Genetic Algorithms

Introduction

The goal of this project was to investigate the effect of different variables on the fitness of a randomly generated population after multiple generations of crossover reproduction. To calculate the fitness of an individual, the following formula was used:

$$F(s) = (x/2^l)^{10}$$

Where x is the integer that results from interpreting a string of 0's and 1's s as an unsigned binary number. The different variables that were examined were: l – the number of genes in the genetic string, N – population size, p_m – mutation probability, p_c – crossover probability, and G – the number of generations in a simulation.

To produce each generation, two parents were chosen and "mated." The crossover probability determined the chance that the parents' genome strings would be mixed in the offspring or just be a copy of one of the parents. After N individuals were produced, the current generation was replaced with the offspring and a new generation was created. This was repeated G times.

To do this, I wrote a program in Python which accepted each of the five variables and output all the data to a .csv file. I ran 10 runs for each set of variables and created graphs using Microsoft Excel that display best and average fitness, as well as the number of correct bits in the most fit individuals.

Analysis

In the fitness graphs that follow, the green lines represent the fitness of the best individual for each generation, whereas the blue lines represent the average fitness over the whole population. To begin, I conducted a test using the suggested parameters, these results are shown below in Figure 1.

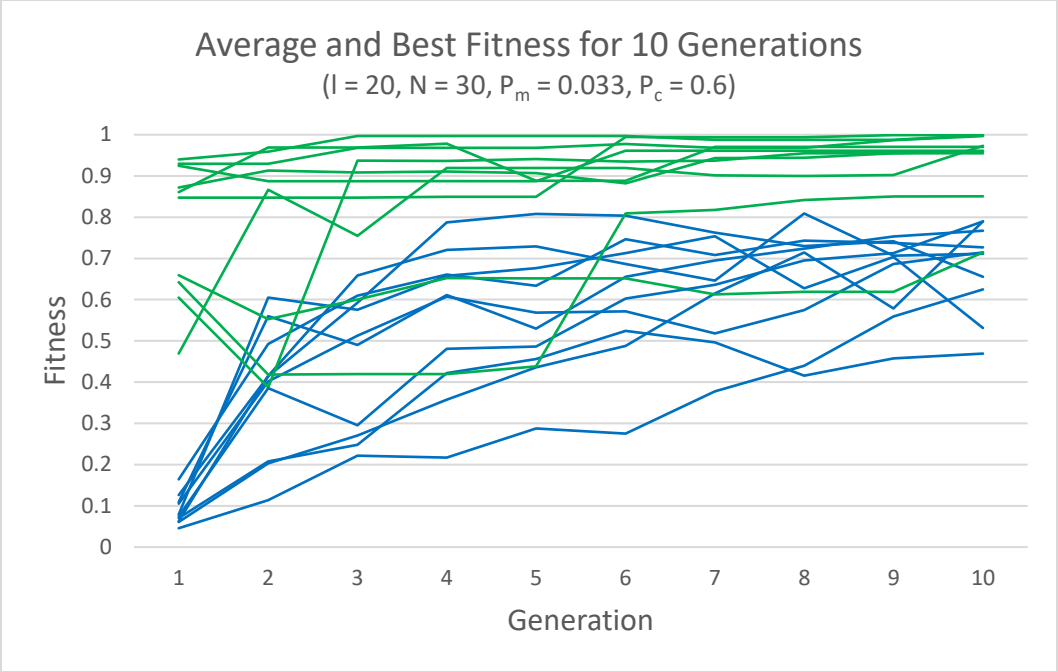

Figure 1) Very little convergence, but the best individual commonly reached close to 1.0 fitness.

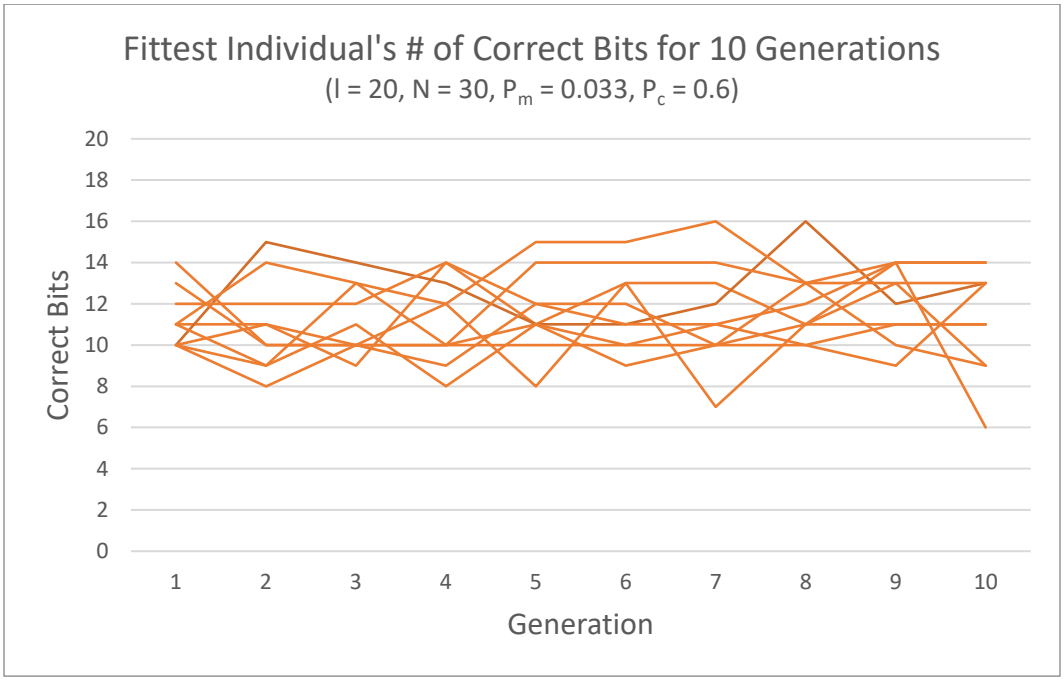

Figure 2) Range of 10 bits, no significant trends either up or down.

