the parameters of the perceptron [6] in order to learn desirable behaviour which keeps the cartpole upright for as long as possible, and therefore maximises the total reward. Our microbial GA makes use of sexual recombination, uniform crossover, creep mutation, demes, and steady-state selection. We identified the best genotypes at different stages of optimization, in order to investigate their behaviour and fitness over a number of iterations.

Figure 1. The interaction between an agent and environment in RL



Introduction

Reinforcement Learning (RL) is a branch of machine learning that focuses on how agents can learn to optimize their behaviour through interacting with the environment, choosing actions, and receiving corresponding rewards. The agent learns the quality of the action based on the reward, and through the sum of rewards for a sequence of actions (the cumulative reward), finds the most appropriate sequence of actions to solve the given problem. (Sutton and Barto, 2018).

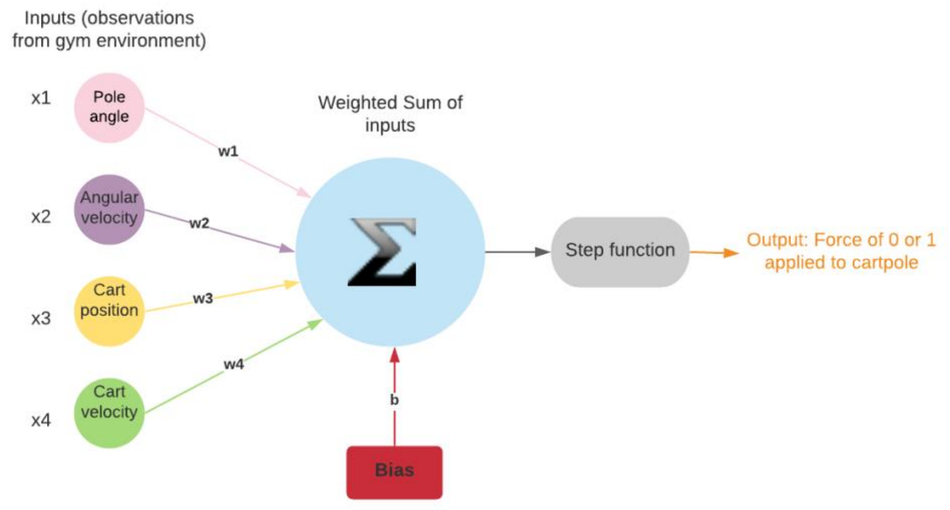
The Cartpole problem consists of a pole of length 1m and mass 0.1kg, attached to a cart of mass 1kg. (Sutton, Barto et al,. 1983). The state at each time step is represented as a vector: cart position x, cart velocity $\dot{x}$, pole angle θ and angular velocity $\dot{\theta}$. An episode terminates when the following conditions are met (Brockman et al,. 2016): the episode length (500) is reached, the cart position leaves the interval [-4, 4] metres, the pole angle leaves the interval [-0.21, 0.21] radians. The only possible actions are moving left or right with a force of +1 or -1. At each time step, the controller receives the cart-pole's state vector at that instant. If the pole falls or the cart hits the track boundary, the controller receives a failure signal, the cart-pole system (but not the controller's memory) is reset to its initial state. [3]

RL tasks are often approached using a policy gradient approach, where the process demonstrated in Figure 1 is repeated until learning stops. However, directly updating the policy at each iteration to find the optimal policy significantly relies on choosing optimal parameters. Genetic algorithms can help choose these parameters, and evolutionary strategies are easier to implement as they do not require backpropagation [11].

Our controller has been optimized using a microbial genetic algorithm [4]. This is a variant of tournament-based selection, where upon evaluating two random individuals, the offspring replaces the losing genotype in the population. This is a steady state rather than a generational approach. [9]

In this report, we explore a gradient-free approach to solving the Cartpole task, and evaluate its performance over multiple runs finding that an genetic algorithm can identify an optimal controller and policy in this control task.

Figure 2. The single layer perceptron used as the architecture for our agents.



