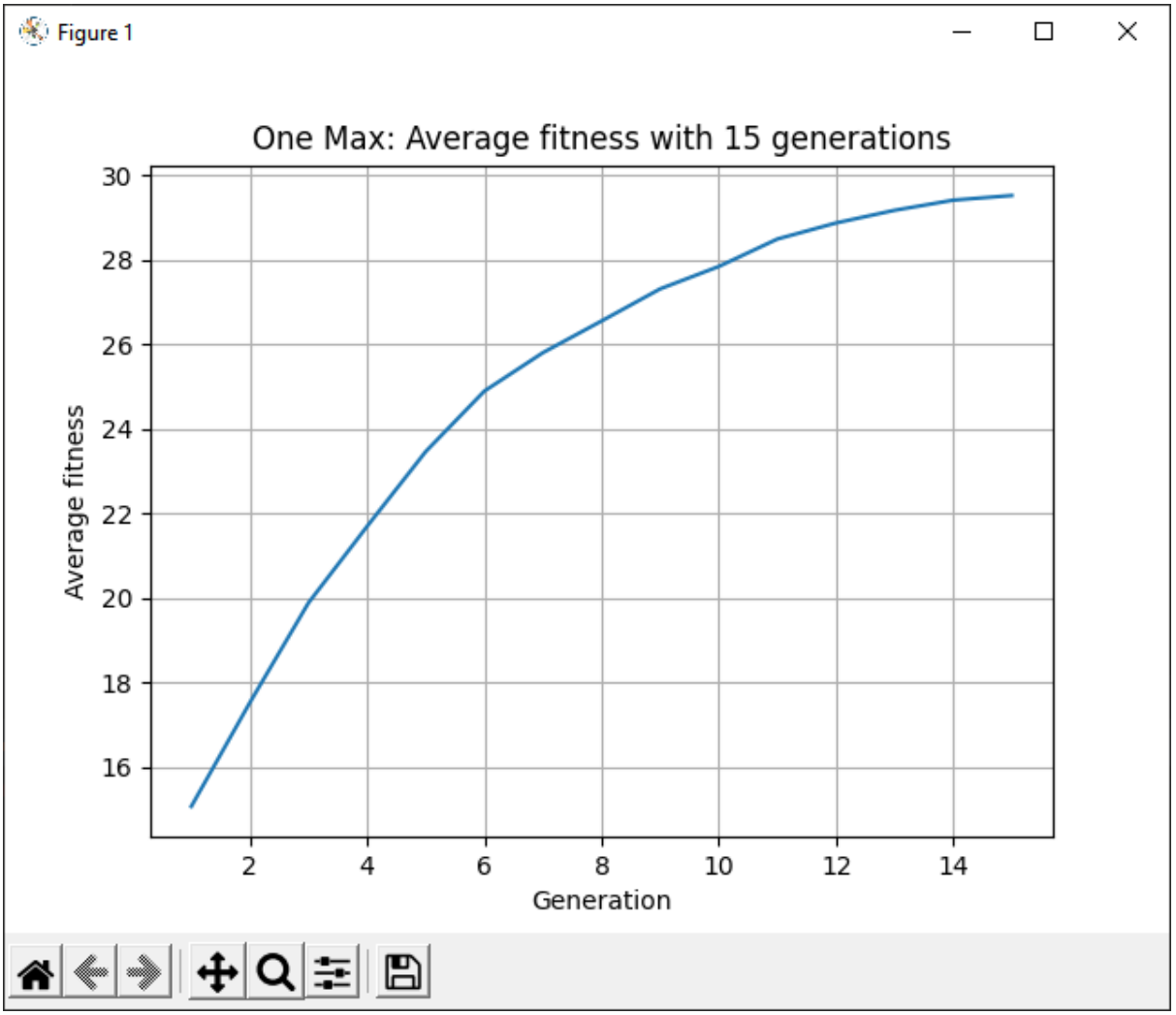
CT421 Assignment 1

Part A

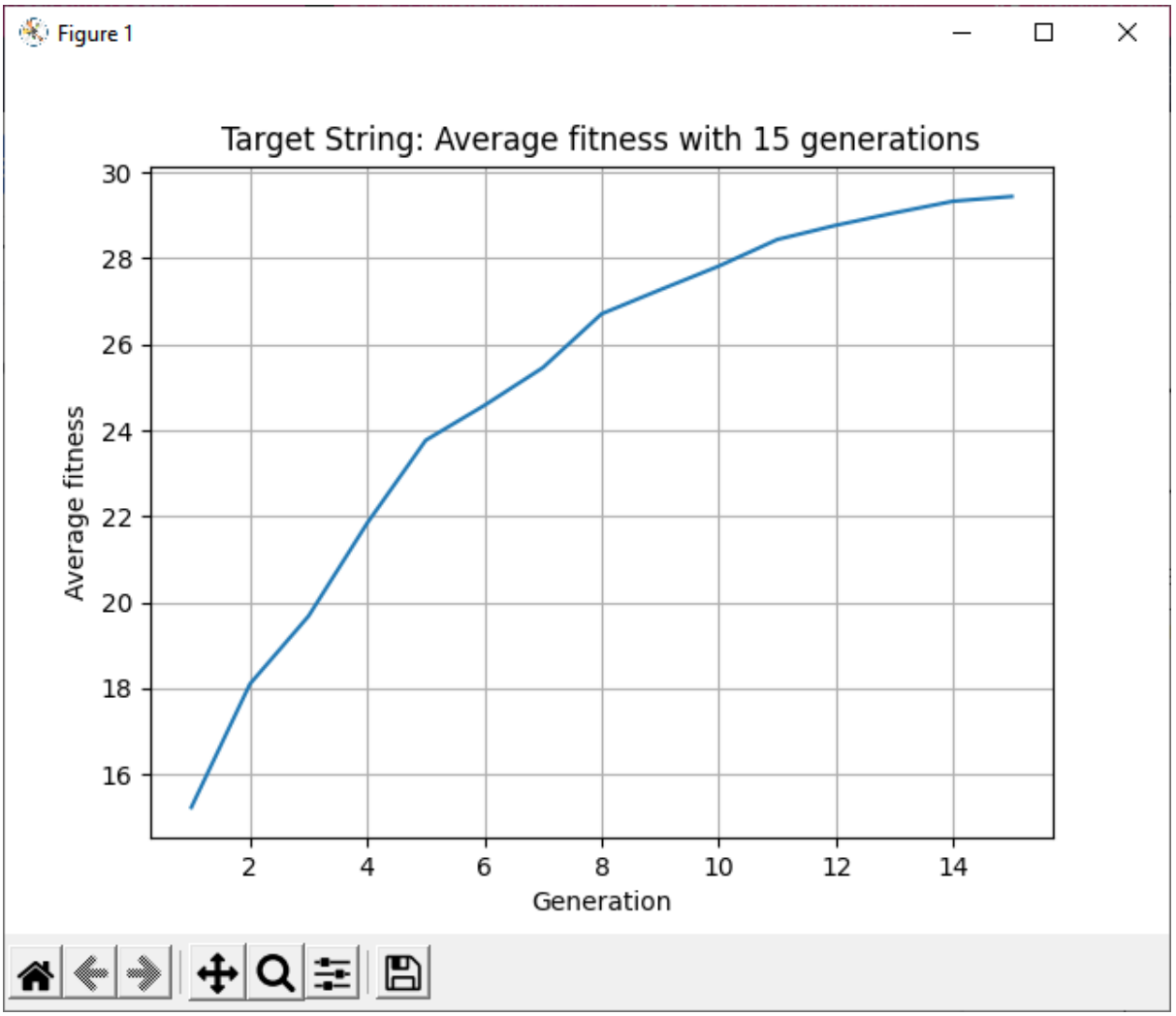
One-max Problem

The one-max problem implementation reached the target of ‘11111111111111111111111111111111’ by generation 8 when the average fitness was 26.38. By generation 15 the average fitness was 29.52.



