Michael Merola
CSCI3202
Assignment 2 Report
15 October 2019

Part 1, Bitstring Genetic Algorithm

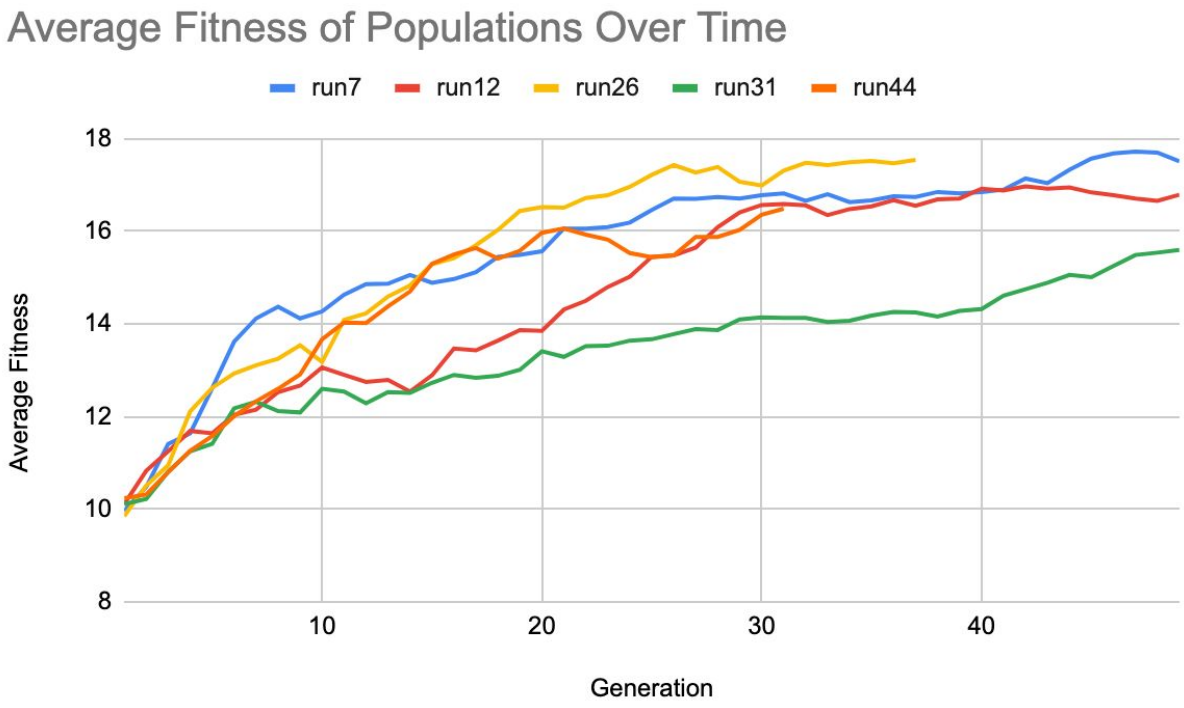
1. For 50 runs of my Genetic Algorithm with parameters:

   population size = 100
   genome length = 20
   crossover rate p = 0.7
   mutation rate p = 0.001

   the best fit string stats were:

