INFO 6205
Program Structures & Algorithms
Summer Full 2018
Project: Genetic Algorithms to Solve Travel Salesperson Problem

Team Number: 107
Team member: Xiangyu Liu (001498478), Shulei Gong (001403952)

1. Problem Description & Approach

Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection. In this project, our team is focused on using it to solve Travel Salesperson Problem.

The Travel Salesperson Problem is an NP-hard problem. TSP problems can be divided into two categories, one is Symmetric TSP and the other is Asymmetric TSP. All TSP issues can be described by a graph:

V={c1,c2,...,ci,...,cn},i=1,2,...,n is a collection of all cities. ci represents the i-th city, n is the number of cities;

E={(r,s):r,s∈V} is a collection of connections between all cities;

C={crs:r,s∈V} is a measure of the cost of connections between all cities (generally the distance between cities);

If crs=csr, then the TSP problem is symmetric, otherwise it is asymmetric.

A TSP problem can be expressed as: Solving the traversal graph G = (V, E, C), all nodes once and back to the starting node, so that the path cost of connecting these nodes is the lowest.

2. Implementation Details

#1 Chromosome: Each chromosome has a gene that represents one possible solution to the given problem. Each gene is a sequence of addresses for a city.

#2 Gene expression: In our case, each gene represents the distance travelman has traveled. The characteristics are expressed as the distance between the two cities that the salesman has traveled after the genetic selection.

#3 The fitness function: Each chromosome has a fitness attribute that is a measure of how close the gene is to the target. This measurement is just a simple sum of the difference of each distance in the gene to the optimal distance in the target matrix.

#4 The Selection function: We use Roulette Wheel Selection to pick. It is a playback random sampling method. The probability that each individual enters the next generation is equal to the ratio of its fitness value to the sum of individual fitness values in the entire population.

#5 The evolution driver: Evolution is the driver of this application. The driver code simply instantiates a new Population instance with a set of values for the population size, crossover ratio, elitism ratio and mutation ratio, as well as a maximum number of generations To create before exiting the simulation, in order to prevent a potential infinite execution.

#6 Track the progress of the evolution: The population is sorted by the fitness, so the best candidate from the final generation is the first element of population list.

#7 Unit tests: please see Part 4.

3. Architecture

The genetic algorithm: GA.java
The datasource includes: data.txt
The driver of the application: GeneticAlgorithmShow.java

I. Genetic Algorithm

GA.java
The implementation steps of the genetic algorithm are as follows (take the minimum of the objective function as an example).
Step 1: Initializing the t←0 evolutionary algebra counter; T is the largest evolutionary generation; randomly generating M individuals as the initial population P(t);
Step 2: Individual evaluation calculates the fitness of each individual in P(t);
Step 3: Select the operation to apply the selection operator to the group;
Step 4: crossover operation, the crossover operator is applied to the group;
Step 5: mutation operation, the mutation operator is applied to the group, and the next generation group P(t + 1) is obtained through the above operation;
Step 6: Terminate condition judgment t≦T:t← t+1 Go to step 2; t>T: Terminate the output solution.

The Population has 2 key attributes (a crossover ratio, a mutation ratio), along with a collection of Chromosome instances, up to a pre-defined population size. There is also an evolve() function that is used to "evolve" the members Of the population.

Chromosome
First, we need to determine the dye coding method. It uses different coding methods according to different problem models. We use integer coding because it is very simple. For each city, use an integer to number. For example, there are 48 cities, use 0 to 47 to identify each city, then a path is a chromosome code, the length of the chromosome is 48, such as: 0,1,2,3,4...47 is a chromosome, which means that the traveler from the city of 0 Departure, visit the city of 1, 2, ... in turn and then return to the city of No. 0; the second point of the genetic algorithm is the evaluation function. The evaluation function of TSP is very simple, that is, the total length of the path of the chromosome coding expression; In this model, the 48 numbers from 0 to 47 are all arranged to find the shortest path.

