Report of Final Project

Problem:

In this project, we devote ourselves to solving *Traveling Salesman Probl*em by using *Genetic Algorithm*. Specifically, a delivery man need to deliver all of the packages to different places in the city. He has to determine the initial position, drives to each receiver’s house and then goes back to the starting point. Therefore, our team wants to help him look for the shortest route for saving time and energy.

Implementation Design:

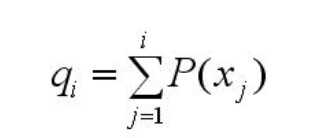
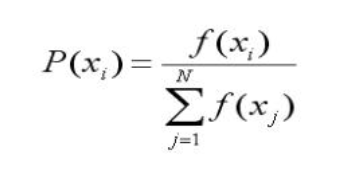
1. Population

We have used the *java.util.Collections.shuffle()* to rearrange the order of all places that the delivery man has to visit. In our project, a different order means a different route. So, we have used the *ArrayList* to store all different 50 routes.

1. Selection

Every route has its own distance. According to the distance, we have count the fitness of each route(f(xi) = 10.0 / distance(xi)).

During the process of section, we have used the roulette wheel selection to choose the better routes from all routes. The advantage of roulette wheel selection is that candidate routes with a higher fitness will be less likely to be eliminated. To be specific, every route has its own select probability(P(xi)) and the cumulative probability(qi). We have used the Math.random() to produce a random number from 0 to 1. When the random number is less than the cumulative probability of this route, the route can be selected.



After roulette wheel selection, we will get the better routes for next process.

1. Crossover
2. We have got the 200 pairs of parents randomly from the last population;
3. Using Math.random() to produce a random number between 0 (inclusive) and 1 (exclusive);
4. If the random number is less than the crossover rate(0.7), this parent need to crossover their genes and generate two different child. Put these two child in a new population.

Parent 1: 6 4 2 3 1 8 5 7 9 0

Parent 2: 3 0 6 7 8 9 4 2 1 5

Child 1: 0 6 7 3 1 8 5 9 4 2

Child 2: 6 4 2 3 8 1 5 7 9 0

1. If the random number is greater than the crossover rate (0.7), this parent does not need to crossover their genes. Put this parent into the new population.
2. Sorting

After performing crossover, we can get a new population of length 400. And then we use priority queue to sort these 400 routes and get the best 200 routes.

1. Mutation

We firstly loop through every home of the route, at the same time generate a random value to check if it is less than Mutation Rate (0.015), if it is, swap this home with any other randomly chosen home in the route, otherwise, just skip this home .

Before mutation: 0 1 2 3 4 5 6 7 8 9

After mutation: 0 7 2 5 8 3 6 1 4 9

1. User interface

Results:

Our team has used 48 vertices and randomly produce an initial population of length 200. After sorting the fitness of each route, we can get the initial shortest distance is about 126949. In this project, we have done experiment with 1000 generations. It means that we let the initial population evolve 1000 times. Finally, we can draw the conclusion that the distance of the shortest route is about 35537 in about 700th generation.

The above graph shows that the more routes in one population, the greater fitness of shortest route. It is because that if there are a lot of routes in the population more routes will be likely to cross over and mutate and since generate more possibilities and find the best. In this way, we can easily get the better solution quicker.

Also, after several times experiments we have found that the best distance in every population become relatively stable. We think it might be dependent on the number of route in every population. If the

If the mutationRate is too big, there is greater possibility that the good result will be replace, so we will set mutaionRate a relative small value(0.015).

