Examining the evolution of behaviour in OpenAI's cart pole gym environment using an genetic algorithm

215758
G6042: Acquired Intelligence & Adaptive Behaviour

May 28, 2021

Abstract

OpenAI provides several environments to solve a given problem, we will be using the cart pole environment to examine the evolution of behavior with a genetic algorithm. The cart pole problem is where a cart on a frictionless track has to keep a pole within the cart upright for five hundred frames. To solve this problem, we will use a microbial genetic algorithm to generate and optimize weights for a controller neural network to solve this problem. In our study, the patterns of behavior which arose showed the controller using waves to counterbalance the pole falling. We also saw that the algorithm tried to mitigate the first problem where the pole falls too far, then the problem of position arose due to the use of velocity to keep the pole upright which then was solved. In future studies we should analyze how effecting genetic algorithms are for more complex environments, additionally, there should be another about how effective it is versus the traditional backpropagation approach.

1 Introduction

The cart pole problem consists of a cart on a straight rail, with a pole attached to the cart with an unactuated joint, the job of the cart of to keep to pole upright, if the pole passes 15° it will fall, or if the cart steers too far away from the starting point, the task will be failed. This problem is typically solved using a reinforcement learning algorithm, such as a multi-layered perceptron, however, we will be using a genetic algorithm to solve this problem.

A genetic algorithm (GA) is a method for optimization to explore a problem space / fitness landscape. It follows the model of evolution where the fittest survive, share their genes via a crossover, while the loser's genotype mutates to try to survive and be the recipient of crossover [3][2].

We will be using a microbial genetic algorithm with demes selection, the general flow of this algorithm starts with generating a population and assess their current fitness'. Next, during n amount repetitions, a demes tourney is called which returns a winner and loser determined on their fitness', the winner gives some genes from their genotype to the loser depending on some probability; The loser is then mutated also depending on some probability too. The fitness of the loser is then reevaluated and added back into the current fitness' [2].

The demes selection is a method where only a certain window around the first randomly selected genotype from the population, this is due to how in nature not every animal competes with all animals around the world, but only in a certain local area; the changes in the best genotype is slowly propagated over time to the areas of the global population. This can be described as an island model due to the features described [4].

Changing the parameters can change the behavior of the evolution, such as large jumps occurring due to having a good level of probability for mutation and crossover, experimentation is key, there are certain values where it causes the evolution progress to halt completely or stop as a whole. There are usually recommended values for these probabilities, one of which is $1/gene\_length$ for the mutation rate. Changing these features can give great benefits or problems [1], but we will experiment outside of this.

The best genotype from the population will be mapped onto an agent controller which generates a neural network using the values of the genes as weight, this controller is then used to complete the task through the OpenAI gym environment returning values which the neural network take input from which will then process and return an output to control the cart.

We will be using the OpenAI gym to implement the problem and to solve it, it is an open-source library that provides several environments for a controller to train in. Each environment has an observation space that provides a boundary for each value returned from an observation. The observation will return values per frame that are relevant to the environment; using the cart pole task as an example, it would return the position and velocity of the cart, additionally the angle of and angular velocity of the pole. It will also return a reward of one per frame, this would be used for the neural network however we will be using this for our fitness function instead.

This study is to examine the evolution of behavior of the cart pole task using a microbial GA, we are expecting to see jumps in fitness and then stabilize around halfway through the evolution progress. We will show the progress of the evolution of a controller at the start, middle, and end of this process, and theorize that the algorithm will try to first try to overcome the angle being too large, then overcoming the position which is caused by the aforementioned problem.

2 Methods

Using the OpenAI gym library environment, we will use the cart pole v1 environment and will be using a microbial GA to optimize the weights of the agent controller neural network to solve the problem. The maximum fitness which can be reached is five hundred; one fitness is gained per frame, and when the video of the controller in the environment is rendered, there it will be ten seconds.

The fitness function runs the simulation until either it fails due to the pole falling past 15° or the cart driving too far away from the center, otherwise, it has to keep it balanced for all five hundred frames which means it was a success. Each frame where it hasn't exceeded a maximum, a reward is given; our fitness function takes the sum of these values and uses it as the measurement of fitness.