Methods

To attempt to solve the Cartpole-v1 task, we used a population of genotypes, each of which were implemented as an agent in a single layer perceptron, as seen in Figure 2. Each genotype has 5 genes, and a population has 15 genotypes. The genes are represented as continuous values from a normal distribution with mean=0, and standard deviation 0.1. Genes encode for the weights and biases in the perceptron, which then specify the 'policy' for behaviour.

We have employed a microbial genetic algorithm (GA) [4] to optimize the single layer perceptron (agent). Our microbial GA exhibits sexual recombination, demes, creep mutation, uniform crossover and steady-state replacement. The pseudocode for the microbial GA used can be seen below (Algorithm 1).

```
Algorithm 1: Microbial Genetic Algorithm
  Result: Genotype with the highest fitness
  initializatize random population P;
  for each individual in P do
      assign each individual a position x in P;
  end
  for each tournament do
      for each genotype in P do
          g1 = random individual in P;
          g2 = random individual within 5 spaces of g1;
          winner, loser = compare(g1, g2);
          loser = crossover(winner, loser);
          loser = mutate(loser)
      end
      update population;
  end
  return the fittest individual;
```

A more detailed outline of our controller's approach to solving the Cartpole-v1 task is given below:

1. Initialise a random population of genotypes
2. Pick 2 individuals at random
3. Compare the fitness of the individuals by passing them to the environment, where a simulation is run for each. Let the individual with higher reward be the 'winner', the other is the 'loser'.
4. Crossover the loser with p =0.5 from the winner
5. Create offspring by mutating the loser
6. Return the winner unchanged and new offspring back into the population
7. Repeat for the number of individuals in the population
8. Repeat steps 2-7 for a number of tournaments
9. Return the individual with the highest fitness

Fitness Function

The fitness function for our GA is the total reward returned, which is calculated by running a simulation of the cartpole task for the agent with a specific genotype. For each episode the cartpole stays upright, a reward of 1 is added to a maximum of 500.

Mutation

We have used a variation of insertion mutation, where a random point in the loser is selected and replaced by a value in the range of the original gene. [9] (p20)
In order to introduce diversity in the population and explore the fitness landscape, at each iteration we must mutate the losing genotype. Since our genes are continuous values, this takes the form of a creep mutation. For each mutation, a random number is generated between 0 and 1, and if this exceeds the mutation rate threshold, gaussian noise is added to the genes with mean=0, and std=0.1.

Crossover

We have used uniform crossover, where each gene in the winning individual has a 50% probability of being passed onto the losing individual (offspring). This produces high variation between parents and offspring and encourages exploration of the search space. [9] (page 19)

Evaluation

To evaluate the performance of the controller, we ran 3 runs of 100/150/200 tournaments and averaged the results depending on the number of tournaments used, to ensure that the behaviour of the controller was not a statistical fluke.

To show that the behaviour of the controller is changing, we recorded the genotypes with the best fitness in the population at different stages in the optimization process and plotted their behaviour and fitness in simulations of the cartpole task.

