**Metaheuristic Optimization**

**Assignment 1**

**Student Id: R0018250*5***

**Name: D**heeraj **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 = (w1 ∨ -w2 ∨ w3 ∨ -w4 ∨ -w5) ∧ (-w3 ∨ w4)**

**SOLUTION:**

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

Let Ci 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 C1 and C2.

Where,

**C1 = (w1 ∨ -w2 ∨ w3 ∨ -w4 ∨ -w5)**

**C2 = (-w3 ∨ w4)**

The above-mentioned clauses Ci can be replaced by a conjunction of clauses in Xi.

Where,

* All Clauses in Xi­ must contain 3 literals.
* Ci is satisfiable iff constructed Xi 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 |

**Reduction of C1 to X1:**

C1 = (w1 ∨ -w2 ∨ w3 ∨ -w4 ∨ -w5)

Here,

w1, w2, w3, w4, w5 are literals

Let K = Number of literals in C1

Therefore K = 5

Since K > 3, we need to introduce K-3 new variables (yi1, yi2, yi3, yi4**……..** yik-3) and replace C1 with K-2 clauses to generate X1

Number of new variables = 2

C1 to be replaced with 3 clauses each having 3 literals.

Replacing C1 by a sequenceof clauses to generate X1:

**X1 =(w1 ∨ -w2 ∨ y11) ∧ (-y11 ∨ w3 ∨ y12) ∧ (-y12 ∨ -w4 ∨ -w5)**

**Reduction of C2 to X2:**

C2 = (-w3 ∨ w4)

Here,

w3, w4 are literals

Let K = Number of literals in C2

Therefore K = 2

Since K = 2, we need to introduce a new variables yi1 and replace C2 with 2 clauses to generate X2

Number of new variables = 1

C2 to be replaced with 2 clauses each having 3 literals.

Replacing C2 by a sequenceof clauses to generate X2:

**X2 =(-w3 ∨ w4 ∨ y21) ∧ (-w3 ∨ w4 ∨ -y21)**

**The 3SAT formula F’ generated is conjunction of X1 and X2:**

F’ = X1 ∧ X2

**F’** **=(w1 ∨ -w2 ∨ y11) ∧ (-y11 ∨ w3 ∨ y12) ∧ (-y12 ∨ -w4 ∨ -w5) ∧ (-w3 ∨ w4 ∨ y21) ∧ (-w3 ∨ w4 ∨ -y21)**

**Truth Table:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **w1** | **w2** | **w3** | **w4** | **w5** | **F** | **y11** | **y12** | **y21** | **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 |

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:**

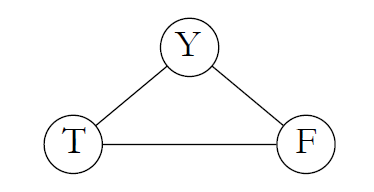
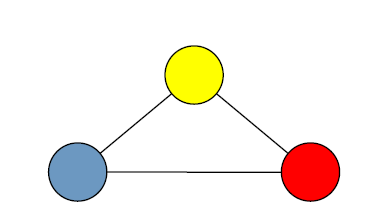
In a 3Col graph, 3 special vertices are created T(true), F(false) and Y(Neutral)

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*

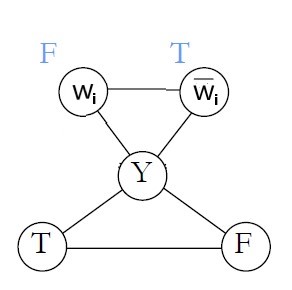
Generally, for the literal in the 3SAT:

* Either wi has color of T and -wi has color of F

Or

* wi has color of F and - wi has color of T

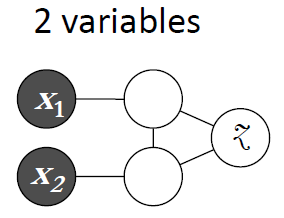
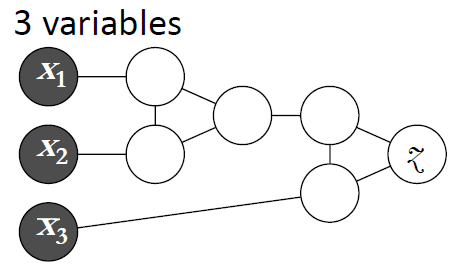
Every literal in the 3SAT should be connected to vertex Y in order to form the structure like Fig 1. Also no two adjacent vertex can have the same color



*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 X1 or X2 is colored T
* Z must be colored F, if X1 and X2 are colored F

3 Variable

* If at least one literal (X1, X2 orX3) is colored T, then Z can be colored T

**SOLUTION:**

Converting the below clauses to 3Col graph:

**(-w3 ∨ w4 ∨ y21) ∧ (-w3 ∨ w4 ∨ -y21)**

Where:

w3 = T, w4 = T, y21 = T

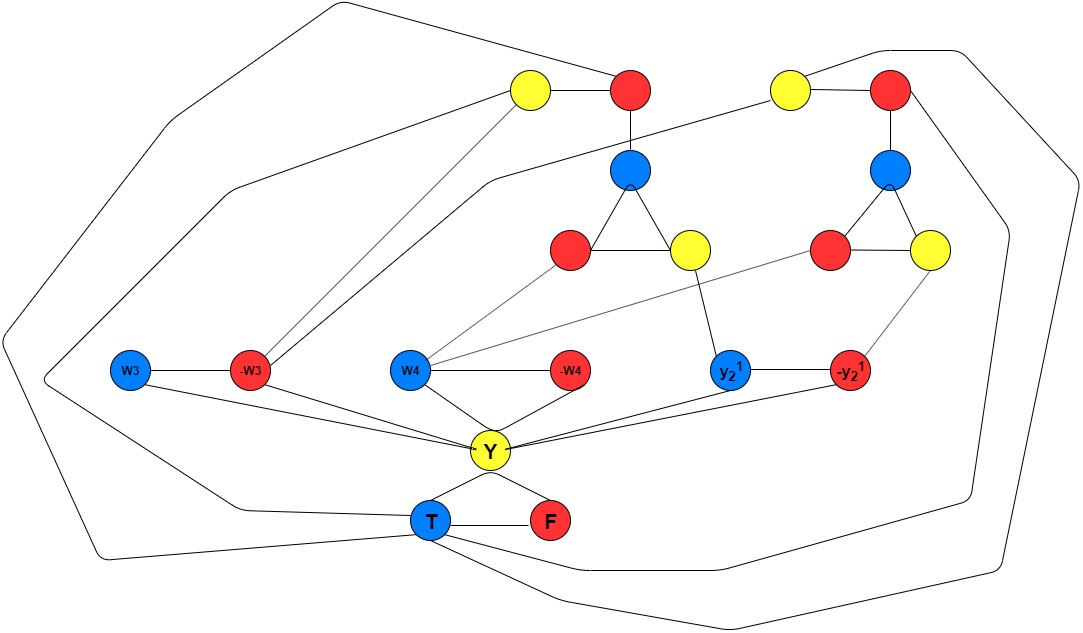
****

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.

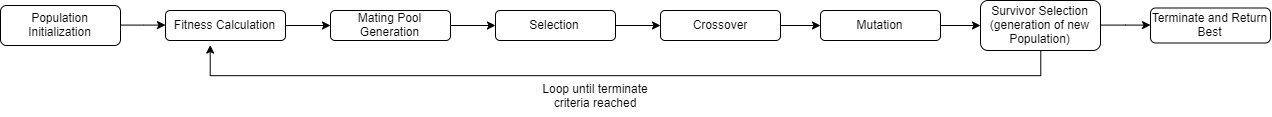


Fig 5

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
7. **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:
   1. Get the maximum fitness value from the list of individual fitness
   2. In order to prevent non selection of individuals with maximum fitness, add a token amount of 1 to the maximum fitness value
   3. 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. Let N be the number of parents to be selected for mating
7. Compute the distance P between successive points: P = F/N; where F is the sum of minimized fitness values
8. 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
9. Select the chromosomes whose range contains a marker

