Part 2: Genetic Algorithm

Student ID: R00183771

First Name: Pavithra Ramasubramanian

Last Name/Surname : Ramachandran

(The lecture slides were extensively used as a reference to formulate this report)

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. [1] Genetic algorithm comprises of these processes:

- Initialization of Population
- Selection of parent genes
- Crossover between parent genes to form children
- Mutation in the child genes

Initialization of Population

This can be done in two ways:

- Random Selection : Random Selection involves randomly selecting the order in which genes in a chromosome will be placed.
- Heuristic approach : This is similar to the TSP approach where the genes are cities, and ordering the cities in the cost effective route gives a sequence of genes which is the chromosome. Here, we choose a first city(gene) randomly. Then, for every city(gene) that is added, the cost(fitness) is calculated with all the cities(genes) already present in the route(budding-chromosome) and the city(gene) is appended wherever appropriate.

Selection of Parent genes

Stochastic Universal Sampling(SUS) technique is used in the selection of parents in my code.

Stochastic Universal Sampling

Stochastic Universal Sampling is a selection technique which enables selection of two parents from a pool of parents for mating based on the fitness parameter. It ensures that parents are picked without a bias.

SUS Algorithm:

- Retrieve the fitness f of all chromosomes in the population P.
- Apply a minimization technique on these fitness values(1/f). This is done as we are implementing the TSP problem, where lesser the distance(cost), the better.
- Sum up the minimized fitness values of all chromosomes S = sum(1/(f of each gene))
- Calculate the probability of each gene's minimized fitness Pr = (1/f)/S

- Now, create a fitness probabilty scale.
- Also, create a scale with N equally spaced pointers such that each pointer points at some range in the fitness probability scale.
- If a pointer points at a particular fitness, then pick that gene.
- After picking N such genes, randomly pick two parents for mating.

For instance, let us assume we have 10 parents with fitnesses as below:

Parent	**Fitness**
1	300
2	200
3	100
4	50
5	25
6	25
7	60
8	70
9	100
10	200

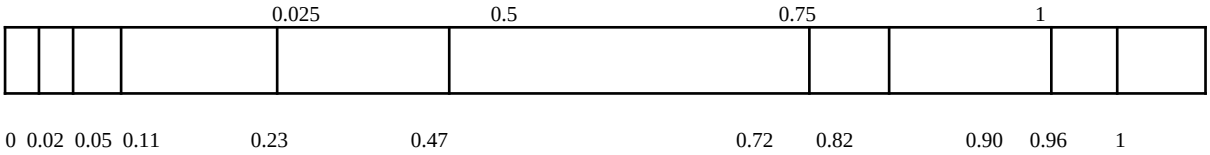
On computing the steps in the SUS algorithm, we obtain these values:

Parent	**Fitness**	**1/f**	**S**	**Pr = (1/f)/S**	**Fitness Probability Scale**
1	300	0.00333333	0.16428571	0.02028986	0.02028986
2	200	0.00500000	0.16428571	0.03043478	0.05072464
3	100	0.01000000	0.16428571	0.06086957	0.11159420
4	50	0.02000000	0.16428571	0.12173913	0.23333333
5	25	0.04000000	0.16428571	0.24347826	0.47681159
6	25	0.04000000	0.16428571	0.24347826	0.72028986
7	60	0.01666667	0.16428571	0.10144928	0.82173913
8	70	0.01428571	0.16428571	0.08695652	0.90869565

9	100	0.01000000	0.16428571	0.06086957	0.96956522
10	200	0.00500000	0.16428571	0.03043478	1.00000000

The fitness probability scale will look like the following scale. This scale will be divided into N equal parts. Let us consider N to be 4 and the random pointer to start be 1/N = 0.025.

N- equal divisions



Fitness Probability Scale

SUS is a good selection method as:

- It eliminates bias by choosing genes uniformly from the fitness scale. Although certain genes might have very less fitness probability, this will ensure those genes don't die.
- Less fit genes might contribute to the formation of the best fit children.

Crossovers

Crossovers are techniques used in Genetic algorithms to produce child chromosomes. The two types of crossovers used in my code are:

- Uniform Crossover
- Partially Mapped Crossover (PMX)

Uniform Crossover

Uniform Crossover follows the following steps to produce new children chromosomes.

Let us assume we have two parent chromosomes P1 and P2 with genes as shown below:

P1:

1	2	3	4	5	6	7	8

P2:

6	2	4	3	5	1	8	7

A uniform Crossover between these two parents would involve the following steps:

- Randomly pick fixed positions.
- Retain the genes in the fixed positions in the child too.
- For the remaining positions, iterate through the other parent and fill in the empty positions with the genes not already present in the child.

Let us assume the fixed positions to be: 1,3,5,7. Genes are retained in the same positions in the children C1 and C2.