Even only over 10 generations, 8/10 of the best individuals were already above a fitness of 0.9. Convergence was very gradual for average fitness, however, and the average did not always even make it over 0.5 fitness. There was still an upward trend at generation 10, though, so we could assume that this trend will continue if more generations are simulated.

For most of the best individuals, they had a majority of bits set to 1, with some having 16/20 correct bits. This makes sense because individuals with more 1's are favored in reproduction.

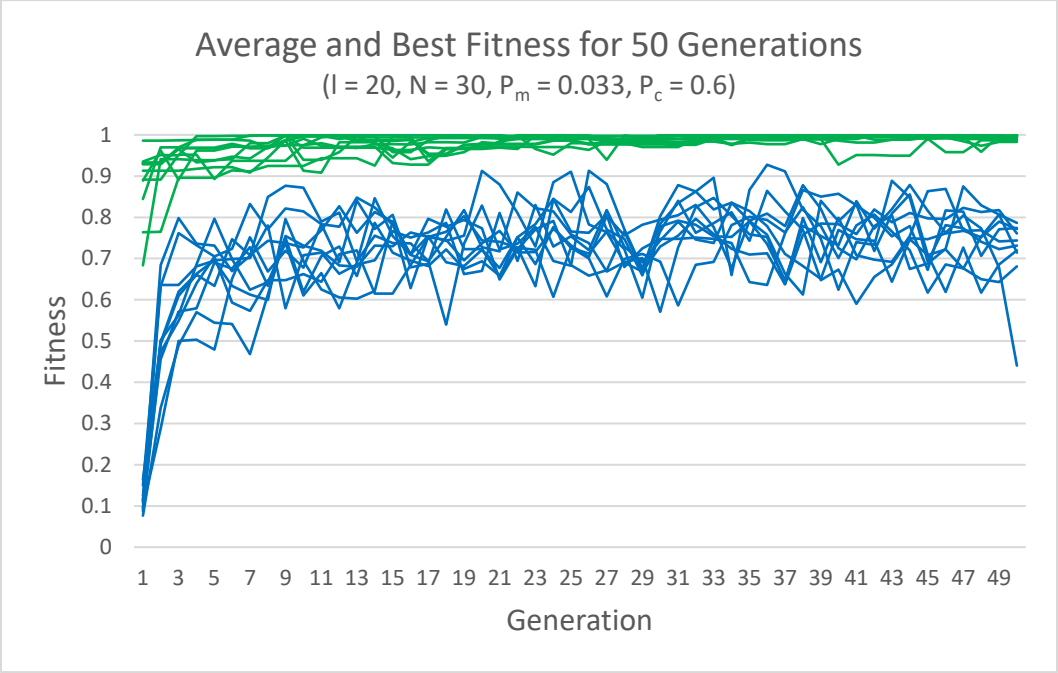

Figure 3) Majority of best individuals make it very close to 1.0 fitness by generation 50. Average fitness continues to raise and seems to cap off around ~20 generations.

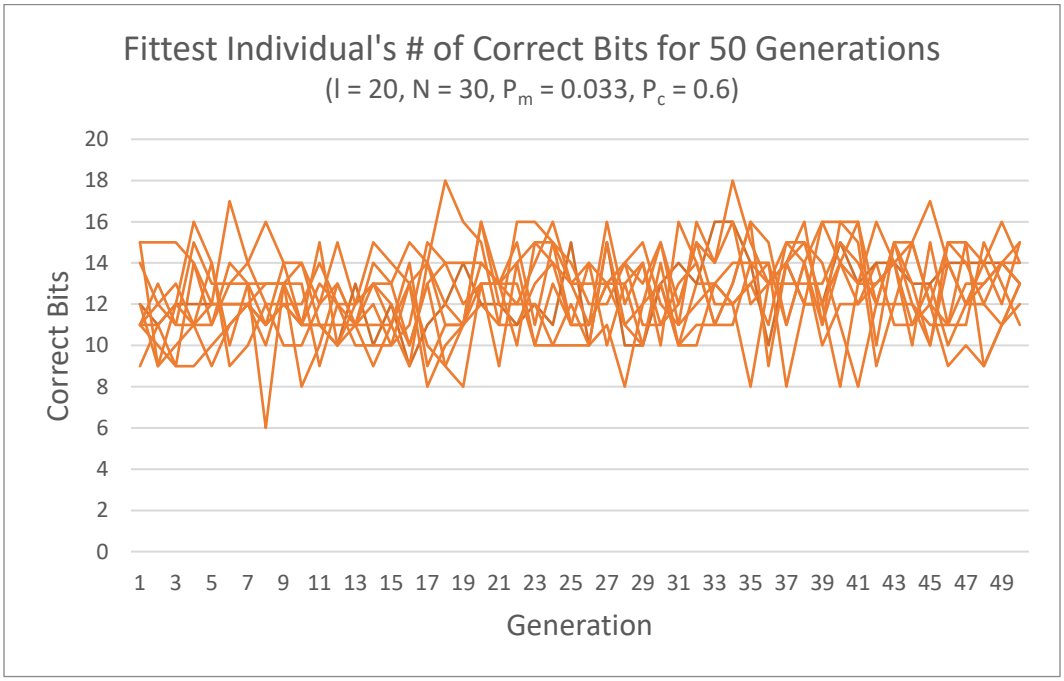

Figure 4) Very slight upward trend. At best, increasing generation size does not decrease the number of correct bits.

