**Homework 2**

Ali Abbas

Ahsan Sanaullah

Fernando Trevino Ramirez

**Abstract**

In this paper, we explore how different representations on a Genetic Algorithm (GA) can affect its performance.

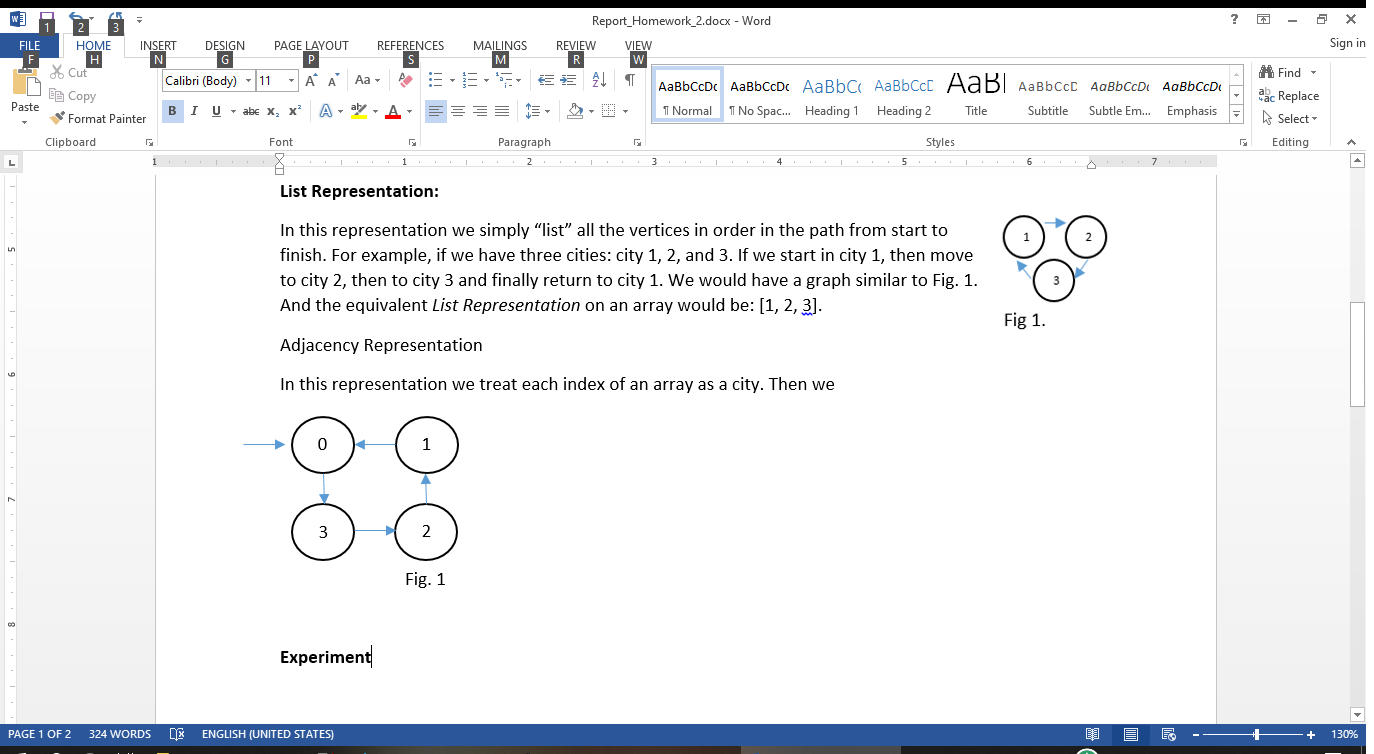
**Introduction**

To test the importance of choosing the correct representation to a problem intended to be solved with a GA, we first start by finding a suitable problem. The problem assigned by our professor Annie Wu [1] was the famous Traveling Salesman Problem or TSP for short. Then we were allowed to choose two different representations: List and Ordinal were the representations we decided to use for this project [3]. Since we were given the coordinate point for each of the cities, we assumed that all cities are connected and thus we initialized a 2 dimensional array with the distance between each vertex at the beginning of each experiment. Where distance d is defined as:

**Traveling Salesman Problem (TSP) review:**

Given a number of cities n, find a path that goes through all the cities exactly one time and ends in the first city. In other words, all cities must be visited and the graph must contain exactly 1 cycle. If any of this conditions isn’t met, then the solution is considered to be invalid. For more information we recommend a book we found called “The Traveling Salesman Problem” [2].