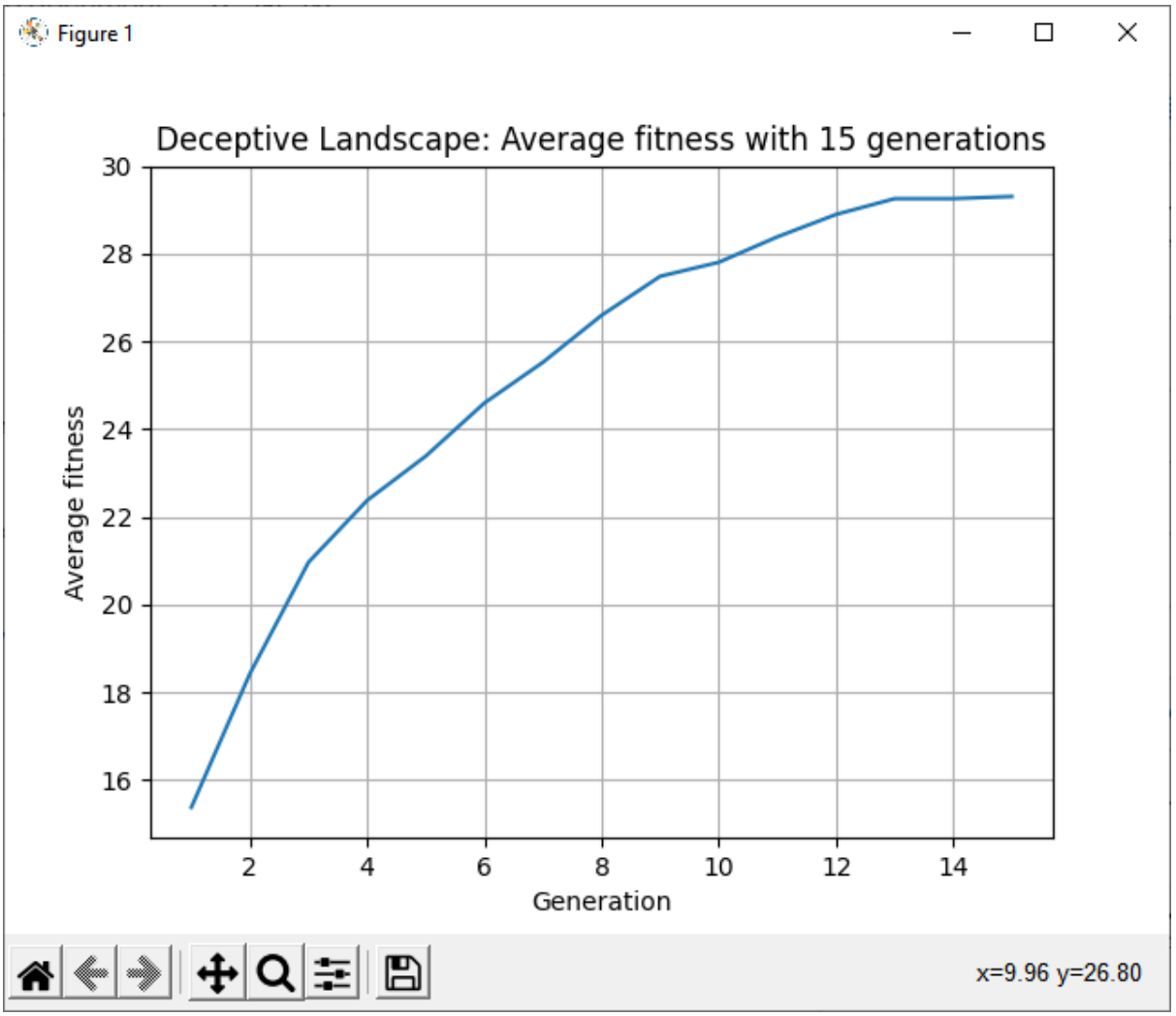
Evolving to a target string

The target string problem implementation reached the target of ‘1100010111000101100111110101110’ (randomly generated every time) by generation 8 when the average fitness was 26.25. By generation 15 the average fitness was 29.53.



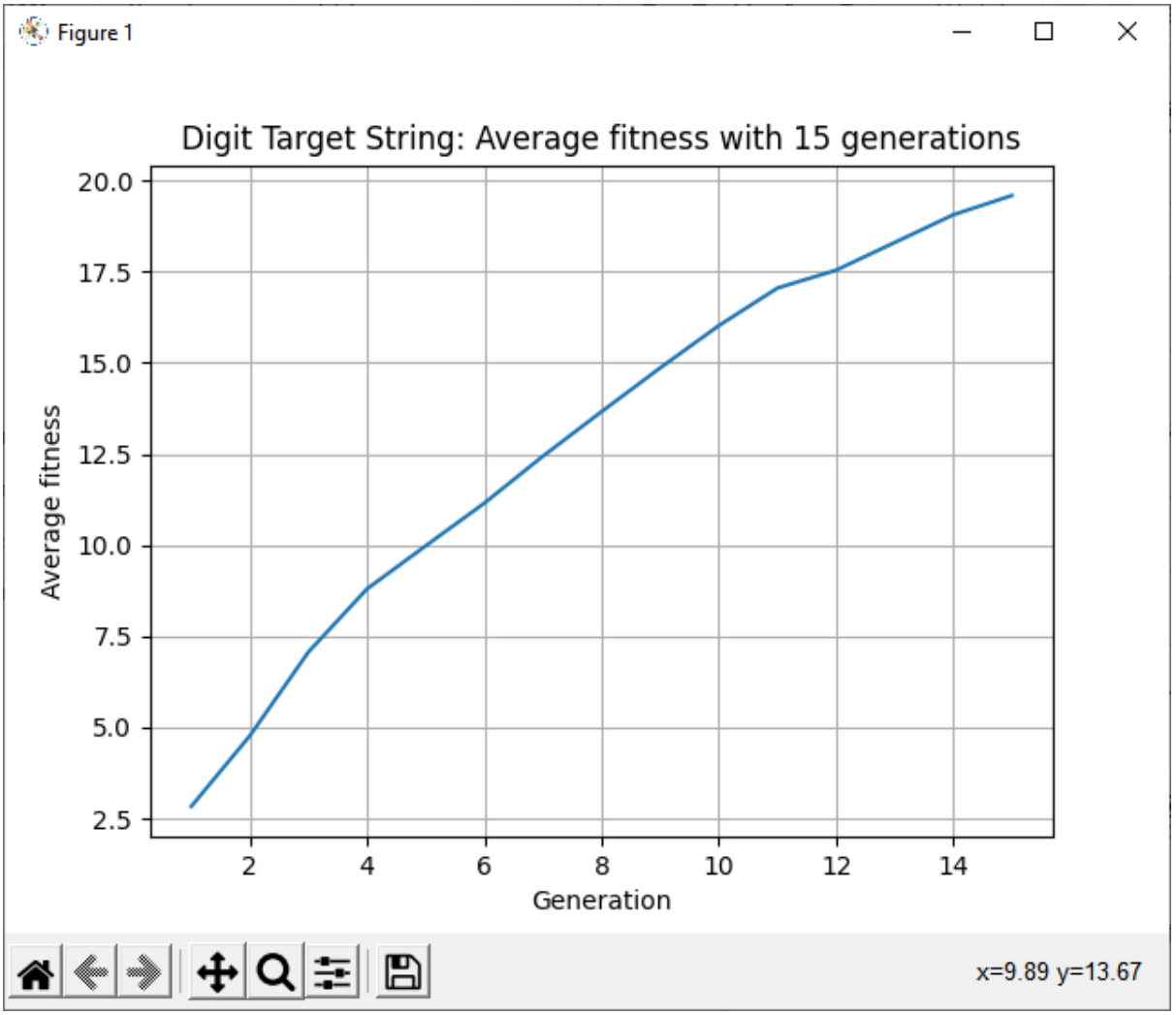
Deceptive Landscape

The deceptive landscape problem implementation reached the target of ‘11111111111111111111111111111111’ by generation 9 when the average fitness was 27.17. By generation 15 the average fitness was 29.42.



Evolving to a target string with a larger alphabet

The target string with digits problem implementation never reached its target string. By generation 15, its fitness was only 21 with an average fitness of 19.8.



Part B

Fitness Function

The fitness function I used I kept track of the total value and weight of the string, and I iterated over every index in the binary number and checked whether it was a 1 or 0. If it was a 1, I added the value at index i in the values array we were given to the sum total, the same was done for weights. Once I had iterated over all of the binary string, I checked if the weight of the string was higher than the knapsack capacity, if it was, I returned the fitness as zero.

Mutation Function

The same mutation function was used for the one max, deceptive landscape, target string, and knapsack problem. It did a mutation 'rate_of_mutation' of the time. When it did, it simply flipped one of the bits in the binary string.