First, I increased the number of generations simulated to 50 to gain a bigger picture on how the other variables worked over time. As was predicted, the average fitness upward trend did continue past generation 10, but capped out right around generation 20. Since mutations exist and crossovers do not happen on individual bits, it incredibly difficult for a population's average fitness to reach 1.0. You can see though that the best individual's fitness does converge very nicely on 1.0.

Additionally, the number of correct bits does seem to trend upward slightly, and the range gets a tad smaller by generation 50.

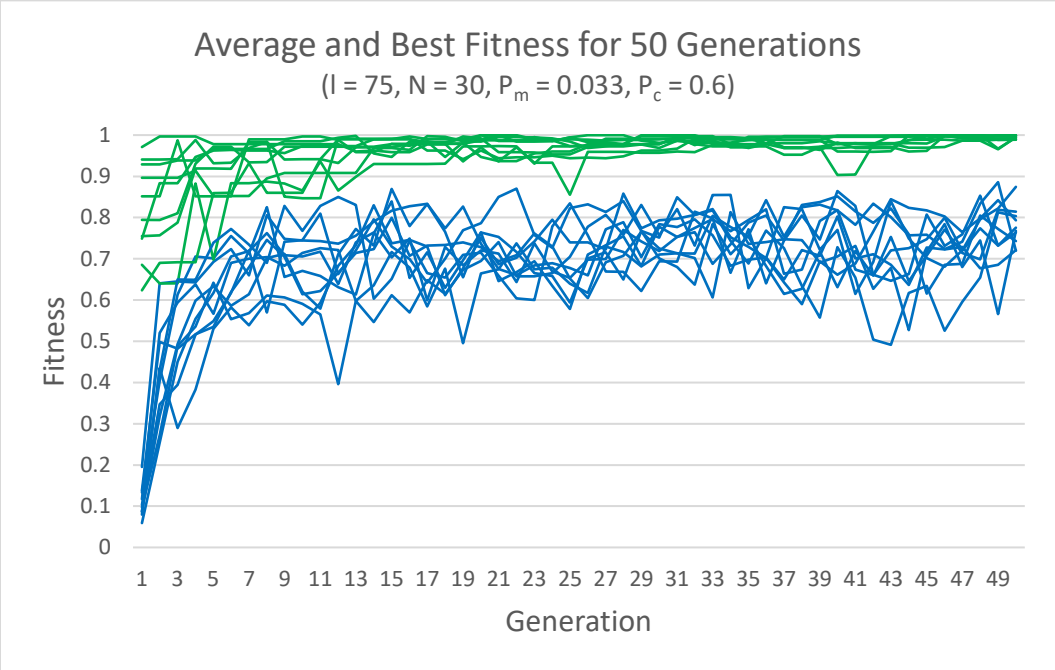

Figure 5) Increasing the number of genes does not appear to affect the convergence of either average or best fitness.

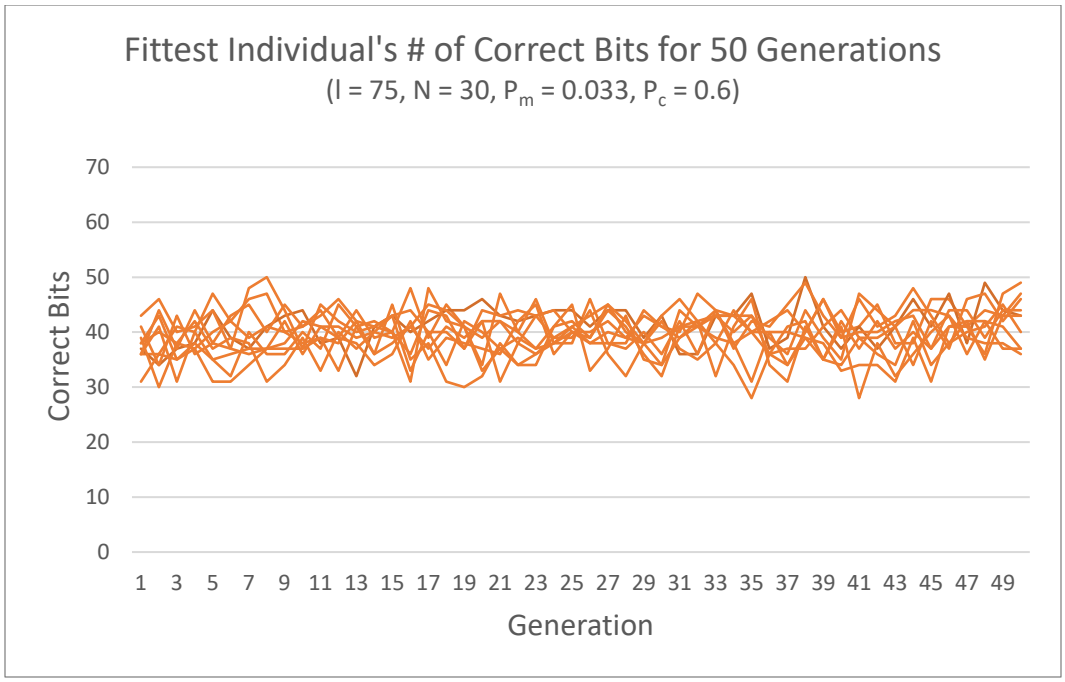

Figure 6) The range may appear to be tighter but it's simply the range of the graph. As before, the range of correct bits spans about 1/3rd of the total number of bits.

