Gaddiel Morales
COSC 527 Biology-Inspired Computing
February 24, 2023

Evolutionary Algorithms

Introduction

Evolution is a common, natural technique which can be implemented in computer systems in order to solve solutions. This lab demonstrates the effectiveness of Evolutionary Algorithms (EAs). The experiment makes use of the LEAP-EC python library in order to simulate 30 generations of evolution. Combinations of four values for four parameters will be tested:

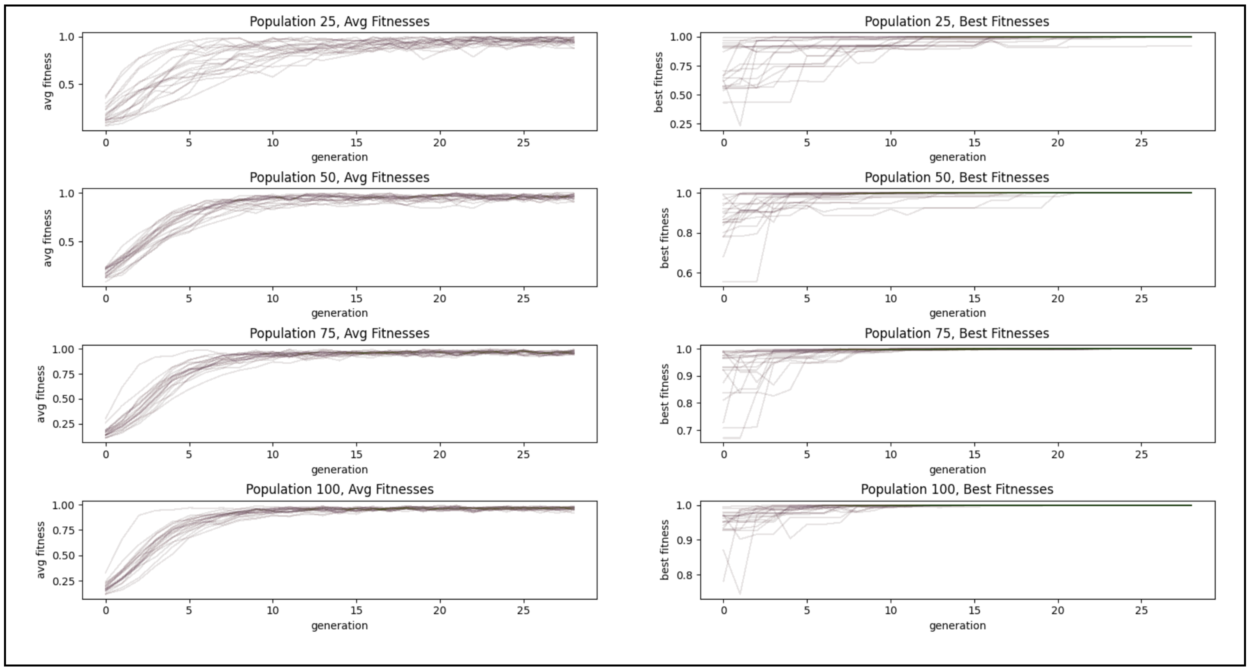
- Population sizes of 25, 50, 75, and 100.
- Mutation probabilities of 0, 0.01, 0.3, and 0.5.
- Uniform Crossover probability of 0, 0.1, 0.3, and 0.5.
- Tournament Size of 2, 3, 4, 5.

The goal of the experiment is to find a genome, represented by a binary string of 40 values, that is equal to a string of all 1s. The EA will use the fitness function below in order to assess the genome during the selection stage.

