CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Metaheuristic Optimization

Assignment 1

Student Id: R00182505

Name: Dheeraj **A**limchandani

Part 1: NP-completeness

1. Convert the formula F below into a 3SAT formula F', find a solution to F' and verify that this is a solution to

b. If the last digit of your student id is either 3, 4 or 5 use

F = ($w_1 \vee -w_2 \vee w_3 \vee -w_4 \vee -w_5$) $\wedge$ ($-w_3 \vee w_4$)

SOLUTION:

The below is the procedure for converting the formula F into a 3SAT formula F'

Let C_i be the clause in the given SAT formula F, where i can range from i to n.

In formula F, we have two clauses and they can be represented by C_1 and C_2.

Where,

C_1 = (w1 ∨ -w2 ∨ w3 ∨ -w4 ∨ -w5)

C2 = (-w3 ∨ w4)

The above-mentioned clauses C_i can be replaced by a conjunction of clauses in X_i.

Where,

- All Clauses in X_i must contain 3 literals.
- C_i is satisfiable iff constructed X_i is satisfiable and vice versa.

Truth Table:

w1	w2	w3	w4	w5	F
F	F	F	F	F	T
F	F	F	F	T	T
F	F	F	T	F	T
F	F	F	T	T	T
F	F	T	F	F	F
T	F	T	F	F	F
T	F	T	F	T	F
T	F	T	T	F	T
T	F	T	T	T	T

Table 1

Reduction of C_1 to X_1:

C_1 = (w1 ∨ -w2 ∨ w3 ∨ -w4 ∨ -w5)

Here,

w1, w2, w3, w4, w5 are literals

Let K = Number of literals in C_1

Therefore K = 5

Since K > 3, we need to introduce K-3 new variables (y_i^1, y_i^2, y_i^3, y_i^4..... y_i^{k-3}) and replace C_1 with K-2 clauses to generate X_1

Number of new variables = 2

C_1 to be replaced with 3 clauses each having 3 literals.

Replacing C_1 by a sequence of clauses to generate X_1:

X_1 = (w1 ∨ -w2 ∨ y_1^1) ∧ (-y_1^1 ∨ w3 ∨ y_1^2) ∧ (-y_1^2 ∨ -w4 ∨ -w5)

Reduction of C_2 to X_2:

C_2 = (-w3 ∨ w4)

Here,

w3, w4 are literals

Let K = Number of literals in C_2

Therefore K = 2

Since K = 2, we need to introduce a new variables y_i^1 and replace C_2 with 2 clauses to generate X_2

Number of new variables = 1

C_2 to be replaced with 2 clauses each having 3 literals.

Replacing C_2 by a sequence of clauses to generate X_2:

$X_2 = (-w3 \vee w4 \vee y_2^1) \wedge (-w3 \vee w4 \vee -y_2^1)$

The 3SAT formula F' generated is conjunction of X_1 and X_2:

$F' = X_1 \wedge X_2$

$F' = (w1 \vee -w2 \vee y_1^1) \wedge (-y_1^1 \vee w3 \vee y_1^2) \wedge (-y_1^2 \vee -w4 \vee -w5) \wedge (-w3 \vee w4 \vee y_2^1) \wedge (-w3 \vee w4 \vee -y_2^1)$

Truth Table:

w1	w2	w3	w4	w5	F	y_1^1	y_1^2	y_2^1	F'
F	F	F	F	F	T	F	F	F	T
F	F	F	F	T	T	F	F	F	T
F	F	F	T	F	T	F	F	F	T
F	F	F	T	T	T	F	F	F	T
F	F	T	F	F	F	F	F	T	F
T	F	T	F	F	F	F	F	T	F
T	F	T	F	T	F	F	F	F	F
T	F	T	T	F	T	F	T	F	T
T	F	T	T	T	T	T	F	F	T

Table 2

2. Convert the following subclauses in your F' to a 3Col graph

The last two clauses of F' if the first letter of your first name is in the range A-I

Explanation:

In a 3Col graph, 3 special vertices are created T(true), F(false) and Y(Neutral)

If a color is assigned to each of the vertices, then:

T represents a color say Blue

F represent a color say Red

X represents a color say Yellow

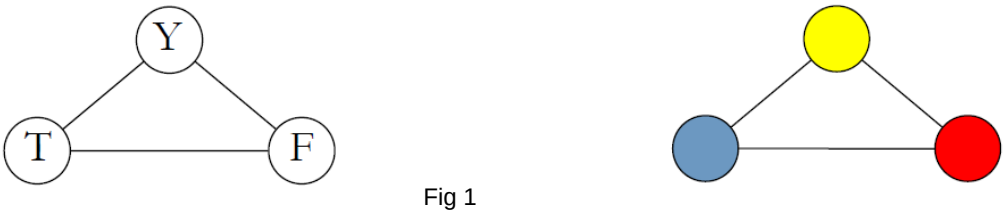


Fig 1

Reference: Lecture Slides

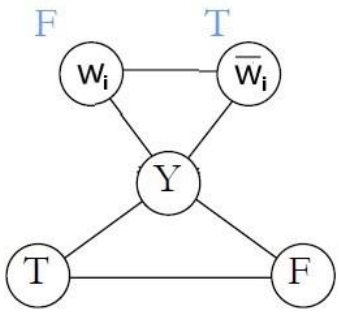
Generally, for the literal in the 3SAT:

- Either w_i has color of T and $-w_i$ has color of F

  Or

- w_i has color of F and $-w_i$ has color of T

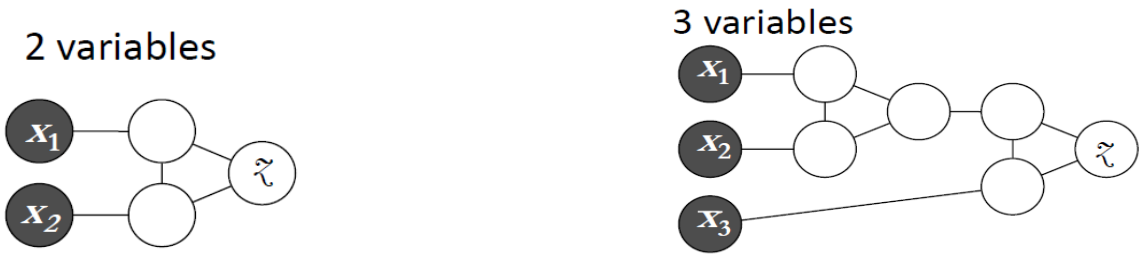
Every literal in the 3SAT should be connected to vertex Y in order to form the structure like Fig 1. Also no two adjacent vertex can have the same color



Reference: Lecture Slides

Fig 2

The disjunction between the literals can be solved using an OR- gadget



Reference: Lecture Slides

Fig 3

2 Variable

- Z can be colored T if either of X_1 or X_2 is colored T
- Z must be colored F, if X_1 and X_2 are colored F

3 Variable

- If at least one literal (X_1, X_2 or X_3) is colored T, then Z can be colored T

SOLUTION:

Converting the below clauses to 3Col graph:

$(-w3 \vee w4 \vee y_2^1) \wedge (-w3 \vee w4 \vee -y_2^1)$

Where:

w3 = T, w4 = T, y_2^1 = T

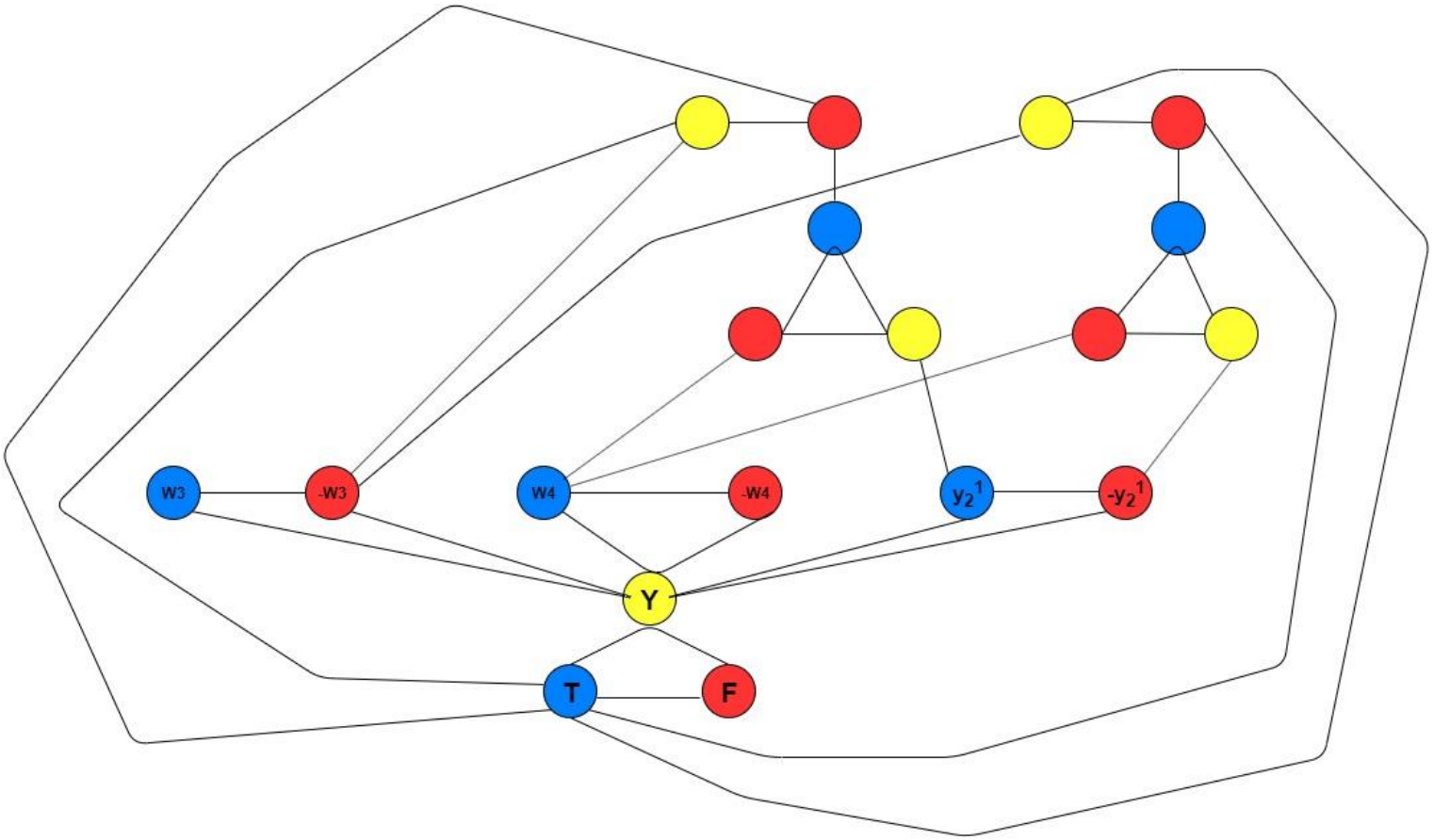


Fig 4

Part 2: Genetic Algorithms

Problem instances

In this project, you will use the following problem instances to evaluate the performance of your algorithms:

if the first letter of your surname is in the range A-I inst-0.tsp, inst-13.tsp, and inst-5.tsp

Solution:

A simple genetic algorithm is being applied over the Travelling salesman problem, in order to generate the optimized solution. The solution here is defined as the optimal path the salesman should take in order to cover every city only once and reach the starting point. The genetic algorithm will try to generate the optimal path the salesman can take in order to reduce to path cost.

A genetic algorithm is defined as solving the problem using natural selection. The idea of the genetic algorithm is to produce the optimized result based on the concept of survival of fittest, natural selection and genetics.

The genetic algorithm takes an initial population and performs various operations over it which include fitness calculation, crossover, mutation and survivor selection.

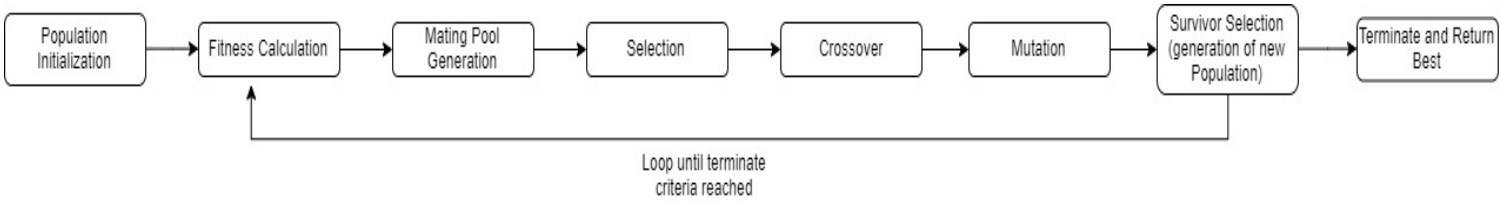


Fig 5

1. Initial Population (Initial Solution)

Generation of the initial population is the first step of a genetic algorithm. The members of the initial population can be referred as Chromosomes with each chromosome having a fitness factor associated with it. The fitness of the chromosome defines how good the chromosome in the population.

In case of TSP, the initial population is defined as the list of paths that will be used by the genetic algorithm in order to generate the optimal solution i.e. path with least cost.