Crossover Function

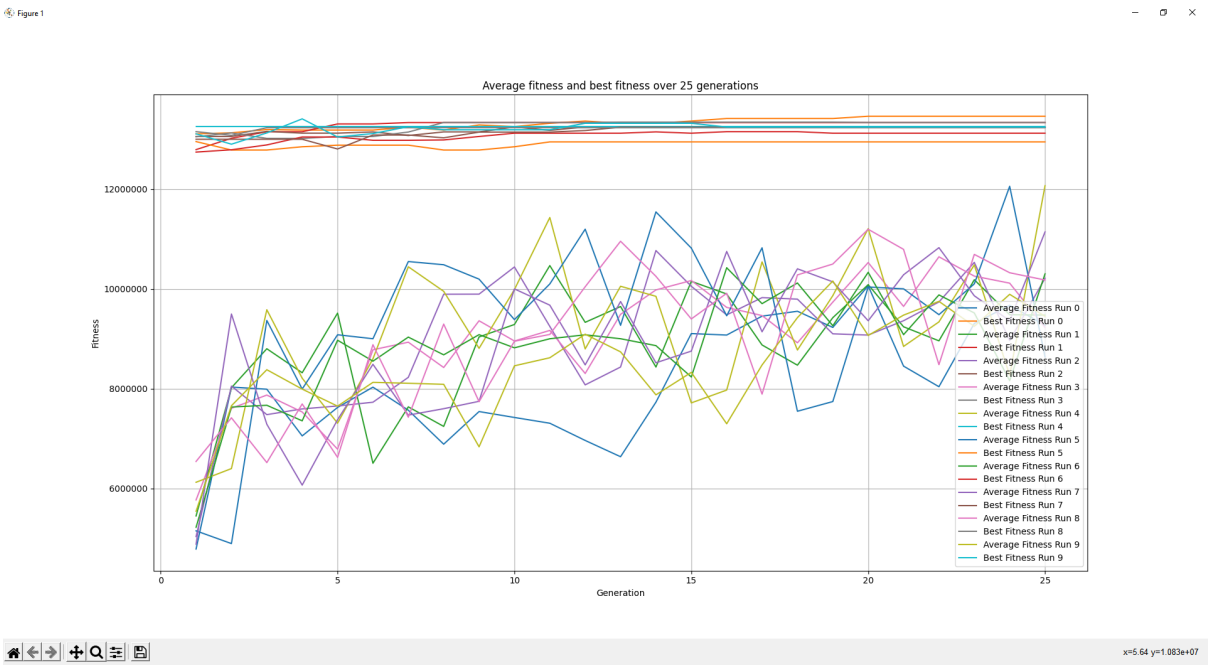
The same crossover function was used for the one max, deceptive landscape, target string, and knapsack problem. It did a crossover 'rate_of_crossover' of the time. When it did, it made a copy of the parents and put them into two children objects that would replace the parents eventually, for each child it took a section from each parent (n bits from the front of one and length - n bits from the other).

Selection Function

The same selection function was used for all problems. I based it off of tournament selection. Tournament selection is a method of selecting an individual from a population of individuals. Tournament selection involves running several tournaments among a few individuals chosen at random from the population. The winner of each tournament (the one with the best fitness) is selected for crossover.

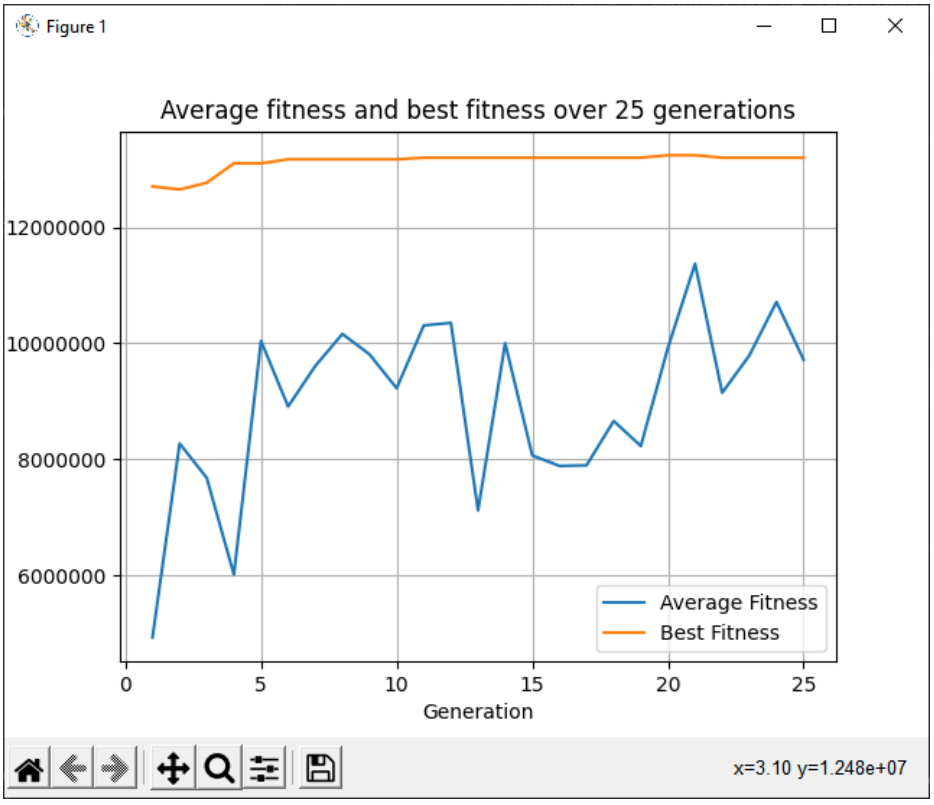
Representation

I tried to add all graphs to one plot but it didn't give a good representation.



