Program Structures and Algorithms

THE TRAVELING SALESMAN PROBLEM

FINAL PROJECT REPORT

Team 521

TEAM MEMBERS:

URVI MARU

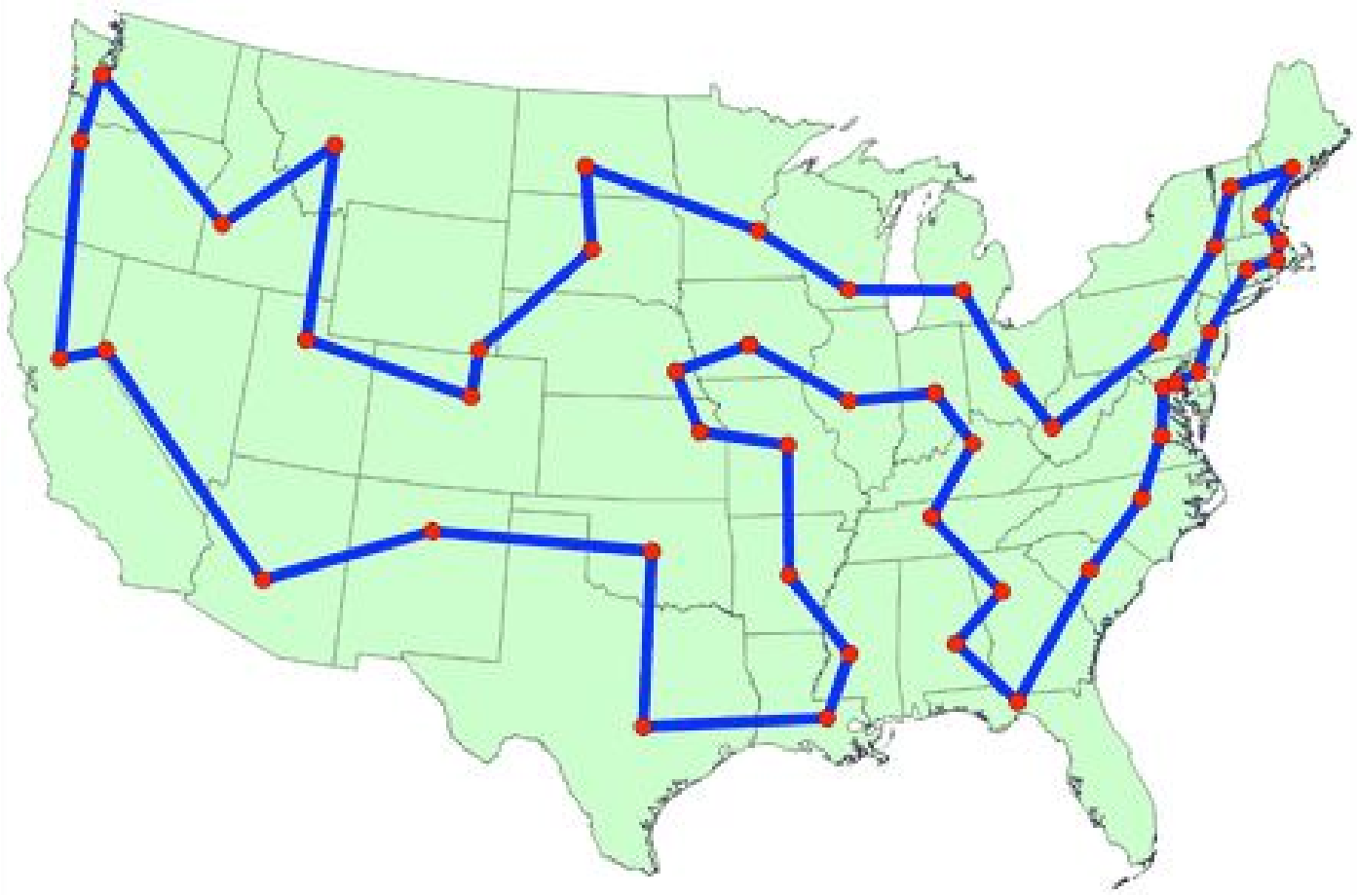
KEYUR DONGA

Under the Guidance of

-Prof Robin Hillyard

PROBLEM STATEMENT

- We have a list of cities to be visited and the pairwise distances between them.
- The Problem Statement poses the situation that given a set of cities and distances between them, we find the shortest path for a route that visits each city and returns to the original city using Genetic Algorithms.
- This Problem can be interpreted using a Graph theory where all the cities can be thought of as vertices and distances can be described as edges(weighted).



IMPLEMENTATION DESIGN

Genetic Code Design

- To apply GA for any optimization problem, one has to think of a way for encoding solutions as feasible chromosomes so that the crossovers of feasible chromosomes result in feasible chromosomes. The techniques for encoding solutions vary by problem and, involve a certain amount of art.
- For the TSP, solution is typically represented by chromosome of length as the number of nodes in the problem. Each gene of a chromosome takes a label of node such that no node can appear twice in the same chromosome.

Genetic Expression

- There are mainly two representation methods for representing tour of the TSP – adjacency representation and path representation.
- We consider the path representation for a tour, which simply lists the label of nodes (nodes being the cities in our problem).
- For example, let {1, 2, 3, 4, 5} be the labels of nodes in a 5 node instance, then a tour {1→ 3→4→ 2→ 5 →1} may be represented as (1, 3, 4, 2, 5).

Fitness Function

- We are trying to evaluate the shortest path route connecting the cities.
- The GA's are used for maximization problem. For the maximization problem, the fitness function is same as the objective function. But for minimization problem, one way of defining a 'fitness function' is F(x) = 1/ f(x), where f(x) is the objective function.

- Since TSP is a minimization problem, we consider this fitness function where f(x) calculates the total distance of the route of visiting all cities.