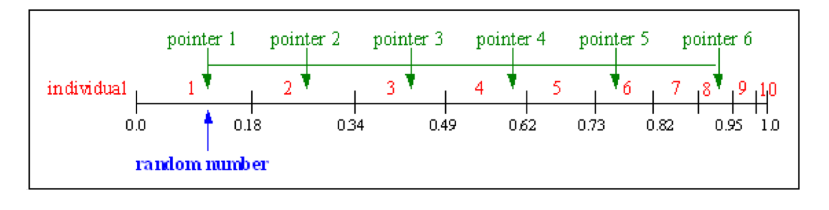
 *Reference: Lecture Slides*

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 Pm (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 Pm 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 Pm in order to perform Mutation
2. If the Pm 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 Pm in order to perform Mutation
2. If the Pm 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  
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: 1 best: 22471494.611521076  
iteration: 6 best: 22445803.09607827  
iteration: 18 best: 22381432.571599867  
iteration: 20 best: 22028068.21195185  
iteration: 21 best: 21955852.40700673  
iteration: 30 best: 21921738.813176632  
iteration: 31 best: 21666475.04099247  
iteration: 82 best: 21365556.16013506  
iteration: 89 best: 21042651.88437949  
iteration: 103 best: 21031746.408530228  
iteration: 104 best: 20509612.976430297  
iteration: 116 best: 20473174.571527984  
iteration: 124 best: 20462283.792039208  
iteration: 139 best: 20279091.428633668  
iteration: 142 best: 19928781.260843072  
iteration: 143 best: 19920575.170052335  
iteration: 171 best: 19269975.817974314  
iteration: 199 best: 19196564.28378297  
iteration: 222 best: 18792686.16886938  
iteration: 224 best: 18683452.916235976  
iteration: 236 best: 18555536.357503258  
iteration: 239 best: 18468393.609336212  
iteration: 242 best: 18122968.84318509  
iteration: 250 best: 18105524.461141914  
iteration: 251 best: 17994892.143396627  
iteration: 255 best: 17559822.876770627  
iteration: 267 best: 17485825.977828786  
iteration: 273 best: 17344189.54141799  
iteration: 284 best: 17196553.4022584  
iteration: 287 best: 16977270.32340074  
iteration: 290 best: 16719163.616200127  
iteration: 292 best: 16712863.581430962  
iteration: 293 best: 16353472.270283708  
iteration: 320 best: 16317326.473226584  
iteration: 321 best: 16285014.52048116  
iteration: 327 best: 16192161.77193501  
iteration: 333 best: 15233220.79965791  
iteration: 356 best: 15012359.801212704  
iteration: 383 best: 14845820.730503779  
iteration: 389 best: 14638014.928974953  
iteration: 409 best: 14508867.93741881  
iteration: 410 best: 14348376.129717464  
iteration: 415 best: 14314322.11319841  
iteration: 427 best: 14154810.29485501  
iteration: 434 best: 14150536.0858882  
iteration: 436 best: 14061038.964746851  
iteration: 446 best: 13823190.693310272  
iteration: 487 best: 13486965.515880967  
iteration: 497 best: 13408987.664876346  
iteration: 497 best: 13390423.806329755  
Total iterations: 500  
Best Solution: 13390423.806329755

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 23348960.301562894  
iteration: 0 best: 23323170.89546207  
iteration: 0 best: 22555594.698883668  
iteration: 0 best: 22252722.579482093  
iteration: 1 best: 22070702.77454176  
iteration: 16 best: 21978353.61805722  
iteration: 18 best: 21966129.415194068  
iteration: 18 best: 21849123.76435778  
iteration: 21 best: 21623848.160289213  
iteration: 167 best: 21167333.942073118  
Total iterations: 500  
Best Solution: 21167333.942073118

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 23415393.180840787  
iteration: 0 best: 23121301.587285012  
iteration: 1 best: 22794557.2945066  
iteration: 2 best: 22560631.010755982  
iteration: 5 best: 22550589.833587542  
iteration: 5 best: 22472070.31785017  
iteration: 7 best: 22404961.62329707  
iteration: 8 best: 22267084.13041238  
iteration: 9 best: 21724831.68712128  
iteration: 31 best: 21662422.953503285  
iteration: 64 best: 21616598.838939704  
iteration: 180 best: 21481288.87342979  
Total iterations: 500  
Best Solution: 21481288.87342979

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 22716852.42092712  
iteration: 0 best: 22681745.801719062  
iteration: 0 best: 22633243.093624208  
iteration: 1 best: 22193186.810260613  
iteration: 19 best: 22172046.97329495  
iteration: 24 best: 22146575.61708482  
iteration: 28 best: 21967760.176711243  
iteration: 51 best: 21826514.439062778  
iteration: 58 best: 21755268.730199426  
iteration: 66 best: 21692711.486811504  
iteration: 78 best: 21571014.869268507  
iteration: 79 best: 21417833.44414777  
iteration: 91 best: 21391891.227737818  
iteration: 112 best: 21001432.820436325  
iteration: 160 best: 20921936.97124889  
iteration: 169 best: 20888270.041474555  
iteration: 219 best: 20838171.54600735  
iteration: 220 best: 20763454.03368677  
iteration: 222 best: 20249251.50268401  
iteration: 262 best: 20245623.387016937  
iteration: 263 best: 20218696.075973444  
iteration: 276 best: 19907943.671679664  
iteration: 348 best: 19447077.437339798  
iteration: 441 best: 18937470.183063567  
iteration: 453 best: 18817648.80621373  
iteration: 467 best: 18706557.71867159  
Total iterations: 500  
Best Solution: 18706557.71867159

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 3992001.979969066  
Total iterations: 500  
Best Solution: 3992001.979969066

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 4135500.274362368  
Total iterations: 500  
Best Solution: 4135500.274362368

**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.

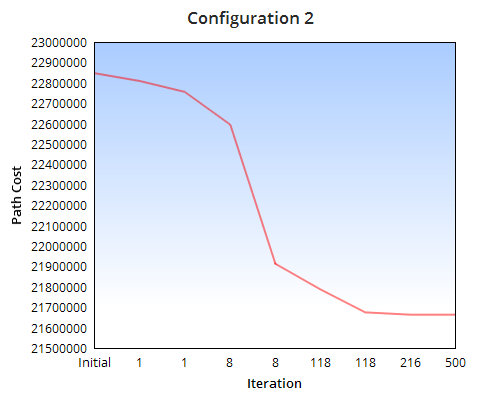
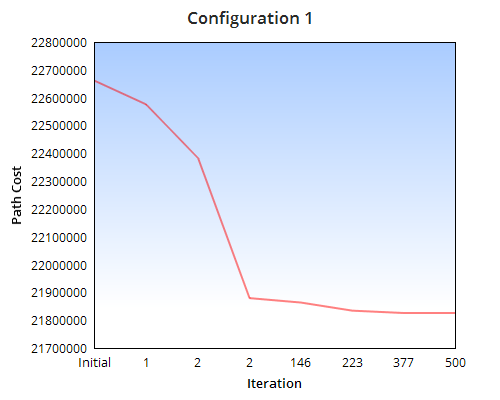
In configuration from 1 to 6, the initial population selection was Random which generated the best initial solution i.e. the path cost on the population generated randomly**.** The genetic algorithm was run over this initial path solution with a combination of various selection, crossover and Mutation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