Evolution()

The evolution algorithm is simple in that it uses the various ratios during the evolution process. First, the elitism ratio is used to copy over a certain number of chromosomes unchanged to the new generation. The remaining chromosomes are then either mated with other chromosomes in the In each case, each of these chromosomes is subject to random mutation, which is based on the mutation ration mentioned earlier.

selectBestGh()

Select the most adaptive individuals in a generation of populations and copy them directly into the offspring.

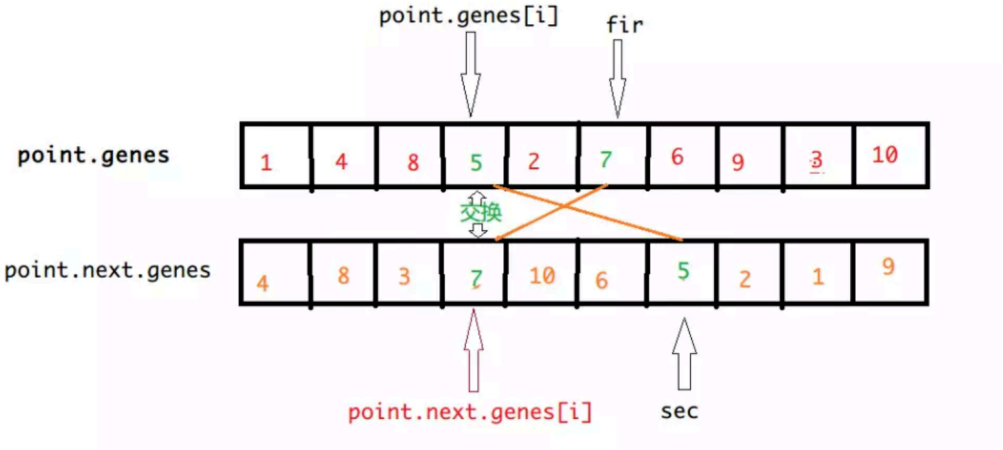
Copy the chromosome, k indicates the position of the new chromosome in the population, and kk indicates the position of the old chromosome in the population

crossover

Crossover operations are performed with a certain probability interval. Explain the steps in detail:

1) Randomly find two individuals (individual point and individual point.next).

2) In a certain probability interval. Do the following for the individual point and the individual point.next.



Mutate():

The mutate() function will randomly replace one character in the given gene.

