Assignment 2 Report

Alejandro Marcano

Student_ID: 4587120

Introduction

Assignment 2 is designed with the objective to create a genetic algorithm that improves the average fitness of the cleaners over a specified amount of generations. This report will guide the reader on the strategies implemented to get the best average fitness scores. My implementation for the genetic algorithm will be tested with different parameters to identify which changes get the best average fitness scores over time.

Approach

In the implementation of the genetic algorithm, I used the tournament selection method. To facilitate this selection approach, I first had to structure my code to accommodate tournament selection. I first had to set up a chromosome with random values between -100 and 100 with a shape of (4, 64). This means that there are 256 chromosomes in total.

Moreover, I used the linear/perceptron like model to determine the actions of the cleaners. To implement the linear/model, I used the 3x5x4 visual tensor which has a total of 60 numbers. The remaining pieces of information that are energy, bin, fails, and 1 are put into a list. I put the number 1 in the end so that I could add the bias at the end of the vector. The visual tensor is flattened to create a vector and combined with remaining information. This creates a vector with 64 numbers. With a vector of 64 numbers, I was able to apply the dot product with the self.chromosome. Although the self.chromosome is of shape (4, 64), I was able to apply the dot product 4 times because each row from 1 to 4 has 64 numbers. The dot product of these four vectors determine the action of the cleaners.

$v1 = w1x1 + w2x2 + ... + w63x63 + b(1)$
$v2 = w65x65 + w66x66 + ... + w127x127 + b(1)$
$v3 = w129x129 + w130x131 + ... + w191x191 + b(1)$
$v4 = w193x193 + w194x194 + ... + w255x255 + b(1)$

Fitness Function

For the fitness function, my first approach is to use the clean stats already provided. I chose this approach for the fitness function because it was simple and it does reward the cleaners that pick up the most dirt. Furthermore, if they pick up more dirt, then it is possible that the cleaners who recharge more are awarded a higher fitness since they can move more. Thus they can pick up more dirt.

Elitism

After evaluating the fitness of all the cleaners, I have a list of all the fitness scores of the 40 cleaners. In my initial approach, I kept the best 5 cleaners in my population after every generation. This tactic is referred to as elitism. Elitism is important because it allows me to preserve the best individuals from one generation to the next generation without any modification. This also means that the optimal solutions are not lost over time.

Tournament Selection

Following each generation, I aim to create a new cleaner using the tournament selection method. In the tournament selection process, a random subset of the population is selected, from which the top two parents are selected to create a new child. In this case, I use the two highest fitness scores from the subset to create the new cleaner.

```
OLD_POPLULATION = [Cleaner 1, Cleaner 2, ………, Cleaner 40]
RANDOM_SUBSET = [Cleaner 3, Cleaner 10, Cleaner 30, Cleaner 25]
BEST_TWO_PARENTS_IN_SUBSET = [Cleaner 3, Cleaner 30]
```

Since I have the best two parents, I can deploy the cross-over strategy of my choice. For this genetic algorithm, I chose the single point crossover strategy. To elaborate, the single point crossover strategy selects a point along the genetic material to exchange genetic materials between the two parents. In my case, I chose to do the crossover at the halfway point. Thus the first half of the parent one's genetic material is sliced and the second half of the second parent's is sliced. The first half of the parent's chromosomes are then concatenated with the second half of the second parent's chromosomes. Therefore, I have successfully created a new child using the single point crossover method.

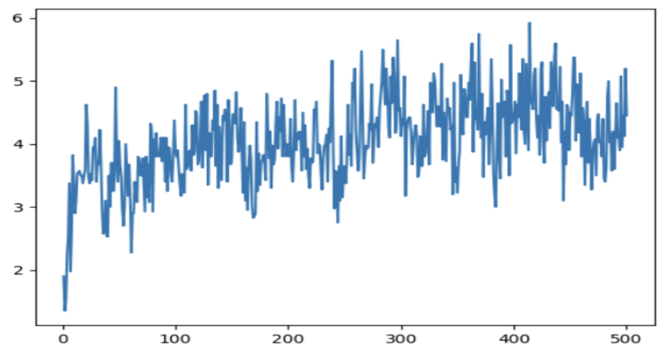
Mutation

Another crucial component of the genetic algorithm is mutation. The reason why mutation is an integral part of the genetic algorithm is because it introduces genetic diversity into the population. In other words, if I don't introduce genetic diversity to my population, then I would not be able to explore different regions of the search space. Mutation facilitates in preventing early convergence of the solution and allows me to explore diverse solutions.

