FINAL YEAR PROJECT REPORT(2020-21)

Meta-heuristics of subset selection problems

SUBHAM KUMAR

B.TECH FINAL YEAR

CIVIL WITH COMPUTER SCIENCE

1700676

DAYALBAGH EDUCATIONAL INSTITUTE
(DEEMED UNIVERSITY),AGRA

Under the supervision of
Dr. V. Prem Prakash
Assistant Professor
Department of Electrical Engineering
Faculty of Engineering

Preface:

1. Introduction……
   - 1.1 Some popular TSP solutions
     - 1.1.1 Brute Force Method
     - 1.1.2 Branch and Bound Method
     - 1.1.3 Nearest Neighbour Method
   - 1.2 Real World TSP Applications
   - 1.3 Vehicle Routing Problem
   - 1.4 Genetic Approach
   - 1.5 Structure of Genetic Algorithm
   - 1.6 Representation of solution
     - 1.6.1 Binary Representation
     - 1.6.2 Real Valued Representation
     - 1.6.3 Integer Representation
     - 1.6.4 Permutation Representation
   - 1.7 Comparative Study
2. Travelling Salesman Problem(TSP)
   - 2.1 TSP Characterization
   - 2.2 Application and Linkages With Other Problems
     - 2.2.1 Drilling of Printed Circuit Board
     - 2.2.2 Overhauling Gas Turbine Engines
     - 2.2.3 X-Ray Crystallography
   - 2.3 Implementation of TSP
     - 2.3.1 Naïve Method
     - 2.3.2 Genetic Approach
   - 2.4 Pseudo-Code for Genetic Algorithm
   - 2.5 TSP generally used Crossover operators
     - 2.5.1 Cycle Crossover Operator
     - 2.5.2 One Point Crossover Operator
     - 2.5.3 Two Point Crossover Operator
     - 2.5.4 Uniform Crossover Operator
     - 2.5.5 Ordered Crossover Operator
     - 2.5.6 Partially Matched Crossover Operator
     - 2.5.7 Uniform Partially Matched Crossover Operator
   - 2.6 Description of Dataset
     - 2.6.1 Edge Weight Section
     - 2.6.2 Edge Weight Type
     - 2.6.3 Edge Weight Format
     - 2.6.4 Display Data Type
   - 2.7 Statistics of Crossover Operator
     - 2.7.1 One Point Crossover Operator
     - 2.7.2 Two Point Crossover Operator
     - 2.7.3 Ordered Crossover Operator
     - 2.7.4 Partially Matched Crossover Operator
     - 2.7.5 Uniform Crossover Operator
     - 2.7.6 Uniform Partially Matched Operator

2.8 Description of Dataset
2.9 Ordered Crossover Operator on Large Datasets
2.9.1 Dataset applied: “att532.tsp”
2.9.2 Dataset applied: “d198.tsp”
2.9.3 Dataset applied: “d493.tsp”
2.9.4 Dataset applied: “d1291.tsp”
3. Vehicle Routing Problem (VRP)
3.1 Types of VRP
3.1.1 GVRP
3.1.2 CVRP
3.1.3 VRPSPD
3.1.4 VRPTW
3.2 VRP Characteristics
3.3 Applications of VRP
3.4 Implementation
3.5 Most Commonly Used Crossover Operators
3.6 Description of Dataset
3.6.1 One Point Crossover Operator
3.6.2 Two Point Crossover Operator
3.6.3 Uniform Crossover Operator
3.6.4 Ordered Crossover Operator
3.6.5 Partially Matched Crossover Operator
3.6.6 Uniform Partially Matched Operator
3.7 Local Conclusion on VRP
3.8 Application of UPMX on Large Datasets
3.8.1 Dataset applied: “d198.tsp”
3.8.2 Dataset applied: “d493.tsp”
3.8.3 Dataset applied: “d1291.tsp”
4. Conclusion and Future Work
5. References

1.Travelling salesman problem

The travelling salesman problem is the challenge of finding the shortest possible route yet the most efficient route travelling all the cities. The travelling salesman problem is a NP-hard problem and mathematicians have spent decades to solve this problem i.e. to connect the dots in such a manner to incur minimum cost or distance.

Travelling salesman problem is easy to describe yet it is so difficult to solve because the complexity goes on increasing as we keep increasing the number of nodes to the graph. This behavior of the problem is accountable for it to be termed in the category of combinatorial optimization NP-complete problems. The reason it is classified in the category of NP-hard as it has no “quick” solution.

The problem can be solved by analyzing every round-trip route to determine the shortest one. This becomes a great problem when the number of round trips increases suppose it increases from 10 to 15 which increases the number of permutation and combination from 3,00,000 to 87 billion and surpasses the capabilities of the best computers available at the present time.

1.1 Some of the popular TSP solutions include:

1.1.1 Brute force approach:

This approach is also known as the naïve approach calculates and compares all possible permutations of routes or paths to determine the shortest unique solution. To solve the TSP using the Brute-Force approach, you must calculate the total number of routes and then draw and list all the possible routes. Calculate the distance of each route and then choose the shortest one—this is the optimal solution.

1.1.2 Branch and bound method:

This method breaks a problem to be solved into several sub-problems. It’s a system for solving a series of sub-problems, each of which may have several possible solutions and where the solution selected for one problem may have an effect on the possible solutions of subsequent sub-problems. To solve the TSP using the Branch and Bound method, you must choose a start node and then set bound to a very large value (let’s say infinity). Select the cheapest arch between the unvisited and current node and then add the distance to the current distance.

1.1.3 Nearest neighbour method:

This is perhaps the simplest TSP heuristic. The key to this method is to always visit the nearest destination and then go back to the first city when all other cities are visited. To solve the TSP using this method, we need to choose a random city and then look for the closest unvisited city and go there. Once you have visited all cities, you must return to the first city.

1.2 Real world TSP applications:

Despite the complexity of solving the Travelling Salesman Problem, it still finds applications in all verticals.

For instance, efficient solutions found through the TSP are being used in the last mile delivery. Last mile delivery refers to the movement of goods from a transportation hub, such as a depot or a warehouse, to the end customer’s choice of delivery. Last mile delivery is the leading cost driver in

the supply chain. Companies usually shoulder some of the costs to better compete in the market. In fact, a last mile delivery costs the company an average of $10.1, but the customer only pays an average of $8.08. This is the reason why businesses strive to minimize the cost of last mile delivery.

1.3 Vehicle Routing Problem:

The minimization of costs in last mile delivery is essentially a Vehicle Routing Problem (VRP)**.** VRP is a generalized version of the TSP and is one of the most widely studied problems in mathematical optimization. It deals with finding a set of routes or paths to reduce delivery costs. The problem domain may involve a set of depot locations, hundreds of delivery locations, and several vehicles.

Determining the optimal solution to VRP is NP-hard, so the size of problems that can be solved, optimally, using mathematical programming or combinatorial optimization may be limited. Therefore, commercial solvers tend to use heuristics due to the size and frequency of real world VRP's they need to solve.

The VRP concerns the service of a delivery company. How things are delivered from one or more depots which has a given set of home vehicles and operated by a set of drivers who can move on a given road network to a set of *customers*. It asks for a determination of a set of *routes*, *S*, such that all customers' requirements and operational constraints are satisfied and the global transportation cost is minimized. This cost may be monetary, distance or otherwise.

1.4 Genetic Approach

The cause of using a metaheuristic approach has vast applications as it includes a vast exploration technique to traverse through all the nodes and the converge to a particular solution which is then optimized in further generations to provide better route and finally the best traversal. This approach is very useful as we keep using various optimized genetic operators as it has been found that different genetic operators which include the crossover operator and the mutation operator in which the crossover operator has a significant role to play .Therefore, the study of the various crossover operators is done to the TSP and VRP problem which has been studied for decades by researchers knowing the impact it has on the various firms like flipkart , amazon and not just these ,there are so many delivery firms that rely on the these algorithms to minimize the delivery costs so that the commodity prices can be kept lowered.

1.5 Structure of Genetic Algorithm:

A genetic algorithm works by building a population of chromosomes which is a set of possible solutions to the optimization problem. Within a generation of a population, the chromosomes are randomly altered in hopes of creating new chromosomes that have better evaluation scores. The next generation population of chromosomes is randomly selected from the current generation with selection probability based on the evaluation score of each chromosome. The simple genetic algorithm follows the structure depicted below. Each of these operations will be described in the following subsections:

Simple Structure:

```
Simple Genetic Algorithm ()
{
```

```
        Initialization;
        Evaluation;
        while termination criterion has not been reached
        {
                Selection_and_Reproduction;
                Crossover;
                Mutation;
                Evaluation;
        }
}
```

Structure of a simple genetic algorithm.

Initialization

Initialization involves setting the parameters for the algorithm, creating the scores for the simulation, and creating the first generation of chromosomes. In this benchmark, seven parameters are set:

- the genes value (Genes) is the number of variable slots on a chromosome;
- the codes value (Codes) is the number of possible values for each gene;
- the population size (PopSize) is the number of chromosomes in each generation;
- crossover probability (CrossoverProb) is the probability that a pair of chromosomes will be crossed;
- mutation probability (MutationProb) is the probability that a gene on a chromosome will be mutated randomly;
- the maximum number of generations (MaxGenerations) is a termination criterion which sets the maximum number of chromosome populations that will be generated before the top scoring chromosome will be returned as the search answer; and
- the generations with no change in highest-scoring (elite) chromosome (GensNoChange) is the second termination criterion which is the number of generations that may pass with no change in the elite chromosome before that elite chromosome will be returned as the search answer.

The scores matrix for the simulation, which is generated in the GenAlgGen script, is the set of scores for which the best solution is to be found. The attempted optimization is to find the code for each gene in the solution chromosome that maximizes the average score for the chromosome. Finally, the first generation of chromosomes are generated randomly.

Evaluation

Each of the chromosomes in a generation must be evaluated for the selection process. This is accomplished by looking up the score of each gene in the chromosome, adding the scores up, and averaging the score for the chromosome. As part of the evaluation process, the elite chromosome of the generation is determined.

Selection and Reproduction

Chromosomes for the next generation are selected using the roulette wheel selection scheme [5] to implement proportionate random selection. Each chromosome has a probability of being chosen equal to its score divided by the sum of the scores of all of the generation's chromosomes. In order

to avoid losing ground in finding the highest-scoring chromosome, elitism [5] has been implemented in this benchmark. Elitism reserves two slots in the next generation for the highest scoring chromosome of the current generation, without allowing that chromosome to be crossed over in the next generation. In one of those slots, the elite chromosome will also not be subject to mutation in the next generation.

Crossover

In the crossover phase, all of the chromosomes (except for the elite chromosome) are paired up, and with a probability CrossoverProb, they are crossed over. The crossover is accomplished by randomly choosing a site along the length of the chromosome, and exchanging the genes of the two chromosomes for each gene past this crossover site.

Mutation

After the crossover, for each of the genes of the chromosomes (except for the elite chromosome), the gene will be mutated to any one of the codes with a probability of MutationProb. With the crossover and mutations completed, the chromosomes are once again evaluated for another round of selection and reproduction.

Termination

The loop of chromosome generations is terminated when certain conditions are met. When the termination criteria are met, the elite chromosome is returned as the best solution found so far. For this benchmark, there are two criteria: if the number of generation has reached a maximumnumber, MaxGenerations, or if the elite solution has not changed for a certain number of generations, GensNoChange.

NOTE:

One of the most important decisions to make while implementing a genetic algorithm is deciding the representation that we will use to represent our solutions. It has been observed that improper representation can lead to poor performance of the GA.

Therefore, choosing a proper representation, having a proper definition of the mappings between the phenotype and genotype spaces is essential for the success of a GA.

Some of the most commonly used representations for genetic algorithms are depicted below. However, representation is highly problem specific and the reader might find that another representation or a mix of the representations mentioned here might suit a specific problem better.

1.6 Representation

1.6.1 Binary Representation:

This is one of the simplest and most widely used representation in GAs. In this type of representation the genotype consists of bit strings.

For some problems when the solution space consists of Boolean decision variables – yes or no, the binary representation is natural. Take for example the 0/1 Knapsack Problem. If there are n items,

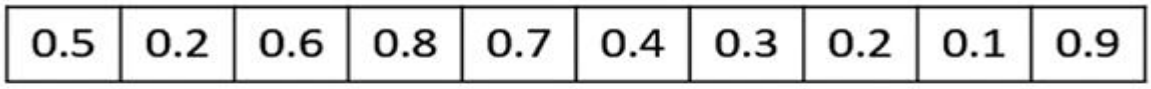
we can represent a solution by a binary string of n elements, where the x^{th} element tells whether the item x is picked (1) or not (0).

For other problems, specifically those dealing with numbers, we can represent the numbers with their binary representation.

Note: The problem with this kind of encoding is that different bits have different significance and therefore mutation and crossover operators can have undesired consequences. This can be resolved to some extent by using **Gray Coding,** as a change in one bit does not have a massive effect on the solution.

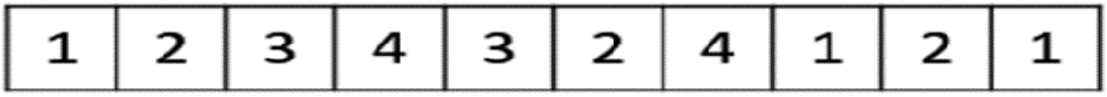
1.6.2 Real Valued Representation:

For problems where we want to define the genes using continuous rather than discrete variables, the real valued representation is the most natural. The precision of these real valued or floating point numbers is however limited to the computer.



1.6.3 Integer Representation:

For discrete valued genes, we cannot always limit the solution space to binary 'yes' or 'no'. For example, if we want to encode the four distances – North, South, East and West, we can encode them as **{0,1,2,3}**. In such cases, integer representation is desirable.



1.6.4 Permutation Representation:

In many problems, the solution is represented by an order of elements. In such cases permutation representation is the most suited.

A classic example of this representation is the travelling salesman problem (TSP). In this the salesman has to take a tour of all the cities, visiting each city exactly once and come back to the starting city. The total distance of the tour has to be minimized. The solution to this TSP is naturally an ordering or permutation of all the cities and therefore using a permutation representation makes sense for this problem.

1.7 COMPARATIVE STUDY**:**

This motive is to present a comparative study of the various aspects and effects that the algorithms of TSP and VRP show when we change various genetic operators and mainly the

crossover operator as it is the main component to cause genetic changes. As it is already known that TSP and VRP are NP-hard problems. Therefore if the time required to reach the optimum solution , can be reduced it can be of great help to the industrial world for door-to-door services including firms of delivery (including flipkart , amazon) and for other purposes such as resource allocation etc.

2. TSP

2.1 TSP Characterization

The main characteristics of the TSP are listed as follows:

1. The objective is to minimize the distance between cities visited.
2. All cities should be visited only once.
3. The tour is closed, i.e. the origin and destination are the same location.

Mathematically it can be represented:

$$\text{Min } z = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \qquad (1)$$

$$\text{subject to}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1, 2, \ldots, n$$

$$x_{ij} = \{0, 1\}$$

where d_{ij} is the distance from city i to city j, $d_{ij} = \infty$ when $i = j$ and $d_{ij} = d_{ji}$.