Our GA main functions are unnecessary to describe, however, the parameters are as they are the driving force of the algorithm and if used incorrectly it can stagnate progress. We will use a population of 25, and 125 epochs, when multiplying these values we can calculate the number of tourneys which will commence: $25 * 125 = 3125$. Additionally, we will use a window size of 3, too large and diversity will lessen, too small and the best genotypes will not propagate their genes throughout the population. We will use a crossover and mutation probability of 5%, these values were obtained through experimentation, but altering them too much or too little will stagnate progress towards a high fitness solution.

The agent creates a neural network controller with an input of four and an output of one value. There is an input of a genotype with five genes, these genes are ten splits, reshaped, and used for the weights and biases of the neural network. This agent and controller are used within the simulation and their fitness is assessed using the method described in the aforementioned.

We will be simulating this problem with the OpenAI gym, however, to visualize this problem using Google Colab, we will be using the ***gym-notebook-wrapper*** library to render videos to see the progress quickly, there will be an example video attached with this paper showing a successful example of a controller.

Each tourney and controller plot, except the evolution of the controller, will use a mean of five tourneys to show the rough path the data takes between them, this is to also show that it reliably reaches the highest fitness of five hundred over several runs.

3 Results

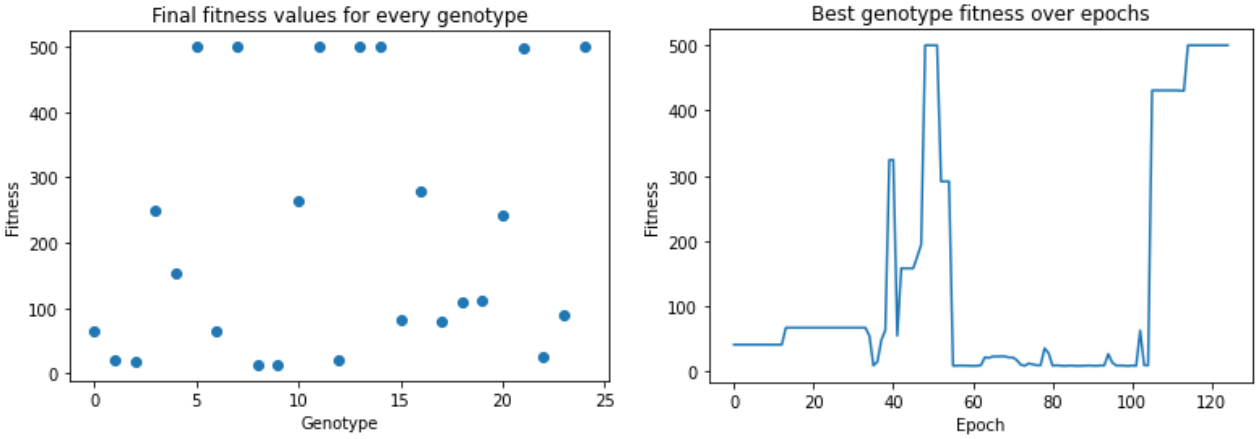

Figure 1: Population fitness result and best genotype fitness history.

First, figure 1, this shows after a full evolution of the genotype mapped onto the agent controller, there are many 500 fitness genotypes' final values, meaning that the current GA parameters are working well with this solving this problem. Looking at the best genotype fitness history, we and see large jumps of fitness, and where there are declines, this is where the genotype's genes have changed via mutation or crossover. Overall it shows the genotype evolves rather than instantly finding the solution for the problem just via brute force.