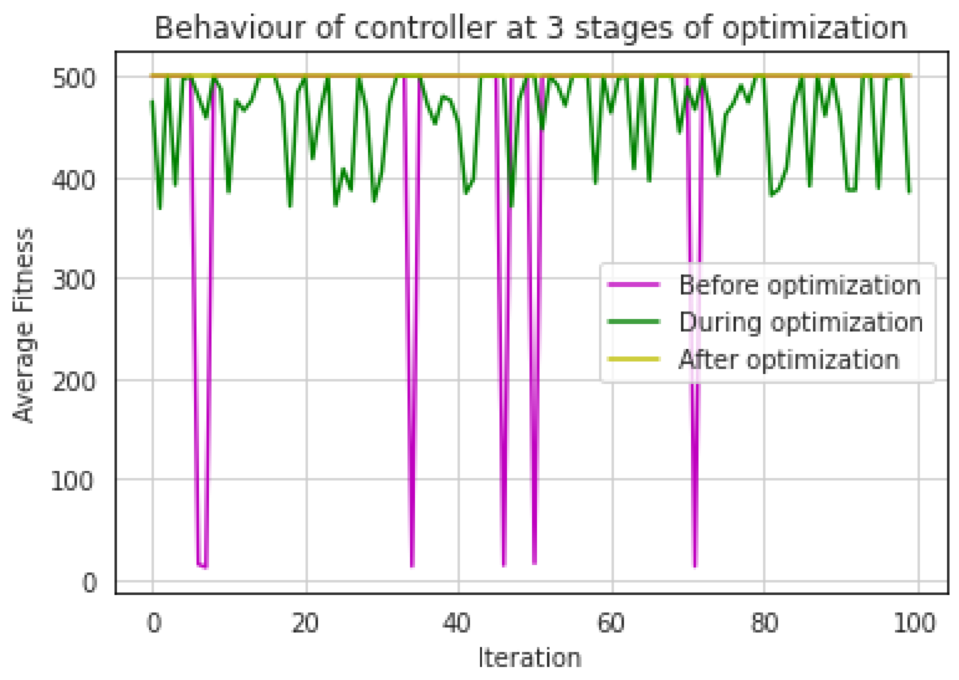
Results

The figures given below demonstrate the evolution of behaviour over 100 tournaments, as the fitness is steadily increasing and on all 3 runs of our GA, the controller reached a a best fitness of 500.

In Appendix 1, you can see how the best fitness improves over 100 tournaments, particularly through a sequence of plateaus then jumps in fitness. The mean fitness of the population also increases over 100 tournaments but does fluctuates significantly. Appendix 2-3 show the average angular velocity/angle after each tournament respectively, and provide clear evidence for the convergence to an optimal policy. As the average fitness starts to plateau in Appendix 1, the behaviour of the controller in terms of angle and angular velocity also flattens.

Figure 3 exemplifies the progression of behaviour over evolutionary optimization as before optimization (pink line), the fitness of the best genotype is still able to reach 500, but dramatically peaks and plummets, to a minimum of ~20. During optimization, there is still fluctuation in the fitness, but this does not fall below 350, showing significant improvement and more consistency. Finally, the best genotype in the population after optimization has a stable fitness of 500, demonstrating that this controller has solved the Cartpole-v1 task.

Figure 3. Plot comparing the average fitness of the best controller at different stages of optimization.



Our algorithm also outperforms a random agent, as seen in Figure 5, as the average fitness of our EA agent diverges from that of the random agent after only a few tournaments. The random agent also struggles to even reach a fitness of 100. We can therefore be sure that there is optimization of the weights and biases in the perceptron, as if not the behaviour would more closely resemble that of the random agent.

Figure 6.a depicts the behaviour of the best controller in the initial population before optimization and does not survive longer than 10 episodes. The behaviour is described through the pole angle, pole angular velocity, cart position and cart velocity. From Fig 6a, it is clear that the controller has not developed any sophisticated behaviour. Fig 6b shows more complex behaviour, however the agent during optimization does not survive past 350 episodes. Finally, Fig 6b demonstrates the behaviour of a successfully optimized controller, reaching 500 episodes while staying upright. The change in the plots from Fig 6a-c shows the emergence of more sophisticated behaviour, through a pattern of oscillations.

Figure 4. Plot showing the change in best fitness and average fitness (total reward) in comparison to the 'solved' threshold.