**List/Path Representation:**

In this representation we simply “list” all the vertices in order in the path from start to finish. For example, if we have four cities: city 0, 1, 2, and 3. If we start in city 0, then move to city 3, then to city 2, then 1, and finally return to city 0. We would have a graph similar to Fig. 1. And the equivalent *List Representation* on an array would be: [0, 3, 2, 1, 0]. But since we always return to the city we started, we can omit the last 0, leaving us with this array: [0, 3, 2, 1].

**Ordinal Representation:**

For this representation is easier to explain how to go from list to ordinal. We start by choosing a valid array as a list. For example list A = [0, 3, 2, 1] (Fig. 1), then starting with the list B = [0, 1, 2, 3], we decide the value of the position using the position of the city in the remaining of list B. In other words, the value at 0 is 0 since city 0 is in position 0 in the list B. Then, 0 is removed from the list B, producing [1, 2, 3]. The value at 1 is 2 since city 3 is in position 2 of the list B. 3 is removed producing [1, 2]. Following the same logic, the full ordinal representation of the path in Fig. 1 is [0, 2, 1, 0].

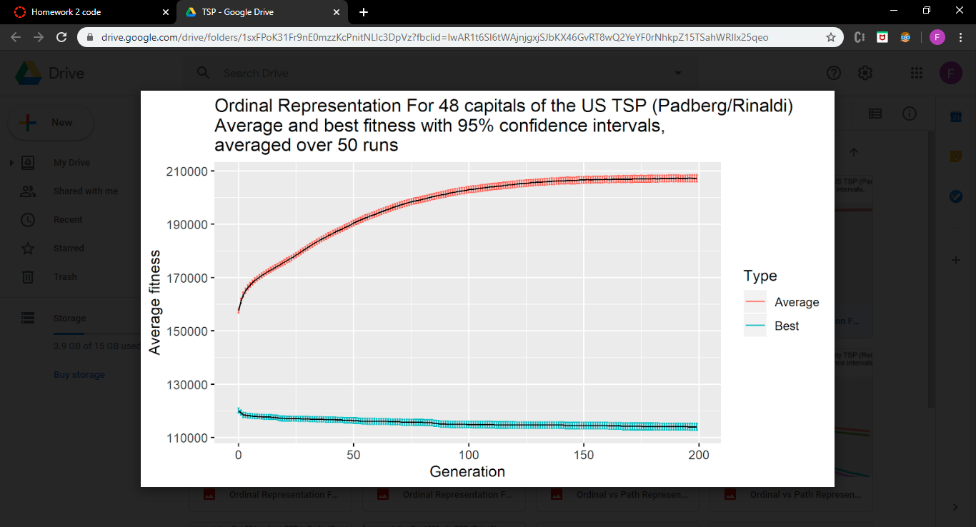
**Experiments**

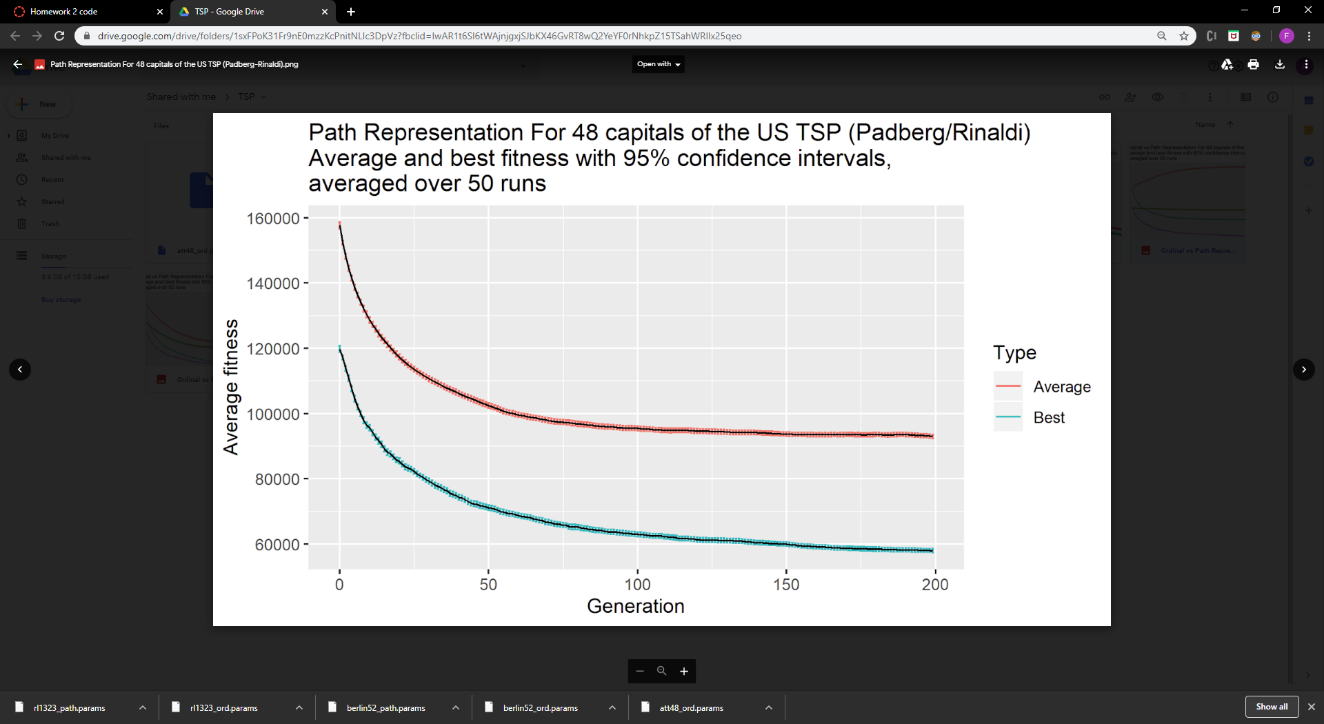
In this project we compare two approaches of solving the Traveling Salesman Problem with a GA, using two representations of a possible path solution.

For the list representation, we tried many different genetic operators. For crossover, we tried Order Crossover and Genetic Edge Recombination Crossover [3]. For mutation, we tried Insertion Mutation and Displacement Mutation. For order crossover, the order of a subset of cities of 1 parent is imposed on the other parent. The edge recombination operator “tries to preserve the edges of parents” [3]. The Insertion Mutation takes a random city and moves it to a random position in the path. The Displacement Mutation takes a random sized windows and moves it to another random position.