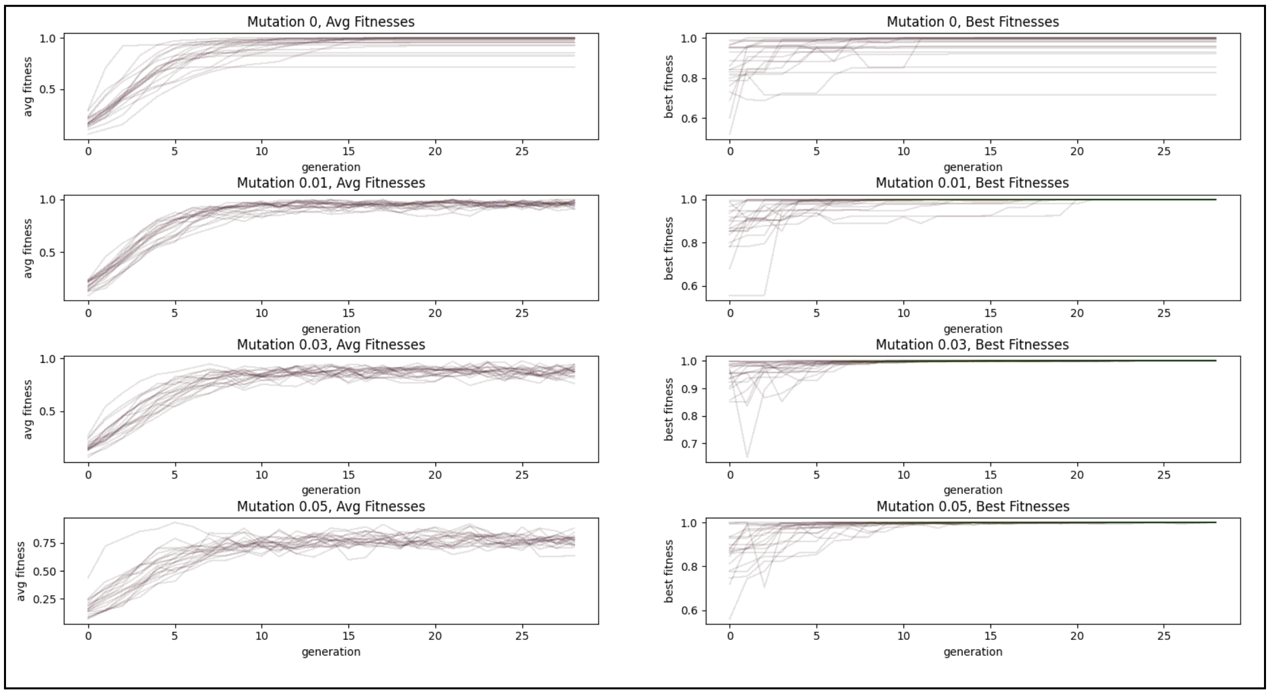
$$F(S) = \left(\frac{x}{2^l - 1}\right)^{10}$$