Next, I increased the genetic string length from 20 bits to 75. This introduced some noise to the average fitness but by generation 50 it reached around the same range as the simulation with 20 genetic bits (potentially even slightly higher). This could imply there is a stability that needs to be reached between genetic length and population size, the longer a genome the greater a population is needed to stabilize it.

The number of correct bits is in a range that is somewhat proportional to the simulation with 20 genetic bits. It does slightly trend upward, but not significantly enough to warrant investigation.

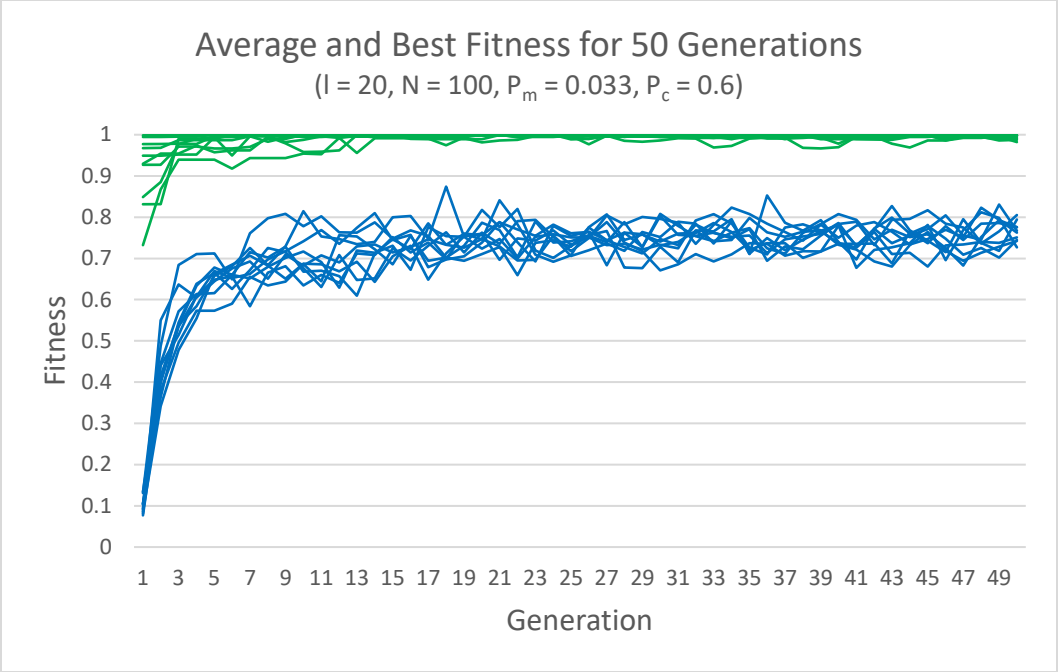

Figure 7) Increasing population size greatly increases the rate at which the best individual's fitness reaches 1.0. Average fitness also converges in a much smaller range between 0.7 and 0.8.

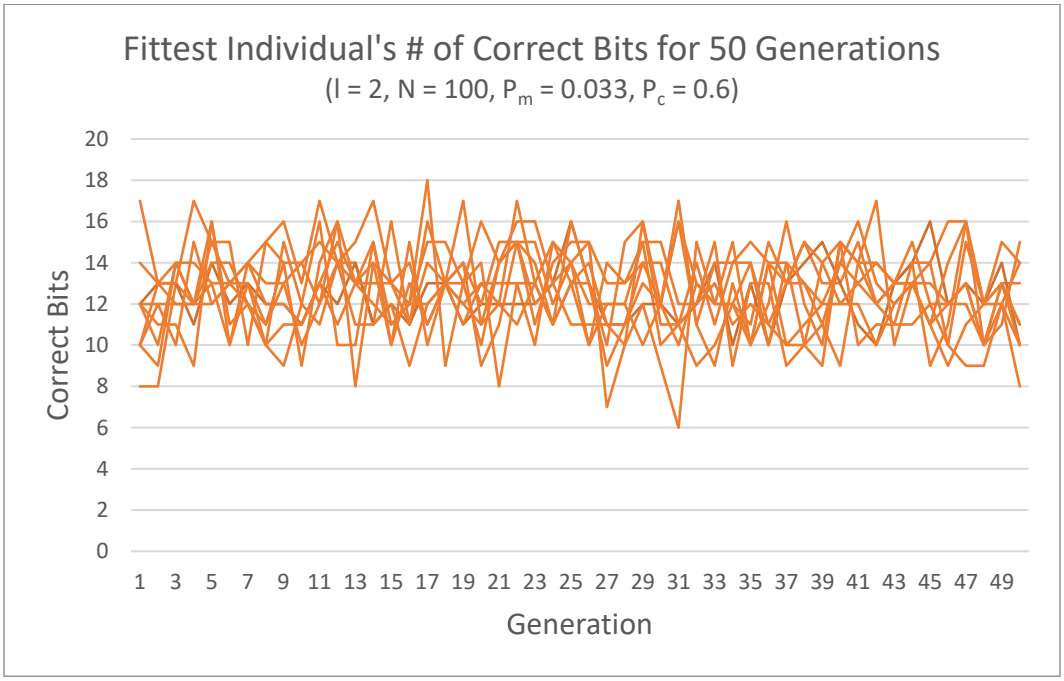

Figure 8) The range of the number of correct bits slightly increases with increased population size due to outliers.

