Genetic Algorithms

Samuel Steinberg

CS420

March 30th, 2019

Introduction:

This project investigates the implementation of a basic genetic algorithm and its effects on a given population over a number of generations. The concepts of selection, mating, mutation, crossover, and heredity are all considered for this project. The project used parameters representing the length of the genetic string, the size of the given population, the number of generations within the population, the probability of genetic mutation, and the probability of crossover. Genetic algorithms are inspired by natural selection and Darwin's theory of evolution. As a result of applying the theory and algorithms derived from nature, optimization (in this project it will be maximizing a quantity, which will be fitness) will occur. This same optimization learned from nature can have massive effects in the industrial process: saving time, money, and environmental/ecological harms. Small improvements in optimization, as will be shown in this project, can make a spectacular impact.

Theory and Methods:

The program to carry out the genetic algorithm takes the following parameters: length of the genetic string, size of the population, number of generations for the experiment to take place over, probability of mutation, and probability of crossover. Additionally, arguments were added for the number of runs to be performed for each combination of parameters (to get better, averaged data) and a unique ID number for each experiment. For each run, the bit string was randomly initialized. For each generation, the fitness was calculated using Figure 1 below:

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

Figure 1: Fitness Function

This equation calculates the fitness of an individual by taking the bit sum x and dividing by 2^l, where l is the length of the genetic string. The bit sum x is calculated by taking the left most bit and iterating down the string. If the bit is a 1, then that bit is given the value 2^i, where i is the index in the string (descending order), and that bit value is added to the bit sum. The quotient from the first portion of the equation is then raised to the power of 10 to get the fitness of the individual. The optimum genetic string (genotype) is a bit string of all 1's. The individual with the highest fitness will be considered the most fit individual in that generation. Each individual's fitness is added to a running sum to get the total fitness.

The normalized fitness was also calculated at the start of each new generation. Normalized fitness is important because decisions for mating and selection need to be scaled and makes sure that all resulting fitness values will equal 1. See Figure 2 below.

$$f_i^{norm} = \frac{f^i}{\sum_i^n f_i}$$

Figure 2: Normalized Fitness Function

The normalized fitness function will divide the fitness value (f^i) for the individual found in Figure 1 by the summed total fitness score of the population mentioned above.

After the fitness calculations, parents are chosen at random using the normalized fitness. This is performed by generating a random value and comparing it to the calculated normal fitness values. If an individual's value fits, it is chosen as a parent. Once two parents are selected; a random number is generated to determine whether crossover will take place. If crossover is performed a random number is generated to determine the point of crossover, and the bit strings of the parent are copied into the bit strings of the offspring up to that point. To see a visual representation of crossover, see Figure 3 below:

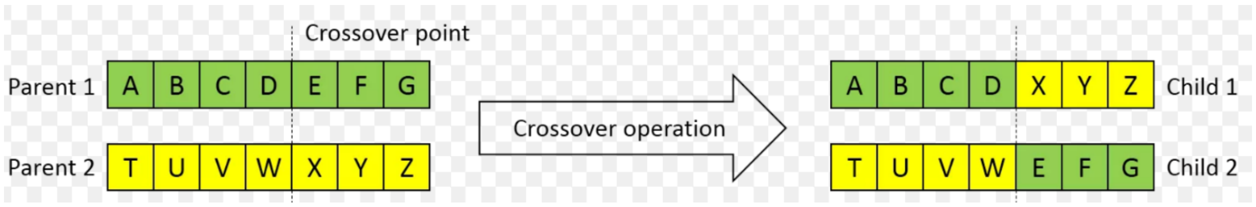

Figure 3: Simple Crossover Example

After the crossover point which offspring inherit from which parent are reversed. If there is no crossover, the parents bit string is copied into the bit strings of the offspring. In sum, crossover randomly mixes genes from both parents, and will result in genetic variation. Crossover points will lead to combinatorial optimization.

Next, any mutations are performed on the offspring. A random number is generated, and if mutation is to be performed on that bit, it is simply flipped. Mutation probability is generally very low. The population is then updated by copying all the offspring bit strings into the current population’s bit strings, and statistics (average fitness, best fitness, individual with most correct bits, and best individual string) for the population are written to a .csv file. Figure 4 below is a visual representation of the flow of the genetic algorithm used in this project:

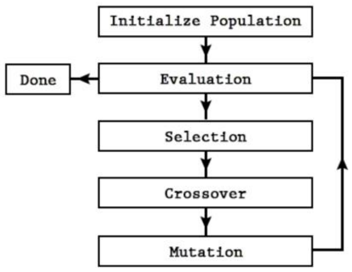

Figure 4: Flow of a Genetic Algorithm

For each different set of parameters, the program will generate a different .csv file to be written to. The graphs for this project are then generated in Excel. Each parameter set is organized using the unique ID number passed in with the arguments.

Results and Discussion:

In this project 17 different experiments were conducted. Each experiment tested a different set of parameters, that were used to test how the average fitness in a population, most fit individual in a population, and the individual with the most correct bits in a population change with different combinations. The gene length, population size, number of generations, probability of mutation, and probability of crossover were isolated and manipulated to see their effect. (NOTE*: In Best vs. Average charts the run number is represented by the number accompanying the letter, and a prepend of 'A' represents average fit, and 'B' represents best fit.)