Experimental Results

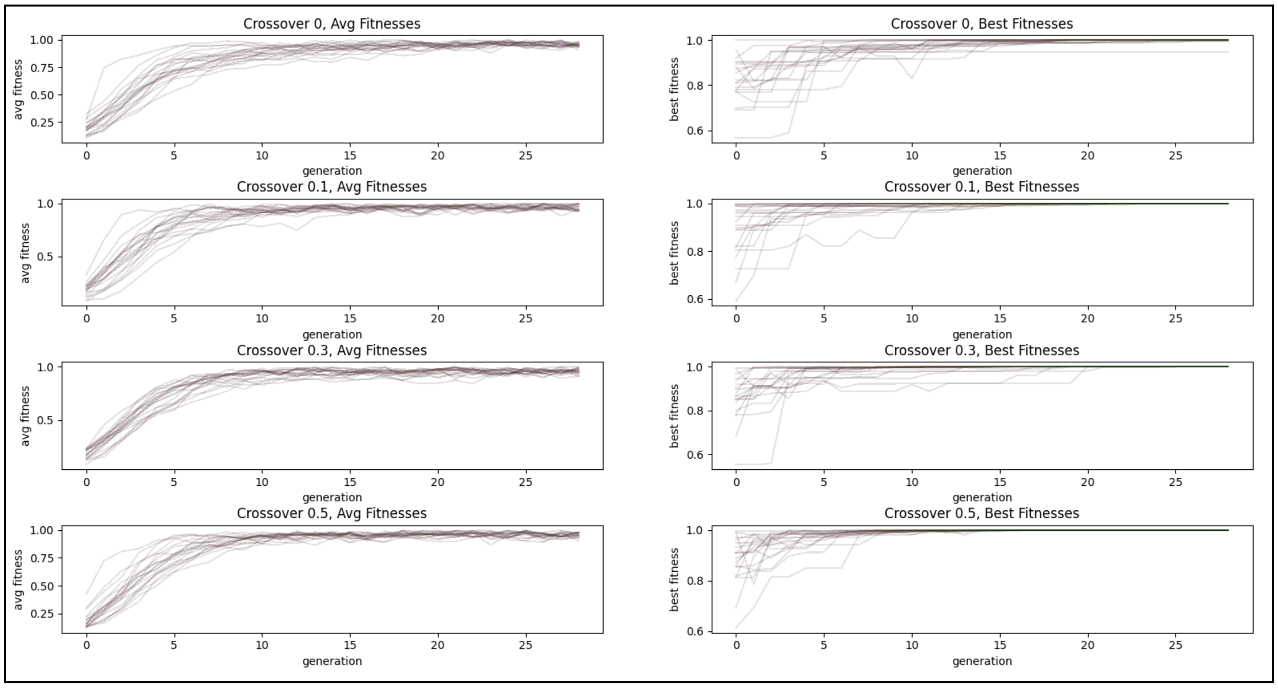
A copy of the full result CSV can be found attached to the report. To illustrate the relation between the different parameters, a set of default values were selected for the graphs below. The population default is 50, the mutation chance default is 0.01, the crossover chance default is 0.01, and the tournament size default is 2. Each graph compares the fitness value for each combination of 3 parameters at the default value and one parameter in a variable state.



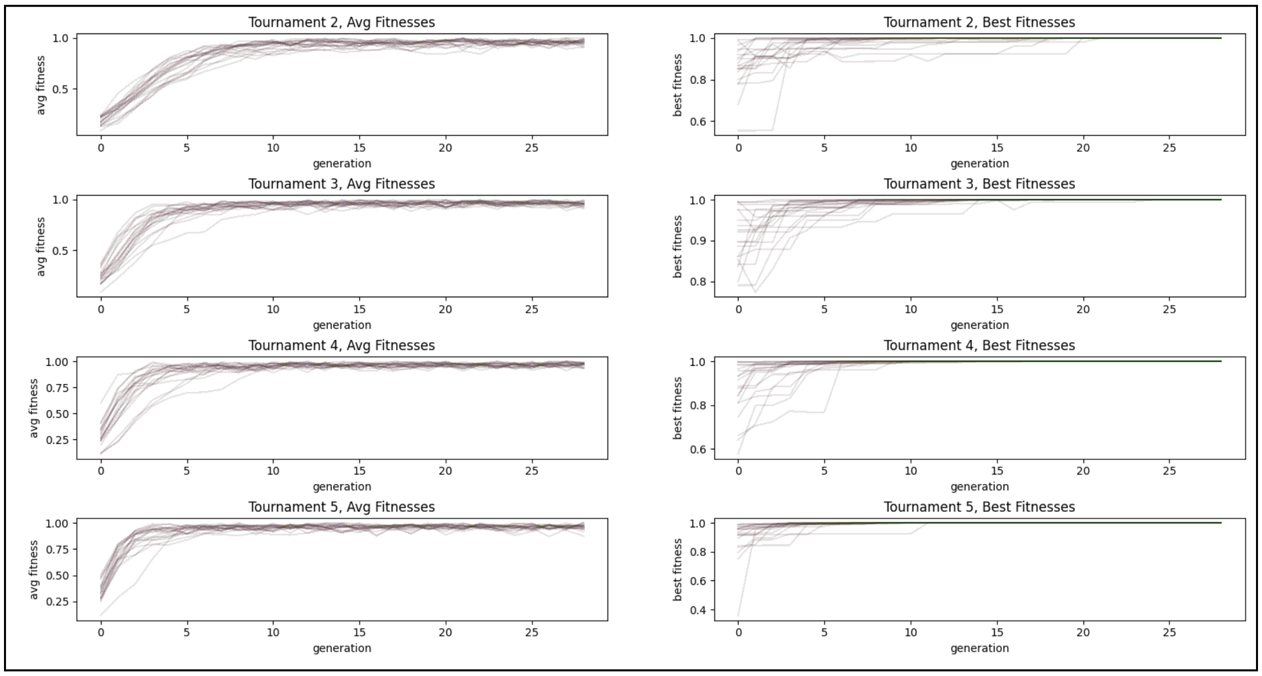
Interestingly, the low population graph seems to vary the most. This may be due to the fact that there is a reduced chance for fit individuals to appear. Additionally, high population sizes are less likely to be affected by outliers, with average values staying relatively stable. Meanwhile a high population value was able to quickly find an individual with a good best fitness since there are more opportunities to create a fit individual



