Travelling Salesperson Problem – Genetic Algorithm with Wisdom of Crowds

Lindie Chenou
[Computer Science & Engineering]
Speed School of Engineering
University of Louisville, USA
lcchen01@louisville.edu

1. **Introduction** (What did you do in this project and why?)

The goal of this project is to calculate the optical cost of visiting each city in a data file. The problem is best known as A Traveling Salesperson Problem (TSP). This project is requiring using the Genetic Algorithm from previous project including Wisdom of Artificial Crowds to find the shortest route, given a dataset of 11, 22, 44, 77, 97, and 222 cities.

2. **Approach** (Describe algorithm you are using for this project)

In TSPs, in a closed loop that visits each node once, a number of nodes or 'cities' must be visited with the intention of minimizing the distance covered over the total path. The TSP serves as a classic example of an NP-complete problem where methods of computationally scalable solutions are not understood to ensure optimal solutions. As the size of the problem increases, infeasible computational resources are quickly needed for optimal solution methods. Instead, different approximation algorithms are employed in order to get close to optimal results. Despite computational complexity present in TSPs, for at least some iterations of the problem, the evidence from observing human

performance is that people are able to construct solutions quickly, while still retaining good performance.

The principle of the Wisdom of Crowds (WoC) is be implementing using the genetic algorithm with some other changes. The genetic algorithm is for solving both constrained and unconstrained optimization problems that are based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. It uses Chromosome, crossover, mutation, and survival of the fittest. One of the biggest advantages of using a Genetic algorithm is the ability to not know how to solve for the optimal solution. What is needed to be able to evaluate the quality of the generated solution coming and through iteration you get a good solution.

The first to do for the genetic algorithm is to randomly generate chromosomes, which are called population. Each chromosome represents an individual solution. With the initial population created, the evolutionary generational cycle begins.

```
def initialPopulation(popSize, cityList):
    def createRoute(cityList):
        route = random.sample(cityList, len(cityList))
        return route
    population = []

    for i in range(0, popSize):
        population.append(createRoute(cityList))
    return population
```

After the initial population has been randomly created the next function that needs to be implemented is the fitness function. The point of the fitness function is to identify how good an individual is; it determines how to fit an individual is. It will assign a fitness score to each string. With that, the individuals with higher fitness value have a bigger probability of contributing offspring to the next generation.

```
class Fitness:
    def __init__(self, route):
        self.route = route
        self.distance = 0
        self.fitness= 0.0

    def routeDistance(self):
        if self.distance ==0:
            pathDistance = 0
            for i in range(0, len(self.route)):
                fromCity = self.route[i]
                toCity = None
                if i + 1 < len(self.route):
                    toCity = self.route[i + 1]
                else:
                    toCity = self.route[0]
                pathDistance += fromCity.distance(toCity)
            self.distance = pathDistance
        return self.distance

    def routeFitness(self):
        if self.fitness == 0:
            self.fitness = 1 / float(self.routeDistance())
        return self.fitness
```

With the fitness and initial population created, the next function is the selection function. The meaning behind the selection phase is to select the fittest parents and to let them pass on their genes to their children or the next generation. For the selection, two parents will be selected, and parents with a high fitness level will have the chance to reproduce.

The final function being added are the crossover and the mutation function. For the crossover function; in the first parent, a subset will be randomly selected. In which the remaining route of the string will be fill by the second parent where no genes are repeated, and they are inserted in the same order as they were for the second parent. In GA, mutation plays an important role as it helps to prevent local convergence by adding new pathways that will enable the program to explore other parts of the solution space. The swap mutation function will be implemented. With a defined low probability, two cities will swap places on their route. The mutate function is going to implement it for one individual. To compare the value of the crossover and the mutation functions, there will be two of each implementation for both of them. The population size and the mutation rate will both be investigated. Both of these are coming from Eric Stolz and the lecture corresponding with the evolution of a salesman. Using the number of generations as the stopping criteria for the result.

```
def mutatePopulation(population, mutationRate):
    def mutation(individual, mutationRate):
        for swapped in range(len(individual)):
            if(random.random() < mutationRate):
                swapWith = int(random.random() * len(individual))

                city1 = individual[swapped]
                city2 = individual[swapWith]

                individual[swapped] = city2
                individual[swapWith] = city1
        return individual
    mutatedPop = []

    for ind in range(0, len(population)):
        mutatedInd = mutation(population[ind], mutationRate)
        mutatedPop.append(mutatedInd)
    return mutatedPop
```

```
def crossoverPopulation(matingpool, eliteSize):
    def crossover(parent1, parent2):
        child = []
        childP1 = []
        childP2 = []

        geneA = int(random.random() * len(parent1))
        geneB = int(random.random() * len(parent1))

        startGene = min(geneA, geneB)
        endGene = max(geneA, geneB)

        for i in range(startGene, endGene):
            childP1.append(parent1[i])

        childP2 = [item for item in parent2 if item not in childP1]

        child = childP1 + childP2
        return child
    children = []
    length = len(matingpool) - eliteSize
    pool = random.sample(matingpool, len(matingpool))

    for i in range(0,eliteSize):
        children.append(matingpool[i])

    for i in range(0, length):
        child = crossover(pool[i], pool[len(matingpool)-i-1])
        children.append(child)
    return children
```

