

Vehicle Routing Problem with GA

Team 502

Zhenwei Yu 001448385 Jialin Chen 001466077

Table of Contents

Problem & Concepts

1.1 Problem Background

1.2 Core Concepts

- 1.1.1 Genotype
- 1.1.2 Expression
- 1.1.3 Phenotype
- 1.1.4 Problem
- 1.1.5 Fitness

Implementation

2.1 Encode & Decode

- 2.1.1 Encode
- 2.1.2 Decode

2.2 Evolution

- 2.2.1 Selection
- 2.2.2 Crossover
- 2.2.3 Mutation

Results & Conclusion

3.1 Results

- 3.1.1 Unit Test

3.2 Conclusions

- 3.2.1 Output
- 3.2.2 Conclusion

Problem & Concepts

1.1 Problem Background

The VRP concerns the service of a delivery company.

How things are delivered from one or more **depots** which has a given set of home **vehicles** and operated by a set of drivers who can move on a given **road network** to a set of **customers**.

It asks for a determination of a set of routes, S , (*one route for each vehicle that must start and finish at its own depot*) such that all customers' requirements and operational constraints are satisfied and the global transportation cost is minimized.

1.1 Problem Background

The common objectives are for a VRP are:

1. **Minimize the global transportation cost based on the global distance travelled as well as the fixed costs associated with the used vehicles and drivers**
2. **Minimize the number of vehicles needed to serve all customers**
3. Least variation in travel time and vehicle load
4. Minimize penalties for low quality service

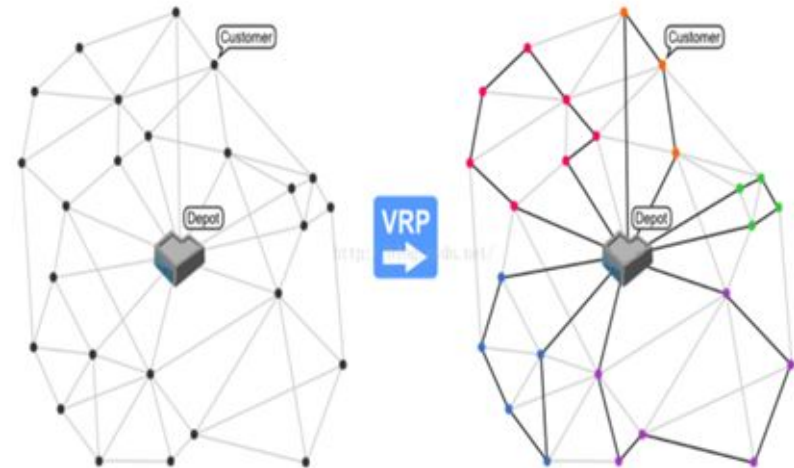
We will consider the first two objectives in our project scenario.

1.1 Problem Background

The road network can be described using a **graph** where the arcs are roads and vertices are junctions between them. The arcs are undirected. Each arc has an associated cost which is generally its length.

Settings for the problem:

- 1 Depot
- 20 Customers (clients)
- 5 Vehicles (at most)
- Cost as the total travel distance of the vehicles used



1.2 Core Concepts

1.2.1 Genotype

Under this problem scenario, a genotype is a set of genes which ranges from 1 to 20 representing the location picked after the previous one. Since there are 20 clients, the length of genotype is 20.

2	14	6	12	11	8	1	5	8	10	14	19	13	9	17	20	3	4	7	6
---	----	---	----	----	---	---	---	---	----	----	----	----	---	----	----	---	---	---	---

1.2 Core Concepts

1.2.2 Expression

The expression is the order of clients serviced by the corresponding vehicles.

1.2.3 Phenotype

In this case, the phenotype represents a possible route solution from the departure to the destination given number of vehicles. For each vehicle, it will start from the depot and back to the depot.

1.2 Core Concepts

1.2.4 Problem

We are trying to find the route which has the lowest cost of transportation, regarding the lowest distance and fewest number of vehicles.

1.2.5 Fitness

The fitness is the total travel distance of the vehicles visiting all client locations. In addition, if the number of the vehicles exceeds the requirement, the punishment factor will be added to the evaluation. In the last, we compress it as $10 / \text{cost}$.