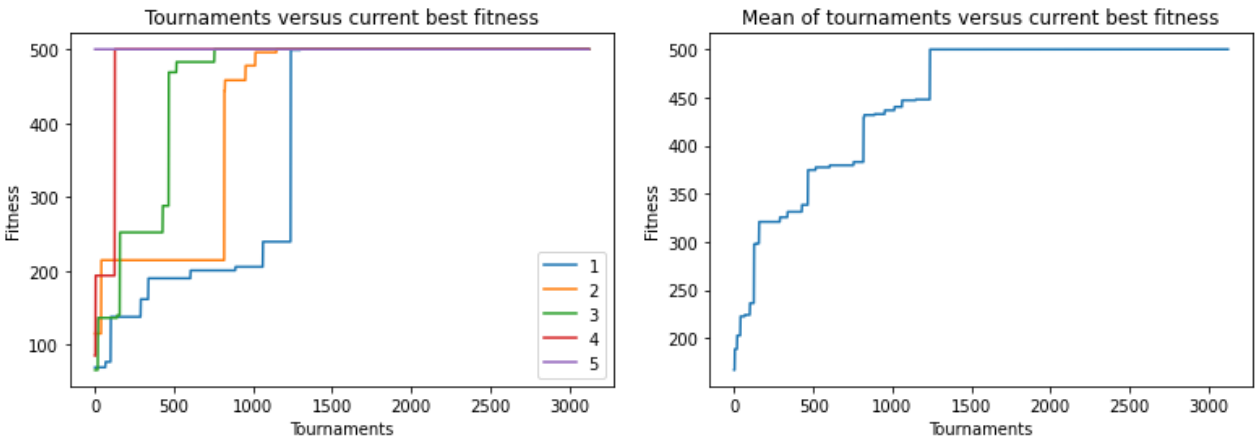

Figure 2: Five tournaments vs fitness and the mean of them.

Next, in figure 2, we see the global evolution of fitness from 3125 tournaments using five separate populations. The fifth population instantly found the solution due to luck, it did not evolve, however, the other populations have due to the climb from a lower fitness value to a higher one. In the mean plot, we can see there are three main big jumps with a small steady increase of fitness before the jumps, then it flattens out when the peak is reached. The evolution is reliant on these big jumps otherwise it would take many more thousands of tourneys until it reaches the same fitness as what we currently are seeing in these plots.

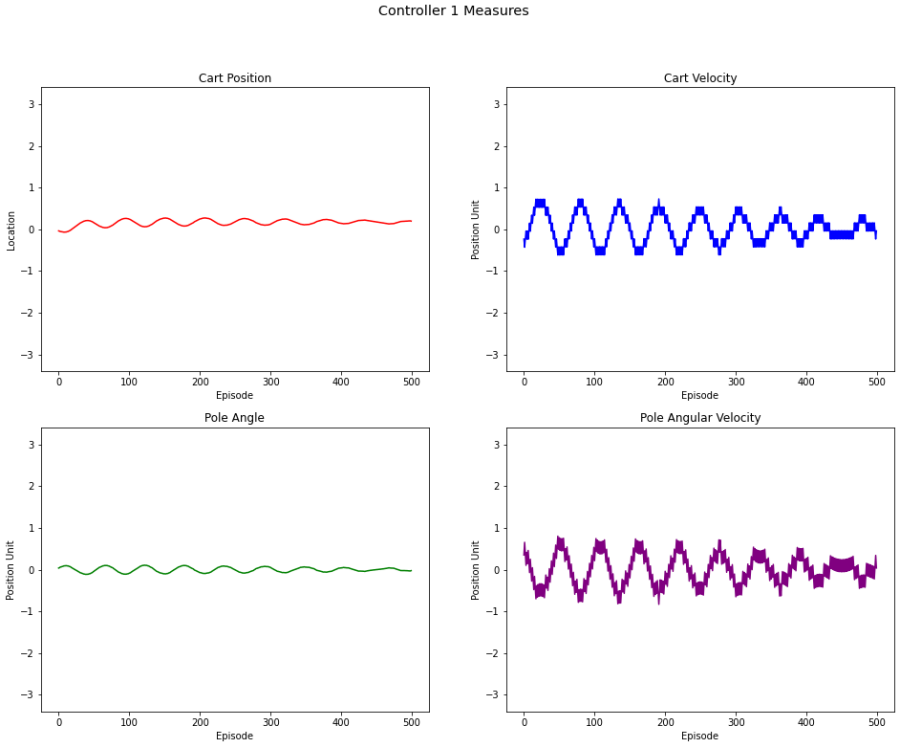

Figure 3: Successful controller with large fluctuations

Figure 3 shows the first controller created from the resulting genotype in figure 2, the part position and pole angle are mirrored, this displays a relationship between them which lead this to become a successful controller. The same occurs with cart velocity and pole angular velocity, there is a link between them as they are both velocities that will be affected by almost the same forces. The cart velocity is much more square with each segment, while the angular velocity had more curves and distortion of the cart velocity, especially as the cart velocity affects the pole angular velocity. There is a waveform showing within both, where there are sparse amount of waves however they are much larger in height and width than other controller results which we see later. These waves are created as this is what keeps the pole balanced by repeating the opposite action to stop the pole from falling.