A. Base Case

This project was provided with a suggested set of parameters that were chosen to act as the base case for comparing the effects parameter manipulation has on a population. The parameters and their values, represented in the charts, are shown below:

Parameters: Gene Length = 20, Population Size = 30, Generations = 10, Mutation Probability = 0.033, Crossover Probability = 0.6

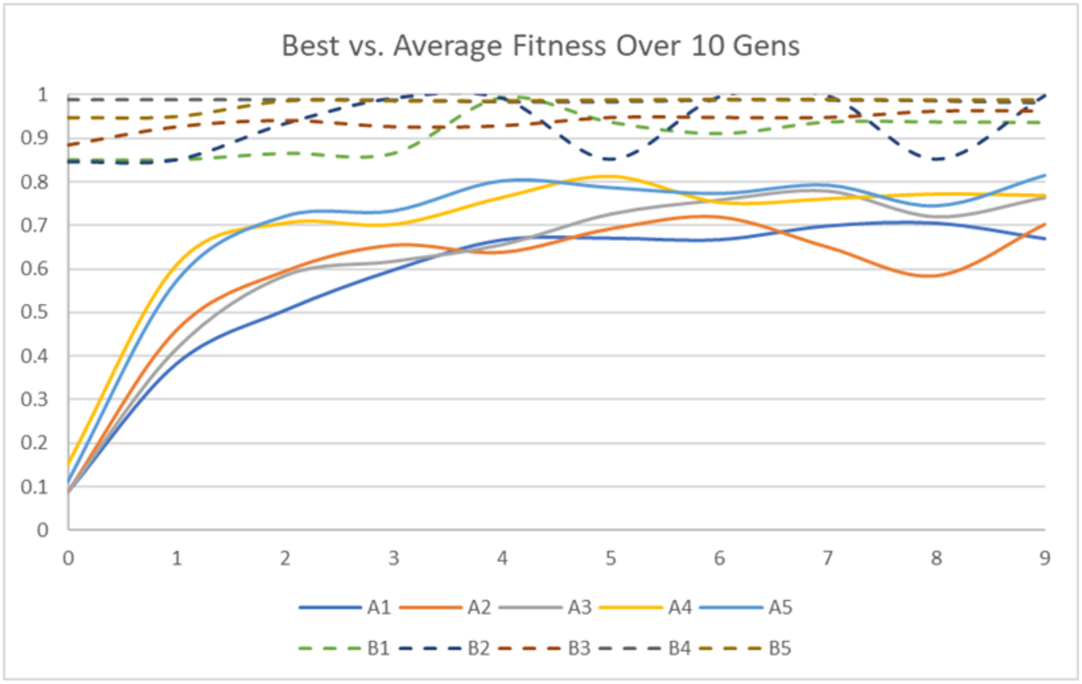

Figure 5: Base Case Best vs. Average

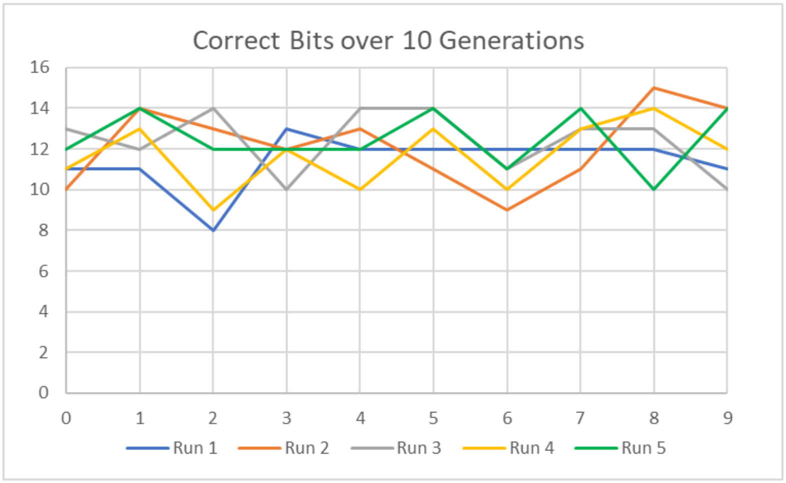

Figure 6: Correct Bits for the Base Case

Figures 5 shows a steady increase in average fitness until the 4th generation, where it then steadies; oscillating between slightly better average fitness and slightly worse average fitness. The most fit individual and the number of most correct bits (Figure 6) stayed steady for the most part, besides the 2nd run. The most fit individual averaged out at ~.95, significantly higher than the average fitness (after steadying) at around 0.72.

B. Mutation Combination Experiments

This project tested different combinations of mutation values with the other base case parameters. Mutation probability is very low but can make a huge impact if a significant bit is flipped (e.g. from a 1 to a 0 or a 0 to a 1). This gives mutation a chance to alter both best fitness and average fitness significantly; and can cause chaos.

Parameters: Gene Length = 20, Population Size = 30, Generations = 10, Mutation Probability = 0.1, Crossover Probability = 0.6

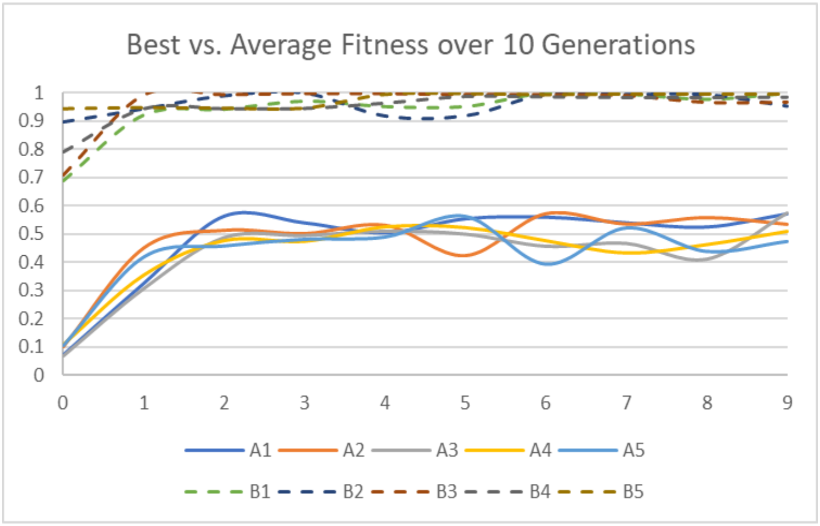

Figure 7: Note the decrease in average fitness with the lower mutation probability.