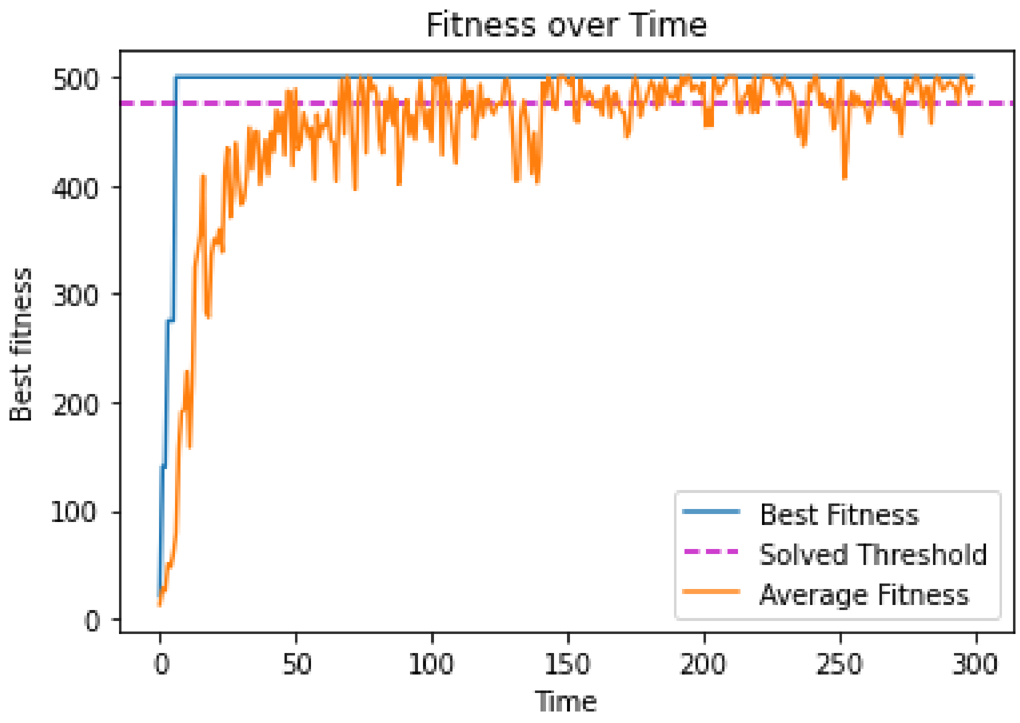


Figure 5. Comparison between the evolution of behaviour on our GA and a random agent on average fitness.

