METAHEURISTICS

ASSIGNMENT 1 – REPORT

Name: Sriranjani Sridharan

Student ID: R00182510

Part -1: NP-Completeness

1) Converting a Formula F into a 3SAT Formula F'

The last digit of my Student-ID(R00182510) is '0'. The Formula F to be used for the current problem is (option 'a') given below.

a. F = (z1 ∨ -z2) ∧ (-z1 ∨ z2 ∨z3 ∨z4 ∨ z5)

First clause → z1 ∨ -z2

If k = 2, we need 1 variable and clauses to make it a 3SAT

The original clause is replaced into 2 new 3SAT clauses as given below

(z1 ∨ -z2 ∨ z6)

(z1 ∨ -z2 ∨ -z6)

Second clause → -z1 ∨ z2 ∨z3 ∨z4 ∨ z5

If the number of literals k > 3, we will need k-3 extra variables and k-2 clauses.

Since, k = 5 we need 2 extra variables and 3 clauses

The original clause is replaced into 3 new 3SAT clauses as given below

(-z1 ∨ z2 ∨ z7)

(-z7 ∨ z3 ∨ z8)

(-z8 ∨ z4 ∨ z5)

The 3SAT Formula,

F' = (z1 ∨ -z2 ∨ z6) ∧ (z1 ∨ -z2 ∨ -z6) ∧ (-z1 ∨ z2 ∨ z7) ∧ (-z7 ∨ z3 ∨ z8) ∧ (-z8 ∨ z4 ∨ z5)

A solution to satisfy the Formula F', will take the below values

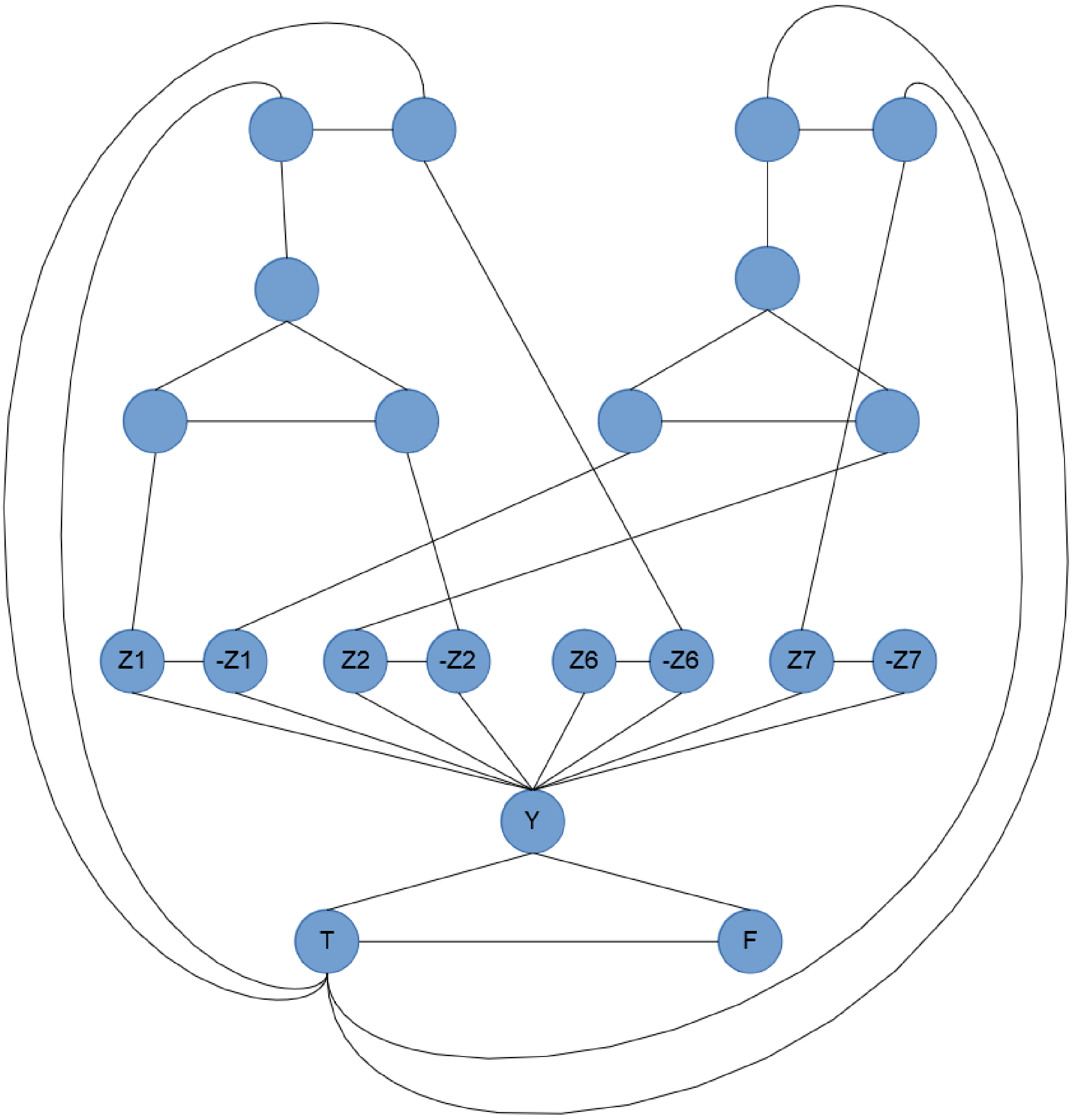
z1 = 1, z2 = 0, z3 = 1, z4 = 0, z5 = 1, z6 = 0, z7 = 1, z8 = 0