2.2 Applications and linkages with other problems:

2.2.1 Drilling of printed circuit boards

A direct application of the TSP is in the drilling problem of printed circuit boards (PCBs). To connect a conductor on one layer with a conductor on another layer, or to position the pins of integrated circuits, holes have to be drilled through the board. The holes may be of different sizes. To drill two holes of different diameters consecutively, the head of the machine has to move to a tool box and

change the drilling equipment. This is quite time consuming. Thus it is clear that one has to choose some diameter, drill all holes of the same diameter, change the drill, drill the holes of the next diameter, etc. Thus, this drilling problem can be viewed as a series of TSPs, one for each hole diameter, where the 'cities' are the initial position and the set of all holes that can be drilled with one and the same drill.

2.2.2 Overhauling gas turbine engines

It occurs when gas turbine engines of aircraft have to be overhauled. To guarantee a uniform gas flow through the turbines there are nozzle-guide vane assemblies located at each turbine stage. Such an assembly basically consists of a number of nozzle guide vanes affixed about its circumference. All these vanes have individual characteristics and the correct placement of the vanes can result in substantial benefits (reducing vibration, increasing uniformity of flow, reduced fuel consumption).

2.2.3 X-ray crystallography

Analysis of the structure of crystals is an important application of the TSP. Here an X-ray diffractometer is used to obtain information about the structure of crystalline material. To this end a detector measures the intensity of Xray reflections of the crystal from various positions. Whereas the measurement itself can be accomplished quite fast, there is a considerable overhead in positioning time since up to hundreds of thousands positions have to be realized for some experiments. In the two examples that we refer to, the positioning involves moving four motors. The time needed to move from one position to the other can be computed very accurately. The result of the experiment does not depend on the sequence in which the measurements at the various positions are taken. However, the total time needed for the experiment depends on the sequence. Therefore, the problem consists of finding a sequence that minimizes the total positioning time. This leads to a traveling salesman problem.

2.3 Implementation of TSP:

There are various ways to implement the problem-

2.3.1 Naïve Method

1. Consider city 1 as the starting and ending point. Since the route is cyclic, we can consider any point as a starting point.
2. Generate all (n-1)! permutations of cities.
3. Calculate the cost of every permutation and keep track of the minimum cost permutation.
4. Return the permutation with minimum cost.

2.3.2 Genetic approach

1. Initialize the population randomly.
2. Determine the fitness of the chromosome.
3. Until done repeat:
   1. Select parents.
   2. Perform crossover and mutation.
   3. Calculate the fitness of the new population.
   4. Append it to the gene pool.

2.4 Pseudo code for genetic approach:

```
Initialize procedure GA{
    Set cooling parameter = 0;
    Evaluate population P(t);
    While( Not Done ){
        Parents(t) = Select_Parents(P(t));
        Offspring(t) = Procreate(P(t));
        p(t+1) = Select_Survivors(P(t), Offspring(t));
        t = t + 1;
    }
}
```

***NOTE**: The following computations were done using:

- **Language** : Python
- **Framework** : Deap (exclusively used for evolutionary computations using GA)

2.5 **Travelling Salesman Problem (TSP) generally used crossover operators** :

2.5.1 **`Cycle Crossover Operator`**

The cycle crossover (CX) operator was first proposed by Oliver et al. Using this technique to create offspring in such a way that each bit with its position comes from one of the parents. It is up to us how we choose the first bit for the offspring to be either from the first or from the second parent.

Other Crossover operators used in DEAP framework in use for this module:

2.5.2 **`cxOnePoint`**

Executes a one point crossover on the input sequence individuals. The two individuals are modified in place. The resulting individuals will respectively have the length of the other.

2.5.3 **`cxTwoPoints`**

Executes a two point crossover on the input sequence individuals. The two individuals are modified in place. The resulting individuals will respectively have the length of the other.

2.5.4 **`cxUniform`**

Executes a uniform crossover that modifies in place the two sequence individuals. The attributes are swapped according to the indpb(individual probability).

2.5.5 **cxOrdered**

Executes an ordered crossover (OX) on the input individuals. The two individuals are modified in place. This crossover expects sequence individuals of indices, the result for any other type of individuals is unpredictable.

2.5.6 **cxPartialyMatched**

Executes a partially matched (PMX) on the input individuals. The two individuals are modified in place. This crossover expects sequence individuals of indices, the result for any other type of individuals is unpredictable.

2.5.7 **cxUniformPartialyMatched**

Executes a uniform partially matched crossover (UPMX) on the input individuals. The two individuals are modified in place. This crossover expects sequence individuals of indices, the result for any other type of individuals is unpredictable.

2.6 Description of Dataset used:

The dataset used for the following computations is named as "bayg29.tsp" and is a collection of 29 states in Bavaria, Germany. The dataset is a symmetric TSP dataset. It consists of geographical coordinates i.e latitudes and longitudes of the various locations present in the Bavaria city of Germany.

2.6.1 EDGE WEIGHT SECTION :

➔The edge weights are given in the format specified by the EDGE WEIGHT FORMAT entry. At present, all explicit data is integral and is given in one of the (self-explanatory) matrix formats. with implicitly known lengths.

2.6.2 EDGE WEIGHT TYPE : EXPLICIT

➔Weights are listed explicitly in the corresponding section.

2.6.3 EDGE WEIGHT FORMAT : UPPER_ROW

➔ Upper triangular matrix (row-wise without diagonal entries)

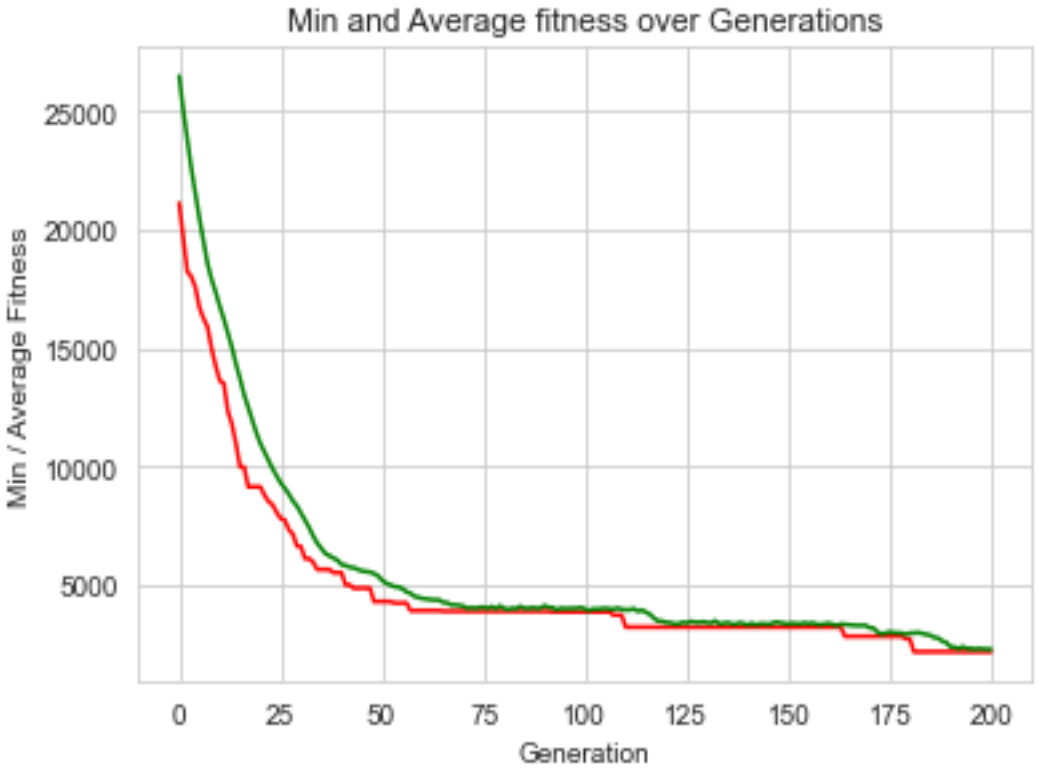
2.6.4 DISPLAY_DATA_TYPE: TWOD_DISPLAY

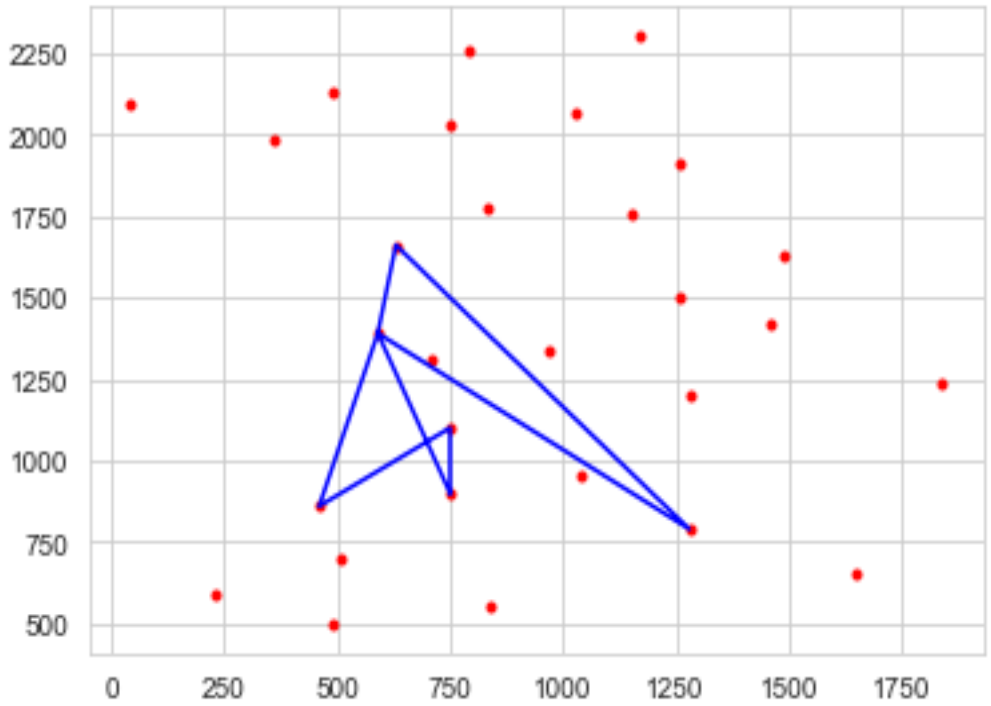
➔The display format of the dataset is in 2-D format.

NOTE:
➔In all graphs "red" denotes Minimum fitness values obtained over generations.
➔In all graphs "green" denotes Average fitness values obtained over generations.

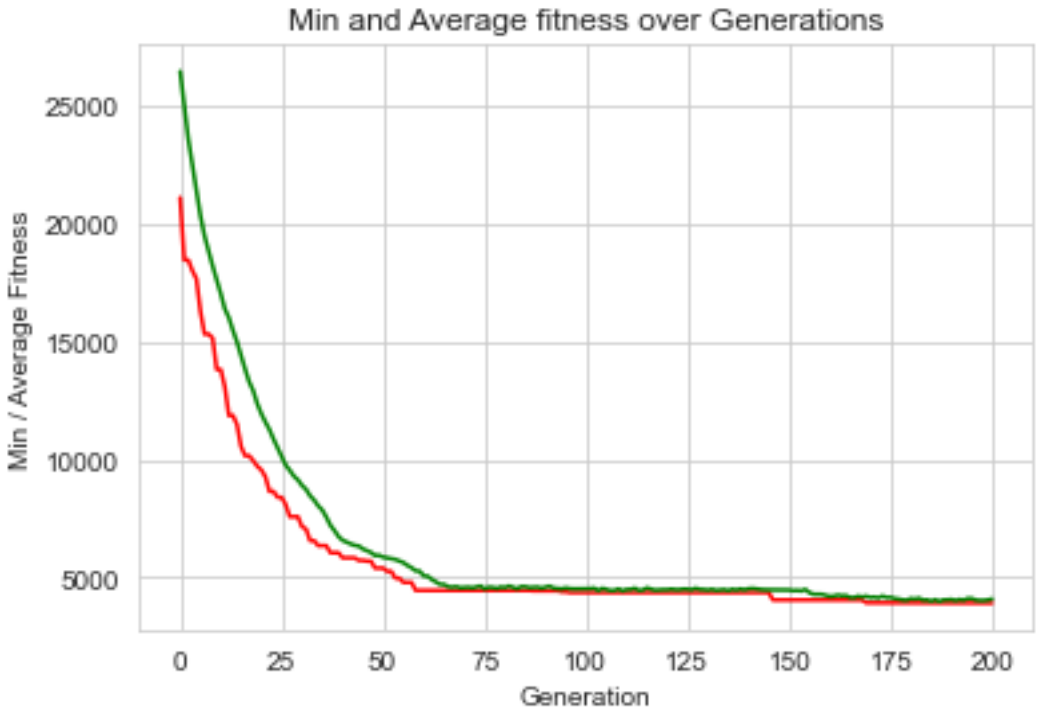
2.7 Statistics of crossover operators and their results:

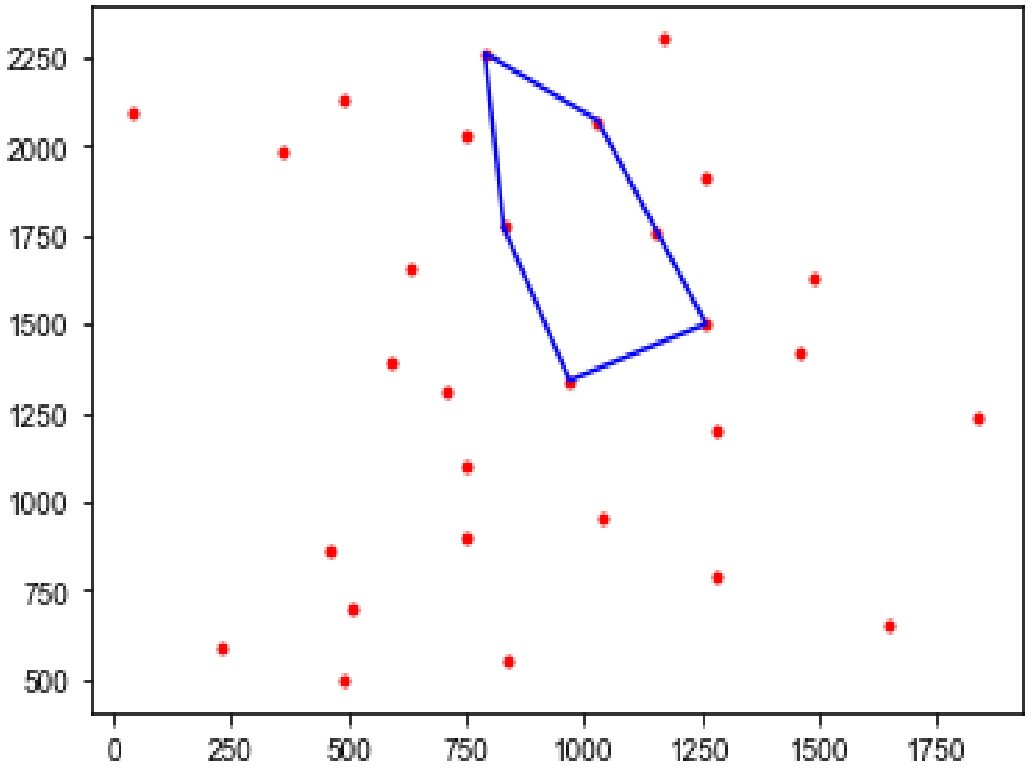
2.7.1 One Point crossover operator(cxOnePoint)



