Using Genetic Algorithms to find Solutions to the Travelling Salesman Problem

Ray Keating — COSC3P71 Assignment 2

I. Introduction

THIS paper is an investigation into the effectiveness of genetic algorithms applied to the ”travelling salesman problem”. The travelling salesman problem is a popular problem in computer science where a salesman is given a number of cities to visit, and he must visit each city exactly once before returning to the city where he began.

It is not feasible to solve the travelling salesman problem using brute force, as the number of possible routes grows at an extremely fast rate compared to the number of cities. Namely, given n cities there are n factorial possible routes. That means that using brute force to calculate the best route out of 50 cities would take $3.04 * 10^{64}$ steps to calculate.

Instead of using brute force, we will be using a genetic algorithm to try and find a reasonably good solution in far less time. The basic idea behind a genetic algorithm is to represent the problem as a string or chromosome, and then apply the basic principles of evolution to come up with a good solution. Those principles are selection, crossover, and mutation. In this paper, we will examine the effectiveness of varying crossover rates, mutation rates, crossover techniques, and selection techniques.

The python code used for implementation of the genetic algorithm is available and is attached in the assignment submission. The results of each experiment are in the ”results” folder.

II. Background

As briefly explained in the introduction, genetic algorithms attempt to mimic the process of natural evolution. This is done via selection of the most fit solutions, crossover of those solutions, and mutation of those solutions. This process will be further explained.

A potential solution to a problem is represented as a string. We call these ”solution strings” chromosomes. In the case of the travelling salesman problem (in this implementation), a chromosome is essentially a route, represented as a series of integers, where each integer corresponds to a city. There are two constraints on the chromosomes: 1. a city must not appear twice in the chromosome (unless it is the starting city), and 2. every city in the set of cities must appear.

Initially, we create a ”population”, a set of completely randomized potential solutions. Through our psuedo-evolutionary process, we can gradually improve the fitness of our population over a number of generations. Fitness refers to how well a chromosome solves the problem. In the case of the travelling salesman problem, fitness is simply the distance of the path. Shorter paths are more fit than longer paths.

Each generation, we select individuals to compete in ”k-tournaments”, tournaments where random chromosomes are selected from the population and their fitness is compared to each other. The winner (the chromosome with the best fitness) will be put into the ”mating pool”. These tournaments are repeated until our mating pool is full.

Once our mating pool is full, we apply crossover to get the next generation. Crossover is essentially combining two parent chromosomes, and mixing their ”genetic material” to get new chromosomes called children. This allows the population to come up with novel solutions to the problem. Crossover can be implemented in a number of different ways. In our experiments, we look at two crossover techniques: ”uniform order crossover”, or UOX, and ”partially mapped crossover”, or PMX. These crossover techniques are explained in more detail in section III below.

After a mating pool of more fit chromosomes is created using tournament selection, and crossover is applied to the chromosomes in the mating pool, we get the next generation of solutions. Finally, the offspring undergo random mutation. For this problem, mutation is implemented through a simple algorithm where two cities swap places with each other.

In most GA’s, a small number of the most fit chromosomes (less than 10 percent of the whole population), are replicated into the next generation, meaning they are directly copied into the next generation without undergoing crossover or mutation. This process of carrying the most fit chromosomes into the next generation is called elitism.

Here is some pseudocode to help recap the basic procedure of this genetic algorithm

```
generate random initial population
for each generation
    replicate a few of the most fit individuals
    select mating pool using k-tournament selection
    apply crossover to the individuals in the mating pool to produce offspring
    apply mutation to the offspring
```

