

**RCC INSTITUTE OF INFORMATION TECHNOLOGY**

**DESIGN & ANALYSIS OF ALGORITHM (CS-501)**

**TRAVELING SALESMAN PROBLEM USING GENETIC ALGORITHM**

**ABSTRACT**

Genetic algorithm is an adaptive heuristic searching algorithm which belongs to a part of evolutionary algorithms. It is based on the idea of natural selection and genetic traits, related to Charles Darwin’s evolution theory. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

This algorithm reflects the way of natural selection and the process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the properties of the parents and will be given to the next generation.If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on happening and at the end, a generation with the most fit individuals will be discovered.

This notion can be applied for a search problem. We consider a set of solutions for a problem and select the set of best ones out of them. Five phases are considered in a genetic algorithm.

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

The traveling sales person is a classic optimization problem which can be solved with the help of genetic algorithm. Given a set of cities and a set of values corresponding to each element in the Cartesian product of the set of cities and itself. Each value represents the distance between the two elements of the corresponding pair. It is used to find the shortest route between a set of points and locations that must be visited.

A genetic algorithm will be used to select several solutions from a pool of solutions, and by using a fitness function determine which solution provided the best result. The solution itself is a list of cities, in which the order of the list corresponds to the sequence in which the cities were visited. The fitness function determines a given solution's total distance traveled by taking the sum of the distances of each pair of successive elements of the solution's list.The solution whose fitness function produces the smallest value is the best fit solution, or one that produces that shortest distance between n cities in which no city was traveled to more than once and every city was traveled to.

**Input:**

* The number of cities to visit (determines the GA chromosome length)
* Intercity distances (can be stored in a 2D array for efficient access)
* The population size (number of tours in each GA generation)
* The maximum number of GA generations
* The GA crossover probability
* The GA mutation probability

**Output:**

* The tour (list of cities in the order they are visited) with the shortest length
* The length of the shortest tour
* The generation number for the best tour found

**METHODOLOGY**

**ALGORITHM:**

Functional or empirically developed mathematical models explicitly link a quantitative dependent variable to certain independent variables. The test time approach to build such models from observed quantitative data is known as regressions analysis. From the comparison it is understood that variation between the experimentally observed values and the predicted values are minimum and it confirmed the potential applicability of the mathematical model.

Genetic algorithm is a simplistic and heuristic approach to any problem. Here, an initial population is considered and from that, the optimal solution is calculated. Genetic algorithm can be broken into two halves, the first being the selection and the second one being the crossover and mutation processes. The Algorithm In the genetic algorithm process is as follows [1]:

Step 1. Determine the number of chromosomes, generation, and mutation rate and crossover rate value

Step 2. Generate chromosome-chromosome number of the population, and the initialization value of the genes chromosome-chromosome with a random value

Step 3. Process

Steps 4-7 until the number of generations is met

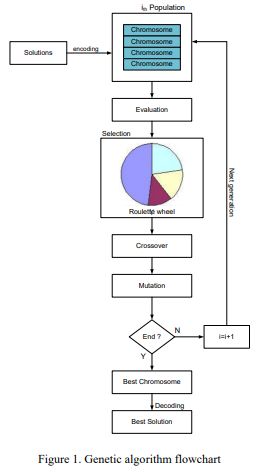
Step 4. Evaluation of fitness value of chromosomes by calculating objective function

Step 5. Chromosomes selection

Step 6. Crossover

Step 7. Mutation

Step 8. Solution (Best Chromosomes)



**MATHEMATICAL MODELING:**

In this paper, we consider N jobs on a number of unrelated parallel machines selected from a set of M potential machines so as to minimize the sum of earliness and tardiness penalties as well as machine costs. If a machine is selected to process any of the jobs (at least one job), a machine cost will be incurred in which they are independent. Different machines may operate at different speeds where different jobs have different earliness and tardiness penalties. Each machine processes one job at a certain time and each job should be completed on one machine. It means that preemption is not allowed. Each job has its own distinct due date that is fixed by the customer. Sequence dependent setup times are considered and triangular law of

inequality, i.e., sijm + sjkm ≥ sikmfor all jobs i, j, and k on machine m is satisfied.