Crossover

- The Crossover operator speeds up the evolution by inheriting different parts of genotype from different parents.
- The need for crossover is understood from the fact that fitness is not only factor that would determine reproductive success and that sexual selection is equally important.
- For Eg: From the two routes like (1, 3, 4, 2, 5) & (5, 2, 1, 4, 3) we will generate a crossbreed of (5, 3, 4, 2, 1)
- A fixed set of random genes from one of the two chromosomes is considered and the remaining genes from the other chromosome are added and the resultant inherited children can be generated.

Mutation

- The mutation operator randomly selects a position in the chromosome and changes the corresponding allele, thereby modifying information.
- The need for mutation comes from the fact that as the less fit members of successive generations are discarded; some aspects of genetic material could be lost forever.
- By performing occasional random changes in the chromosomes, GAs ensure that new parts of the search space are reached, which crossover alone couldn't fully guarantee.
- In doing so, mutation ensures that no important features are prematurely lost, thus maintaining the diversity.
- For the TSP, the classical mutation operator does not work. For this investigation, we have considered the reciprocal exchange mutation that selects two nodes randomly and swaps them.
- For Eg: (5, 3, 4, 2, 1) mutates to (5, 3, 2, 4, 1)

Evolution

- The population is stored in an ArrayList. In the process of **crossbreeding** the population, we implement **Parallelization** using CompletableFuture and generate a tournament population that seeds a fixed size of individuals to the next generation whose genes have been shuffled.
- For each Generation, all individuals reproduce in this manner and each child experiences the **mutation operation**.
- We calculate the **fitness** for each route and sort it using a **collection sorting function**.
- We print the survivors with highest fitness scores along with the total distance to visit all cities on that route.