This will also be a solution to the original formula F as each clause in F has at least 1 literal that is true.

2) Converting a selection of subclauses from the above Formula F’ to a 3COL graph

The first letter of my first name starts with ‘S’. The second and third clauses of F’ was used for the problem as given below.

(z1 ∨ -z2 ∨ -z6) ∧ (-z1 ∨ z2 ∨ z7)



If the literals in the graph were assigned the below values, then the above 3 col graph is a valid reduction to the given subclauses of F’.

Z1 = T, -Z1 = F Z2 = T, -Z2 = F Z6 = T, -Z6 = F Z7 = T, -Z7 = F

Part -2: Genetic Algorithms

Problem Description and Solution overview

The Travelling Salesman problem (TSP) can be stated as: if a traveling salesman wishes to visit exactly once each of a list of m cities (where the cost of traveling from city i to city j is cij) and then return to the home city, what is the least costly route the traveling salesman can take [1].

The solution to this problem is a program that solves the TSP using Genetic Algorithms along with a detailed explanation and implementation of the following Crossover and Mutation operations.

• Uniform order-based crossover
• Partially Mapped Crossover (PMX)
• Reciprocal exchange mutation
• Inversion Mutation

In addition to the above operations, the impact of GAs with the following initial populations were also studied.

• Randomly generated population
• Nearest neighbor insertion: Choose first city randomly, each city thereafter choose city closest to the last city added to the route and append to the route

Input Problem Instances

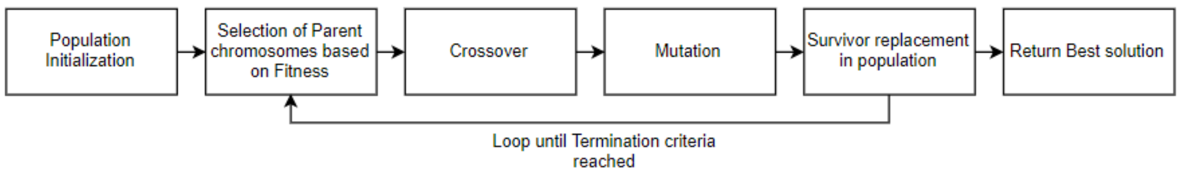
The input files used to evaluate the algorithms are inst-19.tsp, inst-20.tsp and inst-7.tsp from the TSP dataset, as the first letter of my surname is in the range S-Z.

Genetic Algorithms - Concept Description

Genetic algorithms (GA) was developed in the 1970s and has been applied to optimization and machine learning problems, based on the principles present in natural evolution [2]. GA is a subset of a larger class of evolutionary algorithms.

Similar to natural selection, genetic algorithm computation can be viewed as a process to generate an optimal solution for a problem after a number of successive generations. The set of potential solutions to a problem is termed as a population of chromosomes or a population of Individuals. The stronger chromosomes pass on their traits to future generations while the weaker ones tend to disappear as the generation evolves.