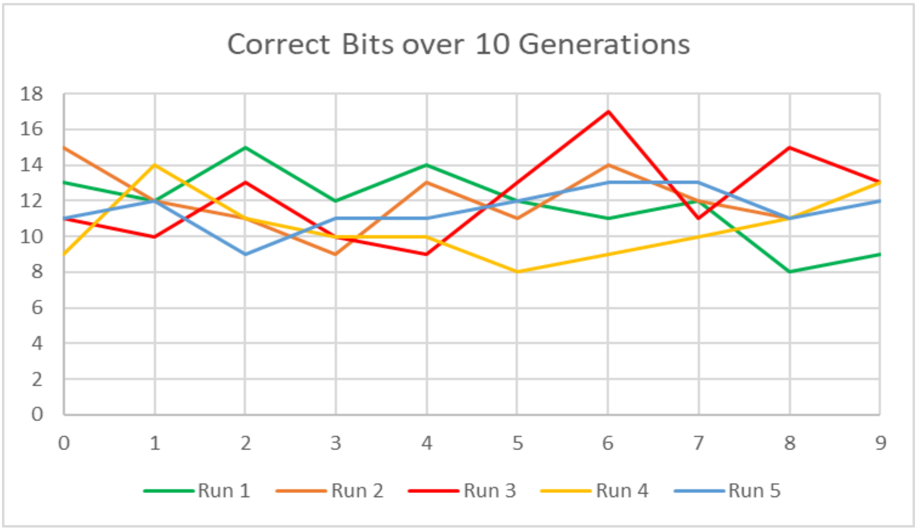

Figure 8: Best Bits did not change too significantly.

Figures 7 and 8 represent a mutation slightly higher than the base case, at 0.1. As seen in Figure 7 the average fitness lowered around ~.2 from the base case and adds credence to the hypothesis that mutation changes can significantly alter fitness. The experiments conducted in this project found that a larger, but realistic, mutation probability overall lowered fitness in a population. The best fitness values, at realistic values, unsurprisingly did not change much. The chance that a fit individual did not have any gene values significantly altered is still favorable. As seen in Figure 9 below, when lowering the mutation probability back down the average fitness jumps to higher values again:

Parameters: Gene Length = 20, Population Size = 30, Generations = 10, Mutation Probability = 0.06, Crossover Probability = 0.6

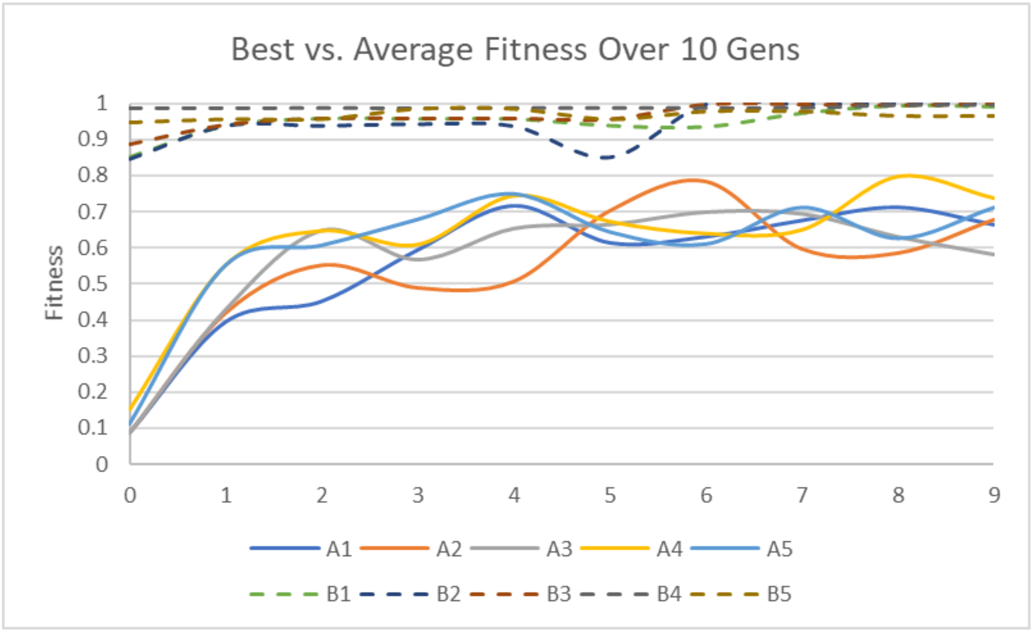

Figure 9: Note that fitness shoots back up after decreasing the mutation probability, and that there is much more chaos.

Average fitness is back into the ~.7 range. This is much more similar to the base case, though it is worth noting that it is much more chaotic from generation to generation. The chaos mutation can cause can be seen when mutation is raised to a quite high and unrealistic value, seen in Figure 10. This experiment was run to show the chaos that mutation can breed: The most fit individual, depending on what value the significant bit will be flipped to, will be either very high or very low. Accordingly, average fitness will always be very low. This shows that too much mutation in a population will lead to complete chaos, and lower values are necessary for a population to survive. Additionally, it was found that a lower mutation value usually leads to a higher overall fitness.

Parameters: Gene Length = 20, Population Size = 30, Generations = 10, Mutation Probability = 0.8, Crossover Probability = 0.6