High mutation probabilities seem to be most likely to create a varied population. It also appears that the highest mutation value causes the average fitness to approach the ideal fitness much slower. This is likely due to the increased chance for mutations to result in worse fitness. However, a rare, lucky mutation can quickly advance the evolution process, which likely explains the outlier. In regards to finding the individual with best fitness, low mutation values seem to be the worst case, with some generations never being able to reach the close to the ideal fitness.

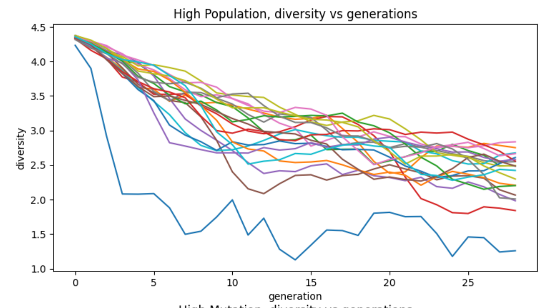


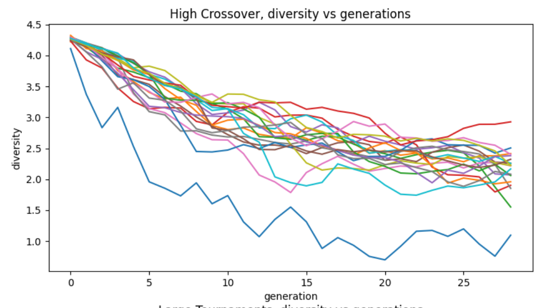
For uniform crossover, increasing the probability did not have much effect on the system. The graphs appear to have a similar slope regardless of the amount of crossover. Since crossover is recycling the material of previous parents, it expectedly reduced effectiveness compared to mutation.

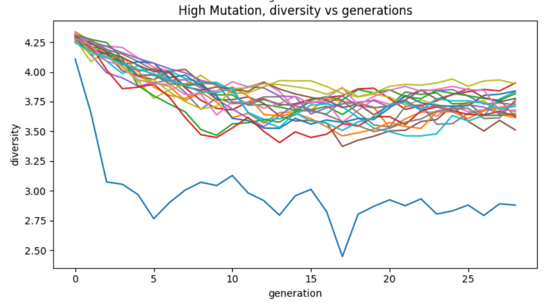


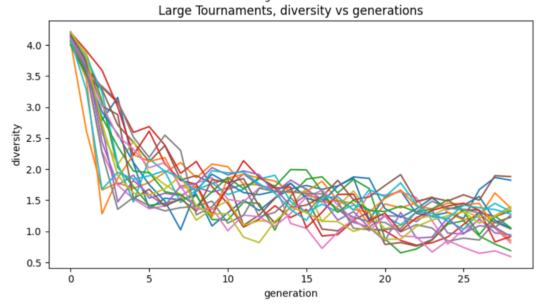
Larger tournament sizes clearly impacted the speed that the system approaches a value. The best fitness value was most positively correlated out of all other parameters.