III. Experimental Setup

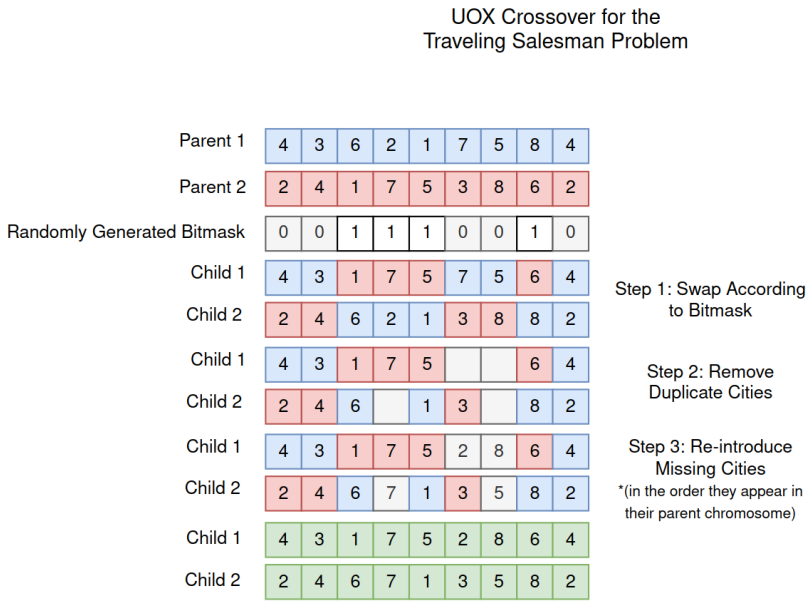
Our investigation will look at the change in fitness values (path length) over a number of generations. We will compare the results from our two crossover techniques previously described (PMX/UOX), as well as look at the impact that different combinations of parameters have on the overall fitness. These parameters are as follows: crossover rate (the chance that two parents will crossover), mutation rate (the chance that offspring will have a mutation applied to them), tournament size, elite size (the number of most fit individuals who replicate into the next generation)

Our results come from two different data sets, one with 22 cities (ulysses22.txt) and one with 51 (eil51.txt). For each data set, we look at the performance of PMX and UOX independently. In total, we have 4 experiments: two crossover techniques times two data sets. For each experiment, we set our population size to 75, and run the GA for 1000 generations. We run the GA 5 times with the same parameters and average out the best/average fitness to get the result fitness data. In some cases, 1000 generations is overkill, but in the larger data set, we do see some improvements up to 1000 generations in. For each experiment, we plot the results from these 5 parameter sets:

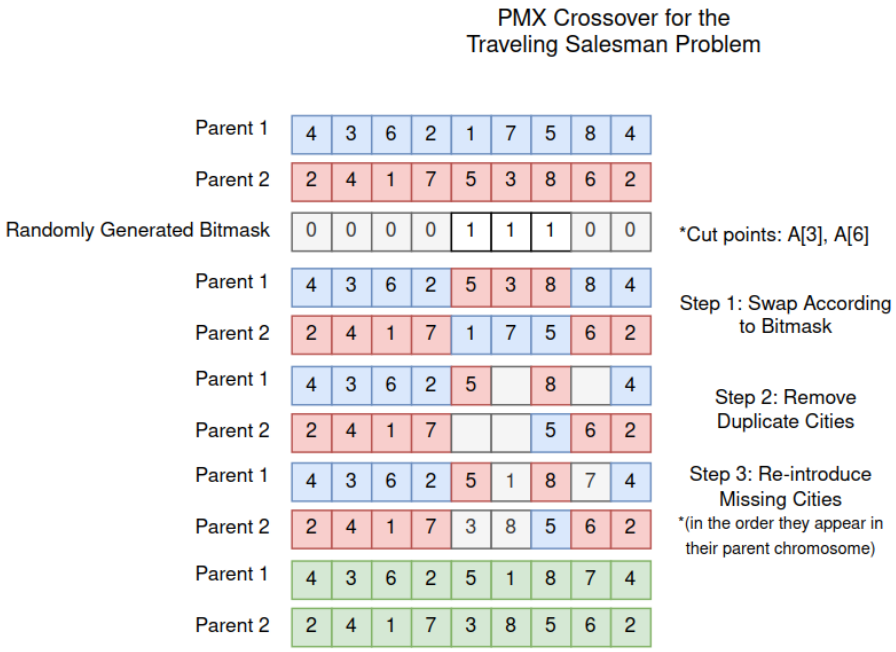
1) Crossover rate: 100%, Mutation rate: 0%, Tournament Size: 3, Elite Size: 5
2) Crossover rate: 100%, Mutation rate: 10%, Tournament Size: 3, Elite Size: 5
3) Crossover rate: 90%, Mutation rate: 0%, Tournament Size: 3, Elite Size: 5
4) Crossover rate: 90%, Mutation rate: 10%, Tournament Size: 3, Elite Size: 5
5) Crossover rate: 90%, Mutation rate: 30%, Tournament Size: 4, Elite Size: 15

Crossover Techniques Used: For this experiment, I have implemented two different crossover techniques. Specific details of the two techniques are explained below and more detail can be found in the code.