C1:

	2		4		6		8

C2:

	2		4		6		7

Now, from P2 genes are sequentially placed in the empty spaces in C1 and the same happens from P1 to C2 provided the gene is not already present in the children

C1:

3	2	5	4	1	6	7	8

C2:

4	2	5	4	3	6	8	7

Partially Mapped Crossover(PMX)

This Technique follows the below steps to create children.

Let us assume we have two parent chromosomes P1 and P2 with genes as shown below:

P1:

1	2	3	4	5	6	7	8

P2:

6	2	4	3	5	1	8	7

A PMX between these two parents would involve the following steps:

- Randomly pick a block of genes.
- Exchange the genes in the block , that is, this block will go to C1 from P2 and C2 from P1.
- For the remaining positions, iterate through the same parent and fill in the empty positions with the genes not already present in the child and in the same index positions. If the index positions have genes in the same positions, then in a cyclic manner, look for a gene through the children.

Let us assume the block of genes to be exchanged be 1 - 5. The children would look like this if the randomly chosen gene block is exchanged.

C1:

	2	4	3	5	1		

C2:

	2	3	4	5	6		

Now, we fill the empty spaces with the numbers that do not appear the child already from the same parent.

C1:

	2	4	3	5	1	7	8

C2:

	2	3	4	5	6	8	7

Since 1 is already present in C1, we find 1 in C1 and look for the gene in the same index in C2 - 6. Similarly, we find 6 in C2 and then look for the gene in the same index in C1 - 1 and fill the empty genes.

C1:

6	2	4	3	5	1	7	8

C2:

1	2	3	4	5	6	8	7

Mutation:

Mutation is performed to create more changes in the children chromosomes so that we have a variation in the population. The two types of mutation implemented in my code are:

- Reciprocal Exchange Mutation
- Inversion Mutation

Reciprocal Exchange Mutation

In the child, two random genes are interchanged.

C1:

1	2	3	4	5	6	8	7

C1 after mutation(Random places chosen - 0,6):

8	2	3	4	5	6	1	7

Inversion Mutation:

In this mutation, an entire block of genes is reversed.

C1:

1	2	3	4	5	6	8	7

C1 after mutation(Random places chosen - 3,6):

8	2	3	8	6	5	4	7

Discussion of Output:

Random Seed : 183771

Input files used: inst-4.tsp, inst-6.tsp and inst-16.tsp

An excel file is present in the folder by the name : Output Aggregate. This file has outputs exhibiting :

Variation in configurations

The 8 configurations required to run are as follows:

1. Random Solution, Random Selection, Uniform Crossover, and Inverse Mutation

2. Random Solution, Random Selection, PMX Crossover, Reciprocal Exchange Mutation

3. Random Solution, Stochastic Universal Sampling, Uniform Crossover, Reciprocal Muatation

4. Random Solution, Stochastic Universal Sampling, PMX Crossover, Reciprocal Muatation

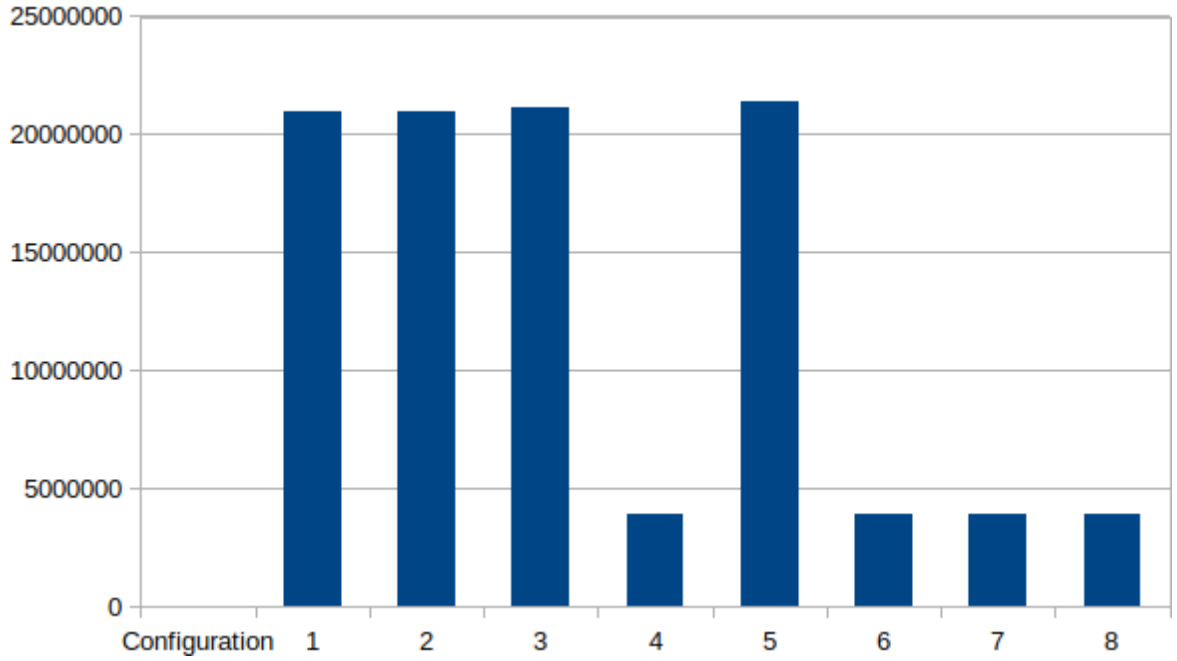
5. Random Solution, Stochastic Universal Sampling, PMX Crossover, Inversion Mutation