An important factor to consider with EAs is the diversity of the population. Less diversity causes a population to slow its ability to explore new solutions. The following graphs showcase diversity over generations with three of the evolution parameters as the default and one parameter as the high value in its combination set.



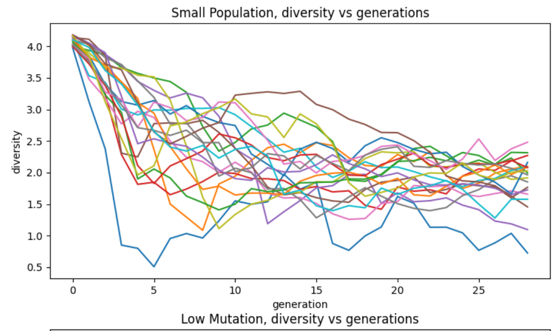


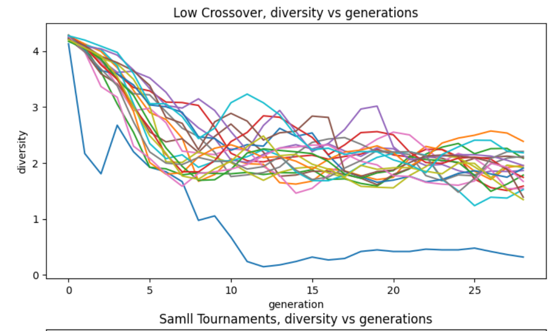


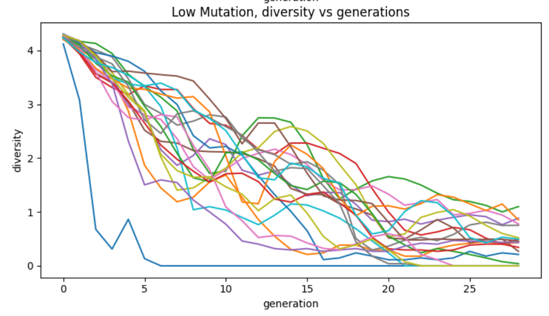


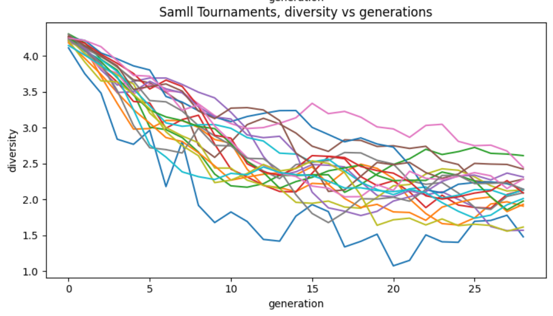
Population size seems to be the least correlated to diversity out of the four parameters. Meanwhile, a high mutation rate is most likely to maintain diversity. This makes sense as mutations provide completely new effects to the environment. In this experiment, fit parents will have a lot of their genome in common, which means crossover becomes less effective for older generations. A large tournament size was the most detrimental to the population diversity, since a higher tournament size makes the system intentionally pick more fit individuals for the next generation.

Low parameter values and their effects on diversity may be seen below. Like the previous graphs, the values besides the parameter in figure titles are kept at default values. The probability parameters for mutation and crossover are 0, while the tournament size is reduced to 2 and the population size is reduced to 25.



