ALGORITHM AND PROBLEM SOLVING LAB

PROJECT REPORT
APPLICATIONS OF GENETIC ALGORITHM

Introduction

This project is based on **GENETIC ALGORITHMS** and their applications.Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomised, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution, especially those follow the principles first laid down by Charles Darwin of "survival of the fittest.". Since in nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weaker ones.

The Two Applications of GA

CARGO SHIP LOADING:

Cargo ship load is a real problem when you have huge amount of cargo to be loaded to a ship with a specific load capacity.Solution to this problem is to take maximum worth of load to be shipped with minimal weight i.e minimum weight and maximum capacity.Either a cargo is loaded or not loaded.It can't be loaded partially.

TRAVELLING SALESMAN PROBLEM:

The Travelling Salesman Problem or the TSP is a representative of a large class of problems known as combinatorial optimization problems. In the ordinary form of the TSP, a map of cities is given to the salesman and he has to visit all the cities only once to complete a tour such that the length of the tour is the shortest among all possible tours for this map.

Applications of the Project

CARGO SHIP LOADING:

It can be used in real life where certain amount of cargo are to be shipped having certain worth.

TRAVELLING SALESMAN PROBLEM:

1. Arranging school bus routes to pick up the children in a school district.
2. A classic example is the scheduling of a machine to drill holes in a circuit board or other object. In this case the holes to be drilled are the cities, and the cost of travel is the time it takes to move the drill head from one hole to the next. The technology for drilling varies from one industry to another, but whenever the travel time of the drilling device is a significant portion of the overall manufacturing process then the TSP can play a role in reducing costs.
3. Routing of trucks for parcel post pickup.
4. Delivery of meals to homebound people
5. Scheduling of stacker cranes in warehouses.

Algorithms used in the project :

CARGO SHIP LOADING & TRAVELLING SALESMAN PROBLEM :

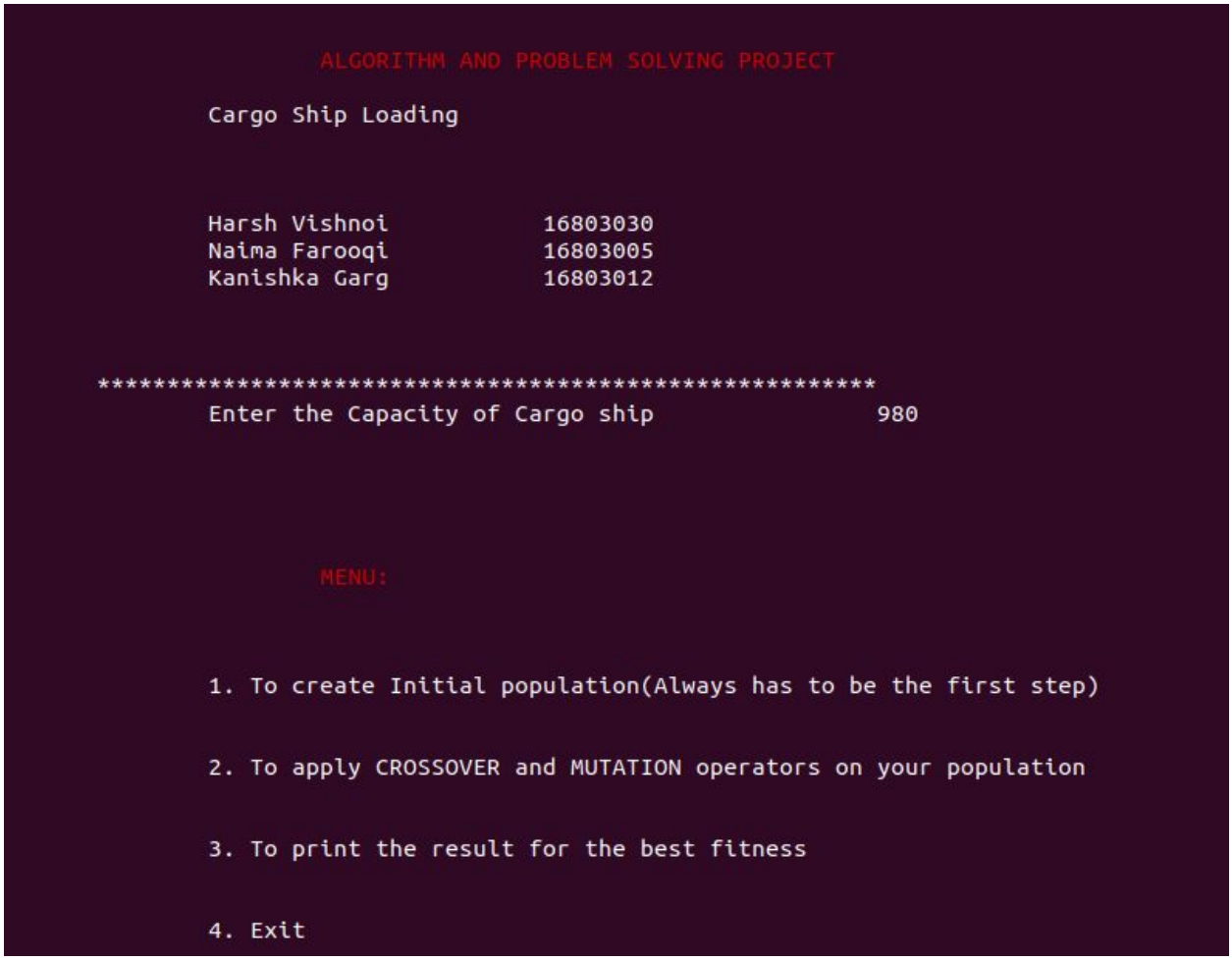
Genetic algorithm is used in this project.

