# CS420 Project 5:
# Genetic Algorithms

Robby Ferguson

April 25, 2016

## Introduction

Project 5 focused on the implementation of genetic algorithms and the effects of different parameter combinations on the general effectiveness of these systems. Put simply, genetic algorithms act as simplified models of genetics and evolution through natural selection. They are generally based on a random initial "population" of individuals, which change over time from generation to generation according to some sort of metric. These systems are most widely applied to optimization problems where some sort of "fitness" measure can be maximized. The merit of potential solutions to these optimization problems is quantified by some variant of a fitness function that is crafted such that higher fitness values are assigned to solutions that most closely match the desired outcome or behavior. In other words, the fitness of each species or individual in the system is gauged by this fitness function. Like in real-world biological systems, the more fit individuals in a particular generation tend to be more likely to reproduce, which results in a greater chance that desirable characteristics will be passed through generations and proliferate within a population over time.

To further complete the analogy to actual biological systems, genetic algorithms incorporate some form of reproduction into the process of producing a new generation such that the characteristics of individuals morph over time. To actually make these systems useful, a fitness-based selection method is used to increase the likelihood of more-fit individuals reproducing. By adding a probabilistic aspect to the reproduction phase, rather than simply choosing the best individual to reproduce each time, the system as a whole is less likely to converge prematurely when a better solution could have been found. To prevent this convergence, algorithmic analogues to genetic crossover and mutation can be implemented, which tend to increase the genetic diversity of subsequent generations within a population while still preserving some of the more prominent characteristics of particular individuals.

## Methodology

The primary objective of Project 5 was to create software that simulated a basic genetic algorithm and study the behavior observed within those simulations. Each individual in the population was represented by a genetic string, which was a simple stream of '0' or '1' bits. The simulation software was required to take several variable parameters as input:

1. Number of genes (L) in each individual's genetic string (L)
2. Population size (N)
3. Mutation probability (Pm, the probability of mutating each bit within an arbitrary individual's genetic string)
4. Crossover probability (Pc, the probability of single-point genetic crossover occurring during the reproduction process)
5. Seed for a random number generator

6. Number of generations (G) to run in the simulation (where a generation replaces all the individuals in the population; therefore, if a "breeding cycle" replaces two individuals, then there will be N/2 breeding cycles per generation. For each breeding cycle, two parents are probabilistically selected according to their fitness levels and two new children are added to the population in their place.)

The fitness function used to evaluate each individual in the population was defined in the following way:

$$F(s) = \left(x/2^L\right)^{10}$$

where *x* was the integer that resulted from interpreting an individual's genetic string *s* as an unsigned binary number. According to the fitness function then, the optimal individual would have a genetic string consisting entirely of '1' bits.

To start each simulation run, the initial population (generation 0) consisted of N individuals with entirely random genetic strings each of length L. The following process was repeated for each generation until G generations had been produced and processed:

1. Calculate the fitness of each individual in the current population.
    a. Convert each individual's binary genetic string into an integer *x*.
    b. Pass that *x* into the fitness function to calculate the current individual's fitness level.
    c. Keep a running sum of the population's fitness values across all N members.
    d. Normalize each individual's fitness value with respect to the total fitness of the population.
    e. For each individual, also keep a running total of normalized fitness values of all individuals before it and its own fitness.
2. Perform N/2 iterations of the reproductive process.
    a. Select two individuals to be parents.
        i. Generate two random numbers between 0 and 1.
        ii. For each random number, find the two closest cumulative normalized fitness values (step 1e) that the random number is between.
        iii. The two individuals in the population that correspond to the upper bound of the range surrounding each random number will become the parents of two offspring in the next generation.
        iv. Continue trying to select the second parent until it is not equal to the first parent (to avoid self-mating).
    b. Mate parents and perform any crossover to get offspring.
        i. Generate a random number to determine whether crossover will be done. This random number must be less than the

defined crossover probability Pc for crossover to take place between these two parents.
   ii. If no crossover is done, directly copy the bit strings of the parents into the new bit strings that will represent the offspring.
   iii. If crossover is done, randomly select a bit to be the crossover point (single-point crossover). Copy the bit strings of the parents the offspring bit strings up to the crossover point, and then reverse which offspring gets the bits from which parent following the crossover point.
c. Perform any mutations on the offspring.
   i. For the two offspring currently being considered, iterate through each individual's bit string.
   ii. At each index, generate a random number to indicate whether that bit will be mutated. The random number must be less than the mutation probability Pm for a bit to be mutated.
   iii. If mutation takes place in one of the offspring, flip the bit at that location in the individual's genetic string. Otherwise, skip over that bit.

3. Update the population.
   a. Copy the offspring bit strings into the container of bit strings representing the current population (previous generation). The offspring will replace the old population.
4. Calculate relevant statistics pertaining to the previous generation to observe how patterns emerge over time.
   a. Find the average fitness of that population.
   b. Find the fitness of the best individual.
   c. Find the number of correct bits in the best individual.

In order to estimate the general trends that should emerge for a particular set of parameters in one simulation, each combination of parameters was rerun 5 different times (each of these is considered a "run," while the collective set of 5 runs is considered a "simulation"). In this way, outliers became easier to ignore and more common patterns became easier to observe and consider.

