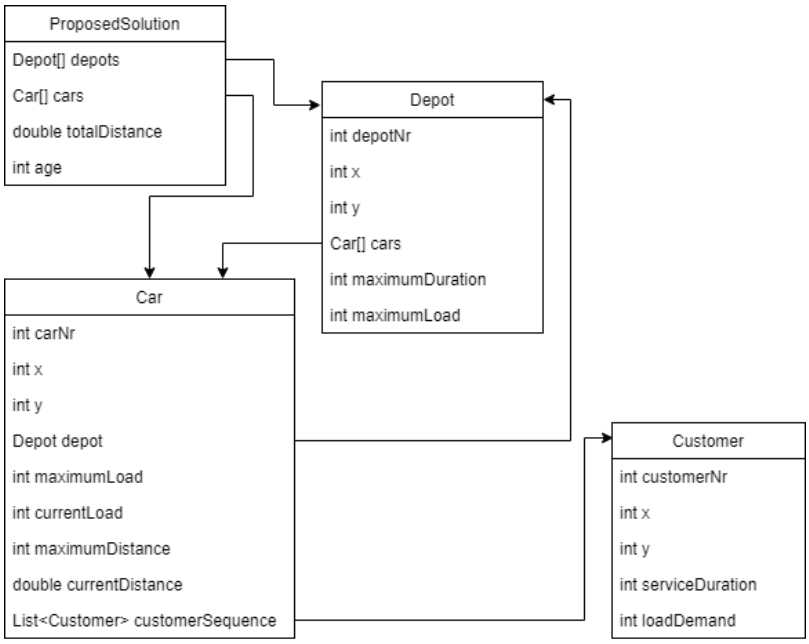
Martin Stigen and Eirik Baug

IT3708 - Project 1

Solving The Multiple Depots Vehicle Routing Problem (MDVRP) Using Genetic Algorithm (GA)

1. **Describe the Chromosome representation that you used in your implementation. Also, mention another representation that could be used for this problem and why this representation is also suitable. Defend your choice of the most suitable representation. (2p)**

For this project we used Java. An object oriented approach for the chromosome representation was therefore selected.
In our project the chromosome is represented as a ProposedSolution object.
The ProposedSolution object keeps track of the the depots, cars and customers, implemented by using lists and Depot, Car and Customer objects.
Under is a simplified image illustrating this relationship:



Another representation for the chromosome could be a string, consisting of sequences of cars and their routes. This option would require very little overhead and is a compact representation of the solution.
We chose to use an object oriented approach because of the advantages of saving data like currentDuration and currentLoad directly in the objects instead of having to calculate it and save it outside the chromosome. The relationships also made it very easy to iterate over depots, cars and customer belonging to a chromosome.

2. **Describe whether the crossover and mutation operators will produce infeasible off-spring(s) after executing. If yes, how did you handle that? If not, why not? (2p)**

The crossover and mutation operator will not produce infeasible off-springs. This is because when a crossover or mutation is made the code checks whether it creates an infeasible solution or not. This is done by checking whether the load and duration requirements are valid before inserting a customer to a given route. If it turns out not to be possible the offspring is discarded and new attempts are made until a feasible solution is created.
We chose to not allow illegal solutions because this simplified how to calculate the chromosome fitness score, making the total distance of the solution the only heuristic we had to deal with.

3. **In GA, often the parameter values (population size, generation number, crossover rate, and mutation rate) have some form of relationship. Considering that you tested several set of parameter values, what relationship did you observe among these parameter values? Also, describe the effects of these parameter values during the early and later stages of evolutionary cycle? (2p)**

We observed that having a large enough population size was very important to not get stuck in local minimums. This also allowed for more diversity in the population, letting non-optimal solutions live long enough to hopefully 'evolve' into the best solution found.
The crossover rate had a strong relationship to the population size. To achieve good results we decided to create at least as many offsprings by crossover as the population size.
The mutation rate also played a crucial part in creating more diversity rather than enhancing the quality of the population.

During the evolutionary cycle we found that in the later stages the population often converged towards a local or hopefully the global minimum, but most often the local one. To avoid getting stuck in the local minimums we found that having the parameters tuned dynamically during each iteration (generation number) improved overall performance. Having a focus on good crossover rate in the early stages and turning up the mutation rate as the evolutionary cycle went into the later stages gave very good results. This allowed for fast convergence towards a minimum eary, and if the minimum turned out to be a local one the increasing mutation rate would help to "shake out" of the local minimum and continue searching for the global one.

To further improve the algorithm we introduced a "age" parameter restriction, allowing us to discard solutions after a given number of iterations. This was very helpful to give room to new solutions that in turn would become better solutions than the old ones.

We spent several hours on tuning the parameters and finding a good implementation of making some of them, like mutation rate, dynamic. As part of the solution we found that allowing for replacing parts of the population with a new generated one if the algorithm got stuck also helped on getting out of the local minimums and increasing diversity.