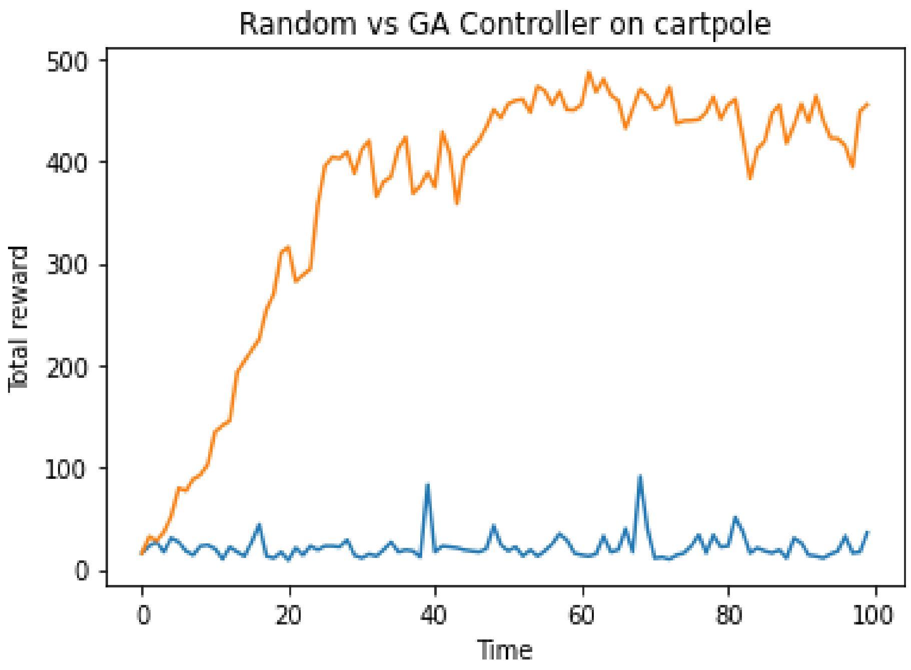
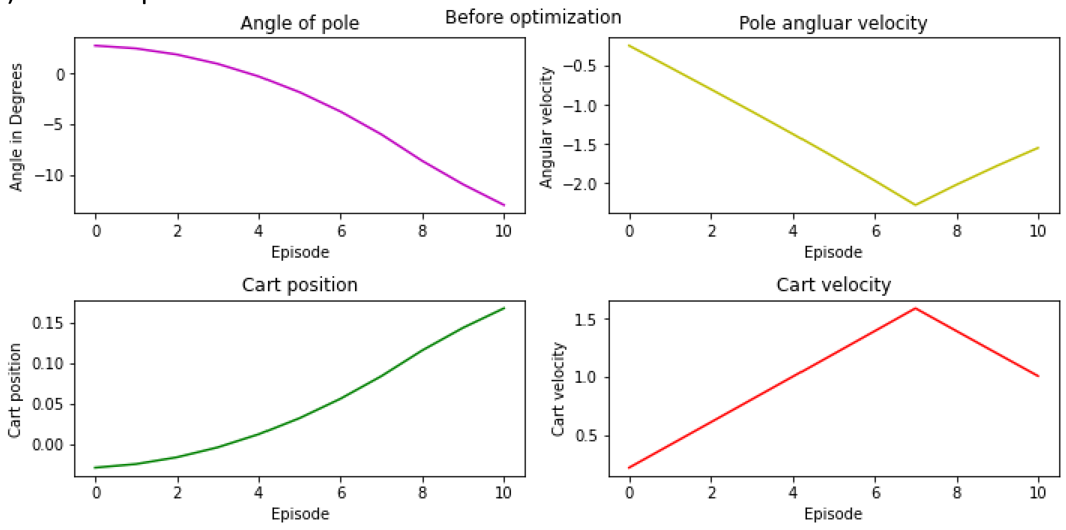
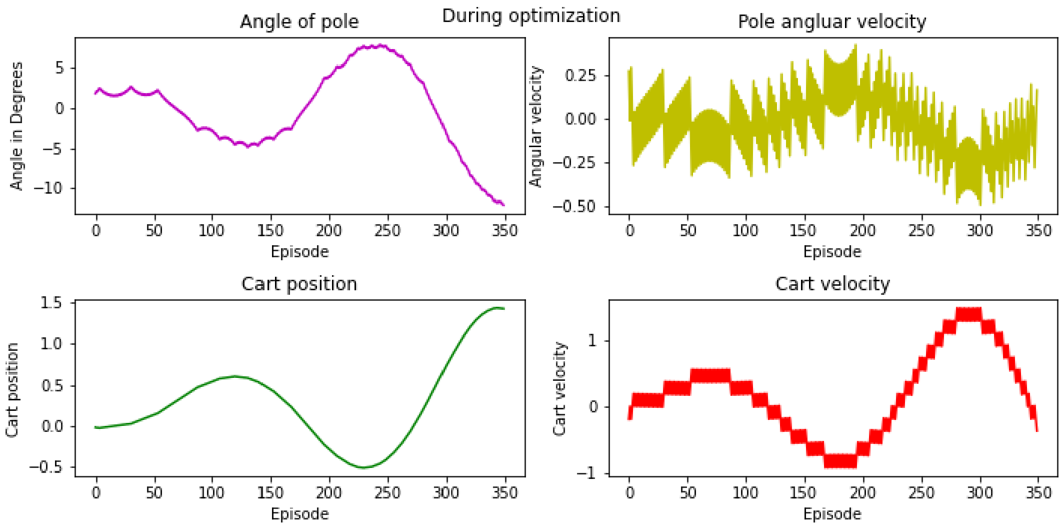


Figure 6. The behaviour of the best controller at 3 stages during optimization. Note, during optimization refers to at T/2 where T=number of tournaments.