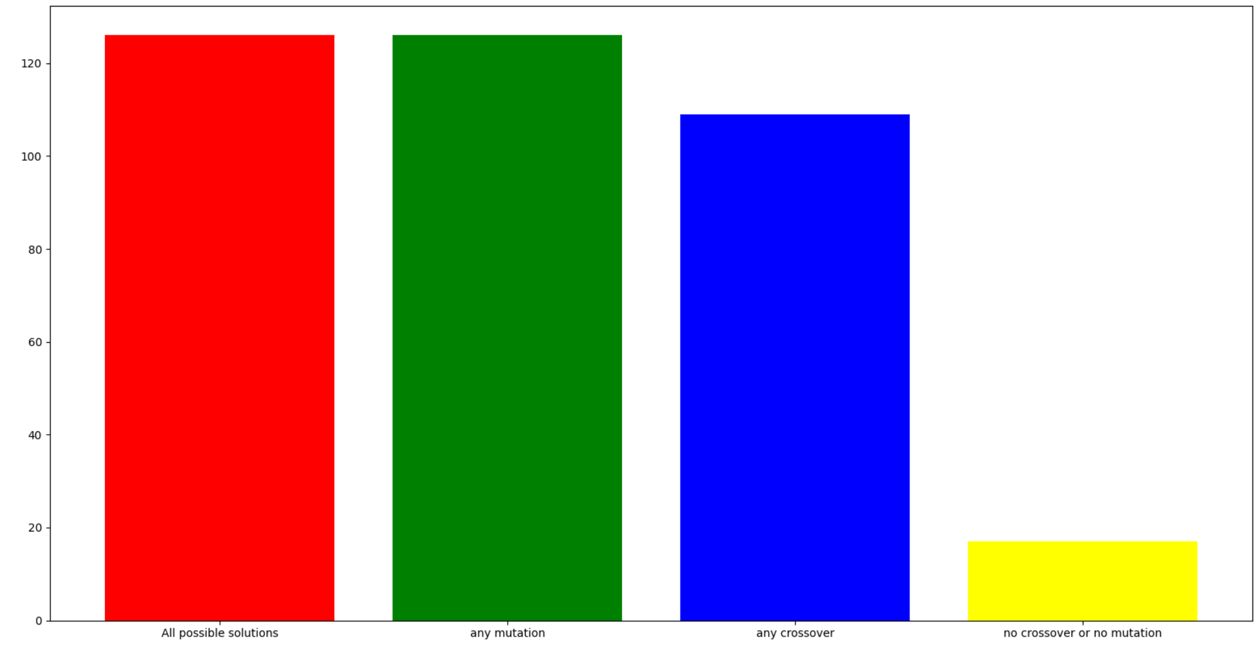






Without crossover, most iterations seemed to maintain a diversity value of 2. In general for all parameters, the lower values made the diversity more varied among iterations. Additionally, with no mutations the mutation graph quickly becomes the least diverse dataset here, which is the opposite of when the mutation values were high.

When trying to figure out the relationship between crossover and mutation, The following graph can provide better understanding. The first bar shows the number of all the generated solution genomes.



The next bars show the number of solutions which had a mutation probability greater than 0, the number of solutions that had a crossover probability greater than 0, and the number of solutions that had either 0 for the crossover probability or 0 for the mutation probability. Therefore, it is possible to get a solution by using only crossover or only mutation.

Conclusion

Overall it seems mutations were the most influential on system diversity. The total simulation iterations were 5120. 126 generations were able to find solutions. Selection pressure, as measured by tournament size, can significantly impact performance by causing the simulation to converge faster, but it also reduces diversity in the population. Both crossover and mutation are not necessary to find solutions, as solutions can be found using either one alone. However, using both can lead to faster convergence and more diverse populations. The parameter with arguably the biggest impact on performance in terms of fitness achieved is mutation probability. High mutation probabilities create a more diverse population and allow for a random chance of getting out of a local minima, but they can also slow down the evolution process. Low mutation probabilities can lead to a population getting stuck in a suboptimal solution. As for population, it certainly affects performance, as higher population sizes can lead to faster generational convergence because of the higher chance of finding a fit individual, but it can increase computational time.