Ryan Filgas
AI – Fall 2022

Program 2 Writeup

State: For the state of the board all solutions have a queen in each row and column. Since we're doing a genetic algorithm necessarily we can only freeze one of those, but this cuts the state space in half. A state is represented by an array of length 8 where each element can be between 1 and 8 inclusive in any order. [1, 2, 3, 4, 5, 6, 7, 8].

Fitness: For the fitness measurement I wrote a function to count all potential collisions from each queen to every other queen. Checking parallel collisions it was as simple as counting duplicates. Counting diagonals required generating 2 arrays (1 ascending and 1 descending for each diagonal direction) and checking for matches between the states.

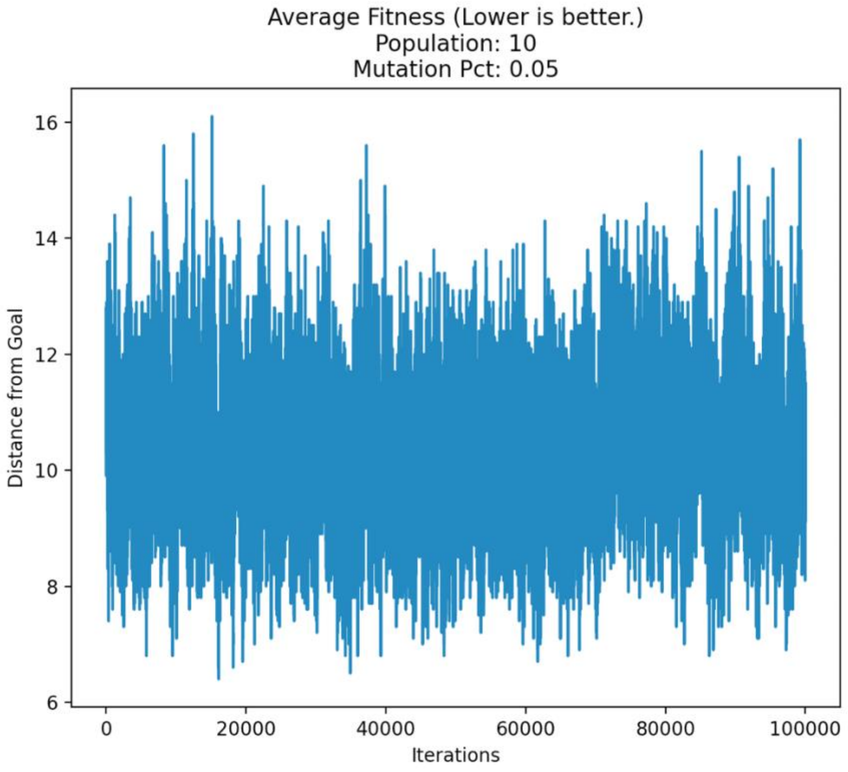
Selection: For the selection I chose from a normalized fitness distribution – where the probability of choosing a parent is equal to its fitness divided by the total fitness of the population

Mutation: The mutation probability was set randomly where a mutation changes the selected index to a random number between 1 and 8 inclusive and the crossover index was selected randomly as a number between 2 and 7 (because otherwise this isn't reproducing). The fitness is recalculated as well as the total and average fitness before adding the children back into the population.

Population Samples: A sample is taken from the best 2 candidates at each 10% increment of the population as seen in the results below. Duplicates are heavily reduced from the start to finish.

Summary of Observations:

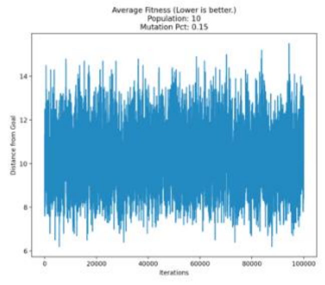
- Choosing a high rate of mutation is counter productive and doesn't perform well.
- Choosing only the top two members for reproduction has a negative effect on population health over time in comparison to using a normalized distribution.
- Increasing the population size results in less fluctuation in average population fitness, and more consistent fitness gains.
- Reducing the population size causes high amounts of fluctuation in average population fitness, however the population fitness improves much faster.
- When population size is very large I experienced diminishing returns after about 50% of 100,000 iterations. This is contrasted however when the population continues to grow – they take much longer to converge and do not result in better average population fitness.
- A population of 1000 with a mutation rate of .25 appears to have the best results when judging by fitness – however this test was only run once. Given more time and different random starts this may not hold true.
- There are diminishing returns to population size, iterations, and mutation probability.