II. Data Source

data.txt

48 city numbers, and a 48x3 matrix of distances

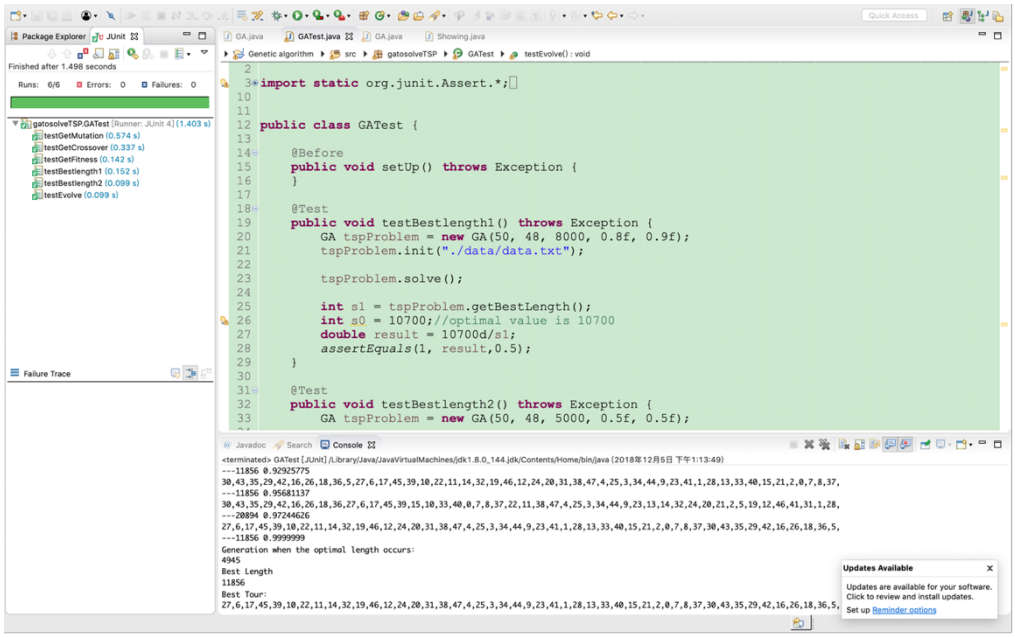
III. Driver

GeneticAlgorithmShow.java

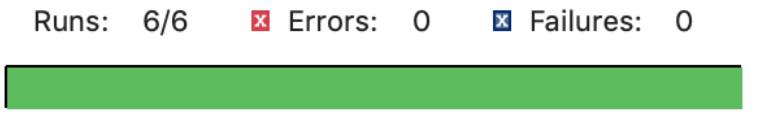
The driver code will eventually generate a path based on the information given by the data and display it on the GUI interface. We used JavaSwing to show us the best path. The green origin indicates the starting city and the ending city at the beginning. The colored lines indicate the path that passes. The red dot indicates the passing city.

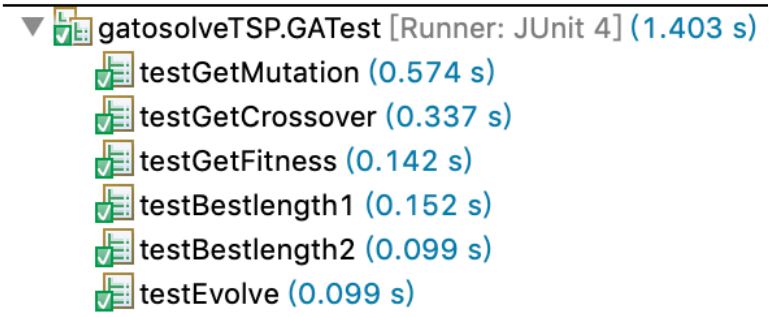
4.Unit Test

I. Test Results



II. Test Methods:





5. Observations & Conclusions

1) Data.txt

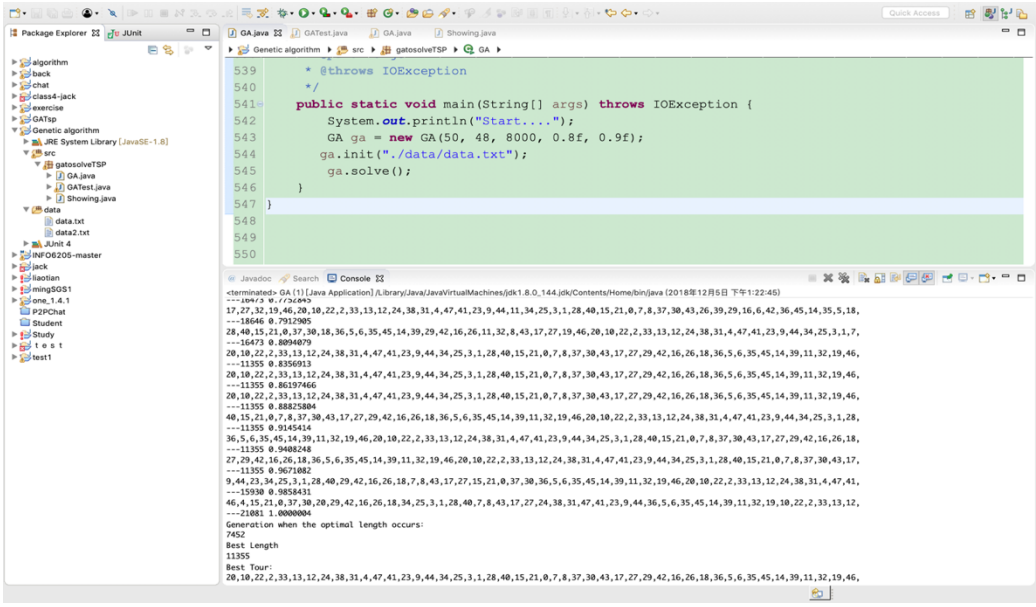
48 city numbers, and a 48x3 matrix of distances