1. Initialization of population
2. Fitness
3. Crossover
4. Mutation

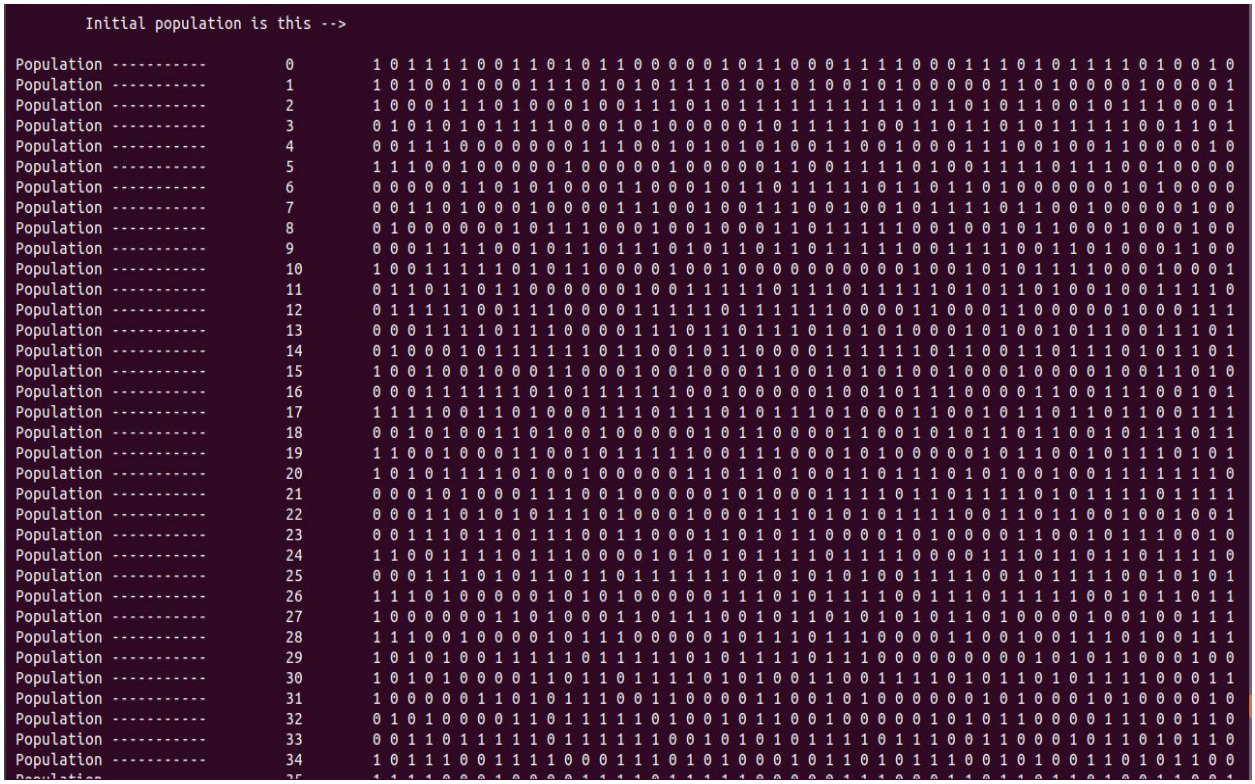
Results and Screenshots of the output

CARGO SHIP LOADING:

In result we get a menu driven program which first asks the capacity of the cargo ship and then displays the menu



Pressing 1 creates our initial population. 1 represent the the weight being present in the bag and 0 denotes it's absence.



Now, printing it's fitness table. Here the total benefit associated with each bag denote it's fitness.

```
    Initial fitness of population is as follows -->


Index         Weights      Benefits

    0             909          1502
    1             703          1507
    2             842          1631
    3             610           942
    4             564           849
    5             909          1502
    6             892          2041
    7             909          1502
    8             770          1378
    9             610           942
   10             770          1378
   11             610           942
   12             877          1402
   13             610           942
   14             888          1724
   15             958          1454
   16             703          1507
   17             842          1631
   18             610           942
   19             842          1631
   20             888          1724
   21             610           942
   22             610           942
   23             901          1900
   24             610           942
   25             773          1237
   26             610           942
   27             904          1759
   28             904          1759
   29             820          1562
```

Pressing 2 applies the crossover and mutation operator on the already created population.

```
Improving Generation ------------       1
Improving Generation ------------       2
Improving Generation ------------       3
Improving Generation ------------       4
Improving Generation ------------       5
Improving Generation ------------       6
Improving Generation ------------       7
Improving Generation ------------       8
Improving Generation ------------       9
Improving Generation ------------       10
Improving Generation ------------       11
Improving Generation ------------       12
Improving Generation ------------       13
Improving Generation ------------       14
Improving Generation ------------       15
Improving Generation ------------       16
Improving Generation ------------       17
Improving Generation ------------       18
Improving Generation ------------       19
```

```
After 100 generations, data seems to be as following



Final population is this -->

Population -----------    0     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    1     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    2     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    3     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    4     0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    5     0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    6     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    7     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    8     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    9     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    10    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    11    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    12    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    13    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    14    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    15    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    16    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    17    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    18    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    19    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    20    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    21    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    22    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    23    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    24    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    25    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    26    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Population -----------    27    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
Population -----------    28    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0
Population -----------    29    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0
```

Final fitness of population is as follows -->

Index	Weights	Benefits
0	940	2150
1	940	2150
2	940	2150
3	892	2041
4	978	2049
5	978	2049
6	892	2041
7	940	2150
8	892	2041
9	892	2041
10	940	2150
11	940	2150
12	940	2150
13	892	2041
14	940	2150
15	940	2150
16	940	2150
17	940	2150
18	892	2041
19	978	2049
20	940	2150
21	892	2041
22	892	2041
23	940	2150
24	892	2041
25	892	2041
26	892	2041
27	940	2150
28	739	1834
29	901	1900
30	901	1900
31	901	1900
32	901	1900
33	969	2190

Pressing 3 gives us our final result where 1 shows the weights to be included from our input and also prints our total weight and total benefit.