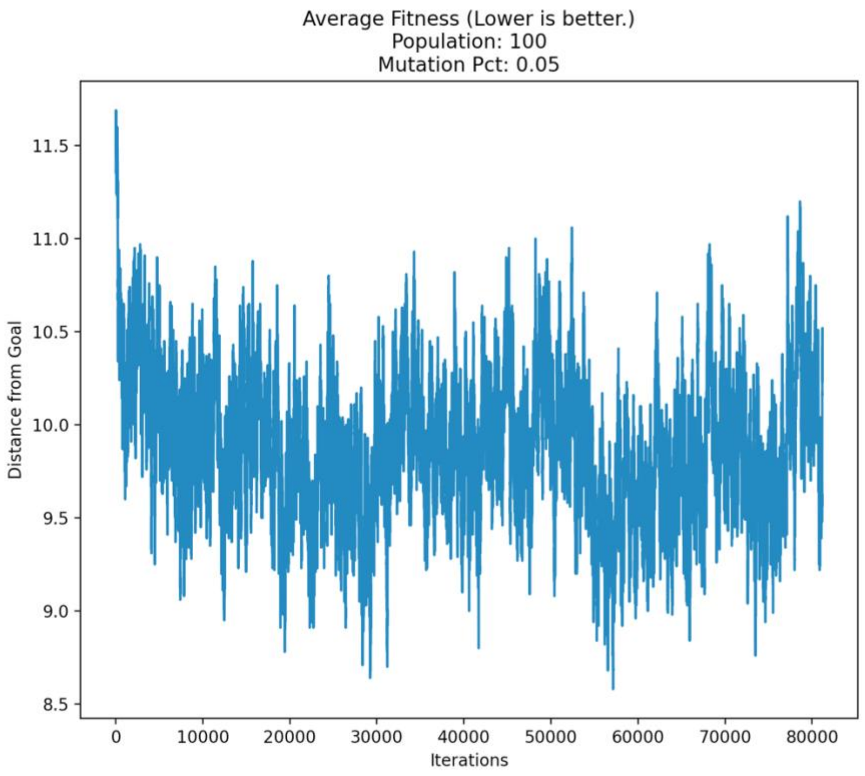


Starting Fitness: 12.7
Ending Fitness: 11.5
Fitness of best start: 9
Fitness of best end: 7
Final Round Was: 100000

[5, 4, 8, 1, 8, 4, 1, 2], [2, 5, 7, 1, 5, 1, 5, 3], [5, 1, 7, 1, 8, 8, 8, 2], [2, 1, 1, 4, 1, 1, 7, 1], [2, 6, 7, 1, 8, 6, 1, 3], [5, 8, 1, 7, 1, 6, 8, 7], [3, 1, 3, 1, 6, 4, 6, 1], [6, 1, 4, 1, 3, 6, 6, 1], [8, 6, 7, 4, 1, 1, 5, 8], [1, 6, 3, 1, 6, 2, 2, 7]

Observations: As seen in the graph over 100,000 iterations, a population size of 10 doesn't perform well with a genetic algorithm and a small mutation rate leading to heavy fluctuations in population fitness. Ass seen in the thumbnail size image below, increasing the mutation rate to .15 has similar results.

