GA Method Solving Travel Salesman Problem

Junyuan Bao 001839532

# Problem Description

A salesman decides to travel through a number of cities to sell his products, he want to meet every city only once and make the total travel distance shortest. Our goal is to use GA to find an optimal solution to this problem.

# GA Implementation Description

## Basic class Description

To solve TSP problem, I implement GA class, MaxPQ class, Chromosome class, GenerateCities class and GAMainPanel class. MaxPQ class is used when doing selection, which can select individuals who has better fitness by order. Chromosome class can generate its own DNA and calculate its own fitness once getting the distance among cities and it can clone itself to the next generation. GenerateCities class can randomly generate city’s location data as the GA input. GAMainPanel can help us see the visual result of GA. GA class store the generations and mainly focus on evolution which contains crossover and mutation functions.

## Details of GA Related Class

GA class has mainly 6 methods, constructor, init, evolve, selection, crossover, mutation, and it contains also the logging methods and setters and getters.

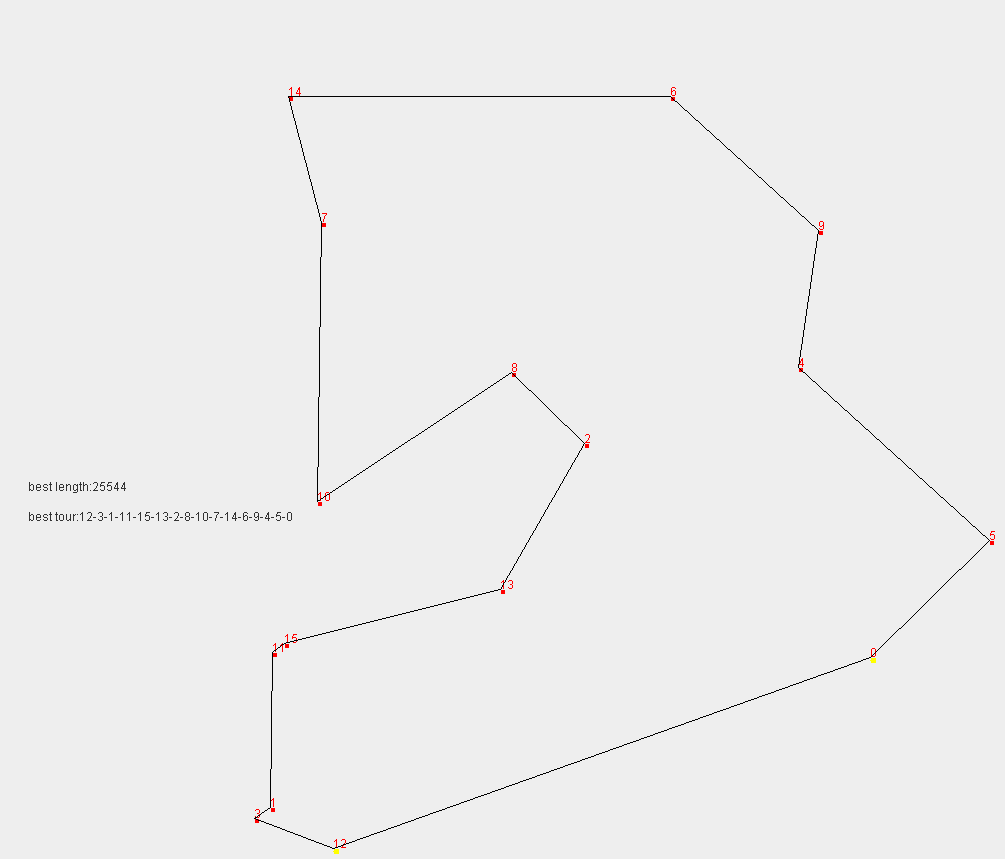
* The constructor sets the initial chromosome number, city number, max generation, crossover possibility, mutation possibility and data file input.
* Init method calculate distance among every city, and generate initial chromosome according to city number and distance calculated.
* Evolve method calculate the average fitness and best fitness of every generation, select chromosome that has better fitness then update current generation “chromosomes”. Then it select pairs of updated generation to do crossover and mutation. It also has a self-adjusted mechanism that it will enlarge the current generation set 4 times of the original size if total number of individuals exceed the boundary of the original set.
* Selection method select top 50% chromosome that has better fitness using MaxPQ, and then the result will be put into the new current generation.
* Crossover method use selected two parent chromosome. It will randomly select two cut points on DNA and then exchange the DNA between two children, there will be no repeated genes and the order of original genes in parent will be the same.
* Mutation method will change part of the selected DNA according to mutation possibility.