The primary goal of this project was to observe the effect of each of the specified parameters on the overall effectiveness of the system. To do this, the parameters were initialized to particular base values, and several simulations were run together such that each of them altered a particular parameter, but the other parameters were kept constant across the set of simulations. Each of these sets of simulations (where a single parameter's influence was being examined) is referred to as an "experiment" (i.e., one experiment was the set of simulations in which the population size was varied). The base values for the invariant parameters were taken from the Project 5 description, which described the following values as being good starting points: N = 30, L = 20, G = 10, Pm = 0.033, and Pc = 0.6. Each of these parameters was separately adjusted and studied. Additionally, the seed for each

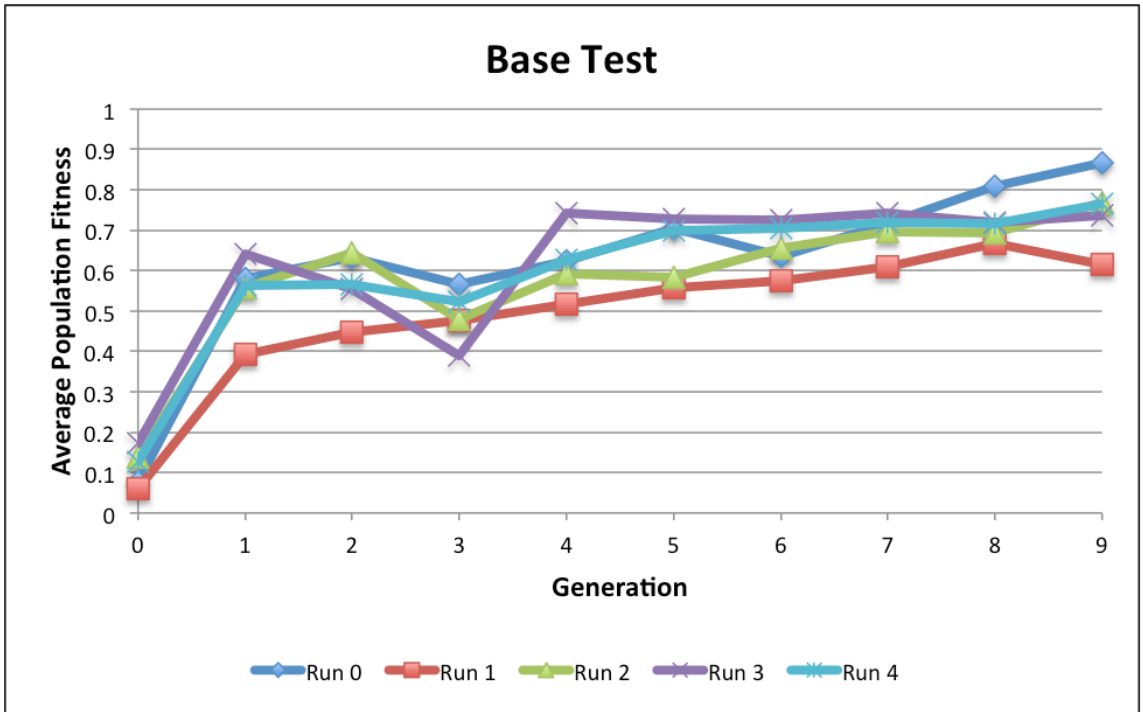
simulation (each of the 5 runs were executed consecutively using the same initial seed) was recorded into the respective data files, but those seed values are not included in the majority of figures or explanations in this report.

When varying the different parameters, related charts would be too convoluted if each of the 5 runs for each simulation (of which there were at least 5 in each experiment) were all recorded together in one figure. To determine which runs to graphically include, the average fitness values of the population in the last generation were considered. The last generation was chosen because those values would most accurately reflect the general convergence behavior of the population over the specified time span. To choose the run that would reflect the average behavior of the system, the run that corresponded to the median average population fitness value in the last generation was used.

## Data/Graphs/Discussion

### Base Test

Initially, the parameter values specified above were examined before any variations were made. The average population fitness across each generation for each of the 5 runs can be seen in the following chart:

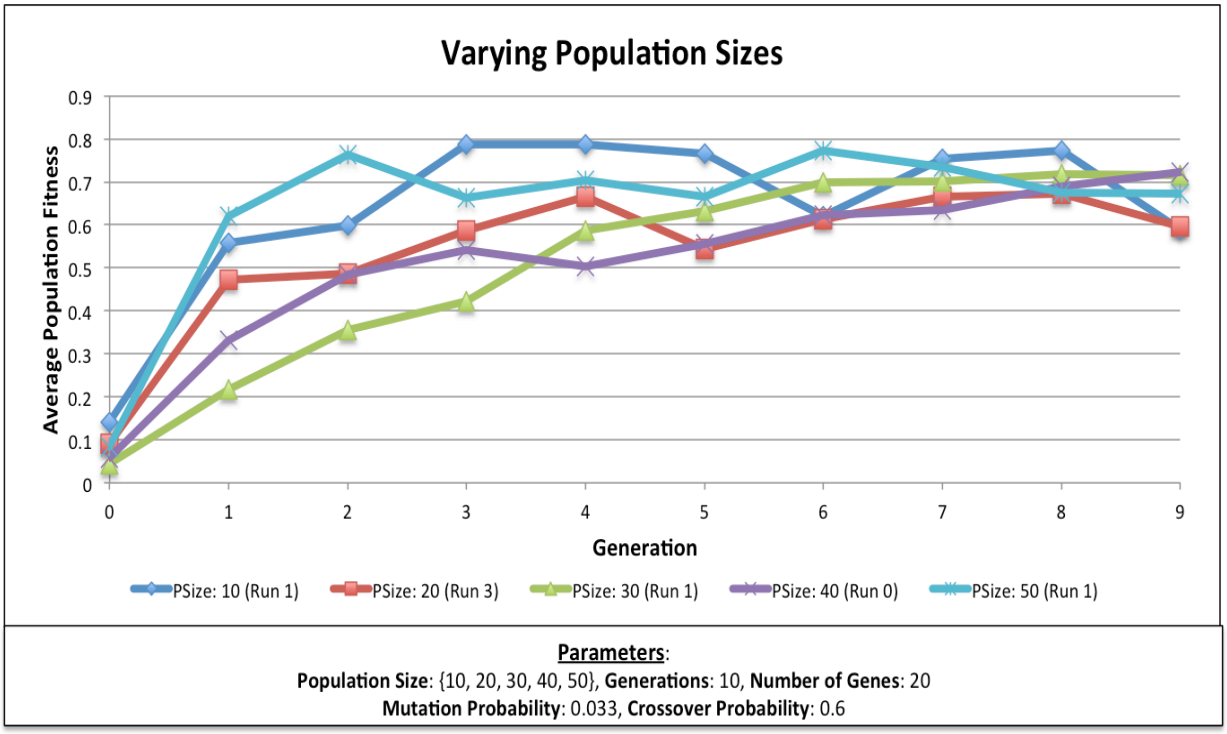



**Parameters**:
**Population Size**: 30, **Generations**: 10, **Number of Genes**: 20
**Mutation Probability**: 0.033, **Crossover Probability**: 0.6, **Seed**: 3856281650

Each of the runs in the above figure are based on the exact same combination of parameters but developed from different, random initial populations. As it shows, the results of these systems have the potential to vary rather drastically from run-to-run, even when no adjustments are made. From this point, each of the parameters were systematically varied and studied.