Next, the population size was increased. As you can see in Figure 7 this results in a much faster rate for an individual with a fitness of 1.0 to be found. This makes sense, as selection with randomness is best countered by increasing sample size. This does not, however, make average fitness increase much. It does have a narrower convergence range between 0.7 and 0.8, however.

A higher population size seems to introduce noise to the number of correct bits in the best individuals' bit strings.

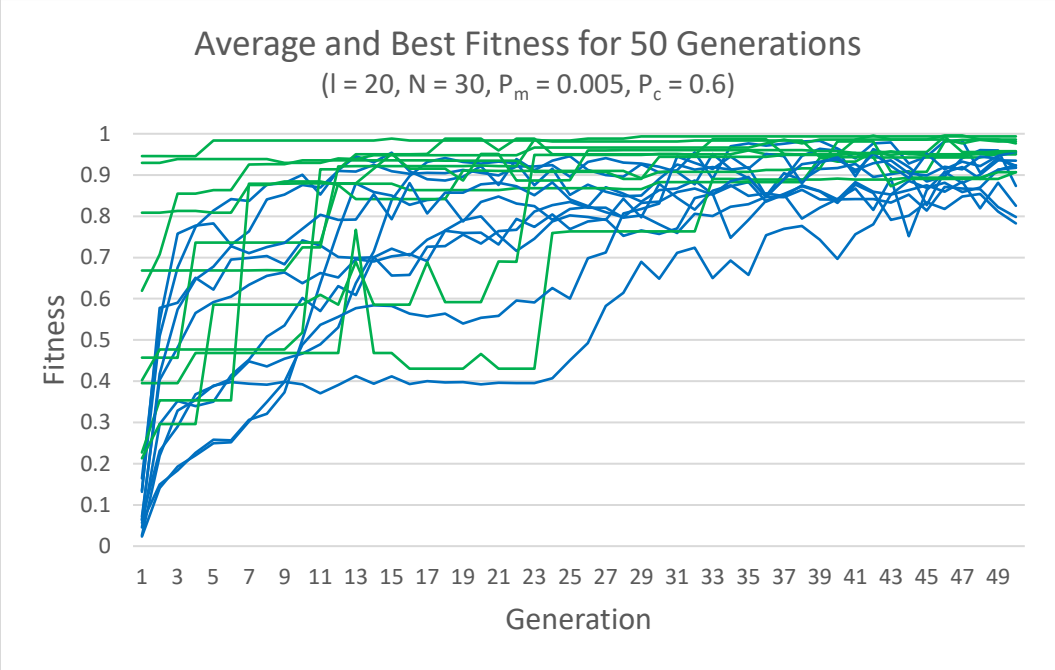

Figure 9) Decreasing mutation decreases the number of individuals with a high best fitness, but overall increases the range of average fitness at generation 50. Convergence is slower but better.

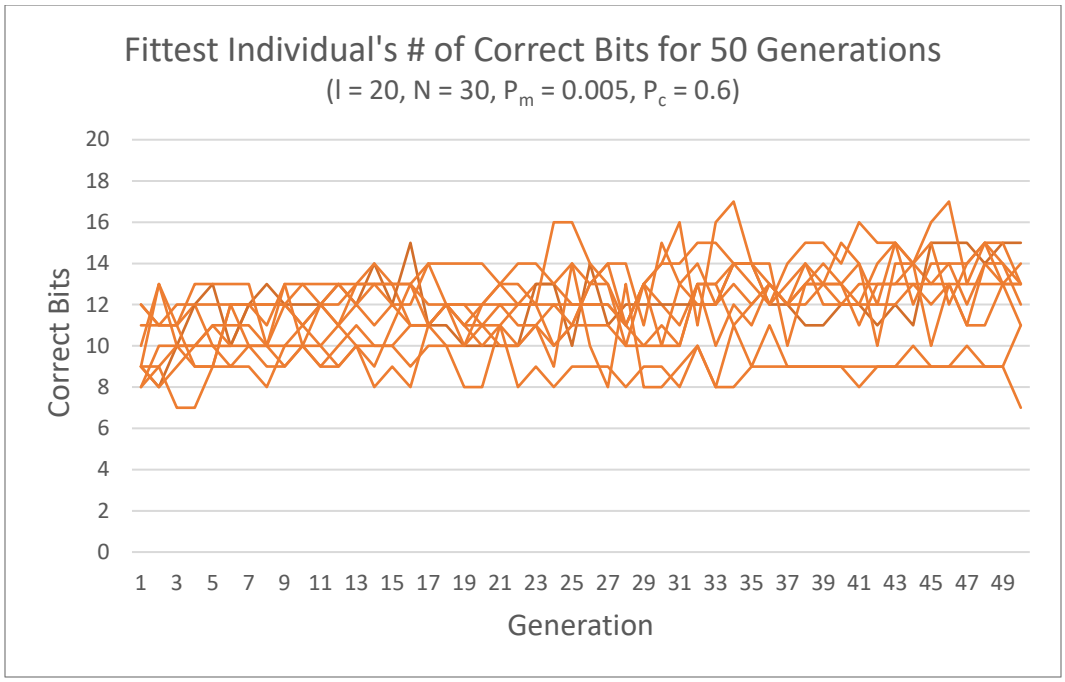

Figure 10) The number of correct bits trends upward more significantly with a decreased mutation rate.

By reducing the mutation rate to 0.005, populations are allowed to reach a much higher average fitness than previously seen, with some averages even ranging as high as 0.95. It makes sense that mutations directly counter high fitness, as once an individual has a majority of 1's, it becomes more likely that a mutation will flip a 1 to a 0 rather than the other way around, overall reducing its fitness.