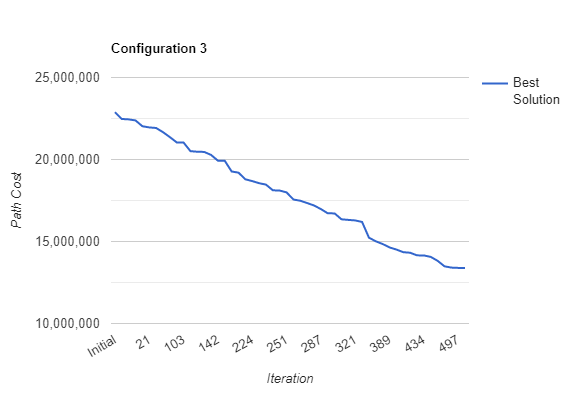
**Best Result**:

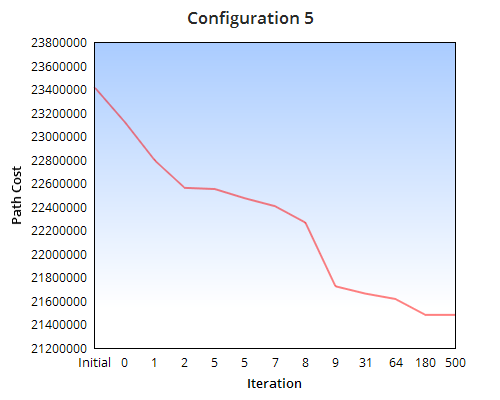
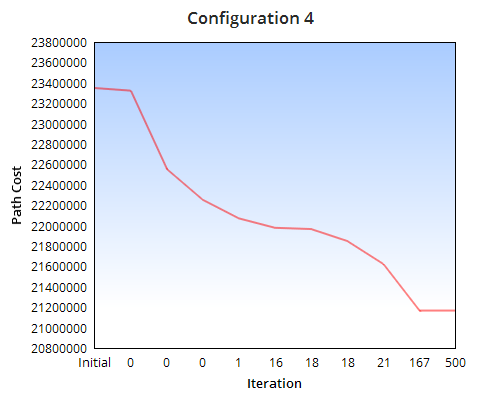
The Genetic Algorithm performs extremely well in case of Configuration 3 (Random Initial Population, Stochastic selection, Uniform Crossover, Reciprocal Exchange Mutation).

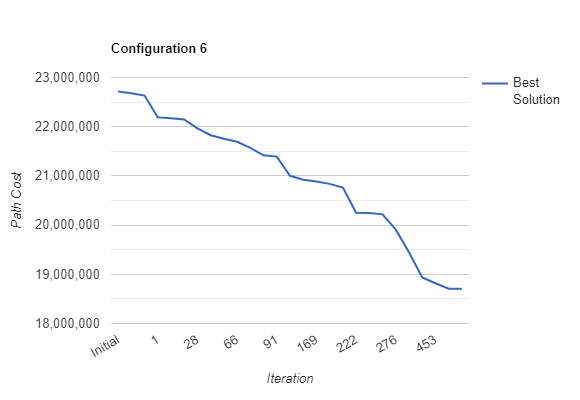
* The path cost is reduced in a well-informed pattern, providing the optimized path cost as: **13390423.806329755**
* Over 500 iterations, the path cost is continuously reduced as it can be seen from the pattern and a reduction of **41.49 %** in path cost is observed
* The random initial population is 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.

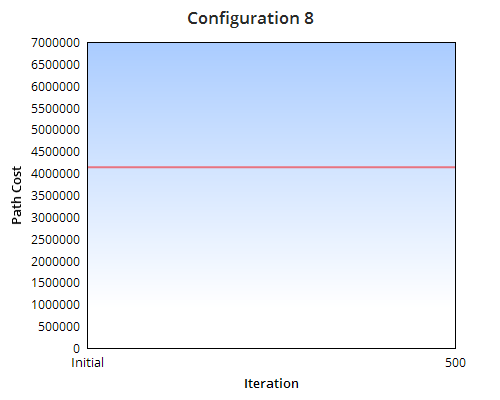
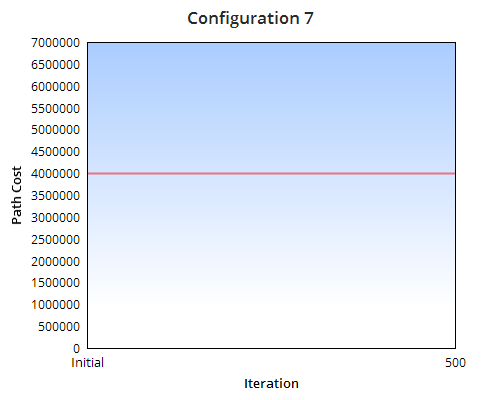
**Result comparison over different configurations**