The computation of a machine learning/optimization problem using GA follows the below structure [3].


Population Initialization
Selection of Parent chromosomes based on Fitness
Crossover
Mutation
Survivor replacement in population
Return Best solution
Loop until Termination criteria reached

The initial population can be a randomly generated population or can be generated based on a heuristic solution to the problem. Once the population is determined, the three evolutionary process of selection, crossover and mutation is applied to create a new individual which will be replaced into the population to generate future generations.

The central concept behind the selection of parents from the population is based on Fitness. Determining the fitness of a chromosome is a way of quantifying the value of a given structure/solution. Given two solutions, it helps in choosing the better one. The selected parents are recombined using the crossover operator to obtain either one or two children/individuals. The process of mutation is then applied to flip a gene or genes in the chromosome. The crossover and mutation operator add to the variation in creating new solutions to the problem.

As the generation evolves with replacement of new individuals, the solution to the problem converges to produce an optimized result.

Components of GA for TSP Problem

In the TSP problem, an individual/chromosome is said to represent the route a travelling salesman can take. The fitness of an individual is equivalent to the cost of taking a route. Every city in the route is equivalent to a gene of the chromosome.

The solution to this problem is to find a route that will minimize the cost. The following operations are involved in creating the potential solutions to the problem.

1) **Initial Population generation**
   The initial population can be generated in the following two ways. Both the methods were incorporated in the experimental study that is covered in the later sections.

   - Randomly generated Population:
     Each individual to be added to the population is generated by swapping the positions of any two randomly identified genes in the chromosome. The individuals are generated as long as the desired population size is reached.
     The function name “initPopulation(self)” is used to implement the above solution in the code.

   - Nearest neighbor insertion:
     Each individual to be added to the population is determined based on a heuristic solution-based approach. The function “heuristicPopulation(self)” is used to implement this.

     To generate a single individual, an initial gene is chosen at random to create a chromosome. Thereafter, each gene to be added to the chromosome is generated based on finding the gene with the closest distance to the recently added gene. The process is repeated as long as the desired population size is reached.

2) **Selection of Parents in mating pool**

Selection of parents from the population is done so that the stronger parents have a better opportunity to reproduce than the weaker ones. Various methods can be used to implement this operator such as the Roulette Wheel, Stochastic Universal sampling, ranking etc.

For the current problem, the selection operators being used are random selection and Stochastic Universal sampling. The mating pool is a clone of the population from which the parents are selected.

- Random Selection
  Two individuals are randomly selected from the mating pool. Refer function “randomSelection(self)” in code.

- Stochastic Universal Sampling
  The fitness of each individual in the population is determined and is transformed for normalization. As the current problem is an example of minimization (finding the route with the minimum cost), the original fitness(F) should be transformed as 1/F. Using the transformed fitness values(F’), the probability of selecting a chromosome is determined as F’/sum(F’). This selection probability is the normalized fitness value(N) of the chromosome. Each chromosome is marked on a scale with a range equal to the length of the normalized fitness values that was determined. A selection pointer with uniform distance is then generated to point to the ones that will be selected from the scale of values indicating the chromosomes. The initial value of the pointer is a random value between the range [0, sum(N)/P], where sum(N)/P is the distance between the pointers pointing over the scale. Once the required number of chromosomes are selected from the scale, two random parents are selected from these stronger parent chromosomes to proceed further for crossover/mutation. Refer function “stochasticUniversalSampling(self)” in code.

3) **Crossover operator**

The variation/diversity in creating the new individuals is added by the crossover and mutation operators. The selected parents from the above step is recombined using crossover operators to produce new individuals. There are various crossover methods such as Order 1, Uniform order-based, Partially Mapped and Cycle crossover.