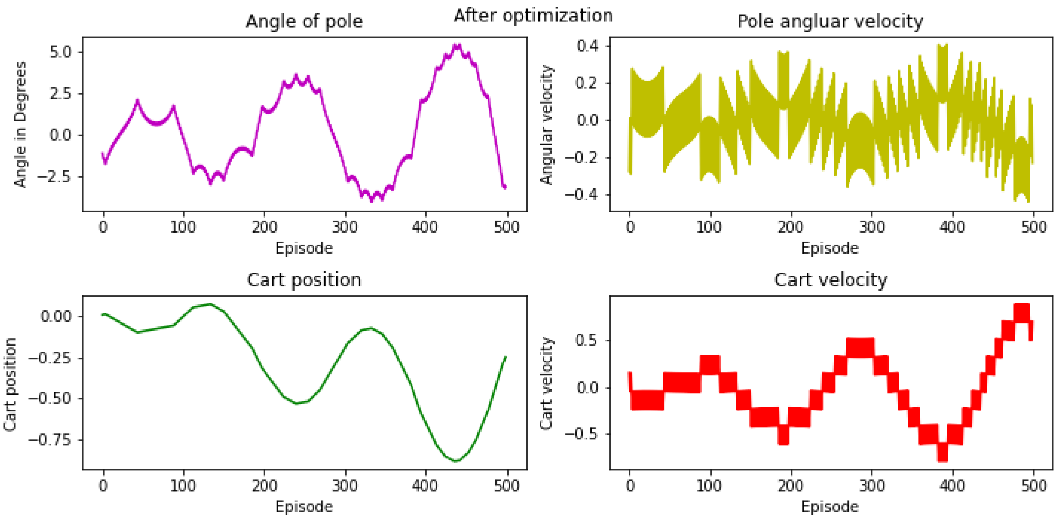
(a) Before optimization



(b) During optimization



(c) After optimization



Analysis and Discussion

By comparing the total reward for a random agent, and an agent controlled by our GA, it is clear that using evolutionary optimization is a useful tool in solving the cartpole task.

The Figures given above show a clear evolution in the behaviour of our controller in the cartpole task. We can see that the maximum fitness of the population very quickly converges on 500, the maximum reward, and the mean fitness also converges near to 500 after X tournaments. The microbial GA is therefore successful at optimizing an agent to solve the Cartpole-v1 task.

However, although we show successful optimization, we cannot claim that we have 'solved' the cartpole task, as per the definition [2]. Further work is needed to achieve consistently high average total reward of 475 (the threshold value).

There are a number of improvements which could be made in further research including the use of hidden layers in the agent's architecture and exploring the effect of parameter choice on the performance of the microbial GA. We believe that hyperparameter tuning of our microbial genetic algorithm would produce a more stable performance in average fitness, however we could not explore that here due to the time constraints.
Mutation rate could be particularly important as one challenge in RL is the trade-off between exploration and exploitation and mutation rate is significant in balancing the diversity and heredity of a population of genotypes, which is directly responsible for the level of exploration/exploitation.

Our model used creep mutation by adding gaussian noise to the gene encoding the weights and biases, however this may be altered to improve the exploration of the fitness landscape by adding a value proportional to the total reward for that agent [10]. Using a mutation operator in this way would allow a more controlled approach to the exploring and exploiting maxima in the search space. A variable mutation rate would also support a controlled exploration of the search space.