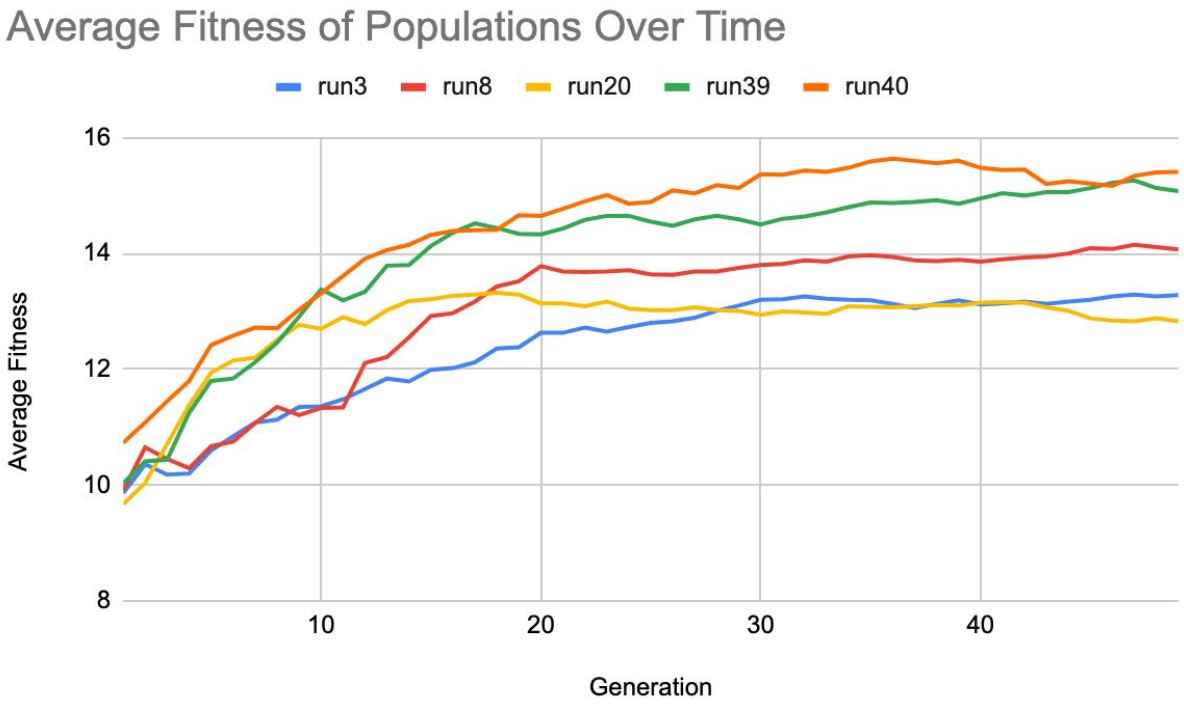
   Average Generation = 32.95
   Max Generation = 49
   Min Generation = 17
   # of None Solution = 29

2. Chart of Runs *7, 12, 26, 31, 44*:



For each of the 5 runs, the Genetic Algorithm slowly improves over the course of its timeline. The subsequent generation does not *always* improve across each run; sometimes the average fitness will decline for a generation, then correct on the next one and steadily increase to the goal. This results in a somewhat logarithmic looking plot with imperfections and ridges throughout it's growth. Interestingly, all of the plots seem to start with a high slope and start to level off after generation 10.

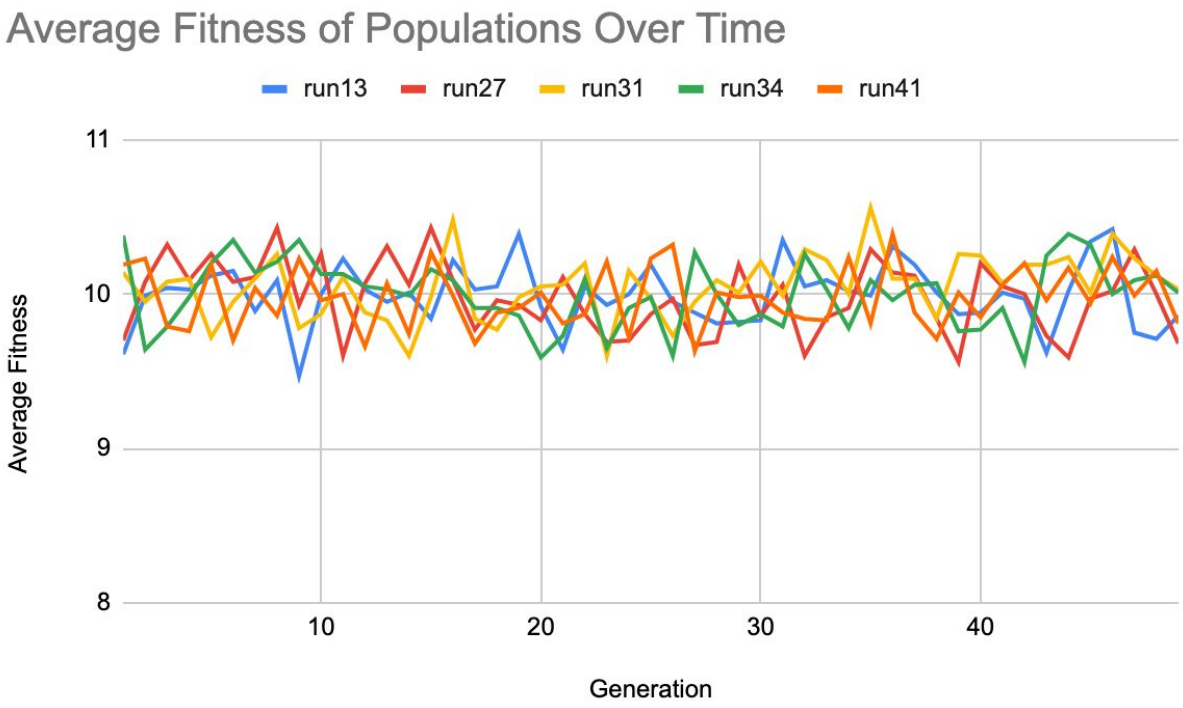
3. When running the experiment with *crossover rate = 0*, I got no runs that successfully converged to a complete fitness bit string; so there are no stats on average, max, and min generation.