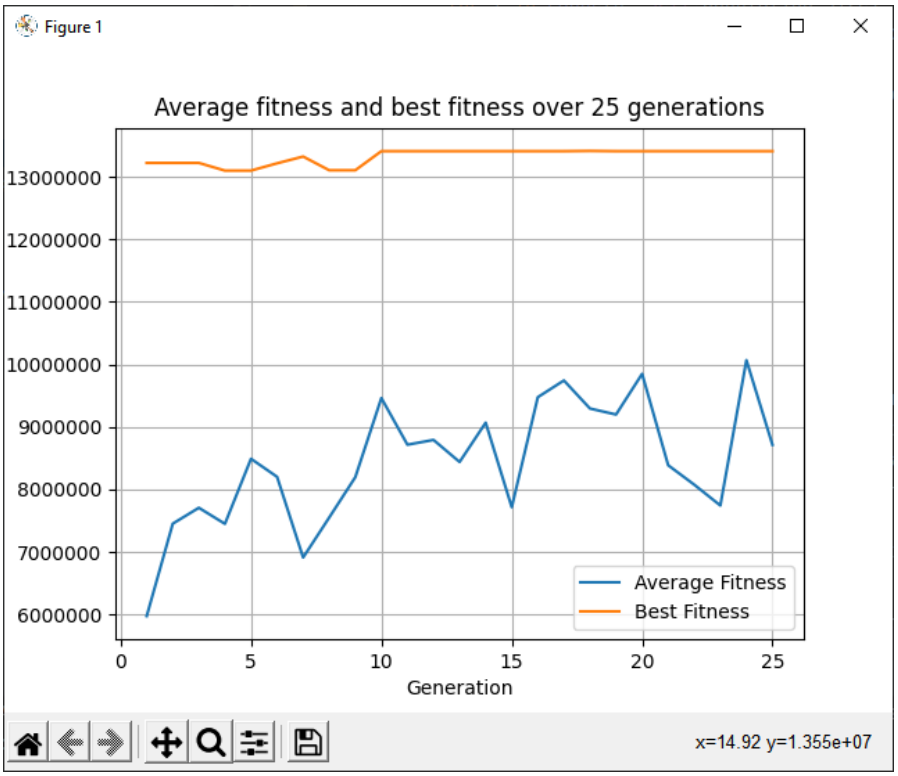
Instead I ran each one manually with a random population between 20 and 180. I'll record the results and graphs below.

