**CSE 545 – Artificial Intelligence – Project 5 – Wisdom of Crowds**

Jordan Richardson

BA Computer Science

Speed School of Engineering

University of Louisville, USA

[Jkrich05@louisville.edu](mailto:Jkrich05@louisville.edu)

**Introduction**

This project is a continuation of a previous project “TSP Genetic Algorithm” and revolves around the idea of implementing a hybrid algorithm for solving TSP through the combination of Genetic Algorithm and Wisdom of Crowds. We were given 6 sets of test date containing 11, 22, 44, 77, 97, and 222 city nodes respectively. I have also created my own test data containing 150, 300, and 500 city nodes. The basic idea behind WOC is that we will allow our GA to run multiple times and pick a selection of the best paths from each run, and then combine these paths using edge frequency analysis to piece them together in a way that returns a better solution, essentially using the wisdom of each run of the GA, to perform a run that returns even better results.

**Approach**

My approach for this problem was systematic I feel, in comparison to our other projects. Due to the experience, I acquired in writing my GA, I had a good idea on how to approach implementing Wisdom of Crowds. I began with implementing a function that keeps the arguments for population size and other variables the same, while doing another loop of the whole program, and storing the results. This was quite hard to analyze so before I went any further, I made it so that it only took the last 20% of my population as my genetic algorithm never regresses, therefore these are also the best 20% of the population. I then actually changed my GUI to a grid layout so I would have more room to view data and manipulate it as it filled my entire screen and more from top to bottom.

After that was completed, I began to work on WOC. I spent around 3 days working on it before essentially scrapping what I had and starting entirely over with my Genetic Algorithm as a base. I did this because my solutions basically all needed a greedy repair, and it was not calculating edges correctly. However, I had a little bit of experience and understanding of the Wisdom of Crowds approach, and I finished the assignment on my next attempt within 8-10 hours from top to bottom even including GUI and graphing. I used a matrix this time for counting the edges and assembled the path from this. I also had it consider distance to other cities when filling in edges, so it creates a preferrable path However, I did not use a greedy algorithm to fill in the remaining cities, and it simply would choose edges with less votes according to distance that matched its requirements to create an applicable path. From there I set up multiple runs again, and had it compared to the top 20% of each run.

However, I realized that right after assembly, the path was not always desirable compared to the best paths. Depending on input for population size, etc., it occasionally would return a worst path than the best path stored. After quite a bit of analysis on it, I realized I needed to let it run one last time with the WOC path inserted into the population, so it can fill in undesirable edges with better edges according to my crossover method and mutation method, while keeping the edges that are most desirable and it cannot find better. This worked great, so I decided to keep both as a representation of the improvement available after Wisdom of Crowds is implemented.

So that you have an idea of the basis for this application, the Genetic Algorithm for this problem does have some functionality since it is a continuation of a previous project, users are able to select from 2 mutation methods, along with 2 crossover methods. Users can modify the subsequent mutation rate and crossover rate, population size and max generations, as well as how many elites are brought over between generations and the size of the tournament selection. For the wisdom of crowds portion I did implement some functionality to allow the user to visualize changes due to different circumstances. Wisdom of Crowds allows users to modify the number of runs, while allowing each run to carry the previous settings for generations, population etc. However, to clarify, I have not added a percentage variable that indicates what percent of the top population is passed over, and it is currently at 10%. If the user desires to change it, you must change the code on line 544.

**Results**

I did not use any special libraries to perform calculations for me as I want to improve my coding capabilities, understanding of arrays, and the manipulation of data better. I also found that generally hardcoding it performs quicker than using a library, so I am glad I did so. For this project, I find that my Wisdom of Crowds path upon creation did improve most of the time over my individual best paths, however it was not a huge improvement, so I have it completing one more run with the implementation of the WOC path into the new population. Regardless, I still do not find my GAWOC finding the optimal path without giving it larger settings for variable, and I believe this is because I did not implement a Greedy Algorithm as I never needed one due to my solution satisfying TSP.

My Wisdom of Crowds is very basic in that I have an edge frequency matrix that counts how often each edge appears in paths, it then creates a path starting from 0, and iterating through until it has filled all spots, and it fills them with the unvisited city that has the highest edge frequency from the current city, and only if it’s a tie does it choose the city closest in terms of distance. It then adjusts the city numbers back to their original.

**Results – Data**

For this project we were supplied with 6 .TSP files that contained some basic information about how it was generated up to line 7, and a list of cities with city number, longitude, and latitude. Our project this time had a variety of city counts from 11 to 222 cities. I have also generated 3 of my own tests containing 150, 300 and 500 cities.