By nullifying the crossover rate of the function, the genetic algorithm is completely prevented from finding a solution. The plots all look much more logarithmic than in the previous experiment, so there is evidence that the mutate function is helping the generations improve with time, but without being able to cross successful genomes, it's difficult for the algo to complete. Other interesting factors: the beginning slope is much less steep, and the later stages are more flat than the previous experiment.

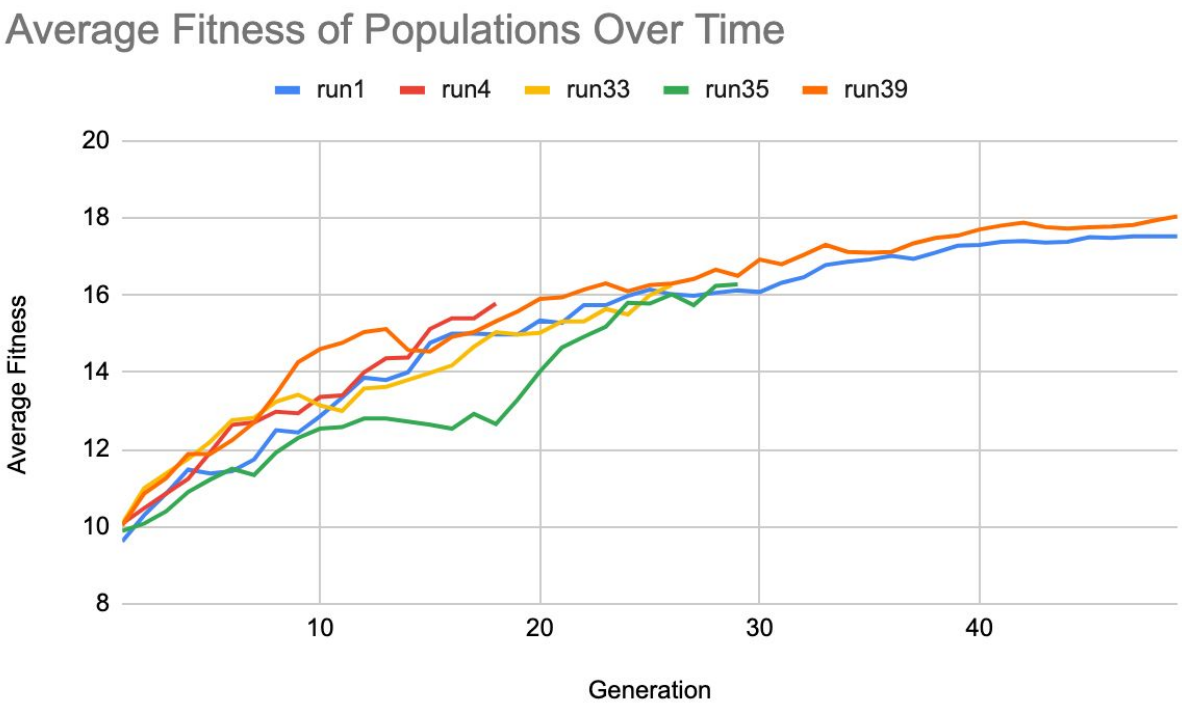
4. **EXPERIMENT 1:**
   population size = 100
   genome length = 20
   crossover rate p = 0.7
   mutation rate p = 0.5 **test with high mutation rate*

   Result: No Successful Runs



EXPERIMENT 2:
population size = 100
genome length = 20
crossover rate p = 0.7
mutation rate p = 0 **test with NO mutation rate*