Summary and conclusion

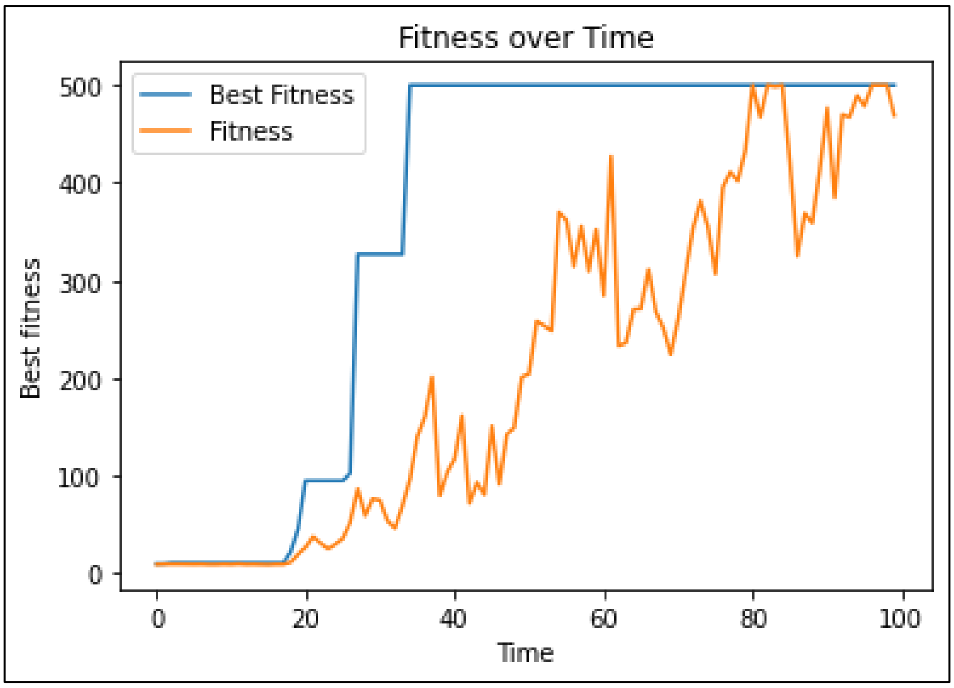
The increase in total reward over subsequent tournaments has demonstrated how a microbial GA is successfully optimizing the weights and biases of the perceptron to solve a control problem. This report has shown the value of evolutionary optimization strategies in reinforcement learning, specifically that of the microbial genetic algorithm. We have seen how optimizing RL though an evolutionary algorithm is useful and effective in solving the Cartpole-v1 OpenAI task. Combining EA and RL should continue to be explored in order to find efficient and high performing solutions to control problems, such as the Cartpole task.

References

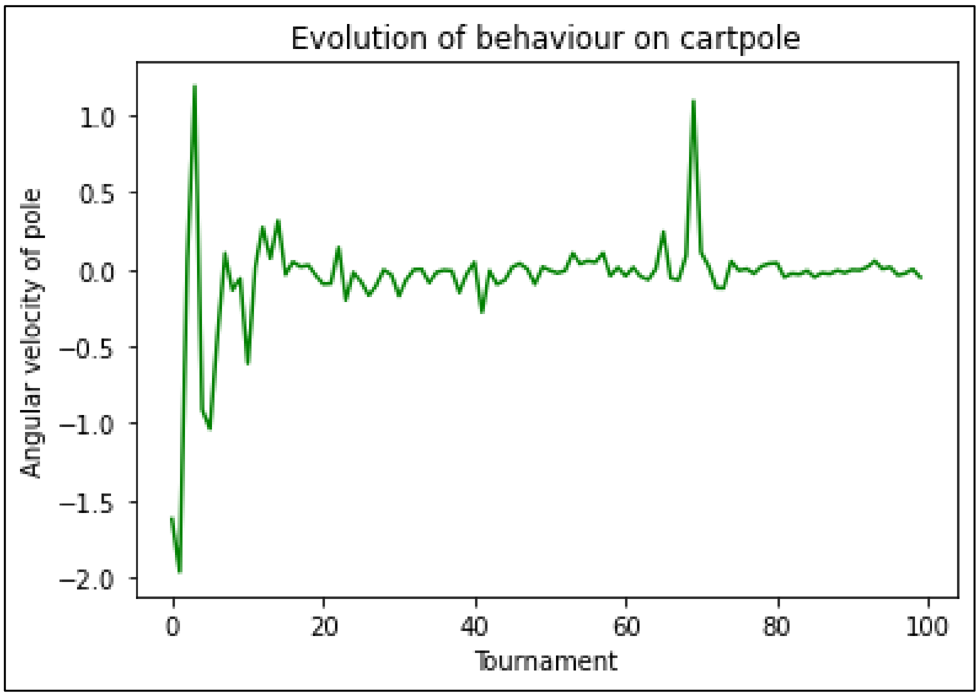
1. Sutton, R. and Barto, A. (2018). ‘Reinforcement learning: an introduction’. The MIT Press, Cambridge, Massachusetts, 2nd edition
2. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym
3. Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). ‘Neuronlike adaptive elements that can solve difficult learning control problems’. IEEE transactions on systems, man, and cybernetics, (5):834–846
4. Harvey, Inman. (199). The Microbial Genetic Algorithm.
5. Ahmed Fawzy Gad ‘Practical Computer Vision Applications Using Deep Learning with CNNs’. Dec. 2018, Apress, 978–1–4842–4167–7
6. Leung, F. H., Lam, H. K., Ling, S. H., & Tam, P. K. (2003). ‘Tuning of the structure and parameters of a neural network using an improved genetic algorithm’. Neural Networks, IEEE Transactions on, 14(1), 79-88.
7. Huang, H. X., Li, J. C., & Xiao, C. L. (2015). ‘A proposed iteration optimization approach integrating backpropagation neural network with genetic algorithm’. Expert Systems with Applications, 42(1), 146-155.
8. Brown, B., and Zai, A. (2020). ‘Deep reinforcement learning in action.’ Manning, New York. Chapter 6, Section 6.3.
9. Beckerleg, M. (2012). ‘Evolution of Robotic Controllers Using Genetic Algorithms.’ PhD, AUT University.
10. Agrawal, D., Sadagopan, K, R., Tlili, F. (2020) ‘Re-Evolutionary Algorithms: Combining Policy Gradient Methods in Reinforcement Learning with Evolutionary Algorithms’.
11. Salimans, T., Xo, J., Chen, Xi., Sidor,. Szymon, S., Sustkever, I. (2017). ‘Evolution Strategies as a Scalable Alternative to Reinforcement Learning;. arXiv preprint arXiv:1703.03864