With everything completed for the Genetic Algorithm, it's time to implement the WoC. First repeat the Genetic Algorithm program for over 20 times to get sufficient data. While running it's important to retrieve the best data or solution for every run that is done with the GA program.

```
for k in range(20):
    bestRoute = geneticAlgorithmPlot(population=cityList, popSize=100, eliteSize=20, mutationRate=0.01, generations=500)
    bestRouteList = []
    sX = []
    sY = []
    IndexRoute = []
    for j in range(len(bestRoute)):
        bestRouteList.append((bestRoute[j].x, bestRoute[j].y))
        sX.append(bestRoute[j].x)
        sY.append(bestRoute[j].y)
        #import pdb; pdb.set_trace()
        IndexRoute.append(key_list[val_list.index(bestRouteList[j])])
    sX.append(bestRoute[0].x)
    sY.append(bestRoute[0].y)
    plotPath(sX, sY)
    #print(IndexRoute)
    crowd.append(IndexRoute)
```

Following the previous step, is to aggregate all of the runs into a single solution. Using the aggregate matrix function to accomplish getting a single solution for the Wisdom of Crowds.

```
def agg_matrix(A:list):
    for r in A:
        for s in range(len(r)-1):
            u = r[s]
            v = r[s+1]
            Agg[u,v] +=1
        Agg[len(r)-1,r[0]] +=1
    return Agg
for k in range(cityList_np.shape[0]):
    cityList_dict[k] = tuple(cityList_np[k])
key_list = list(cityList_dict.keys())
val_list = list(cityList_dict.values())
#import pdb; pdb.set_trace()
cityList = []
for i in range(0,cityList_np.shape[0]):
    cityList.append(City(x=cityList_np[i][0], y=cityList_np[i][1]))
```

3. **Results** (How well did the algorithm perform?)

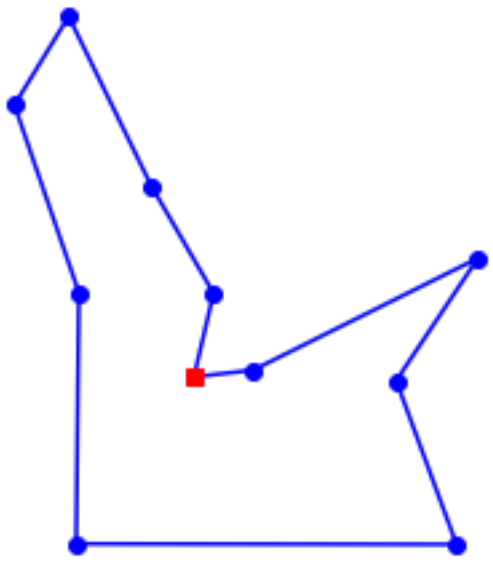
3.1 Data (Describe the data you used.)

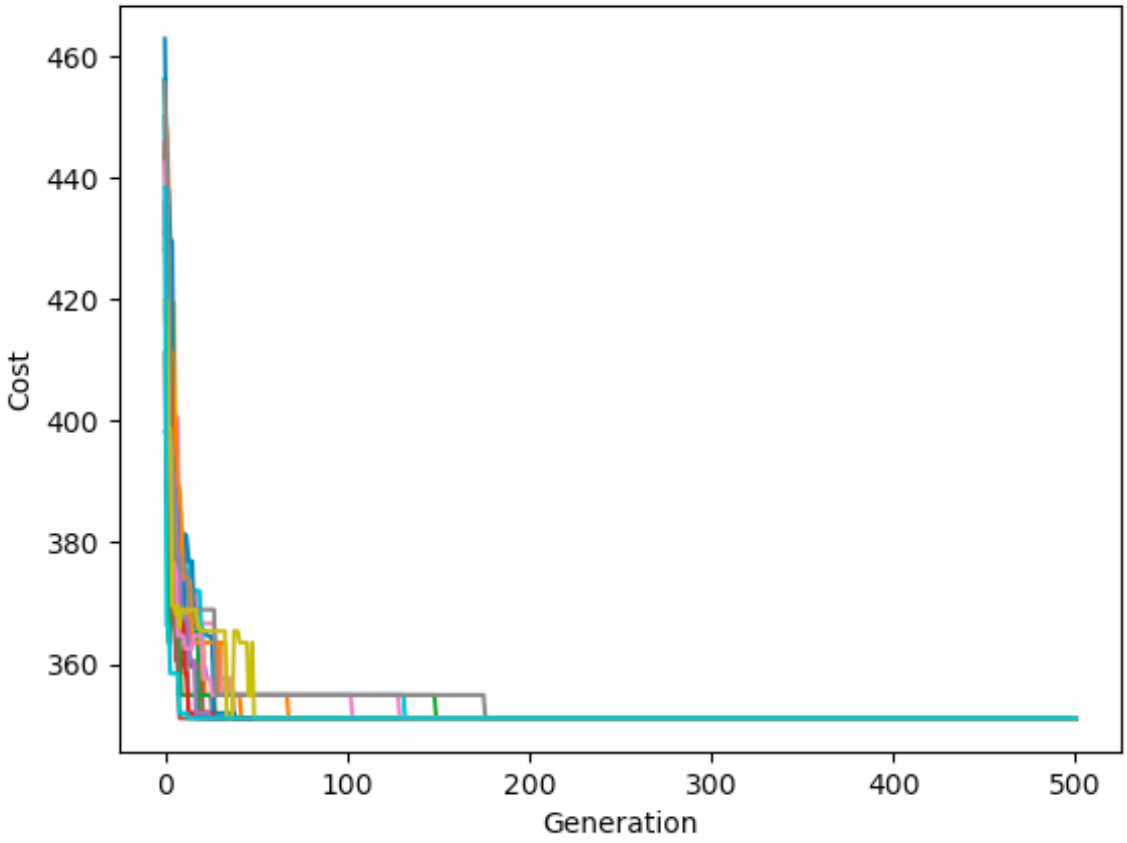
A set of cities/nodes and their coordinates where provide. The sets contained 11, 22, 44, 77, 97, and 222 cities. Cities and also provided the connection between each city.