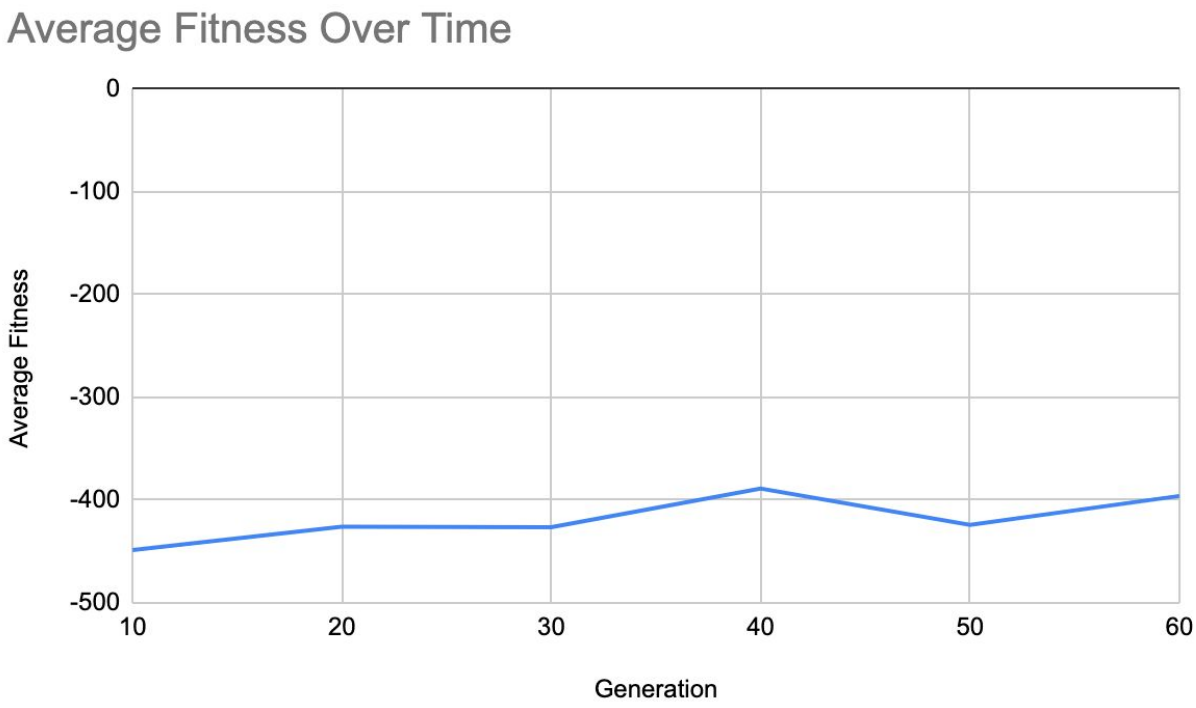
Result:
Average Generation = 26.533
Max Generation = 43
Min Generation = 13
nones = 35



In my experiments, I wanted to see how the mutation rate variable affects the performance of the genetic algorithm. In Experiment 1, I tested a high mutation rate of 0.5 because it was magnitudes higher than the initial experiment with a rate of 0.001. The results were interesting, as the high mutation actually prevented the algorithm from finding a solution; in fact, the average fitness rate bounced around 10 similar to a sine wave. Because the mutations were so frequent and random, the generations never had a chance to consistently improve.
With Experiment 2, I tested the opposite: setting the mutation rate off (mrate = 0.0). Interestingly, this seemed to actually *improve* the performance of the algorithm. With the initial settings, the *Average Generation = 32.95, Max Generation = 49, and Min Generation = 17* whereas without mutation rate I get *Average Generation = 26.53, Max Generation = 43, and Min Generation = 13* - all better stats. The plots are also notably smoother, with less ridges and are closer to a linear plot than a logarithmic. It's possible that the randomness of the mutate function is actually detrimental to the algorithm and hinders each generation's potential for improving fitness. Because mutate is random, however, means that it can either act for or against the algo - an agent of chaos.

Part 2, Bitstring Genetic Algorithm

1. **EXPERIMENT:** (default)
   population size = 100
   crossover rate p = 1.0
   mutation rate p = 0.005



Best Strategy Found:

```
[00000000012100110013000001336603152020113321456520122624654310151422263401020264565034334103155425000614330453553205363311211123421260626200436222650212004312162620036512604104412636505562630631300552662465443604652215540362202626406366264026 5]
```

**Disclaimer: algorithm ran too slow, only generation 60 was reached in 1.5 hours*

The Genetic Algorithm for Robby did not make much progress throughout the generations. I am not sure if this was due to miscalculations/inefficiencies in my selectPair function or the crossover and mutate are not doing their job correctly. The plot generally has an upwards trajectory, as robby hits less walls in each generation. In future experiments, it would be good to adjust the algo to get it faster.