data.txt

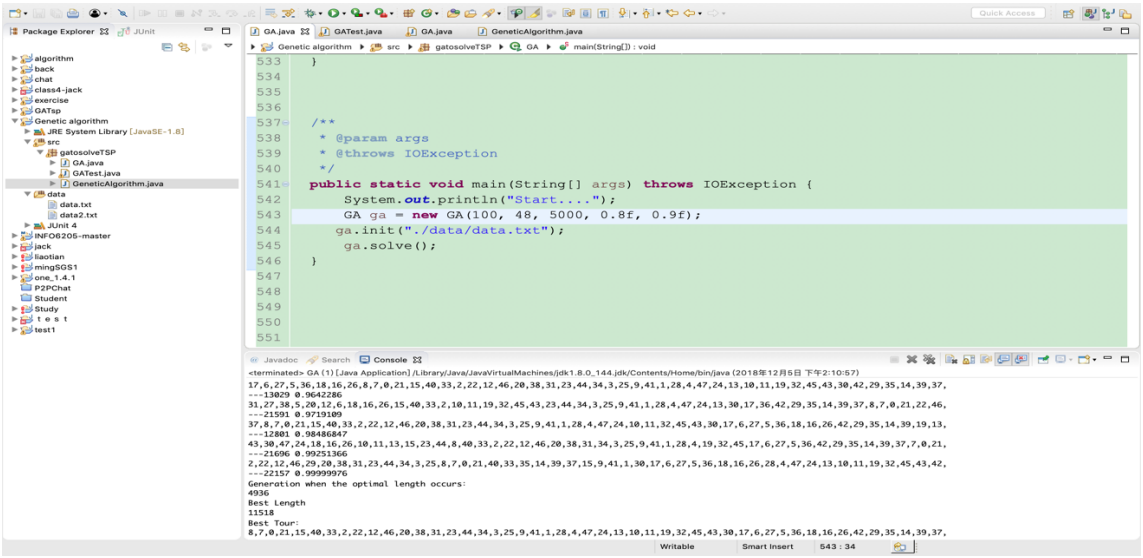
```
1 6734 1453
2 2233 10
3 5530 1424
4 401 841
5 3082 1644
6 7608 4458
7 7573 3716
8 7265 1268
9 6898 1885
10 1112 2049
11 5468 2606
12 5989 2873
13 4706 2674
14 4612 2035
15 6347 2683
16 6107 669
17 7611 5184
18 7462 3590
19 7732 4723
20 5900 3561
21 4483 3369
22 6101 1110
23 5199 2182
24 1633 2809
25 4307 2322
26 675 1006
27 7555 4819
28 7541 3981
29 3177 756
30 7352 4506
31 7545 2801
32 3245 3305
33 6426 3173
34 4608 1198
35 23 2216
36 7248 3779
37 7762 4595
38 7392 2244
39 3484 2829
40 6271 2135
41 4985 140
42 1916 1569
43 7280 4899
44 7509 3239
45 10 2676
46 6807 2993
47 5185 3258
48 3023 1942
```

2) The data is processed by genetic algorithm and the shortest distance and the best path are obtained.

```
GA ga = new GA(50, 48, 8000, 0.8f, 0.9f);
s Population size = 50
n Number of cities = 48
g max generation = 8000
c Cross rate = 0.8f
m Mutation rate = 0.9f
```