Results (Numerical results and any figures or tables.)

The program was able to handle all of the files provided to test the program. With Random11 file, running the genetic algorithm produce the same number for the final distance. With the result retrieve for the Wisdom of Crowds being 182.85, it shows that there was a great deficit in the final distance produce from the Genetic algorithm.

```
Initial distance: 450.1289933553935
Final distance: 351.04587994184993
Initial distance: 448.18441027261645
Final distance: 351.04587994184993
Initial distance: 456.05408644150447
Final distance: 351.04587994184993
Initial distance: 431.1577624189039
Final distance: 351.04587994184993
Initial distance: 443.5465652869908
Final distance: 351.04587994184993
Initial distance: 411.19552621019403
Final distance: 351.04587994184993
Initial distance: 435.2665210954042
Final distance: 351.04587994184993
Initial distance: 443.98870325320706
Final distance: 351.04587994184993
Initial distance: 447.11563122510506
Final distance: 351.04587994184993
Initial distance: 456.2387262502274
Final distance: 351.04587994184993
Initial distance: 462.9025294945583
Final distance: 351.04587994184993
Initial distance: 418.7998636156223
Final distance: 351.04587994184993
Initial distance: 428.23113407671934
Final distance: 351.04587994184993
Initial distance: 435.86716639472564
Final distance: 351.04587994184993
Initial distance: 398.22166833542053
Final distance: 351.04587994184993
Initial distance: 445.7903935864473
Final distance: 351.04587994184993
Initial distance: 442.54059541087463
Final distance: 351.04587994184993
Initial distance: 455.66026697940623
Final distance: 351.04587994184993
Initial distance: 419.76924903913584
Final distance: 351.04587994184993
Initial distance: 438.35076988330115
Final distance: 351.04587994184993
```



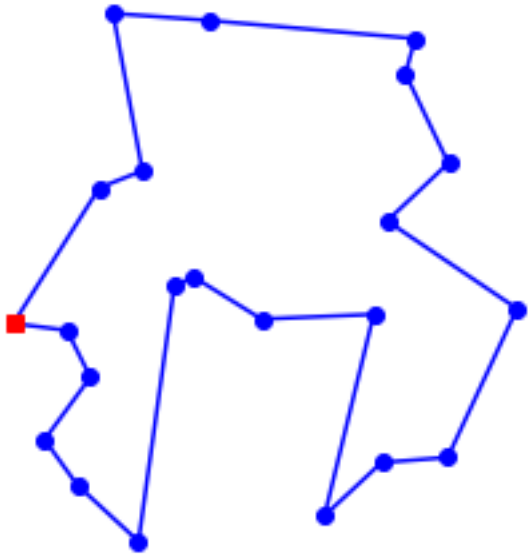


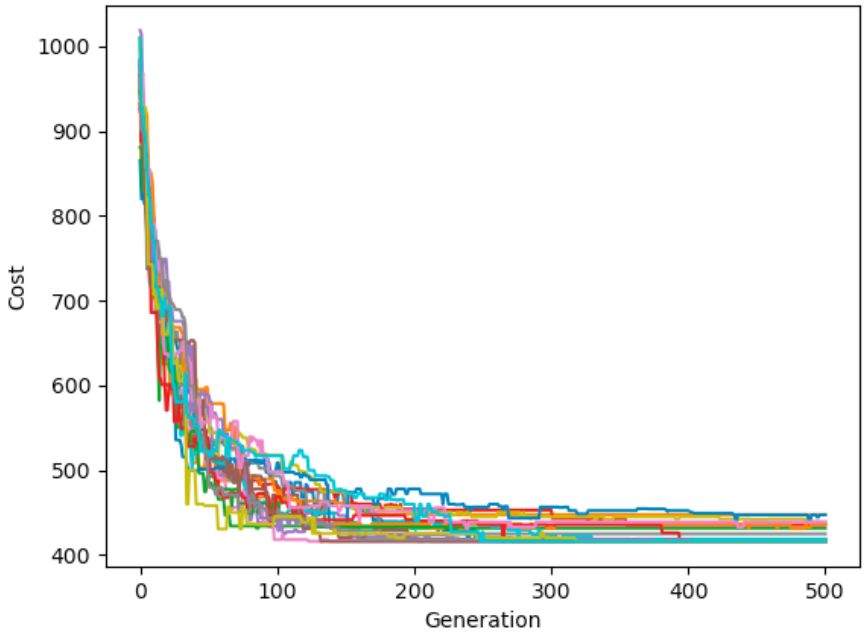

```
[0, 6, 2, 9, 7, 10, 1, 8, 4, 3, 5] 182.85378276305357
```

For the Random22 file, the median for the genetic algorithm receive for all 20 compilation was 417.287. With the result of the Wisdom of Crowds being 379.337, the deficit with WoC was much lesser than the difference receives with the Random11 file.

```
[0, 6, 14, 2, 15, 10, 20, 16, 7, 11, 5, 9, 12, 18, 3, 8, 13, 1, 19, 21, 17, 4] 379.33756209990435
```