**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 | 1339042 |
| 6 | Random | Uniform Crossover | Inversion Mutation | Stochastic Universal Sampling | 22716852 | 18706557 |

1. Configuration 3 and 4 perform decent compared to Configuration 1 and 2 as it can be observed that there in config. 1 ,2 the new generation generated is not able to provide a better solution in most of the iterations. In conf. 4,5 the same behaviour is obtained but there is good reduction in path cost compared to config 1,2. A reduction of average 9% is observed for config 4,5 compared to a reduction in path cost of average 5% for config. 1,2
2. 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: 3 best: 109762261.6011975  
iteration: 18 best: 108319051.96825494  
iteration: 19 best: 106307870.4790285  
iteration: 22 best: 105045021.13261868  
iteration: 23 best: 102212463.36227827  
iteration: 87 best: 101645990.80074452  
iteration: 128 best: 99285976.1710425  
iteration: 129 best: 97817981.88281642  
iteration: 143 best: 94202183.04160501  
iteration: 160 best: 92820903.8550797  
iteration: 196 best: 92539286.12781566  
iteration: 197 best: 90342362.09214623  
iteration: 211 best: 88426159.3795466  
iteration: 216 best: 86191306.18735704  
iteration: 220 best: 84843050.88012263  
iteration: 224 best: 83116985.97675794  
iteration: 234 best: 83041750.89782763  
iteration: 240 best: 82199166.69534506  
iteration: 245 best: 80451341.56375124  
iteration: 246 best: 79165653.54483077  
iteration: 248 best: 77671102.22200024  
iteration: 248 best: 77250614.94213647  
iteration: 258 best: 76020927.19000682  
iteration: 269 best: 74057144.354884  
iteration: 272 best: 71616170.74415562  
iteration: 280 best: 69538223.4676283  
iteration: 283 best: 68356452.15675327  
iteration: 285 best: 66627676.3439675  
iteration: 287 best: 66001381.93830251  
iteration: 306 best: 65055000.47166632  
iteration: 316 best: 64640007.43537626  
iteration: 319 best: 64551885.722757645  
iteration: 322 best: 63369179.420318425  
iteration: 326 best: 63028074.0788957  
iteration: 329 best: 62281658.57473408  
iteration: 334 best: 62026223.69549197  
iteration: 335 best: 61700278.34609314  
iteration: 344 best: 61483020.57547593  
iteration: 350 best: 58871033.76832249  
iteration: 358 best: 58871016.210657984  
iteration: 359 best: 58352627.666143306  
iteration: 360 best: 56939103.98511548  
iteration: 367 best: 56871536.538853034  
iteration: 369 best: 56767896.4102546  
iteration: 370 best: 56616554.133731976  
iteration: 370 best: 56387810.35797626  
iteration: 372 best: 55074210.25411995  
iteration: 372 best: 53159799.0796051  
iteration: 390 best: 52143710.17498062  
iteration: 397 best: 51867668.94950251  
iteration: 405 best: 51648440.68529818  
iteration: 421 best: 51067965.59319388  
iteration: 425 best: 51024925.54263022  
iteration: 438 best: 50382649.66747541  
iteration: 488 best: 50244197.43898067  
iteration: 496 best: 49904871.83830272  
iteration: 499 best: 49158772.46703622  
Total iterations: 500  
Best Solution: 49158772.46703622

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 113277957.51939209  
iteration: 0 best: 111611077.83245206  
iteration: 1 best: 111465874.04891317  
iteration: 1 best: 109695127.90957215  
iteration: 1 best: 109694674.66272767  
iteration: 15 best: 108455420.36192057  
iteration: 15 best: 107805812.50165954  
iteration: 31 best: 106970048.6187906  
iteration: 162 best: 106635346.61122002  
iteration: 273 best: 106581975.57395886  
iteration: 372 best: 106071882.46507831  
iteration: 373 best: 104937380.27823988  
Total iterations: 500  
Best Solution: 104937380.27823988

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 117477482.39030655  
iteration: 0 best: 116549562.44317846  
iteration: 0 best: 113314781.0993942  
iteration: 2 best: 112457789.03532521  
iteration: 2 best: 110180975.23067638  
iteration: 2 best: 110055379.58138493  
iteration: 5 best: 109282427.22952312  
iteration: 15 best: 107719290.02683143  
iteration: 54 best: 107665486.48004267  
iteration: 105 best: 106927917.14532502  
iteration: 166 best: 104785585.68324858  
Total iterations: 500  
Best Solution: 104785585.68324858