Run 1



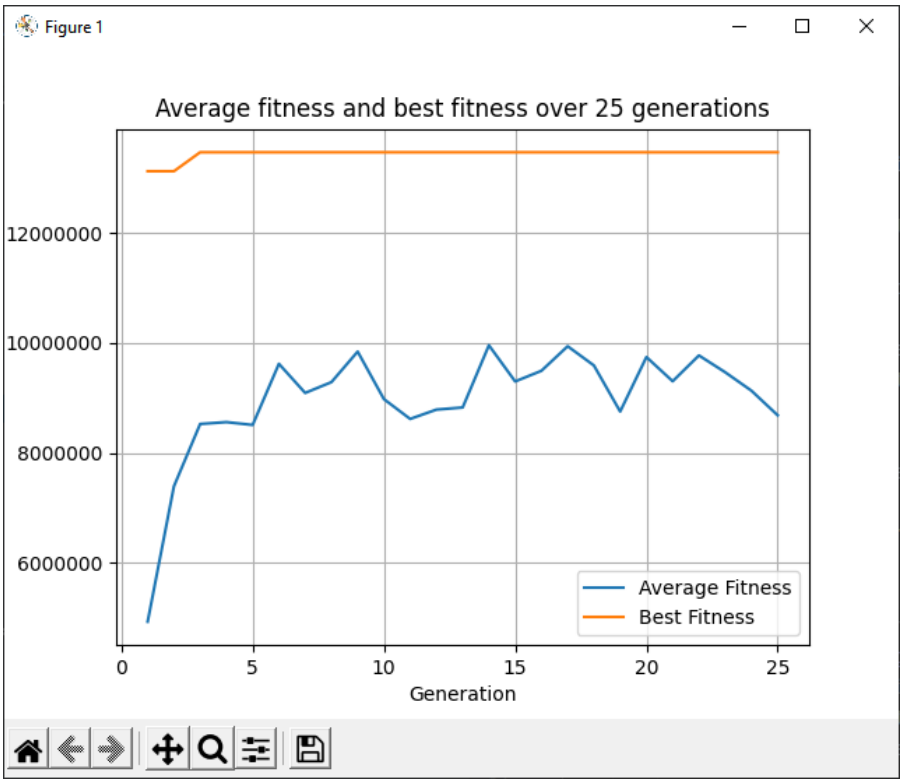
Population size: 32

Run 2



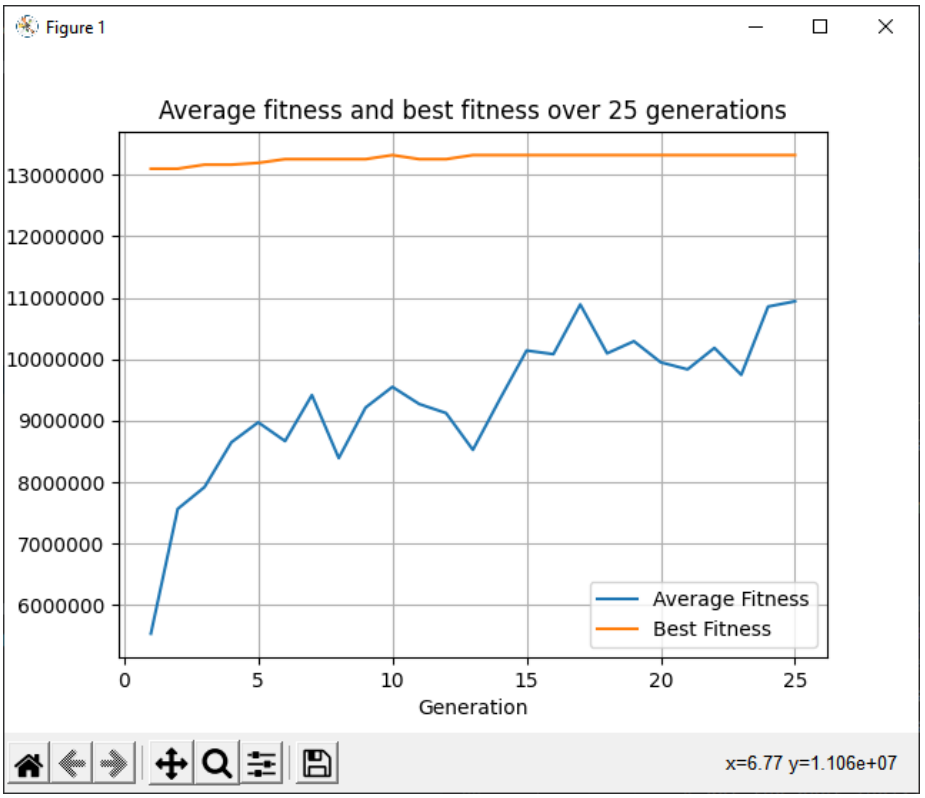
Population size: 122

Run 3



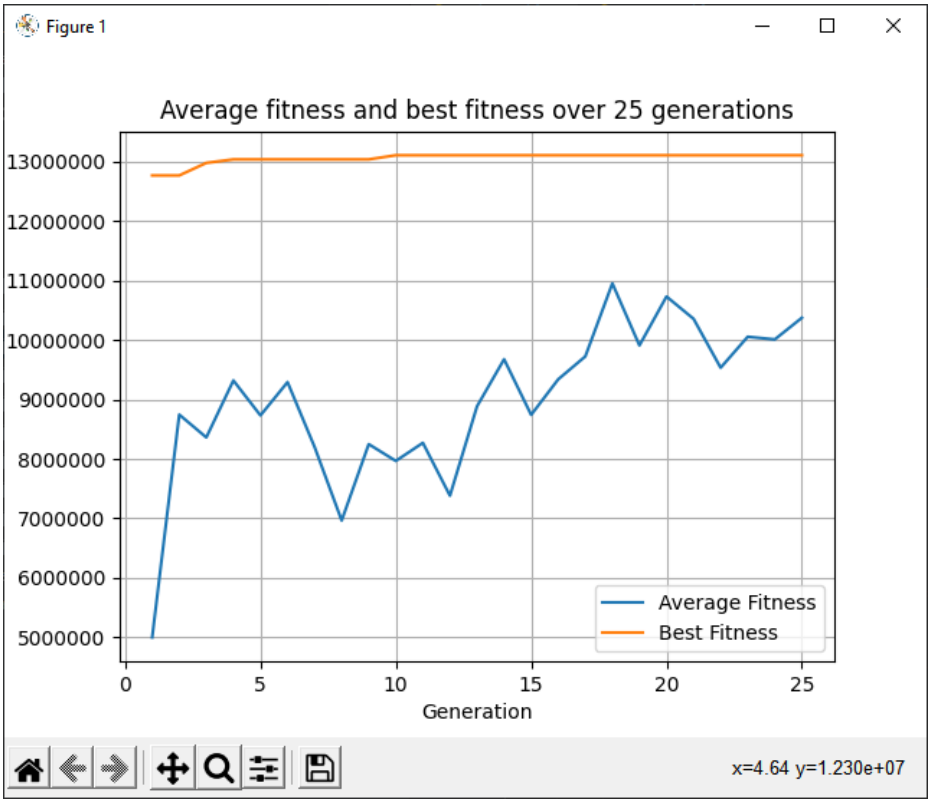
Population size: 74