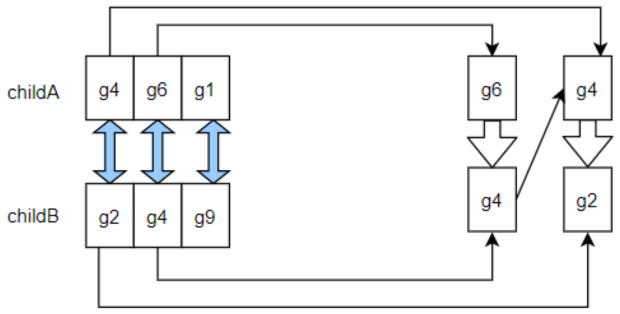
For the current problem, the following implementations are included. The crossover can generate one or two new individuals. In the current scenario, two children will be generated and passed over to the next generation. In order to maintain the population size at the same value we started with, the number of times to repeat the selection, crossover and mutation process will be reduced to half of the population size as we are creating two individuals per operation.

- Uniform order-based crossover

Genes at random positions are selected from one parent chromosome, that will remain unchanged in the newly generated chromosome. The remaining positions of the newly generated chromosome is filled in by the missing genes from the other parent in the order in which they appear. This will create one child. Similarly, the other child is created starting with the other parent. Refer function “uniformCrossover(self, indA, indB)” in code.

- Partially Mapped crossover

  Two randomly generated positions are selected from parentA and parentB, and the genes between these positions in both the parents are swapped to create two new chromosomes childA and childB. The child A contains the swapped genes from parentB between the random positions selected. Similarly, childB contains the swapped genes from parentA between the random positions selected. The remaining positions of childA and childB are filled in a two-step process.
  - In the first step, only the missing genes in childA is filled from the parentA in the same positions as they appear in the parentA. Simialrly, childB is filled in for the required positions.
  - The second step handles a mapping process to fill in the positions of the genes from parentA that are already present in the childA chromosome using the below mapping procedure. The mapping is between the genes that were swapped initially in childA and childB as shown in the below figure. Say a gene, g6 from the parentA needs to be added to the childA. Since g6 is already present in childA(swapped positions), its position from the parentA is filled from the mapped position in childB which is g4. If g4 is also present in childA, then its mapped value g2 from childB is filled in. The chain continues as long as we find a value that is not in childA.



Similarly, childB is also generated. The two newly generated chromosomes then move further to the mutation process. Refer function “pmxCrossover(self, indA, indB)” in code.

4) Mutation operator

The mutation is being applied on a chromosome level rather the gene level and this is determined by the parameter named mutation rate. The higher the mutation rate, the greater the chances of the chromosomes to be mutated.

The mutation operators currently implemented in the problem is explained below.

- Inversion Mutation

  Two random points in the chromosome are selected and the genes in between these points are reversed to create a new individual. Refer function, "inversionMutation(self, ind)" in code.

- Reciprocal Exchange Mutation

  Two random positions in the chromosome are selected and the genes in these positions are swapped to create a new individual. Refer function. "reciprocalExchangeMutation(self, ind)" in code

5) **Replacement**

   A full replacement is implemented in the current problem. Every individual after mutation is replaced into the population with its updated fitness.

The above operations are repeated until the number of generations to study is generated. The parameters such as setting the population size, mutation rate and the number of iterations to converge into a possible solution can be varied in order to study the impact of it on approaching a solution to the problem.

Experimental Design

The experiment was designed to study the results from 8 different configurations that includes the combination of operators discussed in the previous section. In addition, the experiment was repeated for configurations 3 – 8 to investigate the impact of varying population size and the mutation rate.

The 8 configurations studied are given below.

```
Configuration 1 - Random population, Random Selection, Uniform Crossover,
Inversion mutation

Configuration 2 - Random population, Random Selection, PMX Crossover, Reciprocal
Exchange mutation



Configuration 3 - Random population, Stochastic Universal sampling, uniform
crossover, Reciprocal Exchange mutation

Configuration 4 - Random population, Stochastic Universal sampling, PMX
Crossover, Reciprocal Exchange mutation

Configuration 5 - Random population, Stochastic Universal sampling, PMX
Crossover, Inversion mutation

Configuration 6 - Random population, Stochastic Universal sampling, Uniform
crossover, Inversion mutation
```

```
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX
crossover, Reciprocal Exchange

Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform
crossover, Inversion mutation
```

The basic configurations 1 and 2 were run together as a single experiment with a population size of 100 and mutation rate 0.1.

The remaining configurations 3 – 8 were run together for different experiments with varying population size and mutation rate.