Here, two approaches have been used in order to generate the initial population

- Randomly generated population
- Heuristic Approach: Nearest Neighbour Insertion (Choose first city randomly, each city thereafter choose city closest to the last city added to the route and append to the route)

Randomly generated population:

1. Generate a list of all the cities.
2. Calculate the length of the list containing all the cities, say n
3. Shuffle the cities inside the list for the n times in order to generate a random city chromosome
4. Repeat step 3 for P times, where P is the size of the population

Heuristic Approach:

1. Generate a list of all the cities
2. Select a random city as the starting point
3. Calculate the path cost of all the cities from the randomly selected city
4. Append the city with the least path cost after the city selected in step 2
5. Repeat step 3 and 4 for the city generated in step 4
6. Repeat above steps for P times, where P is the size of the population

2. Selection

The selection is the process where the parent chromosomes are selected for mating in order to produce offspring. The offspring are then used to create a new generation.

Here, two approaches have been used in order to generate the solution

- Random Selection
- Stochastic Universal Sampling

In order to perform the selection over the population a mating pool is created from the population over which the selection techniques are applied. The mating pool contain the instance of the population over which various operations are performed in order to create new generation.

Random Selection:

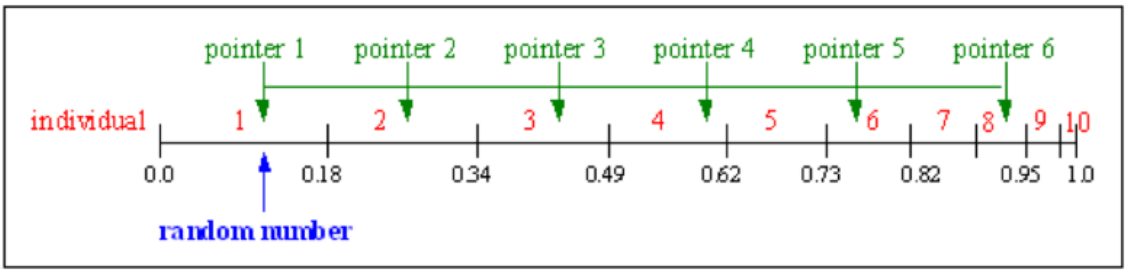
1. Generate a mating pool as a copy of the population
2. Randomly select the parent chromosomes from the mating pool
3. The number of parent chromosomes selected depends upon how many parents are required to perform the mating in order to generate the offspring
4. In order to generate two offspring, two parents are randomly selected from the mating pool.
5. The random selection of parents ensures the diversity in the next generation

Stochastic Universal Sampling:

In Stochastic Universal Sampling procedure, the parent chromosomes are selected based on fitness value. In case of TSP, the parent with least path cost is the fittest and must have the highest probability of getting selected. The Stochastic universal sampling ensures selection of fitter candidate.

1. Compute the fitness of every individual in the population
2. In case of TSP, there is a need to perform minimisation of each individual fitness in order to get the individual with least path cost selected.
3. Perform Minimisation of fitness:
    a. Get the maximum fitness value from the list of individual fitness
    b. In order to prevent non selection of individuals with maximum fitness, add a token amount of 1 to the maximum fitness value
    c. Subtract the (Maximum fitness + 1) from individual's fitness
4. Generate the selection probability for individual by dividing the minimized fitness with the sum of minimized fitness

5. Assign every individual a range equal in length to its minimized fitness and the starting point that is after the end point of the previous individual (e.g. first individual 0< I <=0.3, 0.3< II <= 0.5 and so on till the fitness reaches 1)
6. Compute the distance P between successive points: P = F/N; where F is the sum of minimized fitness values and N equal to the population size
7. Generate a random number between 0 and P as the starting point for the ruler. The ruler has n equally spaced points, each P distance apart
8. Select the chromosomes whose range contains a marker



Reference: Lecture Slides

Fig 6

3. Crossover

The process of selecting 2 individuals (Parents) to obtain two new individuals (the children) is called crossover.

Here we will be looking at two types of crossover techniques with respect to Travelling salesman problem.

- Uniform Crossover
- PMX Crossover

Uniform crossover (TSP):

1. Select two parents from the mating pool over which the crossover is to be performed
2. Generate a list of random numbers which will be the position of the genes, ensuring that the genes of the parent at these positions will not change
3. Update the offspring with the genes from alternative parent that are not already present in the offspring in the order they appear
4. This will generate two children.

Partially Mapped Crossover (PMX):

1. Select two parents from the mating pool over which the crossover is to be performed
2. Generate random indices, the genes at these indices should be present in the offspring of the alternative parent. Example, if the indices are 2,5 then the offspring A will have genes from Parent B at location 2 ,5. Similarly the offspring B will have genes from Parent A at location 2,5.
3. Create a mapping space for the reversed genes present in the offspring
4. Select the gene from parent to be inserted into the offspring.
5. If the gene is already present in the offspring, use the mapping cycle generated in step 3 to find the appropriate gene
6. This will generate two children.

4. Mutation

The process is defined as changing the gene or a set of genes present in the chromosome based on a certain probability called P_m (Mutation Probability) to generate a new chromosome which ensure diversity.

A random number is generated in order to decide if the Mutation in the chromosome is required or not. If the value of the generated random number is greater than P_m then no mutation is performed on the chromosome else the chromosome is mutated.

Here, two approaches have been followed to perform Mutation over a TSP

- Reciprocal Exchange
- Inversion Mutation

Reciprocal Exchange:

1. Define a mutation probability P_m in order to perform Mutation
2. If the P_m is greater than the generated random number, then the chromosome undergoes mutation
3. Select two locations in the chromosome that need to be mutated
4. Exchange the genes present at these locations with each other
5. A new mutated individual is generated

Inversion Mutation:

1. Define a mutation probability P_m in order to perform Mutation
2. If the P_m is greater than the generated random number, then the chromosome undergoes mutation
3. Select a range of indices in the chromosome
4. Reverse the order of the cities in that range, like 5,3,2,8 will be changed to 8,2,3,5
5. A new mutated individual is generated

Evaluation of Genetic Algorithm for Travelling Salesman Problem:

The call to the Genetic Algorithm is made by passing the below set of parameters.

The Parameters are mentioned in order they appear in the call

Parameter List:

1. Filename
2. Population Size
3. Mutation Rate
4. Maximum Iterations
5. Initial Solution = {0: Random, 1: Heuristic}
6. Selection/Mating Pool = {0: Random, 1: Stochastic}
7. Crossover type = {0: Uniform Crossover, 1: PMX Crossover}
8. Mutation Type = {0: Inversion Exchange, 1: Reciprocal Exchange}

The below runs have been performed with

- Population Size = 100
- Mutation Rate = 0.1
- Max. Iterations = 500

1.1. Problem Instance: inst-0.tsp

Configuration	Initial Solution	Crossover	Mutation	Selection
1	Random	Uniform Crossover	Inversion Mutation	Random Selection

Result:

```
ga = BasicTSP(inst-0.tsp, 100, 0.1, 500, 0, 0, 0, 0) => Configuration 1
Best initial sol: 22660915.870452467
iteration: 1 best: 22575805.310351096
iteration: 2 best: 22384124.58659526
iteration: 2 best: 21879392.23622143
iteration: 146 best: 21863425.109339617
iteration: 223 best: 21834213.447653532
iteration: 377 best: 21825705.565501373
Total iterations: 500
Best Solution: 21825705.565501373
```

Configuration	Initial Solution	Crossover	Mutation	Selection
2	Random	PMX Crossover	Reciprocal Exchange	Random Selection

Result:

```
ga = BasicTSP(inst-0.tsp, 100, 0.1, 500, 0, 0, 1, 1) => Configuration 2
Best initial sol: 22847093.082295213
iteration: 1 best: 22808902.487161294
iteration: 1 best: 22755466.652624987
iteration: 8 best: 22596327.39771201
iteration: 8 best: 21913952.422215622
iteration: 118 best: 21788078.984594848
iteration: 118 best: 21674639.242117763
iteration: 216 best: 21663320.33557625
Total iterations: 500
Best Solution: 21663320.33557625
```

Configuration	Initial Solution	Crossover	Mutation	Selection
3	Random	Uniform Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-0.tsp, 100, 0.1, 500, 0, 1, 0, 1) => Configuration 3
Best initial sol: 22886068.64860538
iteration: 0 best: 22516461.110894207
iteration: 1 best: 22453699.268370792
iteration: 2 best: 22402856.100089893
iteration: 3 best: 22328000.83147648
iteration: 6 best: 22289146.293337077
iteration: 12 best: 22157634.46091007
iteration: 14 best: 22111604.861584924
iteration: 14 best: 22111537.444386072
iteration: 17 best: 22092721.274437405
iteration: 18 best: 22012149.77696025
iteration: 21 best: 21732993.8092514
iteration: 43 best: 21237337.365332413
iteration: 70 best: 21183760.479646575
iteration: 130 best: 21123242.82324797
iteration: 131 best: 21082321.30993389
iteration: 150 best: 21024091.51920423
iteration: 168 best: 20931573.058328267
iteration: 173 best: 20777675.48475249
iteration: 177 best: 20631727.670333944
iteration: 189 best: 20504870.861805994
iteration: 199 best: 20394066.173724215
iteration: 214 best: 20126734.232214708
iteration: 225 best: 19790874.29048226
iteration: 248 best: 19652598.86416818
iteration: 255 best: 19337298.78953982
iteration: 258 best: 19044557.56750981
iteration: 280 best: 18754220.92965149
iteration: 290 best: 18517291.30118629
iteration: 302 best: 18503786.43576676
iteration: 333 best: 18212514.85577832
iteration: 349 best: 18193966.64865517
```

```
iteration: 352 best: 17878371.623039637
iteration: 373 best: 17659689.942933306
iteration: 387 best: 17649688.743698742
iteration: 388 best: 17444678.92116526
iteration: 395 best: 17375223.49338467
iteration: 403 best: 17357293.32798947
iteration: 403 best: 17059174.123856038
iteration: 404 best: 16860576.46273848
iteration: 408 best: 16594199.240656896
iteration: 438 best: 16451141.700147297
iteration: 442 best: 15991990.434037982
iteration: 459 best: 15972203.155799385
iteration: 462 best: 15781534.15930021
iteration: 477 best: 15744810.325517647
iteration: 492 best: 15736331.57344456
iteration: 492 best: 15542988.732507162
Total iterations: 500
Best Solution: 15542988.732507162
```

Configuration	Initial Solution	Crossover	Mutation	Selection
4	Random	PMX Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-0.tsp, 100, 0.1, 500, 0, 1, 1, 1) => Configuration 4
Best initial sol: 23012973.680887677
iteration: 0 best: 22776783.25554856
iteration: 0 best: 22319684.403304577
iteration: 8 best: 21767351.0706664
Total iterations: 500
Best Solution: 21767351.0706664
```

Configuration	Initial Solution	Crossover	Mutation	Selection
5	Random	PMX Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-0.tsp, 100, 0.1, 500, 0, 1, 1, 0) => Configuration 5
Best initial sol: 22333309.479640435
iteration: 6 best: 22308523.586315416
iteration: 9 best: 21882983.86831303
Total iterations: 500
Best Solution: 21882983.86831303
```

Configuration	Initial Solution	Crossover	Mutation	Selection
6	Random	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-0.tsp, 100, 0.1, 500, 0, 1, 0, 0) => Configuration 6
Best initial sol: 22923572.532410886
iteration: 1 best: 22899676.43871158
iteration: 2 best: 22895660.77738882
iteration: 3 best: 22751499.327354733
iteration: 4 best: 22542709.409422938
iteration: 5 best: 21917423.540461235
iteration: 41 best: 21702575.285882667
iteration: 78 best: 21688546.45471455
iteration: 79 best: 21586688.29835017
iteration: 99 best: 21440111.19600911
iteration: 102 best: 21082362.514791094
iteration: 111 best: 20772797.127014495
iteration: 156 best: 20647332.288178287
iteration: 157 best: 20259170.888847996
iteration: 165 best: 20187322.1262012
iteration: 171 best: 20085942.10267234
iteration: 173 best: 20065326.86047129
iteration: 173 best: 19525736.75131744
iteration: 348 best: 19377748.041368447
iteration: 464 best: 19330987.525509275
iteration: 471 best: 19241970.27061434
iteration: 483 best: 19209204.63195392
iteration: 494 best: 19199746.41837105
iteration: 499 best: 18552891.310645808
Total iterations: 500
Best Solution: 18552891.310645808
```

Configuration	Initial Solution	Crossover	Mutation	Selection
7	Heuristic	PMX Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-0.tsp, 100, 0.1, 500, 1, 1, 1, 1) => Configuration 7
Best initial sol: 4146917.8682248485
Total iterations: 500
Best Solution: 4146917.8682248485
```

Configuration	Initial Solution	Crossover	Mutation	Selection
8	Heuristic	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-0.tsp, 100, 0.1, 500, 1, 1, 0, 0) => Configuration 8
Best initial sol: 4126111.863870551
Total iterations: 500
Best Solution: 4126111.863870551
```

1.2. Result Analysis: inst-0.tsp

It can be observed from the output that different configurations lead to a different optimized path cost.

The above test was performed with two set of initial population Random and Heuristic.