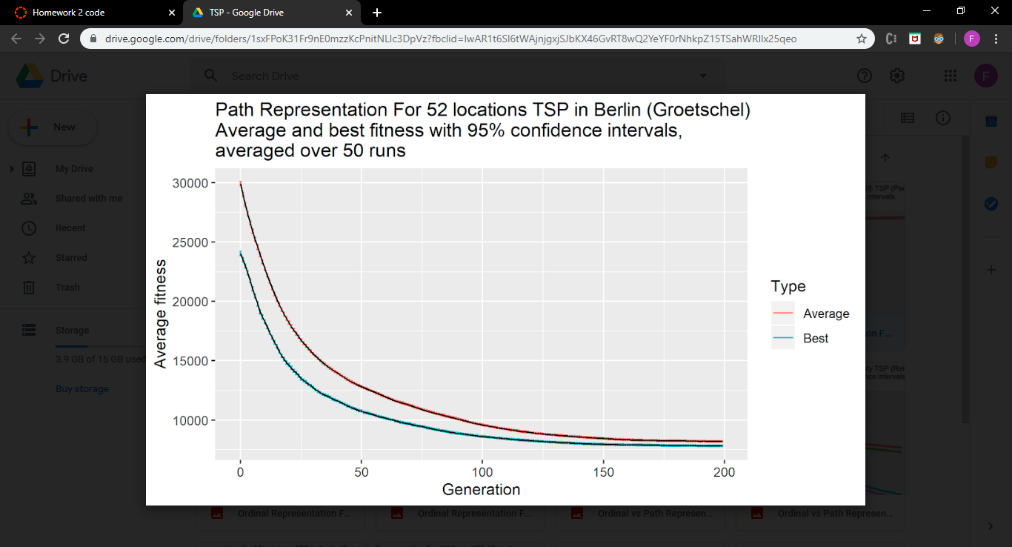
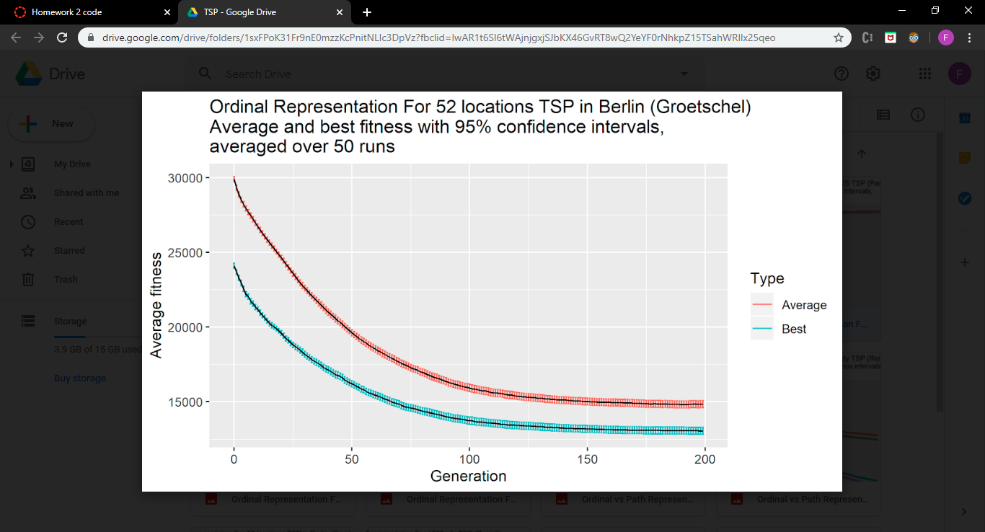
The advantage of ordinal representation is that it allows traditional operators to be used. We used one point crossover and mutation of a single index of the array. One point crossover is self-explanatory, it works in the same way as in binary representations. For mutation of a single index, a random index is chosen. Then, a random value in is chosen ( is the number of cities). This replaces the old value at in the ordinal representation. This always results in a valid representation.