Chromosome Class can generate its own DNA and calculate its own fitness once getting the distance among cities and it can clone itself to the next generation.

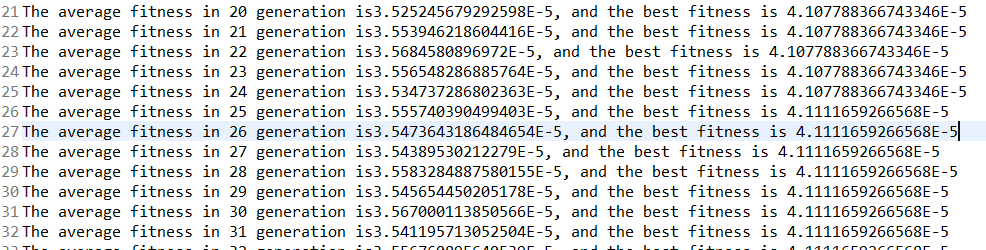
* Its genotype is a set of city numbers called “tour” which represent the actual location of each city. The phenotype is exactly the tour.
* Its fitness is calculated by fitness function using distance matrix which is calculated by GA class’ init method. The fitness value is 1/length of tour, the length is the sum of distance between neighbors of the set tour, plus distance between the begin point and the end point. The larger it is, the better result we get.
* It contains methods that can generate random “tour”
* It implements Cloneable interface which means it can be easily copied and stored from one set to another.

# Result

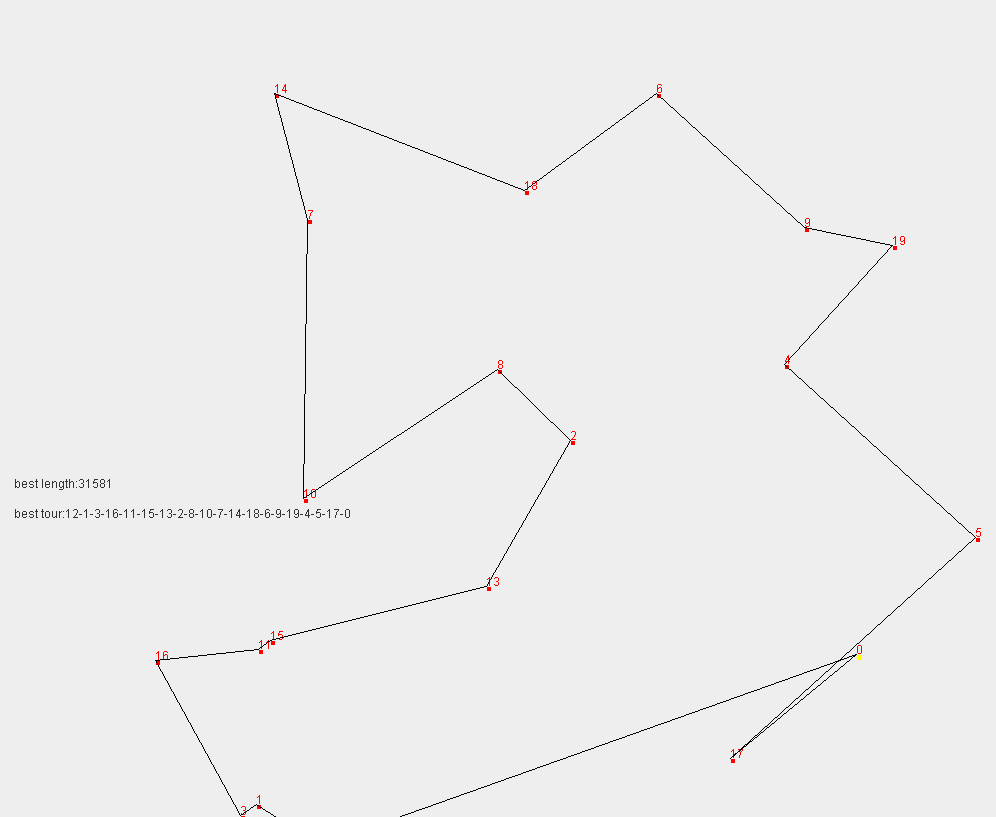
1.We set city number 16, initial chromosome number is 5000, generation number is 100, crossover possibility is 0.9 and mutation possibility is 0.1. We run the GAMainPanel class and then it gives us this final result. (see below)