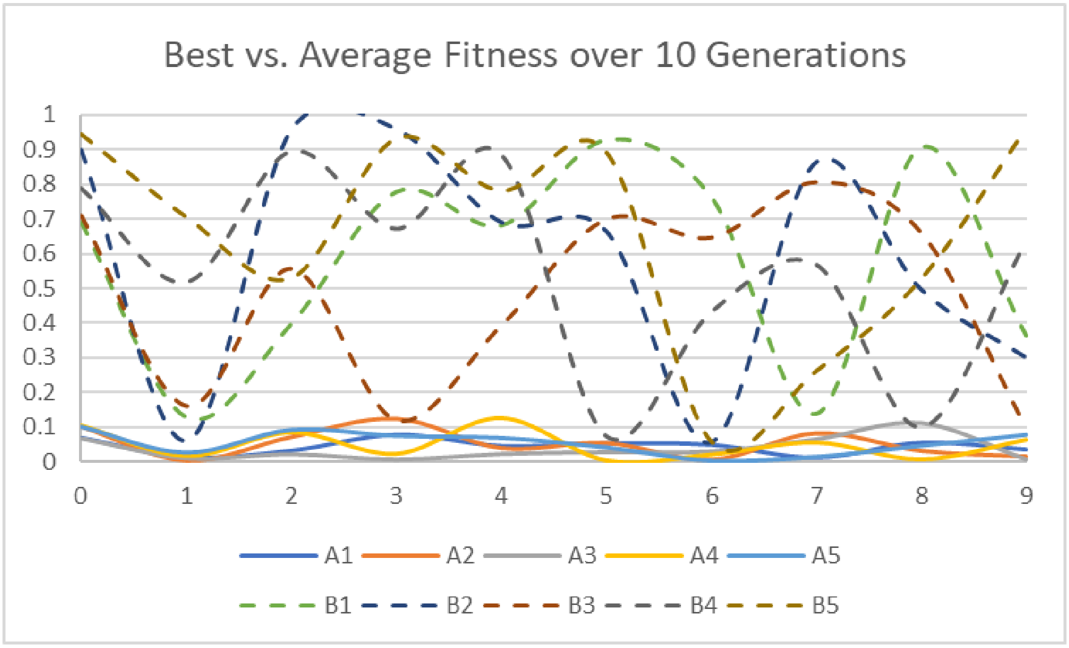

Figure 10: Note the complete and utter chaos taking place.

C. Crossover Combination Experiments

The testing of crossover in this project found that while it creates genetic variation and randomness, it does not significantly affect the fitness of individuals in the population. This can be seen by contrasting two significantly different probabilities of crossover:

Parameters: Gene Length = 20, Population Size = 30, Generations = 10, Mutation Probability = 0.033, Crossover Probability = 0.2

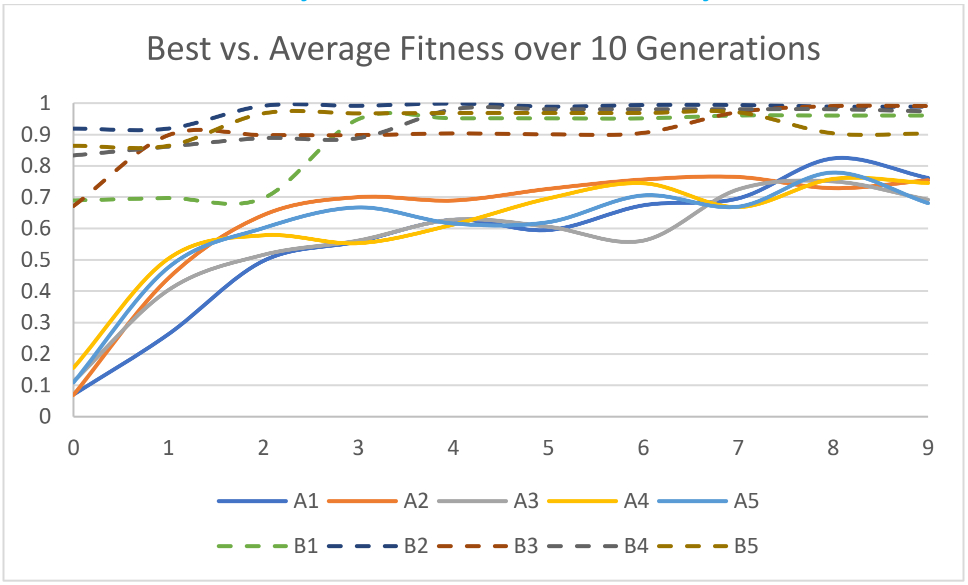

Figure 11: Fitness values with a crossover probability of 0.2 did not yield a significant difference.

Parameters: Gene Length = 20, Population Size = 30, Generations = 10, Mutation Probability = 0.033, Crossover Probability = 0.8

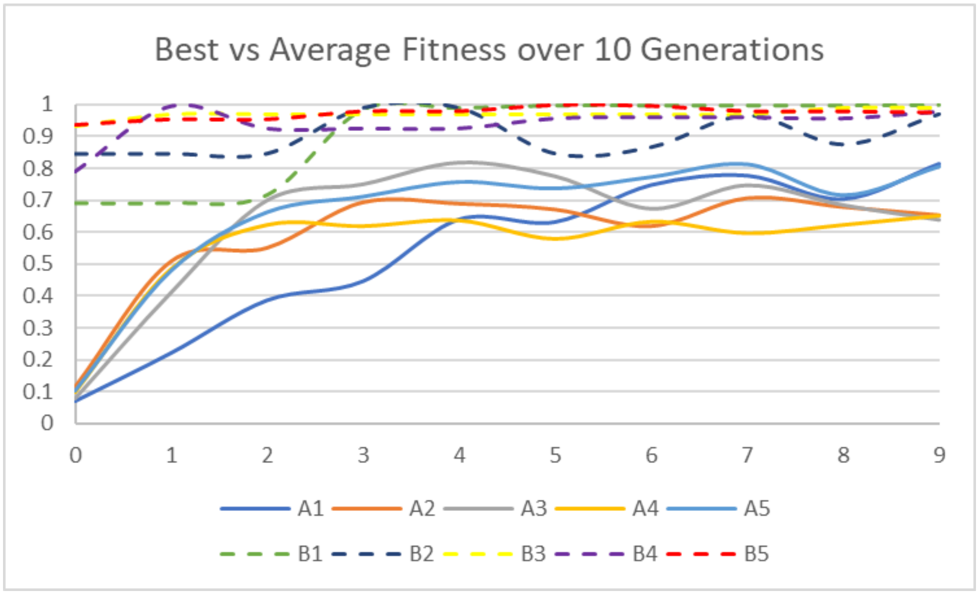

Figure 12: Fitness values did not alter much, even at 0.8

Even with significant changes in crossover probability, the graphs generally remained the same. The randomness that crossover creates proves to create genetic variation, but does not have a huge effect on the resulting fitness values. It is worth noting, however, that the first two generations in Figures 11 and 12 are lower than normal (around .7), and the best fitness values are a bit more varied when crossover is higher. Withstanding this, the values generally even out.