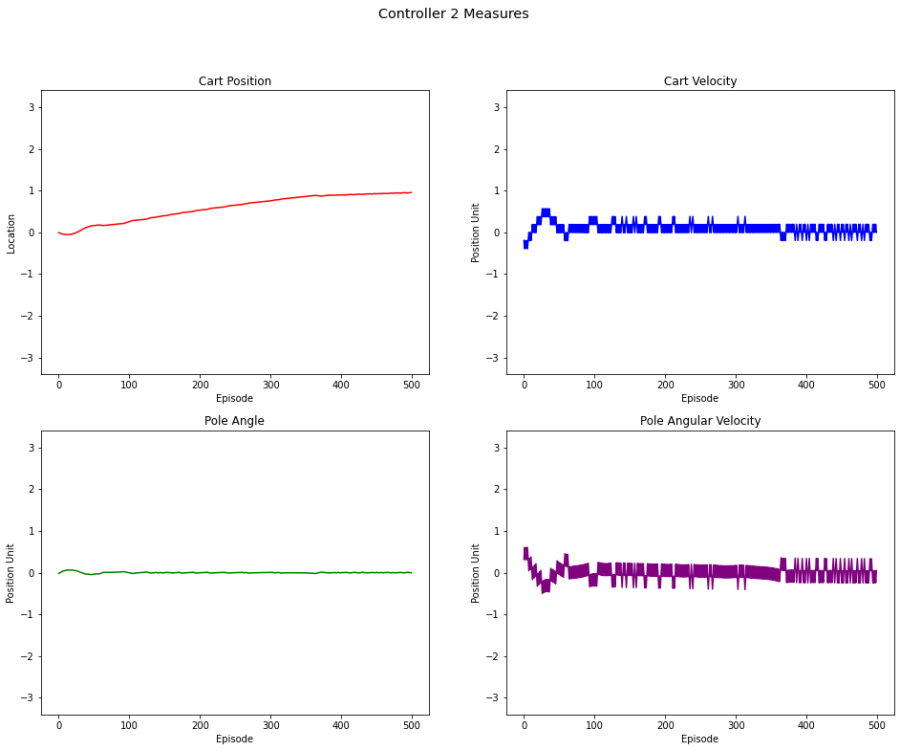

Figure 4: Successful controller with fast fluctuations

Continuing this, figure 4 shows a very different result to the former. The pole angle is almost a straight line throughout the simulation, however unlike figure 3, the cart position does not mirror the pole angle, it found success via different means. This controller was successful due to the rapid change of direction to balance the pole with a small velocity in each direction, to counterweight the pole quickly. The cart position was quickly moving to the end of the observation space meaning that it would fail, however, it stabilizes from ≈ 350 on-wards.