1) UOX: UOX is a crossover technique where a randomly generated bitmask is used to combine solutions to a problem. In the travelling salesman problem, UOX must be altered so that duplicate cities don't occur. My specific implementation of UOX is described in the graphic below. Note that the chromosomes in green would be the final result.



2) PMX: I implemented my PMX crossover in exactly the same way as UOX, however the bitmask is not a random set of 0s and 1s. Instead, 2 arbitrary ”cut points” are selected, and all points between the two cut points are set to 1, and the other points set to 0.



IV. Results

Here we can see the results as a series of line graphs. For more detail, the results can be found in the attached spreadsheet documents in the assignment folder.

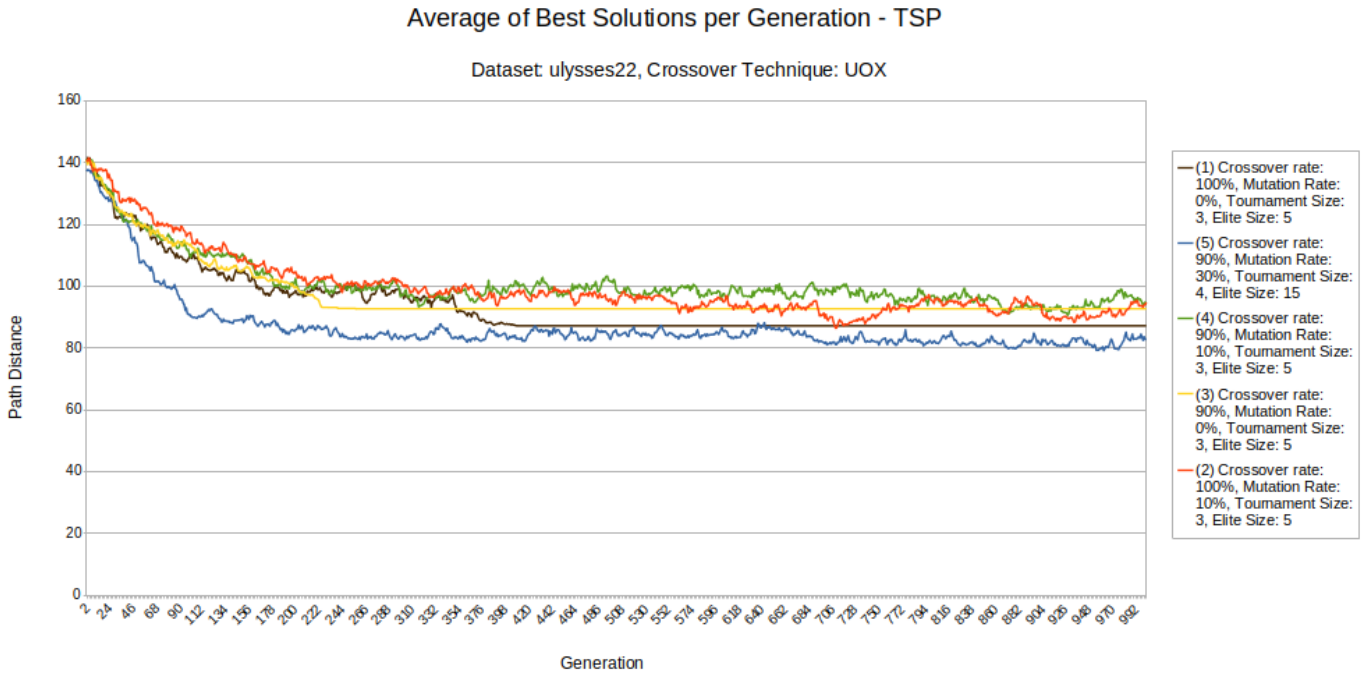


Fig. 1. UOX crossover on the smaller data set (22 cities).

TABLE I
STATISTICS SUMMARY: UOX CROSSOVER ON SMALLER DATA SET

Parameter Set	Best Distance	Worst Distance	Mean Distance	Median Distance	Standard Deviation
1	87.22	141.60	93.73	87.22	11.01
2	86.45	141.54	99.69	96.36	10.74
3	92.72	140.71	97.01	92.72	9.68
4	90.63	141.26	100.85	98.44	8.50
5	79.14	137.74	87.03	84.08	10.15

The solutions to this GA start to converge around generation 250. I would not recommend using 1000 generations on this dataset, but that's what I've done here. Parameter set 3 and 4 seem to converge totally, getting stuck at a single solution around generation 250. A common trend among all of the graphs is present here, where the GA's with a mutation rate set above 0 outperform those with a mutation rate of 0.