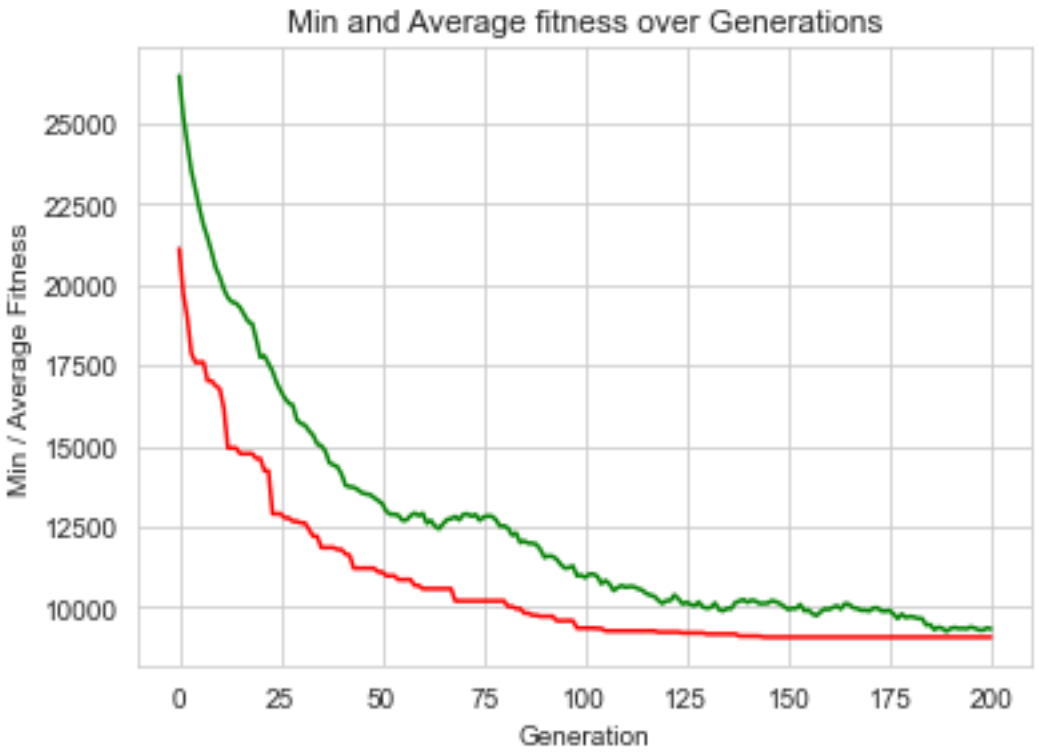


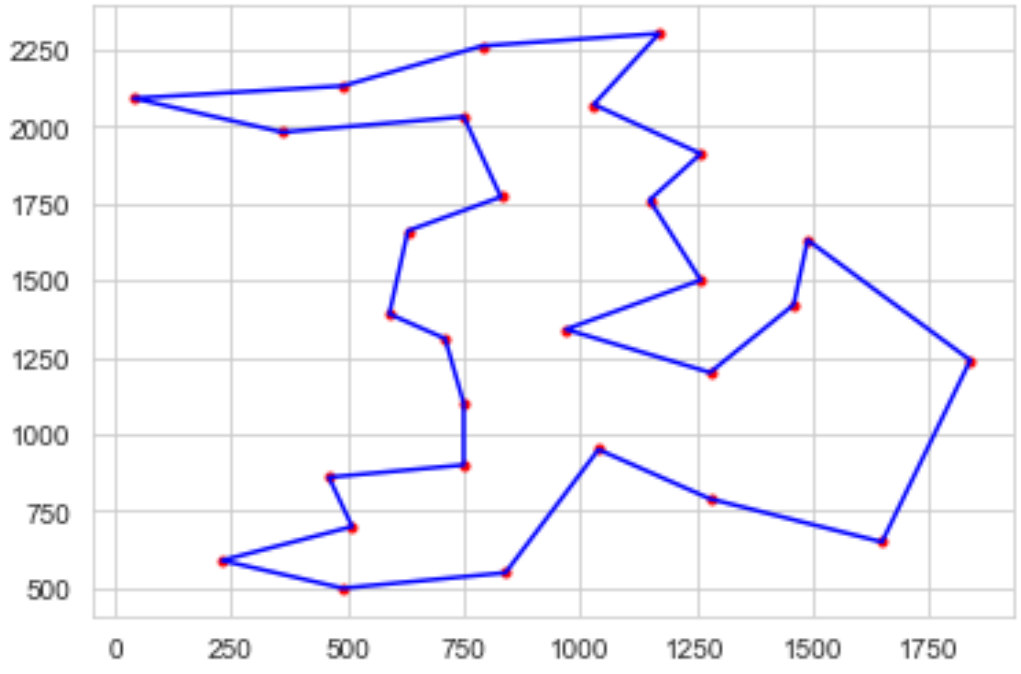
2.7.2 Two Point crossover operator(cxTwoPoint)



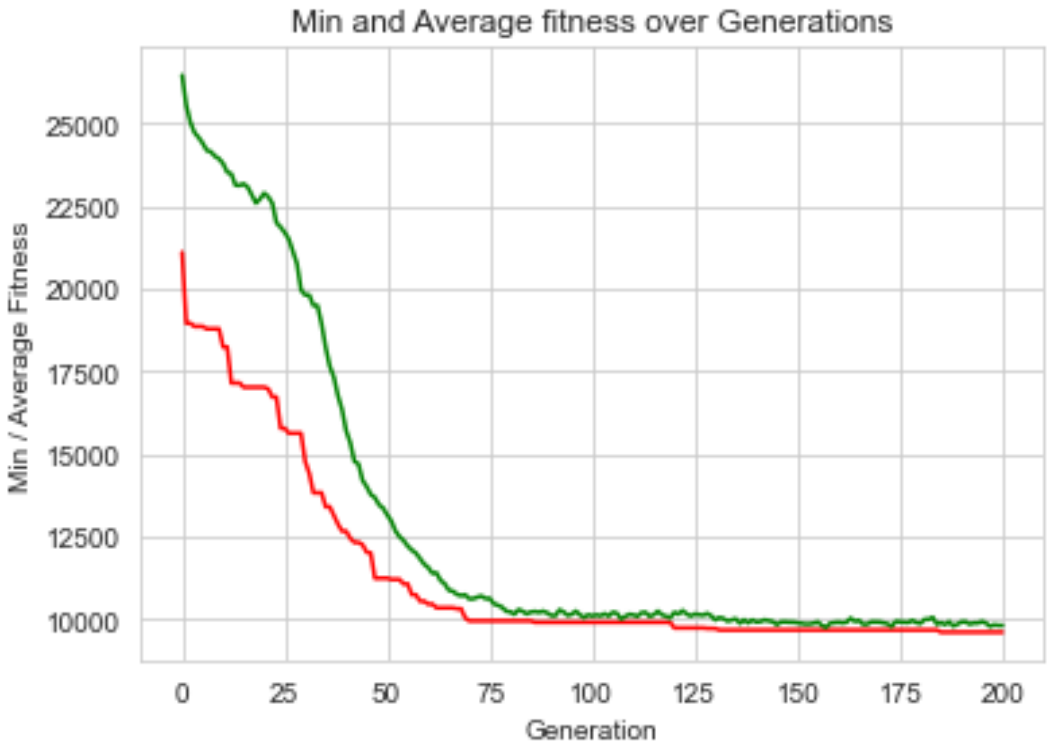


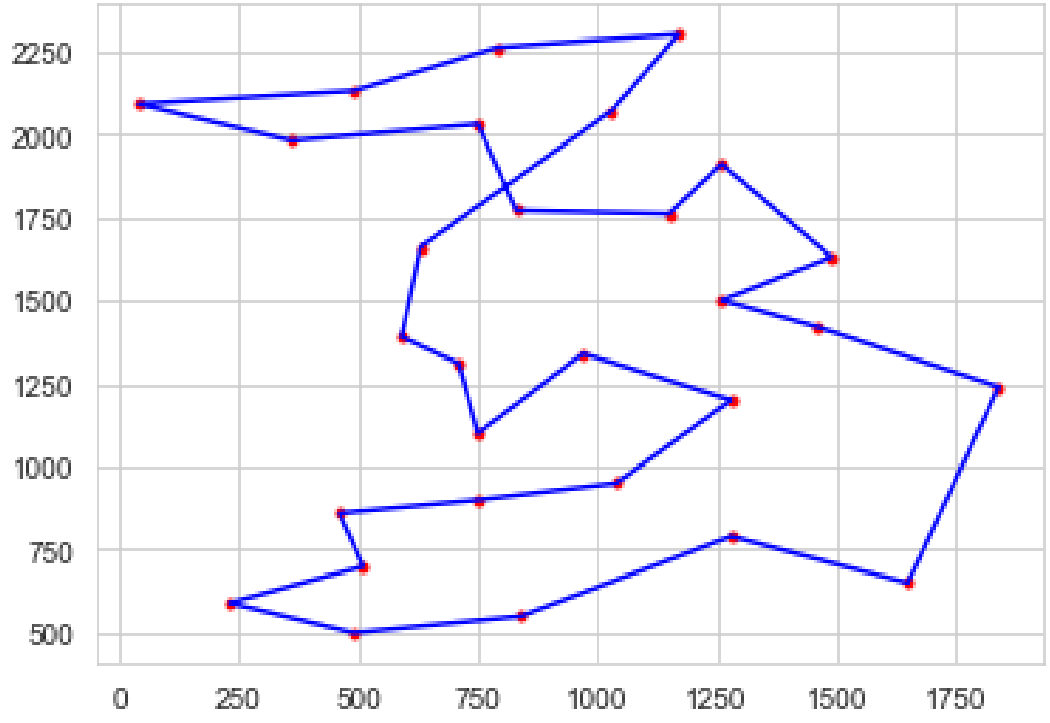
2.7.3 Ordered crossover operator(cxOrdered)



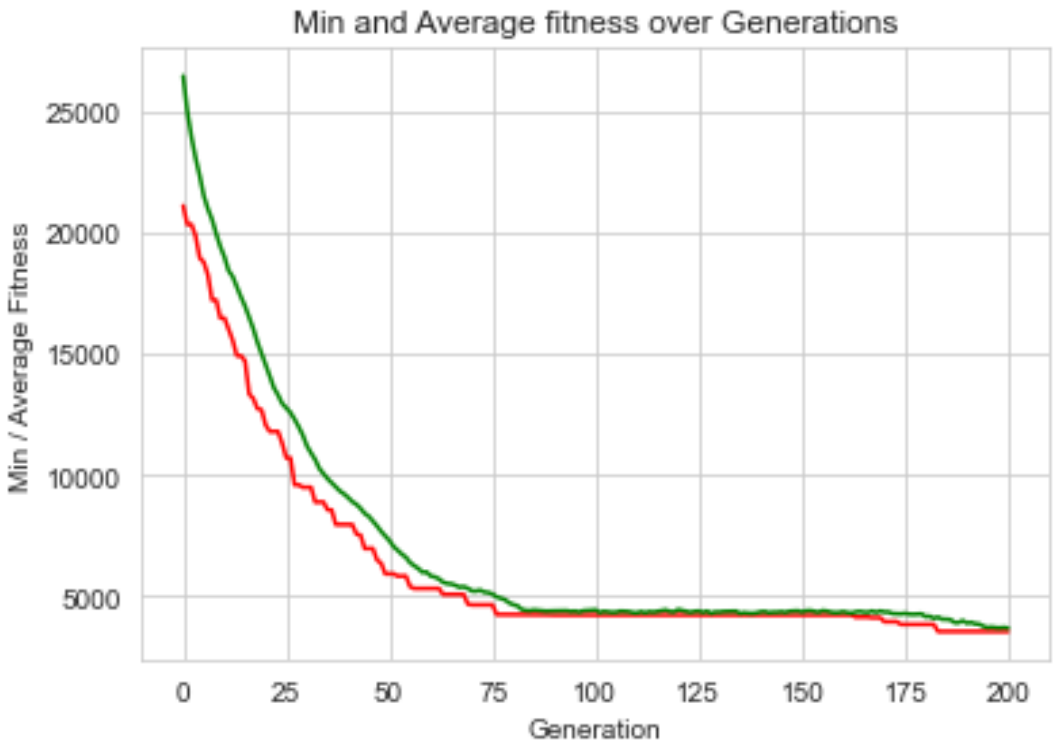


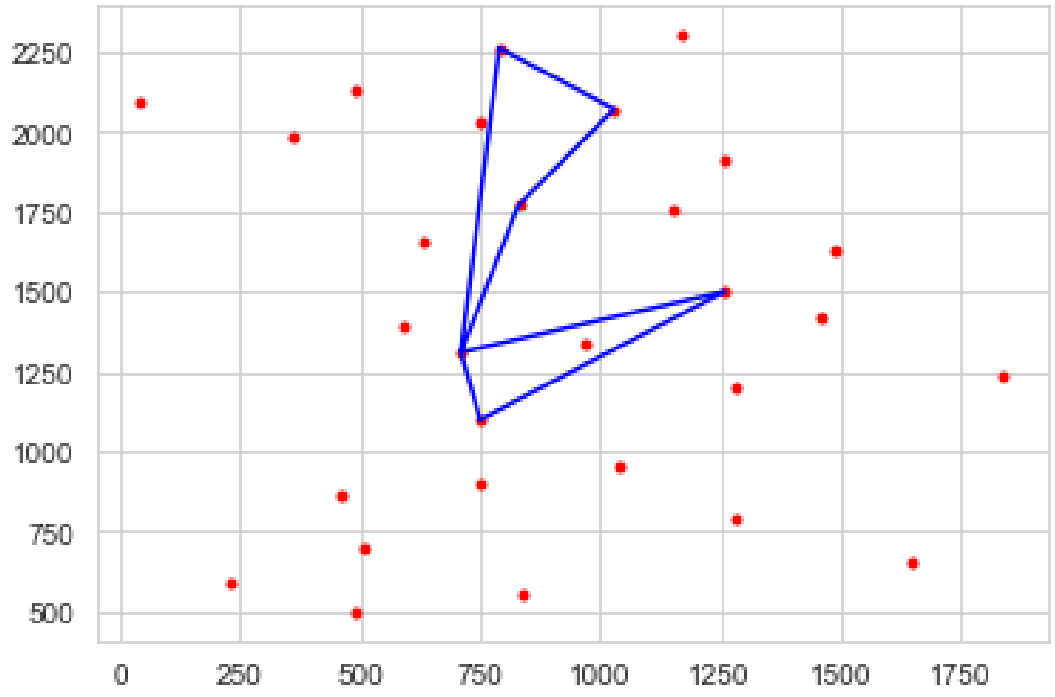
2.7.4 Partially Matched crossover operator(cxPartialyMatched)



