**DSA Project Documentation**

**Saulat Jahan FA17-BCS-071**

**Muhammad Usama FA17-BCS-146**

**Introduction**

The Travelling Salesman Problem is a classic algorithmic problem in the field of computer science. The Travelling Salesman Problem describes a salesman who must travel between N cities irrespective of the order in which he does so as long as he visits each during his trip once, and finishes where he was at first. The salesman wants to keep the distance he travels as low as possible.

The easiest (and most expensive solution) is to simply try all possibilities. Exact solutions to the problem can be found using branch and bound algorithms, but we will be using genetic algorithm to solve this problem for up to 30 cities (which is 8.841762e+30 possible combinations).

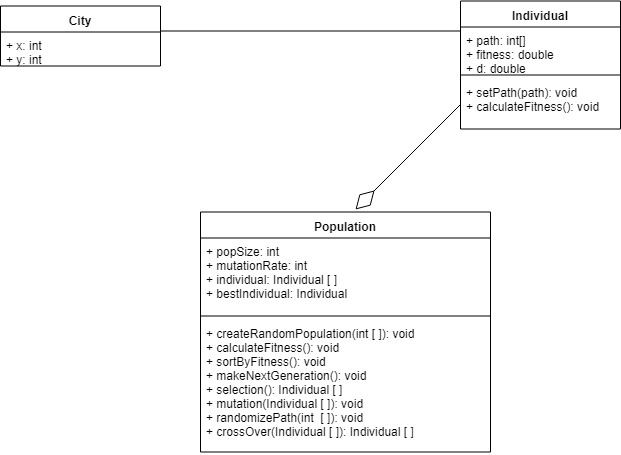
**Motivation and Scope**

The objective of this project is to show that how flexible genetic algorithms can be to solve most of the real world problems. To demonstrate that how neat a good written genetic algorithm can be and it’s accuracy with respect to the time taken is phenomenal.

You must use the best combination of population size, mutation rate and max generations in order to get the best result. The script is written in an organized way so that it can be easily modified in the future in order to adopt new crossover techniques and further improve the results.

**UML Diagram**

Members of classes and methods are public because there is no native support for private and protected members in JavaScript. We have to use Babel or some other library to do so. My code is mostly based on procedural programming whereas 3 files are class (object oriented) based.



**Time Complexity**

**Lexicographical order approach:**

Let,

n = number of cities

Time complexity = O (n \* n!)

**Genetic Algorithm:**

Let,

g = number of generations

p = population size

n = number of cities

m = mutation rate

O (g\*((p\*n) + (p^2) + (p/2) + (p\*n) + (p/2) + ((p\*m)/100)))

Now Simplifying

O (g\*((p\*n) + (p^2) + (p) + (p\*n)) + ((p\*m)/100)))

O (g\*((p\*n) + (p^2) + (p\*n)))

O (g\*((p\*n) + (p^2))).

**Comparison of Lexicographical approach VS Genetic Algorithm:**

In Lexicographical approach we need to find every combination which turns out to be pain in the back in the number of cities become more and more while genetic algorithm can solve this problem in very short amount of time

We can take an example of finding the best path between 10 cities. For Lexicographical approach, if we put the values in the formula above, the result is 36288000.

On the other hand we use genetic algorithm. Let’s consider 10 cities with 50 population size, 50 generations, mutation rate of 2, and then the result is 150000.

There is a huge difference between both approaches.