**Population Size**

To start, the population size within each simulation was altered. Separate simulations were run with population sizes of 10, 20, 30, 40, and 50 individuals, and the resulting trends for the average population fitness across each generation can be seen below.




By the final generation, most of the average population fitness values for these typical simulation runs had converged to similar numbers. Interestingly, the first two runs that reached peak average population fitness values were from simulations that used a population size of 50 individuals and a population size of 10 individuals, respectively. By the third generation, the fitness of the 50-individual population had already peaked and seemed to shakily level off for the subsequent generations. The 10-individual population did not peak until the fourth generation, but it reached a slightly higher maximum than the 50-individual population did, and the 10-individual population maintained the highest average population fitness until it dropped heavily on the seventh generation.

In the final generations, the average population fitness levels for each simulation run had converged somewhere between 0.6 and 0.7. Although the simulations with population sizes of 10 and 50 were able to quickly reach maximal values in early generations, both runs were characterized by relatively unstable fluctuations in average population fitness. Neither of these values for the population size parameter provided the best results for the general population fitness.

In fact, by the tenth generation, the 30-individual population actually had the highest average population fitness. This simulation was characterized with the steadiest increase in average fitness levels over time. Unlike most of the other simulations, the average population fitness of the 30-individual population consistently increased, and it dropped very slightly only from the seventh to the eighth generation. Similarly, the average fitness of the 40-individual population seemed to remain relatively stable between generations. From this observation, a population size somewhere in the range of 30 to 40 may be best suited for maximizing the average fitness of the population as a whole (when the other parameters are kept near the base values).

In addition to the average population fitness shown in the previous chart, the following table shows information about the best individuals in the final generation across all 5 runs of all the simulations in which population size was varied.

**Most Fit Individual in the Final Generation (Varying Population Sizes)**

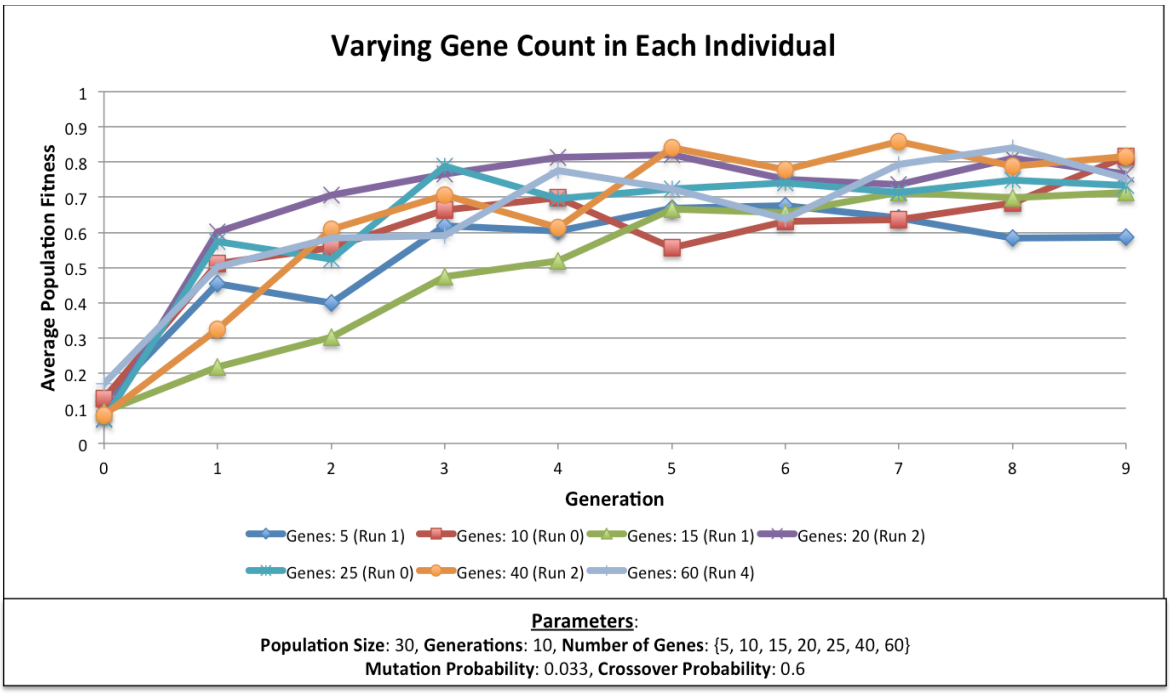
**Parameters**:
**Population Size**: {10, 20, 30, 40, 50}, **Generations**: 10, **Number of Genes**: 20
**Mutation Probability**: 0.033, **Crossover Probability**: 0.6

| Population Size | Run Number | Best Fitness in Final Generation | Number of Correct Bits | Genetic String of Best Individual |
|---|---|---|---|---|
| 30 | 3 | 0.998399 | 15 | '11111111111101011000' |
| 50 | 3 | 0.99779 | 14 | '11111111111100011000' |
| 10 | 3 | 0.997675 | 14 | '11111111111100001100' |
| 30 | 4 | 0.997409 | 15 | '11111111111011110000' |
| 50 | 2 | 0.995878 | 16 | '11111111111001001111' |
| 40 | 1 | 0.995194 | 14 | '11111111111000000111' |
| 10 | 4 | 0.993989 | 13 | '11111111110110001000' |
| 20 | 1 | 0.993866 | 17 | '11111111110101111011' |
| 50 | 1 | 0.992889 | 13 | '11111111110100010100' |
| 50 | 0 | 0.98972 | 15 | '11111111101111000101' |

When considering only the best individual at the end of each run, a population size of 30 seems to be most conducive to increasing individual fitness. However, the best individuals from runs with population sizes of 50 and 10 also trail very closely behind that top individual from the 30-population simulation. Perhaps population sizes of 10 and 50 are simultaneously good at producing particular individuals with high fitness values while also not as good at increasing the fitness of the population as a whole. On the other hand, population sizes of around 30 or 40 seem to be advantageous for both individuals and the general population.

