CS 420 Lab 2: Effects of Parameters on Genetic Algorithms and their performance

Introduction:

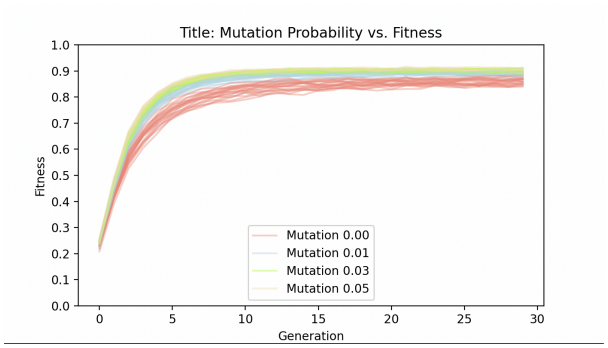
First things first, what is Genetic Algorithms? Genetic Algorithms are basically optimization techniques that are inspired by natural selection and genetics. The goal of this project is to explore the impact of various kinds of parameters on genetic algorithms performance using the LEAP in Python.The parameters include the population size, mutation probability, crossover probability, and tournament size.

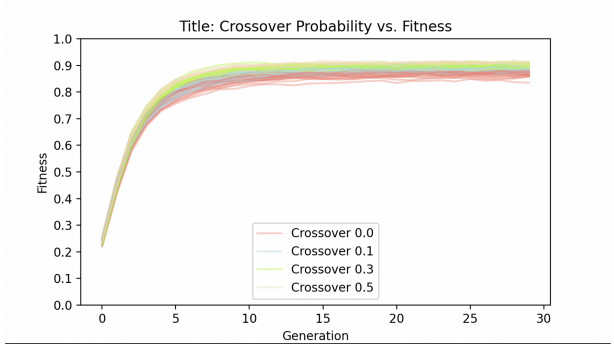
For this project, I conducted multiple experiments with varying each kind of parameter within defined ranges and running for a total of 30 generations. The four parameters were Population Size(N): [25,50,75,100], Mutation Probability(pm): [0, 0.01, 0.03, 0.5], Crossover Probability(pc): [0, 0.1, 0.3, 0.5], and Tournament Size: [2, 3, 4, 5]. The calculation for fitness function was F(S) = (x/(2^l)-1)^10, where S represented the genome, x represented the integer from interpreting S as an unsigned binary number, and l was the length of the genome.

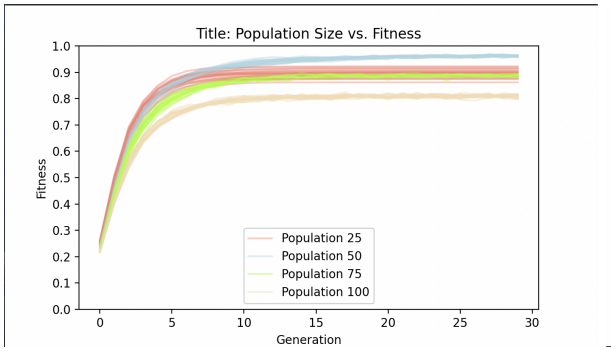
Calculation:

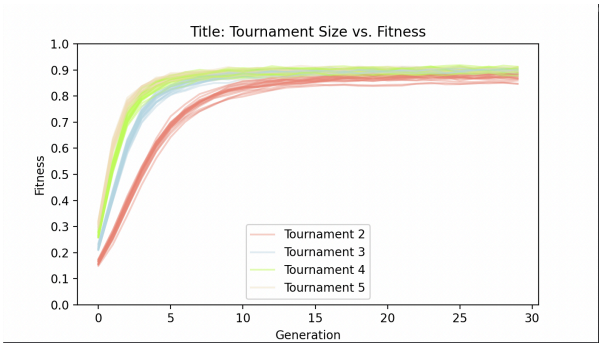
For each parameter combination, I calculated the average fitness, the best fitness, the best genome, presence of solution, and the number of solutions found per generation. And, the diversity metric was calculated based on the distances between individuals in the population.

Graphs:

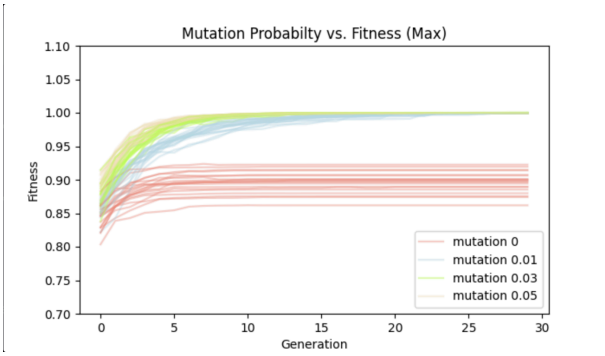


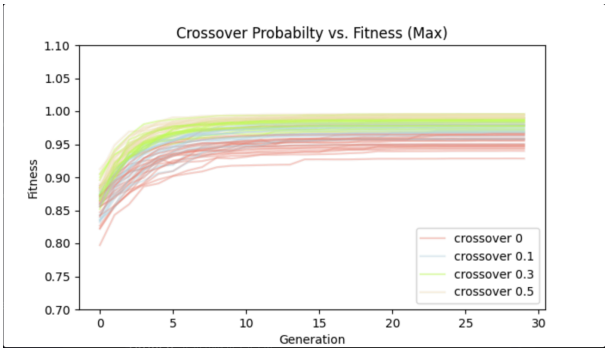


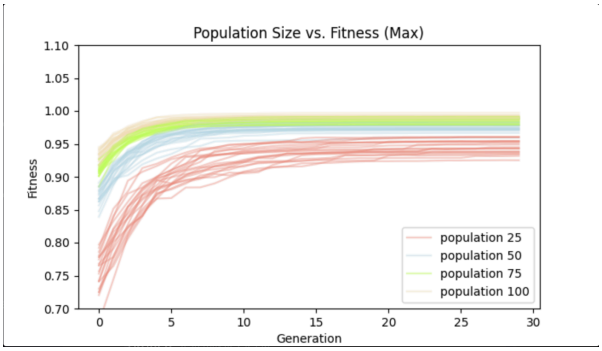


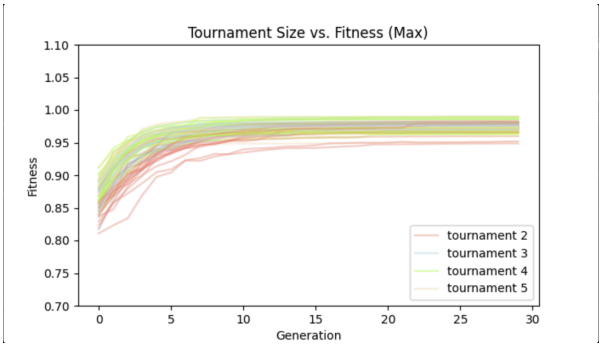


I generated a total of 8 graphs, 4 for the average fitness and 4 for the best/max fitness cases. The graphs illustrate the impact of each parameter on fitness. In the graphs, the x-axis represents generations up to 30, and the y-axis represents fitness.









Analyzing the graphs:

- Population Size vs. Fitness:
  Increasing population size generally results in a better fitness level, and we can see that in the graphs above. Larger populations explore a broader solution space, which helps to enhance the likelihood of finding optimal solutions. On the other hand, smaller populations tend to almost converge faster but they also risk premature convergence to suboptimal solutions. For example, Population 25 converged much quicker but achieved lower fitness compared to larger populations.
- Mutation Probability vs. Fitness:
  Mutation plays quite an important role in introducing diversity into the population. Low mutation probabilities resu;t in slower exploration of the solution space, while high probabilities may disrupt convergence. I also observed that a moderate mutation rate of 0.03 achieved the best balance between exploration and exploitation, leading to higher overall fitness.
- Crossover Probability vs. Fitness:
  Crossover makes the exchange of genetic information between individuals easy, which also promotes the convergence towards better solutions. The best crossover probability appears to be around 0.3, where the algorithm strikes a balance between exploration and exploitation. Lower values are causing a bit of hindrance for genetic diversity, while higher values may lead to premature convergence.
- Tournament Size vs. Fitness:
  Tournament size affects the selection pressure, with larger tournaments favoring to higher-sitness individuals. Smaller tournaments like tournament 2 result in faster convergence but may also limit the algorithm in local optima. Larger tournaments provide a much better exploration but slower convergence rate.

Discussion:

Let's first talk about the impact of parameters on the performance. Population size and mutation probability showed quite significant impacts on performance. I saw that the larger populations tended to achieve higher fitness levels. On the other hand the moderate mutation rates enhanced exploration. Crossover probability and tournament size had more nuanced effects with optimal values depending on other parameters.

Now, let's talk about the influence of selection pressure on performance. We know that higher tournament sizes correspond to increased selection pressure which results in faster convergence but can also potentially have premature convergence to local optima. Optimal selection pressure varied depending on the problem's complexity and diversity.

Both crossover and mutation played an important role in maintaining genetic diversity and facilitating exploration. While crossover facilitated the exchange of genetic information between individuals, mutation introduced novel genetic variations that are important for escaping local optima.

Now let's talk about parameter interactions. Population size and mutation probability exhibited some strong interactions, with larger populations requiring higher mutation rates to maintain diversity. Also, crossover probability has interactions with both mutation probability and population size which influenced the exploration and exploitation trade-offs.

Increasing population size generally led to improved performance as larger populations provided us with a better genetic pool for exploration and exploitation. However, beyond a certain threshold, increasing population size yielded diminishing returns due to increased computational complexity.

Conclusion:

After doing multiple experiments and analysis, I have gained some very valuable insights into the effects of various parameters on the performance of the genetic algorithm. Fine-tuning these parameters is quite important to strike a good balance between exploration and exploitation, which basically leads to optimal solutions. Understanding these dynamics is important for effectively applying genetic algorithms to solve any kind of optimization problems.