Project Report

Intelligent Transportation System

(Under Ground Subway System)

DATA MINING FALL 21

Semester Project

Muhammad Hammad (19I-0446)

Muneeb Bhalli (19I-0591)

Mirza Abdur Rahman (19I-0584)

Section : E

Submitted to: Sir Wasseem Shehzad

Department of Computer Science BS(CS)

FAST-NUCES Islamabad

Intelligent Transportation System

(Under Ground Subway System)

An intelligent transportation system (ITS) is an advanced application which aims to provide innovative services relating to different modes of transport and traffic management and enable users to be better informed and make safer, more coordinated, and 'smarter' use of transport networks.

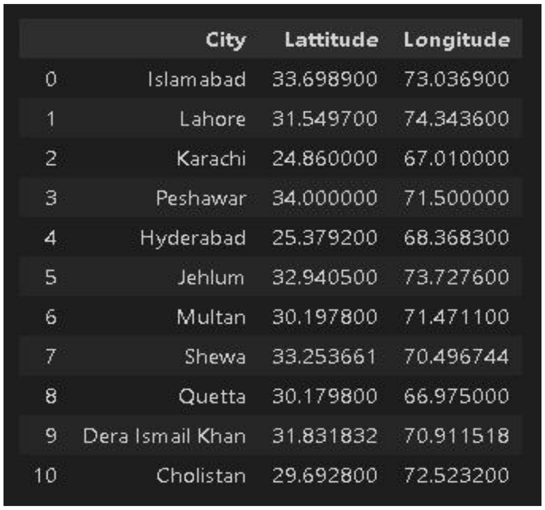
In ITS we are making a system for underground subway systems which predict the best optimal path from one station to other

Our Approach

For determining the best optimal path, we have used **Genetic Algorithm** in which we perform selection, fitness evaluation, crossover, mutation. After performing Genetic Algorithm, we also performed evaluation by using graph of the fitness values of first iteration and fitness value of the final iteration.

Raw Data

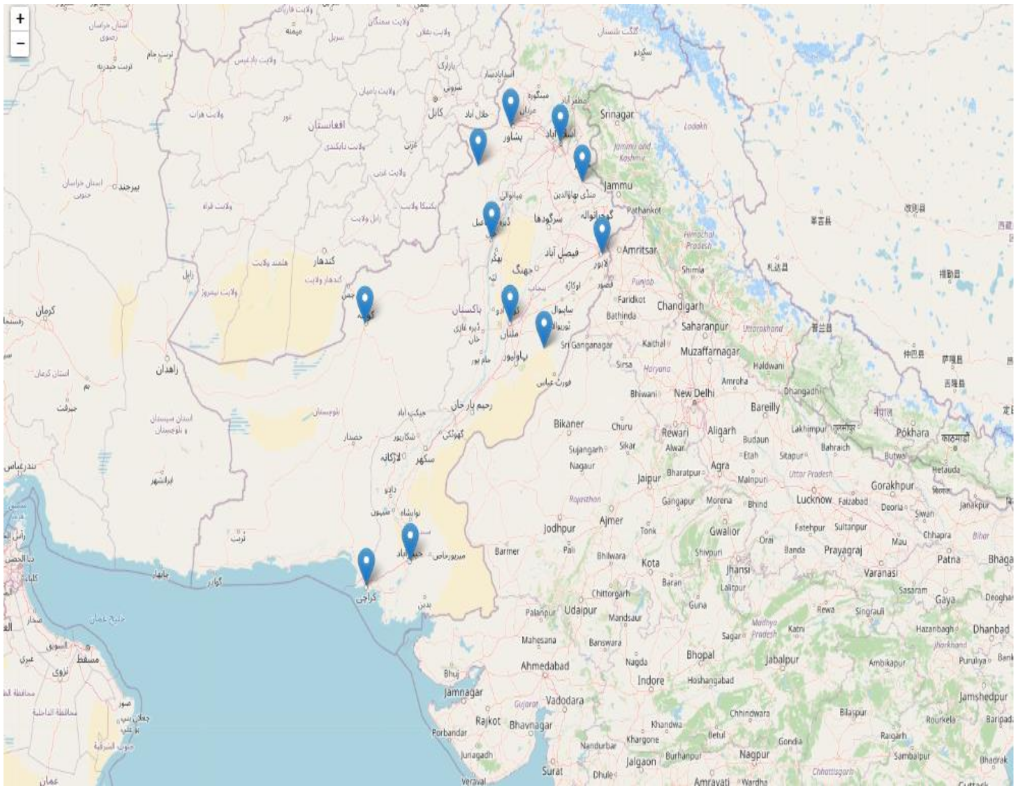
We have used a data set from data.csv which consists of the names of stations and their longitude and latitude values. After reading it by data frame:



	City	Lattitude	Longitude
0	Islamabad	33.698900	73.036900
1	Lahore	31.549700	74.343600
2	Karachi	24.860000	67.010000
3	Peshawar	34.000000	71.500000
4	Hyderabad	25.379200	68.368300
5	Jehlum	32.940500	73.727600
6	Multan	30.197800	71.471100
7	Shewa	33.253661	70.496744
8	Quetta	30.179800	66.975000
9	Dera Ismail Khan	31.831832	70.911518
10	Cholistan	29.692800	72.523200

Map Plotting

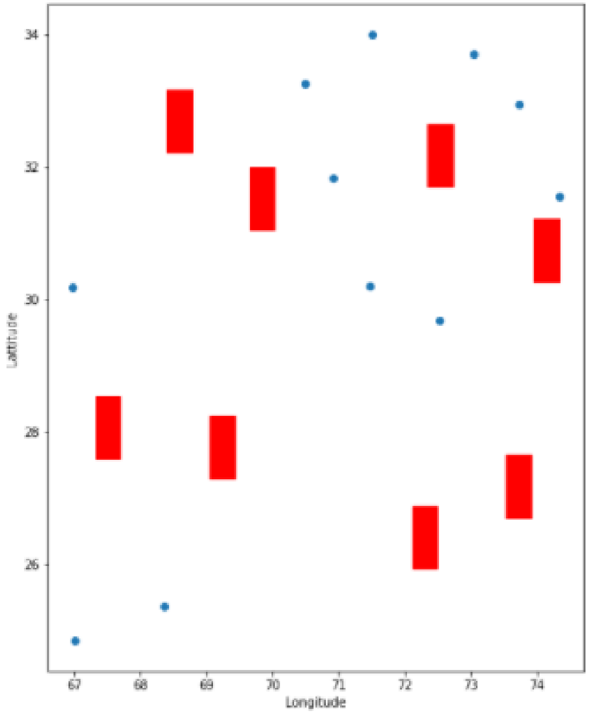
After reading it by data frame we plot those latitude and longitude values on map to show the exact location of stations. We have done this by using a function Create_map(df) which accepts the data frame as parameter. This function is plotting the latitude and longitude values on map by using folium library. The exact location of stations on map generated by Create_map(df) are



Plotting of Stations and Obstacles/Trains on Graph:

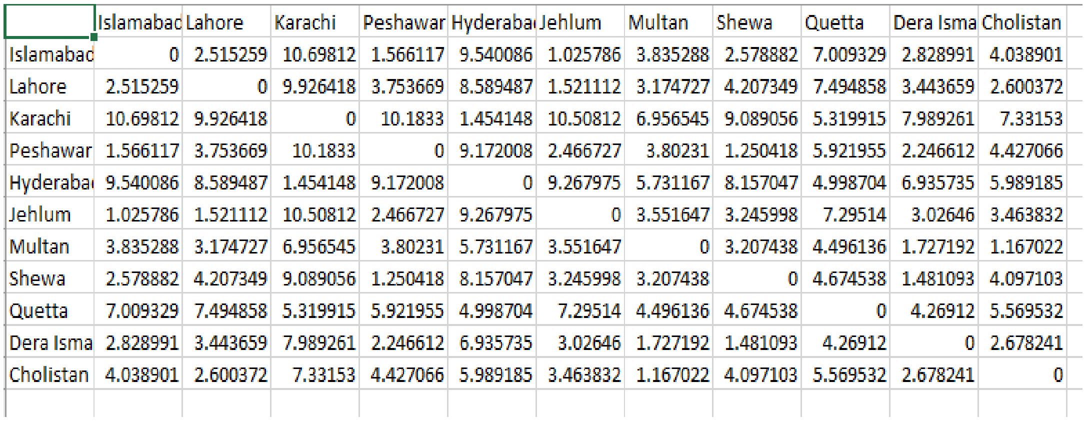
Then we have shown all the stations and obstacles on graph by using Create_Station_and_Obstacle(df) function. This function shows stations and obstacles on longitude-latitude plane. The stations are shown as blue dot points and obstacles are shown as red rectangles.

Note: Obstacle can be any train aur blockage on the route.



Distance between Stations

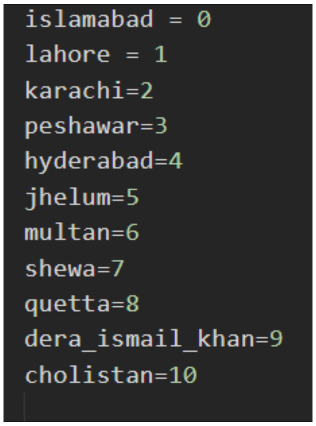
Then we have calculated the distance of every station from other every station. We have generated a distance matrix function by using Calculate_distance_Matrix(df). We write our distance matrix in a separate csv file named as dist_m.csv. After performing this function, the dist_m.csv will be



	Islamabac	Lahore	Karachi	Peshawar	Hyderaba	Jehlum	Multan	Shewa	Quetta	Dera Isma	Cholistan
Islamabad	0	2.515259	10.69812	1.566117	9.540086	1.025786	3.835288	2.578882	7.009329	2.828991	4.038901
Lahore	2.515259	0	9.926418	3.753669	8.589487	1.521112	3.174727	4.207349	7.494858	3.443659	2.600372
Karachi	10.69812	9.926418	0	10.1833	1.454148	10.50812	6.956545	9.089056	5.319915	7.989261	7.33153
Peshawar	1.566117	3.753669	10.1833	0	9.172008	2.466727	3.80231	1.250418	5.921955	2.246612	4.427066
Hyderaba	9.540086	8.589487	1.454148	9.172008	0	9.267975	5.731167	8.157047	4.998704	6.935735	5.989185
Jehlum	1.025786	1.521112	10.50812	2.466727	9.267975	0	3.551647	3.245998	7.29514	3.02646	3.463832
Multan	3.835288	3.174727	6.956545	3.80231	5.731167	3.551647	0	3.207438	4.496136	1.727192	1.167022
Shewa	2.578882	4.207349	9.089056	1.250418	8.157047	3.245998	3.207438	0	4.674538	1.481093	4.097103
Quetta	7.009329	7.494858	5.319915	5.921955	4.998704	7.29514	4.496136	4.674538	0	4.26912	5.569532
Dera Isma	2.828991	3.443659	7.989261	2.246612	6.935735	3.02646	1.727192	1.481093	4.26912	0	2.678241
Cholistan	4.038901	2.600372	7.33153	4.427066	5.989185	3.463832	1.167022	4.097103	5.569532	2.678241	0