**Number of Genes**

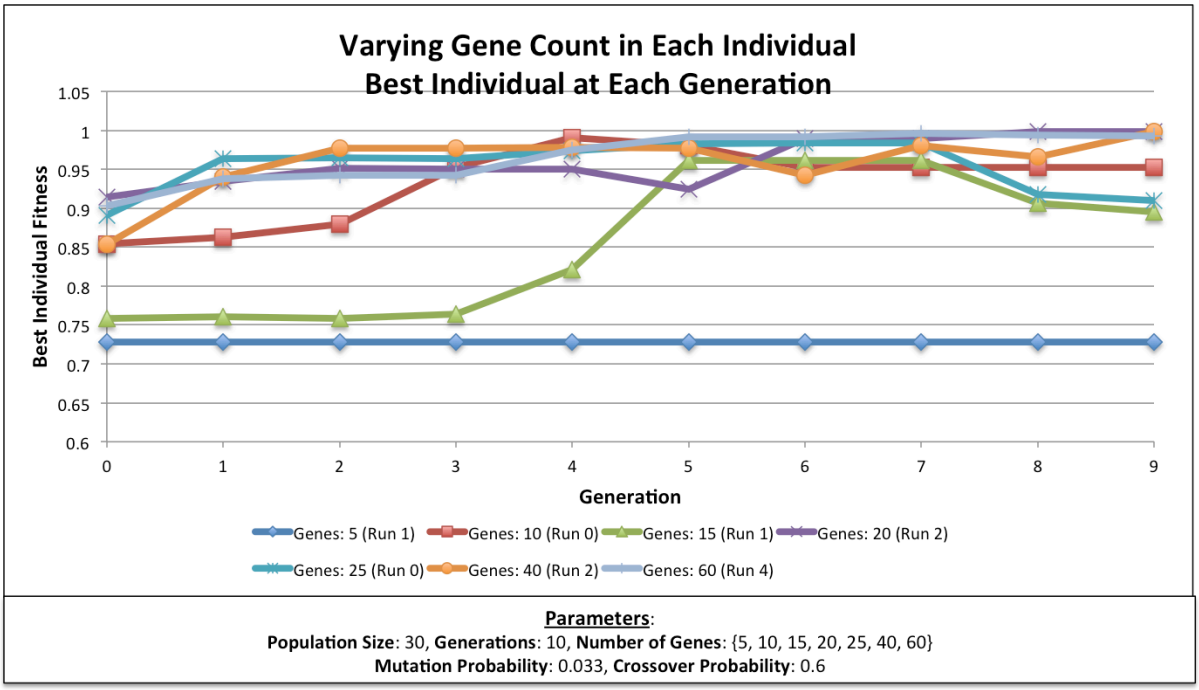
After varying the population size, the number of genes in each individual's genetic string was altered and studied. For this experiment, genetic string lengths of 5, 10, 15, 20, 25, 40, and 60 were used. The following graph shows the variation of the average population fitness across 10 generations with each of the gene counts described above.




The most prominent initial observation is that the simulation based on a genetic string length of 5 seemed to fluctuate most sporadically over time, and it fared the worst with respect to the average fitness of the population by the final generation. This could be because the genetic string was so short that it was too susceptible to mutations and crossovers that altered one the most significant bits. With fewer bits in each string, each alteration would have a larger impact on the resulting fitness.

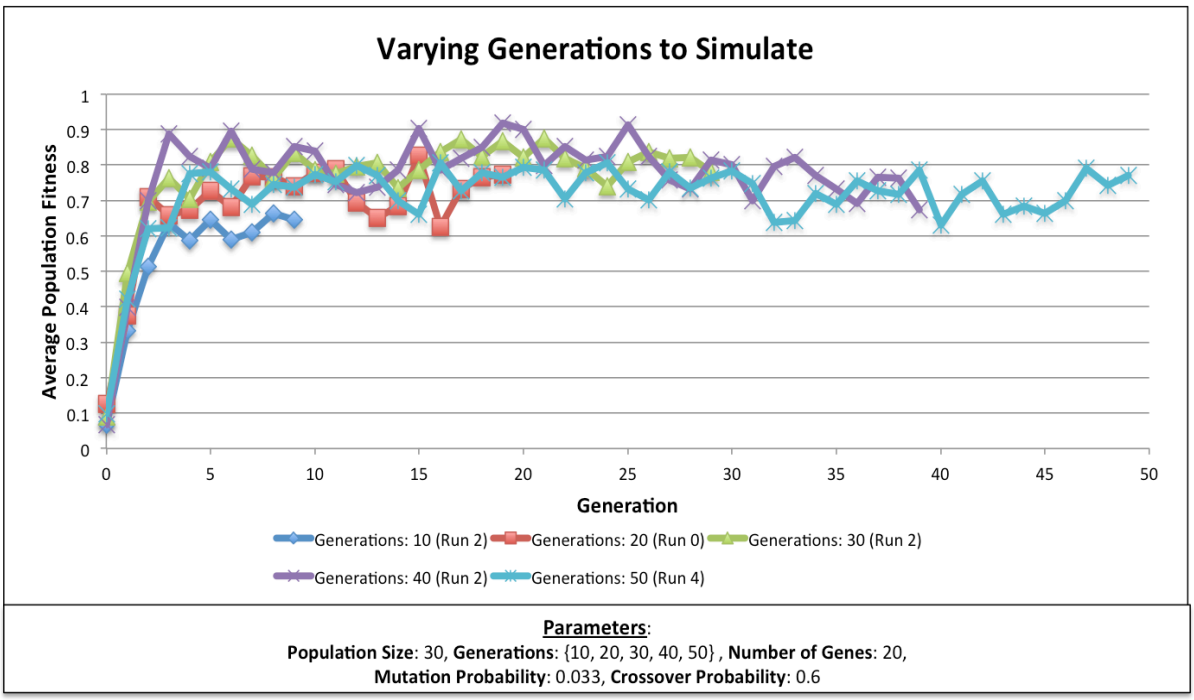
Oddly enough, the run with a gene count of 15 took the longest to reach a peak average population fitness, but the run with a gene count of 20 was the fastest to