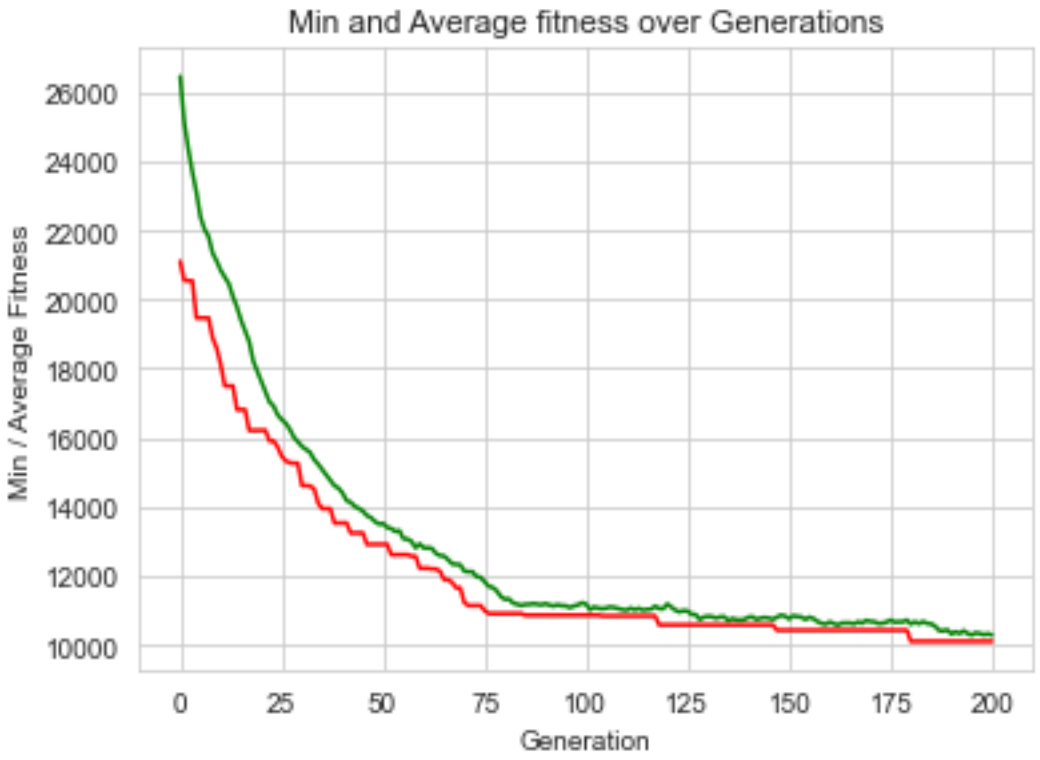


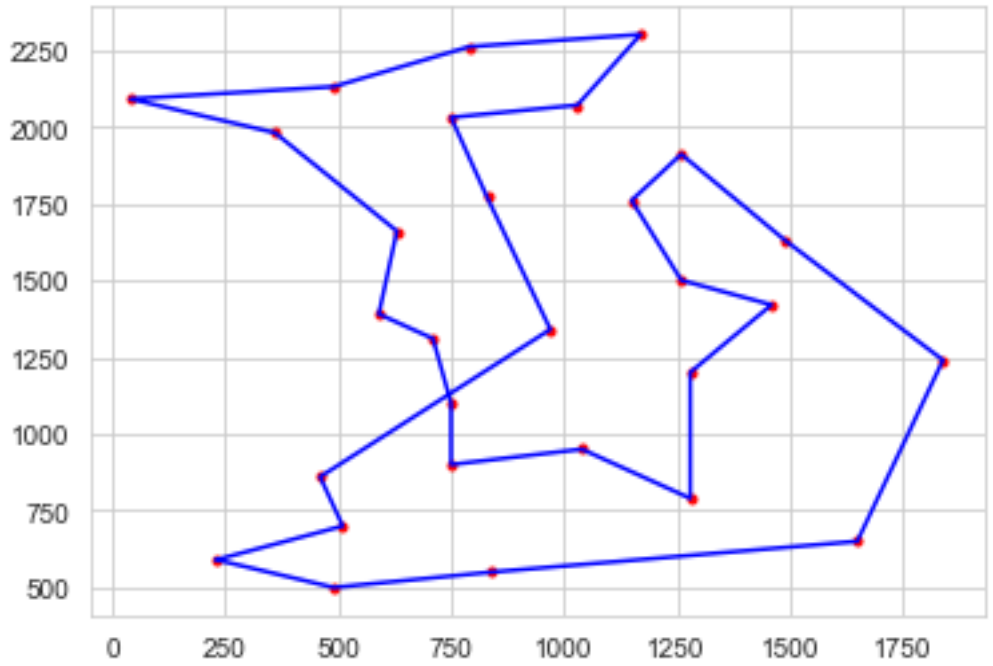
2.7.5 Uniform crossover operator(cxUniform)





2.7.6 Uniform partially Matched crossover operator(cxUniformPartialyMatched)





➔It was observed from above observations that on a consistent basis cxOrdered crossover operator was performing much better as compared to the other five operators which were used.

➔All of these operators were individually tested 50 times each and the readings were found to be consistent with all the preconditions , were satisfied by the crossover operator and the results were consistently found to converge to a particular point i.e. the standard deviation was found to be minimum as the values were less spread out as compared to other crossover operators whose values were found to be more spread out. Another point to note was that the other crossover operators usage also proved to be fatal to the results as the algorithm failed to traverse all the nodes in the network.

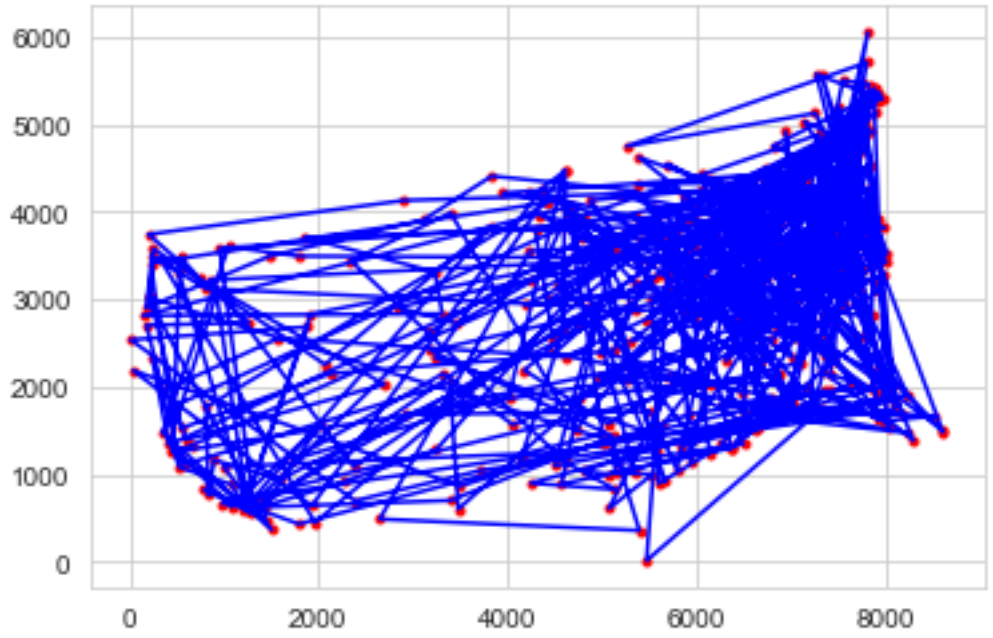
2.8 Description of Dataset used:

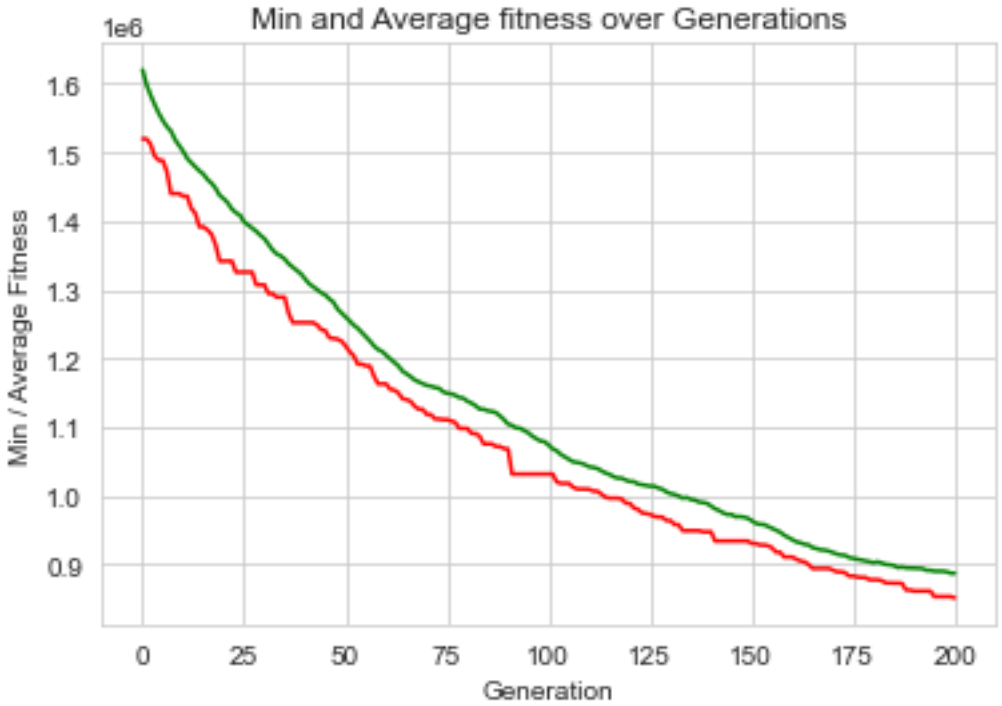
➔The dataset used is a collection of 532 cities provided by Padberg/Rinaldi. The dataset used for the following computations is named as “att532.tsp. The edge weight type ATT corresponds to a special “pseudo-Euclidean” distance function. Other details remain the same as explained earlier.

➔ The dataset used is a collection of 198,493,1291 drilling locations given by Reinelt. It has an Edge weight type of Euclidien 2D-model. These datasets are named as “d198.tsp” , “d493.tsp” , “d1291.tsp”.

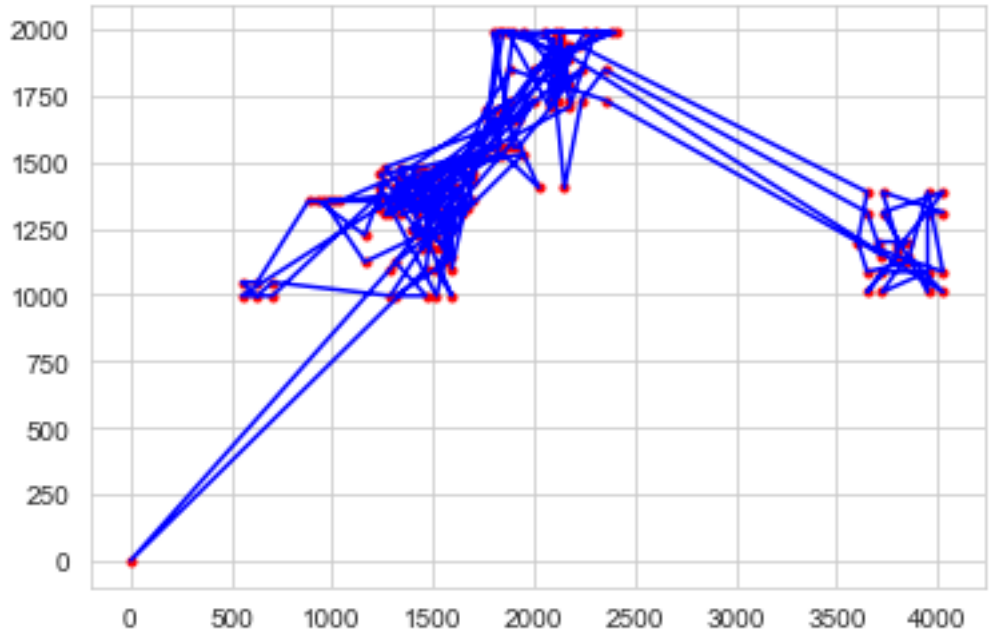
2.9 As per above observations , the cxOrdered crossover operator was applied on larger datasets and the following result was obtained:

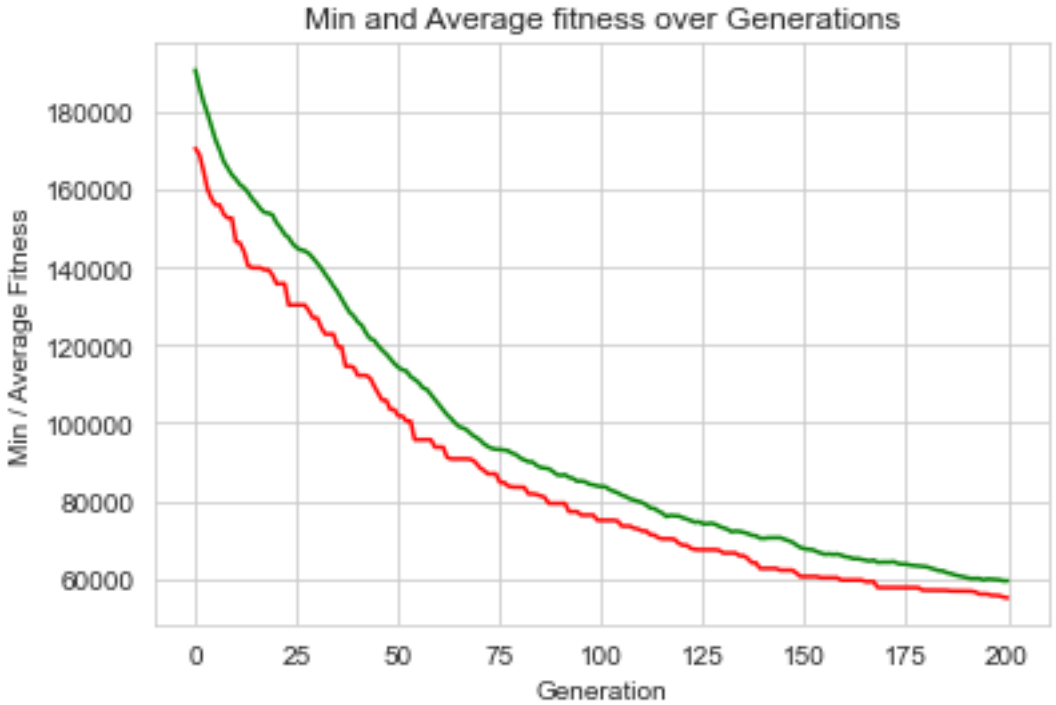
2.9.1 Dataset applied: “att532.tsp”