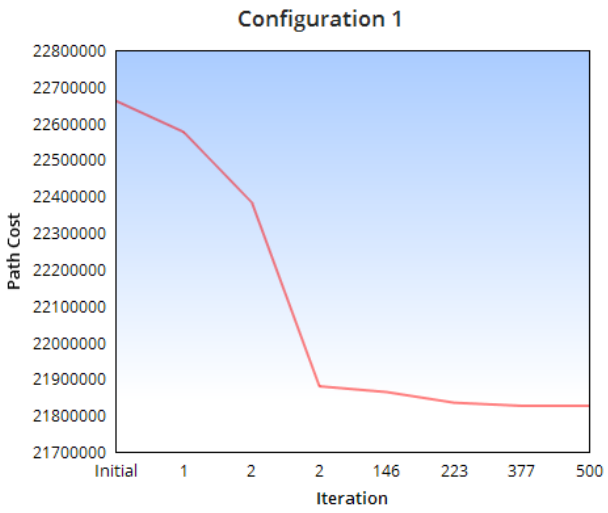
In configuration from 1 to 6, the initial population selection was Random which generated the best initial solution i.e. the path cost on the population generated randomly**.** The genetic algorithm was run over this initial path solution with a combination of various selection, crossover and Mutation operators. With the Population Size =100, Mutation Probability = 0.1 and Iterations = 500.

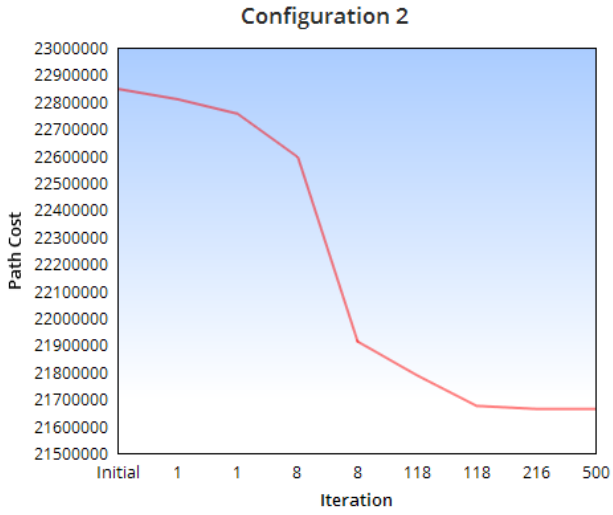
Best Result:

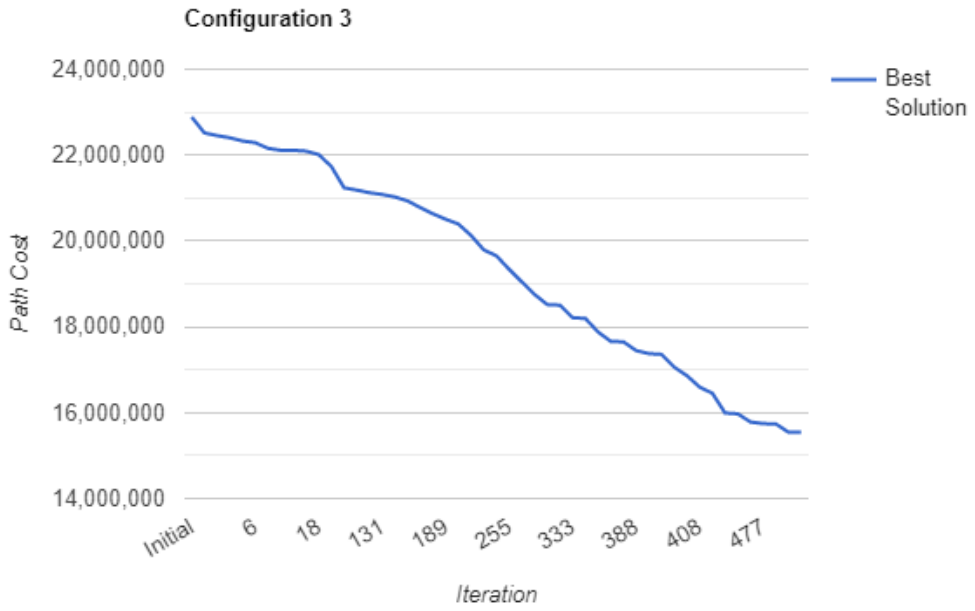
The Genetic Algorithm performs extremely well in case of Configuration 3 (Random Initial Population, Stochastic selection, Uniform Crossover, Reciprocal Exchange Mutation).

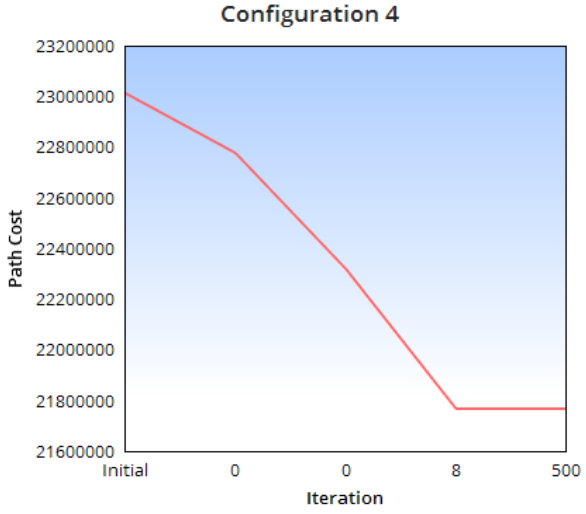
- The path cost is reduced in a well-informed pattern, providing the optimized path cost as: **15542988.732507162**
- Over 500 iterations, the path cost is continuously reduced as it can be seen from the pattern and a reduction of **32.08 %** in path cost is observed
- The random initial population is passed to a Stochastic Selection and a resulting pool is populations of individuals along with their fitness, which ensures that the individual with better fitness (least path cost) has more chances to get selected over the other.
- Two parents are selected from this pool and are mated in order to produce two offspring.
- The two offspring are passed to a Reciprocal Exchange mutation function and based on the mutation probability, the offspring are mutated.
- The offspring are then added to the population set in order to generate a new population over which the genetic algorithm steps are reapplied
- The process is continued for 500 iterations and the best of all the path cost is returned.

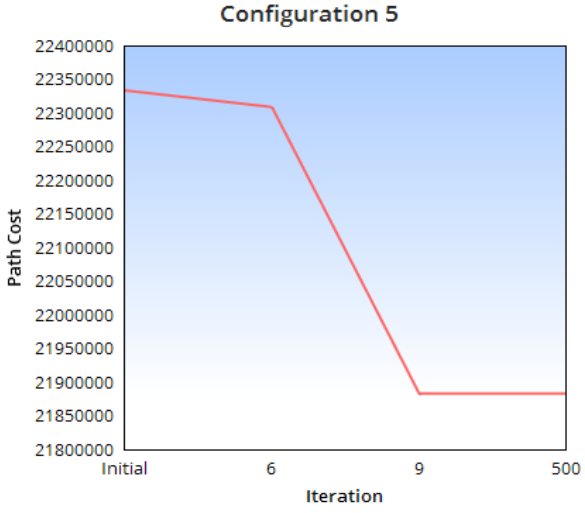
Result comparison over different configurations

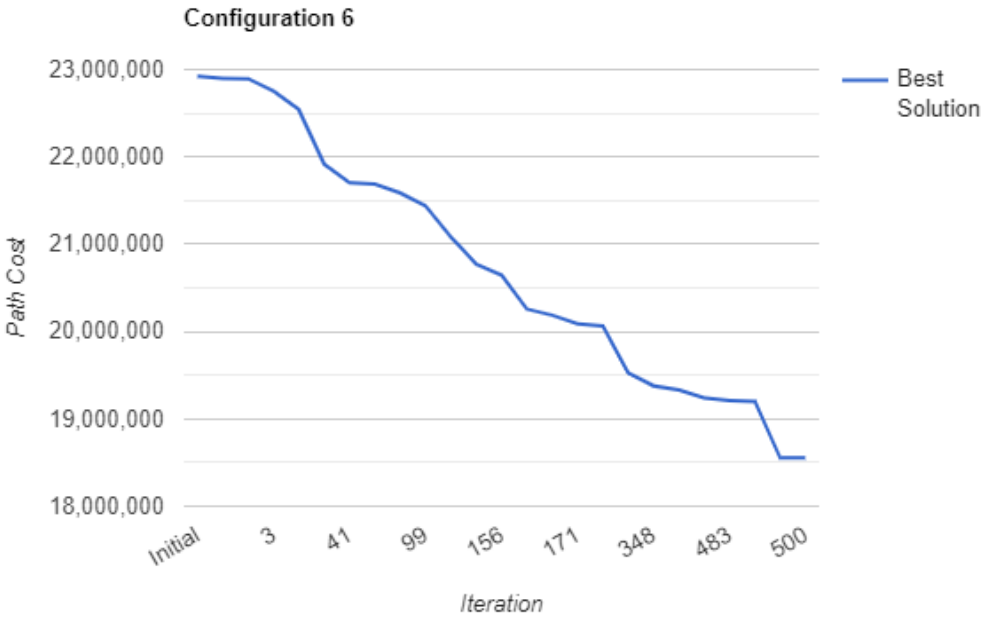












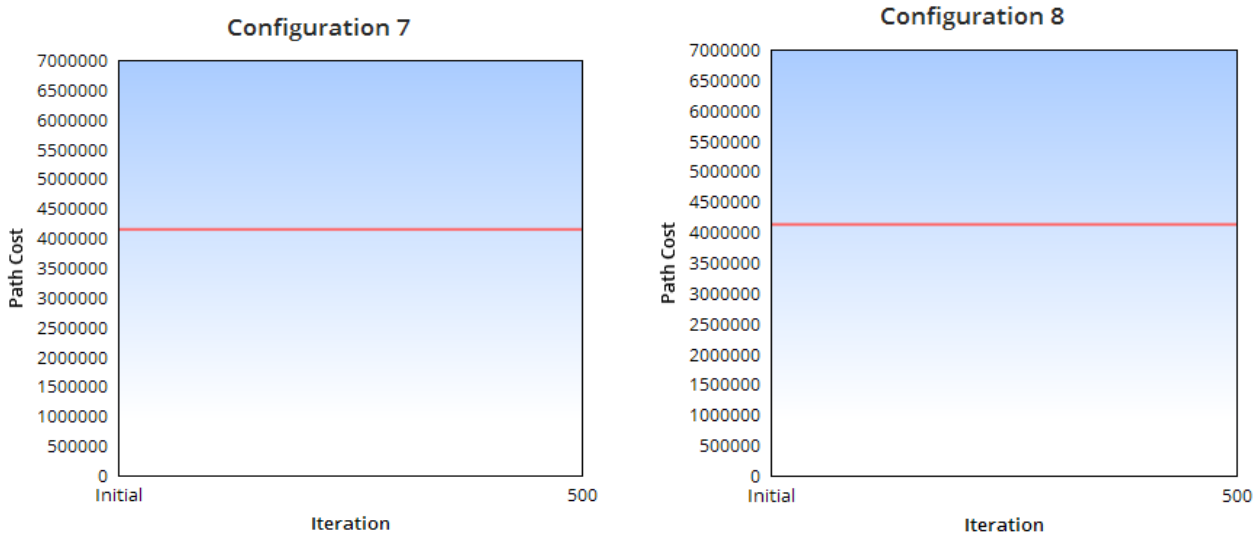


Fig 7

Conclusion:

1. It can be concluded from the above line graphs that the genetic algorithm stands out in case of configuration 3, followed by configuration 6. Therefore, the configuration 3 i.e. A random initial solution on which stochastic universal sampling is performed to select parents followed by generating offspring using Uniform crossover and Reciprocal exchange with a mutation rate of 0.1 works well in optimizing the path for TSP.