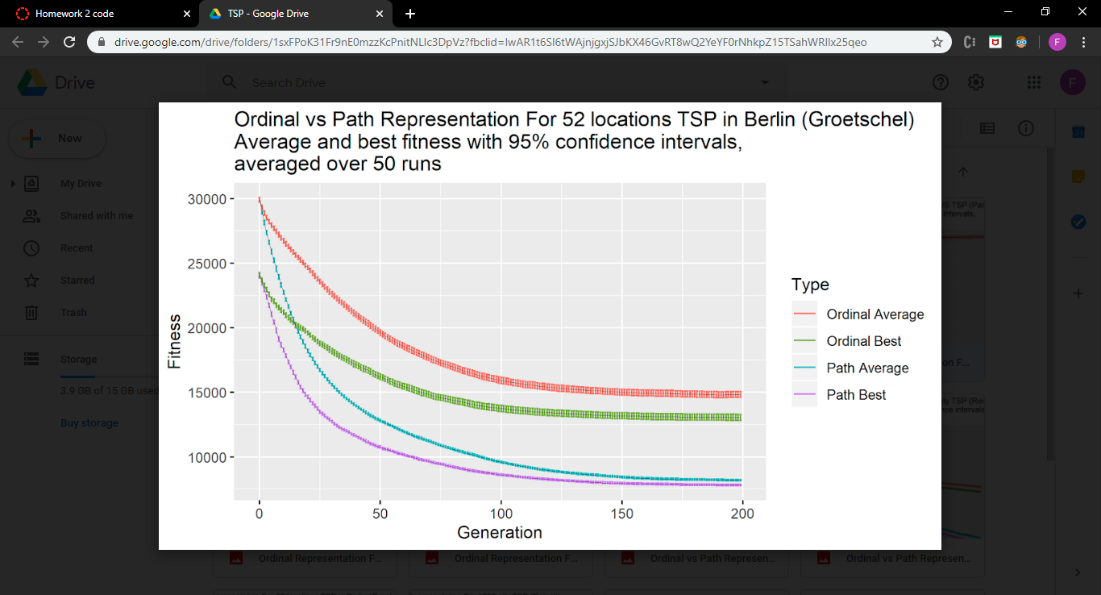
**Results**

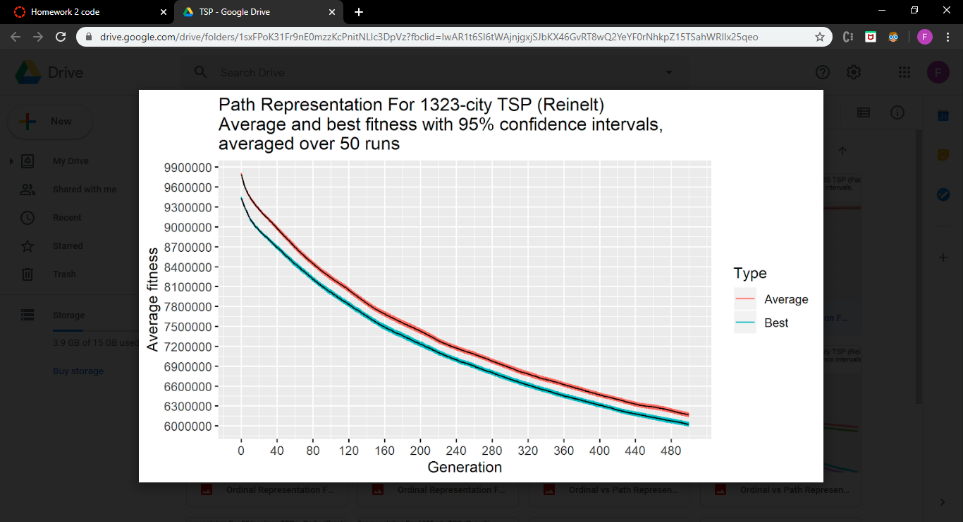
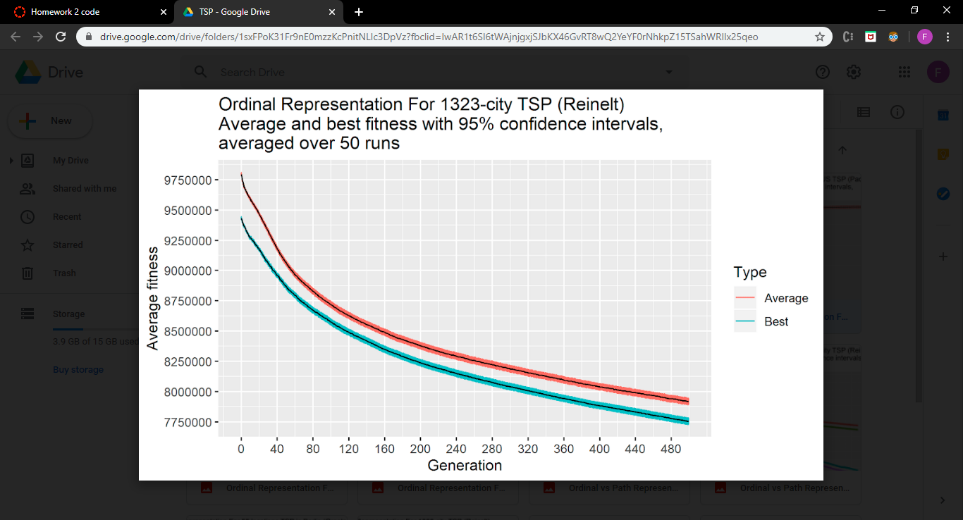
We did multiple runs to find optimal parameters for each dataset. For the Berlin 52 we used a population size of 2000 with 200 generations over 50 runs. And for rl1323 we had to reduce the population size to 500, but increased to 500 generations, and did 50 runs.