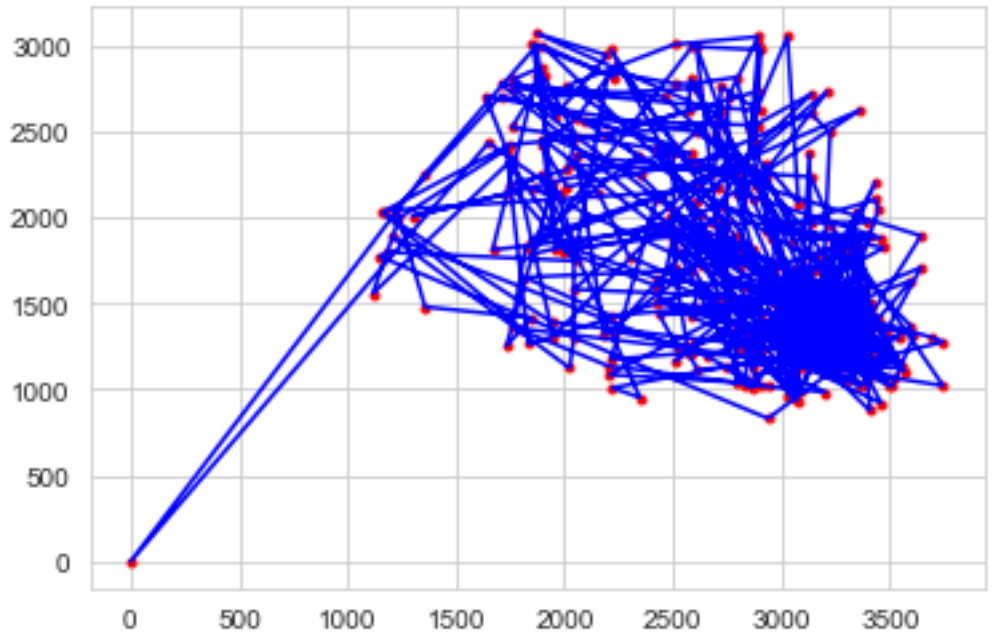


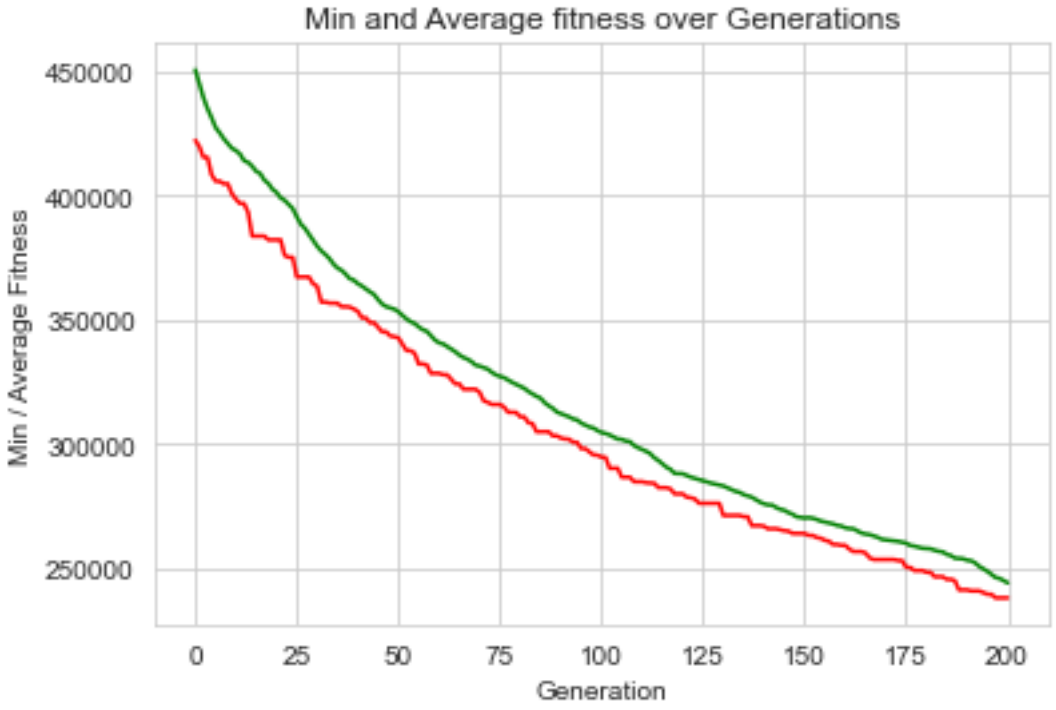
2.9.2 Dataset applied: “d198.tsp”



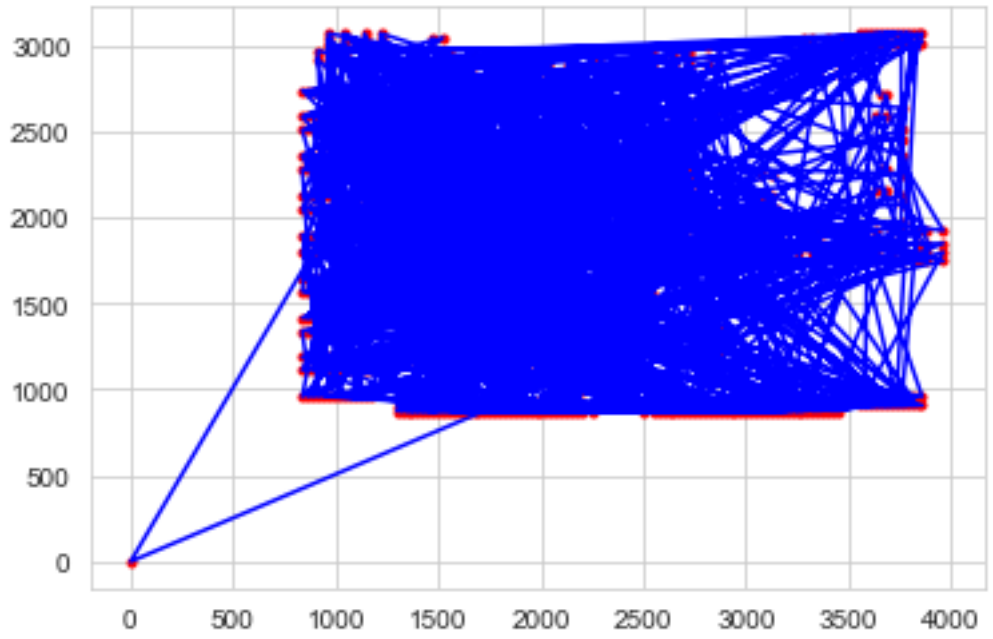


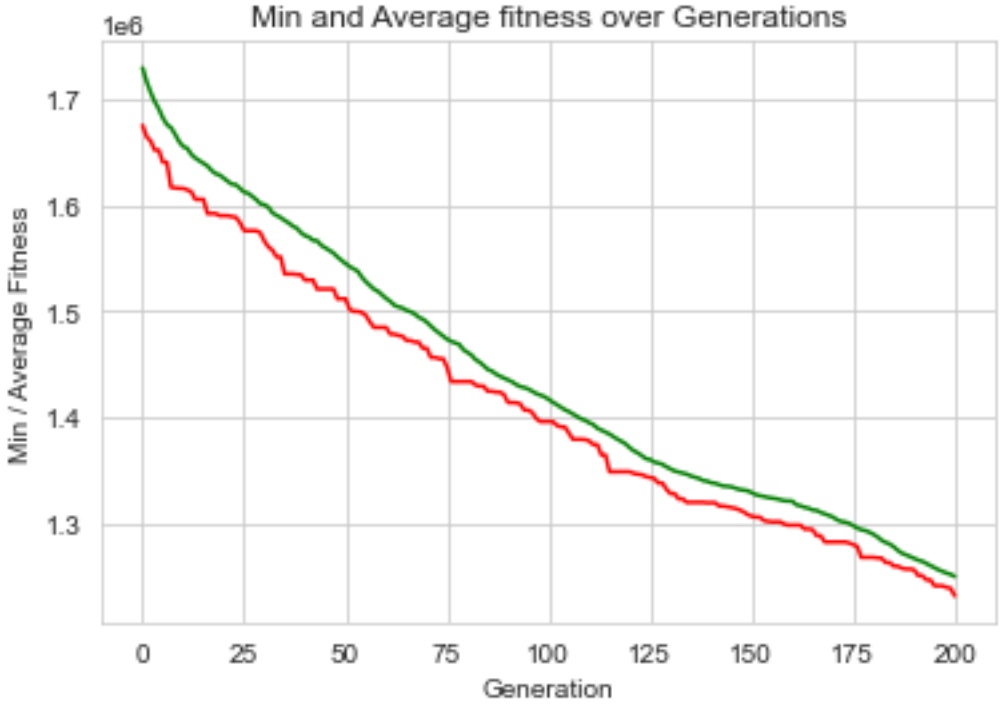
2.9.3 Dataset applied: “d493.tsp”





2.9.4 Dataset applied: “d1291.tsp”





➔Thus, it can be said that Ordered crossover operator(cxOrdered) performs well with large datasets as well.

3.Vehicle Routing Problem(VRP)

3.1 There are several types of VRPs which can be categorized based on their nature and complexity. For the sake of understanding, VRPs’ few variants are defined as:

3.1.1 The general VRP was first introduced by Danzig and Ramser in 1959 by the name truck dispatching problem which is a generalization of the Travelling Salesman Problem (TSP).

3.1.2 Capacitated vehicle routing problem (CVRP) is the basic version of the VRP, which attempts to find number of routes for m number of vehicles with Q units homogenous capacity to minimize total transportation cost of routes while satisfying the delivery demands of n number of customer nodes.

3.1.3 In Vehicle Routing Problem with Simultaneous pick-up and delivery (VRPSPD), a number of routes are found for v number of vehicles with C units' homogenous capacity, which minimize the total transportation cost of all the routes while satisfying the pick-up and delivery demands of the m number of customer nodes simultaneously.

3.1.4 The Vehicle Routing Problem with Time Windows (VRPTW) is an extension of the VRP wherein a time window is associated with each customer in which each customer provides a time frame within which a particular service or task must be completed.

3.2 VRP characteristics:

Vehicle routing problem and its variants belong to family of NP-hard problems. Solving such problems will take enormous amount of computation time while using exact methods. Due to this complex nature of the problem, solutions using heuristics will be helpful in order to address larger solution space in less computational time. Genetic Algorithm is one of the efficient metaheuristics applicable to a large set of combinatorial problems. It is one of the widely used and powerful optimization methods based on the process of natural selection and evolution. Crossover operator and mutation operators are used for transformation to form new individuals. During encoding integer representations of chromosomes are used to in genetic algorithm to solve VRP.

There are three steps on solving VRP with Self-Developed algorithms, namely:

1). Initialization step

2). Iteration step by processing saving and combining route in accordance with the order made for each route in each depot

3). Pair step, to mate generated route for each depot with existing vehicles.

3.3 Applications of Vehicle Routing Problem (VRP):

Vehicle Routing Problem VRP was applied to resource allocation in a power distribution system with needs for service restoration. The objective function consisted in minimizing the displacement by the fleet of maintenance crew and others.

NOTE:

Genetic Algorithm adjusted into three steps of MDVRP. In grouping stage, customers are grouped into the nearest depot. Encoding of individual uses permutation encoding technique. On permutation encoding, chromosomes is a collection of a number representing the position of the customer or depot on a route. Formation of the population can use Josephus' permutation. Based on the choromosomes, each route (individual) has their fitness. Later in the scheduling phase, selection using wheel roulette method used, crossovers using Order Crossover (OX), and mutation using inversion mutation .

3.4 Implementation:

Algorithm flow:

```
➔While not all the Customers have been visited, and there is time left
    Select the next truck, or if all were selected, select the first
    While it has more "tries to find a next Customer"
        Try to find a next customer, with the given constraints
        If success, move, update data and try further with resetting
"tries…"
    End
    Move back to the depot, reload and update truck data
End
```

3.5 Most commonly used crossover operators on VRP problems:

1.cxOnepoint

2.cxTwoPoint

3.cxUniform

4.cxOrdered

5.cxPartialyMatched

6.cxUniformPartialyMatched

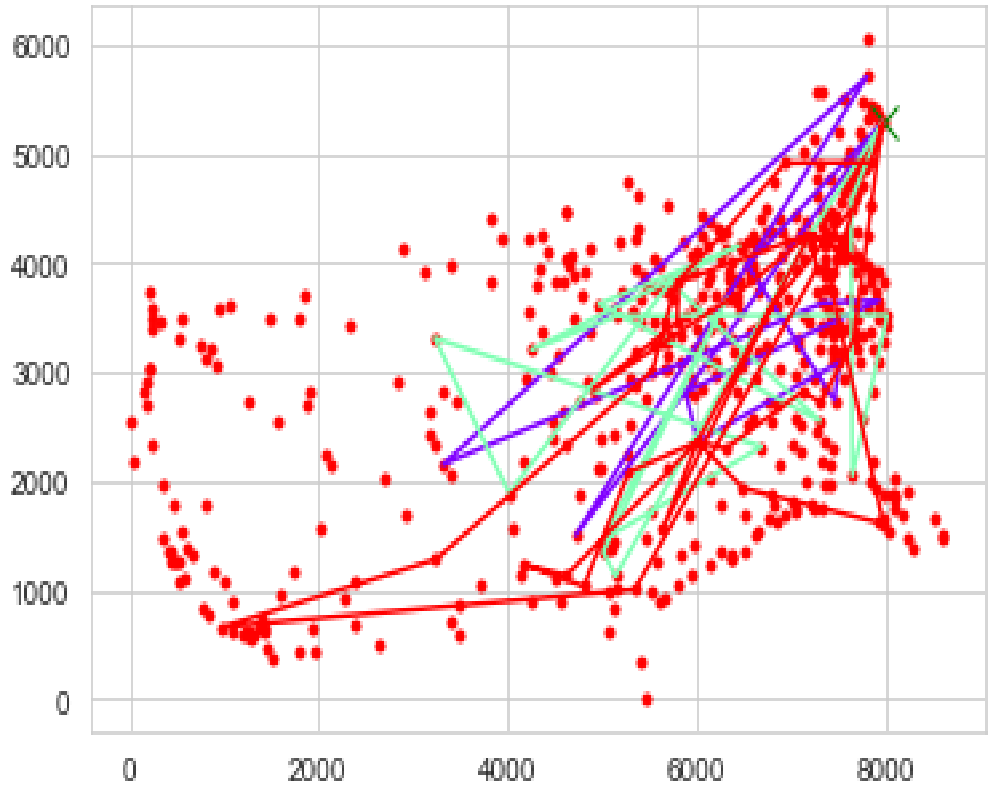
NOTE: These operators are the same as used in TSP problem because VRP is basically a practical aspect of the TSP problem as we know that that TSP has limited use in the industry but based on the concepts of TSP, VRP can help various organizations to save on resourses such as by finding the best possible route as explained earlier.