Configuration	Initial Solution	Crossover	Mutation	Selection	Best Initial Solution	Best Final Solution
3	Random	Uniform Crossover	Reciprocal Exchange	Stochastic Universal Sampling	22886068	15542988
6	Random	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling	22923572	18552891

2. Configuration 1 and 2 perform decent compared to Configuration 4 and 5 as it can be observed that there in config. 4 ,5 the new generation generated is not able to provide a better solution in most of the iterations. In conf. 1,2 the same behaviour is obtained but there is good reduction in path cost compared to config 4,5. A reduction of average 5% is observed for config 1,2 compared to a reduction in path cost of average 3.7% for config. 4,5

3. Configuration 7 and 8 which use a Heuristic approach to generate the initial solution are not able to get any better solution from the initial solution which is the optimal path cost generated using K nearest neighbour. There is a need to verify the configuration by changing the Mutation Rate, Population size and initial population generation to see if the genetic algorithm can return a better solution

2.1. Problem Instance: inst-13.tsp

Configuration	Initial Solution	Crossover	Mutation	Selection
1	Random	Uniform Crossover	Inversion Mutation	Random Selection

Result:

```
ga = BasicTSP(inst-13.tsp, 100, 0.1, 500, 0, 0, 0, 0) => Configuration 1
Best initial sol: 116982960.19802372
iteration: 0 best: 116081484.29379007
iteration: 1 best: 113716602.58195189
iteration: 5 best: 112545962.92544843
iteration: 21 best: 111181108.27228457
iteration: 70 best: 111094744.1823467
iteration: 80 best: 110978374.39712472
iteration: 126 best: 109830572.27319342
iteration: 207 best: 109318666.18771437
iteration: 258 best: 108920452.2622373
iteration: 314 best: 108449794.10797097
Total iterations: 500
Best Solution: 108449794.10797097
```

Configuration	Initial Solution	Crossover	Mutation	Selection
2	Random	PMX Crossover	Reciprocal Exchange	Random Selection

Result:

```
ga = BasicTSP(inst-13.tsp, 100, 0.1, 500, 0, 0, 1, 1) => Configuration 2
Best initial sol: 111901722.68610172
iteration: 0 best: 111413051.9177431
iteration: 28 best: 111271743.55565692
iteration: 28 best: 111049791.997235
iteration: 35 best: 108620390.32709691
iteration: 173 best: 107201307.29585366
Total iterations: 500
Best Solution: 107201307.29585366
```

Configuration	Initial Solution	Crossover	Mutation	Selection
3	Random	Uniform Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-13.tsp, 100, 0.1, 500, 0, 1, 0, 1) => Configuration 3
Best initial sol: 112590906.77819058
iteration: 5 best: 112508104.85055391
iteration: 19 best: 109298777.93804802
iteration: 48 best: 108777815.45410454
iteration: 48 best: 107284526.45226043
iteration: 60 best: 107102527.25304906
iteration: 68 best: 106379995.30998725
iteration: 76 best: 106351272.25621435
iteration: 78 best: 106215308.92814976
iteration: 84 best: 105698905.9248304
iteration: 90 best: 105408550.60845655
iteration: 93 best: 104145566.84567013
iteration: 96 best: 102105032.1843735
iteration: 110 best: 97154605.59298928
iteration: 141 best: 95720158.74691135
iteration: 144 best: 93815614.74092367
iteration: 145 best: 92435593.00921442
iteration: 145 best: 88513521.87378135
iteration: 170 best: 88150969.45838259
iteration: 174 best: 84265746.75481679
iteration: 189 best: 82988225.80134173
iteration: 195 best: 82469743.85976893
iteration: 199 best: 82270755.9403867
iteration: 200 best: 81552895.15509084
iteration: 200 best: 79883361.82864763
iteration: 201 best: 79070031.23156731
iteration: 201 best: 78408290.83528735
iteration: 202 best: 77921969.72726554
iteration: 219 best: 77582067.17504564
iteration: 220 best: 77178124.92693746
iteration: 223 best: 75923659.11206605
iteration: 224 best: 75360324.11322674
iteration: 228 best: 72735633.65955727
iteration: 238 best: 72531833.11924373
iteration: 240 best: 70120561.42378011
iteration: 248 best: 69251665.12891233
iteration: 248 best: 67252892.85487972
iteration: 251 best: 66647100.49607952
iteration: 258 best: 66588628.98032125
iteration: 260 best: 66395267.09575622
iteration: 265 best: 63789770.9579033
iteration: 279 best: 62453533.17267204
iteration: 282 best: 61084948.518785104
iteration: 286 best: 60951250.60431431
iteration: 315 best: 59704421.28586427
iteration: 316 best: 59698021.690774314
iteration: 353 best: 58842839.60798073
iteration: 410 best: 56823710.12785909
iteration: 415 best: 56501406.61181191
iteration: 416 best: 56205406.53880962
iteration: 417 best: 55775500.37989149
iteration: 428 best: 55078363.15105846
iteration: 441 best: 54070762.73053891
```

```
iteration: 446 best: 53860293.86936242
iteration: 448 best: 52097863.65657239
iteration: 449 best: 51563917.244559236
iteration: 486 best: 51255710.393093936
iteration: 498 best: 49471728.943859056
Total iterations: 500
Best Solution: 49471728.943859056
```

Configuration	Initial Solution	Crossover	Mutation	Selection
4	Random	PMX Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-13.tsp, 100, 0.1, 500, 0, 1, 1, 1) => Configuration 4
Best initial sol: 114378273.11260504
iteration: 0 best: 113321459.2664534
iteration: 2 best: 112100203.80794685
iteration: 9 best: 111952066.05386105
iteration: 38 best: 111466042.14971489
Total iterations: 500
Best Solution: 111466042.14971489
```

Configuration	Initial Solution	Crossover	Mutation	Selection
5	Random	PMX Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-13.tsp, 100, 0.1, 500, 0, 1, 1, 0) => Configuration 5
Best initial sol: 115010857.021496
iteration: 0 best: 112118328.39548004
iteration: 6 best: 112114998.83220676
iteration: 6 best: 110525638.18037358
Total iterations: 500
Best Solution: 110525638.18037358
```

Configuration	Initial Solution	Crossover	Mutation	Selection
6	Random	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-13.tsp, 100, 0.1, 500, 0, 1, 0, 0) => Configuration 6
Best initial sol: 115424898.38681377
iteration: 0 best: 111661835.30078483
iteration: 4 best: 109325661.45548317
iteration: 5 best: 109264218.3999125
iteration: 49 best: 107329471.50869654
iteration: 56 best: 106488201.10893932
iteration: 68 best: 105966877.12718844
iteration: 70 best: 105748716.1297273
```

```
iteration: 148 best: 103790765.42840298
iteration: 190 best: 103771939.91598469
iteration: 202 best: 103739449.93491232
iteration: 213 best: 103594150.5415196
iteration: 242 best: 103553339.00026456
iteration: 339 best: 102522002.44158483
iteration: 346 best: 102278996.83068873
iteration: 377 best: 100579129.24414167
iteration: 401 best: 100172858.9526735
iteration: 402 best: 97717739.91825423
iteration: 427 best: 95898928.56502724
Total iterations: 500
Best Solution: 95898928.56502724
```

Configuration	Initial Solution	Crossover	Mutation	Selection
7	Heuristic	PMX Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-13.tsp, 100, 0.1, 500, 1, 1, 1, 1) => Configuration 7
Best initial sol: 7203070.735067
Total iterations: 500
Best Solution: 7203070.735067
```

Configuration	Initial Solution	Crossover	Mutation	Selection
8	Heuristic	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-13.tsp, 100, 0.1, 500, 1, 1, 0, 0) => Configuration 8
Best initial sol: 7218941.1220835615
Total iterations: 500
Best Solution: 7218941.1220835615
```

2.2. Result Analysis: inst-13.tsp

It can be observed from the output that different configurations lead to a different optimized path cost.

The above test was performed with two set of initial population Random and Heuristic.

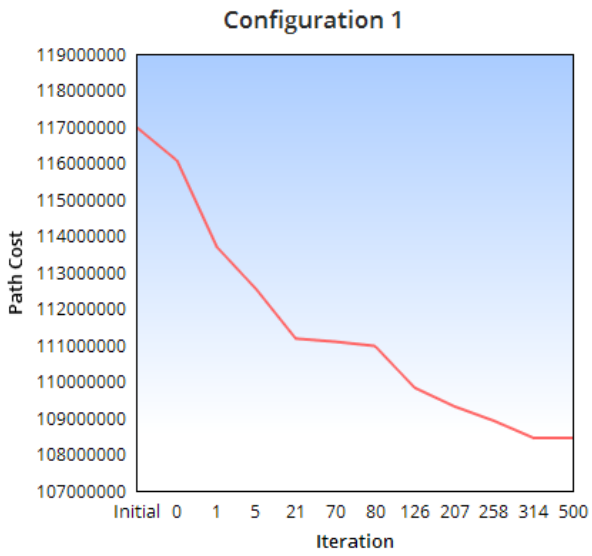
In configuration from 1 to 6, the initial population selection was Random which generated the best initial solution i.e. the path cost on the population generated randomly**.** The genetic algorithm was run over this initial path solution with a combination of various selection, crossover and Mutation operators. With the Population Size =100, Mutation Probability = 0.1 and Iterations = 500.

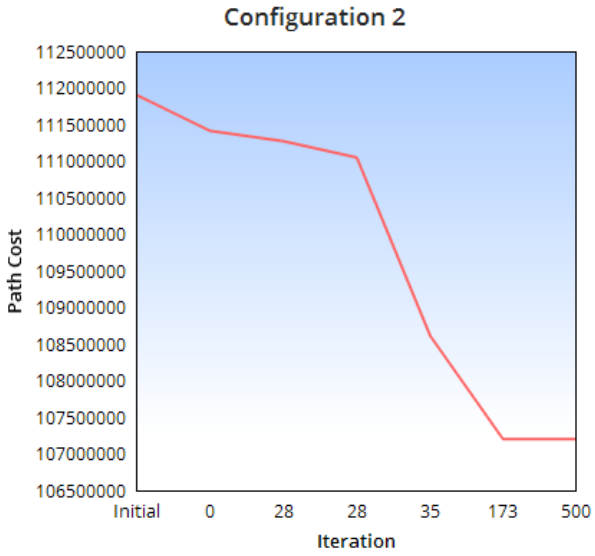
Best Result:

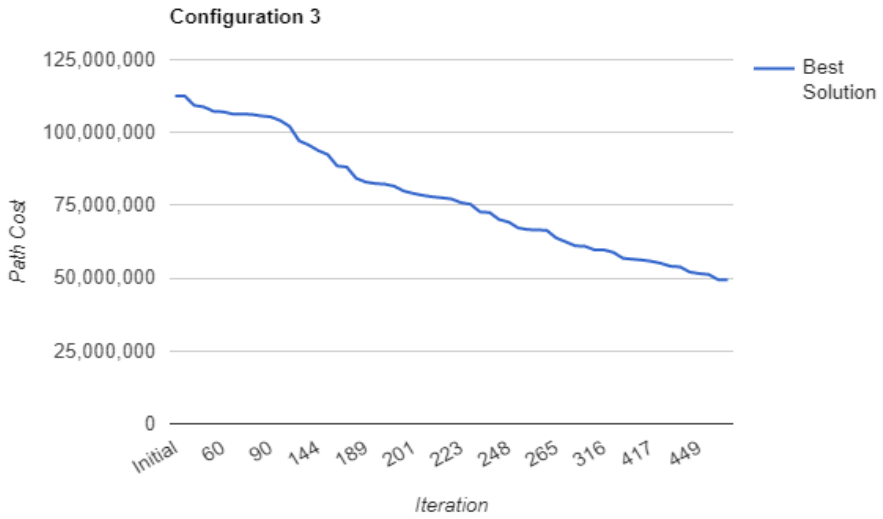
The Genetic Algorithm performs extremely well in case of Configuration 3 (Random Initial Population, Stochastic selection, Uniform Crossover, Reciprocal Exchange Mutation).

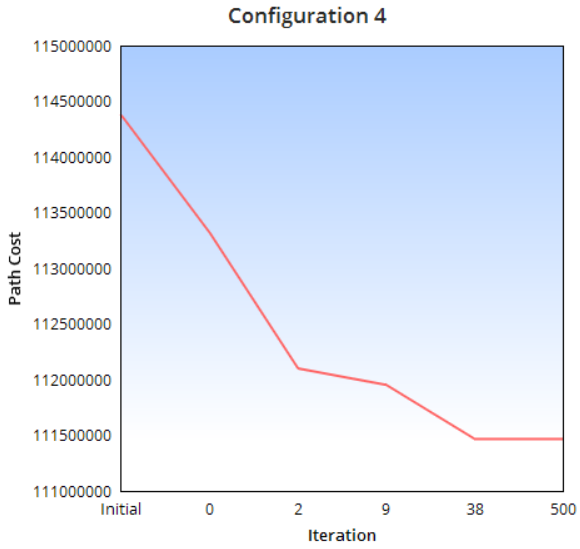
- The path cost is reduced in a well-informed pattern, providing the optimized path cost as: **49471728.943859056**
- Over 500 iterations, the path cost is continuously reduced as it can be seen from the pattern and a reduction of **56.06 %** in path cost is observed
- The random initial population is passed to a Stochastic Selection and a resulting pool is populations of individuals along with their fitness, which ensures that the individual with better fitness (least path cost, used Minimization) has more chances to get selected over the other.
- Two parents are selected from this pool and are mated in order to produce two offspring.
- The two offspring are passed to a Reciprocal Exchange mutation function and based on the mutation probability, the offspring are mutated.
- The offspring are then added to the population set in order to generate a new population over which the genetic algorithm steps are reapplied
- The process is continued for 500 iterations and the best of all the path cost is returned.