We propose an integrated mixed-integer model for this problem. First, we give notations and

variables:

i, j job indicates where job 0 is a dummy job which is always at the first position on a machine

(i, j = 0, 1, …, N)

eiearliness penalty of job i

titardiness penalty of job i

βk machine cost (k = 1 ,…, K)

k machine index (k = 1 ,…, K)

Ci completion time of job i

Eiearliness of job i

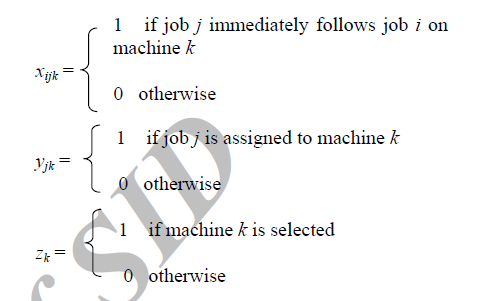
Titardiness of job i

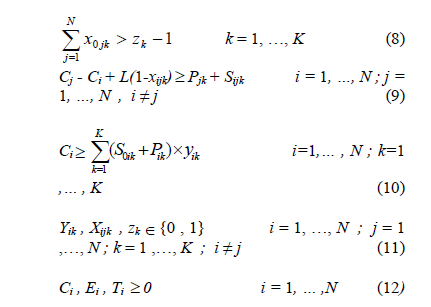
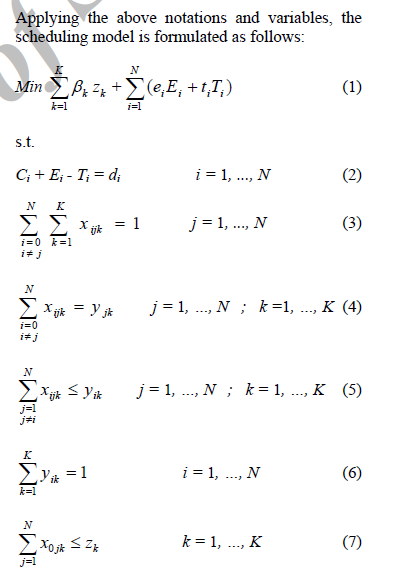
di due date of job i

L large positive number

Pikprocess time of job ion machine k (i =1,… N ;k =1,…, K)

Sijksetup time to switch from job ito job j on machine k



****

In this proposed model, Equation 1 is the objective function consisting of three terms. These terms are the machine cost, the total weighted earliness penalties, and the total weighted tardiness penalties, respectively. Equation 2 calculates the earliness or tardiness of job i based on its completion time. Equation 3 ensures that a job must be processed at one and only one position on a machine. Equation 4 illustrates that if job j is assigned to Machine *k*, it should come after one of the jobs including job 0. Equation 5 states that if

job j is assigned to Machine *k*, at least one job will immediately follow. Equation 6 ensures that eachjob assigned to one machine in which preemption is not allowed. Equation 7 ensures that if a machine is not selected, no job should go on it. Equation 8 guarantees that if a machine is selected,at least one job should be assigned to it. Equations 9 and 10 establish the relationship between the completion times of jobs *i*and *j* according to setup times as long as both jobs are assigned to the same machine. Equations 11 and 12 define zero-one variables and nonnegative variables, respectively.By applying data given in [4], we solve the proposed model in a small size by the Lingo software. The number of jobs is 4, the number of machines is 2, and objective function value is 93.

The sequence on the first machine is 1→ 4. The sequence on the second machine is 2→ 3.