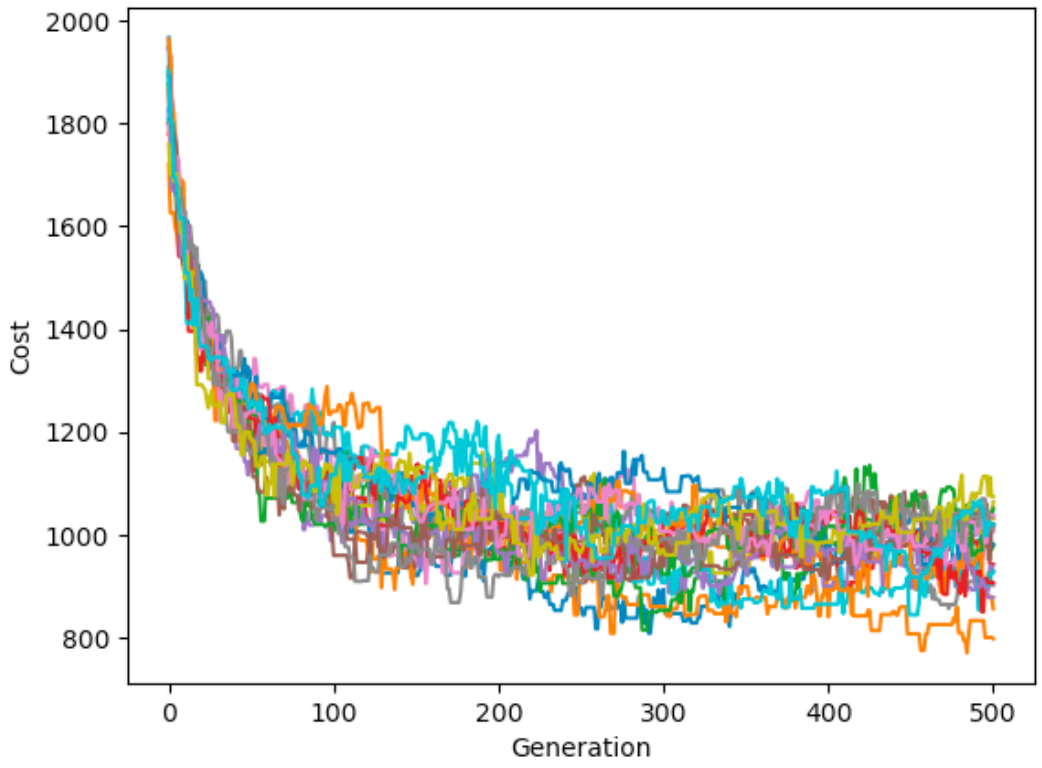
```
Initial distance: 978.7634532114414
Final distance: 418.4804953226307
Initial distance: 956.3487674277467
Final distance: 447.6136763869483
Initial distance: 948.4626199697318
Final distance: 431.56764428472053
Initial distance: 978.9635510796993
Final distance: 435.548728682813
Initial distance: 923.0229432764105
Final distance: 416.0946418662266
Initial distance: 926.0438581488596
Final distance: 416.0946418662266
Initial distance: 980.4152315367244
Final distance: 416.0946418662266
Initial distance: 984.4568408521947
Final distance: 424.8851931335046
Initial distance: 1010.7859828971104
Final distance: 438.78006511676057
Initial distance: 945.6444759323045
Final distance: 418.719176722867
Initial distance: 865.1219387852667
Final distance: 447.4158299439008
Initial distance: 945.8222620514299
Final distance: 436.6315793885517
Initial distance: 880.8381517358275
Final distance: 416.0946418662266
Initial distance: 966.1558596769663
Final distance: 416.0946418662266
Initial distance: 1018.7985411340642
Final distance: 416.0946418662266
Initial distance: 931.7442010592495
Final distance: 416.0946418662266
Initial distance: 953.6029007522749
Final distance: 439.0309366902317
Initial distance: 983.2752707449299
Final distance: 416.0946418662266
Initial distance: 1008.0064863305039
Final distance: 416.0946418662266
Initial distance: 1009.4190007636106
Final distance: 416.0946418662266
```



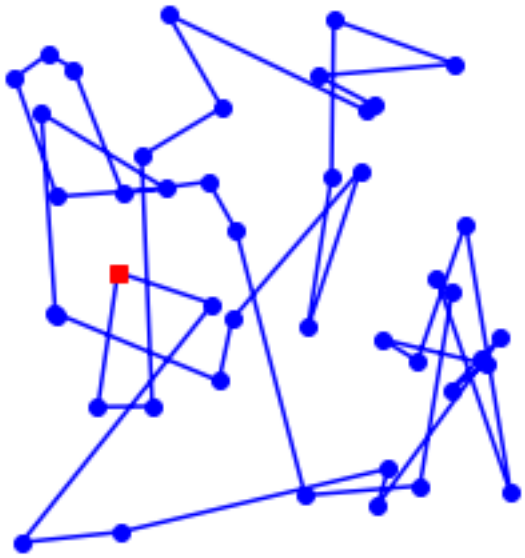


For the Random44, the Wisdom of Crowds result was 907.178. for this file the WoC was not able to reduce the optimal path receive from the Genetic Algorithm. The GA receives multiple result being lower than the WoC that was generated. Compare to the median receive from the GA, WoC was able to reduce the optimal path of GA. The median value received for GA was 994.736.

```
[0, 21, 27, 29, 11, 18, 12, 36, 32, 2, 40, 41, 37, 38, 9, 34, 26, 6, 22, 28, 8, 23, 24,
10, 33, 31, 14, 15, 1, 35, 17, 13, 16, 42, 43, 25, 30, 20, 39, 4, 3, 7, 5, 19]
907.1783036337778
```