Each of the above configurations were run 5 times for an experiment with a defined population size, mutation rate and number of Iterations. The results were recorded. The overall performance for each configuration displayed in this report is the average performance resulted from the 5 runs of the same configuration and setup.

Experimental Results and Evaluation

The input files used for evaluation are inst-19.tsp, inst-20.tsp and inst-7.tsp. The reported performance results for each configuration is the overall average performance calculated after 5 iterations per execution. The configuration with a lower solution cost indicates a better performance for the TSP problem.

Basic Experiments and Evaluations

A basic evaluation discusses configurations 1 and 2 with population size 100, mutation rate 0.1 and 500 iterations for all the input files.

File: inst-19.tsp		
	Best solution Mean	Best solution median
Configuration 1 - Random population, Random Selection, Uniform Crossover, Inversion mutation	17715593.305488	17648036.923368
Configuration 2 - Random population, Random Selection, PMX Crossover, Reciprocal Exchange mutation	17885108.467121	17891099.278088

File: inst-20.tsp		
	Best solution Mean	Best solution median
Configuration 1 - Random population, Random Selection, Uniform Crossover, Inversion mutation	123712081.965292	123908308.792073
Configuration 2 - Random population, Random Selection, PMX Crossover, Reciprocal Exchange mutation	123810045.958419	122851419.084292

File: inst-7.tsp		
	Best solution Mean	Best solution median
Configuration 1 - Random population, Random Selection, Uniform Crossover, Inversion mutation	307495834.253198	306341353.654208
Configuration 2 - Random population, Random Selection, PMX Crossover, Reciprocal Exchange mutation	307033402.207348	306846164.040906

For the first two files inst-19 and inst-7 containing 128 and 364 records respectively, the average performance of Configuration 1 is a little higher than Configuration 2. Whereas for the file inst-7 with 832 records, the average performance of it is the vice versa, though all the differences are small. As the population size and mutation rate is lower and the selection is random, there might not be any significant changes, to distinguish between the configurations.

The overall results are more or less similar with respect to the impact of both the configurations in all the three files.

Extensive Experiments and evaluation

An extensive evaluation discusses the configurations 3 to 8 with varying population sizes or mutation rates for all input files. The number of iterations for all the experiments is 500.

1) Population size – 100, Mutation rate – 0.1

   For the inst-19 file, configurations 3 and 4 has not added much value to the performance than the basic evaluation discussed above. However, the configurations 6 and 7 shows better results. The difference observed in these two pairs of combinations might be due to the mutation operator. The Inversion mutation operator has generated better results when compared to the reciprocal exchange mutation.

   The configurations 7 and 8 has significantly reduced the route cost, and the large part of the performance improvement is due to the heuristic based initial population. The different crossover and mutation operations has not added any further value to these results.

File: inst-19.tsp		
	Best solution Mean	Best solution median
Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	17949880.81411750	17951491.30034990
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	17904593.86519860	17941550.28991230
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	17795066.93656940	17940461.90992780

Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	17497372.51463190	17545638.39784550
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	4099688.267594	4099688.267594
Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	4099688.267594	4099688.267594

The inst-20 file follows a different patter of results. The Heuristic based configurations have shown minor differences in their performance. The configuration 7 with the PMX crossover and Reciprocal exchange mutation generated a better solution. The experiments might need to be repeated further to analyze which of the operations has added value to the performance of Configuration 7.

With respect to the other 4 configurations, the ones with uniform crossover mutation has shown better results for this file.

File: inst-20.tsp		
	Best solution Mean	Best solution median
Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	122424760.005490	123261271.382765
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	124038612.046314	124493114.158769
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	123469916.185098	124038768.660267
Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	122946212.911361	122584277.287019
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	7662934.906305	7664515.681772
Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	7665464.534373	7642936.515290

The inst-7 file has the largest number of records (800 records). The heuristic based configurations (7 and 8) have generated more or less similar results. The results from the other configurations (3 to 6) are also closer to one another. Configuration 1 has been showing to perform better with larger input files, than the other configurations with random initial solution.