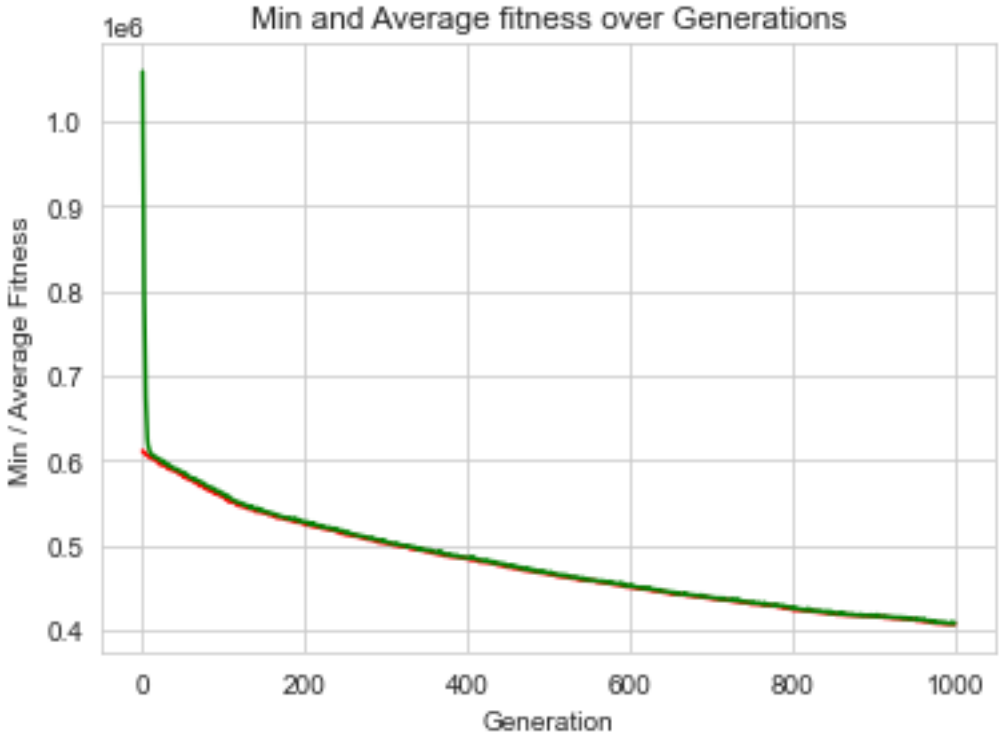
3.6 Description of Dataset used:

➔The dataset used is a collection of 532 cities provided by Padberg/Rinaldi. The dataset used for the following computations is named as "att532.tsp. The edge weight type ATT corresponds to a special "pseudo-Euclidean" distance function. Other details remain the same as explained earlier.

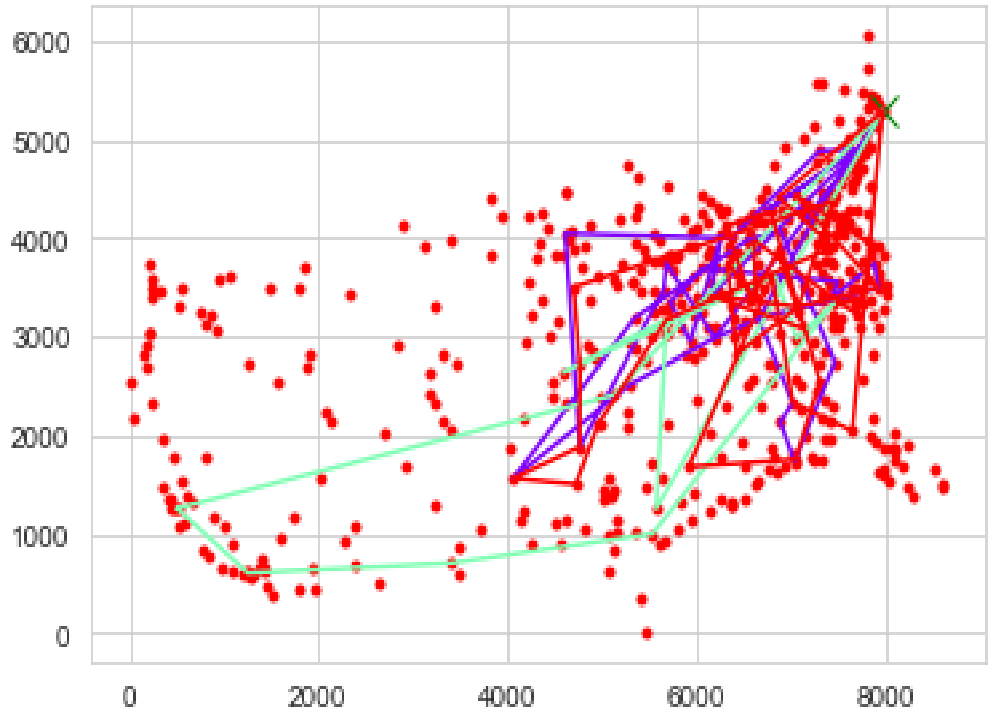
➔ The dataset used is a collection of 198,493,1291 drilling locations given by Reinelt. It has an Edge weight type of Euclidien 2D-model. These datasets are named as "d198.tsp" , "d493.tsp" , "d1291.tsp".

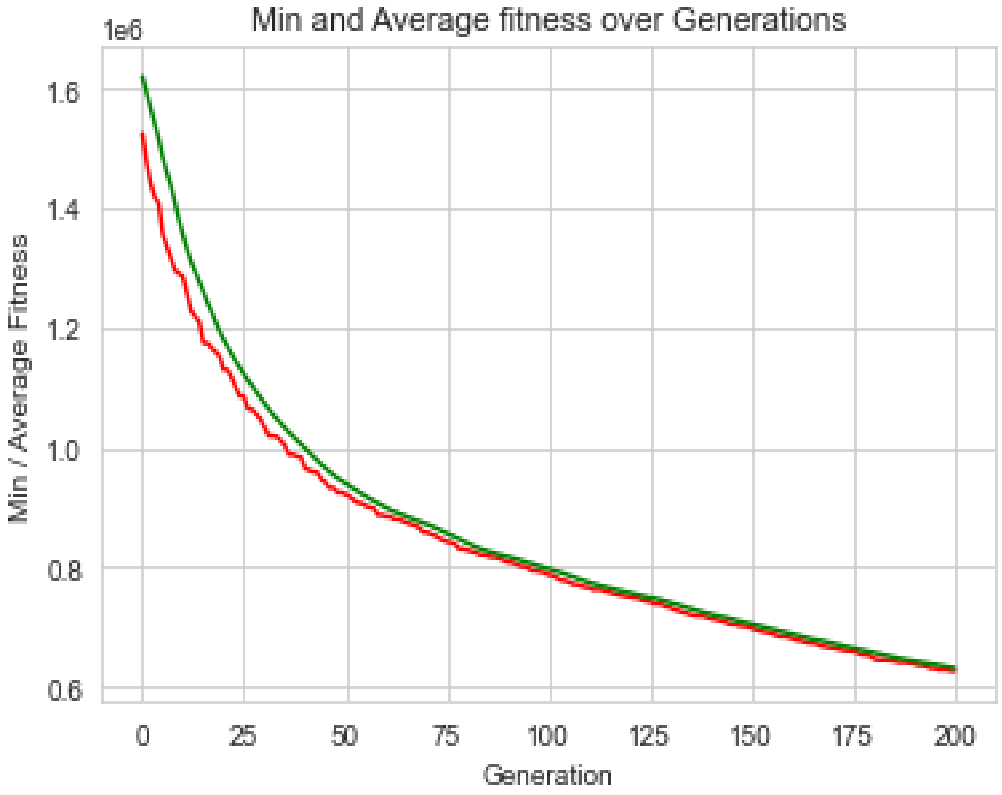
3.6.1 One Point crossover operator(cxOnePoint)



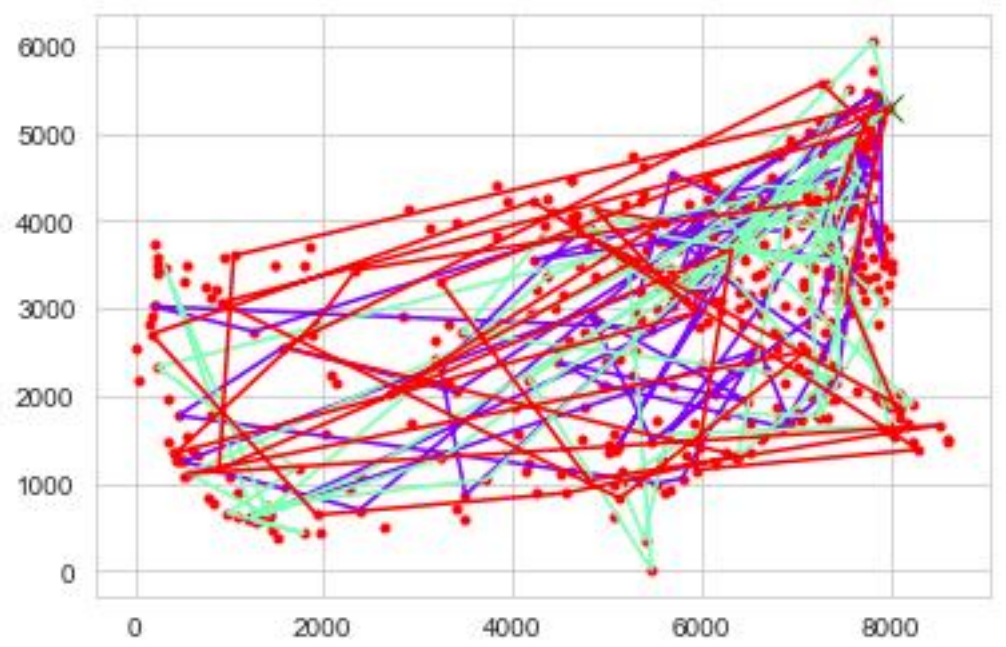


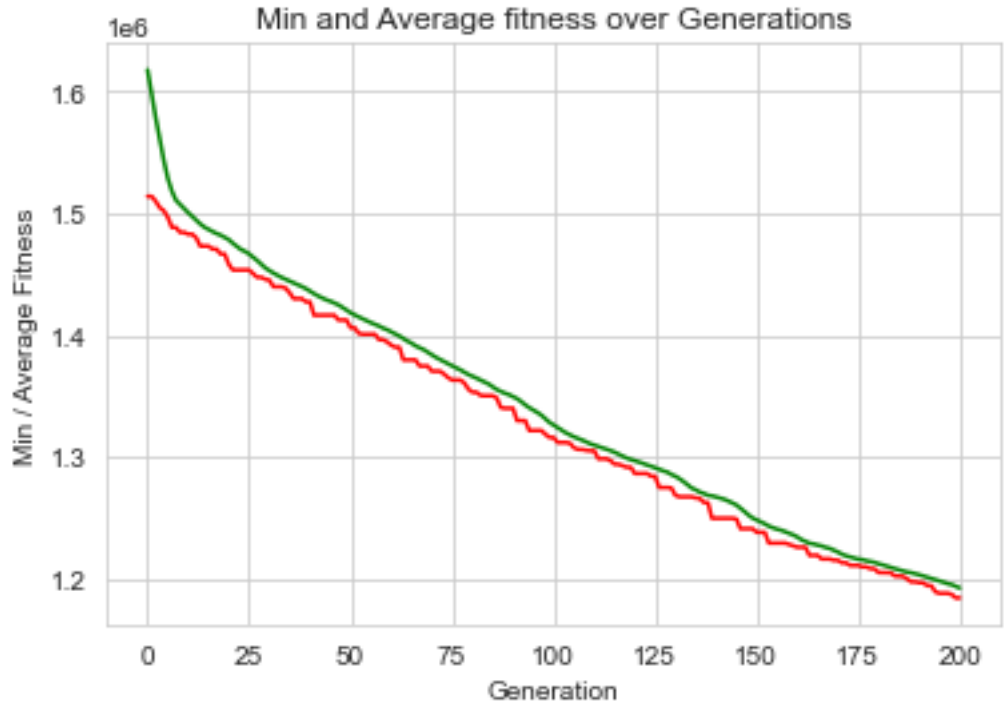
3.6.2 Two point crossover operator(cxTwoPoint)