reach a local maximum (by the second generation). Both of these variations for the genetic string length parameter seemed to produce simulations in which the average fitness of the population increased fairly steadily across most of the generations considered. Most of the other parameter values seemed to cause larger fluctuations in the average population fitness between generations.




## Number of Generations

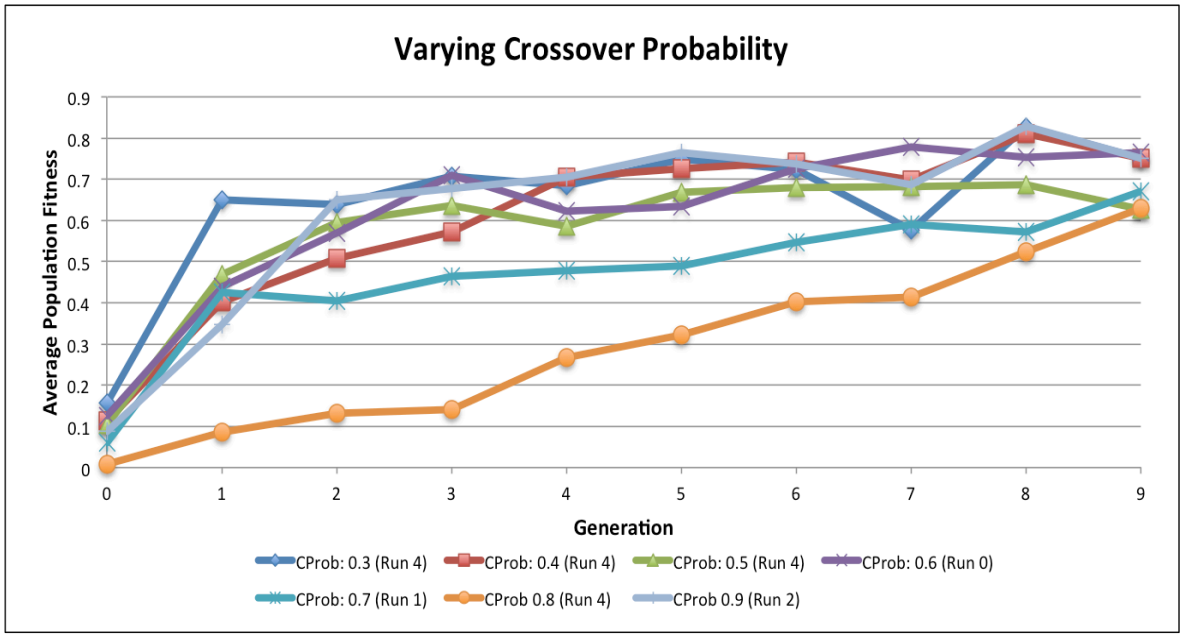
When determining the affect of varying the number of generations to simulate, it was more difficult to display those results in a meaningful way. Most of the other comparisons made and conclusions drawn were with respect to differences in fitness values at particular generations. However, when varying the number of generations, those comparisons do not line up correctly across all of the simulation variants. For example, the 10-generation simulation displays 10 fewer data points than the 20-generation simulation. Nevertheless, the chart below shows how the genetic simulation can change when it runs for different total time steps (or generations).




From the above graph, it seems that, regardless of the number of generations, the average fitness values for the populations converge to particular critical points at which those values never improve further. In general, this seems to occur at around 10-15 generations (at least, that seems to be the trend for these parameters). Instead, the average population fitness values seem to sporadically jump up and back down at subsequent time steps but they almost never improve beyond the maximal values obtained in the first 15 generations.