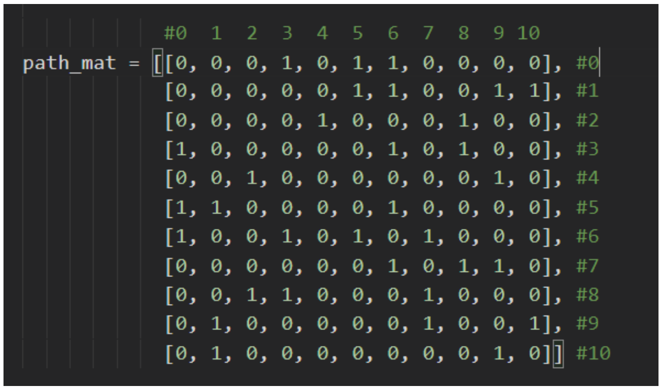
Adjacent Stations

Then we have described the adjacent stations by making adjacency matrix. We have also given values to our cities from 0 to 10 as it will be used further



```
islamabad = 0
lahore = 1
karachi=2
peshawar=3
hyderabad=4
jhelum=5
multan=6
shewa=7
quetta=8
dera_ismail_khan=9
cholistan=10
```

Adjacency Matrix

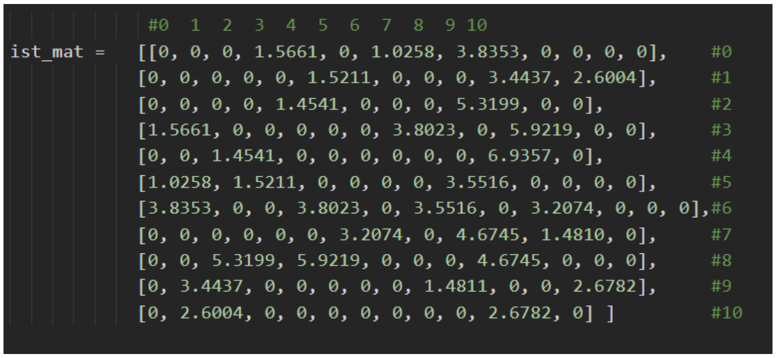


```
              #0  1  2  3  4  5  6  7  8  9 10
path_mat = [[0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0], #0
            [0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1], #1
            [0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0], #2
            [1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0], #3
            [0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0], #4
            [1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], #5
            [1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0], #6
            [0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0], #7
            [0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0], #8
            [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1], #9
            [0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0]] #10
```

Here 0 represents that it is not adjacent and 1 represents that it is adjacent.

Weighted Adjacency Matrix

We have created a weighted adjacency matrix in which we have given weights to adjacent stations. We have used the distance matrix to determine the weights of neighbor stations



```
            #0  1  2  3  4  5  6  7  8  9 10
ist_mat =   [[0, 0, 0, 1.5661, 0, 1.0258, 3.8353, 0, 0, 0, 0],     #0
            [0, 0, 0, 0, 0, 1.5211, 0, 0, 0, 3.4437, 2.6004],      #1
            [0, 0, 0, 0, 1.4541, 0, 0, 0, 5.3199, 0, 0],           #2
            [1.5661, 0, 0, 0, 0, 0, 3.8023, 0, 5.9219, 0, 0],      #3
            [0, 0, 1.4541, 0, 0, 0, 0, 0, 0, 6.9357, 0],           #4
            [1.0258, 1.5211, 0, 0, 0, 0, 3.5516, 0, 0, 0, 0],      #5
            [3.8353, 0, 0, 3.8023, 0, 3.5516, 0, 3.2074, 0, 0, 0],#6
            [0, 0, 0, 0, 0, 0, 3.2074, 0, 4.6745, 1.4810, 0],      #7
            [0, 0, 5.3199, 5.9219, 0, 0, 0, 4.6745, 0, 0, 0],      #8
            [0, 3.4437, 0, 0, 0, 0, 0, 1.4811, 0, 0, 2.6782],      #9
            [0, 2.6004, 0, 0, 0, 0, 0, 0, 0, 2.6782, 0] ]          #10
```

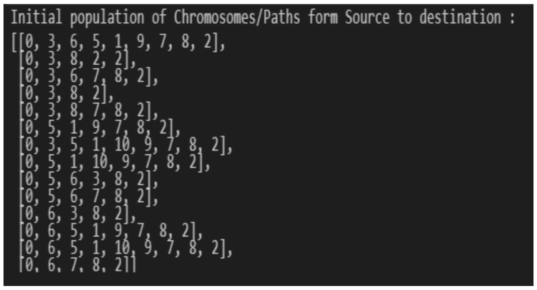
Adjacency Edge Graph

Then we have created adjacency edge graph by using Class Graph. This class create the adjacency edge graph and also gives all paths from source to destination. We give the values of our source and destination as previously defined to function of this class named as printAllPaths(source, destination). This function returns all the paths from source to destination

