Intelligent Systems Assignment

Developing a Genetic Algorithm for the Travelling Salesman Problem.

Shane Whelan – 09005763

# 1. Parameter Settings Evaluated

In the ten submitted output files each combination of crossover and mutation were tested together with the same rates chosen for crossover (0.89) and mutation (0.09). My id number without the 0 at the start (9005763) was used as a random number generator seed and incremented by 1 for each run.

# 2. Most Effective Parameters

In experimenting with large versus small population sizes it was found that although small population sizes lead to impressive performance it also introduces erratic numbers for best fitness in each generation. Small population sizes still produced interesting final results though, finding a reasonably impressive path fairly quick. Large population sizes lead to best fitness numbers tapering off and stagnating after a many generations. A population size of 100 was found to be about right with reasonably impressive numbers produced. This medium population size also meant that even though the numbers tapered off with the right mix of crossover and mutation there stood a chance for best fitness to improve.

Choosing a large number of generations was by far the best way to get good results. A generation of 100,000 was found to run for about 7.5 minutes on a current desktop hardware (Intel Ivy Bridge, Solid State Drive (SSD), 8gb RAM) interestingly the algorithm was exactly *twice* as slow on two year old laptop hardware. It was hard to test all possible values for conversion rate and mutation rate but by choosing a large generation number it was found to get the best results and compensate for poor choice of parameters. If I was doing the assignment again and if I had more time I would have automated the testing of parameters and just let the algorithm run for a few days collecting the best results.

I found exchange mutation to be the least effective mutation by far and I suspect that is caused by it being too random. Alternating Position crossover seemed to be the least effective crossover.

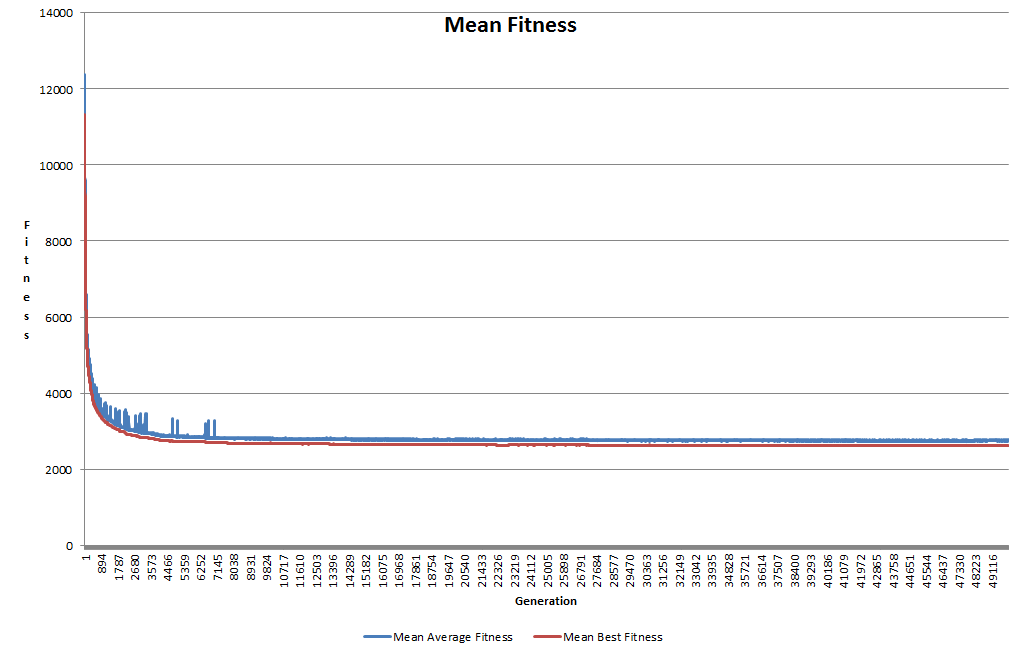
Originally I made the mistake of mutating tours in the current mating pool, saving them in that pool and not putting them into the next mating pool. This yielded an impressive result of 2109.88 on generation 449758 with parameters - 100 500000 m 0.89 i 0.9 output.txt 9005772. If I had more time I would have implemented a way for the algorithm to continue running if it generated good results close to the end of a test. Another interesting idea was the idea of elitism – where the best individual is copied from generation to generation – but for medium population sizes and up it’s not really necessary as the best individual will most lightly be selected anyway.

After much testing it was found that crossover rate of 0.89 and mutation rate of 0.09 were effective and achieved good results. The parameters (100 500000 m 0.89 v 0.09 output.txt 9005772) have a best fitness of 2115.88 on generation 458687. This result can be seen in output\_10.txt.

# 3. Optional Operators Chosen

I chose Maximal Preservative Crossover (MPX) as my optional crossover and Inversion Mutation as my optional mutation. When both optional operators were chosen together some of the strongest results were achieved. Maximal Preservative crossover chooses a sub tour whose length is greater than or equal to 10 and smaller than the path size divided by two. This means there is a decent level of inheritance from the parent strings. MPX only destroys a limited number of edges.

Inversion mutation randomly selects a sub tour removes it from the parent tour and inserts it in a randomly selected position in reverse order. It’s a relatively simple form of mutation but yielded some impressive results.



*Graph showing mean average fitness and mean best fitness across 10 output files.*