File: inst-7.tsp		
	Best solution Mean	Best solution median
Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	304250310.185133	305008997.164997
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	306789292.862084	307006176.659049
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	307450493.702291	308817096.328547
Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	306810143.924109	306018317.800163
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	12124896.913877	12125957.836089
Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	12122540.167266	12113601.961107

2) Population size – 300, Mutation rate – 0.1

The population size was increased from 100 to 300, keeping the mutation rate unchanged. This has improved the overall performance for all the files. Increasing the population results further might help us converge to a better optimized solution.

From the three files, out of all the configurations using random initial solution (3 to 6), configuration 4 shows a better performance with the combination of Stochastic universal sampling, PMX crossover and Reciprocal exchange mutation.

File: inst-19.tsp		
	Best solution Mean	Best solution median
Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	17307700.59965340	17406446.24508940
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	17271581.54179290	17496855.86286050
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	17483907.23019290	17531404.83452910
Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	17559628.85019270	17553580.71876560

Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	4099688.26759420	4099688.26759420
Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	4099688.26759420	4099688.26759420

File: inst-20.tsp		
	Best solution Mean	Best solution median
Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	121944299.58832800	122935638.10320100
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	121503209.77319400	121581649.13165100
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	121892954.61613000	122371783.24008200
Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	121722302.30207500	121700168.23549500
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	7635941.82329988	7636830.63225449
Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	7640841.93149854	7636830.63225449

File: inst-7.tsp		
	Best solution Mean	Best solution median
Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	304691018.17377800	305655805.12594000
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	304260452.39564200	305125507.02183500
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	304423888.27325000	303701810.91625900
Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	304458486.62998200	304248476.23084200
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	12084941.99397980	12065496.84648570

Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	12076184.33986540	12065496.84648570

3) Population size – 300, Mutation rate – 0.5

Keeping the population size unchanged from the previous experiment, the mutation rate was increased from 0.1 to 0.5. This change hasn't significantly affected the Heuristic based solution results for the three files.

With respect to the other configurations (3-6), the inst-19 file shows a better performance with configurations 5 and 6. The change in mutation rate in the implementation of Inversion mutation seems to have generated better results.

For the inst-20 and the inst-7 file, all the random population configurations (3-6), have shown an improvement in performance, with the increase in mutation rate.

File: inst-19.tsp		
	Best solution Mean	Best solution median
Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	17392436.90756560	17420669.15785290
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	17287786.20204950	17304794.18670520
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	17137681.11887410	17145134.43355820
Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	17467362.95827010	17542894.31600290
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	4099688.26759420	4099688.26759420
Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	4117337.56492003	4099688.26759420

-File: inst-20.tsp		
	Best solution Mean	Best solution median

Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	119873072.10317800	119856262.27264600
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	119320442.44142100	119757292.10996100
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	119123240.18825600	119707211.05463900
Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	119544961.52225100	119326404.28489200
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	7648507.35983233	7636830.63225449
Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	7631833.88900113	7628502.72683223

File: inst-7.tsp		
	Best solution Mean	Best solution median
Configuration 3 - Random population, Stochastic Universal sampling, uniform crossover, Reciprocal Exchange mutation	303098595.50449900	303624485.37464000
Configuration 4 - Random population, Stochastic Universal sampling, PMX Crossover, Reciprocal Exchange mutation	302513933.85091800	302527054.96830300
Configuration 5 - Random population, Stochastic Universal sampling, PMX Crossover, Inversion mutation	303368714.20572600	303542959.52256000
Configuration 6 - Random population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	301663867.62510900	302200749.26755900
Configuration 7 - Heuristic population, Stochastic Universal sampling, PMX crossover, Reciprocal Exchange	12068468.38328660	12065496.84648570
Configuration 8 - Heuristic population, Stochastic Universal sampling, Uniform crossover, Inversion mutation	12065879.99921490	12065496.84648570

From the overall observation of the different experiments, the heuristic based population has generated constant results across the same files. However, the initial solution has already added to the increased performance. The experiments can be further extended towards a higher population size and iterations to see if better results are generated from the same.

The inversion mutation operator along with the Stochastic Universal sampling has shown to contribute towards a better performance in some instances. As the population size increases further, and with an optimum mutation rate, the results might converge to produce better solutions for the TSP problem.

References

The below resources were referred for some of the theoretical content produced in this report.

[1] Posted Assignment 1 Document

[2] Metaheuristics: From Design to Implementation by El-Ghazali Talbi

[3] Lecture slides