Run 4



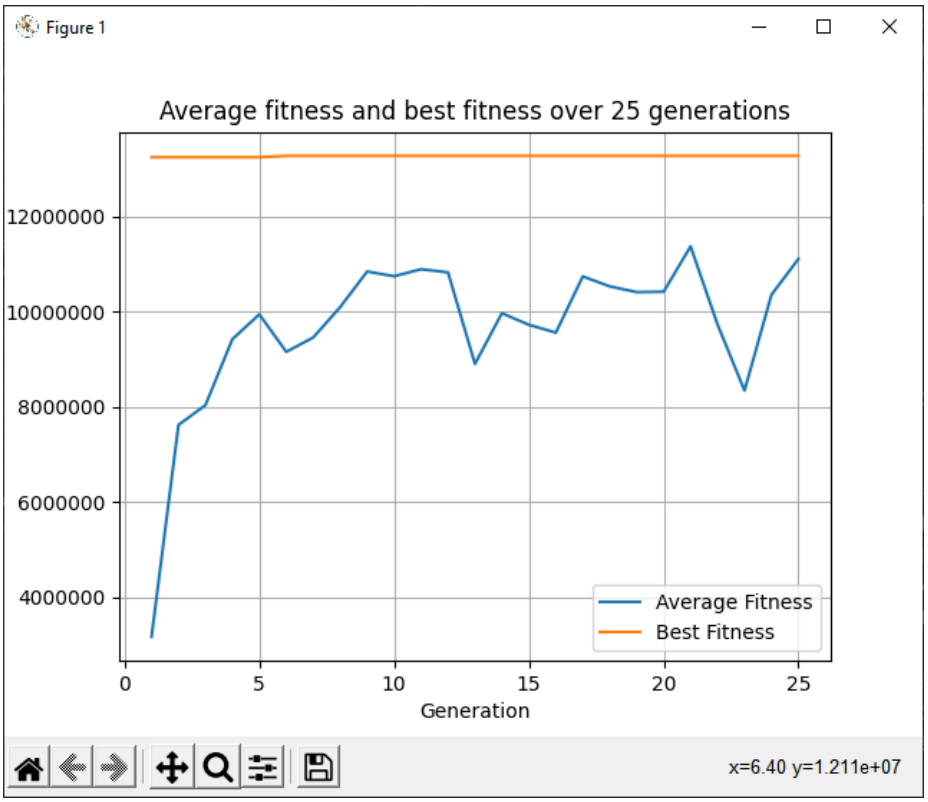
Population size: 84

Run 5



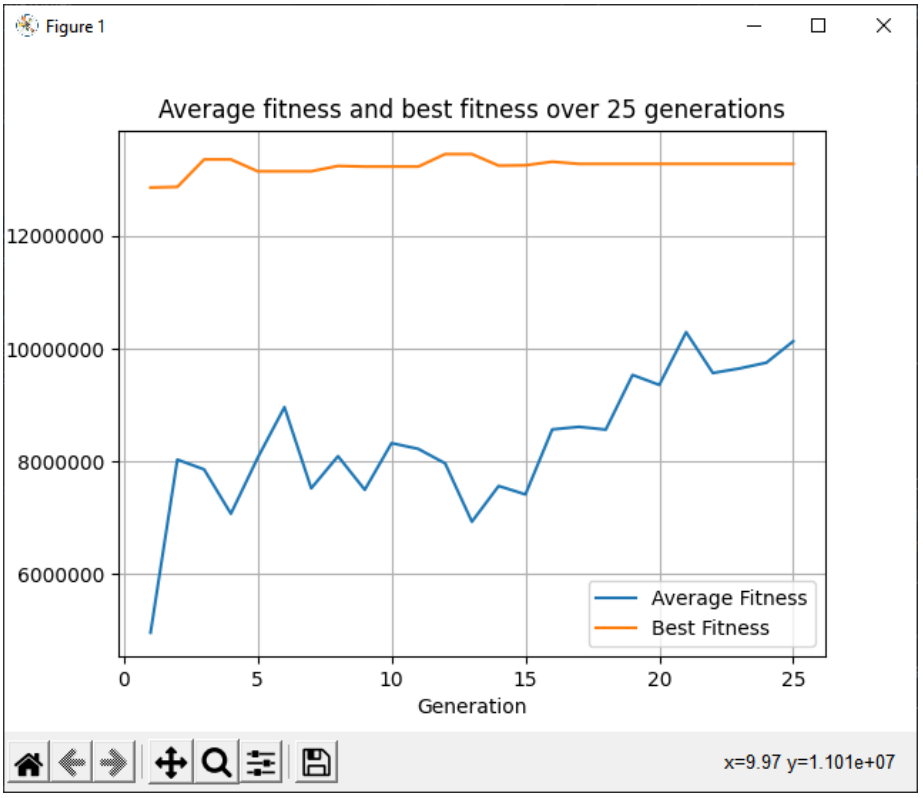
Population size: 56

Run 6



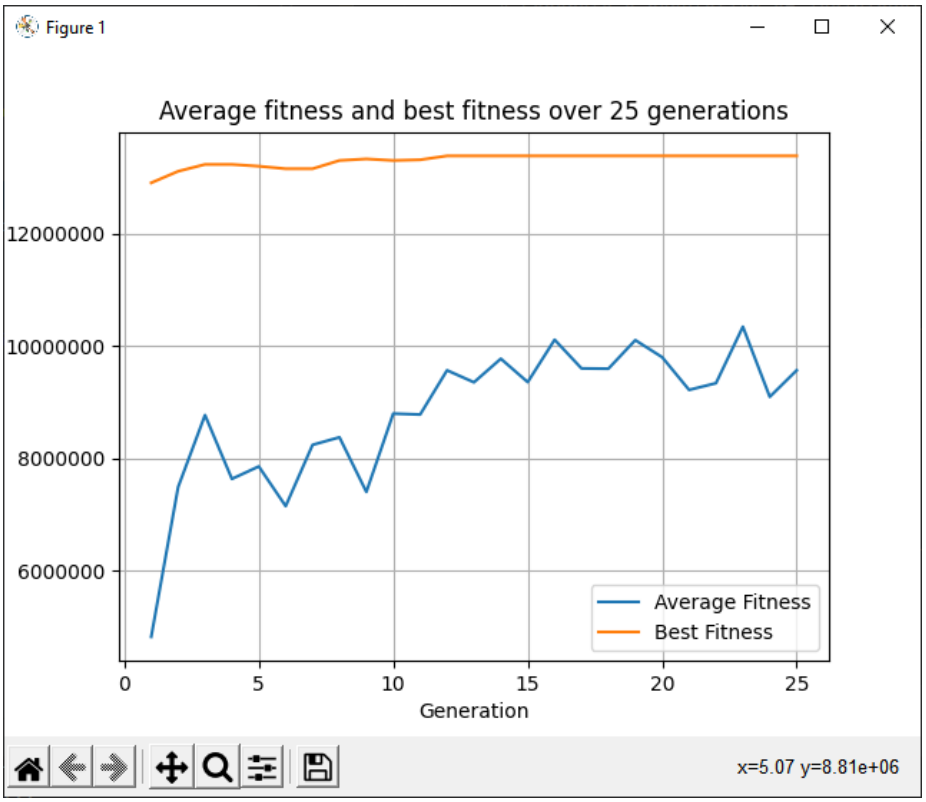
Population size: 50