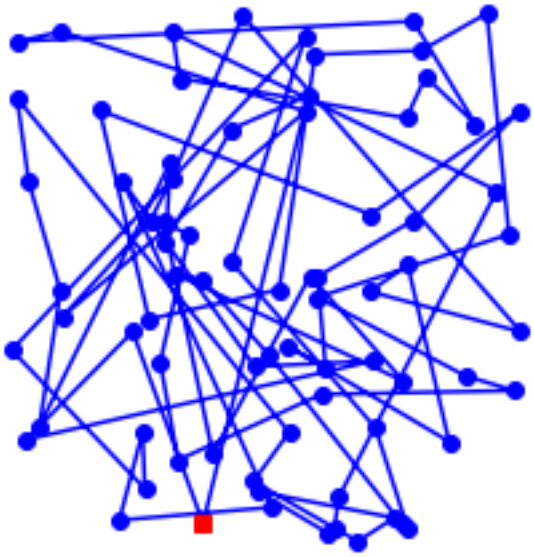
```
Initial distance: 1877.150314890222
Final distance: 1020.6726488468968
Initial distance: 1721.4384109023802
Final distance: 798.5916075285515
Initial distance: 1909.5159714351687
Final distance: 980.4143286130512
Initial distance: 1946.5811679900808
Final distance: 942.9389097456839
Initial distance: 1800.9173300439954
Final distance: 928.9852340534478
Initial distance: 1872.7411719170395
Final distance: 1032.985021849226
Initial distance: 1828.9400153432478
Final distance: 1008.0633266958507
Initial distance: 1894.2141197026478
Final distance: 902.4275833043309
Initial distance: 1868.2849662580668
Final distance: 1064.2380992941498
Initial distance: 1826.2814391665117
Final distance: 929.5775481798622
Initial distance: 1966.844028430363
Final distance: 981.4103267944394
Initial distance: 1963.2954419923994
Final distance: 856.6370429999936
Initial distance: 1800.7890510042105
Final distance: 1051.4126483273367
Initial distance: 1779.0391197932156
Final distance: 907.3682347490485
Initial distance: 1893.3159352967934
Final distance: 878.5761973742506
Initial distance: 1864.2334963523265
Final distance: 1019.5588918102054
Initial distance: 1855.4604749772673
Final distance: 1038.6199684031462
Initial distance: 1808.5770489080162
Final distance: 1012.8067148293868
Initial distance: 1759.152369790038
Final distance: 1074.6483428664162
Initial distance: 1901.2481261451273
Final distance: 1020.4541973572741
```