The number of correct bits has a higher affinity to trend upward with a decreased mutation right. This can also be attributed to the reasoning I was just mentioning.

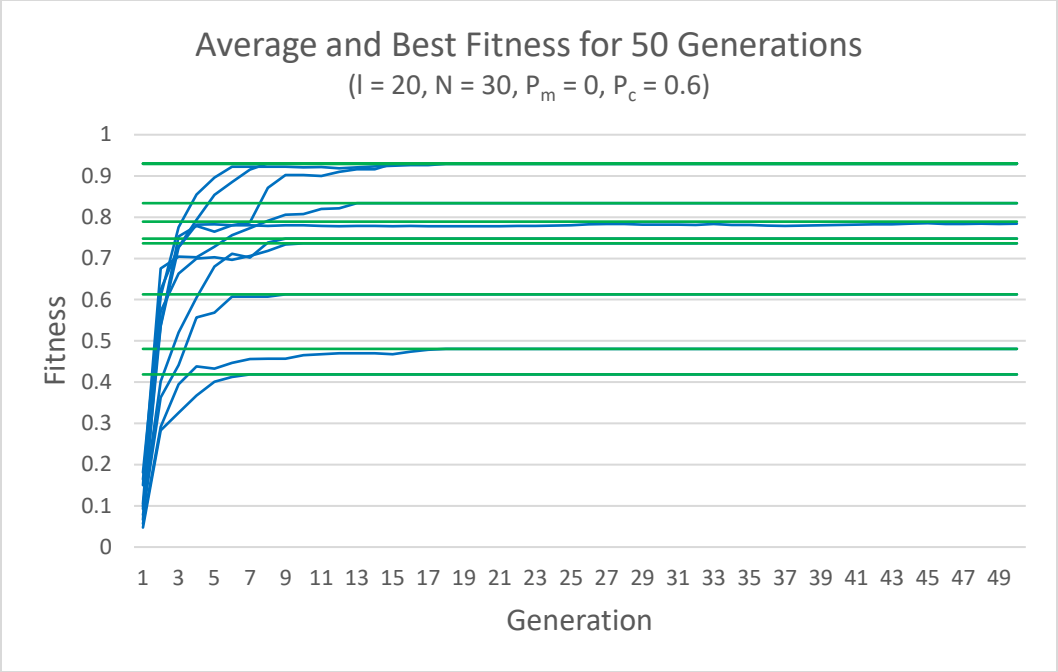

Figure 11) No mutation results in populations remaining fairly stagnant as the only way for their fitness to increase is through crossover.

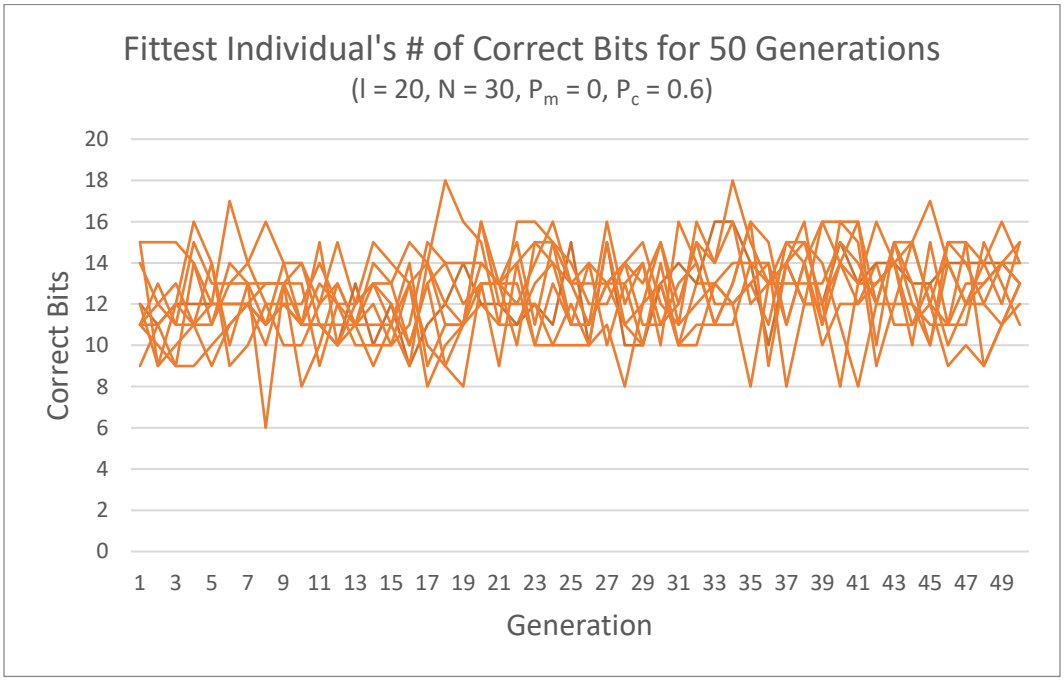

Figure 12) Having zero mutation does not significantly effect the number of correct bits over 50 generations.

Mainly just to see what would happen, I reduced the mutation rate to zero. This resulted in fairly quick stagnancy of fitness. Since mutation is how new bit strings are really introduced

into the population, change is much slower to take place. Once the best individual is created from a combination of the original generation's bit strings, there is no way for it to get better.

The number of correct bits is fairly uninteresting.