D. Population Combination Experiments

The combinations for population size proved to significant; to an extent. Larger populations do not change the data, since the data is based off generations. Smaller populations, however, do make a serious impact by manufacturing a significant amount of noise and chaos.

Parameters: Gene Length = 20, Population Size = 10, Generations = 10, Mutation Probability = 0.033, Crossover Probability = 0.6

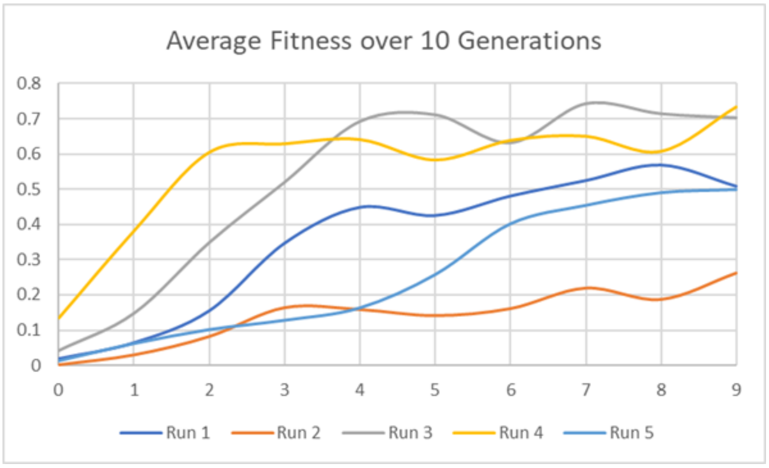

Figure 13: Note the chaos that takes place with a small population size

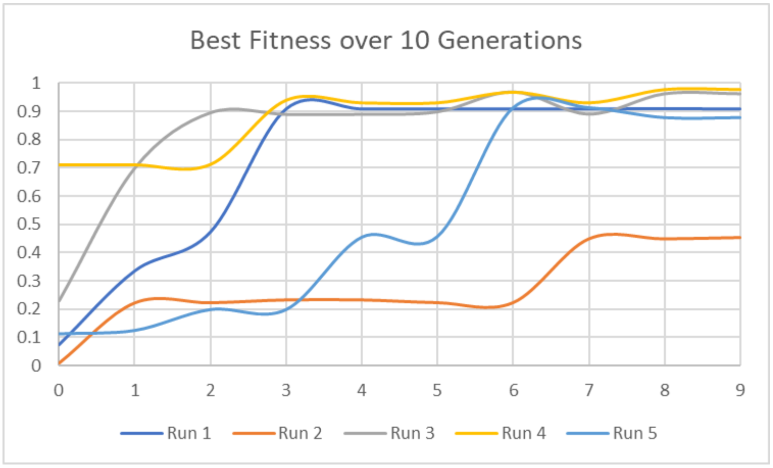

Figure 14: Note the inconsistencies in the best fitness. Without many individuals, a single highly fit or highly unfit individual can make a significant impact on fitness data

Figures 13 and 14 represent the average fitness and most fit individual, respectively. There is a significant amount of chaos taking place throughout the generations with a small population size (10 in this case), with a single individual able to impact the data significantly. It can be seen in the Figures that the average fitness curves for the runs tend to follow their best fitness curves. Looking more closely at Run 4; a lower best fitness resulted in a lower average fitness. This might seem to be an obvious case, but the curves are nearly an inverse of each other: When best fitness is stagnate at a value above the average fitness, the average will climb, and when it falls the opposite happens. With larger populations the effect of these most fit individuals is far less, as seen in Figures 15 and 16 below, this time with a population of 100:

Parameters: Gene Length = 20, Population Size = 100, Generations = 10, Mutation Probability = 0.033, Crossover Probability = 0.6