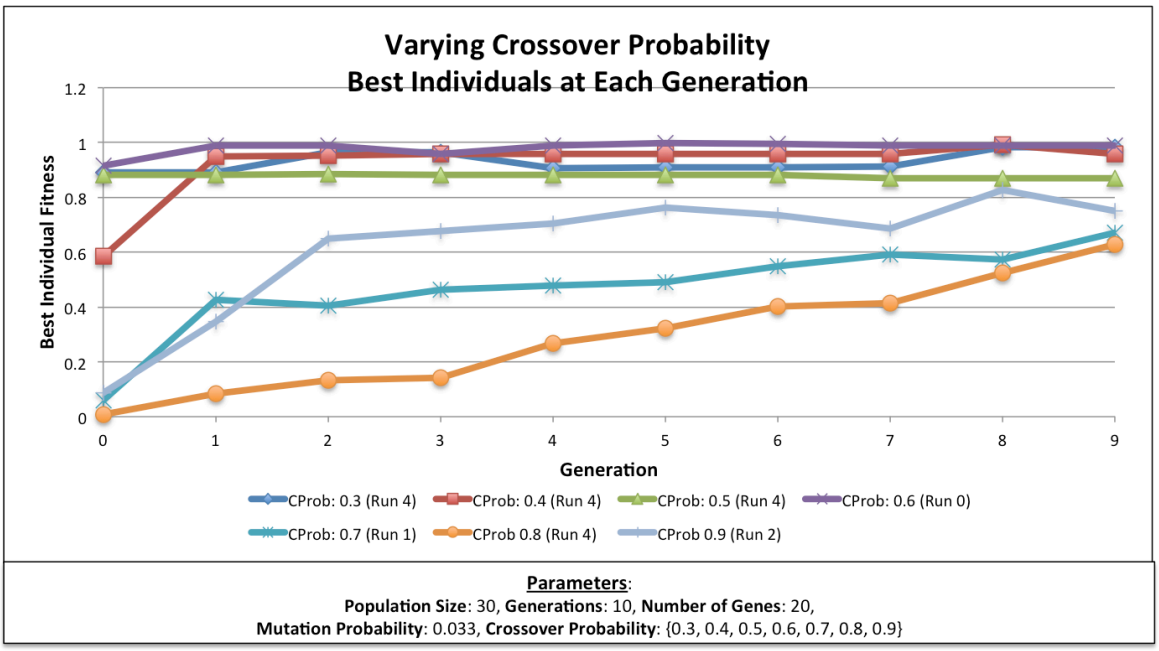
**Crossover Probability**

Because the crossover probability pertains to the likelihood that genes will be mixed between two individuals in the population, it seems reasonable to assume that altering this parameter could have drastic effects on the overall efficacy of the reproductive phase of development in these systems. The following figure shows how the crossover probability affected the average fitness of the population in each generation for the selected runs.




Unexpectedly, the different runs pictured above seem to display much less interesting behavior than what was anticipated. In fact, most of the variations in the crossover probability parameter seem to display relatively similar trends, despite the fact that the range of values used for the parameter was quite large (0.3-0.9).

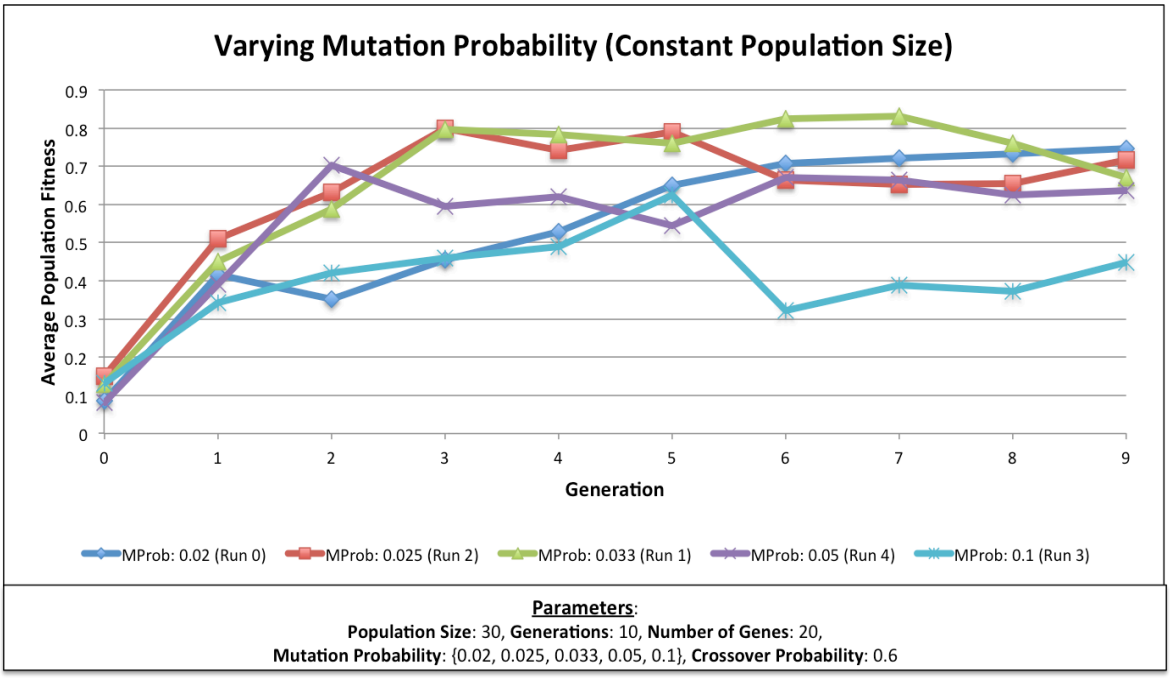
However, the chosen run from the simulation that used a crossover probability of 0.8 proved to be an exception. The average population fitness within this run increased much more slowly across the 10 generations that were studied than the fitness did for the other simulations. Interestingly, that run was the only one in this set where the average population fitness only increased between generations. The value always increased, and never dipped back down. Additionally, even though it took longer to reach values comparable to the other runs, the simulation with a crossover probability of 0.8 still reached a final average population fitness value that was nearly identical to a couple of the other runs.