6. Random Solution, Stochastic Universal Sampling, Uniform Crossover, Inversion Mutation

7. Heuristic Solution, Stochastic Universal Sampling, PMX Crossover, Reciprocal Muatation

8. Heuristic Solution, Stochastic Universal Sampling, Uniform Crossover, Inversion Mutation

The below graph represents the variation in best solution across the 8 configurations.



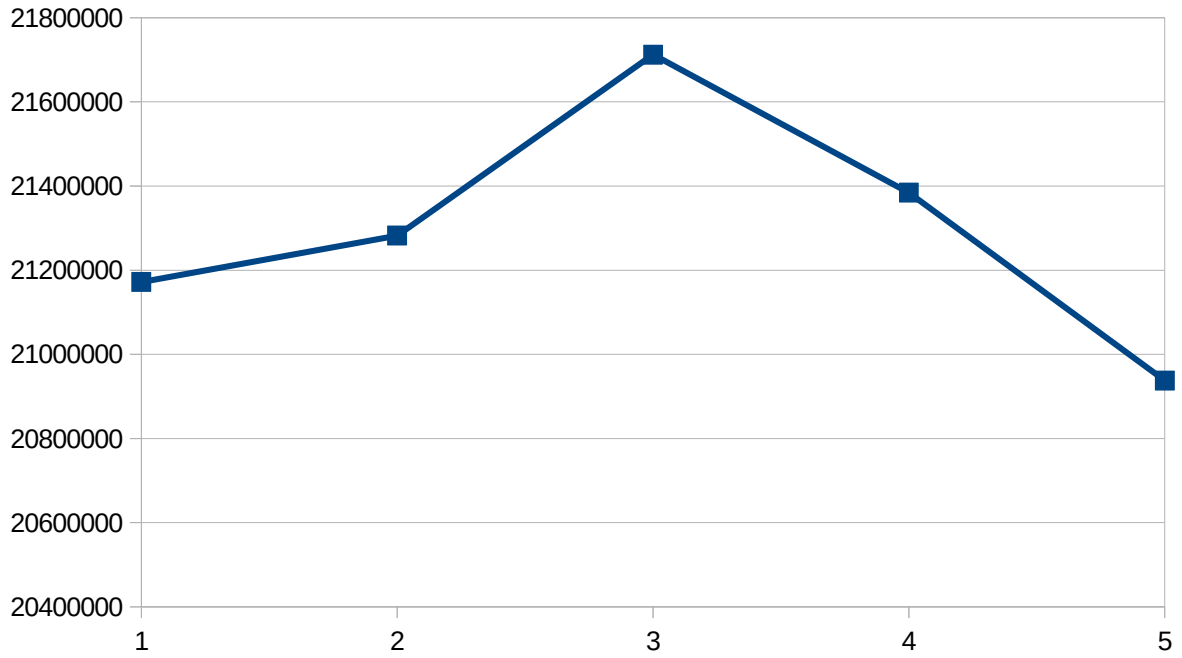
(Input file used: inst-4.tsp)

The heauristic solutions provide the best solutions.

Variation in Runs(5 runs)

The below chart depicts the variation in values of best final solutions across 5 runs through configuration 2.