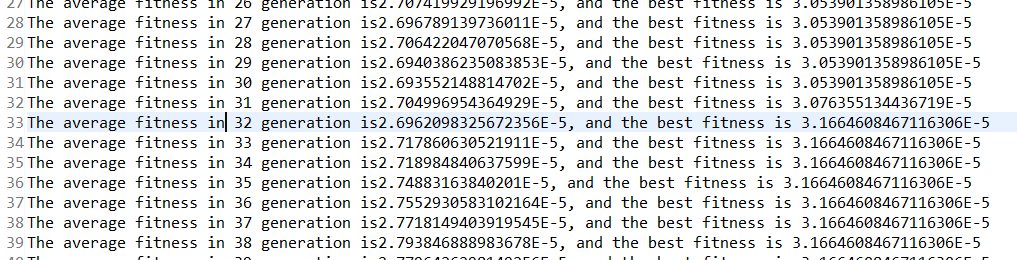
We can see from the console or the logging file that the best fitness is increasing and it converge at about 26 generation (see below graph).



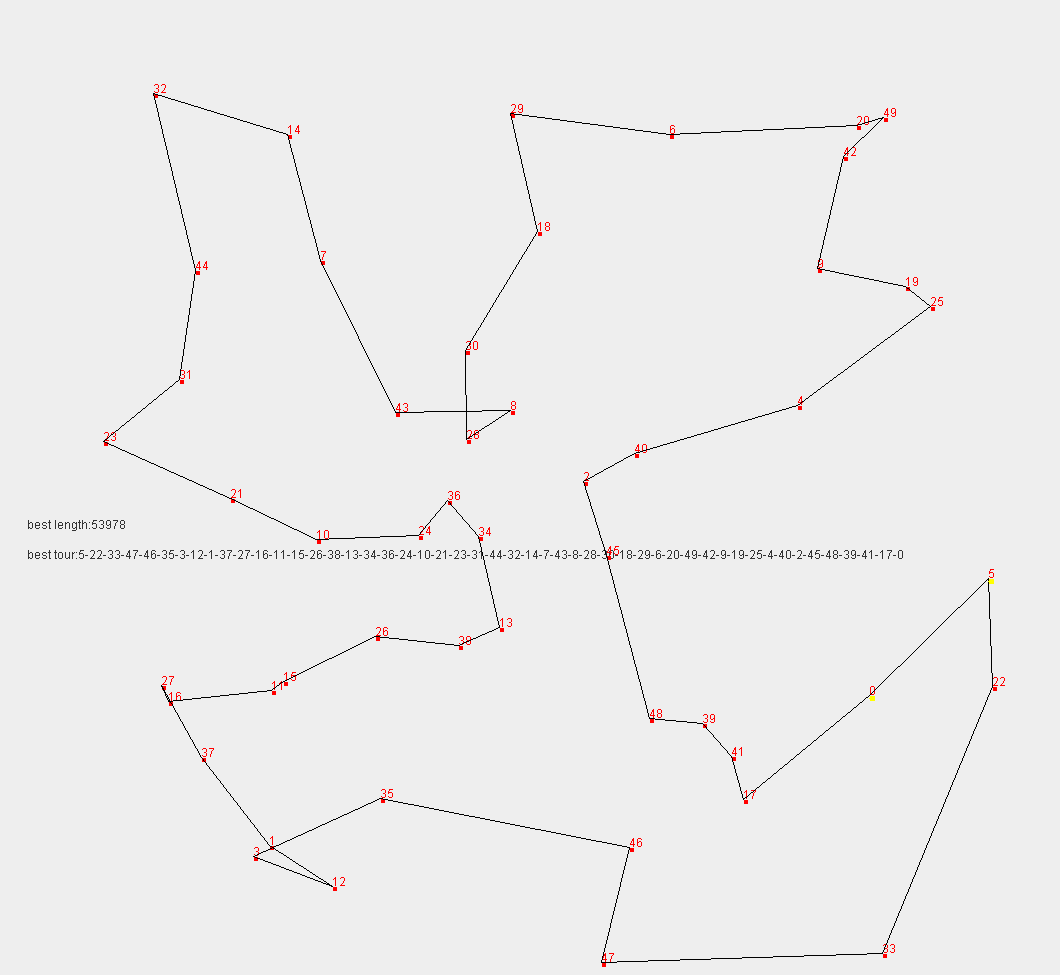
2.We set city number 20, initial chromosome number is 5000, generation number is 100, crossover possibility is 0.9 and mutation possibility is 0.1. We run the GAMainPanel class and then it gives us this final result. We can see that this is not a perfect solution, may we need more generations.



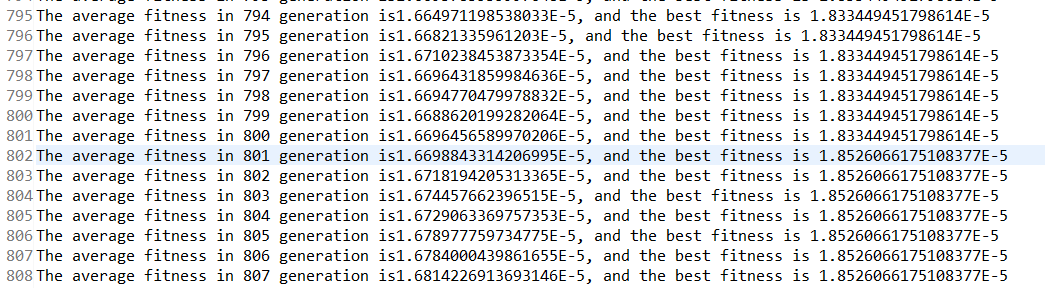
We can see from the console or the logging file that the best fitness is increasing and it converge at about 33 generation (see below graph).



3.We set city number 50, initial chromosome number is 5000, generation number is 1000, crossover possibility is 0.9 and mutation possibility is 0.1. We run the GAMainPanel class and then it gives us this final result. We can see that this is not a perfect solution, may we need more generations. And during running time we can see that the best gene will last longer and generally spread through the whole generation, it will take longer time to have a better solution when it get a nearly perfect solution.