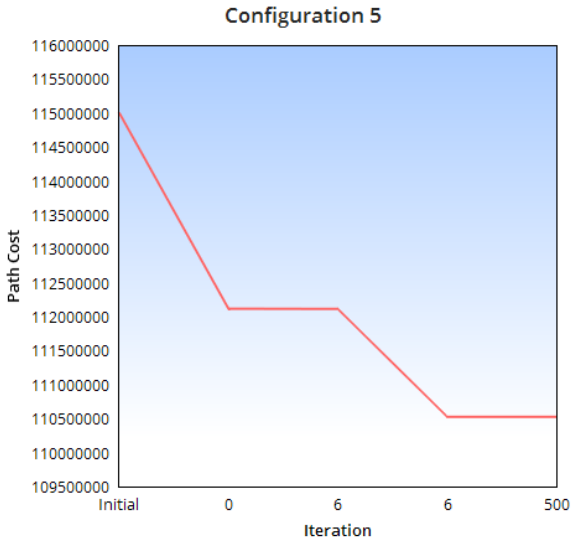
Result comparison over different configurations

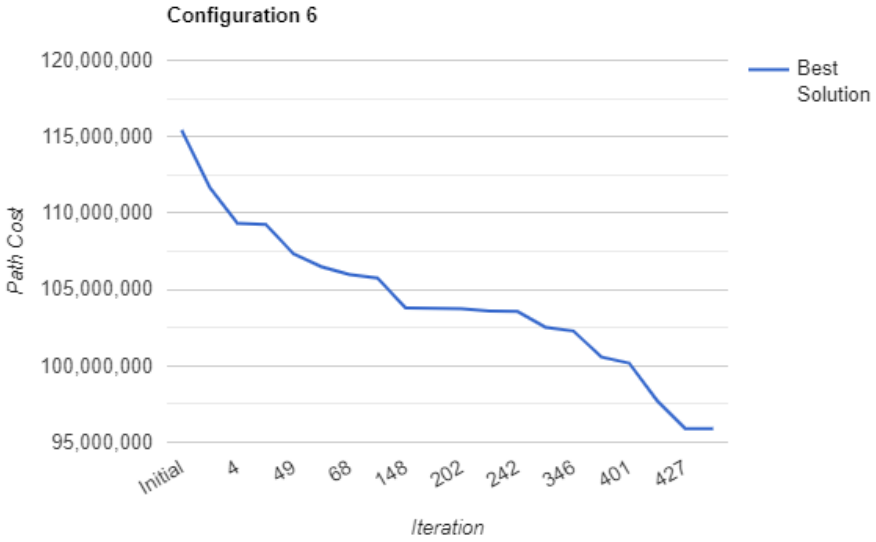





Configuration 3
125,000,000
100,000,000
75,000,000
50,000,000
25,000,000
0
Path Cost
Initial
60
90
144
189
201
223
248
265
316
417
449
Iteration
Best Solution


Configuration 4
115000000
114500000
114000000
113500000
113000000
112500000
112000000
111500000
111000000
Path Cost
Initial
0
2
9
38
500
Iteration


Configuration 5
116000000
115500000
115000000
114500000
114000000
113500000
113000000
112500000
112000000
111500000
111000000
110500000
110000000
109500000
Path Cost
Initial
0
6
6
500
Iteration


Configuration 6
120,000,000
115,000,000
110,000,000
105,000,000
100,000,000
95,000,000
Path Cost
Initial
4
49
68
148
202
242
346
401
427
Iteration
Best Solution

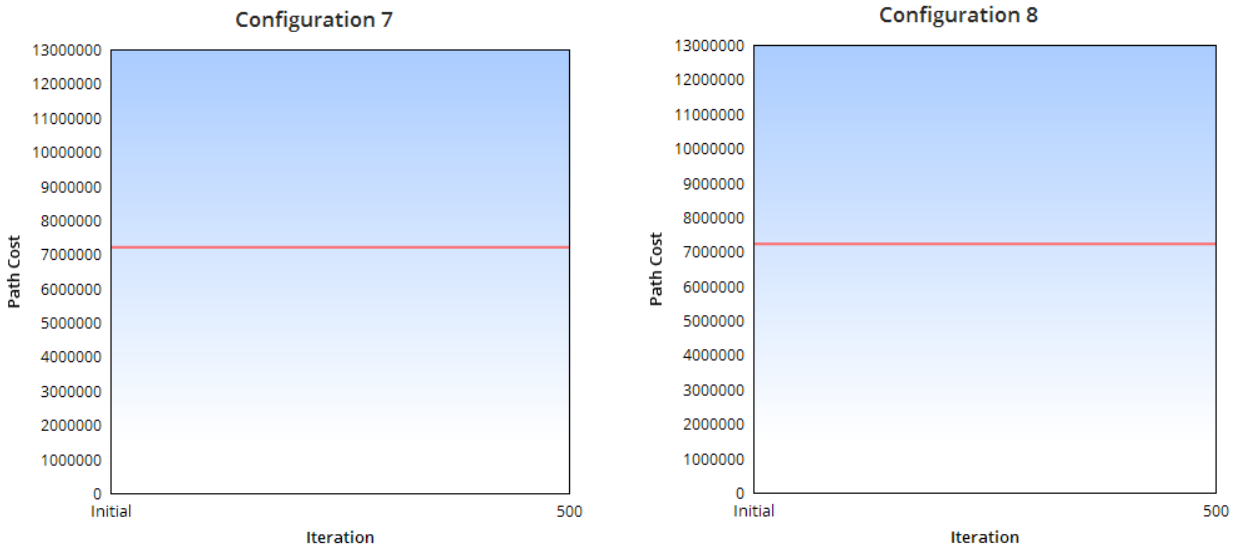


Fig 8

Conclusion:

1. It can be concluded from the above line graphs that the genetic algorithm stands out in case of configuration 3, followed by configuration 6. Therefore, the configuration 3 i.e. A random initial solution on which stochastic universal sampling is performed to select parents followed by generating offspring using Uniform crossover and Reciprocal Exchange with a mutation rate of 0.1 works well in optimizing the path for TSP.

Configuration	Initial Solution	Crossover	Mutation	Selection	Best Initial Solution	Best Final Solution
3	Random	Uniform Crossover	Reciprocal Exchange	Stochastic Universal Sampling	112590906	49471728
6	Random	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling	115424898	95898928

2. Configuration 1 and 2 perform decent compared to Configuration 4 and 5 as it can be observed that there in config. 4,5 the new generation generated is not able to provide a better solution in most of the iterations. In conf. 1,2 the same behaviour is obtained but there is good reduction in path cost (**average 5.7 %**) compared to config 4,5 (**average 3.2%**).

3. Configuration 7 and 8 which use a Heuristic approach to generate the initial solution are not able to get any better solution from the initial solution which is the optimal path cost generated using K nearest neighbour. There is a need to verify the configuration by changing the Mutation Rate, Population size and initial population generation to see if the genetic algorithm can return a better solution

3.1. Problem Instance: inst-5.tsp

Configuration	Initial Solution	Crossover	Mutation	Selection
1	Random	Uniform Crossover	Inversion Mutation	Random Selection

Result:

```
ga = BasicTSP(inst-5.tsp, 100, 0.1, 500, 0, 0, 0, 0) => Configuration 1
Best initial sol: 442292086.7272037
iteration: 0 best: 439174047.4337871
iteration: 2 best: 436389032.27438384
iteration: 4 best: 430254778.7319146
iteration: 45 best: 427614956.43884045
iteration: 123 best: 425544615.39597756
iteration: 207 best: 423773574.8426651
Total iterations: 500
Best Solution: 423773574.8426651
```

Configuration	Initial Solution	Crossover	Mutation	Selection
2	Random	PMX Crossover	Reciprocal Exchange	Random Selection

Result:

```
ga = BasicTSP(inst-5.tsp, 100, 0.1, 500, 0, 0, 1, 1) => Configuration 2
Best initial sol: 442958144.7820282
iteration: 0 best: 442542889.3332863
iteration: 0 best: 440716707.42196465
iteration: 2 best: 439794042.1214025
iteration: 5 best: 438570204.4219845
iteration: 8 best: 436621869.11033696
iteration: 9 best: 435124391.81678957
iteration: 9 best: 431355177.69057953
iteration: 10 best: 430554217.5523967
iteration: 43 best: 426575787.1800647
iteration: 310 best: 425542753.5636455
Total iterations: 500
Best Solution: 425542753.5636455
```

Configuration	Initial Solution	Crossover	Mutation	Selection
3	Random	Uniform Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-5.tsp, 100, 0.1, 500, 0, 1, 0, 1) => Configuration 3
Best initial sol: 439726119.60478437
iteration: 0 best: 439674983.3052773
iteration: 1 best: 435456703.0402727
iteration: 3 best: 435184494.48711264
```

```
iteration: 10 best: 434562439.29797435
iteration: 14 best: 433662725.33515394
iteration: 19 best: 431387143.8598588
iteration: 23 best: 429096811.0204498
iteration: 44 best: 427919827.7034029
iteration: 53 best: 426819848.93294865
iteration: 56 best: 426126107.6228172
iteration: 63 best: 420208822.2954254
iteration: 110 best: 419779609.2447534
iteration: 115 best: 418481624.5405844
iteration: 124 best: 413388097.42137337
iteration: 143 best: 406835738.52725816
iteration: 231 best: 406266631.4375808
iteration: 278 best: 403793315.6684574
iteration: 286 best: 402130579.8687609
iteration: 286 best: 401164197.36988944
iteration: 287 best: 396091603.0758568
iteration: 363 best: 395210722.21768796
iteration: 364 best: 394971493.79898924
iteration: 377 best: 394311461.17771
iteration: 379 best: 393480813.84330326
iteration: 388 best: 390420739.3523249
iteration: 405 best: 389841128.0034549
iteration: 486 best: 388121895.97890097
iteration: 496 best: 385551975.4523037
Total iterations: 500
Best Solution: 385551975.4523037
```

Configuration	Initial Solution	Crossover	Mutation	Selection
4	Random	PMX Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-5.tsp, 100, 0.1, 500, 0, 1, 1, 1) => Configuration 4
Best initial sol: 438227237.3938074
iteration: 1 best: 436300160.0689194
iteration: 8 best: 435380110.93164396
iteration: 8 best: 425406560.95473903
Total iterations: 500
Best Solution: 425406560.95473903
```

Configuration	Initial Solution	Crossover	Mutation	Selection
5	Random	PMX Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-5.tsp, 100, 0.1, 500, 0, 1, 1, 0) => Configuration 5
Best initial sol: 439916934.95398176
iteration: 1 best: 439868613.75613517
iteration: 1 best: 437826467.3743276
iteration: 1 best: 437006985.30824
iteration: 3 best: 436137111.6052191
Total iterations: 500
Best Solution: 436137111.6052191
```

Configuration	Initial Solution	Crossover	Mutation	Selection
6	Random	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-5.tsp, 100, 0.1, 500, 0, 1, 0, 0) => Configuration 6
Best initial sol: 445005130.0223137
iteration: 0 best: 436280182.3850226
iteration: 2 best: 430710365.0038706
iteration: 5 best: 429753375.124218
iteration: 44 best: 429691648.374963
iteration: 44 best: 428456272.3099735
iteration: 126 best: 424578004.86284995
iteration: 134 best: 424364857.78645515
iteration: 279 best: 423527580.3027021
iteration: 302 best: 421403123.12396187
iteration: 340 best: 420860692.46102035
iteration: 438 best: 417586951.0033147
iteration: 441 best: 416149802.49569803
iteration: 442 best: 414153370.6057604
Total iterations: 500
Best Solution: 414153370.6057604
```

Configuration	Initial Solution	Crossover	Mutation	Selection
7	Heuristic	PMX Crossover	Reciprocal Exchange	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-5.tsp, 100, 0.1, 500, 1, 1, 1, 1) => Configuration 7
Best initial sol: 12995945.232143892
Total iterations: 500
Best Solution: 12995945.232143892
```

Configuration	Initial Solution	Crossover	Mutation	Selection
8	Heuristic	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling

Result:

```
ga = BasicTSP(inst-5.tsp, 100, 0.1, 500, 1, 1, 0, 0) => Configuration 8
Best initial sol: 12995945.232143892
Total iterations: 500
Best Solution: 12995945.232143892
```

3.2. Result Analysis: inst-5.tsp

It can be observed from the output that different configurations lead to a different optimized path cost.

The above test was performed with two set of initial population Random and Heuristic.

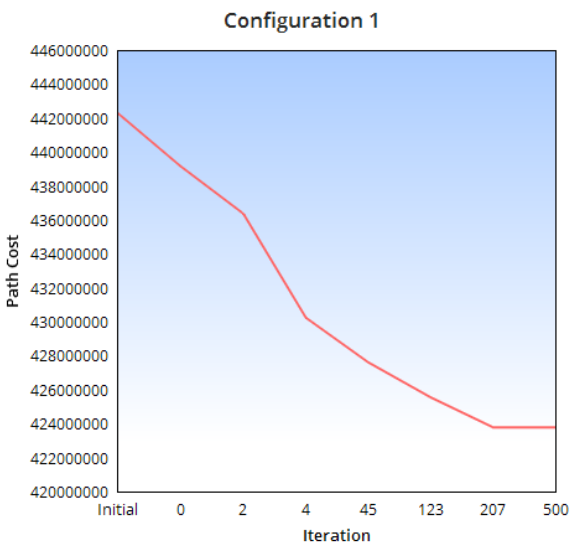
In configuration from 1 to 6, the initial population selection was Random which generated the best initial solution i.e. the path cost on the population generated randomly**.** The genetic algorithm was run over this initial path solution with a combination of various selection, crossover and Mutation operators. With the Population Size =100, Mutation Probability = 0.1 and Iterations = 500.

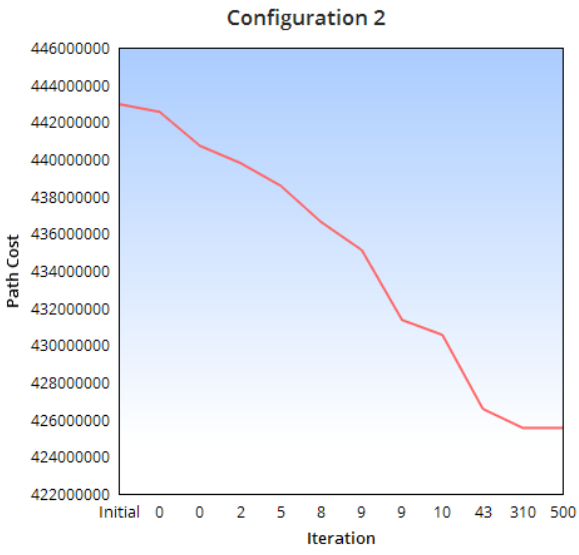
Best Result:

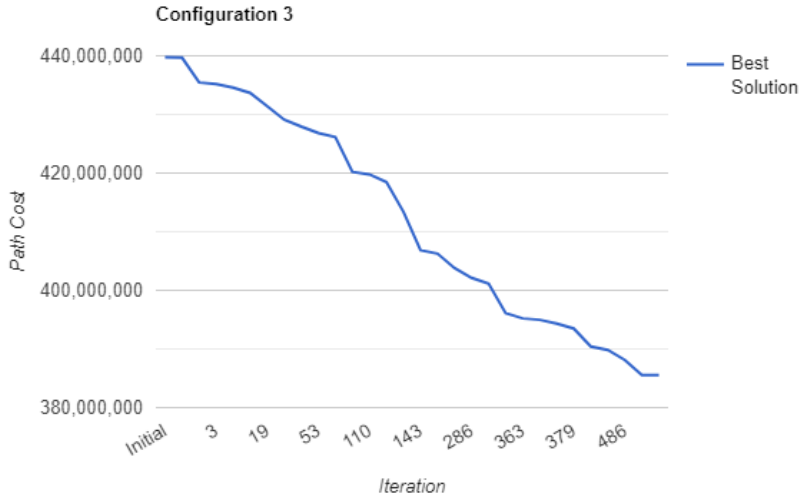
The Genetic Algorithm performs extremely well in case of Configuration 3 (Random Initial Population, Stochastic selection, Uniform Crossover, Reciprocal Exchange Mutation).

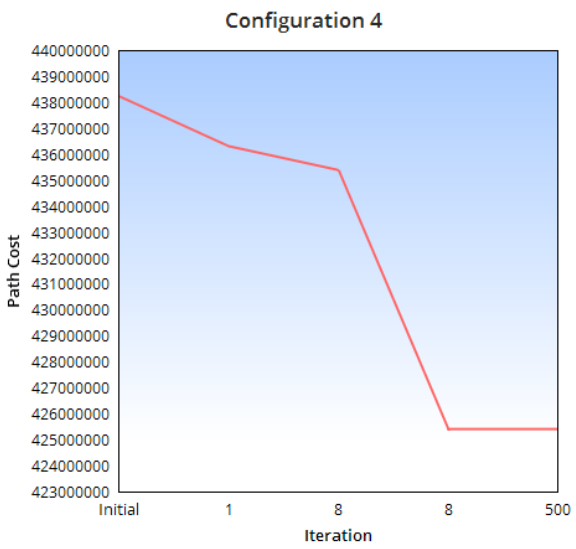
- The path cost is reduced in a well-informed pattern, providing the optimized path cost as: **385551975.4523037**
- Over 500 iterations, the path cost is continuously reduced as it can be seen from the pattern and a reduction of **12.31 %** in path cost is observed
- The random initial population is passed to a Stochastic Selection and a resulting pool is populations of individuals along with their fitness, which ensures that the individual with better fitness (least path cost, used Minimization) has more chances to get selected over the other.
- Two parents are selected from this pool and are mated in order to produce two offspring.
- The two offspring are passed to a Reciprocal Exchange mutation function and based on the mutation probability, the offspring are mutated.
- The offspring are then added to the population set in order to generate a new population over which the genetic algorithm steps are reapplied
- The process is continued for 500 iterations and the best of all the path cost is returned.

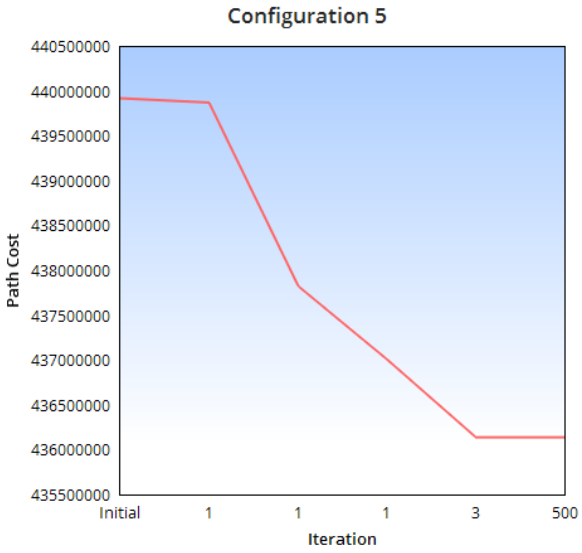
Result comparison over different configurations

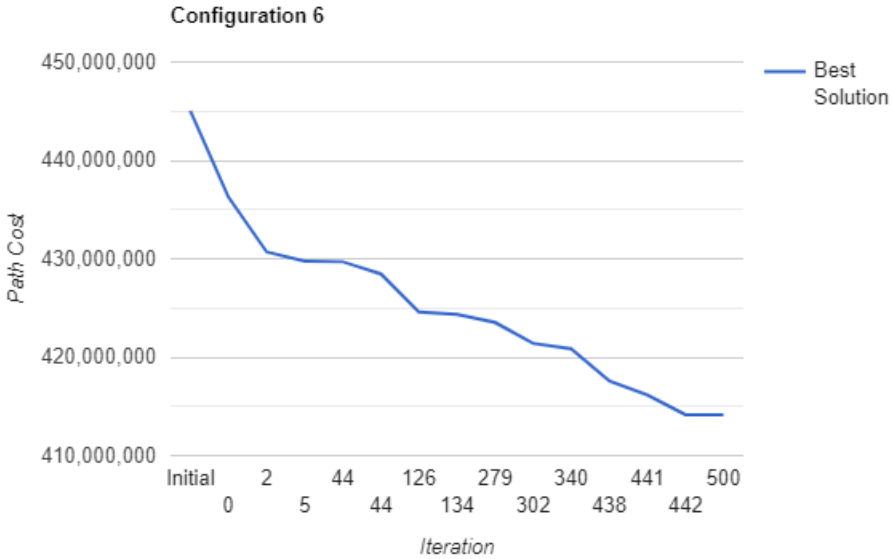





Configuration 3
440,000,000
420,000,000
400,000,000
380,000,000
Path Cost
Initial
3
19
53
110
143
286
363
379
486
Iteration
Best Solution


Configuration 4
440000000
439000000
438000000
437000000
436000000
435000000
434000000
433000000
432000000
431000000
430000000
429000000
428000000
427000000
426000000
425000000
424000000
423000000
Path Cost
Initial
1
8
8
500
Iteration


Configuration 5
440500000
440000000
439500000
439000000
438500000
438000000
437500000
437000000
436500000
436000000
435500000
Path Cost
Initial
1
1
1
3
500
Iteration


Configuration 6
450,000,000
440,000,000
430,000,000
420,000,000
410,000,000
Path Cost
Initial
0
2
5
44
44
126
134
279
302
340
438
441
442
500
Iteration
Best Solution

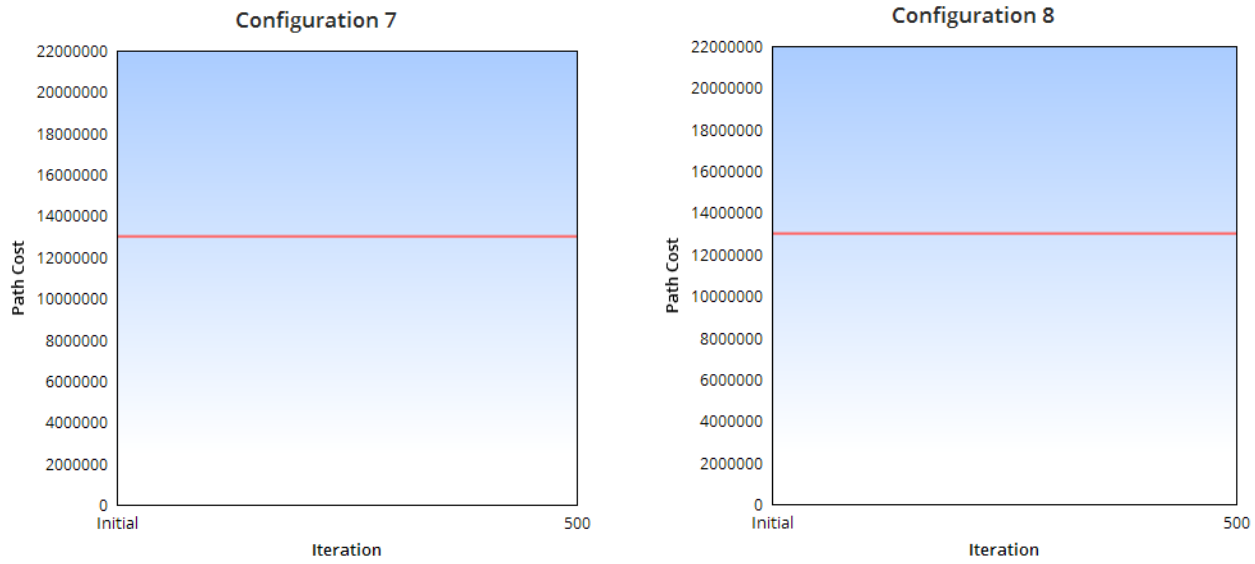


Fig 9

Conclusion:

1. It can be concluded from the above line graphs that the genetic algorithm stands out in case of configuration 3, followed by configuration 6. Therefore, the configuration 3 i.e. A random initial solution on which stochastic universal sampling is performed to select parents followed by generating offspring using Uniform crossover and Reciprocal Exchange with a mutation rate of 0.1 works well in optimizing the path for TSP.

Configuration	Initial Solution	Crossover	Mutation	Selection	Best Initial Solution	Best Final Solution
3	Random	Uniform Crossover	Reciprocal Exchange	Stochastic Universal Sampling	439726119	385551975
6	Random	Uniform Crossover	Inversion Mutation	Stochastic Universal Sampling	445005130	414153370

2. Configuration 1 and 2 perform decent compared to Configuration 4 and 5 as it can be observed that there in config. 4,5 the new generation generated is not able to provide a better solution in most of the iterations. In conf. 1,2 the same behaviour is obtained but there is good reduction in path cost (**average 4 %**) compared to config 4,5 (**average 1.88%**).

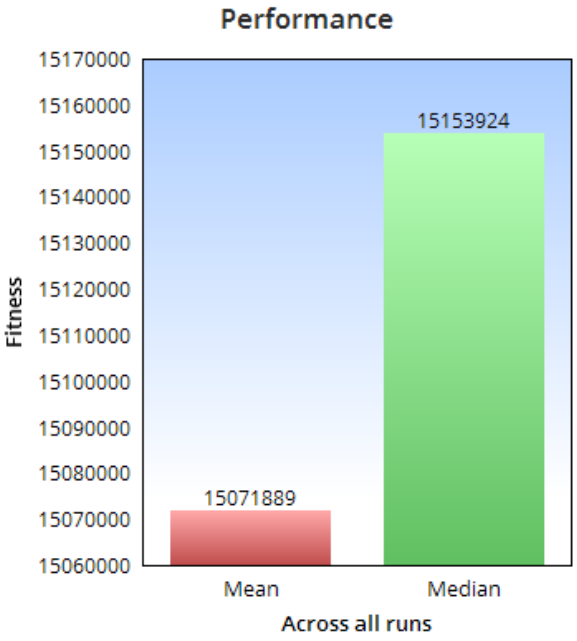