Mean = 21297735.2

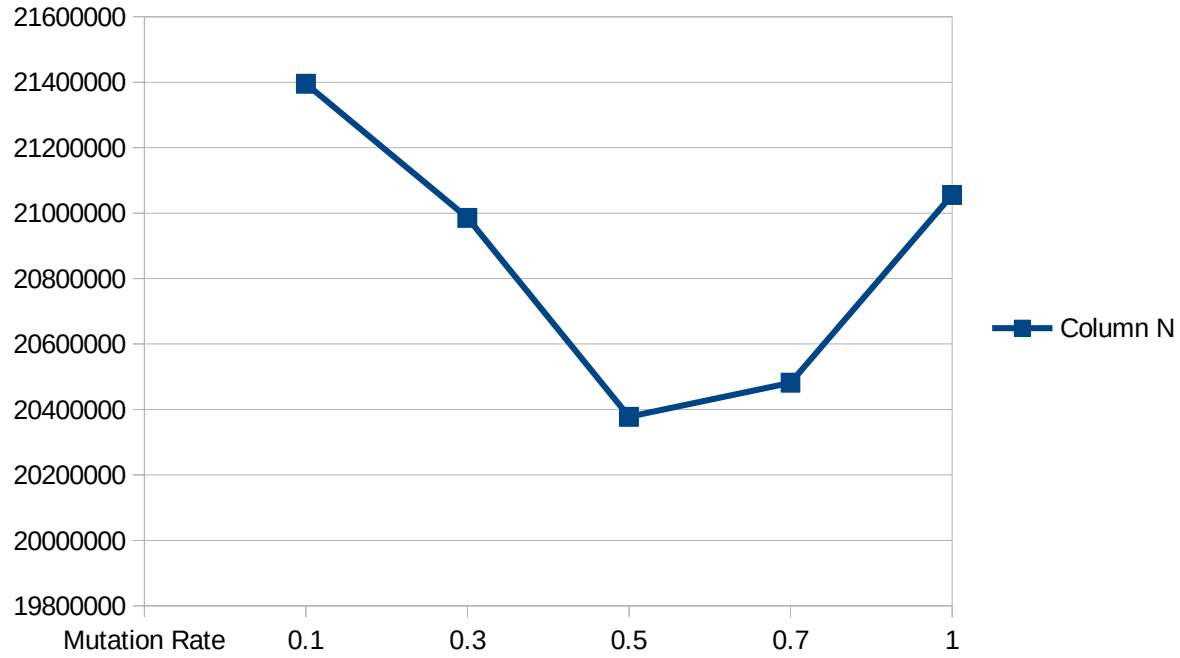


Input file : Inst-5.tsp

The 5 runs do not influence the generation of the best solution.

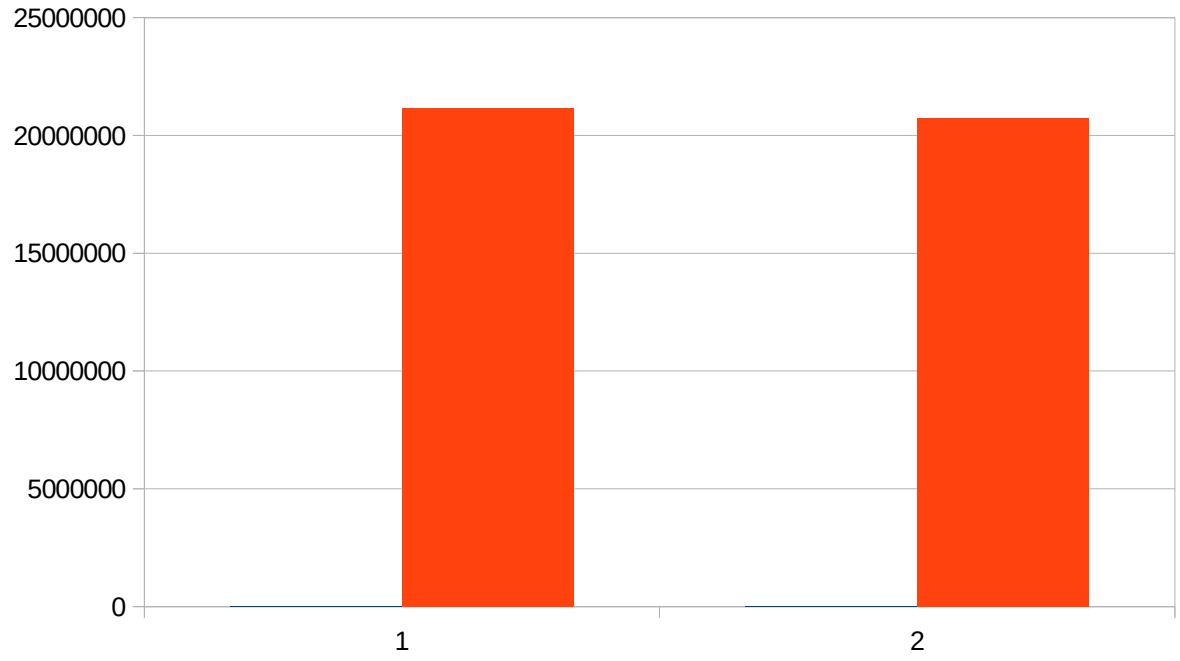
Variation in Mutation Rate

Mutation rate increases the after the 0.5.



Variation in Population

As the population increases , best solution is better.



References:

1. https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3