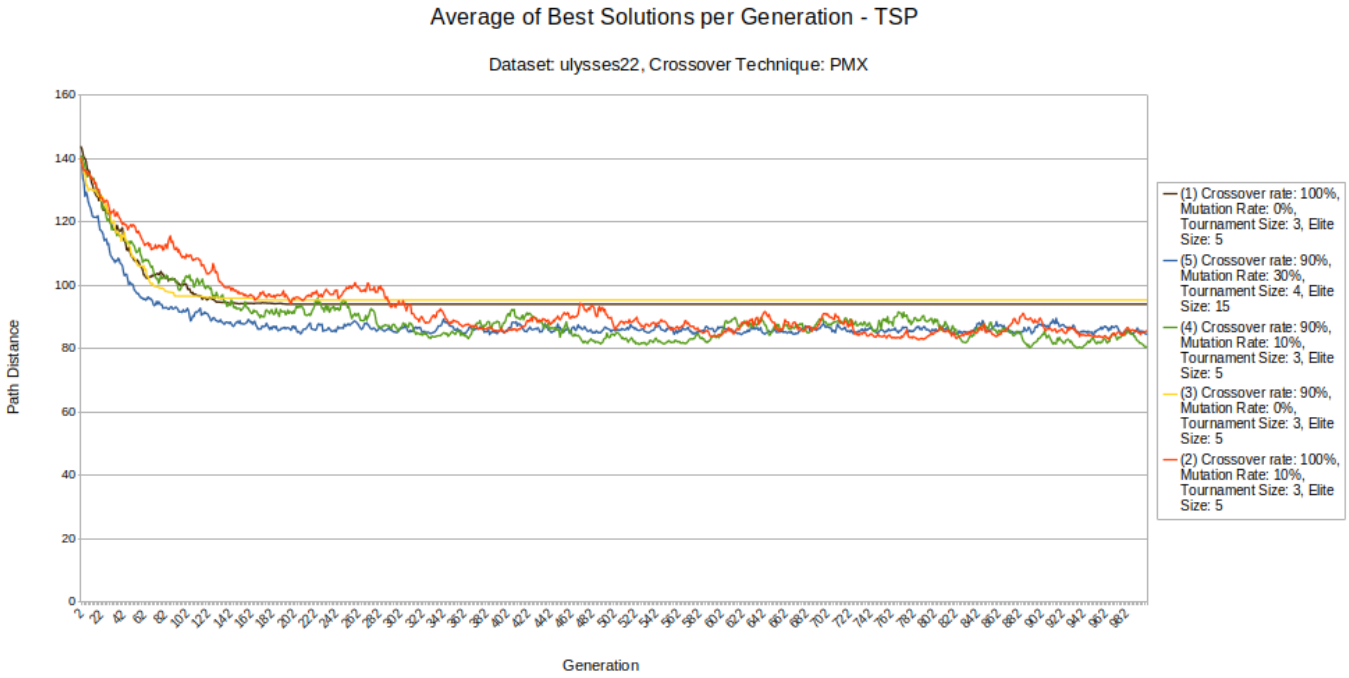


Fig. 2. PMX crossover on the smaller data set (22 cities).

TABLE II
STATISTICS SUMMARY: PMX CROSSOVER ON SMALLER DATA SET

Parameter Set	Best Distance	Worst Distance	Mean Distance	Median Distance	Standard Deviation
1	93.98	143.78	96.01	93.98	7.06
2	82.78	140.22	92.62	88.42	10.81
3	95.30	139.99	97.00	95.30	6.47
4	80.10	140.77	89.77	87.28	9.84
5	84.36	138.88	88.11	86.20	6.87

Again, we have an example of GA's without any mutation getting stuck at a certain solution. An indication that there is no diversity in the population and since only crossover can be applied, no novel solutions can be found. The rest of the parameter sets seem to perform equally well.