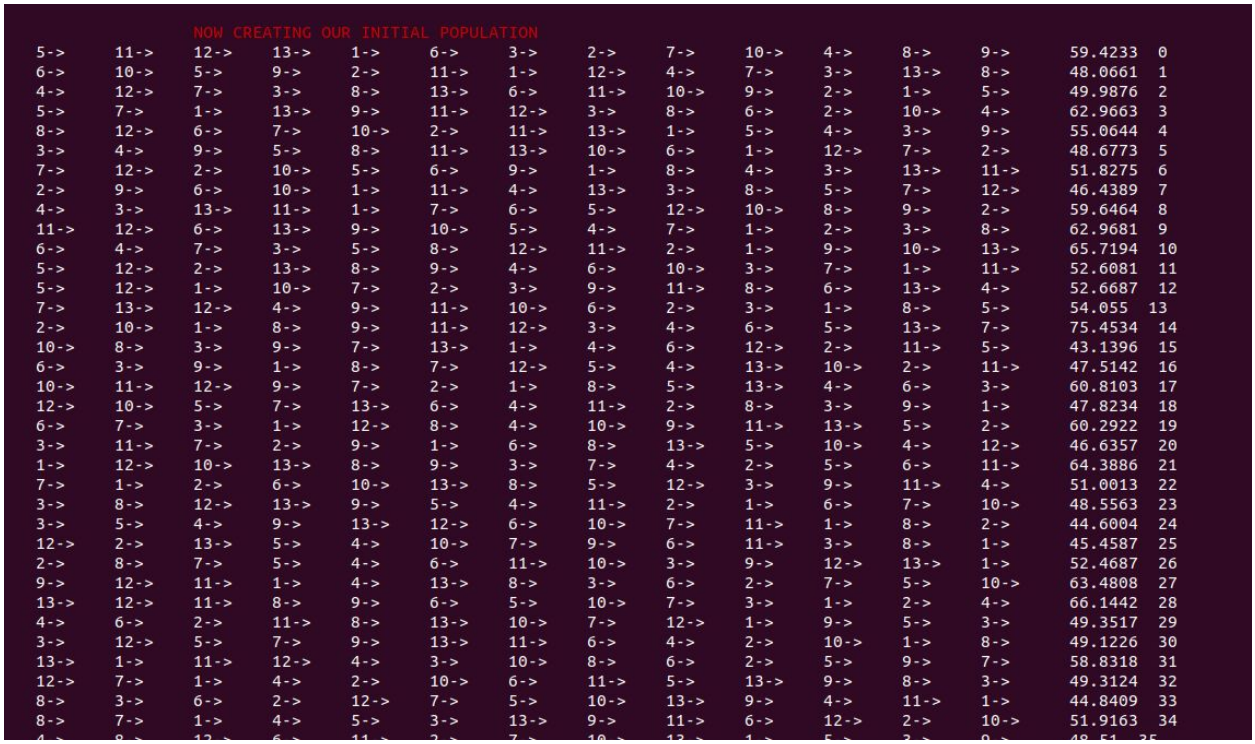
```
                RESULT:

 Best output is  -> 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 Index           -> 33
 Weight          -> 969
 Benefit         -> 2190

Cargo can contain atmost 969 tons of weight having an optimal worth of 2190 crores.
```

TRAVELLING SALESMAN PROBLEM:

In result we get a menu driven program which has a menu like this

```
                    ALGORITHM AND PROBLEM SOLVING PROJECT

              Travelling Salesman Problem by Genetic Algorithm


              Harsh Vishnoi           16803030
              Naima Farooqi           16803005
              Kanishka Garg           16803012


      ******************************************************


                       MENU:


              1. To create Initial population(ALways has to be the first step)

              2. To apply CROSSOVER and MUTATION operators on your population

              3. To print the result for the best fitness

              4. Exit
```

Pressing 1 creates our initial population. And outputs the population as the Index number of the city along with it's fitness and a serial number.