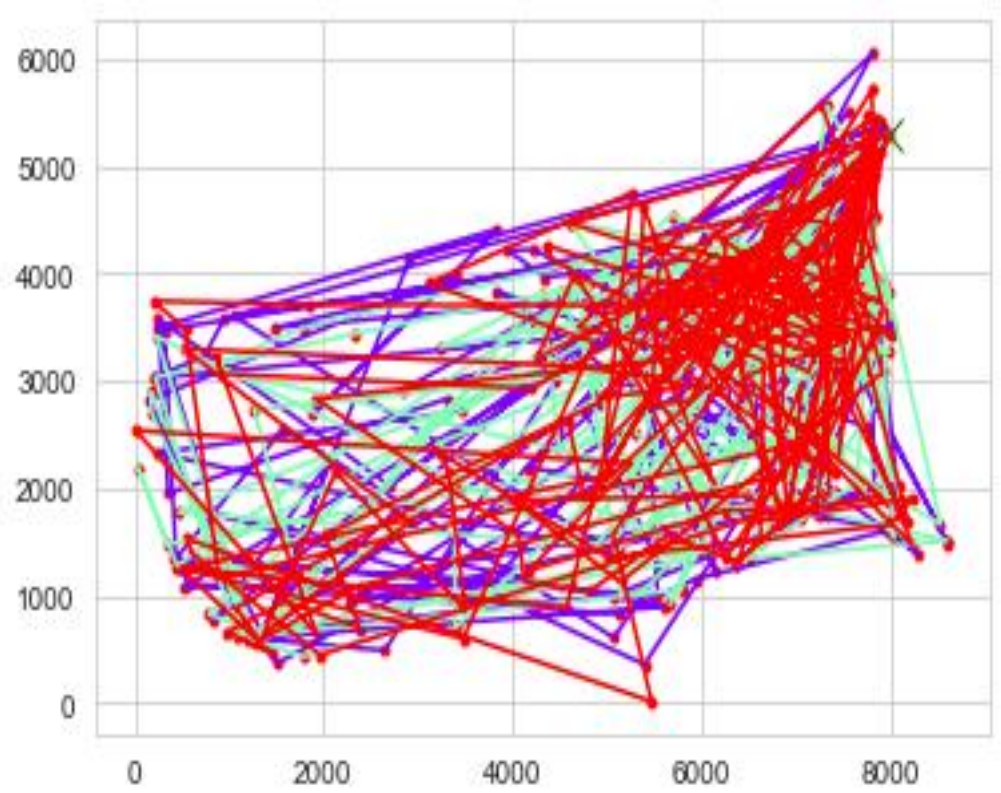


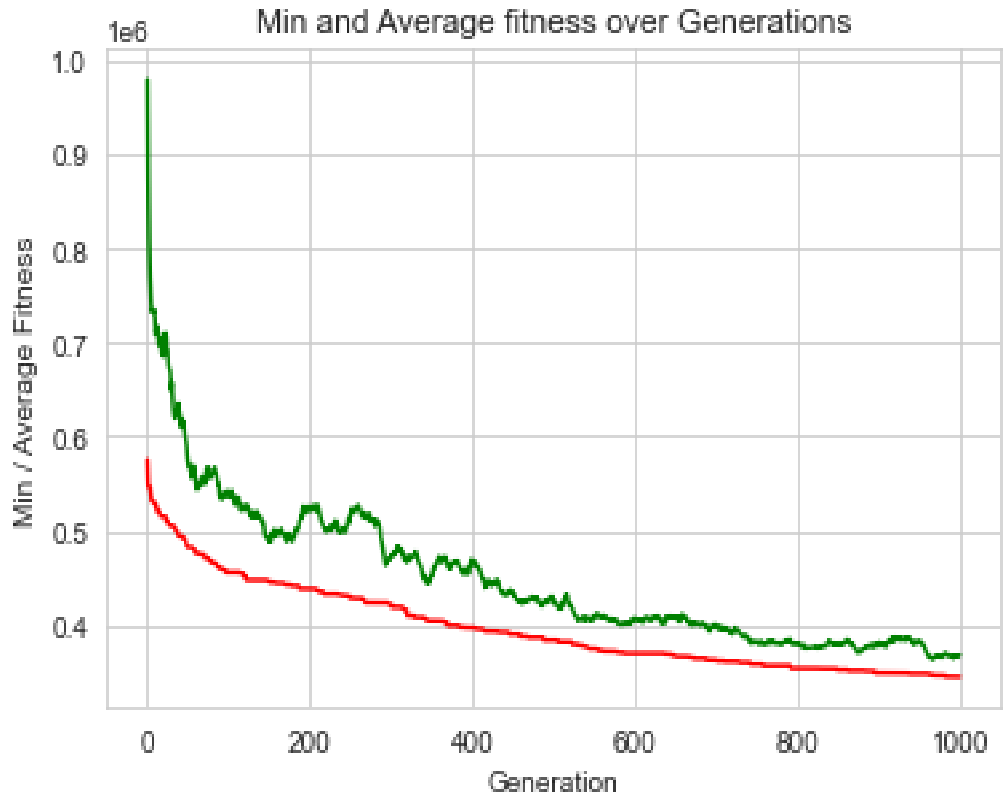
3.6.3 Uniform crossover(cxUniform)



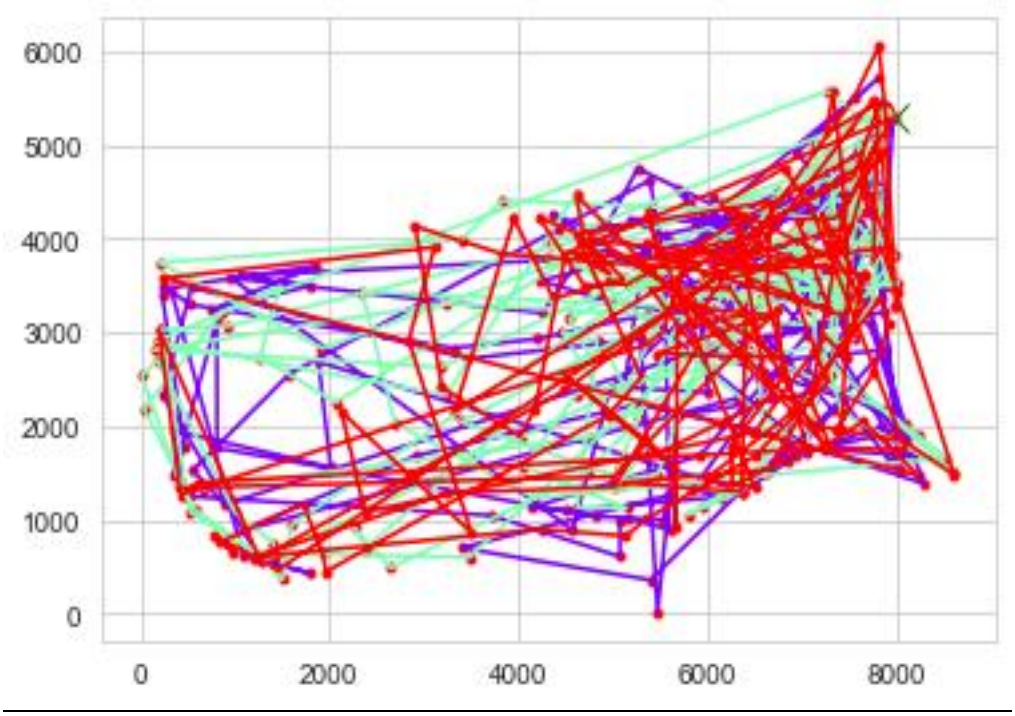


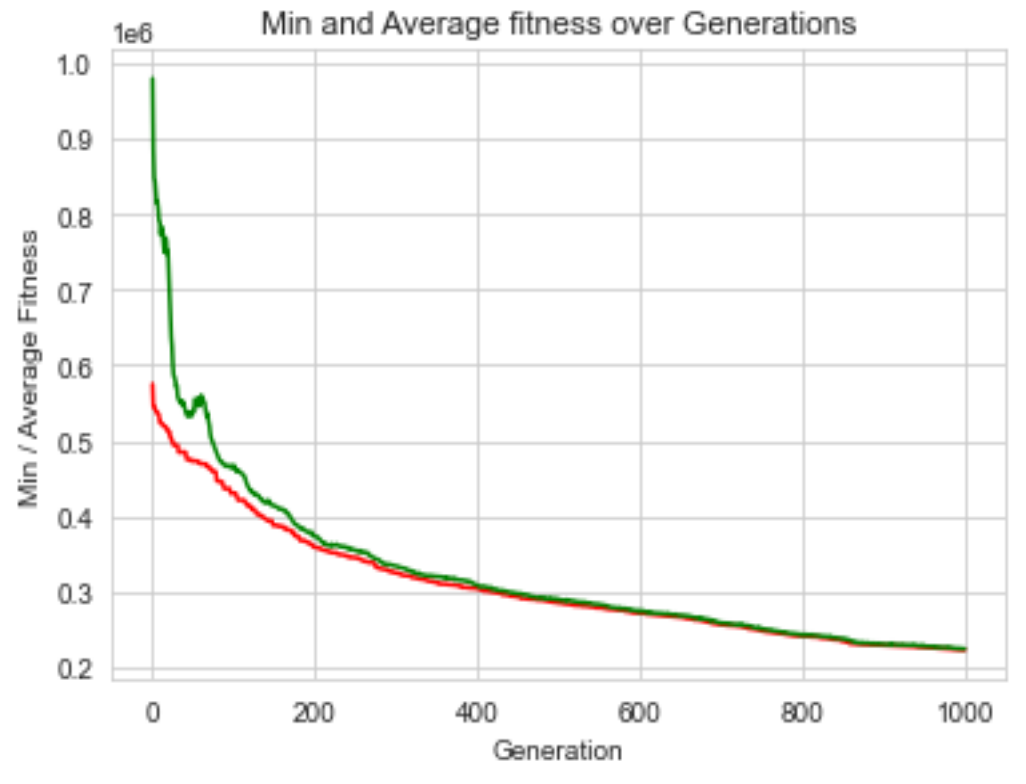
3.6.4 Ordered crossover operator(cxOrdered)



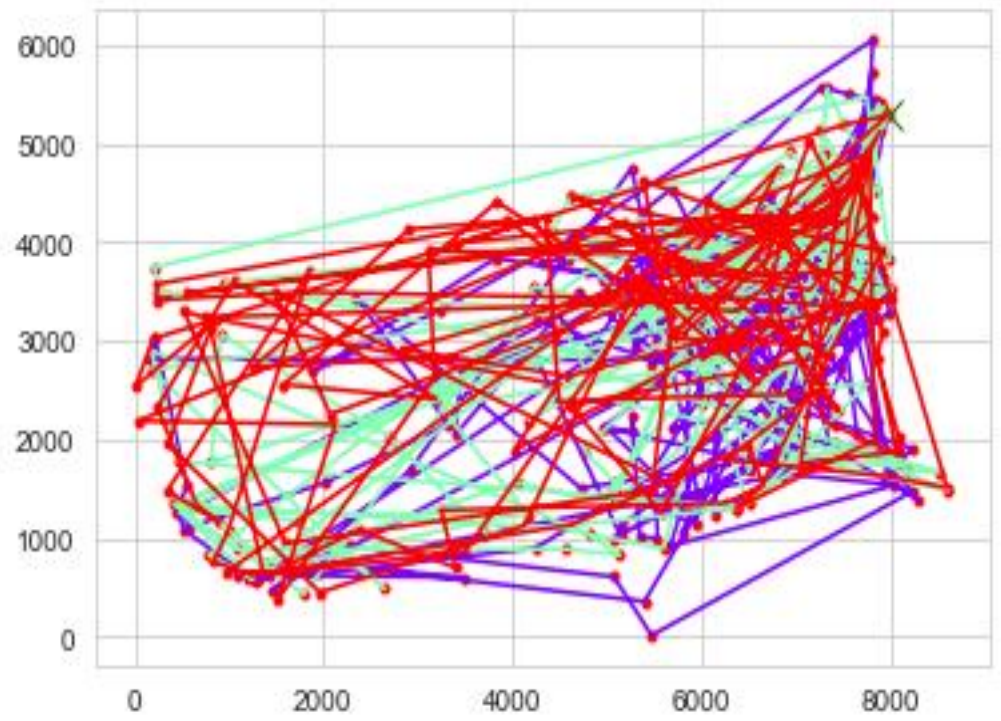


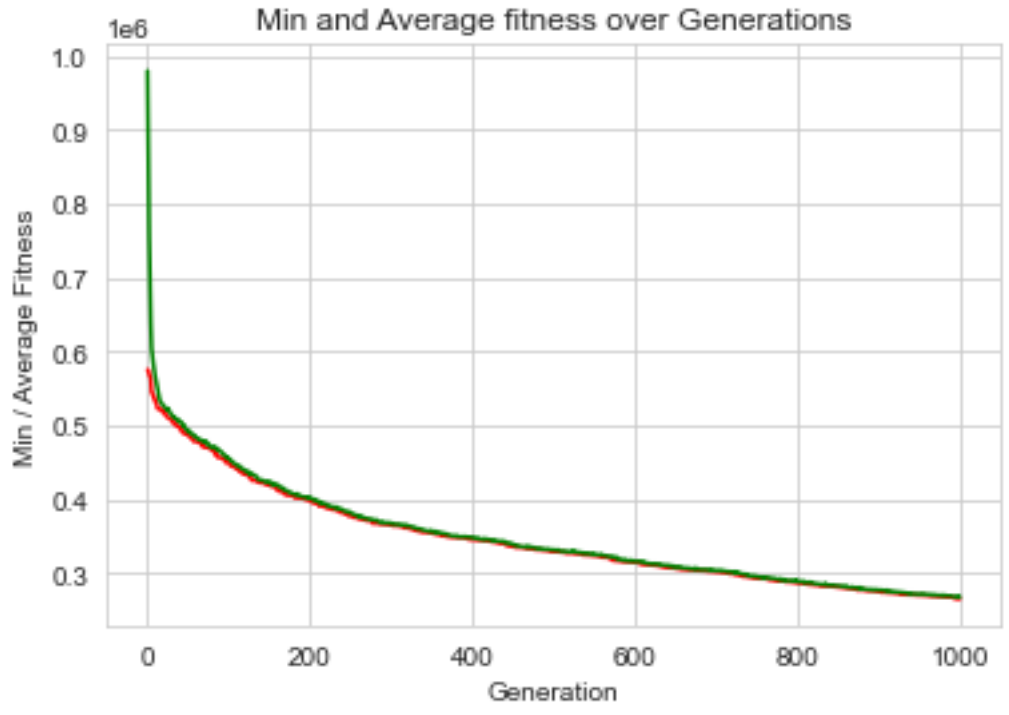
3.6.5 Partially Matched crossover operator(cxPartialyMatched)





3.6.6 Uniform Partially Matched crossover operator(cxUniformPartialyMatched)





3.7 Using the above statistics and graphs it was observed that :

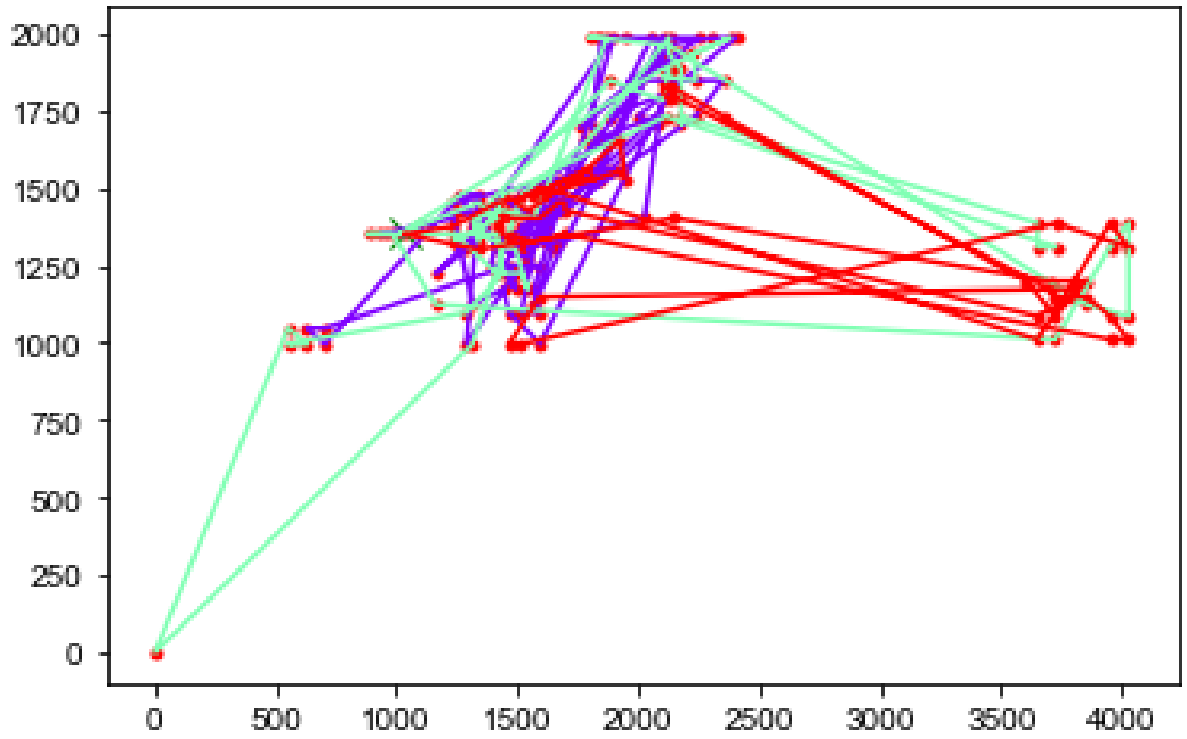
➔Uniform Partially Matched(UPMX) i.e. cxUniformPartialyMatched performs better due to faster convergence i.e. it can be clearly seen the in the first generation itself that the value has converged to value that is closer to the optimum obtained after a thousand generations.