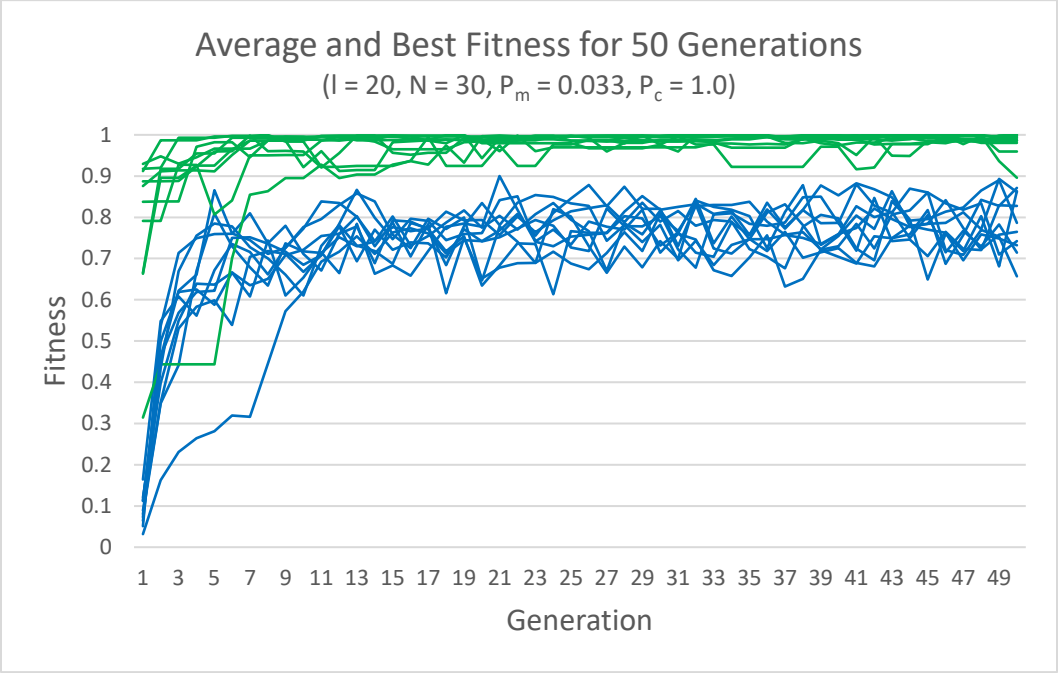

Figure 13) Having a 100% crossover chance does not significantly alter convergence either best or average fitness. It could be argued the average is slightly higher at generation 50 than with a crossover probability of 0.6.

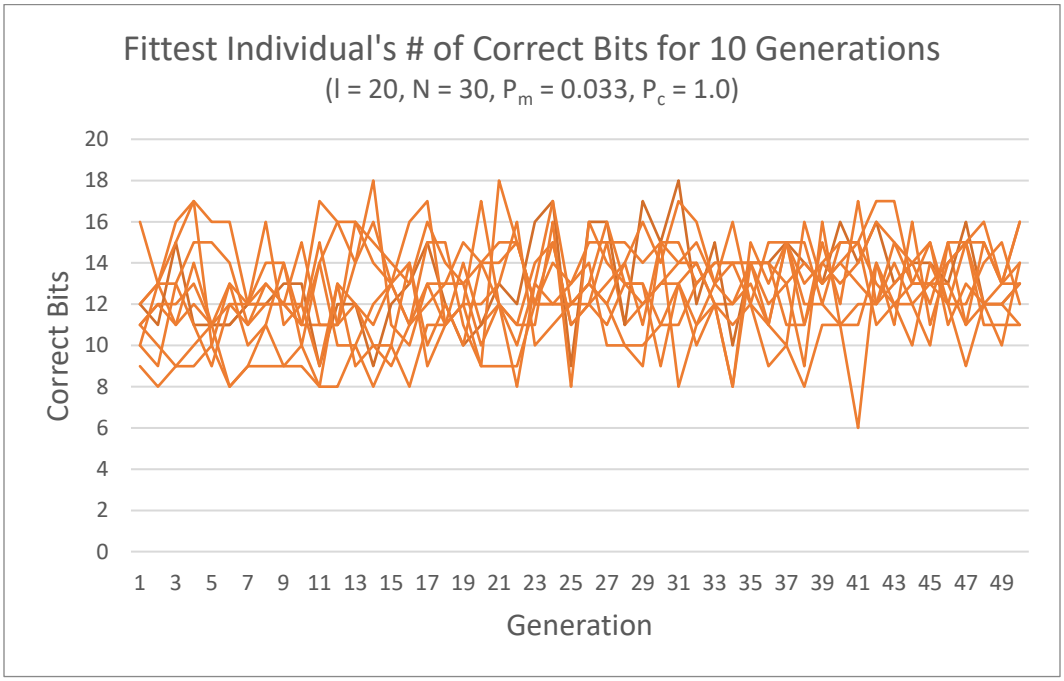

Figure 14) Nothing of particular interest occurs from increasing crossover probability.

Next, I increased the crossover probability to 1.0. This did not affect fitness or the number of correct bits much. It appears that the mutation rate sets a strict upper bound on how high the average fitness may go.

