Ke Lin NUID: 001817984
Rong Gao NUID: 001210360
Professor: Robin Hillyard
4.15.2018

Genetic Algorithm

Traveling Salesman Problem

1. Problem Description

Suppose a travel businessman wants to visit n cities. He must choose the path to be taken. The limit of the path is that each city can only visit once, The path selection goal is that the required path distance is the minimum of all paths.

This problem can be described by a graph G = (V, E, C):

V={C1,C2....} is a set of cities. Ci represents the i-th city

E={(r,s):r,s∈V} is a set of connection between all cities

C={crs:r,s∈V} is a set of distance for the connection between all cities

So the tsp problem can be translate as:Traversing the graph G's all nodes once and make the cost for the path lowest.

2. Introduce for program

Above all, we mapping the TSP into our java program with a graph and

Faced with the TSP problem, I use the integer coding, because it is very simple, for each city with an integer number, for example, 20 cities, with 0 to 19 to identify each city, then a path is a chromosome encoding, The chromosome length is 20, such as: 0, 1, 2, 3, 4... 20 is a chromosome, which means that the traveler starts from city No. 0, and then visits city No. 1, No. 2, No. 20 again. Back to City 0.

The second key point of the genetic algorithm is the evaluation function. The evaluation function of the TSP is very simple, which is the total length of the path expressed by the chromosome encoding. In fact, in this model, the 20 numbers from 0 to 19 are completely arranged. And our goal is finding the shortest path (think 20 numbers in full order, then compare...). With a clear understanding of these, we can program according to the above genetic algorithm framework.

The Description

The Components	The Description	The function
1. The genetic code and a random generator/mutator of such codes;	Genetic code is the city number; in our program, every chromosome have 20 genetic code, and genetic code have 20 types. We used a txt file to describe the location of this city and initialize the genetic code by function initGroup() and function init()	Generator:init(); initGroup()
2.Gene expression	At first, we calculate the distance matrix by using the xy coordinate of cities. In the chromosome, the sequence of city numbers represents the trait, we use the trait to calculate the length of trait which is the gene expression.	Gene expression: the total length of trait.

3.Fitness function	We calculate the length, and use length calculate the fitness and p[], which will be used in roulette wheel selection to decide whether their gene will be copied to next population.	evaluate();
		countRate();
		copyGh();
		select();

4.a sort function to order the organisms by their fitness function;	We use parameter fitness[] and Pi[] to decide their selected possibility so we do not have the sort function	

5.An evolution mechanism	In our program, we have two evolution mechanism:crossover and mutation	evolution();
		OXCross();
		OnCVariation();

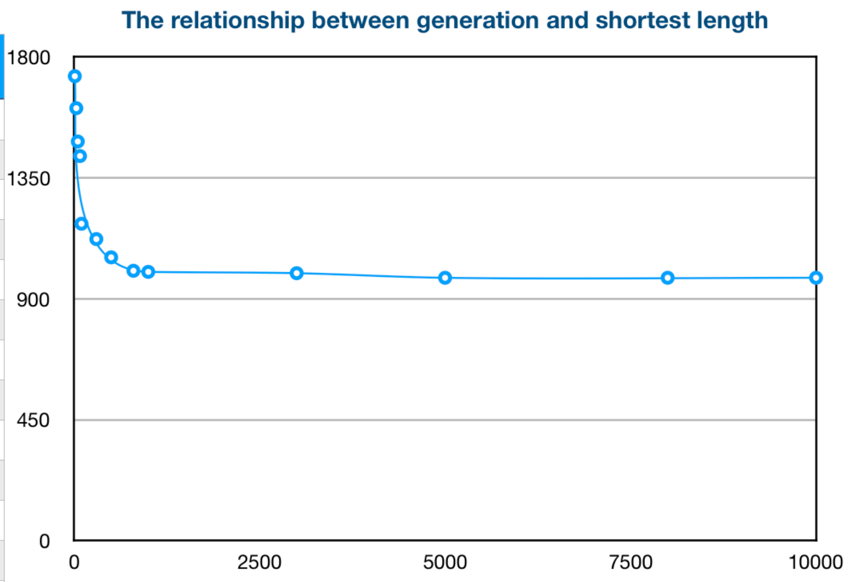
6.A logging function to keep track of the progress of the evolution	We use solve() function run the whole evolution process, and print the data to display the track of this progress	solve()
7.A set of unit tests		Class GeneticAlgorithmTest()
8.(optional) A user interface	Display the trait which have the shortest distance	Class MyPanel()
Extra1. Elitist	We use elitist selection strategy to make sure the best one in the old population can be copied to the next population. In this way, we accelerate the speed of evolution	selectBestGh()
Extra2. Expected Result	We use getexpected() function to simulate the expected result for comparative convenience	getexpected()

3. Raw Data

3.1 Relationship between generation times and shortest length

The length's evolution

Generation times	The shortest length
10	1728
30	1609
50	1485
80	1431
100	1179
300	1122
500	1054
800	1004
1000	1000
3000	995
5000	978
8000	978
10000	978



* population size = 20; chromosome length = 20; crossover possibility = 0.8; mutation possibility = 0.05

4.Algorithm optimization

4.1.Elitist selection

This strategy is known as elitist selection and guarantees that the solution quality obtained by the GA will not decrease from one generation to the next. Which can make sure our algorithm is global convergence.