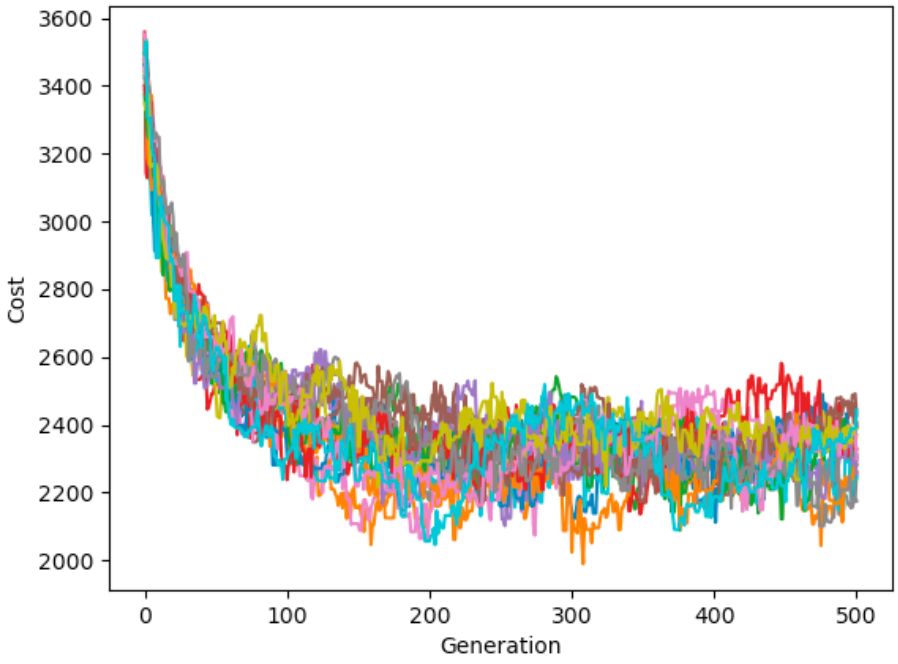


For the Random77, the program failed with reducing the optimal path receive from the Genetic Algorithm. What was received for WoC was 2378.017, compare to the median receive from GA being 2300.47 and the lowest value being 2174.5994. The result does fluctuate at lot. One of the results showed that WoC did succeed in reducing the optimal path from GA.

```
[0, 38, 52, 19, 20, 59, 3, 25, 5, 11, 55, 21, 15, 44, 36, 57, 18, 8, 27, 50, 37, 63, 45,
66, 70, 65, 12, 33, 49, 23, 9, 56, 51, 32, 46, 47, 13, 62, 61, 73, 54, 72, 22, 10, 42, 39,
67, 69, 43, 24, 60, 76, 35, 1, 58, 48, 29, 26, 6, 40, 16, 31, 53, 68, 14, 17, 28, 74, 7,
71, 30, 41, 34, 75, 64, 4, 2] 2378.017050318844
```

```
Initial distance: 3463.5481652665626
Final distance: 2310.1162716567046
Initial distance: 3475.028021587772
Final distance: 2282.337048147859
Initial distance: 3537.5965026741255
Final distance: 2301.2502705817683
Initial distance: 3558.639122118023
Final distance: 2393.1091471822024
Initial distance: 3518.873368409408
Final distance: 2293.1334962045617
Initial distance: 3535.427692216538
Final distance: 2427.570289182833
Initial distance: 3399.2051505794348
Final distance: 2292.4122079175113
Initial distance: 3386.406794489045
Final distance: 2258.834128142936
Initial distance: 3544.585566146893
Final distance: 2370.579980248464
Initial distance: 3431.0618979734354
Final distance: 2242.589415026127
Initial distance: 3461.8101406000083
Final distance: 2393.3209587203714
Initial distance: 3443.059726544125
Final distance: 2261.697287813872
Initial distance: 3375.599172532509
Final distance: 2280.6712375008024
Initial distance: 3399.7394552971236
Final distance: 2328.5076723871807
Initial distance: 3460.8044208180386
Final distance: 2252.7203916934477
Initial distance: 3495.5397271910138
Final distance: 2404.62979697589
Initial distance: 3552.704528592485
Final distance: 2299.693752907709
Initial distance: 3422.6373573936535
Final distance: 2174.5994500849865
Initial distance: 3346.8868751876416
Final distance: 2387.9245269510875
Initial distance: 3492.4257389450113
Final distance: 2443.6925007017053
```

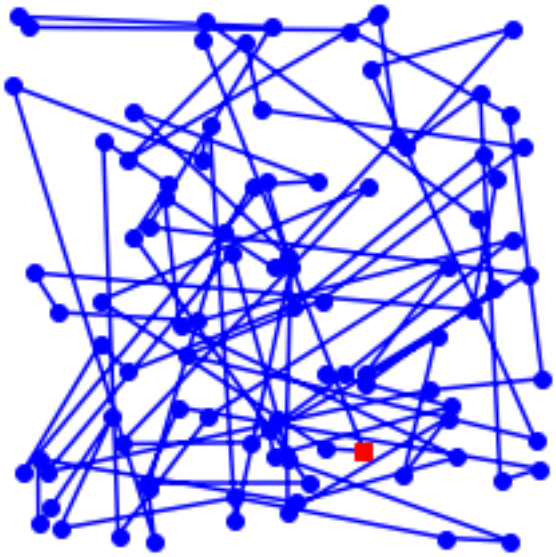


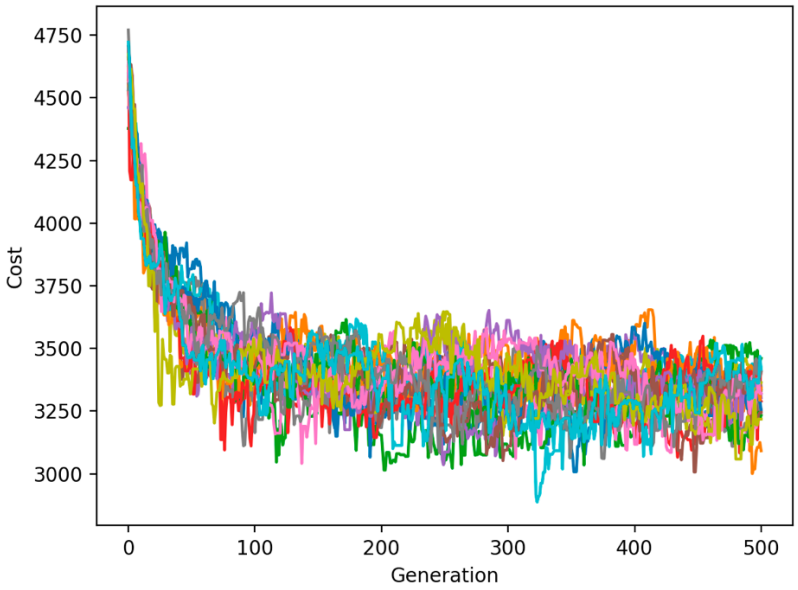


Similar to what happen with the Random77 file, Random97 file WoC result failed to reduce the Genetic algorithm optimal path. With the GA average being 3312.488 the WoC result was 4081.5562 which shows that WoC was very much not successful at all. The lowest GA value happen to be 3092.0535

```
[0, 43, 11, 34, 64, 57, 65, 10, 86, 85, 91, 42, 96, 95, 33, 6, 77, 90, 79, 73, 27,
74, 23, 24, 63, 75, 30, 26, 71, 22, 46, 58, 80, 59, 92, 39, 49, 37, 51, 89, 15,
48, 45, 12, 25, 76, 29, 83, 87, 40, 54, 53, 62, 66, 7, 52, 84, 3, 61, 78, 81, 55,
70, 19, 14, 67, 72, 32, 38, 2, 21, 5, 82, 69, 28, 1, 60, 4, 50, 93, 18, 13, 56,
44, 17, 36, 68, 16, 88, 41, 9, 20, 47, 35, 94, 31, 8] 4081.5562597016005
```