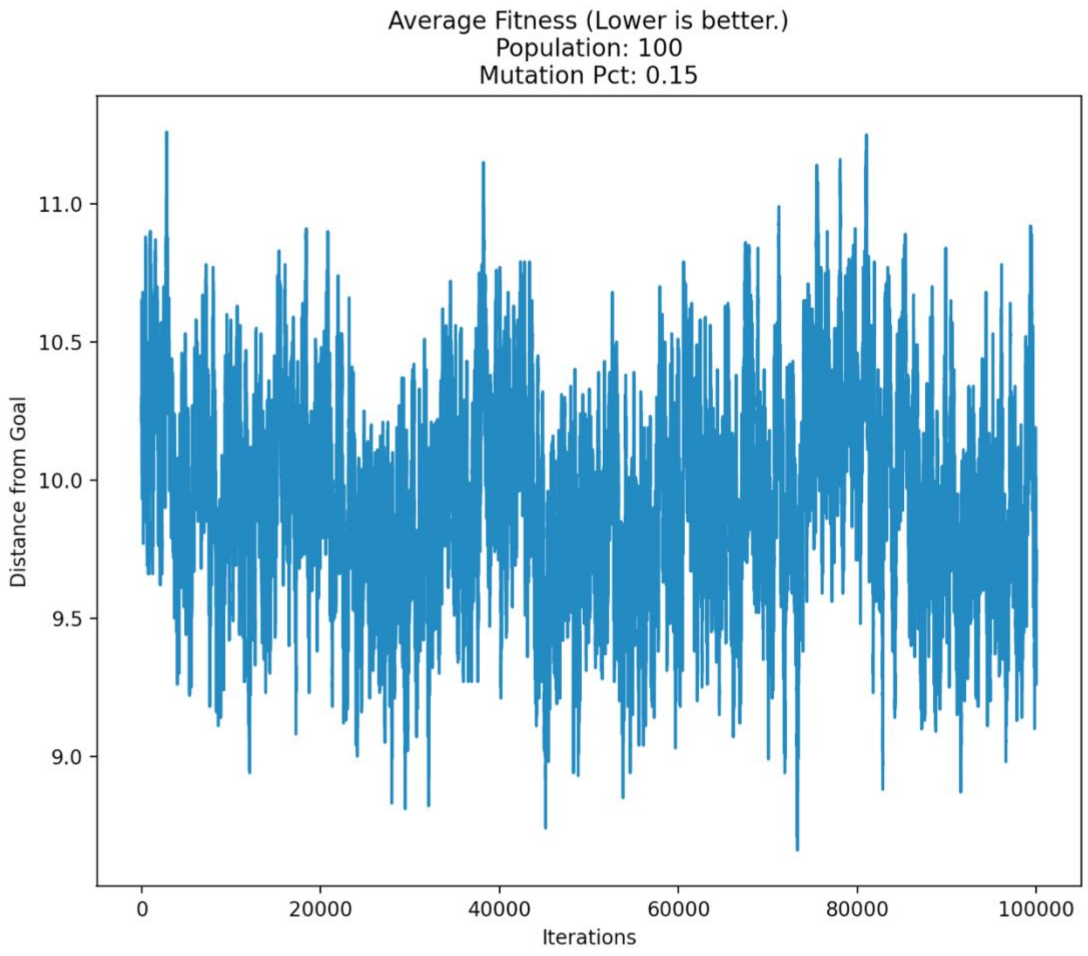




Starting Fitness: 11.55
Ending Fitness: 10.38
Fitness of best start: 5
Fitness of best end: 0
Final Round Was: 81200 – Found Solution Early

[6, 8, 2, 7, 4, 7, 1, 3], [7, 3, 8, 8, 5, 1, 4, 6], [7, 2, 5, 8, 4, 1, 3, 5], [2, 8, 1, 3, 7, 3, 6, 8], [2, 2, 8, 5, 7, 1, 1, 6], [8, 3, 8, 6, 2, 7, 5, 7], [4, 7, 1, 8, 1, 7, 6, 8], [1, 1, 1, 1, 6, 1, 3, 1], [7, 1, 6, 1, 7, 7, 7, 2], [3, 6, 4, 1, 8, 5, 7, 2]

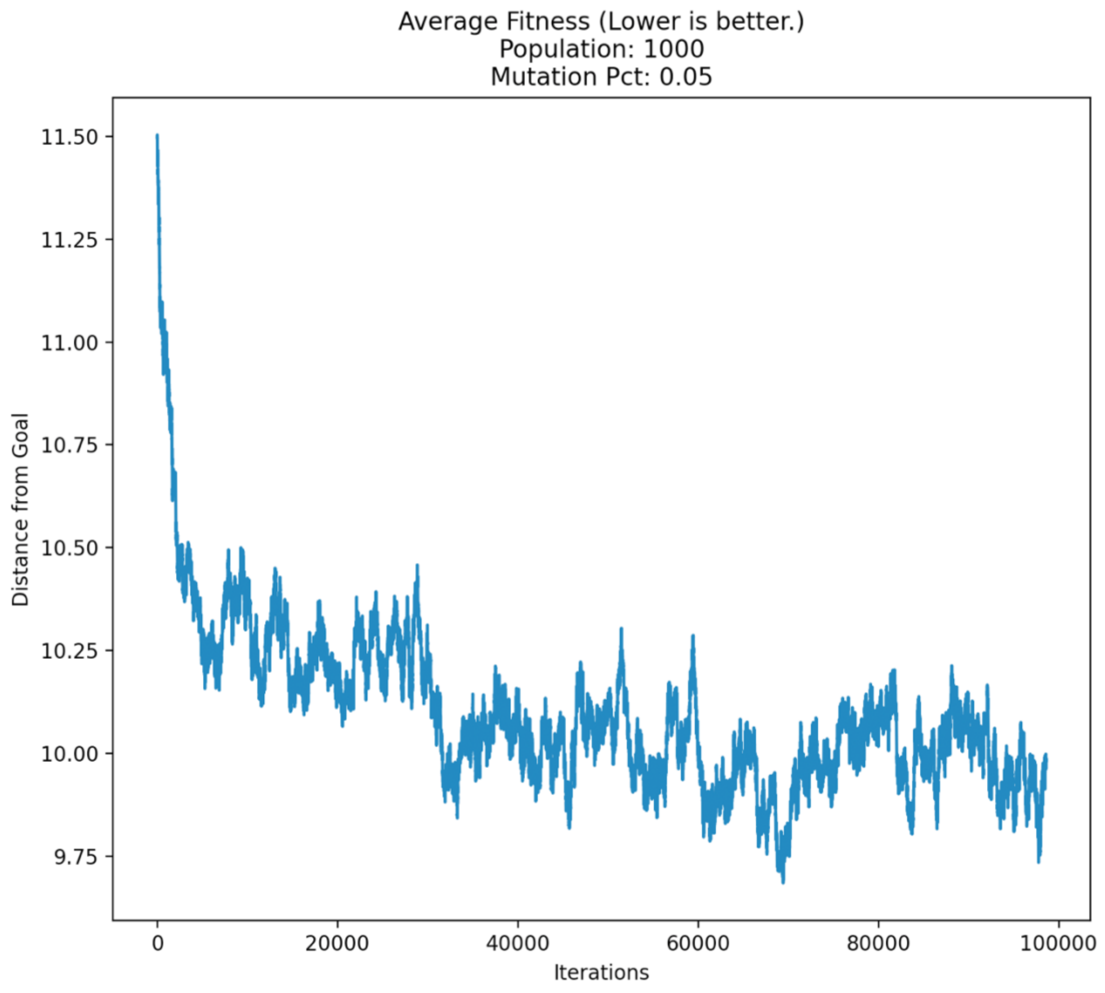
Observations: Increasing the population to just 100 vastly improves the range of fluctuation and provided a winning configuration. As before there are diminishing returns however the new minimum fluctuation bound is around 8.25 with a max at 11.3. This is much tighter than the 6-16 fitness fluctuations for a population of 10.