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 112612990.78264506  
iteration: 2 best: 110964705.5039848  
iteration: 4 best: 110140811.54028063  
iteration: 12 best: 107542525.57357179  
iteration: 76 best: 103720742.69761553  
iteration: 240 best: 102641482.5136445  
iteration: 284 best: 102103165.62575407  
iteration: 289 best: 100240982.22326396  
iteration: 302 best: 97639515.88916612  
iteration: 332 best: 97570357.008983  
iteration: 364 best: 97411118.67853072  
iteration: 366 best: 95891347.5903513  
iteration: 399 best: 95837288.40307395  
iteration: 407 best: 94656841.53962125  
iteration: 449 best: 93031051.05169089  
iteration: 450 best: 93007841.24993697  
Total iterations: 500  
Best Solution: 93007841.24993697

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 7206705.038676705  
Total iterations: 500  
Best Solution: 7206705.038676705

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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: 7266946.156008778  
Total iterations: 500  
Best Solution: 7266946.156008778

**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.

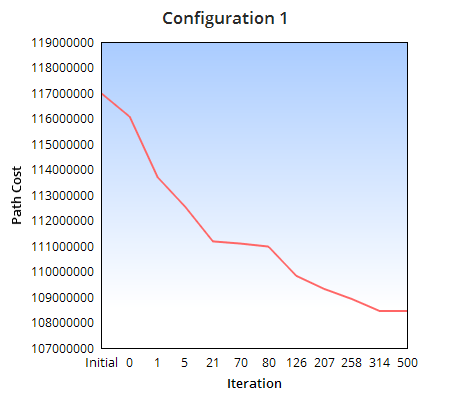
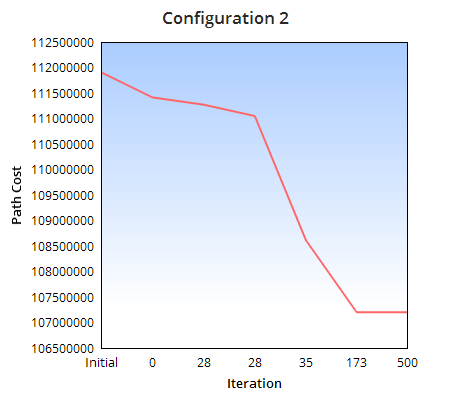
In configuration from 1 to 6, the initial population selection was Random which generated the best initial solution i.e. the path cost on the population generated randomly**.** The genetic algorithm was run over this initial path solution with a combination of various selection, crossover and Mutation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

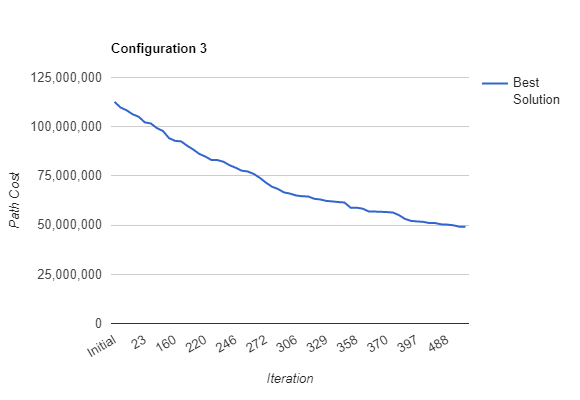
**Best Result**:

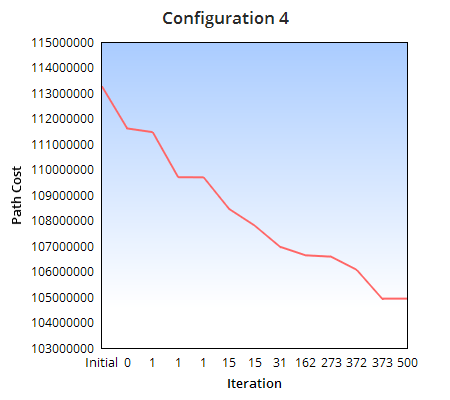
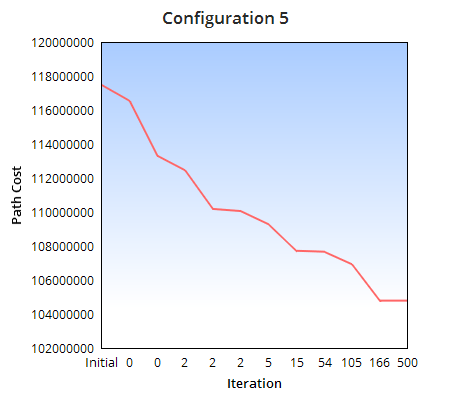
The Genetic Algorithm performs extremely well in case of Configuration 3 (Random Initial Population, Stochastic selection, Uniform Crossover, Reciprocal Exchange Mutation).