Control Parameters

These are the parameters that govern the GA search process. Some of them are:

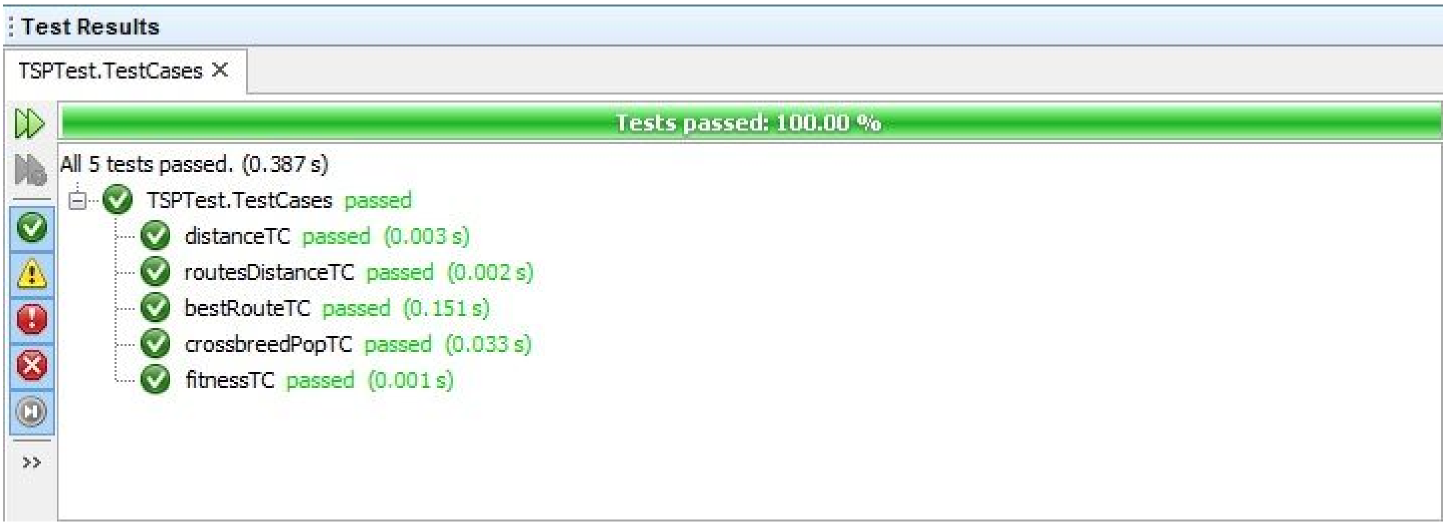
(a) Population size: - It determines how many chromosomes and thereafter, how much genetic material is available for use during the search. If there is too little, the search has no chance to adequately cover the space. If there is too much, the GA wastes time evaluating chromosome

(c) Mutation ratio: - It specifies the rate of doing index-wise mutation. In our program, we compare the mutation rate with the random generated value and perform mutation only if random value is less than mutation rate, which allows mutation to take place randomly during cross breeding.