For this project I decided to display 2 different types of graphs, along with the solutions for WOC initially as well as after running a single generation with their path being implemented. I am already expecting it to not perform as well on small tsp files as the better the initial solutions are able to be, the harder it can be on the edge frequency matrix to accurately assign edges and create a better path, and then it would hit a local maxima even in the WOC generation as many of the edges it does have are still optimal, it is just unsure what to replace unless given enough time.

For this project and all the following data, I have decided to use static variables to show an accurate representation of the Wisdom of Crowds differences between large and small datasets and how it performs. These are hardcoded in to appear as recommended values should you run my application, however I will list them for reference as well.

Population Size = 100

Max Generations = 150

Number of Runs = 5

Mutation Rate = 0.10

Elite Size = 3

Tournament Size = 9

Crossover Rate = 0.70

Crossover Method = Edge Recombination

Mutation Method = Inverse

The two graphs of every application iteration are one that displays distance vs path. It displays the paths contributing to WOC as well as displays WOC and WOCGA. The other displays the actual path taken by all the points on the other graph along with their specific distance, the last two graphs every time on this graph are the WOC Standalone and WOC with GA. You will notice that WOCGA is Purple with Star’s for nodes and WOC Standalone is Green with X’s for nodes. I have paths contributing to WOCGA and WOC Standalone are Red with Diamond’s for nodes.

Here is my table with my Data, Results will contain the graphs for each of the following iterations.

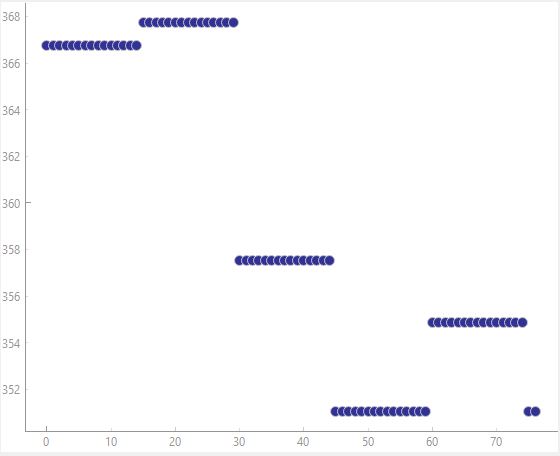
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 11 Nodes | 22 Nodes | 44 Nodes | 77 Nodes | 97 Nodes | 222 Nodes | 150 Nodes | 300 Nodes |
| Time Taken for WOCGA | 0.88 seconds | 1.3695 seconds | 2.75 seconds | 5.87 seconds | 8.29 seconds | 33.75 seconds | 18.15 seconds | 61.95 seconds |
| Time Taken for Best GA | 0.82 seconds | 1.328 seconds | 2.7 seconds | 5.78 seconds | 8.11 seconds | 33.46 seconds | 17.76 seconds | 60.95 seconds |
| WOC Initial Distance | 351.05 | 418.48 | 619.34 | 1376.088 | 1832.57 | 5265.935 | 4004.91 | 10062.14 |
| WOCGA Distance | 351.05 | 418.48 | 619.34 | 1376.088 | 1832.57 | 5265.935 | 4004.91 | 10062.14 |
| Best GA Distance | 351.05 | 418.72 | 666.84 | 1289.29 | 1845.69 | 5766.27 | 4141.19 | 11393.09 |

Only for my final test 500 nodes, will I change the parameters to give the user a chance to see what happens when given enough time to evaluate a large data set, as in keeping parameters the same for fairness in visualization between data sizes, has limited the capability of larger data sets to get closer to the optimal solution.

**Results – Results**

***11 Nodes***

Paths contributing to WOC Per Run VS Distance



WOC Standalone and WOCGA are the same path and distance.

A graph of a star

Description automatically generated

WOC Path: [1, 7, 3, 10, 8, 11, 2, 9, 5, 4, 6, 1]

WOC Path Distance: 351.04587994185

Best Path with WOC: [[1, 7, 3, 10, 8, 11, 2, 9, 5, 4, 6, 1]]

Best Distance with WOC [351.04587994185]

***22 Nodes***

Paths contributing to WOC Per Run VS Distance

A graph of a number of dots

Description automatically generated with medium confidence

WOC Standalone and WOCGA are the same path and distance.