Implementation of Mutation

For this problem, there were different ways of deploying a mutation strategy. In my case, I started with a mutation rate of .15. I created a matrix of shape (4, 64). In this matrix, using numpy random, I was able to set all the values between 0 and 1. Subsequently, I iterated over my matrix to see which values were less than .1. The .1 in this case is an arbitrary number. I can choose to make it larger if I feel like I need more mutations in the chromosomes or decrease this value if I need less mutations. Using the index of the matrix where the number is less than .1, I mutated the chromosome in the corresponding index of the new cleaner. Using this strategy, I ensure that I keep diversity in the cleaner population.

Average Fitness over 500 generations



Analysis of Average Fitness Over 500 Generations

The first thing I observed from the plot above is that in the first couple generations, the fitness increases drastically. I attribute this occurrence to the process of crossover and elitism. Elitisms allow me to keep the best cleaners in subsequent generations from the beginning. This results in rapid gains in overall fitness in the populations from the start. Alongside elitism, single point crossover also led to rapid gain in fitness in early generations because of the combination of the best features from two parents. This results in a more fit population from the start.

After the initial quick increase in the average fitness of the population, the average fitness increase slows down. Although the average fitness goes up and down a lot over the generations; the trend of the average fitness is still increasing. This is a good sign because it means it's exploring a range of optimal solutions while not converging at a single point. The underlying cause for my perspective on this is that mutation creates a diverse population over time. Some of the mutations might be beneficial in increasing the overall fitness of the population while other mutations might not be so good.

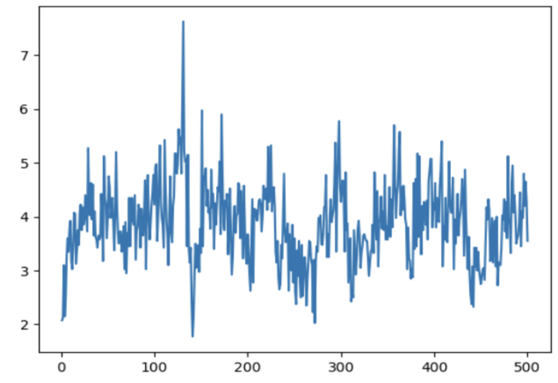
Changing The Parameters

The initial parameters I have for my genetic algorithm are:

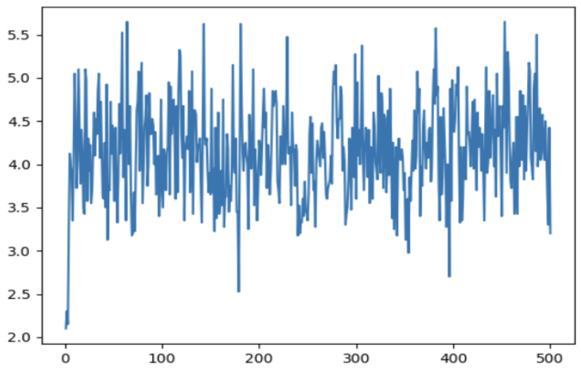
Elitism: 5,
Subset of Tournament Selection: 10
Fitness Function: "#Cleaned"
Mutation_rate: 0.15

My goal is to change one parameter at a time to see if I can see improvements in the average fitness score of the population and keep the increase in a slower upward trend. My first approach is to lower the elitism. The reason why I want to lower the elitism is so that I can slow down the rapid increase in fitness in the beginning. This is so that we don't have the very best parents from the start and slowly get a more fit population through single point crossover. The results were way worse than I expected. Not only did my average fitness decrease, there is no longer an upward trend across the generations. I tried increasing elitism to 7. I saw that the average fitness was better than when elitism was 3, but the trend of the average fitness across the generations is not increasing. The best results were obtained when I had elitism at 5.

Results for Decrease in Elitism from 5 to 3



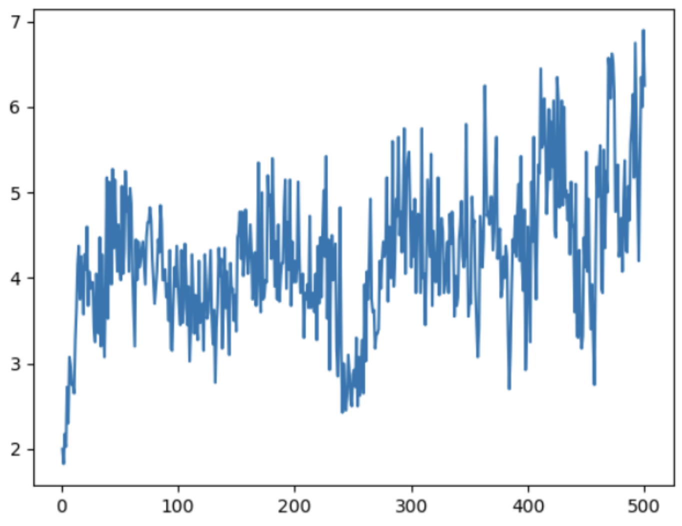
Results for Decrease in Elitism from 5 to 7