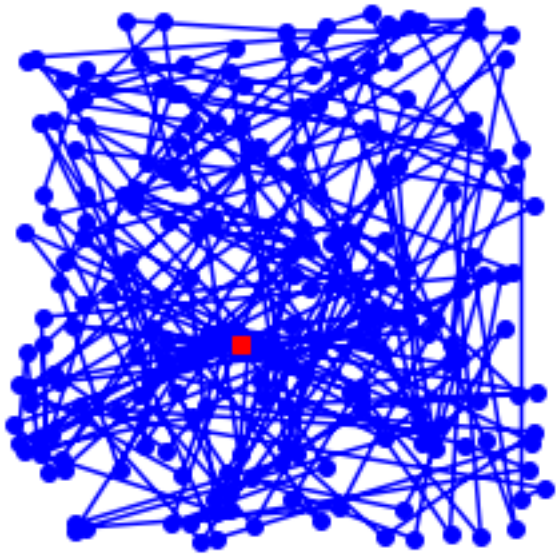
```
Initial distance: 4626.435628724264
Final distance: 3331.8288593009274
Initial distance: 4573.771594337463
Final distance: 3092.0535507706913
Initial distance: 4556.309227970092
Final distance: 3460.6428543990473
Initial distance: 4615.877777651508
Final distance: 3289.6877844073865
Initial distance: 4523.4334217310525
Final distance: 3303.7474847692174
Initial distance: 4591.551467549462
Final distance: 3340.0330179331113
Initial distance: 4685.539686800677
Final distance: 3242.7536242639103
Initial distance: 4770.739884619123
Final distance: 3254.0628124374525
Initial distance: 4547.316640935073
Final distance: 3364.185447159605
Initial distance: 4595.501346424383
Final distance: 3409.2377502540976
Initial distance: 4703.688368867067
Final distance: 3254.103262673444
Initial distance: 4621.651725127452
Final distance: 3341.9017728561953
Initial distance: 4378.0416550784075
Final distance: 3232.9212150325475
Initial distance: 4704.385377873289
Final distance: 3405.1231330227
Initial distance: 4461.214090340985
Final distance: 3320.227041207966
Initial distance: 4659.712648104884
Final distance: 3245.3241516162216
Initial distance: 4644.363298343393
Final distance: 3321.049700758831
Initial distance: 4533.029563348073
Final distance: 3405.8283930255793
Initial distance: 4674.280644003882
Final distance: 3218.447146914354
Initial distance: 4721.95421640377
Final distance: 3416.6057398927455
```