Working of Genetic Algorithm

Test Data to run the Genetic Algorithm

To run the whole genetic algorithm, we have taken Islamabad (0) as source and Karachi (2) as destination. We then used printAllPaths (source, destination) of class Graph to know all paths from Islamabad to Karachi. These paths are all chromosomes. This is our population. We will use this population for further processing



```
Initial population of Chromosomes/Paths form Source to destination :
[[0, 3, 6, 5, 1, 9, 7, 8, 2],
 [0, 3, 8, 2, 2],
 [0, 3, 6, 7, 8, 2],
 [0, 3, 8, 2],
 [0, 3, 8, 7, 8, 2],
 [0, 5, 1, 9, 7, 8, 2],
 [0, 3, 5, 1, 10, 9, 7, 8, 2],
 [0, 5, 1, 10, 9, 7, 8, 2],
 [0, 5, 6, 3, 8, 2],
 [0, 5, 6, 7, 8, 2],
 [0, 6, 3, 8, 2],
 [0, 6, 5, 1, 9, 7, 8, 2],
 [0, 6, 5, 1, 10, 9, 7, 8, 2],
 [0, 6, 7, 8, 2]]
```

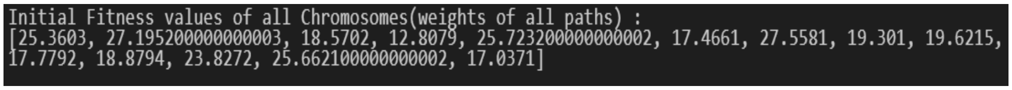
Looping Condition

We have iterated our Genetic Algorithm 50 times to get the optimal path and fitness values (weights of paths).

Fitness Evaluation

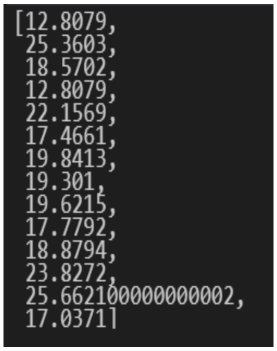
Now we will perform the fitness evaluation of our population by using Calculate_fitness(adjancecy_list). This function will calculate the weight of every path.

The weight of every path after first iteration will be



```
Initial Fitness values of all Chromosomes(weights of all paths) :
[25.3603, 27.195200000000003, 18.5702, 12.8079, 25.723200000000002, 17.4661, 27.5581, 19.301, 19.6215,
17.7792, 18.8794, 23.8272, 25.662100000000002, 17.0371]
```

The weight of every path after last iteration will be

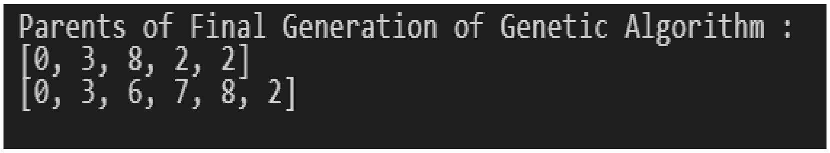


```
[12.8079,
 25.3603,
 18.5702,
 12.8079,
 22.1569,
 17.4661,
 19.8413,
 19.301,
 19.6215,
 17.7792,
 18.8794,
 23.8272,
 25.662100000000002,
 17.0371]
```

Parents Determination

Then we determine Parents by using Calculate_Parents(fitness_list). This function takes argument the weighted list. The first parent will have the best fitness value and the second parent will be the second-best fitness value. The best fitness value is the minimum value of the weighted list and the second-best fitness value of weighted list.