2.1 Encode & Decode

2.1.1 Encode

We regard a client location as a gene, and basically encode every client locations into a integer from 1 to 20. For example, as the genotype showed below, the order of a possible route is: For Vehicle 1, its route will be depot \rightarrow client 2 \rightarrow client 14 \rightarrow client 6 \rightarrow client 12 \rightarrow depot, and then for Vehicle 2, its route will be depot \rightarrow client 11 \rightarrow client 8 \rightarrow client 1 \rightarrow client 5 \rightarrow client 8 \rightarrow client 10 \rightarrow depot, ... and so on for Vehicle 3.

2	14	6	12	11	8	1	5	8	10	14	19	13	9	17	20	3	4	7	6
---	----	---	----	----	---	---	---	---	----	----	----	----	---	----	----	---	---	---	---

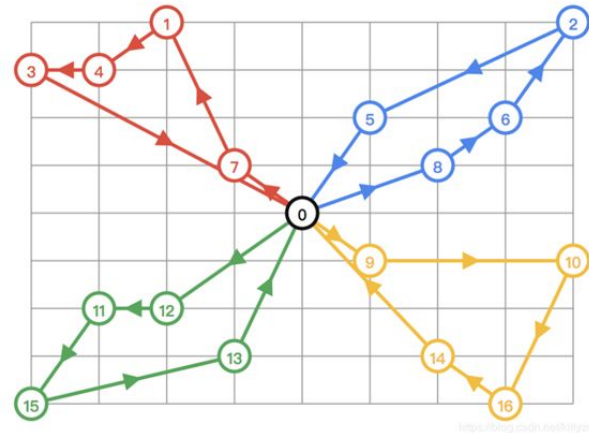
Implementation

2.1 Encode & Decode

2.1.2 Decode

Given a genotype, we interpret it as the possible route solution as show below:

- **Vehicle 1:** red route
- **Vehicle 2:** blue route
- **Vehicle 3:** yellow route
- **Vehicle 4:** green route



2.2 Evolution

2.2.1 Selection

We select the fittest individuals (**the highest fitness by definition**) in each generation and let them pass their genes to the next generation.

With the **CountRate()** method, **copyChromosome()** method and **calculateFitness()** method, we are able to select two pairs of individuals based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

2.2 Evolution

2.2.2 Crossover

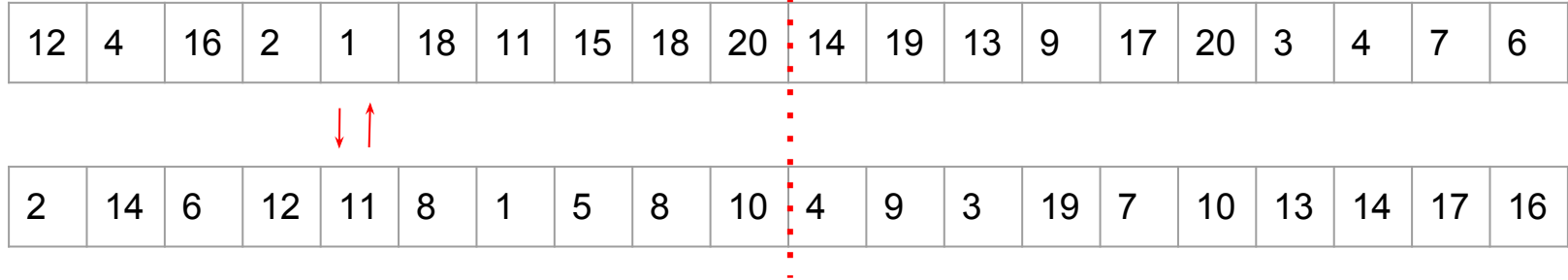
For each pair of parents to be mated, a **crossover point** is chosen at random from within the genes.