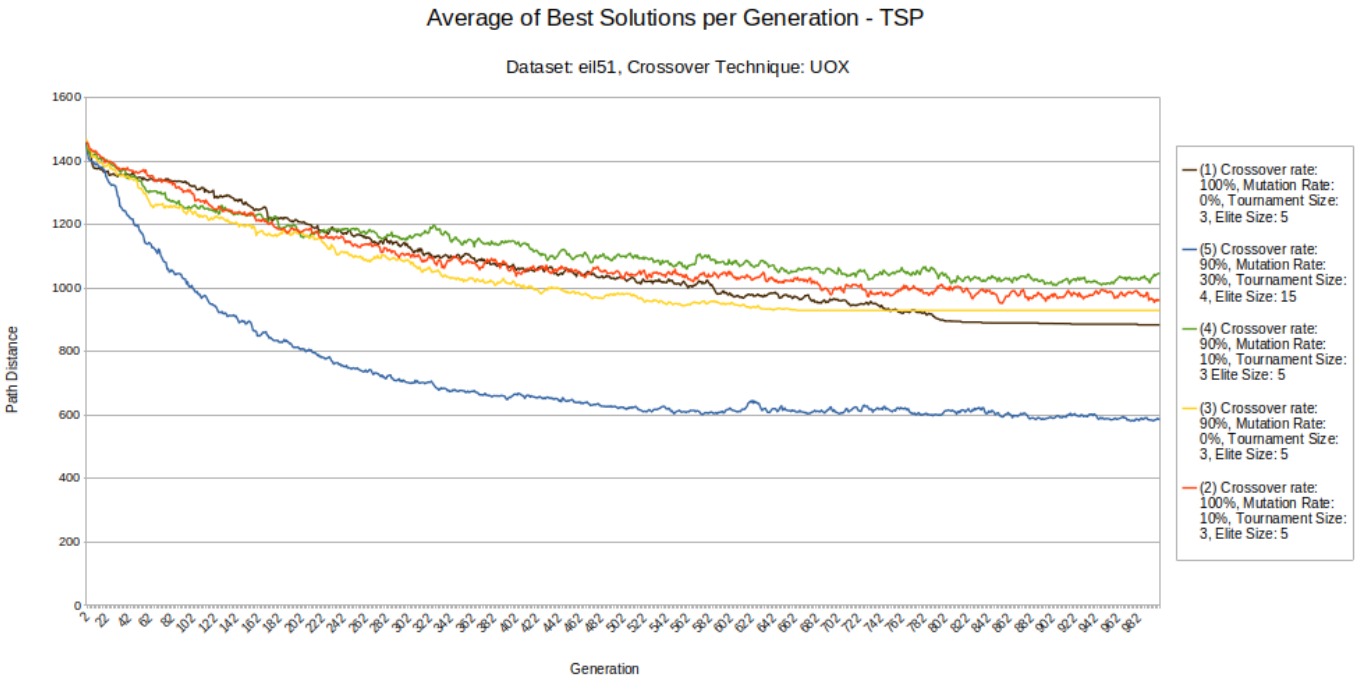


Fig. 3. UOX crossover on the larger data set (51 cities).

TABLE III
STATISTICS SUMMARY: UOX CROSSOVER ON LARGER DATA SET

Parameter Set	Best Distance	Worst Distance	Mean Distance	Median Distance	Standard Deviation
1	883.97	1460.16	1061.89	1030.67	150.85
2	952.12	1458.99	1088.83	1047.44	119.45
3	929.57	1467.95	1033.33	972.11	130.56
4	1008.94	1447.24	1125.00	1096.38	96.86
5	580.95	1456.47	719.60	627.77	186.79

This data set is one of the most interesting. Clearly parameter set 5 has outperformed all the other ones pretty significantly. A higher mutation rate seems to prevent convergence at a single solution. In the larger data set, even the GA's with a mutation rate of zero don't seem to get stuck as easily. This is probably due to the fact that my crossover technique has a certain element of mutation built into it.