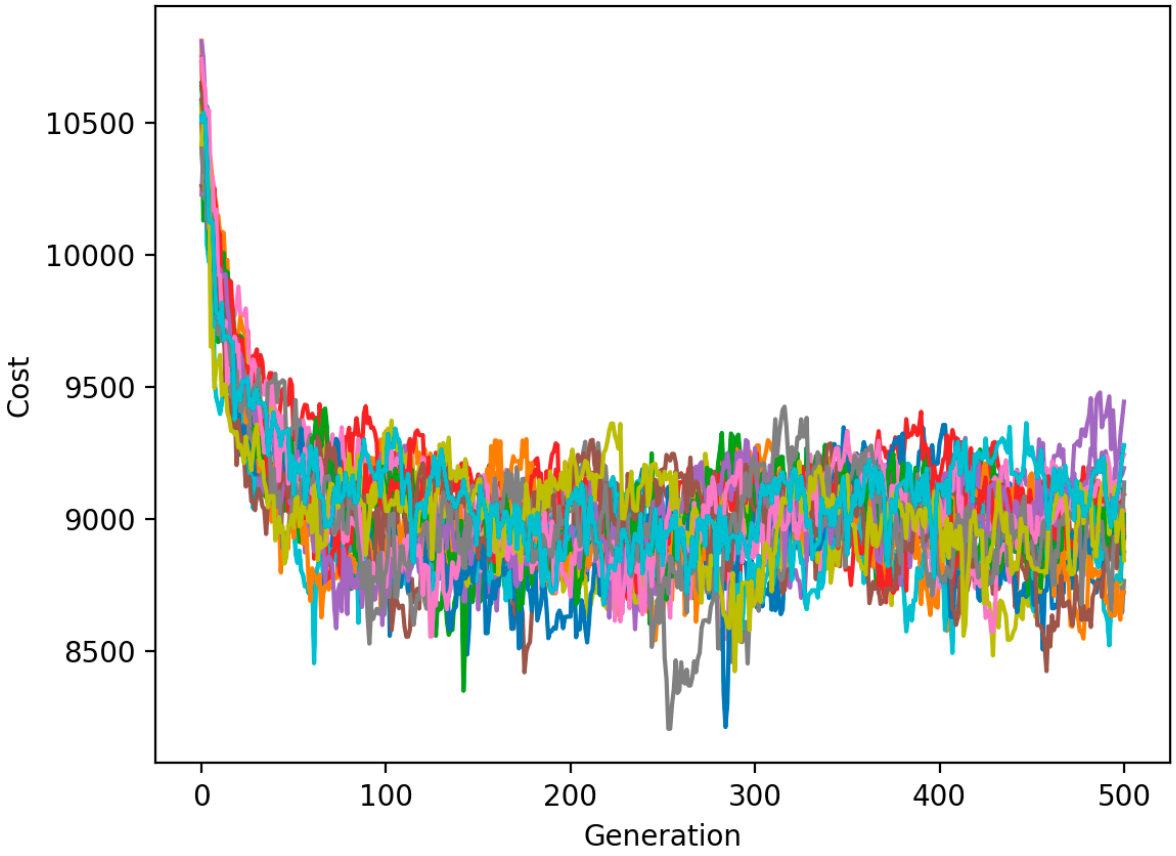




And lastly with the Random 222 file, WoC completely failed on every level. The initial values for genetic algorithm for the file were all under 11000 and the resulting Wisdom of Crowds value was 18858.400. The GA median was 8965.1731 with the lowest value being 8724.0356

```
Initial distance: 10651.264405027683
Final distance: 9133.619087761079
Initial distance: 10638.91681232726
Final distance: 8880.011360994547
Initial distance: 10584.813150558595
Final distance: 8975.893382551973
Initial distance: 10514.264765809428
Final distance: 8925.354515006973
Initial distance: 10230.24630148062
Final distance: 9445.683919583473
Initial distance: 10262.803071084027
Final distance: 8940.18115182781
Initial distance: 10810.591485537016
Final distance: 9034.413037900073
Initial distance: 10419.31246239231
Final distance: 8766.854713300152
Initial distance: 10541.892342347748
Final distance: 8910.277960727943
Initial distance: 10521.95324494484
Final distance: 8873.734510333541
Initial distance: 10729.951024082655
Final distance: 8932.647243495432
Initial distance: 10812.637282661837
Final distance: 8724.035680927534
Initial distance: 10651.455161678556
Final distance: 9018.563889102132
Initial distance: 10748.80402873405
Final distance: 9094.339315311528
Initial distance: 10809.43597917235
Final distance: 9194.142910760063
Initial distance: 10591.78116088108
Final distance: 8954.452963722693
Initial distance: 10744.060527151503
Final distance: 9244.77434334127
Initial distance: 10406.683909860709
Final distance: 9139.734623775043
Initial distance: 10419.915396358503
Final distance: 8845.154625785122
Initial distance: 10506.06877499499
Final distance: 9280.985309176878
```






```
[0, 101, 146, 180, 217, 47, 125, 110, 67, 76, 39, 64, 162,
141, 97, 58, 103, 69, 93, 221, 70, 104, 19, 154, 134, 203,
51, 193, 209, 21, 187, 52, 147, 53, 215, 204, 82, 14, 48,
156, 114, 113, 153, 6, 170, 160, 196, 26, 30, 107, 91, 216,
80, 142, 126, 23, 169, 75, 151, 50, 123, 40, 188, 164, 119,
172, 8, 195, 25, 18, 155, 16, 175, 68, 15, 5, 171, 118,
158, 66, 13, 78, 59, 45, 44, 72, 197, 60, 165, 31, 178,
210, 46, 139, 37, 92, 192, 10, 102, 57, 219, 35, 166, 131,
148, 41, 179, 9, 190, 65, 88, 136, 12, 73, 121, 28, 137,
71, 98, 202, 129, 94, 24, 120, 55, 17, 143, 74, 79, 49,
117, 194, 218, 207, 177, 106, 11, 90, 85, 174, 133, 157,
182, 33, 2, 32, 89, 185, 42, 212, 176, 56, 7, 206, 122,
200, 205, 161, 128, 109, 173, 77, 220, 211, 22, 167, 213,
138, 100, 214, 168, 84, 61, 87, 115, 43, 152, 62, 144, 159,
95, 145, 112, 140, 198, 54, 186, 27, 29, 201, 116, 130,
199, 96, 208, 183, 191, 149, 184, 135, 150, 4, 86, 105, 83,
81, 181, 111, 1, 34, 3, 189, 99, 63, 20, 127, 163, 132, 36,
38, 124, 108] 18858.40039695431
```

Algorithm	PopSize	EliteSize	mutationRate	Generation	Final Distance
GA_Random11	100	20	0.01	500	351.045
GA_WoC_Random11	100	20	0.01	500	182.85
GA_Random22	100	20	0.01	500	416.094
GA_WoC_Random22	100	20	0.01	500	379.33
GA_Random44	100	20	0.01	500	994.736
GA_WoC_Random44	100	20	0.01	500	907.178
GA_Random77	100	20	0.01	500	2300.47
GA_WoC_Random77	100	20	0.01	500	2378.017
GA_Random97	100	20	0.01	500	3312.488
GA_WoC_Random97	100	20	0.01	500	4081.5562
GA_Random222	100	20	0.01	500	8965.1731
GA_WoC_Random222	100	20	0.01	500	18858.4

4. **Discussion** (Talk about the results you got and answer any specific questions mentioned in the assignment.)
There are two key results from the experiments mentioned in section 3 of this document that are highlighted. The findings show that the wisdom of the solution of crowds provides a more efficient solution than any individual genetic algorithm for smaller files, but will fluctuate when dealing with larger files; when the relative number of chromosomes is sufficiently limited for any given genetic algorithm compared to the number of vertices in the graph. Secondly, the findings show that a higher threshold of dominance leads to a higher resultant distance traveled while not affecting the run time much.

With Genetic Algorithm, the biggest improvement regarding the final distance is the mutation rate. The lower the mutation rate happens to be the lower the result will be. While the population size does have a factor in minimizing the final distance, it not a major improvement from the original

population size of 100. With the mutation rate, there is about a 2000 drop on the final distance between 0.01 and 0.001. meanwhile to compile the project for each set of datasets it took about a minute for each one. This was completely different with Wisdom of Crowds, while testing when lowering the mutation rate, even though GA will happen to go down the WoC will stay consistent to the value it has been receiving.

5. **References** (If you used any sources in addition to lectures please include them here.)

https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35
https://www.sciencedirect.com/topics/engineering/genetic-algorithm
https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b