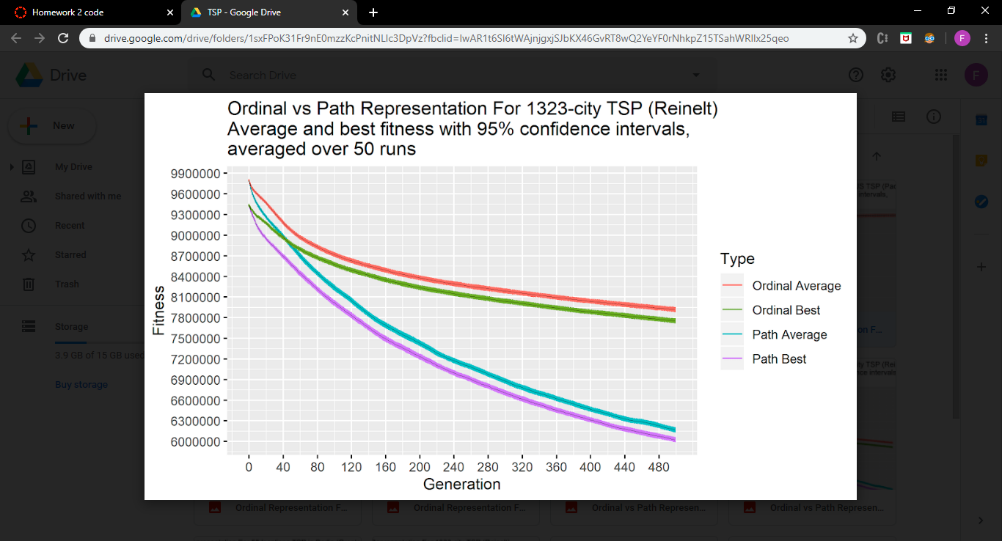








**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



As we can see from the previews graphs, Ordinal repreentation had slow low rate of improvement over time with the expetion of 48 capitals (Padber/Rinaldi) where the average actually got worst and the best is almost a horizontal line (No improvement). For the rest of the experiments we consistently got a better performance from the Path representation.

**Conclusion**

In conclusion, the Traveling Salesman Problem is a difficult problem that does not have a one size fits all solution, even with Genetic Algorithms. We found that the performance of GAs on Traveling Salesman Problems significantly depend on the input cities provided, different inputs will need different parameters and may even perform better with different problem representations. In general the path representation with genetic edge recombination and insertion mutation is a good genetic algorithm for medium sized problems. A GA with an extremely small population and many generations may perform well on problems with a large amount of cities. The ordinal representation searches the search space almost randomly, however, the GA proves it is not fully random. The operators do not make sense intuitively, however the results are not very poor. The path representation is very intuitive, however its crossover operators are complicated and computationally expensive. Each representation has tradeoffs. In general, we found that Genetic Algorithms are good for finding a pretty good solution quickly. Furthermore, the parameters and representation may need tuning to the specific input.

**Extensions**

By only testing the GA on the traveling salesman problem and only using two representations, we limit the generality of the results. To solidify our results we would need to: Experiment in a bigger range of problems. And expand the number of representations per problem.

**References**

[1] Hal Stringer & Annie Wu (2004). “A Teaching GA” [Computer software]. Florida, Orlando: UCF.

[2] Applegate, D. L.; Bixby, R. M.; Chvatal, V.; Cook, W. J. (2006), “The Traveling Salesman Problem”, ISBN 978-0-691-12993-8.

[3] Larranaga, P. (1999). “Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators”. Retrieved February 19, 2020, from <https://link.springer.com/article/10.1023/A:1006529012972>