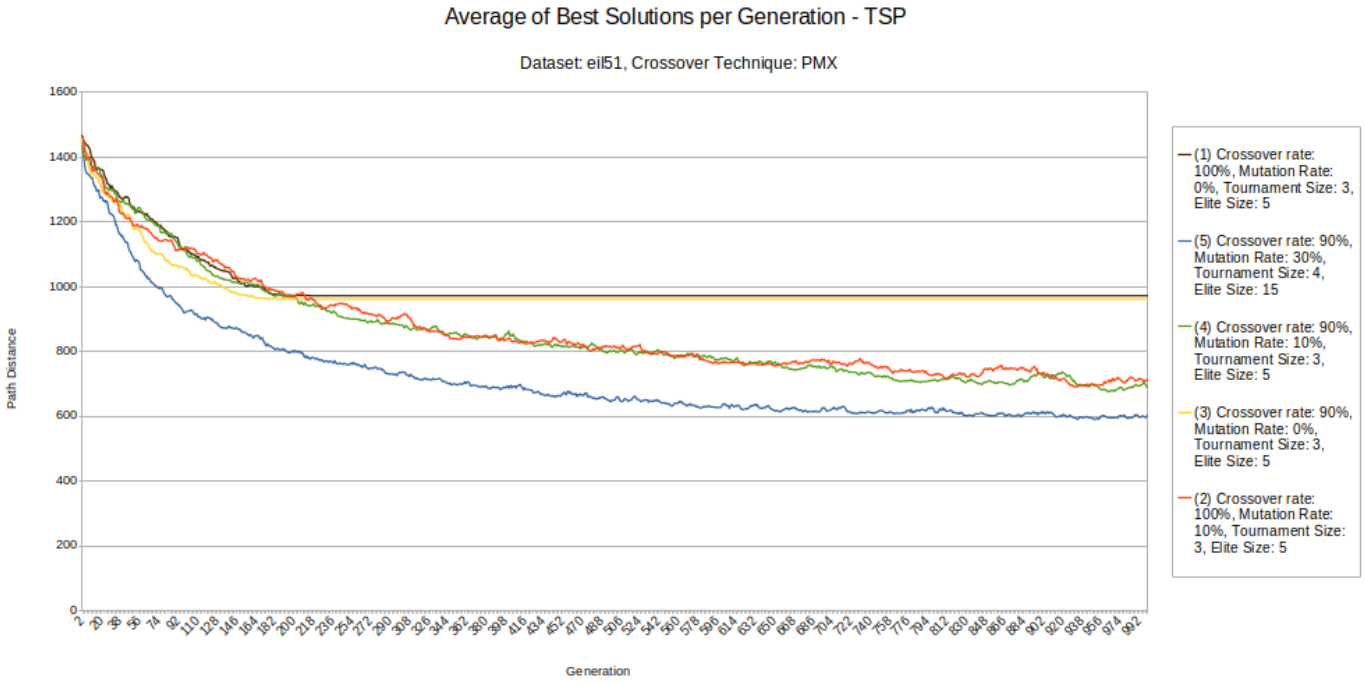


Fig. 4. PMX crossover on the larger data set (51 cities).

TABLE IV
STATISTICS SUMMARY: PMX CROSSOVER ON LARGER DATA SET

Parameter Set	Best Distance	Worst Distance	Mean Distance	Median Distance	Standard Deviation
1	971.55	1467.85	1005.48	971.55	91.64
2	689.97	1464.86	862.86	811.48	157.78
3	961.36	1458.56	987.57	961.36	79.38
4	675.32	1447.93	853.37	802.15	164.37
5	590.34	1451.35	719.45	652.81	160.82

The PMX crossover seems to reduce the amount of implicit mutation in the crossover process because GA's with parameter sets 3 and 4 have converged far before the other GA's. This may indicate that a PMX crossover has a lower amount of mutation inherent to the crossover technique. Although UOX had the shortest distance in the entire experiment set, the PMX best distances seemed to be lower in general. However, there is not enough evidence to suggest that PMX is a superior technique compared to UOX.

V. DISCUSSIONS AND CONCLUSIONS

Given the results of the experiments, there is little that can be said concretely about the effectiveness of the two different crossover techniques (PMX and UOX). Similarly, the crossover rate, tournament size, and elite size seem to have little effect. The one factor which seemed to play an important role in ensuring novel solutions would come up was the mutation rate, but only in the PMX experiments. In every PMX experiment, the GA's which mad a mutation rate greater than zero outperformed the GA's whose mutation rates were zero. In the UOX experiments, the mutation rate had less of an effect.

From this, we can conclude that when using PMX as our crossover technique for the traveling salesman problem we should ensure that we increase the mutation rate, to prevent convergence.