**EXAMPLE ILLUSTRATION:**

Let us assume there is equality a + 2b + 3c + 4d = 30, genetic algorithm will be used to find the value of a, b, c, and d that satisfy the above equation. First, we should formulate the objective function, for this problem the objective is minimizing the value of function f(x) where f(x) = ((a + 2b + 3c + 4d) - 30). Since there are four variables in the equation, namely a, b, c, and d, we can compose the chromosome as follow: To speed up the computation, we can restrict that the values of variables a, b, c, and d are integers between 0 and 30.

**Step 1. Initialization**

For example we define the number of chromosomes in population are 6, then we generate random value of gene a, b, c, d for 6 chromosomes

Chromosome[1] = [a;b;c;d] = [12;05;23;08]

Chromosome[2] = [a;b;c;d] = [02;21;18;03]

Chromosome[3] = [a;b;c;d] = [10;04;13;14]

Chromosome[4] = [a;b;c;d] = [20;01;10;06]

Chromosome[5] = [a;b;c;d] = [01;04;13;19]

Chromosome[6] = [a;b;c;d] = [20;05;17;01]

**Step 2. Evaluation**

We compute the objective function value for each chromosome produced in initialization step:

F\_obj[1] = Abs(( 12 + 2\*05 + 3\*23 + 4\*08 ) - 30) =Abs((12 + 10 + 69 + 32 )-30 = Abs(123 - 30)= 93

F\_obj[2] = Abs((02 + 2\*21 + 3\*18 + 4\*03) - 30) = Abs((02 + 42 + 54 + 12) - 30)= Abs(110 - 30)= 80

F\_obj[3] = Abs((10 + 2\*04 + 3\*13 + 4\*14) - 30)= Abs((10 + 08 + 39 + 56) - 30)= Abs(113 - 30 = 83

F\_obj[4] = Abs((20 + 2\*01 + 3\*10 + 4\*06) - 30)= Abs((20 + 02 + 30 + 24) - 30)= Abs(76 - 30) = 46

F\_obj[5] = Abs((01 + 2\*04 + 3\*13 + 4\*19) - 30)= Abs((01 + 08 + 39 + 76) - 30)= Abs(124 - 30)= 94

F\_obj[6] = Abs((20 + 2\*05 + 3\*17 + 4\*01) - 30) = Abs((20 + 10 + 51 + 04) - 30) = Abs(85 - 30= 55

**Step 3. Selection**

The fittest chromosomes have higher probability to be selected for the next generation. To compute fitness probability we must compute the fitness of each chromosome. To avoid divide by zero problem, the value of F\_obj is added by 1.

Fitness[1] = 1 / (1+F\_obj[1]) = 1 / 94 = 0.0106

Fitness[2] = 1 / (1+F\_obj[2]) = 1 / 81 = 0.0123

Fitness[3] = 1 / (1+F\_obj[3]) = 1 / 84 = 0.0119

Fitness[4] = 1 / (1+F\_obj[4]) = 1 / 47 = 0.0213

Fitness[5] = 1 / (1+F\_obj[5]) = 1 / 95 = 0.0105

Fitness[6] = 1 / (1+F\_obj[6]) = 1 / 56 = 0.0179

**Step 5. Mutation**

Number of chromosomes that have mutations in a population is determined by the mutation\_rate parameter. Mutation process is done by replacing the gen at random position with a new value. The process is as follows. First we must calculate the total length of gen in the population. In this case the total length of gen is total\_gen = number\_of\_gen\_in\_Chromosome \* number of population=4\*6= 24

Mutation process is done by generating a random integer between 1 and total\_gen (1 to 24). If generated random number is smaller than mutation\_rate(ρm) variable then marked the position of gen in chromosomes. Suppose we define ρm 10%, it is expected that 10% (0.1) of total\_gen in the population that will be mutated: number of mutations = 0.1 \* 24 = 2.4 ≈2

Suppose generation of random number yield 12 and 18 then the chromosome which have mutation are Chromosome number 3 gen number 4 and Chromosome 5 gen number 2. The value of mutated gens at mutation point is replaced by random number between 0-30. Suppose generated random number are 2 and 5 then Chromosome composition after mutation are:

Chromosome[1] = [02;05;17;01]

Chromosome[2] = [10;04;13;14]

Chromosome[3] = [12;05;23;02]

Chromosome[4] = [20;04;13;14]

Chromosome[5] = [10;05;18;03]

Chromosome[6] = [20;01;10;06]

Finishing mutation process then we have one iteration or one generation of the genetic algorithm. We can now evaluate the objective function after one generation:

Chromosome[1] = [02;05;17;01]

F\_obj[1] = Abs(( 02 + 2\*05 + 3\*17 + 4\*01 ) - 30)= Abs((2 + 10 + 51 + 4 ) - 30) = Abs(67 - 30) = 37

Chromosome[2] = [10;04;13;14]

F\_obj[2] = Abs(( 10 + 2\*04 + 3\*13 + 4\*14 ) - 30)= Abs((10 + 8 + 33 + 56 ) - 30) = Abs(107 - 30)= 77

Chromosome[3] = [12;05;23;02]

F\_obj[3] = Abs(( 12 + 2\*05 + 3\*23 + 4\*02 ) - 30) = Abs((12 + 10 + 69 + 8 ) - 30)= Abs(87 - 30)= 47

Chromosome[4] = [20;04;13;14]

F\_obj[4] = Abs(( 20 + 2\*04 + 3\*13 + 4\*14 ) - 30) = Abs((20 + 8 + 39 + 56 ) - 30)= Abs(123 - 30)= 93

Chromosome[5] = [10;05;18;03]

F\_obj[5] = Abs(( 10 + 2\*05 + 3\*18 + 4\*03 ) - 30)= Abs((10 + 10 + 54 + 12 ) - 30) = Abs(86 - 30)= 56

Chromosome[6] = [20;01;10;06]

F\_obj[6] = Abs(( 20 + 2\*01 + 3\*10 + 4\*06 ) - 30) = Abs((20 + 2 + 30 + 24 ) - 30)= Abs(76 - 30) = 46

From the evaluation of new Chromosome, we can see that the objective function is decreasing, this means that we have better Chromosome or solution compared with previous Chromosome generation.These new Chromosomes will undergo the same process as the previous generation of Chromosomes such as evaluation, selection, crossover and mutation and at the end it produces new generation of Chromosome for the next iteration. This process will be repeated until a predetermined number of generations. For this example, after running 50 generations, best chromosome is obtained: Chromosome = [07; 05; 03; 01]

This means that: a = 7, b = 5, c = 3, d = 1

If we use the number in the problem equation:

a + 2b + 3c + 4d = 30

7 + (2 \* 5) + (3 \* 3) + (4 \* 1) = 30

We can see that the value of variable a, b, c and d generated by genetic algorithm can satisfy that equality.

**RESULT DESCRIPTION**

[0, 1, 0, 0][0, 1, 1, 0][0, 1, 0, 1][1, 0, 0, 0][0, 0, 1, 1][0, 0, 0, 0][0, 0, 0, 1][1, 0, 1, 0][1, 0, 0, 1][0, 1, 1, 1][0, 0, 1, 0] fitness: 64

[0, 1, 1, 1][0, 1, 0, 0][0, 0, 1, 1][0, 1, 1, 0][1, 0, 0, 1][0, 1, 0, 1][1, 0, 0, 0][0, 0, 0, 0][0, 0, 0, 1][1, 0, 1, 0][0, 0, 1, 0] fitness: 73

[0, 0, 1, 0][1, 0, 0, 1][1, 0, 1, 0][0, 1, 0, 0][1, 0, 0, 0][0, 0, 0, 0][0, 1, 1, 0][0, 1, 0, 1][0, 0, 1, 1][0, 0, 0, 1][0, 1, 1, 1] fitness: 87

[1, 0, 0, 0][1, 0, 1, 0][0, 0, 1, 1][0, 1, 0, 0][0, 0, 1, 0][0, 0, 0, 0][0, 1, 0, 1][0, 1, 1, 1][0, 1, 1, 0][1, 0, 0, 1][0, 0, 0, 1] fitness: 94

Most Fit Solutions after 10 generations.