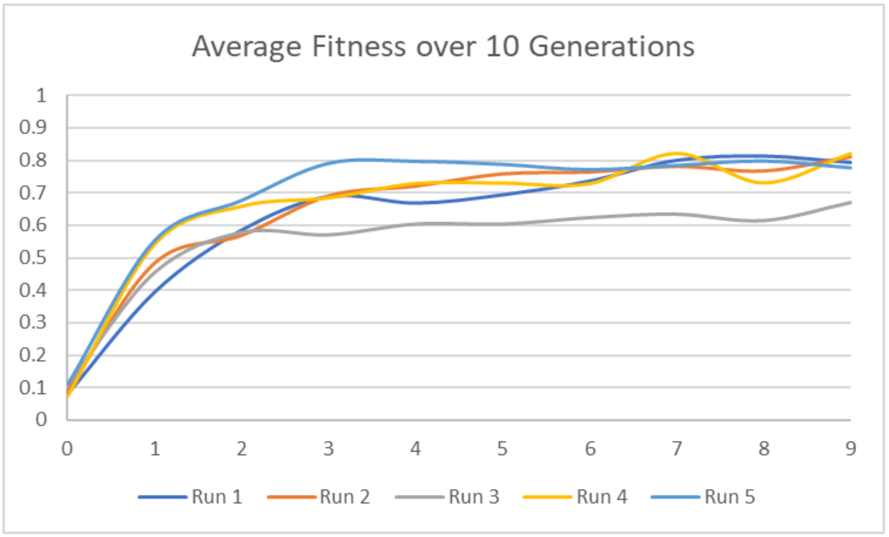

Figure 15: Average fitness does not absorb as much impact from large populations

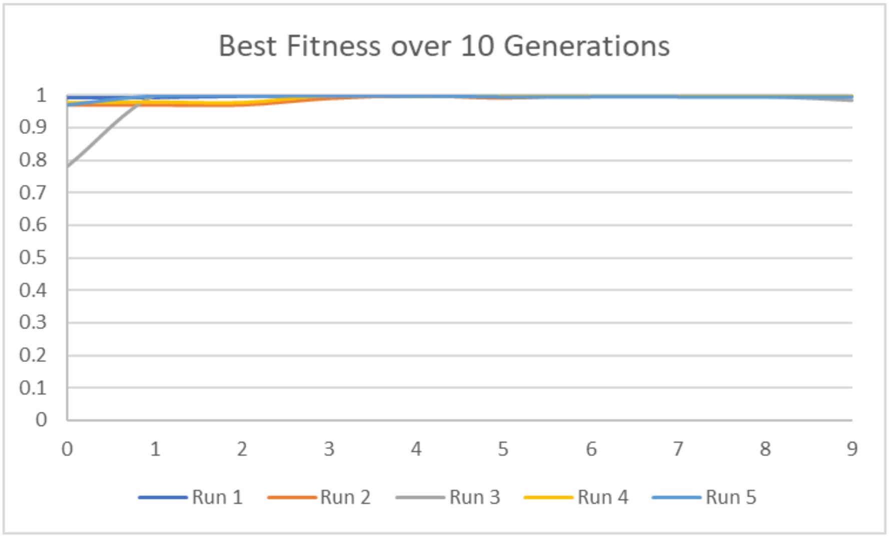

Figure 16: Best fitness does not affect average fitness nearly as much in large populations

Fitness does not take nearly the same impact from best or worst fit in large populations as compared to smaller populations. Smaller populations are much more prone to have data skewed by single individuals, such as a very fit individual significantly raising average fitness. Larger populations are much more diverse and are better prepared to handle outlier individuals.

E. Gene Length Combination Experiments

The testing of gene length in this project concluded that gene length does not have a significant impact on fitness levels. Both small and large bit strings proved to not differ much, which can be explained by the random activation of individual genes in initialization and the fact that fitness is scaled. This is illustrated in the two experiment sets below:

Parameters: Gene Length = 8, Population Size = 30, Generations = 10, Mutation Probability = 0.033, Crossover Probability = 0.6

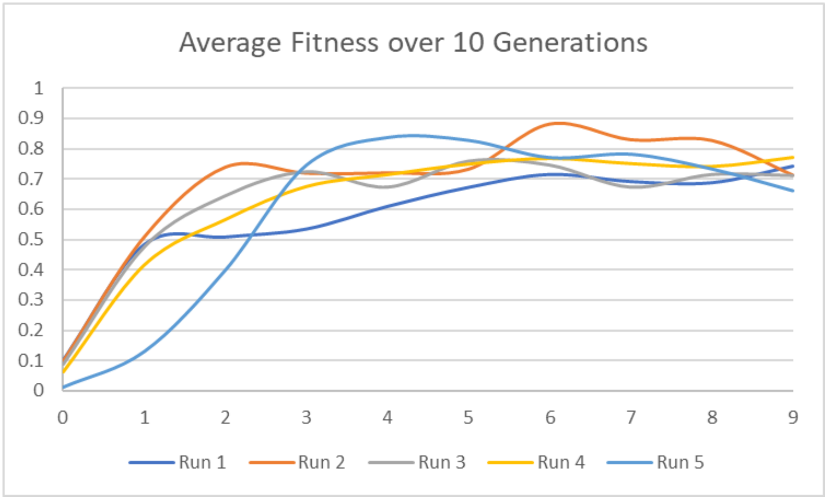

Figure 17: Average fitness seemed to stay along the baseline in a short genetic string of length

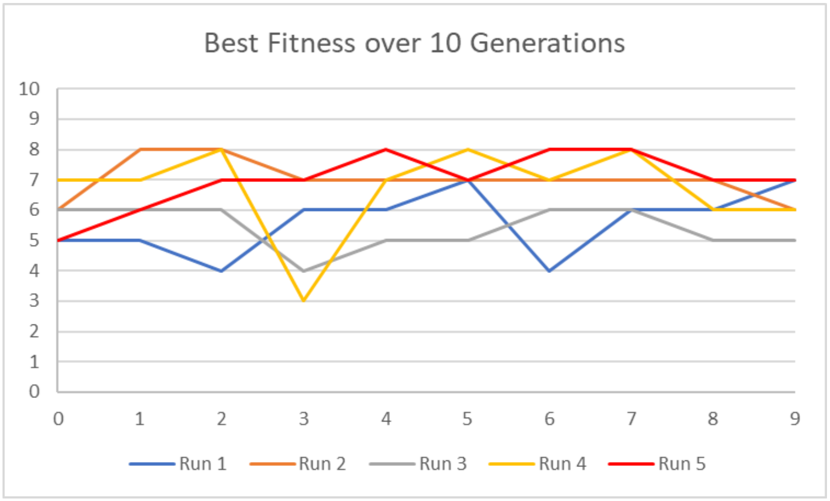

Figure 18: Best Bits are also scaled according to the length of the genetic string

Parameters: Gene Length = 30, Population Size = 30, Generations = 10, Mutation Probability = 0.033, Crossover Probability = 0.6

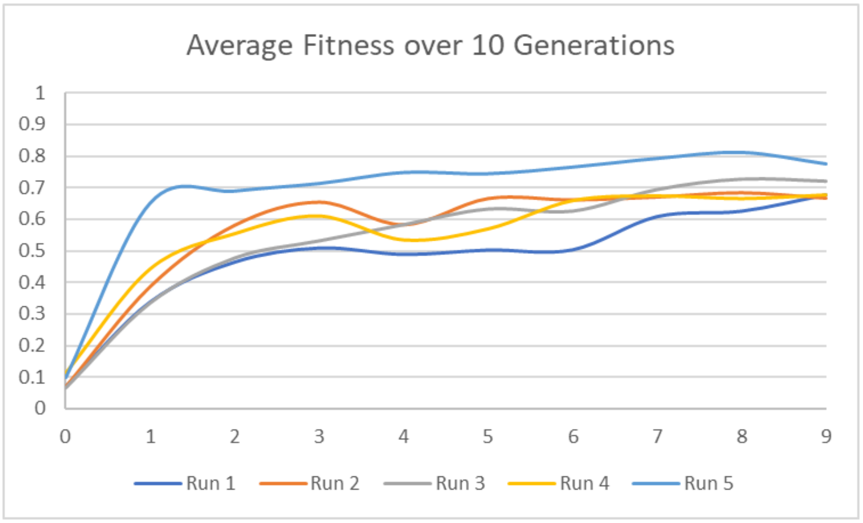

Figure 19: Average fitness in this longer genetic string averages to nearly the same as a short string

