Program Structure & Algorithms

Genetic Algorithm Problem in Travel Salesman Problem

Xinying Shi & Wendi Yu
Group 531
Spring 2018, Section 5

INFO6205 Final Project Report

Problem Description:

The travel salesman problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"

Findings :

In this program, we are trying to use genetic algorithm to solve the Traveling Salesman Problem. We use "A-Z" to define the cities series.

- The city index is the gene in this solution, and the number of genes is 26. from(0-25)
- The order [A, B, C, D] means the path result is A -> B -> C -> D, each trait includes city index from 1 to 48. 26 traits form a group.
- The fitness of each trait is the sum calculated by the distance.
- We will find the traits to achieve the function 'crossover' & 'mutate'
- And we will print all the current best solution of TSP

Items Describe :

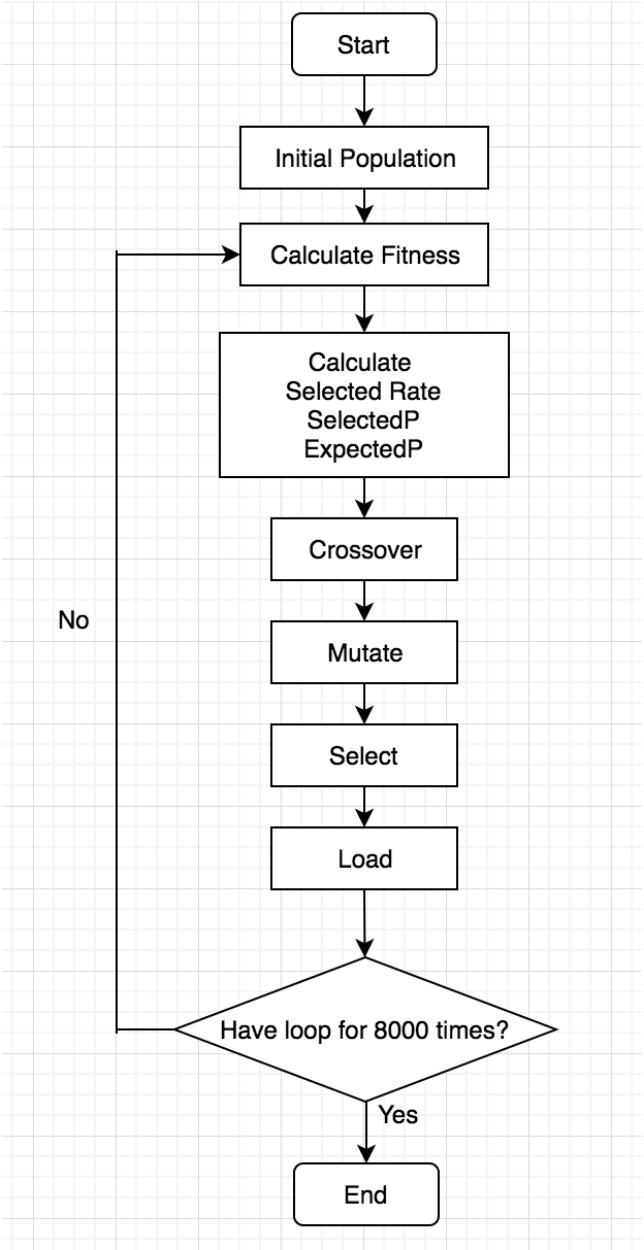
1. Genetic Code & Phenotype (gene expression):

Genetic code is represented as an letters array. Every individual has various of genes. They are random generated by the array of the letters from class "Tsp".

2. Fitness:

The fitness function calculates the sum of a city's distance value to other cities. Larger fitness means the longer route. And the "CalSelectP" will use fitness to calculate the select rate. Which will decide to chose or not. The bigger value will have less possible to be chosen here.