Pressing 2 works the crossover and mutation operators on our initial population giving us the 2 selected parents and the offspring as output. It then calculates the fitness of the offspring and replaces it with the lowest fitness person of our initial population.

```
        NOW AFTER APPLYING CROSSOVER AND MUTATION OPERATORS :
Parent 1 Index: 14 with fitness score: 75.4534
2-> 10-> 1-> 8-> 9-> 11-> 12-> 3-> 4-> 6-> 5-> 13-> 7->
Parent 2 Index: 67 with fitness score: 72.7687
10-> 3-> 11-> 13-> 6-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
 Their Offspring :
10-> 3-> 11-> 13-> 6-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
Now element with lowest fitness function will be removed i.e: 94 which was with fitness score 39.9967
Afer mutation the Offspring becomes:
10-> 11-> 3-> 13-> 6-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
and replaced with new fitness score 50.2884

---------------------------
Parent 1 Index: 14 with fitness score: 75.4534
2-> 10-> 1-> 8-> 9-> 11-> 12-> 3-> 4-> 6-> 5-> 13-> 7->
Parent 2 Index: 67 with fitness score: 72.7687
10-> 3-> 11-> 13-> 6-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
 Their Offspring :
10-> 3-> 11-> 13-> 6-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
Now element with lowest fitness function will be removed i.e: 59 which was with fitness score 41.2472
Afer mutation the Offspring becomes:
10-> 3-> 11-> 6-> 13-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
and replaced with new fitness score 46.5922

---------------------------
Parent 1 Index: 14 with fitness score: 75.4534
2-> 10-> 1-> 8-> 9-> 11-> 12-> 3-> 4-> 6-> 5-> 13-> 7->
Parent 2 Index: 67 with fitness score: 72.7687
10-> 3-> 11-> 13-> 6-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
 Their Offspring :
10-> 3-> 11-> 13-> 6-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
Now element with lowest fitness function will be removed i.e: 79 which was with fitness score 41.5667
Afer mutation the Offspring becomes:
10-> 3-> 11-> 6-> 13-> 2-> 8-> 5-> 1-> 9-> 4-> 7-> 12->
and replaced with new fitness score 46.5922

---------------------------
```

Pressing 3 gives us our calculated output i.e. the best fitness path along with the distance.

```
                    RESULT:
Best path is at the Index: 14 with fitness score: 75.4534

THE PATH IS
2-> 10-> 1-> 8-> 9-> 11-> 12-> 3-> 4-> 6-> 5-> 13-> 7->

        REPRESENTING IN TERMS OF CITIES:


 Goa -> Chennai->  Delhi -> Ghaziabad-> Meerut-> Pune-> Kerala->  Mumbai ->  Bhopal ->  Lavasa ->  Kolkata -> ECR-> Noida->



The distance of this path is 132.532 units
```

Comparison with other techniques

Genetic Algorithm is a good alternative to other algorithms to solve NP problems.Well, huge dataset may take few year or may be infinite amount of time to be solved and GA do it within few minutes.

References

CARGO SHIP LOADING:

1. The Coding Train (Youtube Channel)
2. www.tutorialspoint.com
3. www.geeksforgeeks.org

TRAVELLING SALESMAN PROBLEM:

1. www.tutorialspoint.com
2. www.geeksforgeeks.org
3. https://pdfs.semanticscholar.org/c31b/fd16935da83e419d631245272d7838262308.pdf
4. http://www.ppgia.pucpr.br/~alceu/mestrado/aula3/IJBB-41.pdf

Division of work among group members

Everyone was equally interested, determined and has contributed to the project.