Appendices

1.

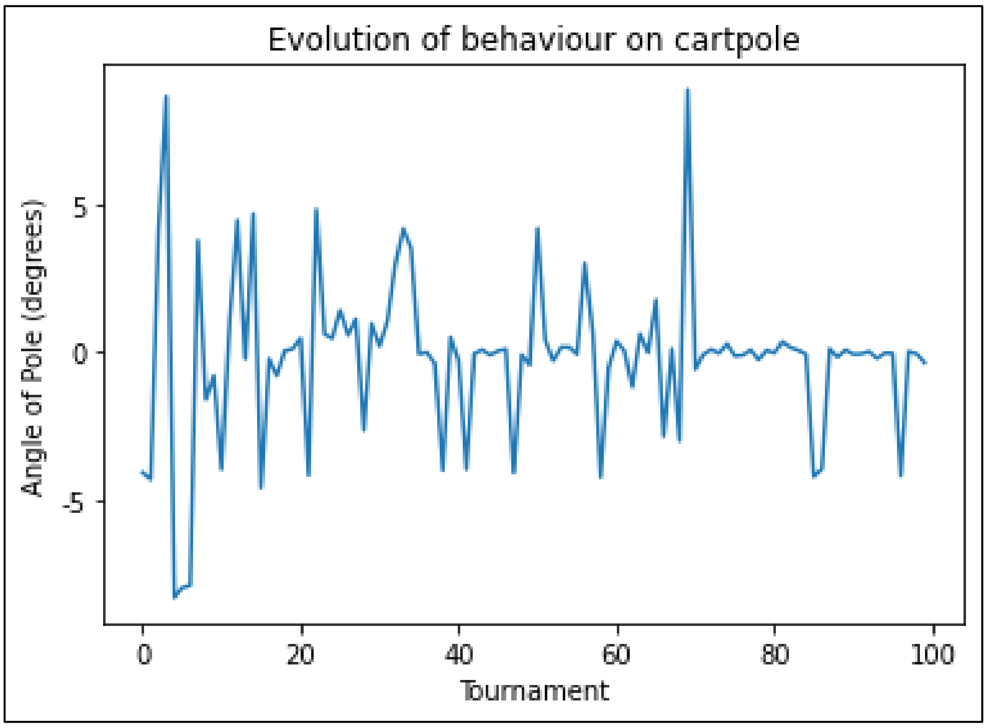


2. –



Evolution of behaviour on cartpole
Angular velocity of pole
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
20
40
60
80
100
Tournament

3.



Evolution of behaviour on cartpole
Angle of Pole (degrees)
5
0
-5
0
20
40
60
80
100
Tournament