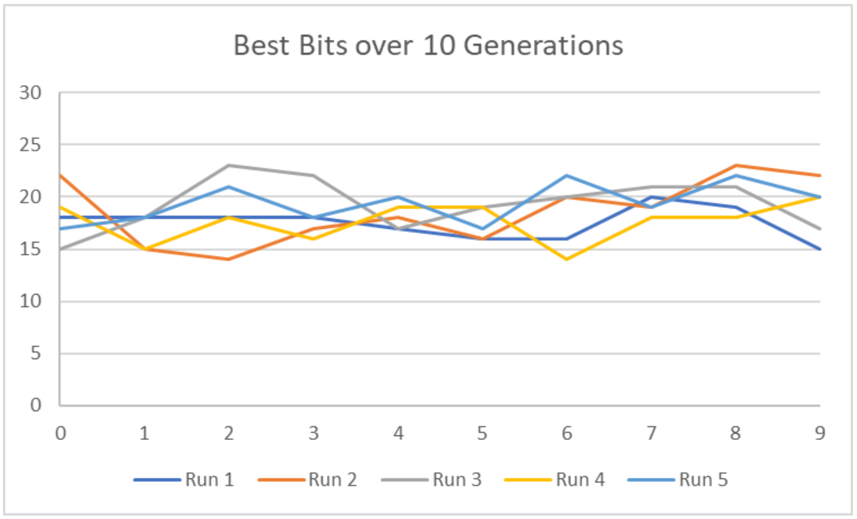

Figure 20: Best Bits are at their expected values when compared to other mid-large genetic strings

Due to fitness being scaled, the average fitness of these genetic lengths do not change significantly from each other and the base case (since other parameters are the same), and average around ~.7. It is worth noting, however, that due to having a short genetic string, these will have a greater chance to have a string consisting of all 1's. This can be seen in Figure 18, while it would be nearly impossible to have a perfectly fit individual in a larger genetic string (see Figure 20). This could potentially lead to slightly higher best fitness values.

F. Generation Combination Experiments

Since the data was calculated against the generation of the population, it was fairly easy to assume the behavior of this combinations data set. Allowing only a small number of generations potentially places limits on how the fitness will behave, and could restrict the full development of average fitness and the most fit individual.

Parameters: Gene Length = 20, Population Size = 30, Generations = 5, Mutation Probability = 0.033, Crossover Probability = 0.6

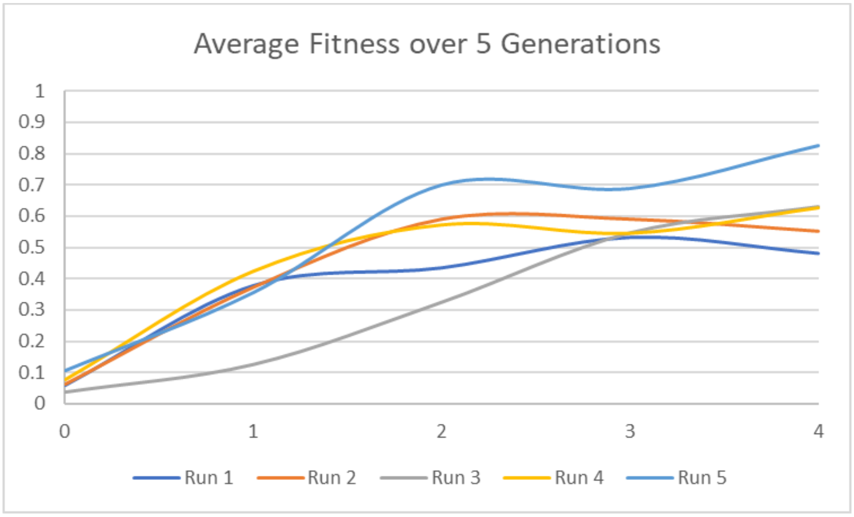

Figure 21: With only 5 generations, it is difficult to get a full picture of the average fitness data.

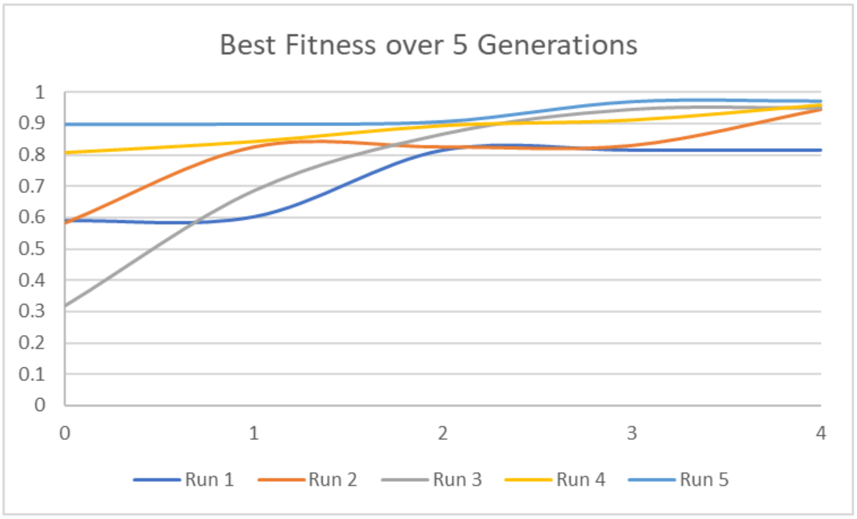

Figure 22: The most fit individual is on the lower side for the first few generations of this run, but seemed to be converging at the expected values by the end of the 4th generation

With only a small number of generations it is difficult to get a full picture of the data. There is not enough time to see the fitness steady (since this is using base case parameters besides the generation numbers), and is convergence is not definitively shown. Below, in Figures 23 and 24, with a larger number of generations convergence becomes clear:

Parameters: Gene Length = 20, Population Size = 30, Generations = 30, Mutation Probability = 0.033, Crossover Probability = 0.6

