Multi-Depot Vehicle Routing Problem

By: Erik Wiker & Agnete Djupvik

Chromosome Representation
The group decided to represent the chromosome as a two-dimensional array of depots, with each depot array containing the customers which the depot's _n_ vehicles will visit.
A shortened (and thus incomplete) example follows:
```
[[3, 12, 14], [5, 6, 7, 10, 11], [4, 8, 9], [1, 2]]
```
The order of the items internal to an array matters, as they are the order in which the vehicles will visit them. In this representation, the route for each vehicle is generated on demand. This is done by checking whether the current vehicle can both drive to the next customer, carry the demanded load, and return to the depot as well within the given boundaries. This method ensures only viable solutions which focuses the algorithm towards optimal solutions faster.

This representation has two main advantages. For one, it simplifies many both inter- and intra-depot mutation operations, and in addition to this, it guarantees that the number of vehicles used is always minimized.

Alternate representation
Another representation that is viable consists of keeping constant track of the vehicles within the depot. This would cause the previous example to look like this:
```
[[[3, 12], [14], []], [[5, 6, 7], [10], [11]], [[4, 8, 9], [], []], [[1], [2], []]]
```
This representation is more explicit on the display of vehicles. This means that vehicles that are not in use are displayed as empty arrays within the depot. The group used this representation for some part of the project, but moved away from it as mutation and minimization benefits were so large with the previously mentioned implementation, as well as the general process of iteration on the array is much quicker in only two dimensions.

Initialization
To prevent a completely random initialization of customer assignments to depots, _depot clustering_ is performed to assign customers to their nearest depots.
With some customers, we can be sure that they will never be visited by a vehicle from another depot. We set a _bound_ so that
_(distance(c,d_i)−min) / min ≤ BOUND_
Where _distance(c,d_i)_ indicates the Euclidian distance between the customer _c_ to a depot _d_i_.
If a customer is within the bound distance for only one depot, we can be sure that it will never contribute to an optimal path by being visited by a vehicle from another depot. If the customer is within the bound distance for several depots, it is initially assigned to the nearest depot. The customer is also added to the other close depots' _borderline_. This is an array for each depot which indicates customers that _may_ be included in an optimal path, and the array is used in _inter-depot mutation_, which we discuss later.

Fitness
We use a _weighted fitness_ function to indicate the fitness of a given chromosome.
(a * num-cars) + (b * path-length)
Where _a_ and _b_ are variables set to weigh the parameters against each other. We find the optimal path by minimizing the number of cars that are driving, so this is a heavy priority _before_ minimizing the total path length. We therefore set _a=100_ and _b=0.001_.
Crossover
Tournament Selection
Parents are selected by using Tournament Selection with a slightly elitist approach.
We begin finding a parent by selecting two random individuals from the population. Then, in 80% of cases, we choose the fittest individual for reproduction. Otherwise, it is chosen randomly.

In the crossover process, some infeasible solutions may be created. The process is as follows:

1. Depots _d1_ and _d2_ are selected from parents _p1_ and _p2_ respectively.
2. We remove all vehicles in _d1_ from _p2_ and vice versa.
3. We find all costs of insertion in a route and store these in an ordered list.
4. We insert vehicles using two methods:
4a. In 80% of cases, we insert a vehicle in the best _feasible_ position.
4b. In 20% of cases, we insert a vehicle in the best position, regardless of feasibility.

These infeasible cases often get a bad fitness, because it requires more vehicles than we even have to maintain the route. This way, it opens up for a mutation that may lead to a better result, but if it proves unuseful, the weighted fitness function will eventually be its demise.

Parameter Values
In the development of the algorithm, fitting parameter values were critical to the success of the system.
Our __population size__ was fairly small, around 30. This could probably have been bigger, but proved sufficient for our purposes and restraints in terms of running time.
The __mutation rate__ was set to around 0.03 in the end. We also implemented a decay for the mutation rate so that the system will mutate less as the run progresses.
Other parameters, such as generation number and crossover rate were not used as we found good results without them, and other studies had created sufficient solutions which did not take these into account.

Higher mutation with a large population size means a lot of noise, which in certain cases can be good but also tends to make runtime of the GA longer. Too high mutation rate will provide only random solutions, while too low mutation rate tends to make the GA have to little variation in individuals which will make the algorithm get stuck on local minima. If the population is small GAs tend to have higher mutation rates to make up for little variation.

It tends to be beneficial to have a high mutation rate in the beginning of a GA to find slopes which will lead to many minimas. This mutation rate is then decreased over time (simulated annealing) so that the algorithm won't over shoot the minima.

Mutation
Inter-Depot Mutation
Inter-depot mutation takes place every 10 generations.

Intra-Depot Mutation
Mutation occurs depending on the given mutation rate, and then one of three mutation methods are randomly selected.

Swap
Two customers in the depot are swapped and routes adjust accordingly.

Reroute
All customers in a route are redistributed to other vehicles. According to a probability, it is either put in the first viable position, or it is put in the best possible position at the given time.

Inverse
A segment of a route is reversed.

Social disasters technique
This technique is used when the individuals in a population is too alike (fitness is similar), this means that the gene pool has grown stagnant and few new features are introduced. The technique is to replace a certain amount of the population (here experimentally set to 70%) with new random individuals to import new features into the gene pool