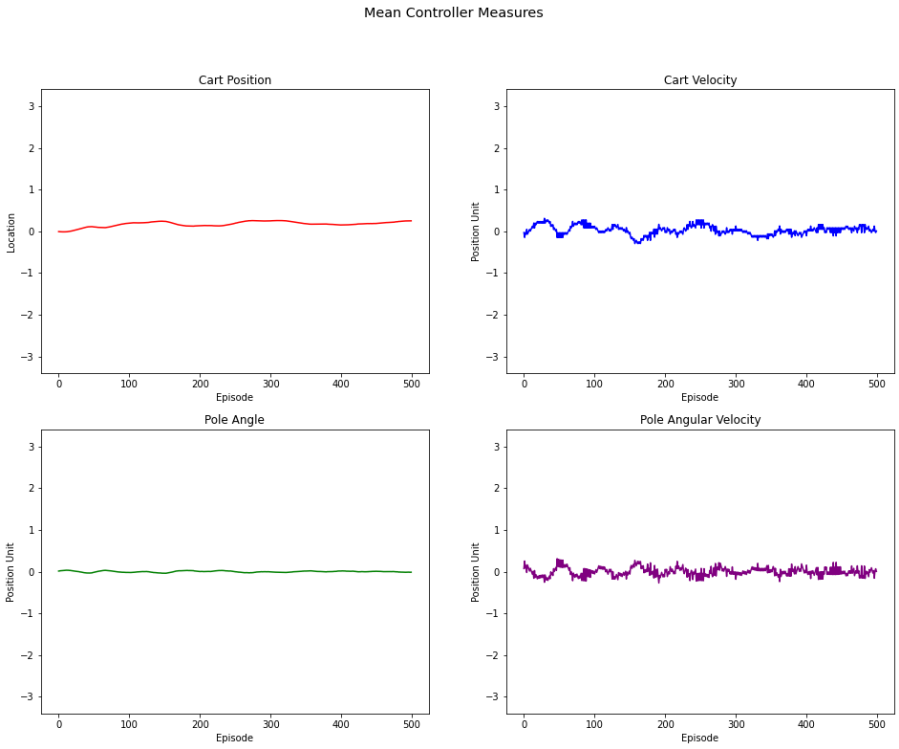

Figure 5: Mean of five successful controllers

The mean of the five runs is shown in figure 5, the pole angle is rather straight, same with the cart position but it isn't an exact mirror, there are larger bumps within it, this could be due to the previous figure having a climb up affecting the final result. The cart and pole velocities seem to be mirrored, showing there is a direct link, which is true due to one velocity will affect the pole within it, causing a counter opposite motion in the pole. The waves seem to be much smaller than in figure 3, they seem to be more sporadic, however near the end it seems like the velocity stabilizes.

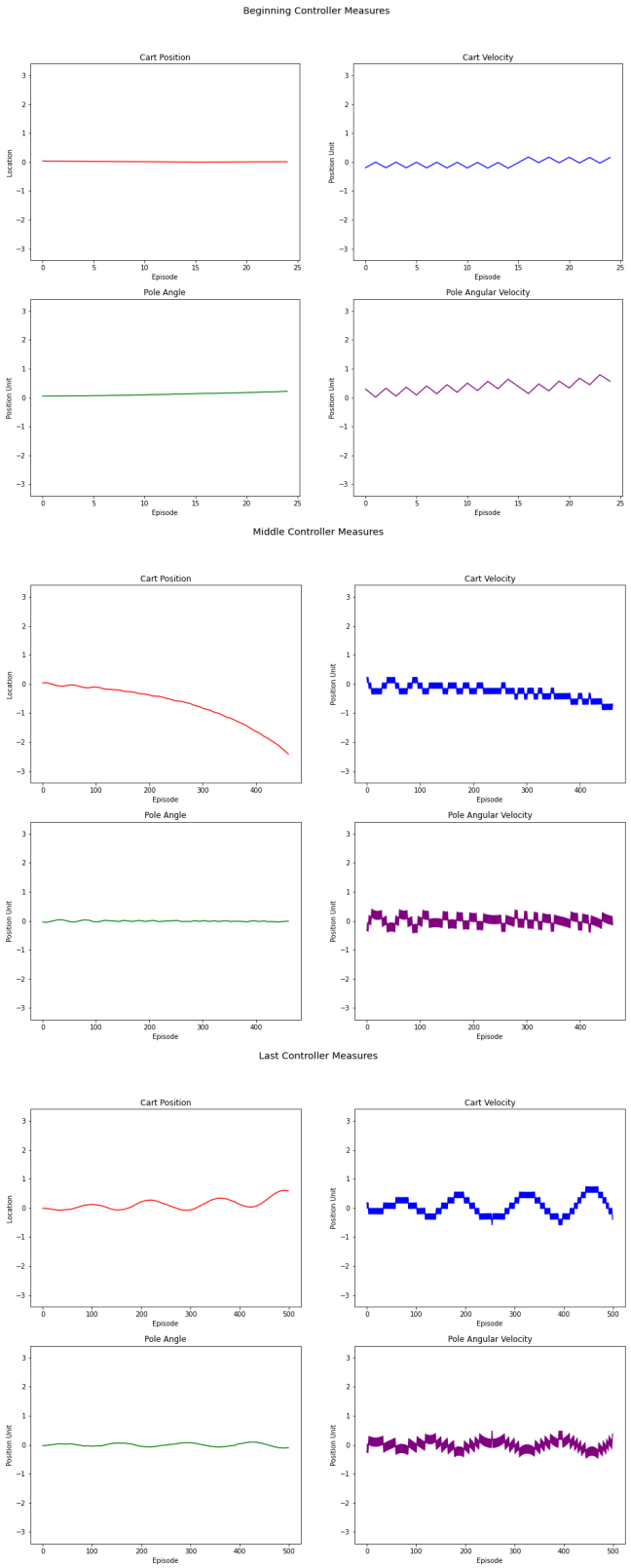

Figure 6: Process of controller evolution.