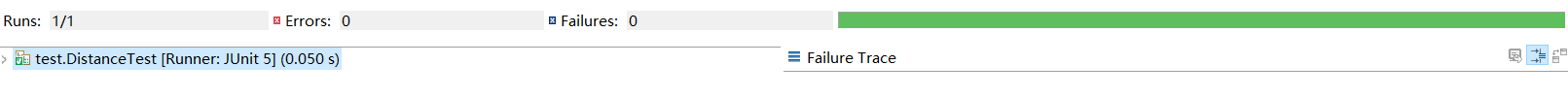
We can see that it get its nearly optimal solution when the generation number is 800, and I believe that it can give us better solution if we run more generations, but it will take a longer time.



# Unit Test

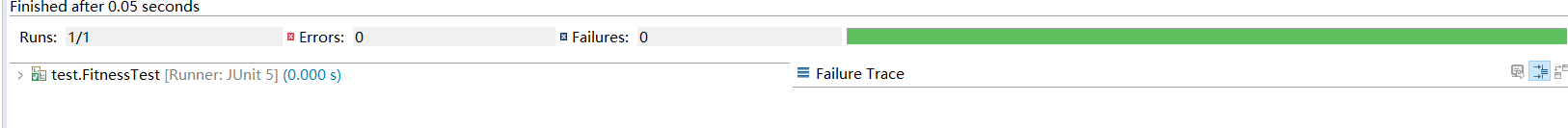
Distance calculation test

It test if GA calculate right distance between every two cities.



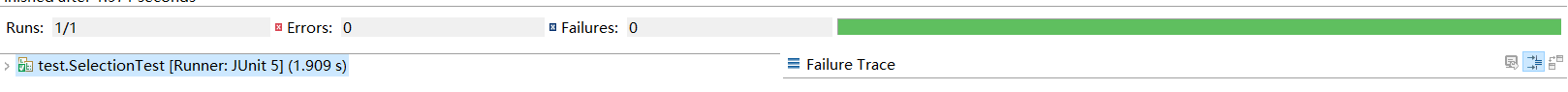
Fitness calculation test

It test whether the fitness calculation is right.



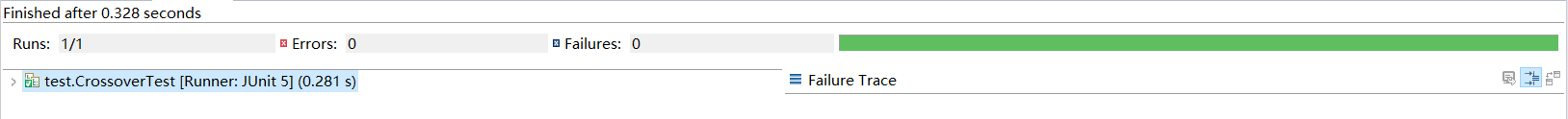
Selection test

It test whether the MaxPQ store the Chromosome according to fitness order



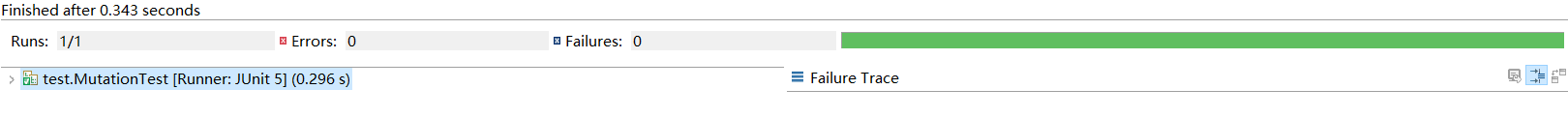
Crossover test

It test whether the crossover is valid.



Mutation test

It test whether the mutation is valid.



# Conclusion

The goal of this project is to find an optimal solution to this NP hard TSP problem. During several test, although we may sometimes get different answers, we can get solution which is nearly the best or actually the best. The crossover function can help us select good genes and mutation can help us get a better solution. As city number increased, we need more and more generations to find a better solution. Every time we got a best fitness gene, it will take longer time to find the next better solution, so mutation is important.

In conclusion, I believe that GA is a good solution to solve this NP-hard problem and it is powerful to solve TSP-like problem.