[0, 1, 0, 0][0, 1, 1, 0][0, 1, 0, 1][1, 0, 0, 0][0, 0, 1, 1][0, 0, 0, 0][0, 0, 0, 1][1, 0, 1, 0][1, 0, 0, 1][0, 1, 1, 1][0, 0, 1, 0] fitness: 64

[0, 1, 0, 0][0, 1, 1, 1][0, 0, 1, 1][1, 0, 0, 0][1, 0, 1, 0][0, 1, 1, 0][0, 0, 1, 0][0, 0, 0, 1][1, 0, 0, 1][0, 0, 0, 0][0, 1, 0, 1] fitness: 71

[0, 1, 1, 0][0, 1, 0, 1][1, 0, 1, 0][0, 0, 1, 0][0, 1, 1, 1][1, 0, 0, 0][1, 0, 0, 1][0, 0, 0, 1][0, 0, 0, 0][0, 1, 0, 0][0, 0, 1, 1] fitness: 59

[1, 0, 0, 1][0, 0, 0, 1][0, 0, 0, 0][0, 0, 1, 0][1, 0, 0, 0][1, 0, 1, 0][0, 1, 0, 1][0, 1, 0, 0][0, 0, 1, 1][0, 1, 1, 0][0, 1, 1, 1] fitness: 61

[0, 0, 1, 0][0, 0, 0, 1][0, 0, 0, 0][0, 1, 0, 0][0, 0, 1, 1][0, 1, 1, 1][1, 0, 1, 0][0, 1, 1, 0][1, 0, 0, 0][0, 1, 0, 1][1, 0, 0, 1] fitness: 63

[1, 0, 1, 0][0, 0, 0, 1][1, 0, 0, 0][0, 0, 0, 0][1, 0, 0, 1][0, 1, 0, 1][0, 1, 0, 0][0, 1, 1, 0][0, 1, 1, 1][0, 0, 1, 1][0, 0, 1, 0] fitness: 55

[0, 0, 1, 0][0, 0, 1, 1][0, 0, 0, 1][0, 1, 1, 1][1, 0, 0, 0][0, 0, 0, 0][1, 0, 0, 1][1, 0, 1, 0][0, 1, 0, 0][0, 1, 1, 0][0, 1, 0, 1] fitness: 52

[1, 0, 0, 0][1, 0, 0, 1][0, 1, 1, 0][0, 0, 1, 1][0, 1, 1, 1][0, 1, 0, 1][0, 1, 0, 0][0, 0, 1, 0][0, 0, 0, 1][1, 0, 1, 0][0, 0, 0, 0] fitness: 53

[0, 0, 1, 1][1, 0, 0, 0][0, 1, 1, 0][0, 0, 0, 0][0, 0, 1, 0][0, 1, 0, 1][0, 1, 0, 0][0, 0, 0, 1][1, 0, 1, 0][1, 0, 0, 1][0, 1, 1, 1] fitness: 70

[1, 0, 1, 0][0, 1, 0, 0][0, 1, 0, 1][0, 1, 1, 0][0, 0, 0, 0][1, 0, 0, 0][0, 1, 1, 1][1, 0, 0, 1][0, 0, 0, 1][0, 0, 1, 0][0, 0, 1, 1] fitness: 63

[0, 0, 1, 1][0, 1, 0, 0][0, 1, 1, 0][0, 1, 0, 1][0, 0, 0, 1][1, 0, 1, 0][1, 0, 0, 1][1, 0, 0, 0][0, 0, 0, 0][0, 1, 1, 1][0, 0, 1, 0] fitness: 68