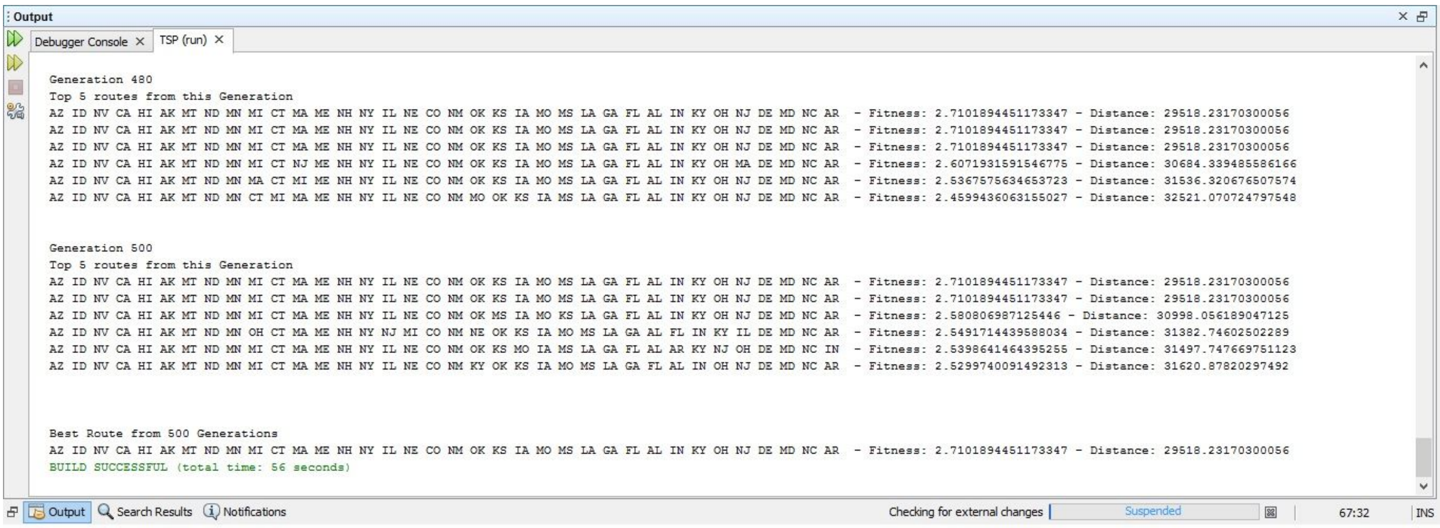
RESULTS

Our Project builds and successfully passes all the Unit Test cases. We are using print statements instead of logging output.

Following is the screenshot:



Below is the screenshot of how the Output looks like for our written Program:



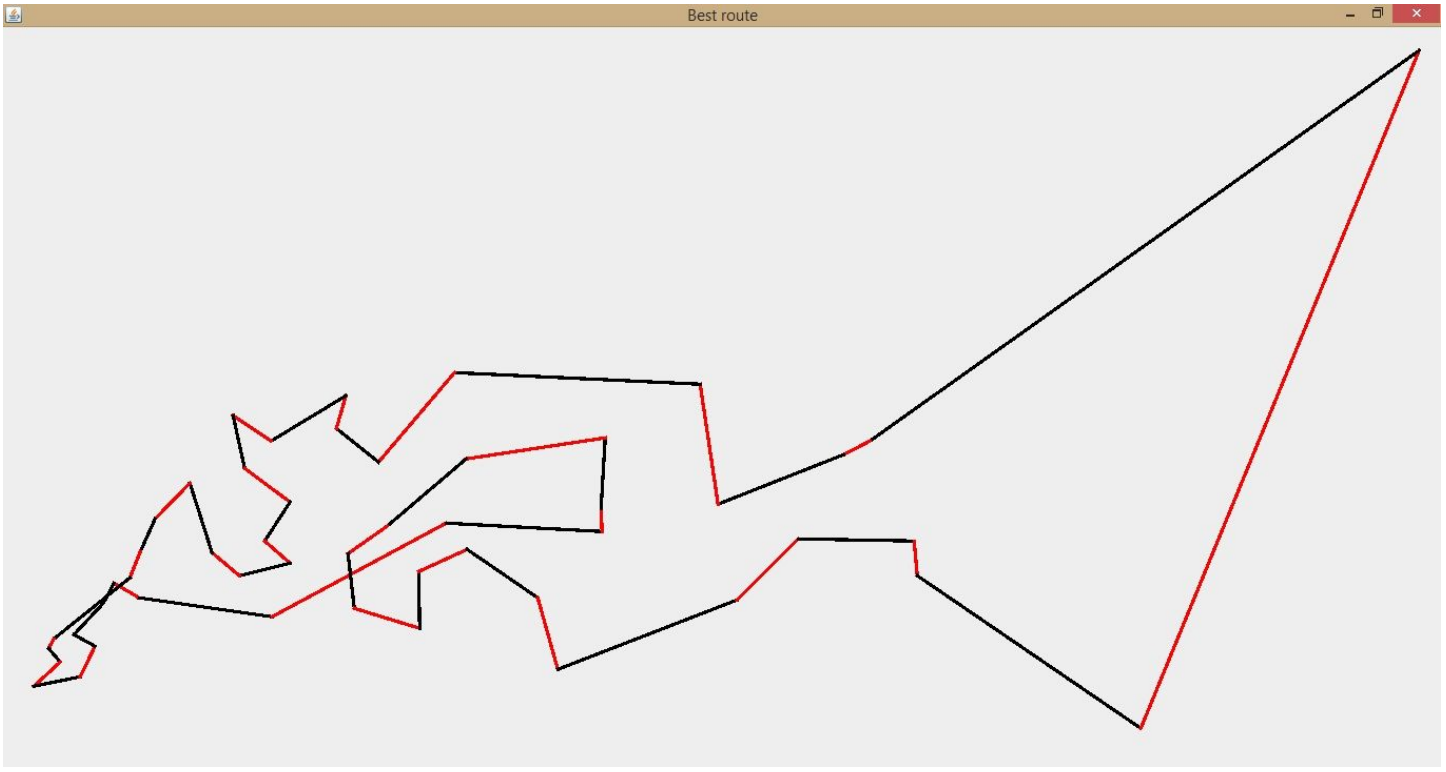
Following is the UI screenshot of the best result found:

Fitness: 2.738760154760697 , Distance: 29210.29790101869 ,

Mutation Rate - 0.125 , Generations - 2000

Initial:

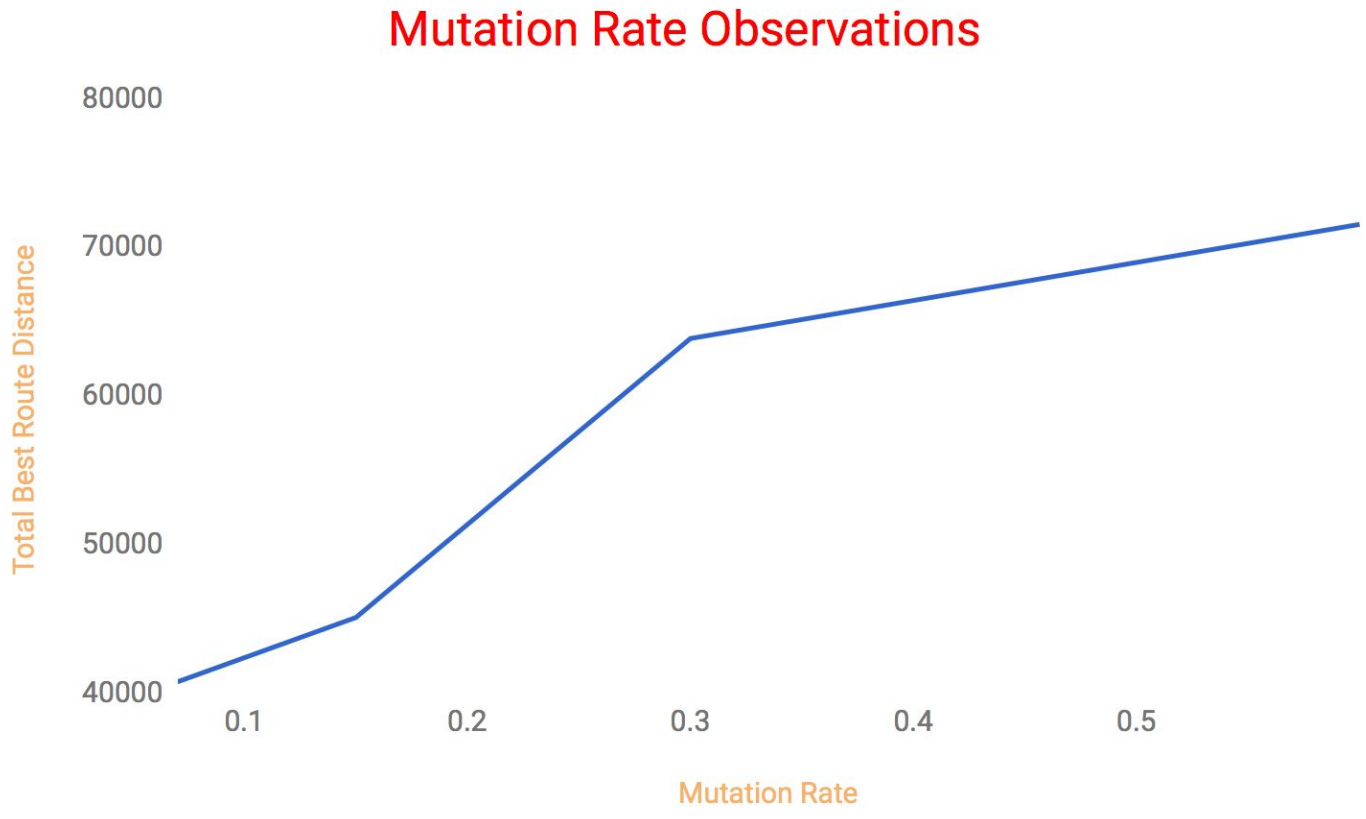
Optimal result:



Following are the various Analysis observed from our results:

1) **Mutation Rate (**For 5000 Generations**)**

Mutation Rate	Fitness	Distance
0.035	2.130359092	37552.35458
0.07	2.03874627	40734.28622
0.15	1.775219916	45064.83916
0.3	1.252987664	63847.39637
0.6	1.118387527	71531.55601



When Mutation rate is set high, distance of best route result is not optimal. With a lower Mutation rate, result converges quickly at some point to a solution close to best.

2) **Time to run 50 vertices, 25, 10, 5 and generations required to get close to optimal result**

(For Population count equal to 500)

	5	10	25	50
Trail 1				
Generations	4	10	15	15
Best Distance	10431.98942	12820.05776	28878.86085	57484.86599
Time to run	2s	3s	3s	6s
Trail 2				
Generations	250	250	250	250
Best Distance	10431.98942	12820.05776	22995.36332	42131.45047
Time to run	7s	7s	15s	43s
Trail 3				
Generations	800	800	800	800
Best Distance	10431.98942	12820.05776	22995.36332	41138.90912
Time to run	14s	44s	45s	1 min 41s
Trail 4				
Generations	1500	2500	2500	1500
Best Distance	10431.98942	12820.05776	22600.08828	34444.62617
Time to run	18s	1 min 1s	2min 19s	2 min 38 s
Trail 5				
Generations	10000	10000	10000	10000
Best Distance	10431.98942	terminated	terminated	37302.43561
Time to run	terminated			24 min 48s

CONCLUSIONS & FUTURE WORK

- In this project of ‘The Travelling Salesman Problem’, we found the shortest path to visit all nodes points (cities) without revisiting any city, using Genetic Algorithm.
- In relatively smaller size of population, the algorithm generates a consistent result quickly which is close to the optimal result. In larger size of population, the program needed more number generations to reach to the optimal route.
- With some amount of mutation, we get the best route quickly. If the mutation rate is set high or have mutation rate of ‘zero’, we do not get optimal solution to our problem.
- In our observation, Genetic Algorithm can help generate optimal result for problems with high time complexity, but it is difficult to exactly know if the obtained path is the most optimal or not.
- It’s result is not very stable. We need to run the program several times to know whether the distance is close to the best.
- In this present study, we have considered only one way of mutating and crossover operators. So, an incorporation of newer ways to compare the quality of solutions by different crossover operators to the algorithm which may exactly provide optimal results, is under our investigation.