2	14	6	12	11	8	1	5	8	10	14	19	13	9	17	20	3	4	7	6
12	4	16	2	1	18	11	15	18	20	4	9	3	19	7	10	13	14	17	16

2.2 Evolution

2.2.2 Crossover

Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

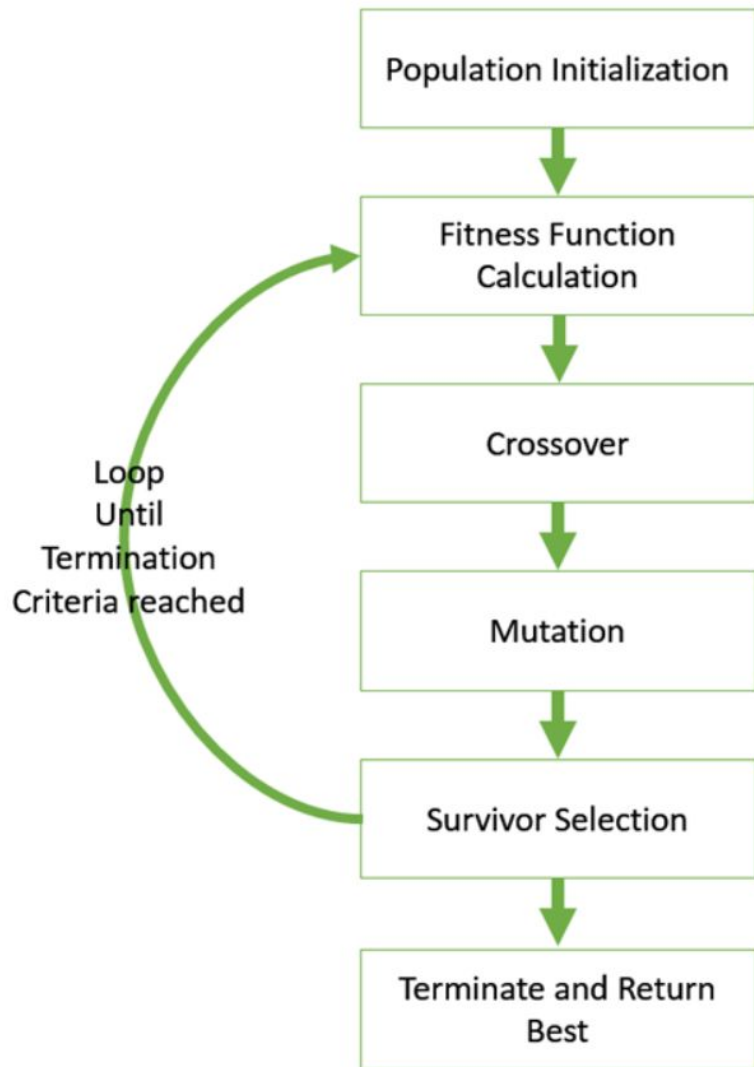


2.2 Evolution

2.2.3 Mutation

In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability.

2	14	6	12	11	8	1	5	8	10	14	19	13	9	17	20	3	4	7	6
				↓			↓												
2	14	6	12	1	8	11	5	8	10	4	9	3	19	7	10	13	14	17	16



The basic structure of a GA is as follows:

We start with an initial population (which may be generated at random or seeded by other heuristics), select parents from this population for mating. Apply crossover and mutation operators on the parents to generate new offsprings. And finally these off-springs replace the existing individuals in the population and the process repeats. In this way genetic algorithms actually try to mimic the human evolution to some extent.

Results & Conclusions

3.1 Results

3.1.1 Unit test

We did 6 Unit test to test our function correctness and performance:

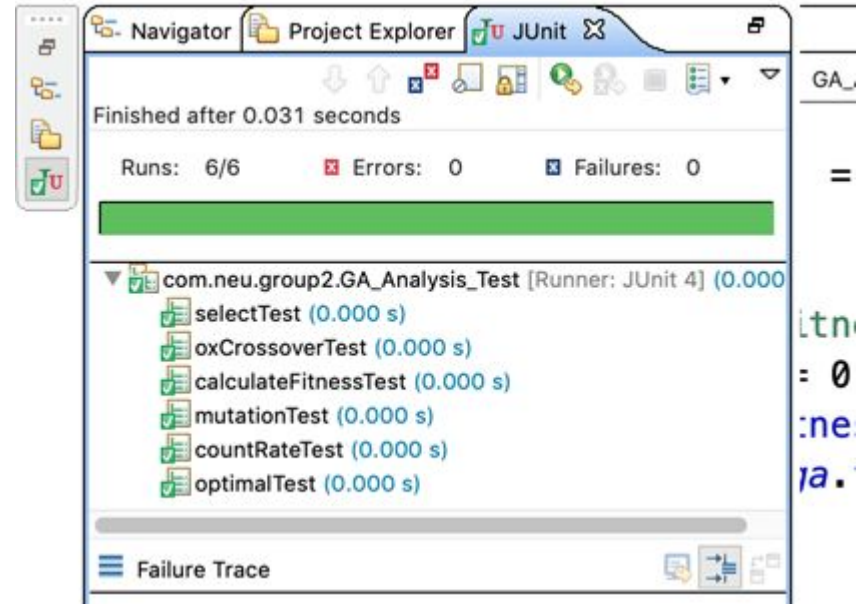
- OptimalTest
To make sure the behavior of next generation is better than super generation based on our fitness.
- Select Test
To make sure our select is based on Roulette Wheel Selection.
- OxCrossTest
To make sure our chromosome after cross should be unique.

3.1 Results

3.1.1 Unit test

Unit tests (Continued):

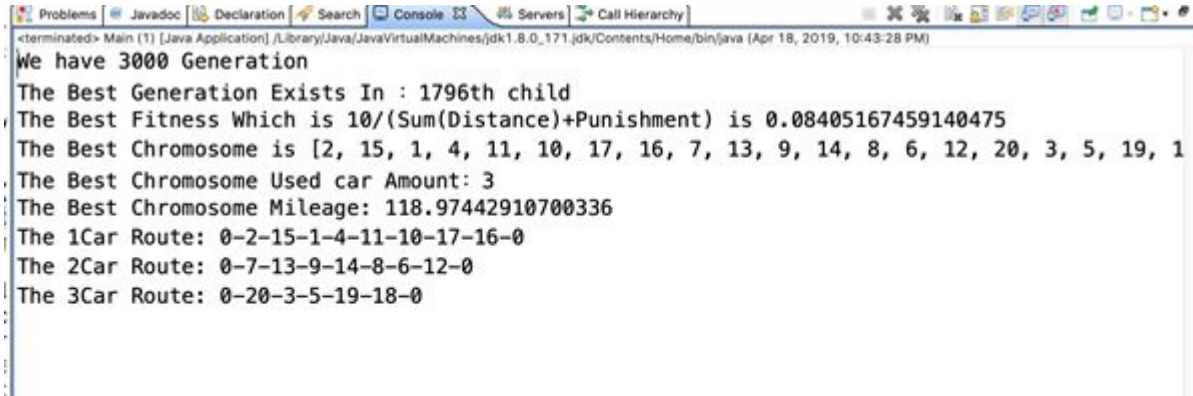
- CalculateFitnessTest
To make sure our fitness sum is 1.
- MutationTest
To make sure our mutation is worked randomly and properly.
- CountRateTest
To make sure our accumulation probability calculated properly



3.2 Conclusions

3.2.1 Output

The output of our GA project is shown below with the *initialization population*, *best result with generation number*, *fitness*, *vehicle number* and the *genotype and phenotype* form:



```
<terminated> Main (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_171.jdk/Contents/Home/bin/java (Apr 18, 2019, 10:43:28 PM)
We have 3000 Generation
The Best Generation Exists In : 1796th child
The Best Fitness Which is  $10 / (\text{Sum}(\text{Distance}) + \text{Punishment})$  is 0.08405167459140475
The Best Chromosome is [2, 15, 1, 4, 11, 10, 17, 16, 7, 13, 9, 14, 8, 6, 12, 20, 3, 5, 19, 1]
The Best Chromosome Used car Amount: 3
The Best Chromosome Mileage: 118.97442910700336
The 1Car Route: 0-2-15-1-4-11-10-17-16-0
The 2Car Route: 0-7-13-9-14-8-6-12-0
The 3Car Route: 0-20-3-5-19-18-0
```

3.2 Conclusions

3.2.1 Conclusion

We also displayed a dynamic visualization of the whole process.

With time goes on, after completing crossover and mutation periodically, the best fitness of each generation actually grows. So the best result is achieved when the genetic algorithms runs. The feasibility and correctness are proved through this display.