Starting Fitness: 10.69
Ending Fitness: 9.7
Fitness of best start: 3
Fitness of best end: 4
Final Round Was: 100000

[3, 1, 8, 4, 1, 1, 5, 1] [8, 5, 3, 1, 6, 1, 1, 7], [2, 2, 2, 7, 7, 4, 1, 1], [2, 6, 4, 1, 7, 5, 7, 2], [4, 1, 3, 8, 2, 7, 3, 7], [6, 4, 1, 8, 3, 7, 7, 1], [3, 7, 3, 1, 8, 5, 1, 2], [3, 7, 4, 1, 8, 1, 1, 6], [2, 2, 8, 3, 1, 8, 2, 2], [4, 7, 3, 6, 6, 1, 5, 6]

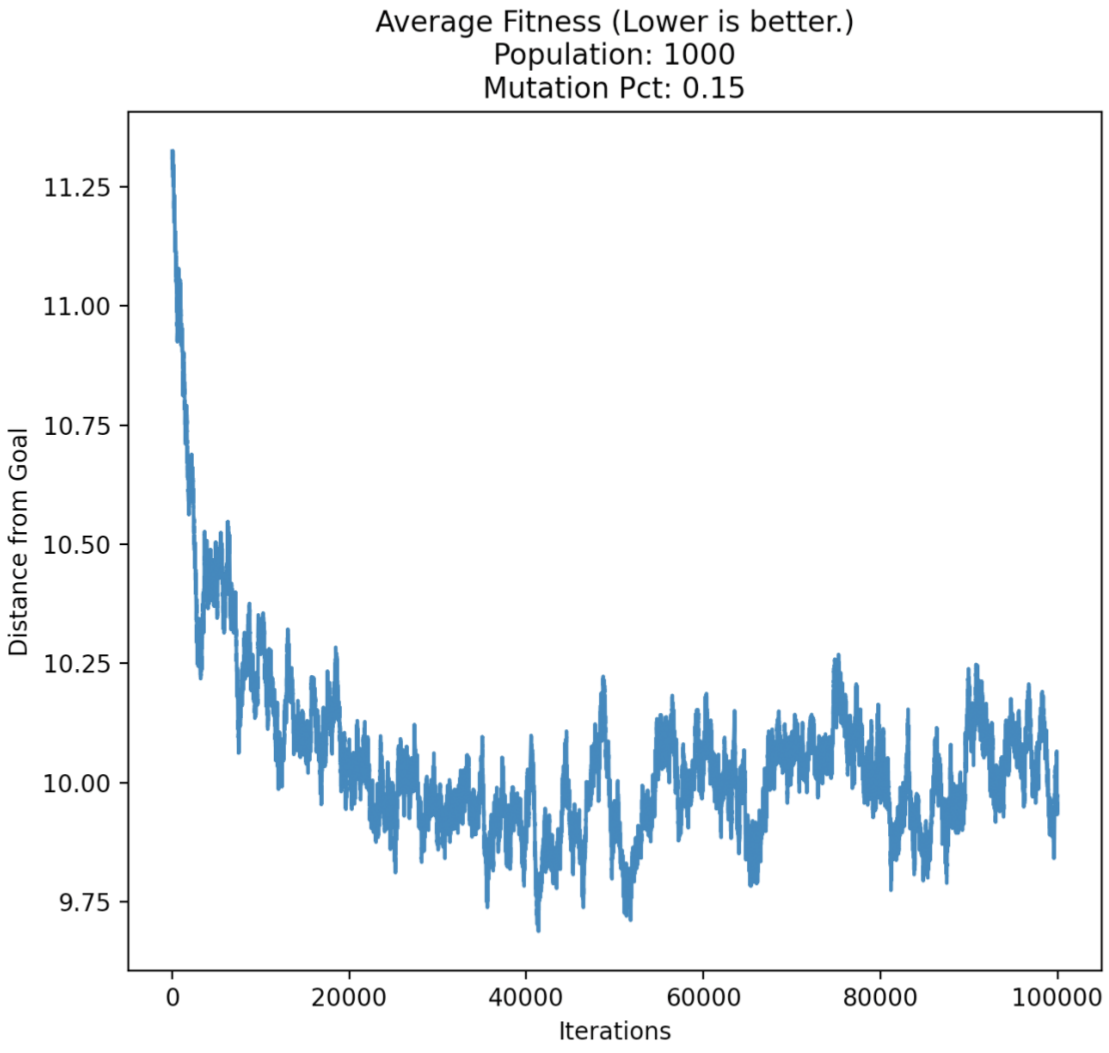
Keeping population equal the fluctuation range is ~8.75 to 11.4. This is just slightly worse than a mutation probability of .05, however the best single offspring actually got worse – this might have changed randomly given time.