A diagram of a star constellation

Description automatically generated

WOC Path: [1, 7, 15, 3, 16, 11, 21, 17, 8, 12, 6, 10, 13, 19, 4, 9, 14, 2, 18, 20, 22, 5, 1]

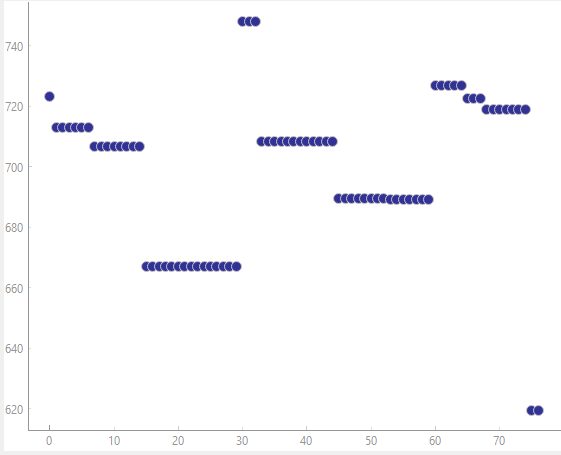
WOC Path Distance: 418.48049532263076

Best Path with WOC: [[1, 7, 15, 3, 16, 11, 21, 17, 8, 12, 6, 10, 13, 19, 4, 9, 14, 2, 18, 20, 22, 5, 1]]

Best Distance with WOC [418.48049532263076]

***44 Nodes***

Paths contributing to WOC Per Run VS Distance



WOC Standalone and WOCGA are the same path and distance.

A screenshot of a graph

Description automatically generated

WOC Path: [1, 13, 22, 28, 30, 20, 8, 6, 33, 4, 5, 42, 3, 38, 40, 10, 39, 35, 27, 7, 18, 23, 31, 26, 29, 9, 25, 11, 24, 34, 12, 19, 32, 17, 43, 44, 2, 14, 36, 21, 37, 41, 16, 15, 1]

WOC Path Distance: 619.34131037934

Best Path with WOC: [[1, 13, 22, 28, 30, 20, 8, 6, 33, 4, 5, 42, 3, 38, 40, 10, 39, 35, 27, 7, 18, 23, 31, 26, 29, 9, 25, 11, 24, 34, 12, 19, 32, 17, 43, 44, 2, 14, 36, 21, 37, 41, 16, 15, 1]]

Best Distance with WOC [619.34131037934]

***77 Nodes***

Paths contributing to WOC Per Run VS Distance

A diagram of dots

Description automatically generated

A screenshot of a graph

Description automatically generatedWOC Standalone and WOCGA are the same path and distance.

WOC Path: [1, 65, 45, 60, 7, 30, 20, 31, 24, 57, 53, 10, 35, 71, 46, 16, 22, 2, 29, 75, 61, 54, 69, 77, 44, 25, 14, 62, 63, 68, 74, 33, 47, 56, 9, 48, 66, 13, 4, 42, 21, 59, 6, 17, 32, 67, 50, 38, 70, 12, 8, 26, 15, 18, 72, 36, 49, 41, 27, 39, 64, 76, 37, 58, 43, 5, 3, 40, 11, 51, 23, 73, 55, 19, 52, 28, 34, 1]

WOC Path Distance: 1376.088368457406

A screenshot of a graph

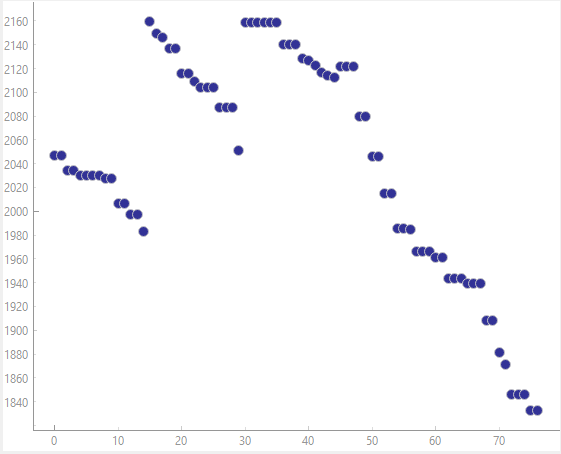
Description automatically generated Best GA Distance Below

Best Paths with WOC: [[1, 65, 45, 60, 7, 30, 20, 31, 24, 57, 53, 10, 35, 71, 46, 16, 22, 2, 29, 75, 61, 54, 69, 77, 44, 25, 14, 62, 63, 68, 74, 33, 47, 56, 9, 48, 66, 13, 4, 42, 21, 59, 6, 17, 32, 67, 50, 38, 70, 12, 8, 26, 15, 18, 72, 36, 49, 41, 27, 39, 64, 76, 37, 58, 43, 5, 3, 40, 11, 51, 23, 73, 55, 19, 52, 28, 34, 1]]

Best Distances with WOC [1376.088368457406]

***97 Nodes***

Paths contributing to WOC Per Run VS Distance.