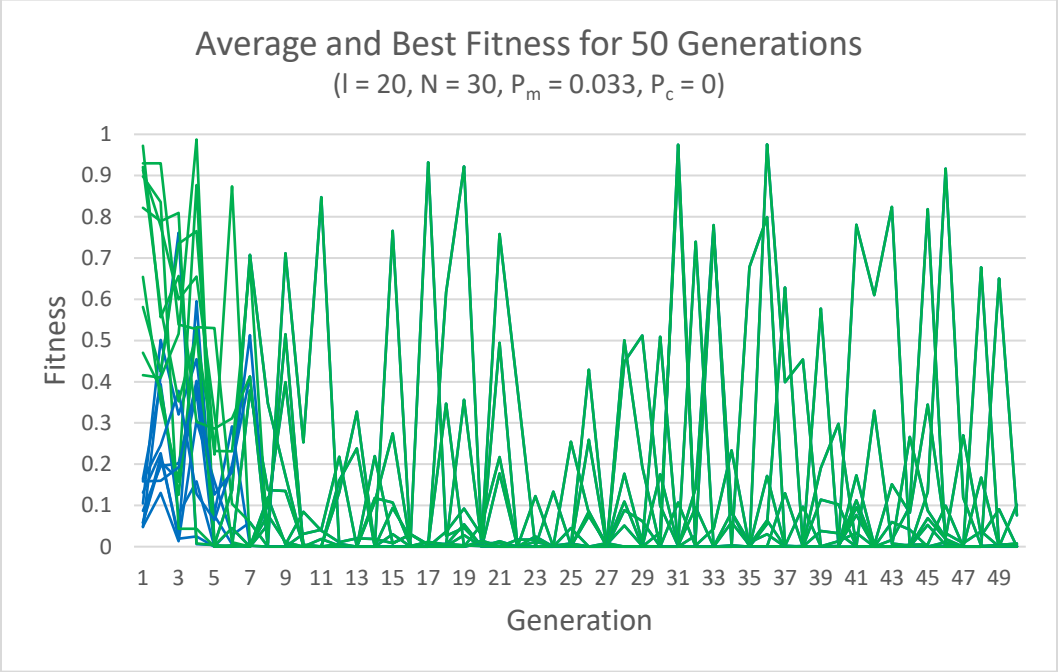

Figure 15) No crossover means that every subsequent generation is completely random based on which mutations occur.

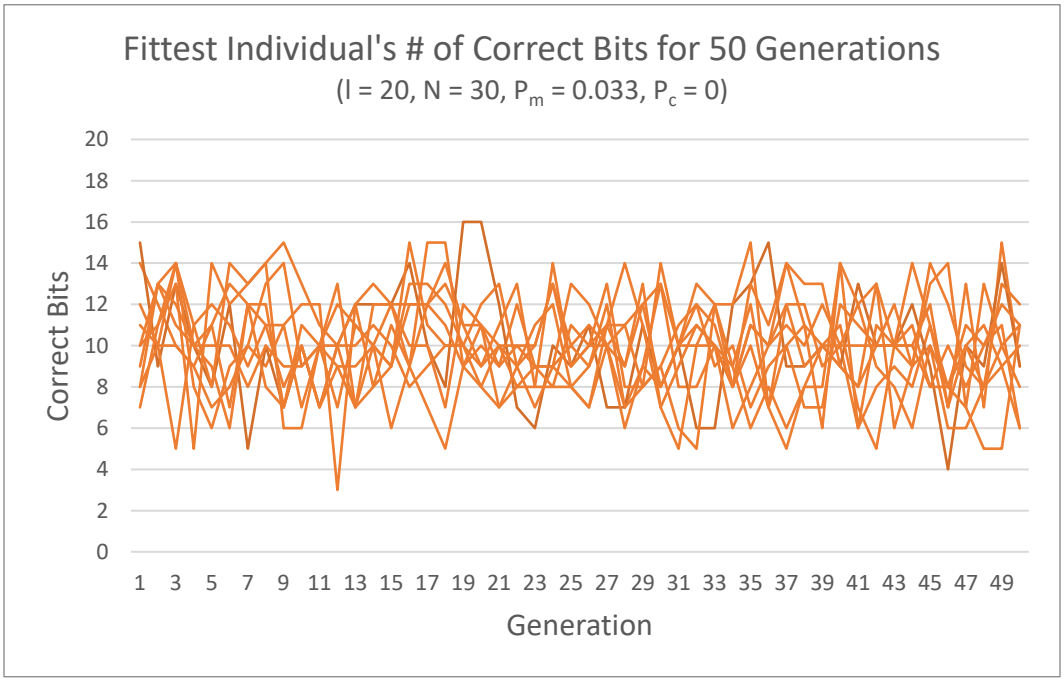

Figure 16) Number of correct bits is actually slightly lower than normal, indicating it's not entirely random.

Finally (and also very uninterestingly), I changed the crossover probability to 0. As could be easily predicted, this results in entirely random fitness. Since crossing over two fit parents is strictly how fitness is introduced into offspring, relying only on random mutations causes the mating of two parents to be pointless. Essentially you are just taking the parents from generation 1 and applying mutations to them G times. As is to be expected with any random system, though, some individuals do reach > 0.9 fitness.

This does confirm that the number of correct bits tends to hover around 50% since they are set randomly.

Conclusion

These results lead to a few ideas. First, a large genetic string size requires a large population to accommodate for greater variability in the selected bit strings. Second, mutation can argued to be both good and bad for a population. With mutation, populations are much quicker to reach their maximum average fitness, meaning much fewer individuals die off in the short term. However, without mutation a population is capable of reaching a higher average fitness. This comes at a cost, though, as many more individuals die off in the short time. That could be negligible though and may be countered over time by having a higher fitness. Finally, it can be assumed that mutation introduces an artificial cap on the average fitness of a population. Throughout all the experiments, no individual attained a perfect 20/20 bit string. As more and more bits get flipped to 1's in a string, a 1 becomes more likely to be flipped back to a 0 by random selection, making it very difficult for an individual to have a perfect string.