Starting Fitness: 11.493
Ending Fitness: 9.979
Fitness of best start: 3
Fitness of best end: 0
Final Round Was: 98609 – Reached goal early

[4, 4, 8, 5, 3, 6, 3, 5], [8, 1, 4, 2, 5, 8, 1, 3], [7, 2, 7, 1, 4, 8, 5, 3], [4, 1, 5, 2, 6, 1, 3, 8], [8, 1, 7, 4, 2, 8, 8, 3], [2, 8, 5, 7, 1, 3, 6, 6], [2, 8, 5, 7, 1, 3, 6, 6], [7, 1, 8, 8, 5, 7, 2, 6], [4, 7, 1, 5, 2, 8, 6, 3], [4, 1, 5, 8, 6, 3, 7, 2]

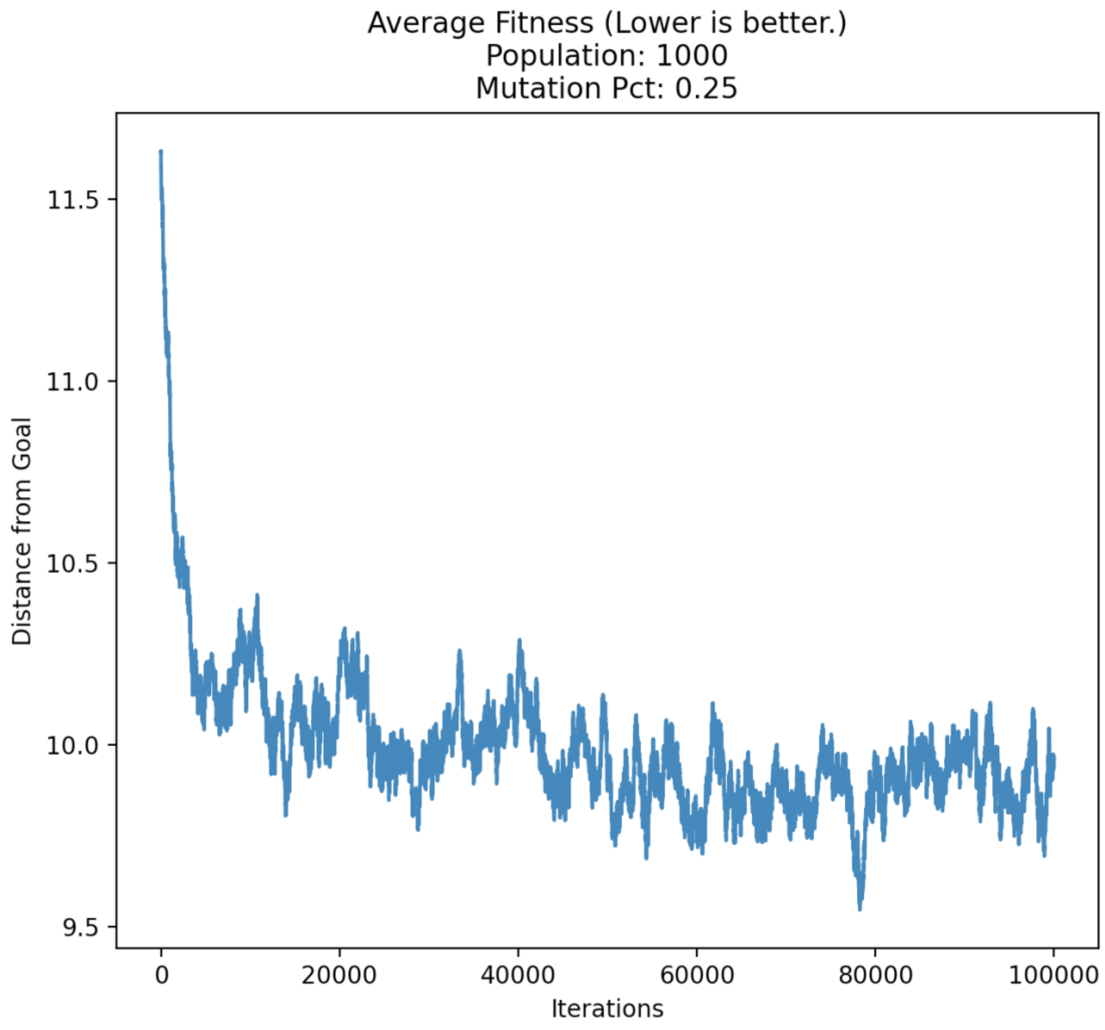
As evidenced above, increasing the population to 1000 has a drastically positive effect on how well the genetic algorithm performs. There's a consistent downward trend that appears to level out at ~9.85 and fluctuate between 9.75 and 10.25. Given the fluctuations aren't as low as a population of 100, the grouping of average fitness over iterations becomes much tighter and more consistent.