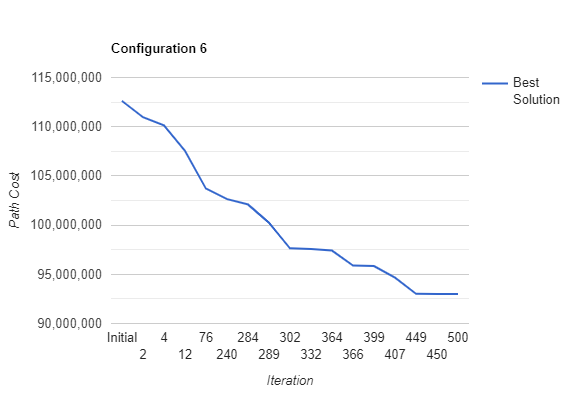
* The path cost is reduced in a well-informed pattern, providing the optimized path cost as: **49158772.46703622**
* Over 500 iterations, the path cost is continuously reduced as it can be seen from the pattern and a reduction of **56.33 %** in path cost is observed
* The random initial population is 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.

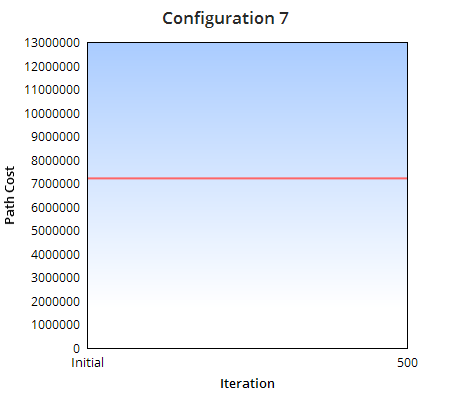
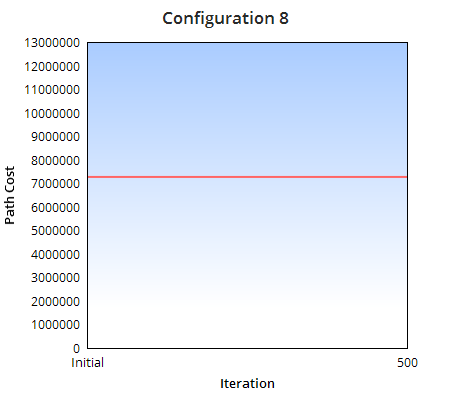
**Result comparison over different configurations**





**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 | 49158772 |
| 6 | Random | Uniform Crossover | Inversion Mutation | Stochastic Universal Sampling | 112612990 | 93007841 |

1. Configuration 4 and 5 perform decent compared to Configuration 1 and 2 as it can be observed that there in config. 1,2 the new generation generated is not able to provide a better solution in most of the iterations. In conf. 4,5 the same behaviour is obtained but there is good reduction in path cost (**average 10 %**) compared to config 1,2 (**average 5%**).
2. 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**

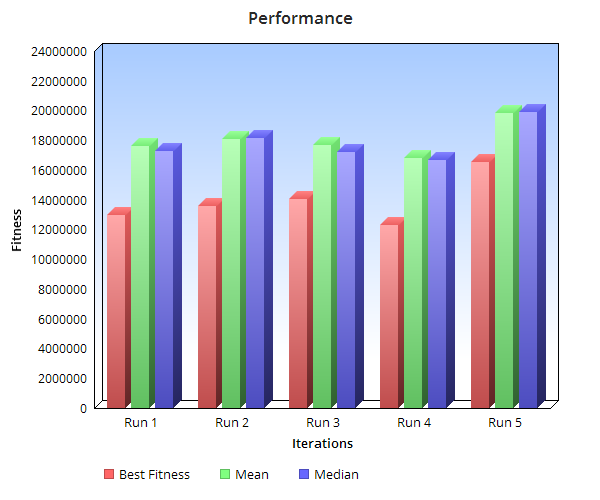
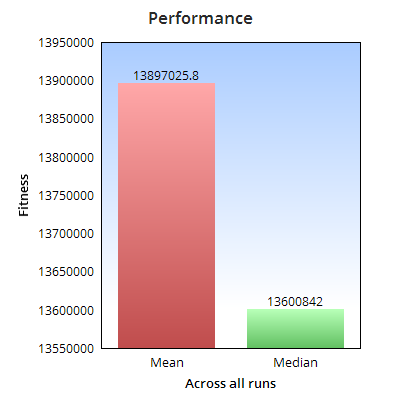
**3.2. Result Analysis: inst-5.tsp**