3. Configuration 7 and 8 which use a Heuristic approach to generate the initial solution are not able to get any better solution from the initial solution which is the optimal path cost generated using K nearest neighbour. There is a need to verify the configuration by changing the Mutation Rate, Population size and initial population generation to see if the genetic algorithm can return a better solution

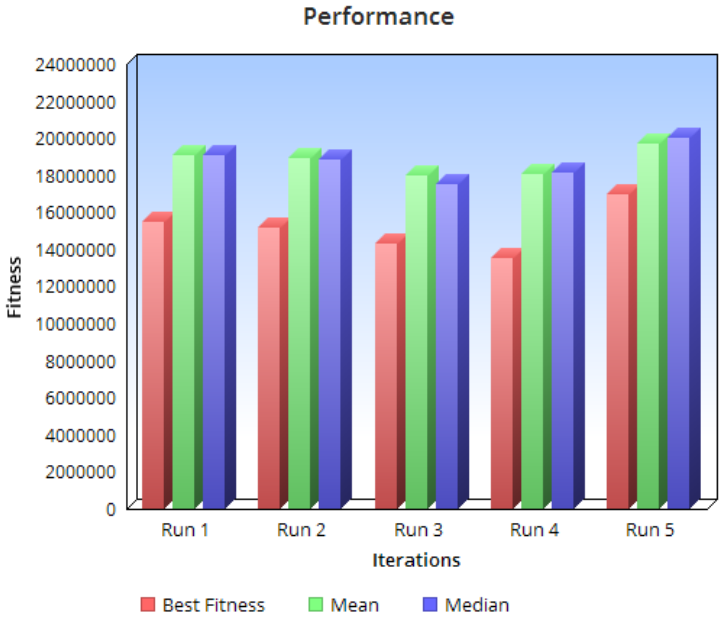
4. Analysing the overall performance of the Genetic Algorithm

It can be well drawn from the previous results that the algorithm performs best in case of Configuration 3 i.e. Random Initial Solution with Stochastic Universal Sampling on which uniform crossover and reciprocal exchange mutation is performed followed by Configuration 6 i.e. Random Initial Solution with Stochastic Universal Sampling on which uniform crossover and Inversion mutation is performed.

To analyse the overall performance of the algorithm, we will be running the algorithm with problem instance inst-0.tsp and configuration 3 and 6 for 500 iterations with population size 100 and mutation rate 0.1 for 5 executions with each execution being run for 500 iterations.

Configuration 3							
Iteration:	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Mean**	**Median**
Best Fitness:	15455157	15153924	14285366	13491097	16973901	**15071889.0**	**15153924.0**
Mean:	19089820.11	18934967.90	18017896.58	18078697.11	19724978.45		
Median:	19064818.00	18863455.00	17501922.00	18135157.00	20025632.00		

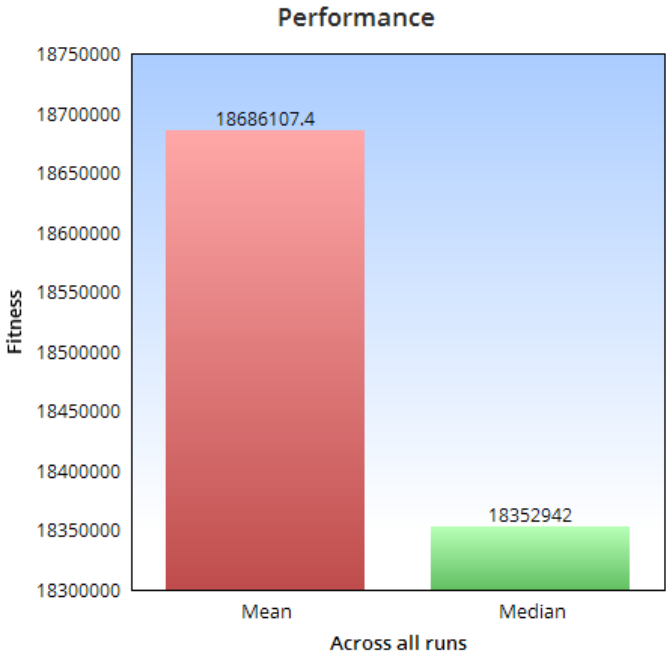


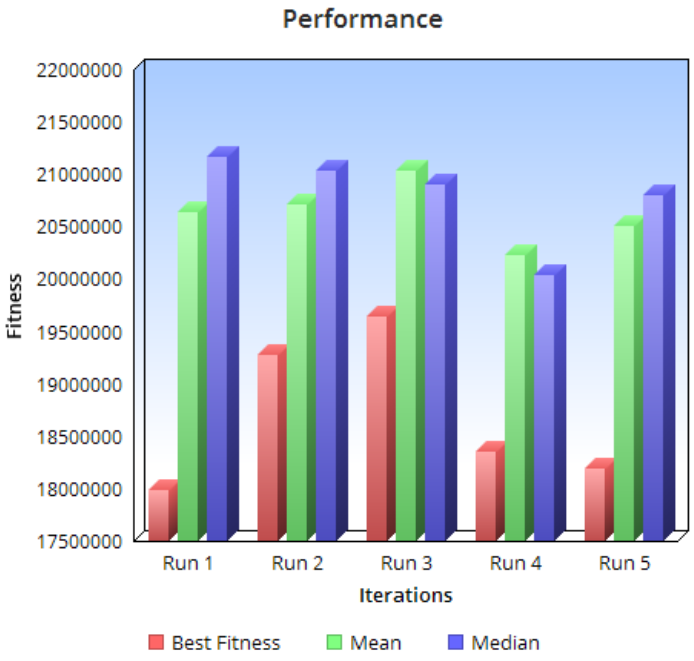


Conclusion:

It can be concluded that on average the genetic algorithm with best mean initial solution of **22715665.8** optimizes that path and reaches to a mean best solution of **15071889.0,** which is equivalent to optimization of **33.64%**.

Configuration 6							
Iteration:	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Mean**	**Median**
Best Fitness:	17983753	19266540	19637496	18352942	18189806	**18686107.4**	**18352942**
Mean:	20642087.36	20714348.00	21031420.27	20229969.83	20504934.48		
Median:	21165917.00	21025387.00	20900749.00	20032761.50	20797243.00		





Conclusion:

It can be concluded that on average the genetic algorithm with best mean initial solution of **22739001.43** optimizes that path and reaches to a mean best solution of **18686107.4,** which is equivalent to optimization of **17.82%**.

5. Additional Experimentation of the Genetic Algorithm

In order to understand the behaviour of the algorithm with a much wider scope, we will be experimenting the impact on the algorithm's performance by varying the following:

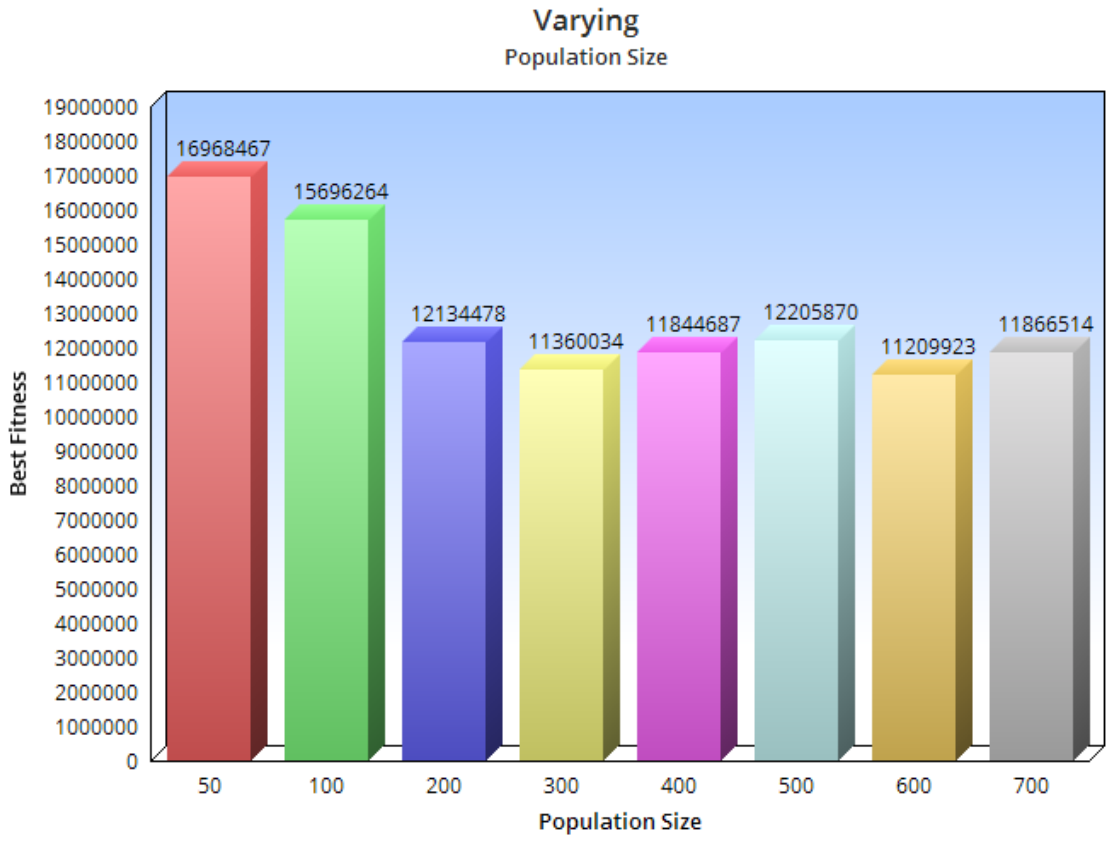
- Population Size
- Mutation Rate

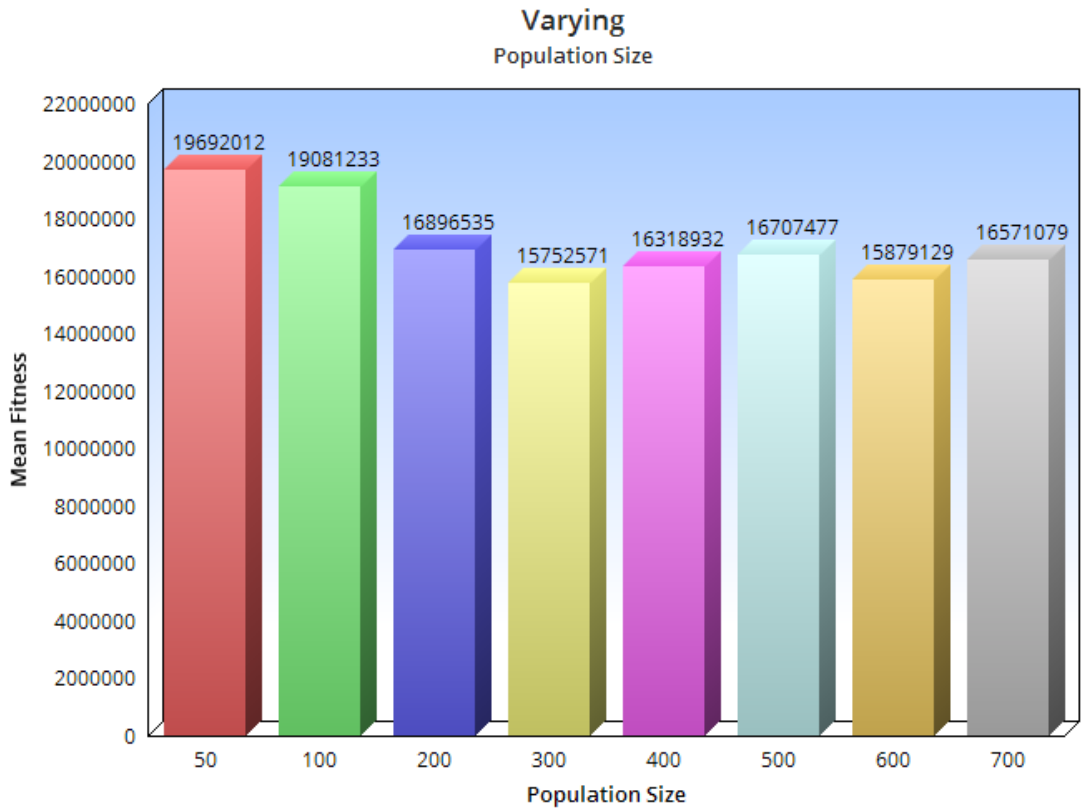
As from the previous performance matrix, the best configuration for the TSP environment we are working on is Configuration 3 and therefore we will be experimenting the above parameters with same configuration and problem instance inst-0.tsp. Please note that the other parameters including initial selection, crossover and number of iterations remain unchanged.

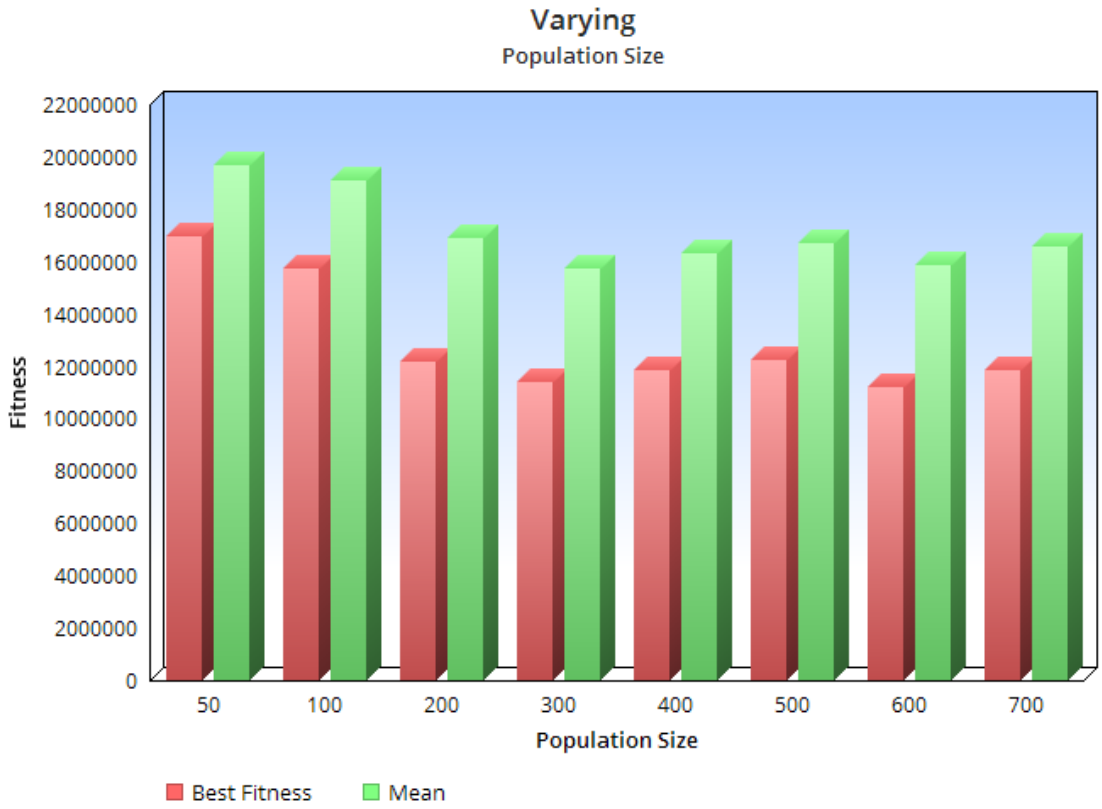
1. Population Size

Mutation Rate: 0.1

Configuration 3								
Population Size:	**50**	**100**	**200**	**300**	**400**	**500**	**600**	**700**
Best Fitness:	16968467	15696264	12134478	11360034	11844687	12205870	11209923	11866514
Mean Fitness:	19692012	19081233	16896535	15752571	16318932	16707477	15879129	16571079




Varying
Population Size
Mean Fitness
22000000
20000000
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
19692012
19081233
16896535
15752571
16318932
16707477
15879129
16571079
50
100
200
300
400
500
600
700
Population Size


Varying
Population Size
Fitness
22000000
20000000
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
50
100
200
300
400
500
600
700
Population Size
Best Fitness
Mean

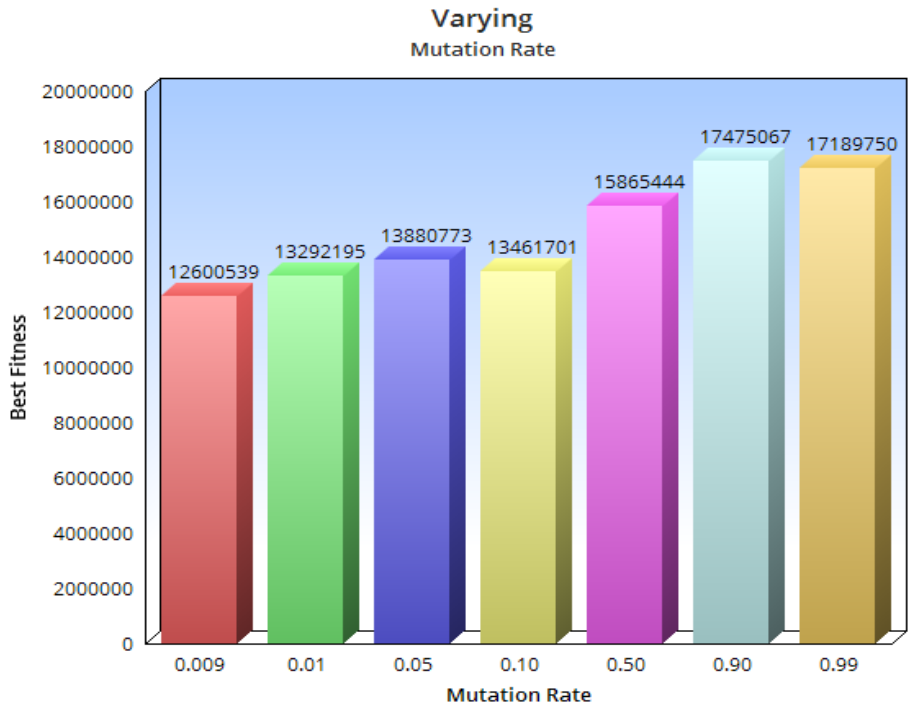
Conclusion:

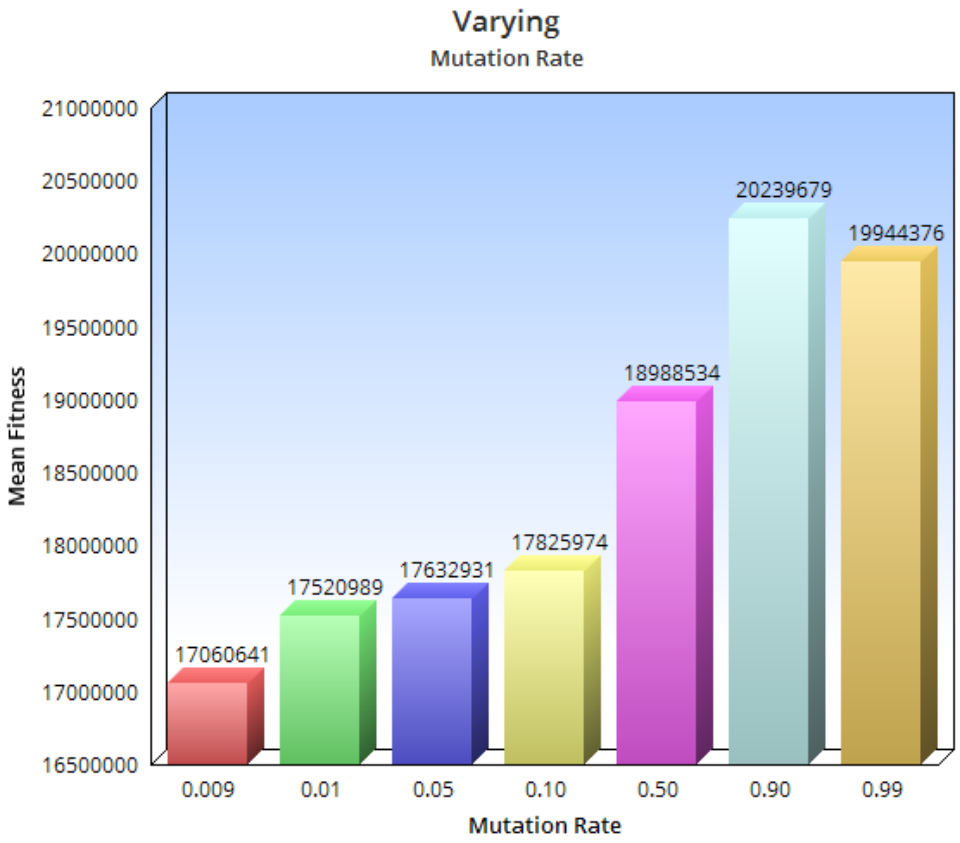
- The best optimal solution is obtained in case of population size = 600, however it can also be concluded that a very near optimal solution is obtained when the population size is half that is 300
- On considering the mean fitness, the best results are obtained when the population is 300.