Starting Fitness: 11.322
Ending Fitness: 9.973
Fitness of best start: 3
Fitness of best end: 1
Final Round Was: 100000

[3, 6, 3, 3, 6, 4, 2, 5], [2, 8, 3, 1, 7, 1, 1, 6], [5, 8, 2, 5, 3, 1, 4, 8], [1, 3, 7, 2, 2, 8, 6, 4], [6, 1, 3, 7, 3, 8, 8, 2], [3, 6, 8, 1, 4, 1, 7, 2], [8, 4, 1, 8, 5, 2, 6, 3], [5, 5, 1, 4, 6, 3, 6, 2], [5, 5, 1, 4, 6, 3, 6, 2], [4, 2, 8, 6, 1, 7, 5, 1]

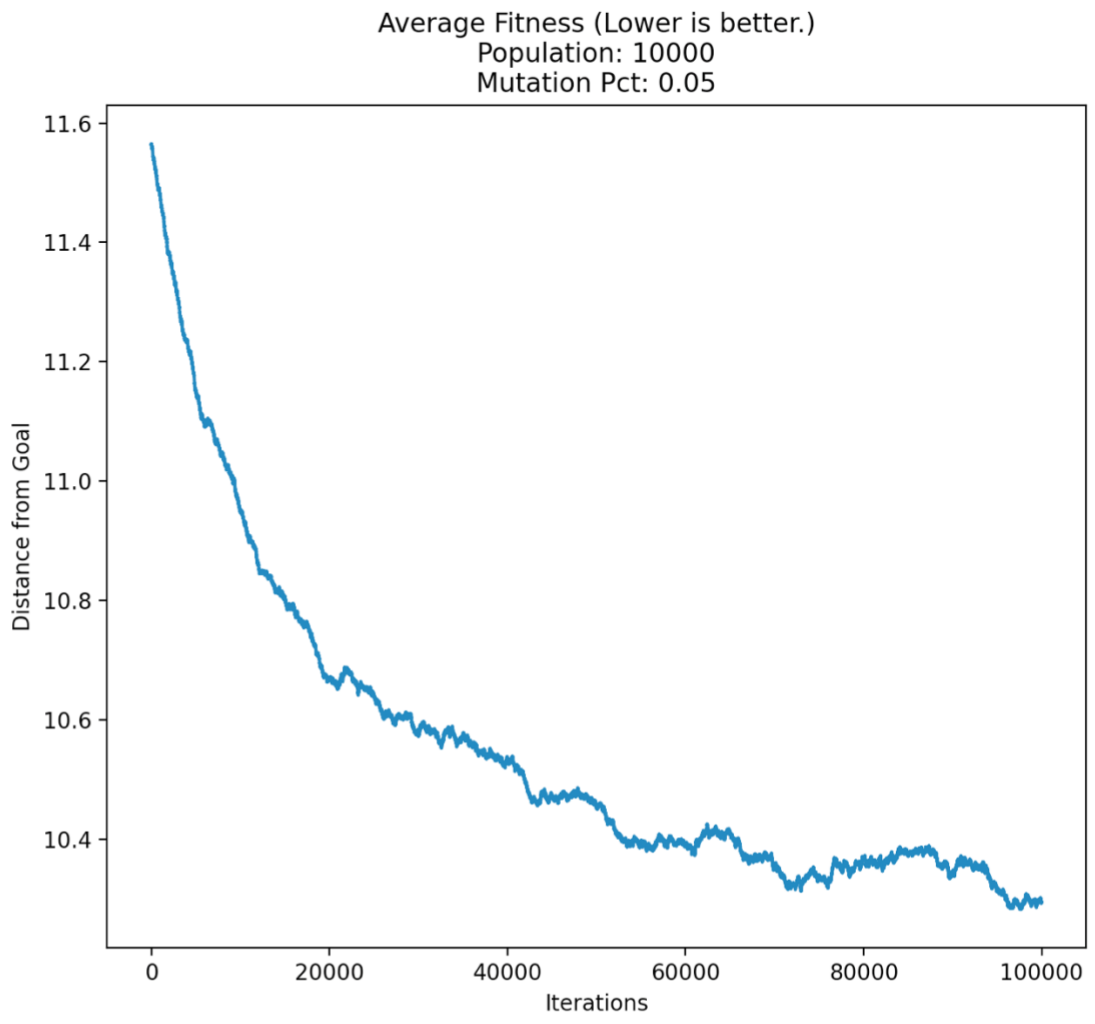
Increasing the mutation rate to .15 on a population of 1000 causes a faster and smoother decline to about the same result as the .05 mutation rate.



Starting Fitness: 11.624
Ending Fitness: 9.94
Fitness of best start: 3
Fitness of best end: 1
Final Round Was: 100000

[5, 8, 6, 4, 7, 7, 4, 7], [3, 7, 7, 1, 1, 5, 8, 6], [6, 1, 3, 7, 3, 8, 3, 5], [8, 8, 4, 1, 7, 5, 1, 6], [3, 6, 8, 1, 1, 4, 2, 7], [5, 5, 5, 1, 4, 6, 8, 3], [2, 5, 1, 8, 1, 3, 6, 8], [1, 8, 5, 2, 2, 7, 2, 4], [7, 3, 1, 8, 2, 5, 3, 1], [2, 7, 1, 3, 8, 6, 4, 2]

