Genetic Algorithm for Capacitated Vehicle Routing Problem

Adam Pehrson[1], **Renan Frantz**[1]

[1]Uppsala University
Lägerhyddsvägen 1, 752 37 Uppsala

Abstract. *Following are sections explaining the algorithm itself, as well as the operators decided upon, then, a section with 30 runs and their results, as well as an analyses of the best one.*

1. Introduction

The objective of this project is to make an implementation of a Genetic Algorithm for the Capacitated Vehicle Routing Problem, the problem itself consists of getting the optimal routes (minimize total travel distance) for each truck, considering a maximum amount of cargo permitted for each truck.

The input was a txt file with every node (delivery destination), each node contains the quantity to be delivered, their coordinates, and their ID. Following are sections explaining what representation was used for the trucks and their routes, the methods chosen to solve the problem, and an analysis of the results.

2. Representation of the problem

To represent the problem a unique approach was used. We have two arrays for each team of trucks, one is an one-dimensional array where each index corresponds to that truck's number of nodes taken, and the other is a two dimensional array where we store the nodes in a certain order. Each truck travels through the nodes that are in it's section of the route, which is defined by the number o nodes it takes.

For example, imagine we have the following sequence of nodes: (1,2,3,4,5,6,7,8), and the following truck sizes: (3,5), in this scenario truck 1's route will be: (1,2,3), and truck 2's will be: (4,5,6,7,8). With this representation, crossover can happen between routes without worrying about nodes being deleted, and every team will always cover all nodes. Restrictions are put in place so that the sizes of the trucks always sum to the total number of nodes.

3. Genetic Algorithm operators and parameters

When implementing a Genetic Algorithm we have a myriad of options to choose from. For crossover we chose the OX operator, for mutation, the scramble operator was chosen. For mutation of the trucks, a add/subtract operator was chose.

3.1. OX Crossover

The OX crossover consists of selecting a sub-sequence from the parent and copying it to the child on the same place, then, every other node is copied from the other parent. The implementation of this operator is done in two steps, first, we copy the string from one of the parents, then we move across the other parent copying their nodes, except for the ones from the other parent, when we reach the sub-sequence of the first parent, we skip it and keep going after it's end.

3.2. Scramble Mutation

The scramble mutation is simple, we go through the sequence of nodes and generate a number between 0 and 100 for each, if the number is below the mutation rate, the node mutates. This mutation consists in scrambling the mutated node and the x (x being the chosen number of nodes mutated, in our case, 5 was used) nodes ahead of it.

3.3. Mutation for the trucks

For mutating the truck's sizes, we go through each truck and generate a number between 0 and 100 for each, if the number is below the mutation rate, the truck mutates. When a truck mutates a number between 0 and 5 is added to it's size, then, another node has that same value subtracted from it, this way, we keep the total nodes visited exactly equal to the number of nodes. If the number to be subtracted is smaller than the size of the truck, the size of the truck-1 is used instead.

3.4. Fitness function

The fitness function is the total distance traveled, with one special instruction: if any trucks exceed their capacity, +10000 distance is added to that team.

4. Results

Following is 30 independent runs:

```
-12504.9
CPU time used: 5.05109 ms

-2320
CPU time used: 5.05279 ms

-12362.5
CPU time used: 5.06748 ms

-12314.4
CPU time used: 5.34497 ms

-2546.49
CPU time used: 4.95206 ms

-2557.7
CPU time used: 5.2637 ms

-2434.05
CPU time used: 5.58227 ms

-12383.7
CPU time used: 5.61351 ms

-2180.18
CPU time used: 6.28582 ms

-2410
CPU time used: 5.77538 ms

-12148.4
CPU time used: 5.50957 ms

-12576.4
CPU time used: 5.99946 ms

-12513.1
CPU time used: 5.62067 ms

-12594.4
CPU time used: 5.50958 ms

-2203.23
CPU time used: 5.20073 ms

-12348.7
CPU time used: 4.9336 ms

-12220.6
CPU time used: 5.06646 ms

-12304.6
CPU time used: 4.99616 ms
```

```
-2256.14
CPU time used: 5.36124 ms

-2630.75
CPU time used: 5.4741 ms

-12430.4
CPU time used: 5.36303 ms

-12225.7
CPU time used: 5.44139 ms

-2474.78
CPU time used: 5.37707 ms

-12353.8
CPU time used: 5.41835 ms

-12434.9
CPU time used: 5.35299 ms

-12230.5
CPU time used: 5.37079 ms

-12459.7
CPU time used: 5.37877 ms

-12077.7
CPU time used: 5.36447 ms

-12451.4
CPU time used: 5.73288 ms

-12630
CPU time used: 5.62428 ms
```

The best result had a total distance of 2180, and can be seen below:

```
Sizes: 5
|20||17||32||24||20|
Total cargo shipped: 78
Total distance traveled: -186.804
---------------------------

Sizes: 7
|22||52||27||42||26||25||44|
Total cargo shipped: 98
Total distance traveled: -287.663
---------------------------

Sizes: 5
|5||18||1||15||8|
Total cargo shipped: 92
Total distance traveled: -252.227
---------------------------

Sizes: 6
|21||47||49||48||6||40|
Total cargo shipped: 98
Total distance traveled: -207.432
---------------------------

Sizes: 8
|30||29||31||28||12||36||37||51|
Total cargo shipped: 93
Total distance traveled: -306.767
---------------------------

Sizes: 5
|51||38||33||14||4|
Total cargo shipped: 84
Total distance traveled: -182.857
---------------------------

Sizes: 8
|10||16||11||44||35||40||54||14|
Total cargo shipped: 100
Total distance traveled: -396.932
---------------------------

Sizes: 7
|46||8||23||41||4||34||3|
Total cargo shipped: 99
Total distance traveled: -231.548
---------------------------

Sizes: 3
|45||9||54|
Total cargo shipped: 97
Total distance traveled: -127.95
```

5. Conclusion

The results were far from optimal. We were unable to arrive at an implementation that could quickly converge to the optimal solution, the best solution we managed to arrive at is approximately 100% worse than the optimal. Time constraints make it impossible to make a better algorithm at the moment, but improvements are certainly possible.

There are two main things to be improved about the algorithm, first a crossover operator for trucks should be included, possibly something correlated to the crossover of the sequence of nodes. Second some kind of constraint so that every generation is valid, and no generation has trucks exceeding the cargo.

With those two improvements the algorithm would be able to perform much better, and get closer to the optimal solution.

References