IT3708 - Assignment 1

Erik Liodden (eriklio)

February 2019

Chromosome representation

The chromosome is represented with a matrix M_{ij} where each column corresponds to a vehicle at a depot and the route for that vehicle is listed downwards starting at row 0. Every route follows the format $[d_{start}, c, c, ..., c, d_{end}]$, which determine the start depot d_{start}, end depot d_{end}, and the customers along the way in ordered apparence. An example chromosome from dataset `p01` is shown in figure 1.

```
chromosome
 50 50 50 50 51 51 51 51 52 52 52 52 53 53 53 53
  3 16 41 50 46 11 45 47  9  8 52 48 19 53 21 53
 17 36 18 -1 13 51 30  7 32 15 -1  4  1 -1 27 -1
 40 14 39 -1  5 -1 25  0 38 49 -1 37 28 -1  2 -1
 50 44 12 -1 26 -1  6 31 29 33 -1 52 20 -1 35 -1
 -1 43 24 -1 51 -1 42 10 52 52 -1 -1 53 -1 34 -1
 -1 50 50 -1 -1 -1 23 51 -1 -1 -1 -1 -1 -1 53 -1
 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 51 -1 -1 -1 -1 -1 -1 -1 -1 -1
```

Figure 1: Example chromosome representation of dataset `p01`. Each column in the matrix represents a route. Note that all customer and depot numbers are shifted down by 1.

Another possible representation could be to have a single list of all the routes where each route is separated by the start and end depot of the current route and the next:

$$[d_{1,start}, c, c, ..., c, d_{1,end}, d_{2,start}, c, c, ..., c, d_{2,end}, ..., d_{n,start}, c, c, ... c, d_{n,end}]$$

I chose the matrix representation because it made it easier for me to separate out the routes based on column indexing. The disadvantage of the Matrix representation is that it takes up more space because all columns have the same length and it has to be able to store a single route with all the customers. A value of -1 indicates an empty element. Both representations are suitable for this problem, but I think the matrix representation is better from a practical point of view.

Crossover and Mutation

All individuals in the initial population are generated to be feasible solutions. The crossover algorithm selects two parents and recombine them into two off-springs. If one (or both) of the off-springs represents an invalid solution, the off-springs are discarded and the same parents are recombined again to generate two new off-springs. This cycle continues until both off-springs are feasible. This ensures that the whole population is feasible at the end of the recombination phase. Mutation does not check for feasibility, and infeasible solutions are kept until the next generation. However, the last generation is guaranteed to not mutate.

Parameter relation

In GA, the parameters population size, generation number, crossover rate, and mutation rate are related. A large population will converge slower and needs many generations to converge to a good solution, while a small population might converge too fast towards a local minima and will not explore enough to find a better solution. In early stages of the evolutionary cycle, a small population might have found a good solution than a large population, but the large population might have found an even better one in later stages of the cycle, mainly because of higher diversity in the population and therefore a higher probability to escape a local minima. As seen in figure 2, the populations with few individuals converge to a less optimal solution than the populations with many individuals.

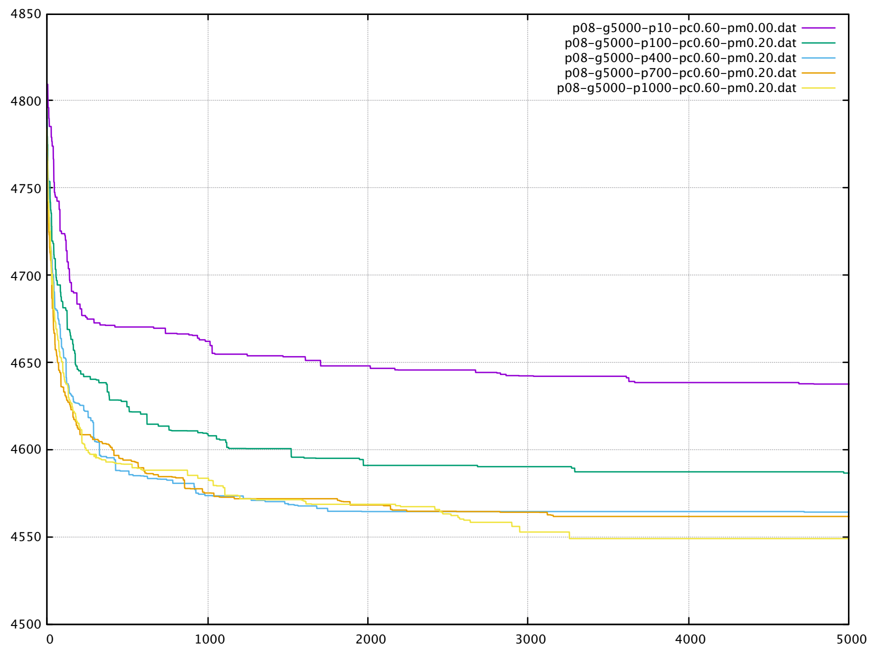

Figure 2: Best individual (by total route length) for a variable population size (10, 100, 400, 700, 1000). over 5000 generations. Probability of crossover is 0.6 and probability of mutation is 0.2.

Source Code

The source code for the implementation of Assignment 1 can be found in `source.zip`. The solution is implemented in C++, using C++14 and the Armadillo library for matrix and vector operations and Gnuplot for plotting.