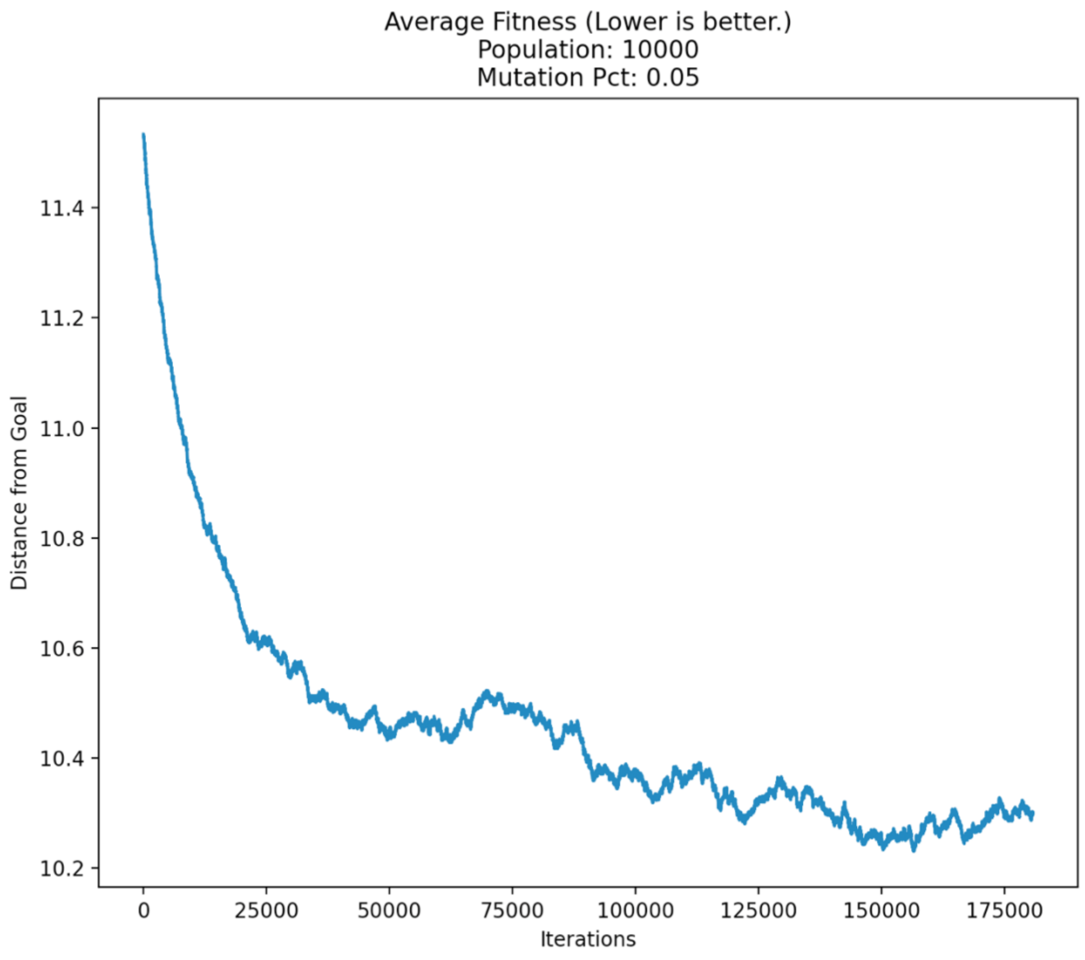
Increasing the mutation to .25 on a large population of 1000 caused the genetic algorithm to converge to a range of fluctuation slower in addition to fluctuations being slightly less dramatic. In general performance decreased as the fluctuation range was slightly smaller in general. There was not a measurable benefit to a higher percentage of mutations occurring.



Starting Fitness: 11.5645
Ending Fitness: 10.2932
Fitness of best start: 1
Fitness of best end: 1
Final Round Was: 100000

[6, 4, 7, 1, 4, 2, 5, 3], [3, 6, 3, 7, 4, 1, 8, 5], [3, 7, 4, 1, 8, 5, 5, 2], [3, 8, 2, 4, 1, 7, 4, 6], [3, 8, 2, 4, 1, 7, 4, 6], [3, 8, 2, 4, 1, 7, 4, 6], [6, 4, 7, 1, 4, 2, 5, 3], [6, 4, 7, 1, 4, 2, 5, 3], [3, 8, 2, 4, 1, 7, 4, 6], [3, 7, 4, 2, 8, 6, 1, 7]