3. Workflow



Results + Conclusions :

```
public int cityNum=cityName.length;      //城市个数 The n
public int population = 1000;              //种群数量 T
public int generation = 8000;            //迭代次数 The
public double pxover = 0.5;             //交叉概率 The ra
public double mutate_Rate = 0.0015;       //变异概率 The
public long[][] distance = new long[cityNum][cityNum];/
public int range = 2000;                 //用于判断何时停止
```

```
public int cityNum=cityName.length;      //城市个数 The nu
public int population = 1000;              //种群数量 Th
public int generation = 8000;            //迭代次数 The r
public double pxover = 0.5;             //交叉概率 The rat
public double mutate_Rate = 0.015;        //变异概率 The r
public long[][] distance = new long[cityNum][cityNum];//
public int range = 2000;                 //用于判断何时停止的
```

```
----------  The number of the Generation : 7984
The best solution: 1163
----------  The number of the Generation : 7985
The best solution: 1163
----------  The number of the Generation : 7986
The best solution: 1163
----------  The number of the Generation : 7987
The best solution: 1163
----------  The number of the Generation : 7988
The best solution: 1163
----------  The number of the Generation : 7989
The best solution: 1174
----------  The number of the Generation : 7990
The best solution: 1174
----------  The number of the Generation : 7991
The best solution: 1174
----------  The number of the Generation : 7992
The best solution: 1174
----------  The number of the Generation : 7993
The best solution: 1174
----------  The number of the Generation : 7994
The best solution: 1174
----------  The number of the Generation : 7995
The best solution: 1174
----------  The number of the Generation : 7996
The best solution: 1174
----------  The number of the Generation : 7997
The best solution: 1174
----------  The number of the Generation : 7998
The best solution: 1174
----------  The number of the Generation : 7999
The best solution: 1174
----------  The number of the Generation : 8000
The best solution: 1174

The best solution sequence is:
W E Z N M X D T I Y F P B Q J S A O C U H R K V G L

The cost of the time is: 5.777s
```

```
The best solution: 1218
----------  The number of the Generation : 7984
The best solution: 1218
----------  The number of the Generation : 7985
The best solution: 1218
----------  The number of the Generation : 7986
The best solution: 1218
----------  The number of the Generation : 7987
The best solution: 1218
----------  The number of the Generation : 7988
The best solution: 1218
----------  The number of the Generation : 7989
The best solution: 1218
----------  The number of the Generation : 7990
The best solution: 1218
----------  The number of the Generation : 7991
The best solution: 1214
----------  The number of the Generation : 7992
The best solution: 1214
----------  The number of the Generation : 7993
The best solution: 1214
----------  The number of the Generation : 7994
The best solution: 1218
----------  The number of the Generation : 7995
The best solution: 1218
----------  The number of the Generation : 7996
The best solution: 1218
----------  The number of the Generation : 7997
The best solution: 1218
----------  The number of the Generation : 7998
The best solution: 1218
----------  The number of the Generation : 7999
The best solution: 1213
----------  The number of the Generation : 8000
The best solution: 1218

The best solution sequence is:
O Y F Q I Z E U G N H W J V M B S K P X D L R C T A

The cost of the time is: 7.711s
```

- When we increase the rate of “mutate_Rate”, which means the possible of mutate, we found that the result of this algorithm become more unstable, and the lower rate of “mutate_Rate” has the better performance.

- When we increase the rate of “pxover” from 0.5 to 1, which means the rate of the crossover. We found that it seems like it will decrease the time to find the best solution and cause more intense evolution, however it will also damage the “almost best” value and make the algorithm more unstable. We thought that it might because this situation has lower possible to fall into local optimum and it will keep seeking the global optimum.

- Also, more number of cities will impact the result.
- Thus, we come to the conclusion that genetic algorithm does not set specific change rules, but uses the rate of change and the optimal direction to highly simulate natural gene evolution. And it could hardly give us a specific quantitative relationship when face the irregular problem.

Evidence :