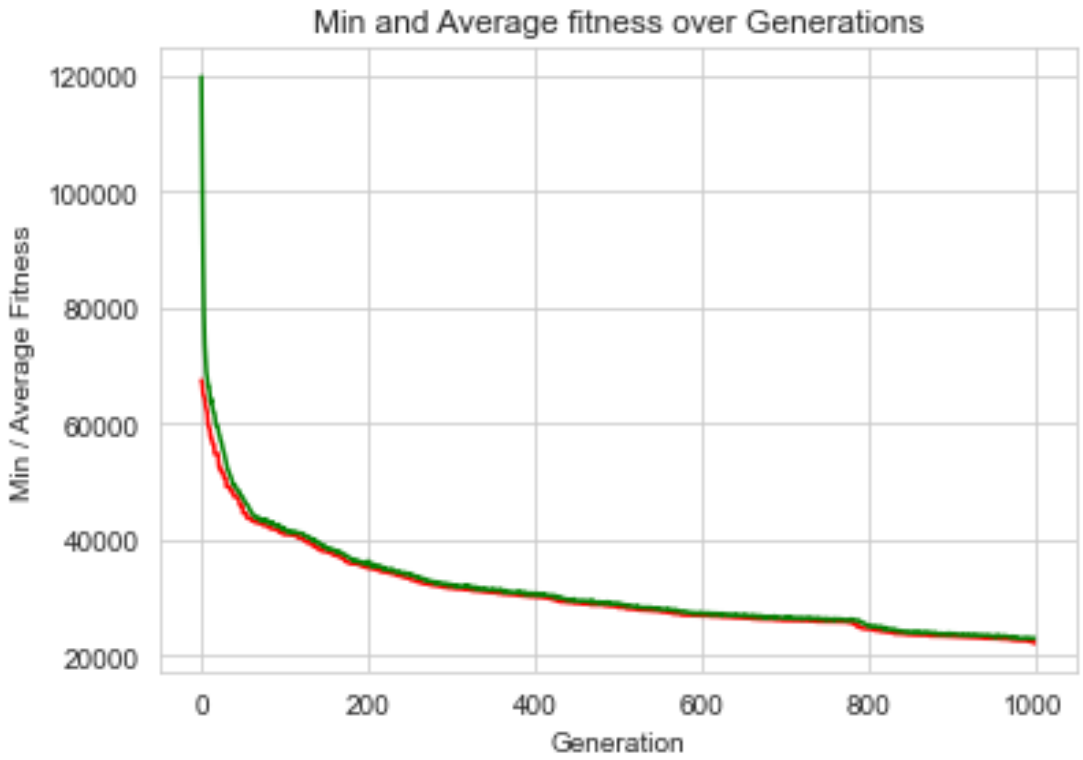
➔Uniform crossover(UX) i.e. cxUniform operator performs better when the dataset is small. cxUniform is very consistent as it consistently gives values with minimum amount of standard deviation as compared to other crossover operators.

➔It was also observed that increasing the probability of crossover operation better fitness values were obtained with number of generations kept unchanged because of more exploration.

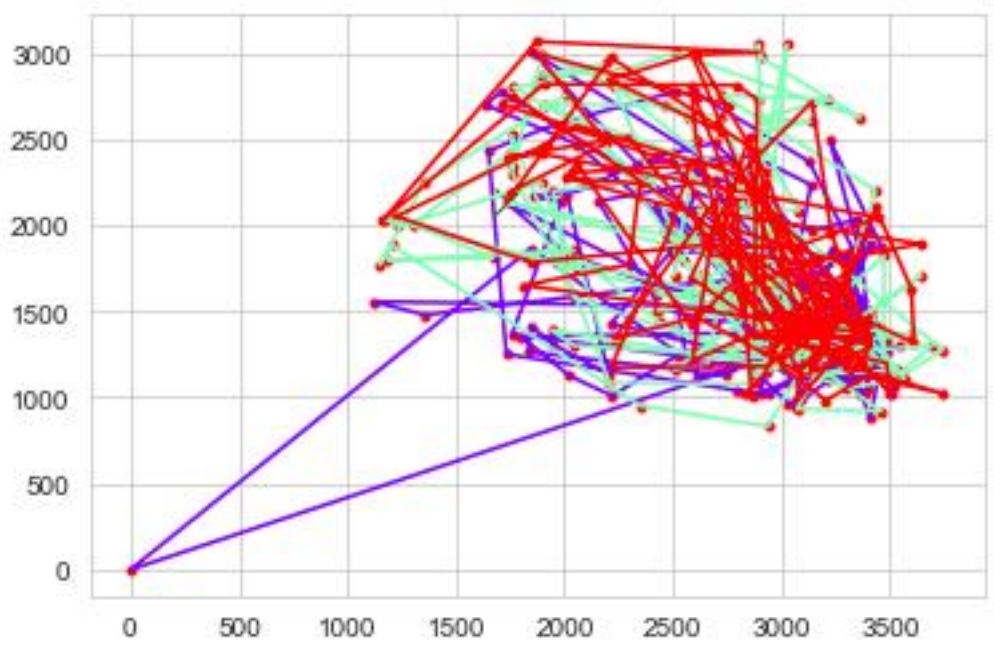
3.8 As the UPMX operator is found to be most efficient , it was tested with other large datasets and the following results were obtained:

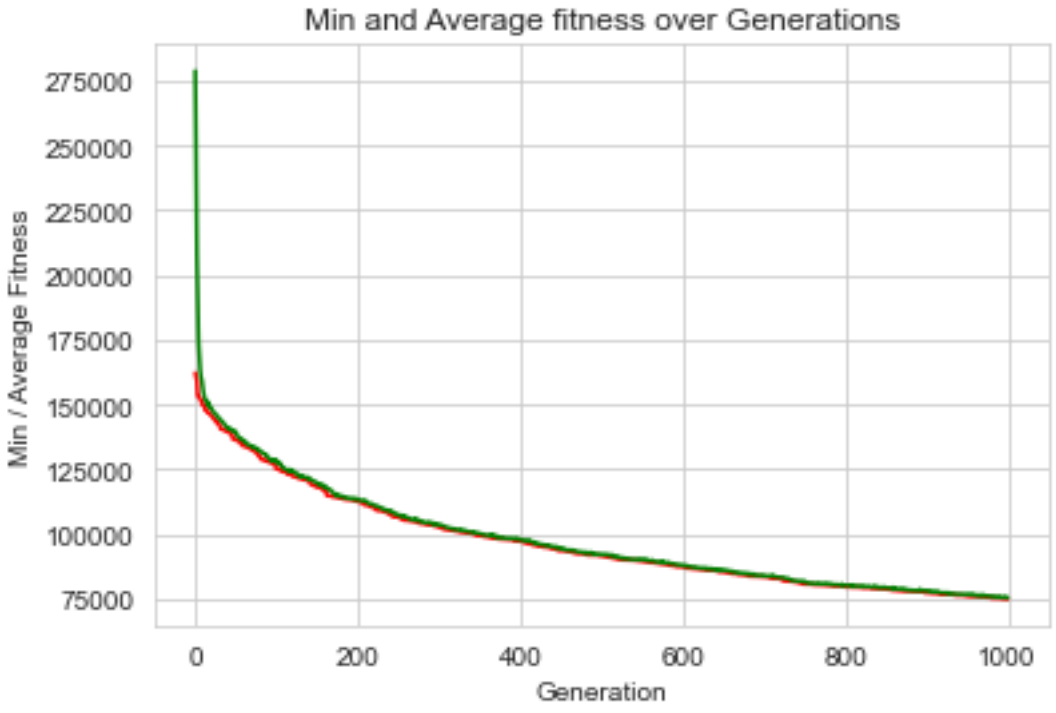
3.8.1 Dataset: “d198.tsp”(198 node problem)



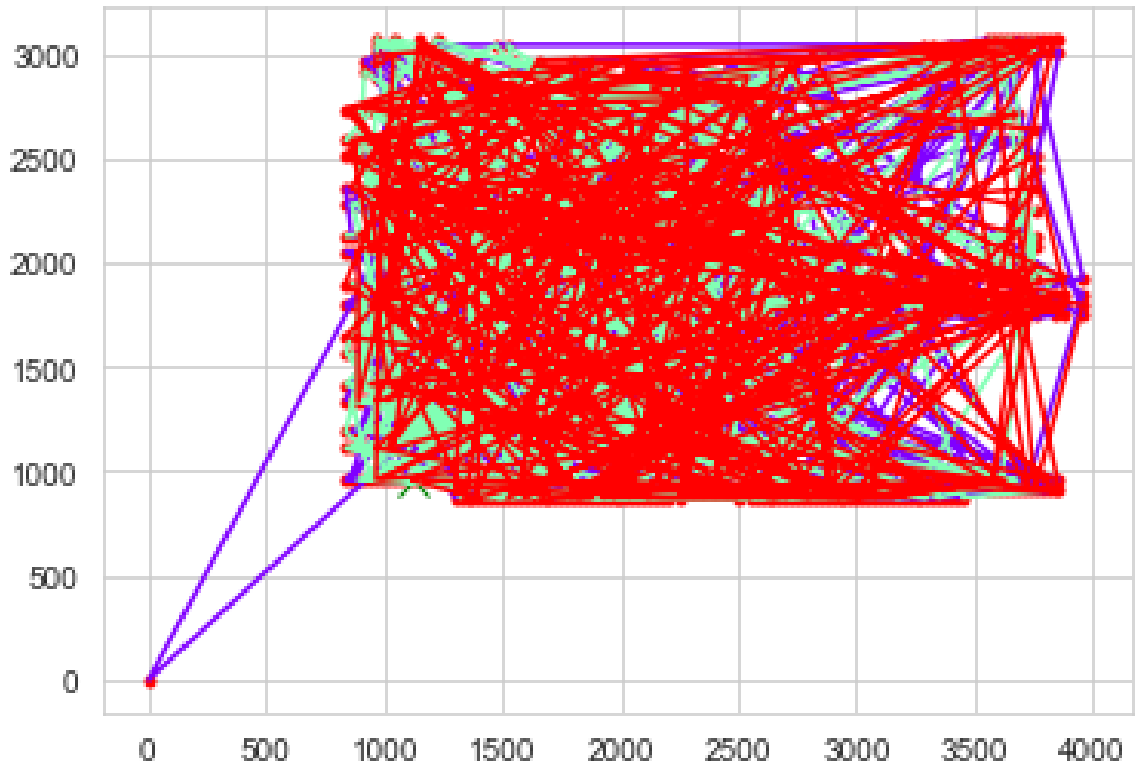


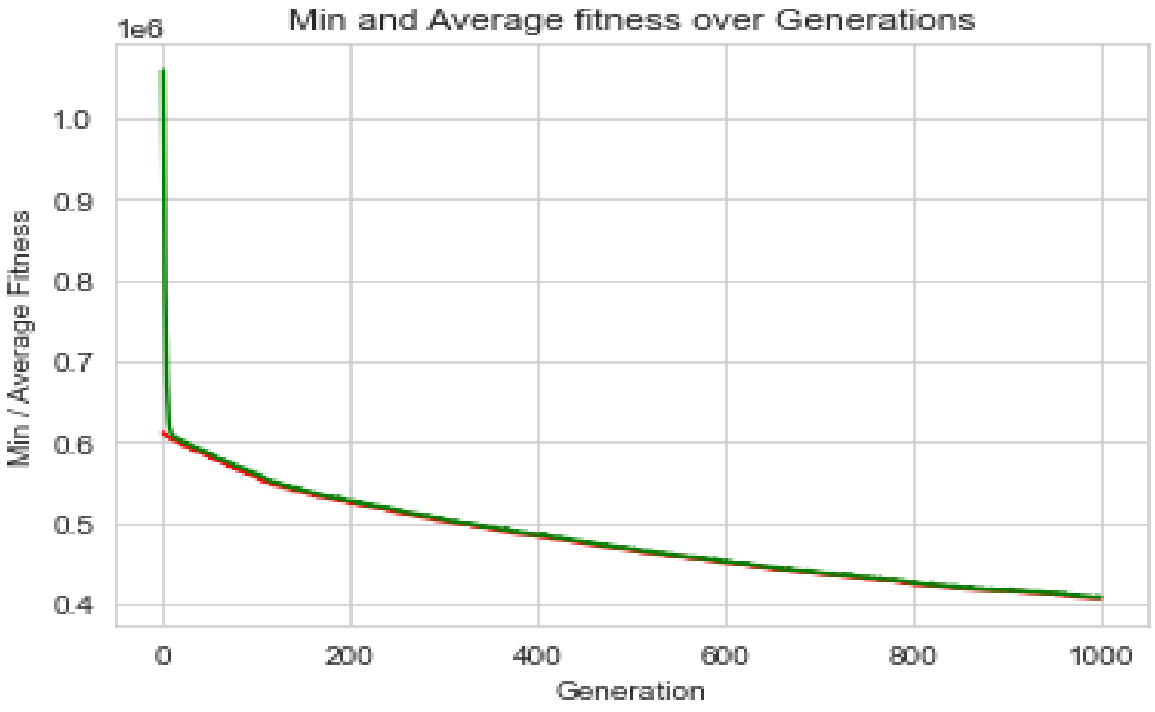
3.8.2 Dataset: “d493.tsp” (493 node problem)





3.8.3 Dataset: “d1291.tsp” (1291 node problem)





4. Conclusion and Future work:

Genetic approach is one of the most interesting methods to solve the NP-hard problems such as Hamiltonian cycles (Vertex cover), 3-SAT problem and the ones discussed above TSP and VRP. Meta-heuristic approaches can be very useful while solving NP-hard problem given that we have the resourses (computational power) and most importantly knowing which genetic operators to use. The above results have proved (on the basis of results) certain crossover operators perform better in comparison to other available and as we know that crossover operators perform 90% of the genetic changes it can be said that these changes accounted for most of the exploration and exploitation operations. According to the observations made it can be concluded that :

➔Ordered crossover (OX) operator performs better than other crossover operators in the case of The Travelling Salesman Problem(TSP) which were used in the testing.
➔Uniform partially Matched crossover (UPMX) operator performs better and is consistent with small as well large datasets fulfilling all criteria of traversal.

Proposal of new crossover operators can be made because these operators are problem specific and perform well on certain problems and yields bizarre results for other problems. So, certain genetic operators can be proposed which reduces the computational complexity of the problem and can be very problem specific and similar things can be tried out with various mutation operators as well.

5. References:

Dataset reference: *http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/*
Dataset Library: *http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf*
Code reference*: https:/ / github. com/ PacktPublishing/ Hands- On- Genetic- Algorithms- with-Python/ blob/ master/*
Documentation reference*: https://deap.readthedocs.io/*
Other references:

- *https://www.geeksforgeeks.org/traveling-salesman-problem-using-genetic-algorithm/*
- *https://www.hindawi.com/journals/cin/2017/7430125/*