WOC Standalone and WOCGA are the same path and distance.

A screenshot of a graph

Description automatically generated

Best GA Distance Below.

A graph of numbers and points

Description automatically generated

WOC Path: [1, 67, 5, 43, 68, 49, 64, 25, 32, 31, 57, 33, 30, 44, 63, 90, 16, 41, 84, 69, 37, 26, 17, 88, 13, 29, 7, 35, 18, 77, 3, 73, 9, 12, 95, 48, 38, 50, 40, 56, 11, 66, 8, 87, 92, 20, 71, 58, 59, 86, 15, 62, 96, 4, 85, 60, 81, 80, 61, 91, 28, 27, 10, 72, 6, 83, 79, 82, 21, 94, 93, 53, 34, 75, 24, 46, 47, 76, 65, 23, 89, 52, 42, 2, 45, 39, 78, 22, 70, 54, 55, 74, 14, 19, 51, 36, 97, 1]

WOC Path Distance: 1832.5733707866411

Best Paths with WOC: [[1, 67, 5, 43, 68, 49, 64, 25, 32, 31, 57, 33, 30, 44, 63, 90, 16, 41, 84, 69, 37, 26, 17, 88, 13, 29, 7, 35, 18, 77, 3, 73, 9, 12, 95, 48, 38, 50, 40, 56, 11, 66, 8, 87, 92, 20, 71, 58, 59, 86, 15, 62, 96, 4, 85, 60, 81, 80, 61, 91, 28, 27, 10, 72, 6, 83, 79, 82, 21, 94, 93, 53, 34, 75, 24, 46, 47, 76, 65, 23, 89, 52, 42, 2, 45, 39, 78, 22, 70, 54, 55, 74, 14, 19, 51, 36, 97, 1]]

Best Distances with WOC [1832.5733707866411]

***222 Nodes***

Paths contributing to WOC Per Run VS Distance.

A group of blue dots

Description automatically generated

WOC Standalone and WOCGA are the same path and distance.

A diagram of a network

Description automatically generated with medium confidence

Best GA Distance Below.

A screenshot of a graph

Description automatically generated

***150 Nodes – My Test 1***

Paths contributing to WOC Per Run VS Distance.

A diagram of dots

Description automatically generated

WOC Standalone and WOCGA are the same path and distance.

A screenshot of a graph

Description automatically generated

Best GA Distance Below.

A screenshot of a graph

Description automatically generated

WOC Path: [1, 72, 48, 110, 148, 22, 4, 122, 52, 25, 140, 117, 102, 86, 89, 97, 62, 30, 38, 123, 39, 29, 90, 7, 137, 44, 10, 103, 47, 112, 85, 71, 115, 113, 84, 6, 109, 132, 41, 24, 136, 101, 76, 70, 46, 15, 82, 143, 3, 53, 37, 14, 42, 129, 119, 134, 69, 96, 91, 121, 133, 18, 27, 2, 63, 16, 128, 125, 55, 56, 26, 88, 32, 94, 20, 135, 61, 124, 19, 77, 12, 60, 81, 150, 31, 68, 5, 33, 131, 130, 92, 35, 9, 144, 127, 149, 58, 145, 116, 104, 87, 17, 54, 51, 111, 34, 79, 65, 141, 49, 13, 139, 114, 120, 105, 107, 142, 64, 108, 118, 80, 23, 126, 11, 100, 28, 59, 83, 74, 93, 138, 78, 99, 98, 45, 106, 66, 147, 50, 67, 43, 21, 146, 8, 73, 36, 95, 75, 57, 40, 1]