So, after final iterations our parents will be



```
Parents of Final Generation of Genetic Algorithm :
[0, 3, 8, 2, 2]
[0, 3, 6, 7, 8, 2]
```

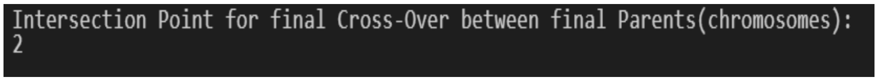
Crossover

Now we will perform single point crossover of these parents but first we have to find a crossover point.

Crossover Point

The cross over point will be the common point in both parents/chromosomes. We get this point by using function Calculate_Crossover_Intersection_Point(Parent1, Parent 2). If it returns more then 1 common points then we will randomly choose one common point. If there are no common points, we will randomly choose a point as an intersection point from first parent.

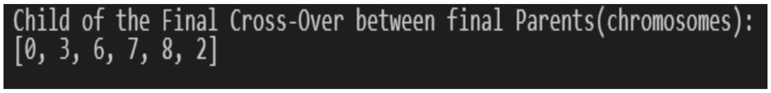
The intersection point after final iteration will be



```
Intersection Point for final Cross-Over between final Parents(chromosomes):
2
```

Single Point Crossover

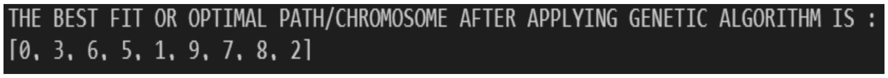
Now we will perform single point crossover by using function Single_Point_crossover(Parent1 ,Parent 2 , Crossover point). This function will perform Single point cross over and return single child. The child after final crossover will be



```
Child of the Final Cross-Over between final Parents(chromosomes):
[0, 3, 6, 7, 8, 2]
```

Best Optimal Path

So, our best optimal path after performing all these steps at final generation will be

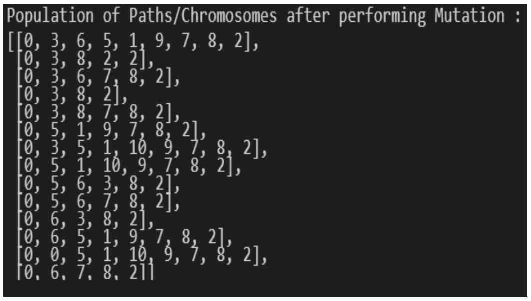


```
THE BEST FIT OR OPTIMAL PATH/CHROMOSOME AFTER APPLYING GENETIC ALGORITHM IS :
[0, 3, 6, 5, 1, 9, 7, 8, 2]
```

So, this is the best path from Islamabad(0) to Karachi(2).

Mutation

We have also shown the working of mutation if needed because sometimes we have to change our unmutated chromosome to get the best optimal solution as unmutated chromosome can lack some important genes. For this purpose, we use mutation(unmutated chromosome). If we use this function it will change mutate our population.



```
Population of Paths/Chromosomes after performing Mutation :
[[0, 3, 6, 5, 1, 9, 7, 8, 2],
 [0, 3, 8, 2, 2],
 [0, 3, 6, 7, 8, 2],
 [0, 3, 8, 2],
 [0, 3, 8, 7, 8, 2],
 [0, 5, 1, 9, 7, 8, 2],
 [0, 3, 5, 1, 10, 9, 7, 8, 2],
 [0, 5, 1, 10, 9, 7, 8, 2],
 [0, 5, 6, 3, 8, 2],
 [0, 5, 6, 7, 8, 2],
 [0, 6, 3, 8, 2],
 [0, 6, 5, 1, 9, 7, 8, 2],
 [0, 0, 5, 1, 10, 9, 7, 8, 2],
 [0, 6, 7, 8, 2]]
```

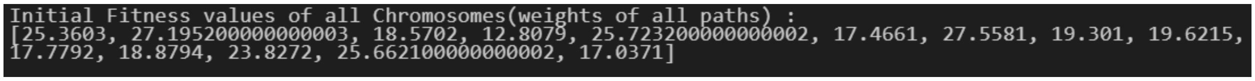
Evaluation of our Genetic Algorithm

Fitness Evaluation

Now we will evaluate our genetic algorithm based on the fitness values (weight of paths)