**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.

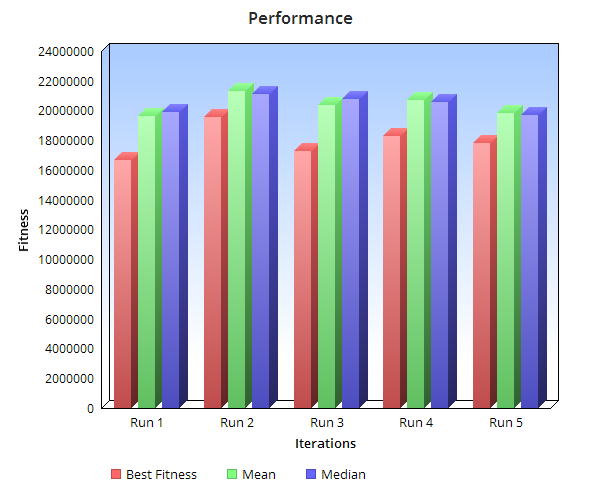
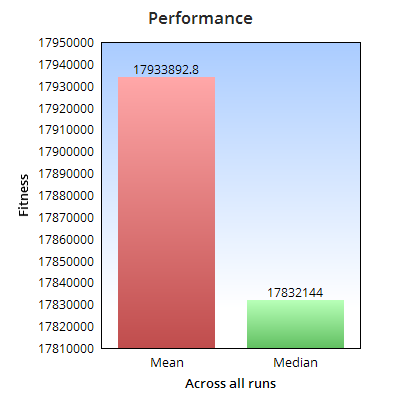
|  |
| --- |
| **Configuration 3** |
| **Iteration:** | **Run 1** | **Run 2** | **Run 3** | **Run 4** | **Run 5** | **Mean** | **Median** |
| **Best Fitness:** | 12997902 | 13600842 | 14071833 | 12270101 | 16544451 | **13897025.8** | **13600842.0** |
| **Mean:** | 17596324.98 | 18083977.67 | 17648886.20 | 16807751.15 | 19836495.56 |
| **Median:** | 17302014.00 | 18131811.00 | 17209851.00 | 16671565.00 | 19899200.00 |



**Conclusion:**

It can be concluded that on average the genetic algorithm with best mean initial solution of

|  |
| --- |
| **Configuration 6** |
| **Iteration:** | **Run 1** | **Run 2** | **Run 3** | **Run 4** | **Run 5** | **Mean** | **Median** |
| **Best Fitness:** | 16659487 | 19583949 | 17275516 | 18318368 | 17832144 | **17933892.8** | **17832144.0** |
| **Mean:** | 19626897.28 | 21338924.41 | 20397439.76 | 20707067.42 | 19847507.29 |
| **Median:** | 19888147.00 | 21081955.50 | 20786399.00 | 20542038.00 | 19709730.00 |



**Conclusion:**

It can be concluded that on average the genetic algorithm with best mean initial solution of

**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:

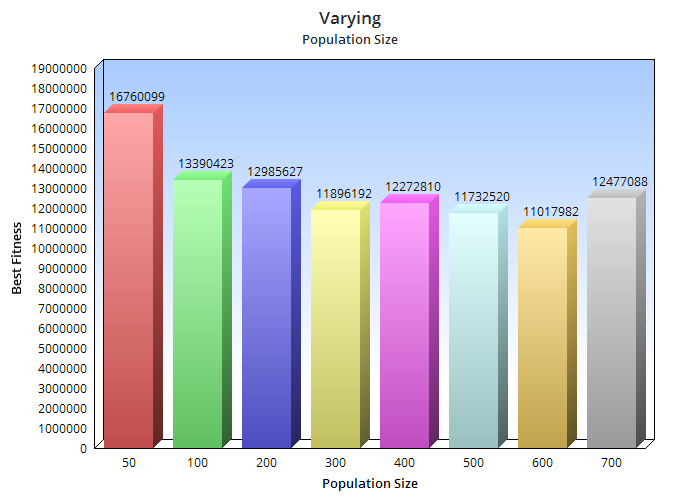
* Population Size
* Mutation Rate

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:** | 16760099 | 13390423 | 12985627 | 11896192 | 12272810 | 11732520 | 11017982 | 12477088 |



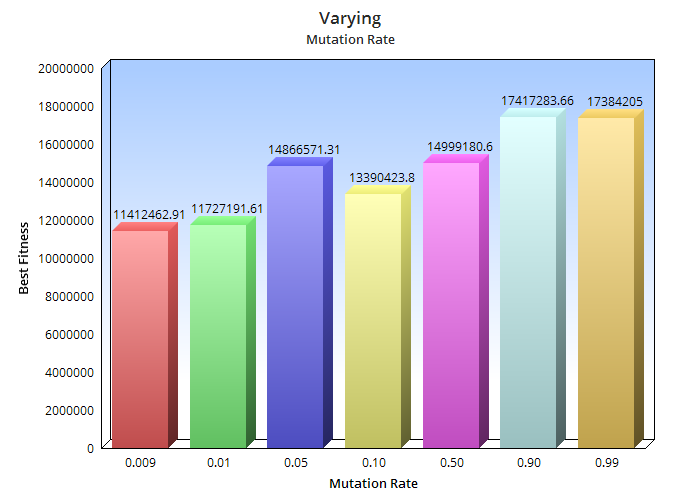
**Conclusion:**

* 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 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.

1. **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:** | 11412462.91 | 11727191.61 | 14866571.31 | 13390423.80 | 14999180.60 | 17417283.66 | 17384205.05 |



**Conclusion:**

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

* As the Pm is reduced the genetic algorithm performs well.
* On increasing the Pm, the genetic algorithm is not able to reach the best optimal solution it is capable of.
* A large Pm, shows that the genetic algorithm gets reduced to a random search
* 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. 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 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.