- On constantly increasing the population size from 50 to 300 a good improvement is seen in the generation of optimal solution
- On further increasing the population the optimal solution was not found of which the genetic algorithm is capable of.
- A very high population size of 700 causes the genetic algorithm to slow down without even producing the best shortest path.
- Therefore, it should be kept in mind that a very high population might not lead to better solution as well as a very low population might not provide diversity in mating pool.
- An optimal population size might be decided with respect to better result and operational time.

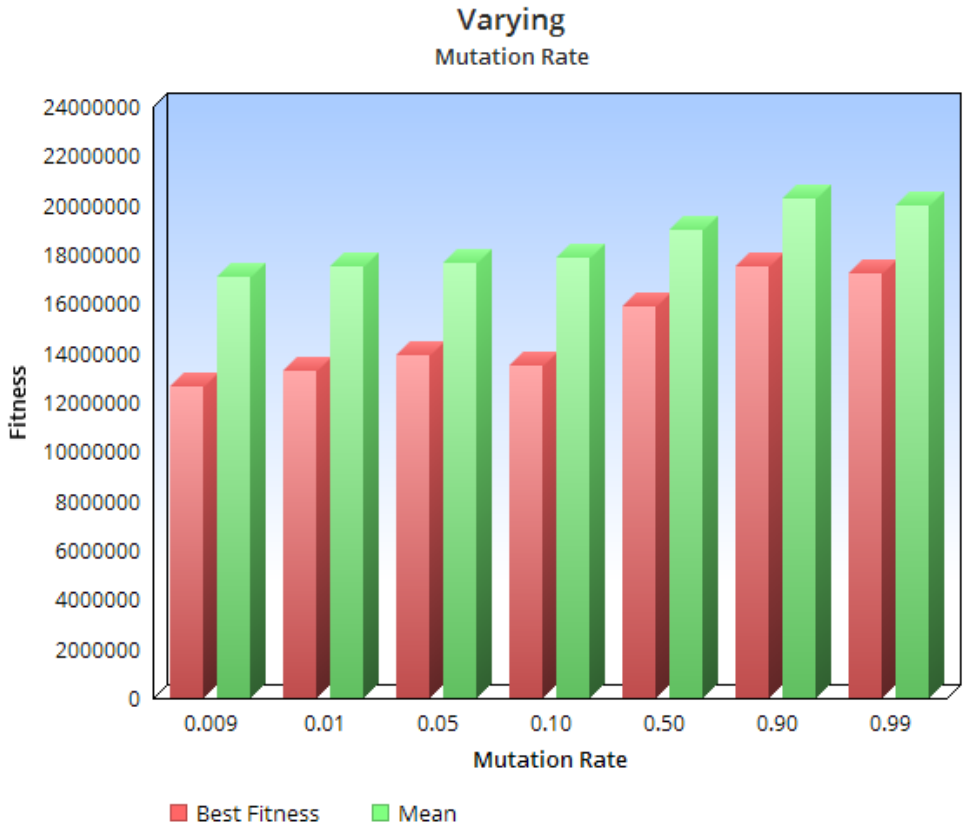
2. Mutation Rate

Population Size = 100

Configuration 3							
Mutation Rate:	**0.009**	**0.01**	**0.05**	**0.10**	**0.50**	**0.90**	**0.99**
Best Fitness:	12600539	13292195	13880773	13461701	15865444	17475067	17189750
Mean Fitness:	17060641	17520989	17632931	17825974	18988534	20239679	19944376




Varying
Mutation Rate
Mean Fitness
21000000
20500000
20000000
19500000
19000000
18500000
18000000
17500000
17000000
16500000
17060641
17520989
17632931
17825974
18988534
20239679
19944376
0.009
0.01
0.05
0.10
0.50
0.90
0.99
Mutation Rate


Varying
Mutation Rate
Fitness
24000000
22000000
20000000
18000000
16000000
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
0.009
0.01
0.05
0.10
0.50
0.90
0.99
Mutation Rate
Best Fitness
Mean

Conclusion:

From the above observations with varying the Mutation Rate, it can be concluded that the genetic algorithm performs wells with low mutation probability (P_m).

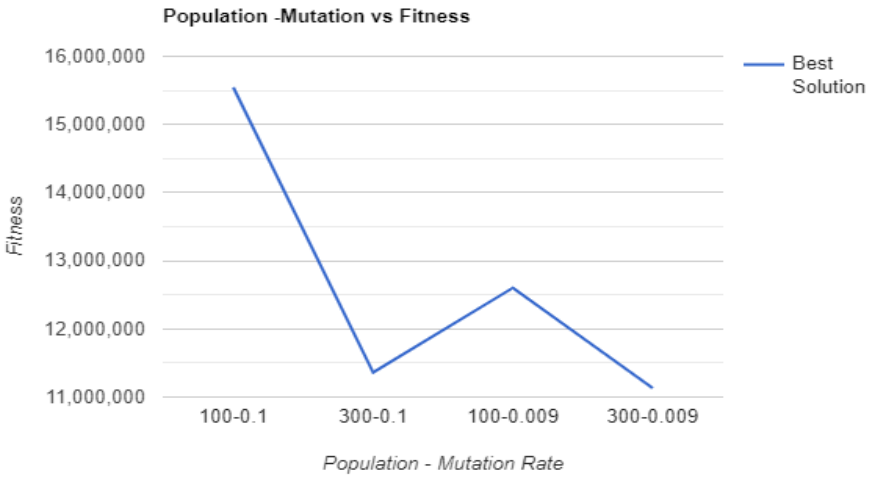
- As the P_m is reduced the genetic algorithm performs well.
- On increasing the P_m, the genetic algorithm is not able to reach the best optimal solution it is capable of.
- A large P_m, shows that the genetic algorithm gets reduced to a random search
- The best mean fitness is observed in case of P_m 0.009
- In the problem here, specific to TSP, a lower mutation rate preforms well
- A very high mutation rate is not able to maintain the diversity in the population
- Therefore, the mutation rate should be kept such that there is a good balance in diversity of the population

6. Analysing the Algorithm with best population and mutation rate.

In the previous section it was observed that the algorithm performs well with population size 300 (when mutation rate = 0.1) and a mutation rate of 0.009 (when population rate = 100). A further experimentation was done to analyse if a much better result that is better optimized path length can be obtained if the best population and mutation rate are combined.

Result:

Population	Mutation Rate	Best Fitness
100	0.1	15542988
300	0.1	11360034
100	0.009	12600539
300	0.009	11132780



Conclusion:

It can be concluded from the above results that the algorithm performs well with a lower mutation rate that is 0.009 and a population of 300 chromosomes.

7. Experimentation of the Genetic Algorithm with Heuristic Initial Population Selection

As it can be seen from the above test result, that the initial population generated using heuristic approach is not able to provide any optimized path after running it through the genetic algorithm.

Various trial and error with the population size and mutation rate were tried but still the algorithm was not able to generate any better final solution than the one generated by initial solution

Approach tried:

Since the initial population was all generated using the heuristic method, there could have been a possibility of less diversity in the population and therefore making it difficult for the GA to generate any optimal path cost.

In order to test this, the initial population was initialized 50% with heuristic approach and another 50% with random generation of individual. On checking the results, it was observed that the with very less mutation rate(0.005 – 0.009) and population size between 100 to 200 the GA tried reaching to the best initial solution but then resulted in population convergence.

The best initial solution was 3992001.97 , the GA with the above changes was able to find a path cost nearly 4000000 but after that lead to convergence.