Genetic algorithms are quite a beautiful idea. It is hard to think of a more intuitive solution to solving a problem than simulating the real world. The most intriguing thing is that we are essentially just managing randomness in a controlled way. It is exciting to think that a machine can handle such huge amounts of randomness and actually use that to come up with a solution to a problem. For reference, I've added a visualization of what a solution in the initial generation might look like, vs what the final solution turns out to be, all through managing the randomness.

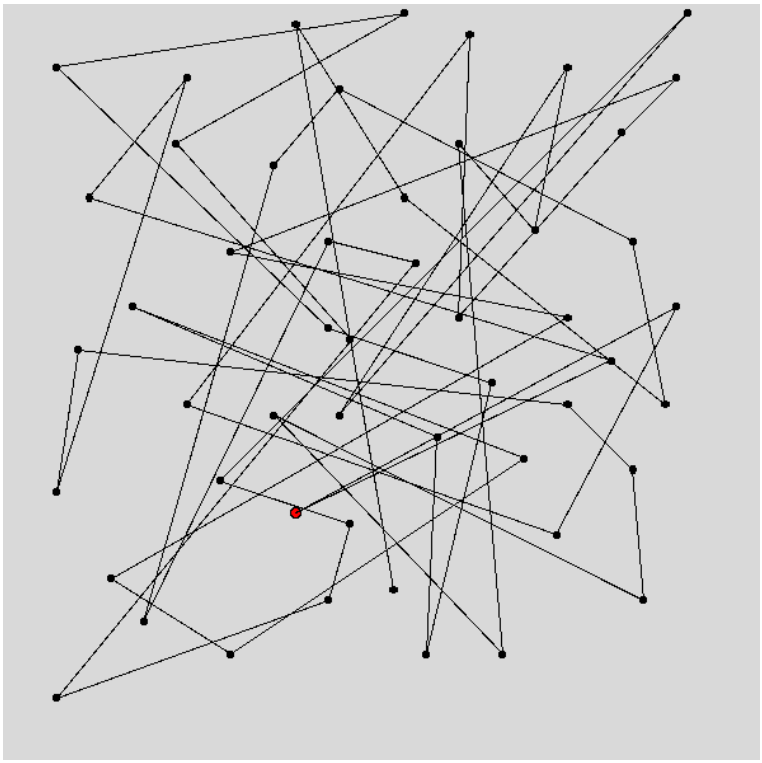


Fig. 5. Solution at generation 0

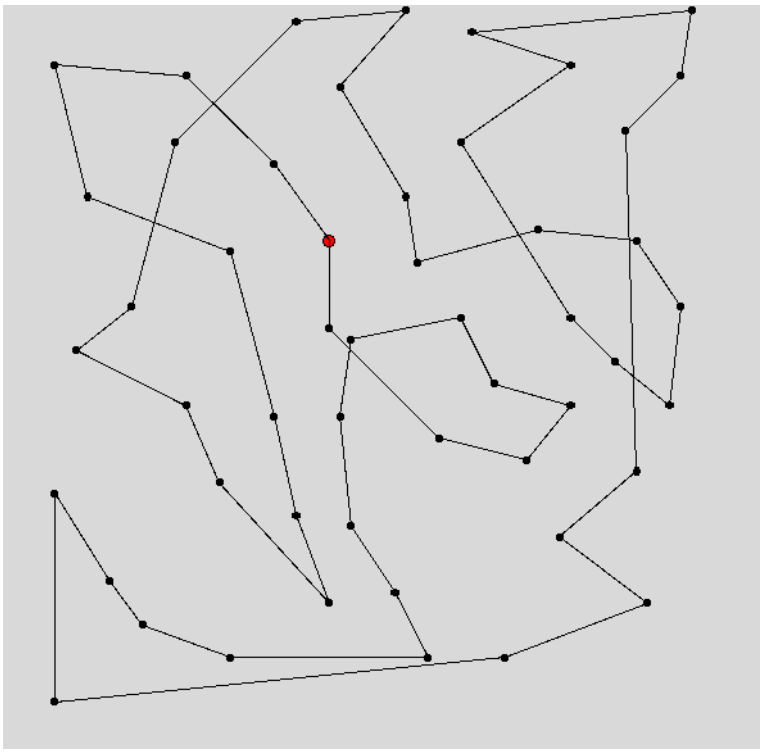


Fig. 6. Solution at generation 1000

VI. References

Dr. Beatrice Ombuki-Berman, “GA.PPT.”, Powerpoint Slides, COSC3P71, Brock University