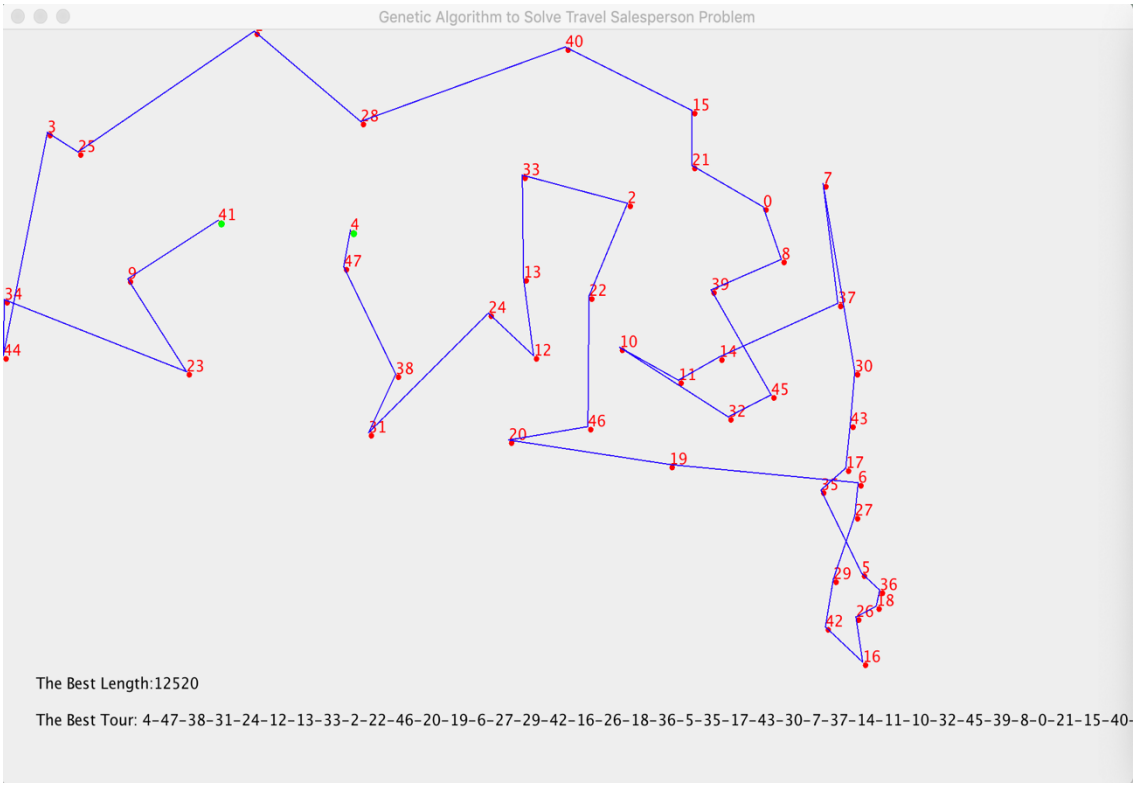



```
GA ga =
new GA(100, 48, 5000, 0.8f, 0.9f);
s Population size = 100
n Number of cities = 48
g max generation = 5000
c Cross rate = 0.8f
m Mutation rate = 0.9f
```



3) GeneticAlgorithmShow

The graphical interface shows the best path for the salesman and the passing city. The green origin indicates the starting city and the ending city at the beginning. The colored lines indicate the path that passes. The red dot indicates the passing city.



Generation when the optimal length occurs：
2671

Best Length：
12520

Best Tour：
4,47,38,31,24,12,13,33,2,22,46,20,19,6,27,29,42,16,26,18,36,5,35,17,43,30,7,37,14,11,10,32,45,39,8,0,21,15,40,28,1,25,3,44,34,23,9,41,

Conclusion

Advantages of genetic algorithms:

1. You can work directly on structural objects such as collections, sequences, matrices, trees, chains, tables, and so on.
2. The genetic algorithm can simultaneously process multiple individuals in a group, which has better global search performance and is easy to parallelize.
3. No auxiliary information is required. The fitness function is evaluated only by the fitness function, and genetic manipulation is performed on this basis.
4. It is not easy to fall into local optimum during the search process. Even if the defined fitness function is discontinuous, irregular or noisy.

Disadvantages of genetic algorithms:

1. A single genetic algorithm encoding does not fully represent the constraints of the optimization problem.
2. Efficiency is usually low.
3. It is prone to premature convergence.
4. There is no effective quantitative analysis method for the accuracy, credibility and computational complexity of the algorithm.