Best Tour :[0, 0, 1, 0][0, 0, 1, 1][0, 0, 0, 1][0, 1, 1, 1][1, 0, 0, 0][0, 0, 0, 0][1, 0, 0, 1][1, 0, 1, 0][0, 1, 0, 0][0, 1, 1, 0][0, 1, 0, 1] fitness: 52

Position(gene value) : index in solution

index 0 value: 5

index 1 value: 2

index 2 value: 0

index 3 value: 1

index 4 value: 8

index 5 value: 10

index 6 value: 9

index 7 value: 3

index 8 value: 4

index 9 value: 6

index 10 value: 7

0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 | - - 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 C 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 | - A 0

0 0 0 0 0 - - - - | 0

0 0 0 0 0 0 0 C 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 | - A 0

0 0 0 0 0 B - - - | 0

0 0 0 0 0 | 0 C 0 0 0

0 - - - - | 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

- | 0 0 0 0 0 | - A 0

0 | 0 0 0 B - - - | 0

0 | 0 0 0 | 0 C 0 0 0

0 I - - - | 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

K | 0 0 0 0 0 | - A 0

| | 0 0 0 B - - - | 0

| | 0 0 0 | 0 C 0 0 0

| I - - - | 0 | 0 0 0

| 0 0 0 0 0 0 | 0 0 0

| 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

K | 0 0 0 0 0 | - A 0

| | 0 0 0 B - - - | 0

| | 0 0 0 | 0 C 0 0 0

| I - - - | 0 | 0 0 0

| - - - - 0 0 | 0 0 0

J 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

K | 0 0 0 0 0 | - A 0

| | 0 0 0 B - - - | 0

| | 0 0 0 | 0 C 0 0 0

| I - - - | 0 | 0 0 0

| - - - D 0 0 | 0 0 0

J 0 0 0 | - - - - - 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

K | 0 0 0 0 0 | - A 0

| | 0 0 0 B - - - | 0

| | 0 0 0 | 0 C 0 0 0

| I - - - | 0 | 0 0 0

| - - - D 0 0 | 0 0 0

J 0 0 0 | - - - - E 0

0 0 0 0 0 0 0 | 0 | 0

0 0 0 0 0 - - - - | 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

K | 0 0 0 0 0 | - A 0

| | 0 0 0 B - - - | 0

| | 0 0 0 | 0 C 0 0 0

| I - - - | 0 | 0 0 0

| - - - D 0 0 | 0 0 0

J 0 0 0 | - - - - E 0

0 0 0 0 0 0 0 | 0 | 0

0 - - - - G - - - | 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

K | 0 0 0 0 0 | - A 0

| | 0 0 0 B - - - | 0

| | 0 0 0 | 0 C 0 0 0

| I - - - | 0 | 0 0 0

| - - - D 0 0 | 0 0 0

J 0 0 0 | - - - - E 0

0 0 0 0 0 0 0 | 0 | 0

0 - - - - G - - - | 0

0 0 0 0 0 0 0 | 0 0 0

0 0 0 0 0 0 0 F 0 0 0

0 0 0 0 0 0 0 0 0 0 0

**CONCLUSION**

Mathematical model is developed and the output of mathematical model is compared with the experimental values, suitability of the model for predicting the cutting force is also assessed.A genetic algorithm is a population-based search and optimization method that mimics the process of natural evolution. The functioning of genetic algorithm is inspired by two main concepts of natural evolution – natural selection and genetics dynamics involving different genetic operations like crossover, mutation etc. Genetic algorithms are highly applicable in immeasurable applications like search, optimization, decision making, machine learning, robotics and many more.The proposed research work focused on Standard Travelling Salesman Problem (TSP) and can be extended to benchmark functions and other optimization problems.

**REFERENCES**

https://github.com/LazoCoder/Genetic-Algorithm-for-the-Traveling-Salesman-Problem

https://en.wikipedia.org/wiki/Genetic\_algorithm

<https://en.wikipedia.org/wiki/Travelling_salesman_problem>