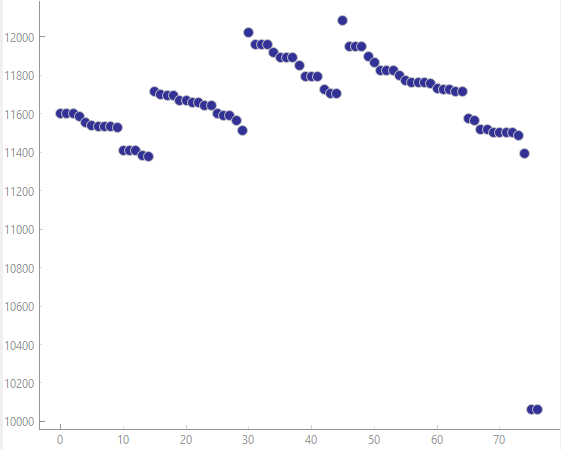
WOC Path Distance: 4004.910516290555

Best Paths with WOC: [[1, 72, 48, 110, 148, 22, 4, 122, 52, 25, 140, 117, 102, 86, 89, 97, 62, 30, 38, 123, 39, 29, 90, 7, 137, 44, 10, 103, 47, 112, 85, 71, 115, 113, 84, 6, 109, 132, 41, 24, 136, 101, 76, 70, 46, 15, 82, 143, 3, 53, 37, 14, 42, 129, 119, 134, 69, 96, 91, 121, 133, 18, 27, 2, 63, 16, 128, 125, 55, 56, 26, 88, 32, 94, 20, 135, 61, 124, 19, 77, 12, 60, 81, 150, 31, 68, 5, 33, 131, 130, 92, 35, 9, 144, 127, 149, 58, 145, 116, 104, 87, 17, 54, 51, 111, 34, 79, 65, 141, 49, 13, 139, 114, 120, 105, 107, 142, 64, 108, 118, 80, 23, 126, 11, 100, 28, 59, 83, 74, 93, 138, 78, 99, 98, 45, 106, 66, 147, 50, 67, 43, 21, 146, 8, 73, 36, 95, 75, 57, 40, 1]]

Best Distances with WOC [4004.910516290555]

***300 Nodes – My Test 2***

Paths contributing to WOC Per Run VS Distance.



WOC Standalone and WOCGA are the same path and distance.

A green lines and dots on a black background

Description automatically generated

This test follows most of the activity the other TSP problems have had after a certain threshold of nodes, this being that the WOC and WOCGA drastically increases over the GA paths.

***500 Nodes – My Test 3 – Changed Parameters***

For this final test I wanted to increase the parameters to show how the WOC algorithm behaves when given more time to look for a solution.

For this problem I am doing the following parameters.

Population Size = 300

Max Generations = 250

Number of Runs = 5

Mutation Rate = 0.10

Elite Size = 9

Tournament Size = 27

Crossover Rate = 0.70

Crossover Method = Edge Recombination

Mutation Method = Inverse

Distance VS Path Contribution to WOCGA

A graph showing a line of blue dots

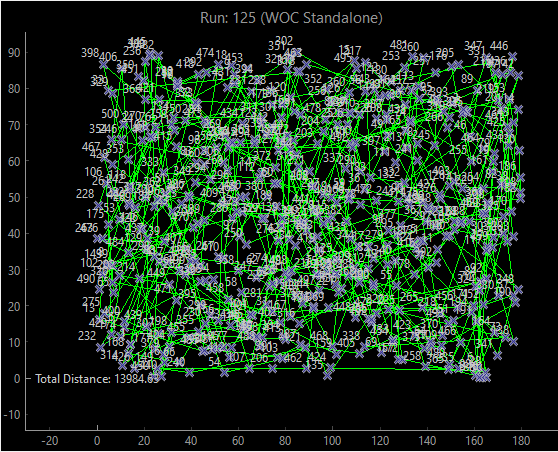
Description automatically generated

Best GA Run

A screenshot of a graph

Description automatically generated

Wisdom of Crowds with GA



**Above the distance improved by over 2000 distance.**

**Discussion**

My Wisdom of Crowds is very basic in that I have an edge frequency matrix that counts how often each edge appears in paths, it then creates a path starting from 0, and iterating through until it has filled all spots, and it fills them with the unvisited city that has the highest edge frequency from the current city, and only if it’s a tie does it choose the city closest in terms of distance. It then adjusts the city numbers back to their original. This means that my algorithm basically always satisfies TSP because my original paths being passed to this already satisfy TSP. So, I had no need for a greedy algorithm, which also means while mine is more traditional, it may not perform as well as other students.

On average, I do believe Wisdom of Crowds performs better than the typical GA. However, this is conditional with time, as well as size. For very small datasets, where the optimal can be found by the GA, the Wisdom of Crowds approach sometimes performs worse than the GA. As well as for very large datasets, while it does almost always find better paths than the GA, the path is still nowhere near optimal. However, considering the time cost of Wisdom of Crowds, I think it is absolutely a great algorithm to use along with GA, as it may provide more optimal paths depending on what you are doing. It is also very fun to put together, and I think anyone can benefit from learning how to use it.

**References**

I did not really have any references, I did ask some students how they were structuring their algorithm but did not end up using any of those ideas.