4.2. Different crossover operator

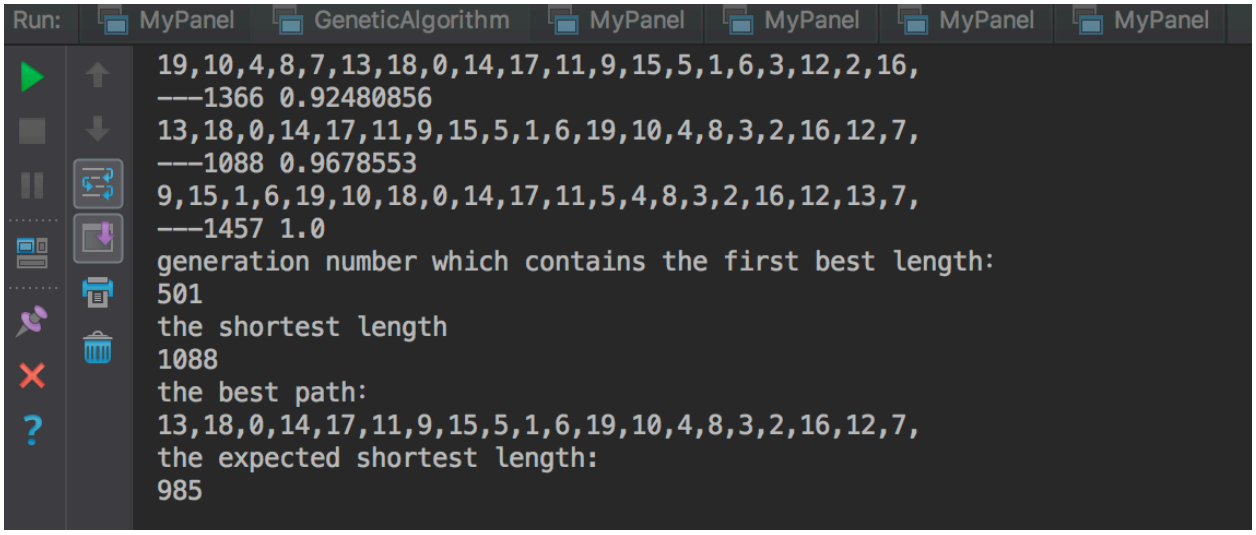
At first, we used the simple order crossover operator, and then, we tried a different crossover operator: crossing the same chromosome to produce different chromosomes which can improve the algorithm's performance.

5.Conclusion

In fact, there are many deficiencies in the basic genetic algorithm. For example, it is easy to fall into local convergence, and the global search ability is not strong enough. However, there are many places where the basic genetic algorithm can be improved, such as the design of the crossover operator and the design of the mutation operator, selection strategies, and so on. About genetic algorithms, I personally think as an intelligent heuristic search algorithm, it is easier to understand than other ordinary algorithms and easier to used in our reality life.

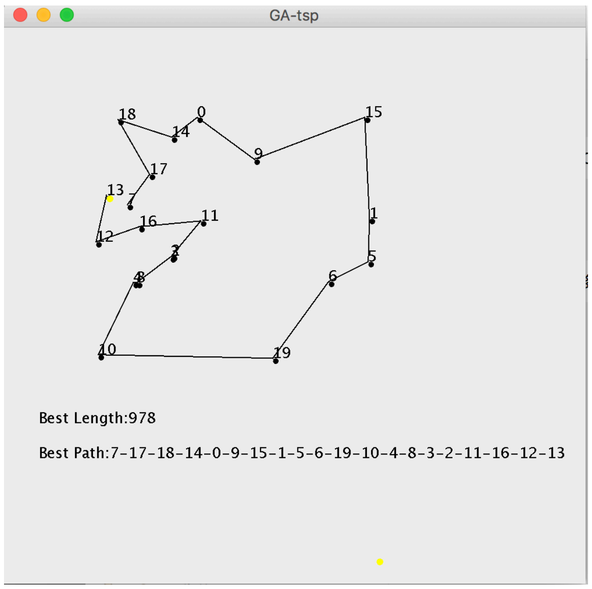
6.Screenshot for test and main function

6.1 Main function



```
19,10,4,8,7,13,18,0,14,17,11,9,15,5,1,6,3,12,2,16,
---1366 0.92480856
13,18,0,14,17,11,9,15,5,1,6,19,10,4,8,3,2,16,12,7,
---1088 0.9678553
9,15,1,6,19,10,18,0,14,17,11,5,4,8,3,2,16,12,13,7,
---1457 1.0
generation number which contains the first best length:
501
the shortest length
1088
the best path:
13,18,0,14,17,11,9,15,5,1,6,19,10,4,8,3,2,16,12,7,
the expected shortest length:
985
```

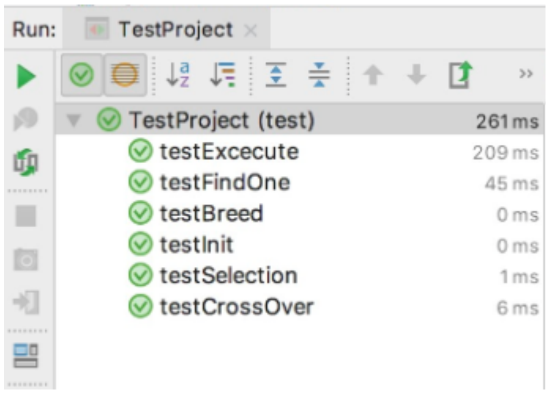
6.2 Interface GUI



Best Length:978

Best Path:7-17-18-14-0-9-15-1-5-6-19-10-4-8-3-2-11-16-12-13

6.3 Test



```
TestProject (test)    261 ms
  testExcecute        209 ms
  testFindOne          45 ms
  testBreed             0 ms
  testInit              0 ms
  testSelection         1 ms
  testCrossOver         6 ms
```