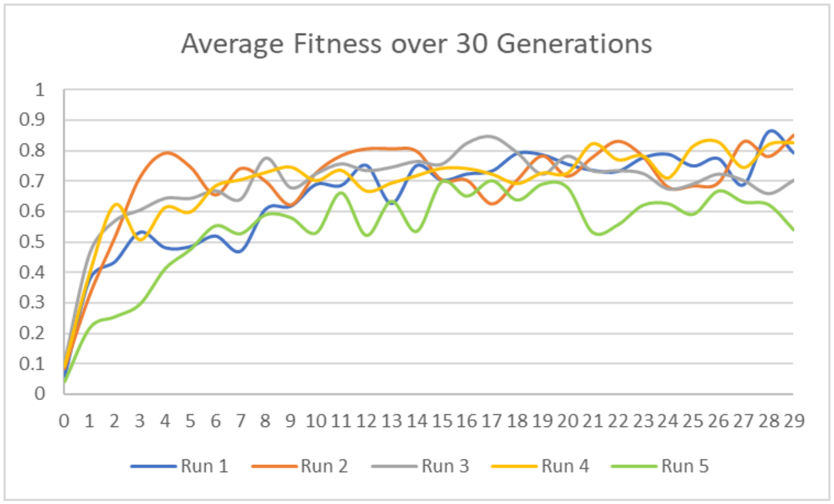

Figure 23: Though this data is a bit more chaotic than normal, there is still a clear range for convergence

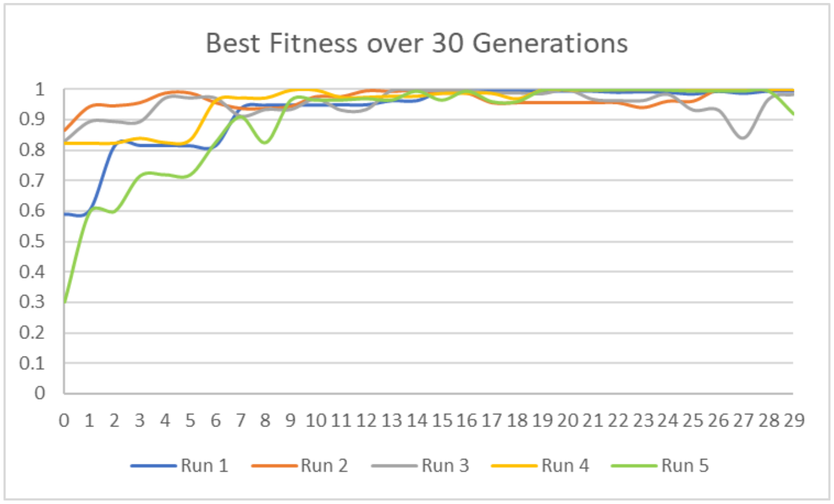

Figure 24: In best fitness there is a clear convergence

With more generations, the data is allowed to develop and converge. In general, runs tended to take their mature shape at around generation 7 or 8. Figures 23 and 24 serve to illustrate the indifference the number of generations make after maturing.

G. Experiment Parameters Master List

Table 1 below shows all the parameter combinations used in this project. As shown, parameter combinations were used to isolate certain measures and conclude which parameters made what kind of impact on the data. Of the 17 experiments, there was a base case, three mutation experiments, three crossover experiments, four population experiments, three gene length experiments, and three generational experiments. Isolating these parameters was vital to getting an accurate, un-cluttered view of how each of them impacts the data and was important for interpreting the data without another bias interfering.

Experiment ID	Gene Length	Population Size	Number of Generations	Mutation Probability	Crossover Probability	Number of Runs
1	20	30	10	0.033	0.6	4
2	20	30	10	0.06	0.6	4
3	20	30	10	0.1	0.6	4
4	20	30	10	0.8	0.6	4
5	20	30	10	0.033	0.2	4
6	20	30	10	0.033	0.5	4
7	20	30	10	0.033	0.8	4
8	20	10	10	0.033	0.6	4
9	20	20	10	0.033	0.6	4
10	20	40	10	0.033	0.6	4
11	20	100	10	0.033	0.6	4
12	8	30	10	0.033	0.6	4
13	15	30	10	0.033	0.6	4
14	30	30	10	0.033	0.6	4
15	20	30	5	0.033	0.6	4
16	20	30	15	0.033	0.6	4
17	20	30	30	0.033	0.6	4

Table 1: All experiments and their parameter combinations

Conclusions:

The conclusion this project reached was that mutation probability and population size were the most potent combinations for determining how a system will perform over time. Mutation probability, though usually quite small, has the potential to add chaos and noise into a system. These changes can significantly alter the fitness (both average and best) of a population and, when manipulated in the experiments, proved to be the case. In general, a lower mutation probability will end up leading to higher fitness in a system. Population size can also have a significant effect on the fitness of a system. Low populations can create a significant amount of noise because a single individual is able to alter the data significantly (with a very high or very low value). Larger populations will make less of an impact on a system, since the data set will be much greater and more adaptive to outlier individuals.

Though crossover is perhaps the most interesting parameter, creating significant genetic variation in a population, it did not have a very significant effect on the data. Though randomness is achieved in the population the fitness is generally unaffected. Gene length was another parameter without much of an impact on the data. Since this project uses a scaled fitness value, a smaller gene length would not be much different from a larger length in terms of the data it produces (outside of the most correct bits, obviously). Combinations of generations were not a helpful indicator of the health of a system. Since the health is being judged on a generational scale, allowing a population very few generations will only give incomplete data when compared to more generations. The data will not be allowed to converge.

In this project statistics were kept for the average fitness in a population, the most fit individual in a population, the most correct bits in an individual in a population, and the genetic string of the most fit individual. The average fitness was the best statistical measure for viewing the overall health of a system, and was therefore considered the most important measure for determining the success or failure different combinations of parameters produced in this project. The most fit individual and its genetic string made for interesting viewing, as it made it apparent that correct bit placement made a much greater impact on fitness than simple number of correct bits. The number of correct bits was generally a useless measure when comparing the data.