## Mutation Probability

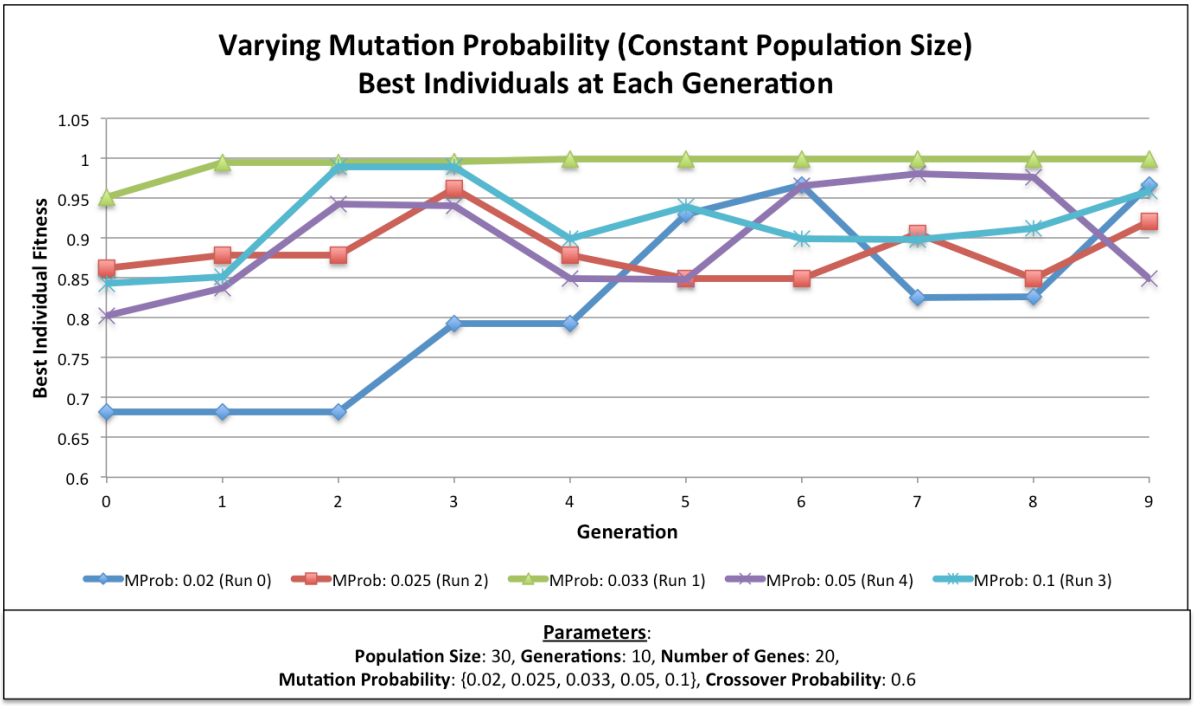
Like the crossover probability, the mutation probability can also play an important role in how the system changes between generations (during the reproductive phase). The project description said to keep the mutation probability approximately inversely proportional to the population size for the best results. However, the previous experiments were based on keeping all other parameters invariant when one was being manipulated. As such, two experiments were conducted with respect to the mutation probability. Initially, the population size was kept constant across all simulations (at 30 individuals), such that the mutation probability was the only parameter that changed between simulations (in accordance with the procedure used for the other experiments). The second experiment, on the other hand, was constructed such that the varied mutation probability values matched those used in the first mutation probability experiment, but the population sizes were adjusted to maintain the inverse proportionality between the two.

The average fitness of the population for the first experiment is shown below:

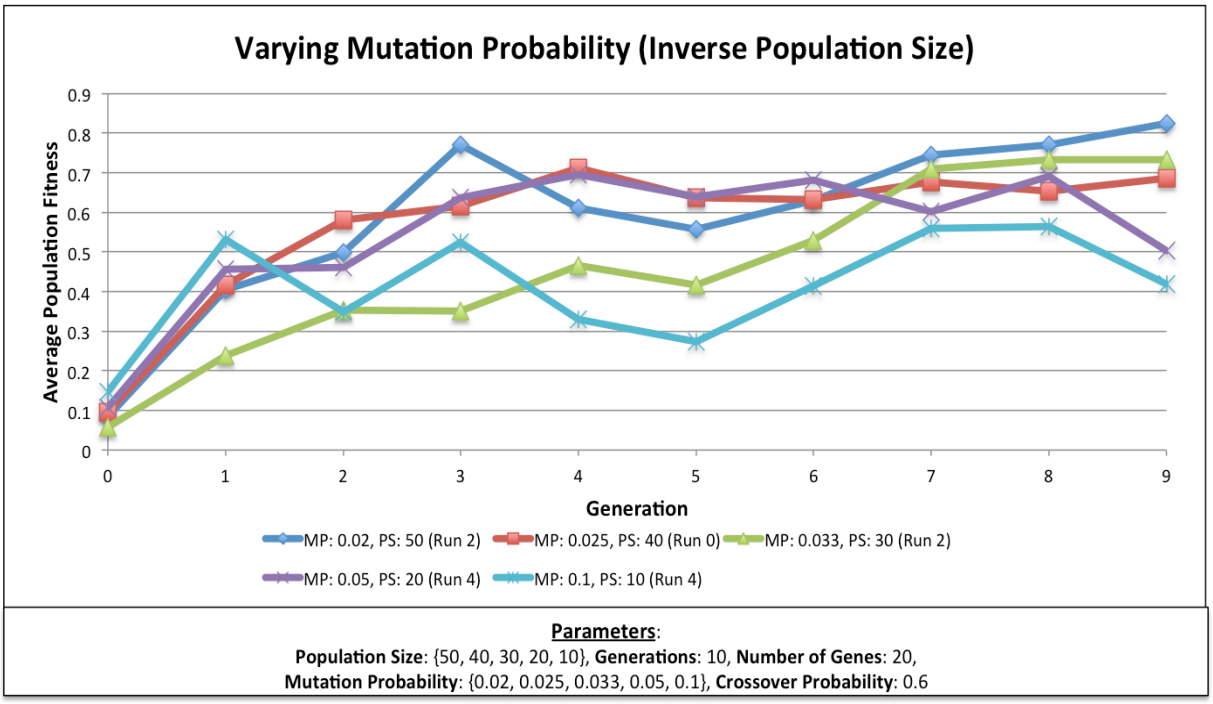



As expected, the mutation probability value of 0.033 seemed to produce the best results. Because the constant population size was set to 30 individuals, it makes sense that the mutation probability that is inversely proportional to the population size would fare the best when compared to other mutation probabilities acting on the same population size of 30.

More generally, the mutation probability seems to be almost directly related to how stable the average fitness of the population is between generations. For example, the highest mutation probability examined was set to 0.1, which means that about 10% of the bits in any individual's genetic string would mutate (and flip) with each new generation. Depending on the location of the bit being mutated (most significant or least significant), a mutation could drastically affect the fitness of a given individual either positively or negatively, leading to unreliable trends across all 10 generations. On the other end of the range, 0.02 was the lowest mutation probability value used in this experiment. While a mutation probability of 0.1 led to larger, more sporadic jumps in fitness between generations, a mutation probability of 0.02 had the opposite effect. In the latter case, a given bit in an individual's genetic string would mutate only 2% of the time, rather than 10%. As can be seen in the chart above, the average population fitness of the simulation with a mutation probability of 0.02 increased much more steadily and made no large positive or negative jumps between generations.