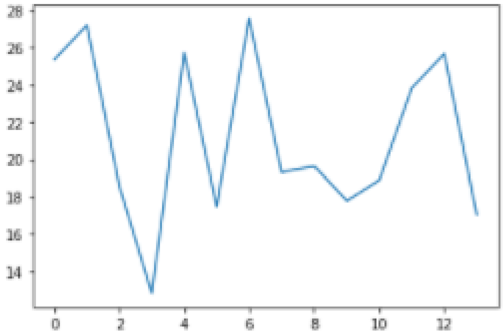
Fitness at first iteration of Genetic Algorithm

Values:



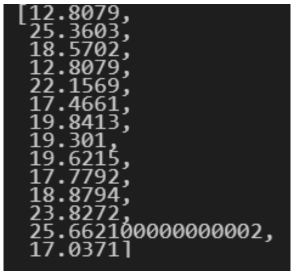
```
Initial Fitness values of all Chromosomes(weights of all paths) :
[25.3603, 27.195200000000003, 18.5702, 12.8079, 25.723200000000002, 17.4661, 27.5581, 19.301, 19.6215,
17.7792, 18.8794, 23.8272, 25.662100000000002, 17.0371]
```

Graph:



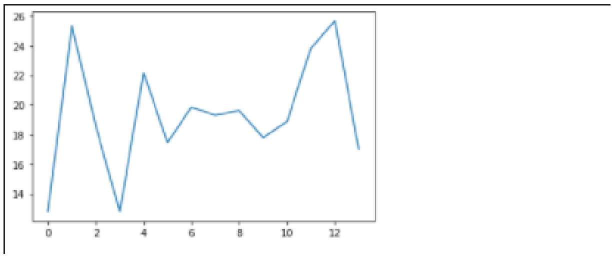
Fitness after last iteration of Genetic Algorithm

Values:



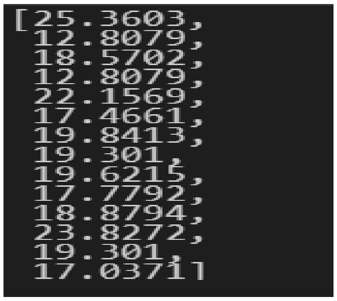
```
[12.8079,
 25.3603,
 18.5702,
 12.8079,
 22.1569,
 17.4661,
 19.8413,
 19.301,
 19.6215,
 17.7792,
 18.8794,
 23.8272,
 25.662100000000002,
 17.0371]
```

Graph:



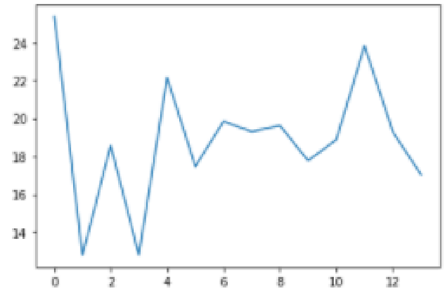
Fitness after mutation

Values



```
[25.3603,
 12.8079,
 18.5702,
 12.8079,
 22.1569,
 17.4661,
 19.8413,
 19.301,
 19.6215,
 17.7792,
 18.8794,
 23.8272,
 19.301,
 17.0371]
```

Graph



Mean of fitness values at first iteration

```
21.199178571428572
```

Mean of fitness values at last iteration

```
19.365578571428 57
```

Mean of fitness values after mutation

```
18.911214285714287
```

Conclusion

As we can see that the fitness values i.e. weights of paths are becoming optimal after every iteration. At first iteration the mean fitness value is 21.19. After last iteration our mean fitness value is 19.36 but after performing mutation, it reduces to 18.9. So, it clearly shows that our genetic algorithm is heading towards the best optimal path and our mean value of fitness of our population is improving after every generation.