Changing the subset amount for Tournament Selection

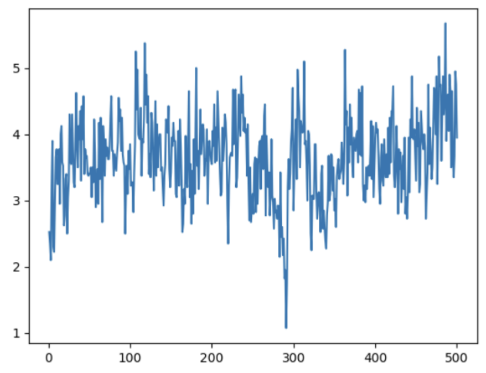
The original subset I used for tournament selection was 10. My objective is to see if I can increase the fitness of the population by either increasing or decreasing the subset for tournament selection.

Results for Increasing subset from 10 to 15



I was surprised by the results by increasing the subset for the tournament selection. I noticed that the change in fitness level over the generations was more dramatic. However, I would say this is a good result since the trend is still increasing. Moreover, using a subset of 15 finally resulted in consistent scores over 200 in the 5 games. On the other hand, decreasing the subset to 5, significantly decreased the average fitness score and there is no longer an increasing trend in fitness either.

Results for Decreasing subset from 10 to 5



Changing the Mutation Rate

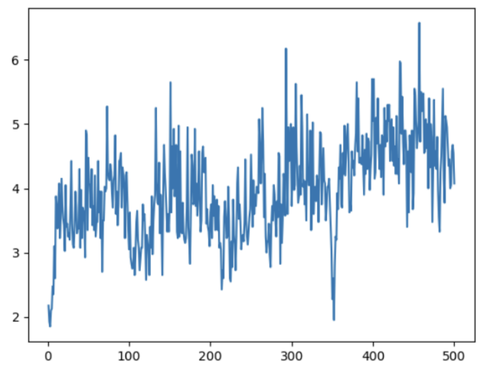
I increased the mutation rate from 0.15 to .20. The results of the fitness score improved a lot.

However, there is still an increasing trend of the average fitness score of the population.

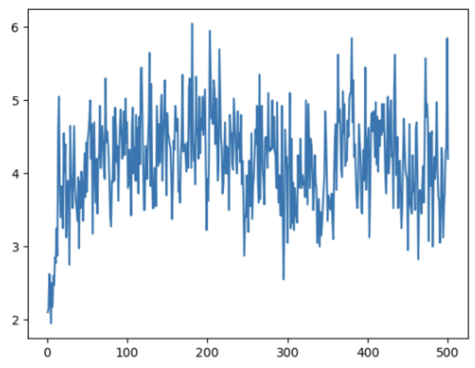
However, sometimes the change in average fitness score of the populations is too unstable.

Therefore, increasing the mutation rate was not helpful in this case.

Increase in Mutation rate from 0.15 to 0.2



Decrease in Mutation rate from 0.15 to .1

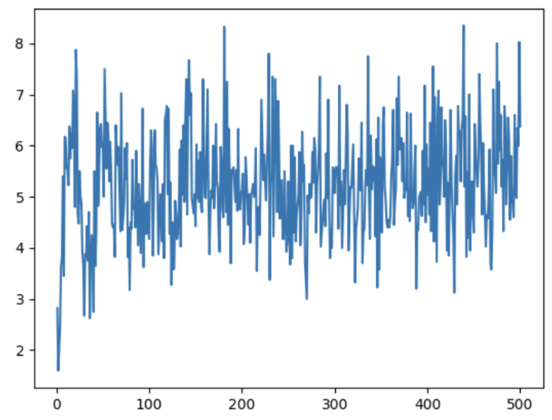


This time, I decreased the mutation rate from 0.15 to .1. The results of the fitness score improved a lot. However, there is no consistent pattern in the increase of the fitness score of the population. Therefore, decreasing the mutation rate did not expand the solution space.

Change in Fitness Function

I changed the fitness function. The original fitness function only measures how many are cleaned. I aim to improve my fitness function by adding “cleaned” and “emptied”. I chose to add “emptied” because the cleaners both offload their dirt in the charger tile and recharge their energy in the charger tile. My thinking is that this also rewards the cleaners that also recharge and offload dirt. The results for the average fitness score improved, but there is no consistent trend in increase of the average fitness. Thus, adding “emptied” to my fitness function is not what I am looking for.

Adding “emptied” and “cleaned” in Fitness Function



Conclusion

I designed a genetic algorithm that shows a trend in increasing the average fitness score of the population of cleaners . Although the increase is drastic in the beginning, the average fitness score of the population is slowly increasing after the initial spike. This result was achieved by implementing a combination of genetic algorithm strategies such as elitism, mutation, and crossover. To conclude this report, calibrating the variables like mutation and elitism were key in achieving this increasing trend in average fitness score of the population over the generations.