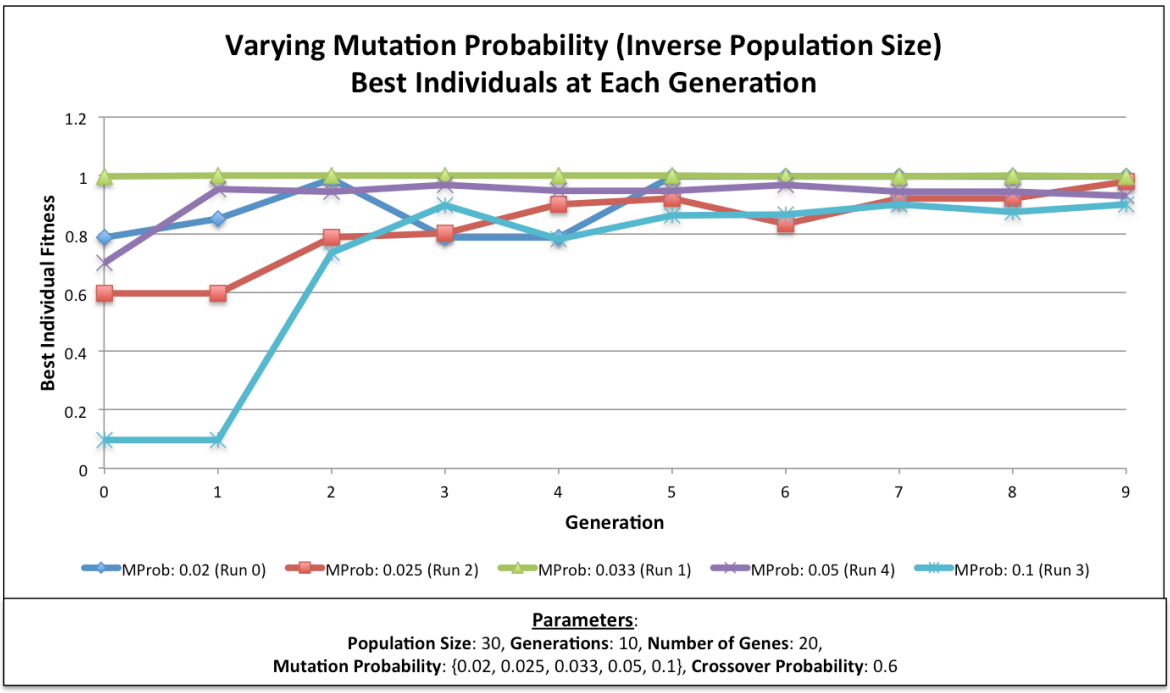



In the second experiment, the population size was varied to reflect an inverse proportion between that size and the mutation probability. The resulting average population fitness values across each generation can be seen in the chart below:




All of the runs shown above display extremely similar behavior to one another across the entire time span (10 generations) analyzed. These trends reinforce the

fact that the best mutation results seem to arise when the mutation probability is defined to be inversely proportional to the population size. This ratio allows for an acceptable amount of mutation to occur without mutating so many bits that the genetic strings are drastically changed from generation to generation. By inversely relating the mutation probability to the population size, any extremely damaging changes to one individual's fitness can be more easily balanced out by fitness gains in another individual.




## Conclusions

Overall, the parameters used to construct these systems seem to be fairly deeply intertwined with one another. By changing a particular parameter, another would likely need to be altered slightly as well to produce a better system. By generally changing only a single value within each experiment, it was difficult to observe and understand the more complicated parameter relationships. Additionally, most of the experiments conducted seemed to have fairly similar resulting values with respect to the average fitness of the final generation. As such, the effects of each parameter seemed to manifest themselves primarily in the transition between generations and the rate at which local maxima were found in the average fitness values of earlier generations, rather than the population fitness values that were observed in the final generations. Nevertheless, a few general conclusions can be drawn about the effects of particular parameters to the efficacy of the system as a whole.

As the size of the population within the system changes, many different variables are also affected. In general, larger population sizes would likely lead to greater

average fitness values over longer periods of time because the each population would contain a more diverse genetic pool from which subsequent offspring could be produced. With more fit individuals being more likely to reproduce and pass on desirable traits to future generations, the likelihood of more extremely fit individuals persisting through longer periods of time also increases. Additionally, when there are more individuals in the community, negative changes to a single individual have less of an influence on the population as a whole because the collective fitness is still fairly high.

When the number of genes in each individual's genetic string becomes too large or too small, the population is negatively affected. With too few genes, mutations will affect individuals much more drastically, which can lead to large jumps (either positively or negatively) in fitness levels. Shorter genetic strings, therefore, can increase the unreliability that emerges over time. Longer genetic strings, on the other hand, suffer from different issues. When there are a lot more bits to consider for each individual, it may take a much longer time for individuals to reach higher fitness values. The crossover probability and mutation probability need to be adjusted accordingly to handle these issues.

The number of generations does not have a very drastic influence on the resulting behavior of a given system. After approximately 10-15 generations, most of these genetic systems will have already converged to approximately optimal fitness values. At a certain point, the general fitness will tend to simply oscillate around some critical point, but new improvements are not likely to occur, even when many more generations are added. In fact, more generations will increase the likelihood of impactful, negative mutations to occur, which could make the system worse in the end.

The crossover probability and the mutation probability are perhaps the most important variables to consider for realistic applications of genetic algorithms. They are what truly determine how the system behaves and changes over time. The crossover probability is crucial for finding optimal solutions within a given genetic system. Crossover is how two good individuals can create two offspring that are even more fit. However, without mutation, crossover alone could result in premature convergence in which a system converges to a seemingly optimal solution, even though a better one could have actually been obtained. Mutation introduces noise to help overcome convergence to these local maxima. Ultimately, crossover may help find an optimal solution, but mutation helps inspire innovation in a system such that new genetic combinations are introduced.

With respect to the best individuals in each generation, the number of correct bits in each of the most fit individuals was not a very informative measure. The location of the correct bits mattered much more to the overall fitness than the actual number of correct bits. Because each genetic string was interpreted as a binary number, the bits that were located on the most significant side had a much larger influence on the fitness level than the bits on the least significant side.