In figure 6, the plot labeled with "beginning", shows the best genotype mapped onto the agent controller from the initial population, the next figure is labeled with "middle", it shows the progress halfway through the epochs, and finally, with the plot labeled with "Last", it shows the final controller result.

In the beginning, the controller lasted 25 frames, so 25 fitness. The cart position and angle are rather straight and do not differ much, there is not much opportunity for the cart to change its position anyhow, there two are mirrored like in the first fully trained controller. The velocities look very basic compared to the successful controllers, this may be due to the scale however this is most likely the main fault with this controller as there is not much sign of the cart trying to counterweight the pole.

Halfway through, the controller has had a large jump in complexity, the fitness is now 462. Straight away, we can see the issue of the cat position quickly navigating to the left, where it exceeds the range of positions the cart can be in. Compared to the last, there is much more complexity within behavior when looking at the cart velocity. Instead of not trying to counterweight the pole, this evolutionary step now tries to perform counterbalancing through flip-flopping the velocity to try and keep it upright and straight, however, the overall result from the beginning push of the cart velocity made it fall out of bounds and fail. The controller needs to additionally think about its position as well as the pole's angle.

Lastly, the controller finds a balance between the cart's position and counterbalancing the pole, but it still can improve. It lasted the whole simulation and the cart positioning is much better with a waveform keeping the balance in check. Near the end of the simulation, the cart position changes drastically compared to the other sizes of the waves which could mean if it lasted much longer it may exceed the space given; we can also see this larger wave in the cart velocity. Strangely, unlike the previous examples, the pole angular velocity is different compared to the cart velocity, this could be due to the wave size increasing with time which may affect the pole angular velocity this way.

4 Discussion

The genotype evolves to use waves as a way of counterbalancing the pole when falling in a specific position.

A typical pattern is either evolving in behavior to use a quick flip-flop method where it extremely quickly changes the direction of the cart with small velocity to make the pole stand up, this method doesn't necessarily need to keep track of its position as all it has to do is equally change its velocity in each direction. The other method is where there are equal size crests and troughs with a medium-sized wavelength which is repeated for the counterbalancing, with a similar, but smaller peaks and troughs pattern appears in the cart position and pole angle. The evolution would need to take a path of additionally mitigating the problem of the pole swaying past 15°, and from this, the problem of position arises and needs to be solved for as long as the simulation lasts. Behavior evolved from improving the velocity to then being aware of the position the cart is in we can see this through the progress in figure 6.

The genetic algorithm finds an optimal solution fairly quickly using the fitness function generated from the simulation awards per frame, however, there are some distinct controllers where if the total simulation time was longer, it would certainly exceed the position range allowed. There is no data needed to train the genetic algorithm, other than the fitness of what the current genotype is. This can give benefits as the time until finding good weights for a controller neural network may be reduced, however, the right parameters need to be set for this time to be reduced. Results may vary on the complexity of the problem, this only had five genes per genotype, this fitness space can be explored rather quickly.

In future investigations, we should explore different environments with more complexity, which may not need to simply counterbalance the pole and keep track of its location. Analyzing different environments may see how far using genetic algorithms to solve these problems can go, another study could be comparing neural networks using backpropagation versus a neural network with their weight assigned from a genotype.

5 Conclusion

The genotype evolves to use waves as a way of counterbalancing the pole when falling in a specific position. It needs to overcome the problem of the angle of the pole, then the position; behavior evolved from improving the velocity to then being aware of the position of the cart. This behavior is most likely specific to this problem, we should further investigate other environments using the same microbial genetic algorithm to see how it performs. We should also measure how effective the algorithm is versus a backpropagation neural network.

6 References

[1] T. Amudha and B. L. Shivakumar. Parameter optimization in genetic algorithm and its impact on scheduling solutions. In Lakhmi C. Jain, Himansu Sekhar Behera, Jyotsna Kumar Mandal, and Durga Prasad Mohapatra, editors, *Computational Intelligence in Data Mining - Volume 1*, pages 469–477, New Delhi, 2015. Springer India.

[2] Inman Harvey. The microbial genetic algorithm. In *Proceedings of the 10th European Conference on Advances in Artificial Life: Darwin Meets von Neumann - Volume Part II*, ECAL'09, page 126–133, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] John H. Holland. Genetic algorithms. *Scientific American*, 267(1):66–73, 1992.

[4] Darrell Whitley, Soraya Rana, and Robert Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7, 12 1998.