Run 7



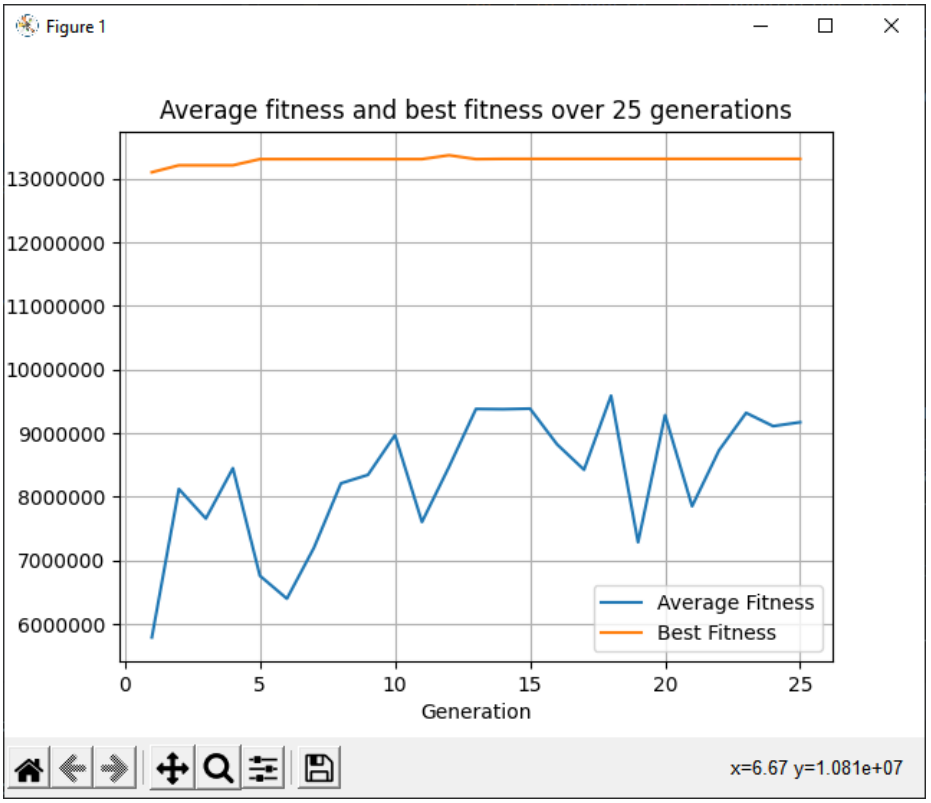
Population size: 146

Run 8



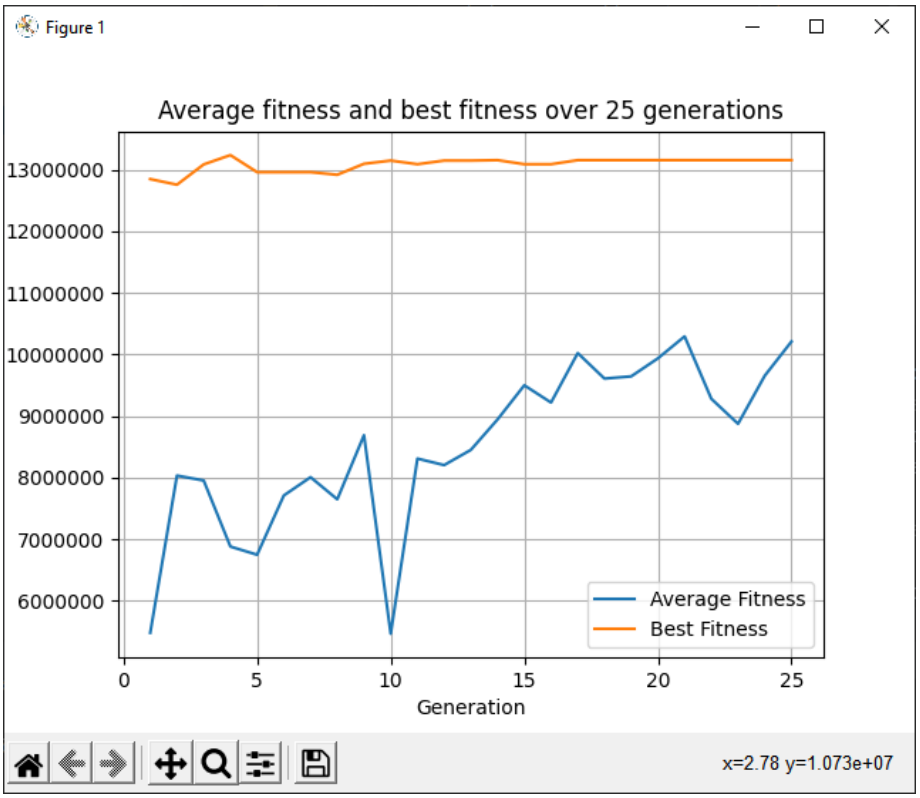
Population size: 152

Run 9



Population size: 112

Run 10



Population size: 62