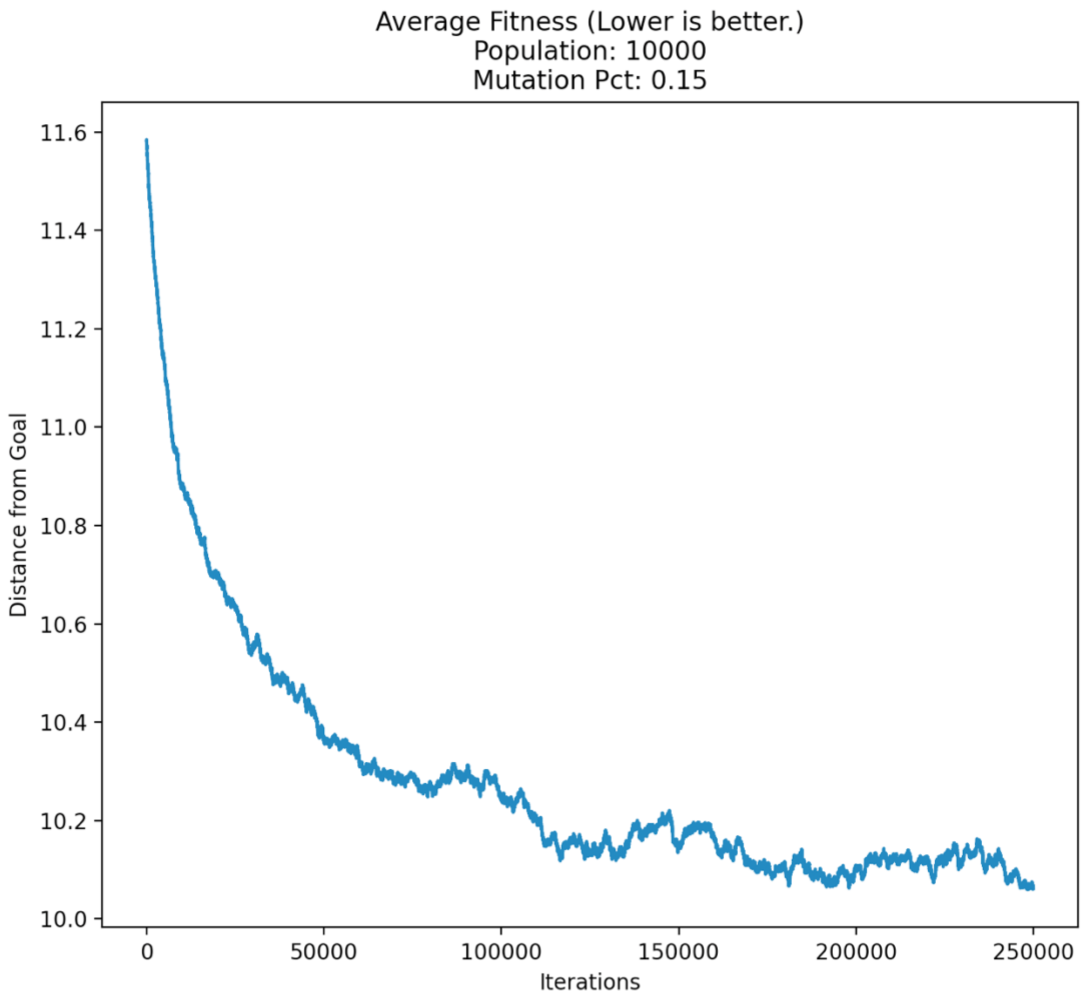
Increasing the population to 10,000 dramatically increases the consistency with which improvement of the average population fitness happens. It causes convergence to happen much slower however, and this performs significantly worse over the same number of iterations. In addition this calculated 1 Billion individuals compared to the previous 100 Million. A 10x increase in calculated states.



Starting Fitness: 11.5344
Ending Fitness: 10.2988
Fitness of best start: 1
Fitness of best end: 0
Final Round Was: 180743 – Ended Early

[1, 6, 1, 5, 2, 8, 3, 7], [2, 5, 1, 8, 4, 2, 7, 3], [2, 5, 1, 4, 7, 3, 6, 3], [5, 1, 8, 4, 7, 1, 3, 6],
[1, 6, 1, 5, 2, 8, 3, 7], [1, 6, 1, 5, 2, 8, 3, 7], [1, 6, 1, 5, 2, 8, 3, 7], [5, 1, 8, 4, 7, 1, 3, 6],
[1, 6, 1, 5, 2, 8, 3, 7], [4, 7, 5, 2, 6, 1, 3, 8]

Running the previous configuration for an additional 75,000 iterations (1.75 Billion cumulative individuals!) shows that the genetic algorithm implementation levels out at about 10.3 average population fitness. In addition larger populations take longer to converge to their approximate average fitness minimum.



Starting Fitness: 11.583
Ending Fitness: 10.0614
Fitness of best start: 2
Fitness of best end: 1
Final Round Was: 250000

[6, 3, 1, 8, 1, 3, 7, 2], [2, 7, 5, 3, 1, 4, 4, 8], [6, 1, 5, 2, 8, 3, 7, 3], [6, 1, 5, 2, 8, 3, 7, 3], [6, 3, 1, 7, 1, 8, 2, 5], [6, 3, 1, 7, 1, 8, 2, 5], [4, 6, 1, 5, 7, 1, 3, 8], [6, 8, 1, 5, 8, 2, 4, 7], [6, 8, 1, 5, 8, 2, 4, 7], [7, 3, 1, 7, 5, 8, 6, 4]

This is the final set of results as the calculations start to get hairy (over 2.5 Billion individuals were calculated in this run). As can be seen, increasing the mutation rate for a large population has negative benefit on population fitness even given 250,000 iterations.