Exercise 5
IT3708

Simon Borøy-Johnsen
MTDT

May 11, 2016

1 Implementation

1.1 MOEA design

The MOEA algorithm follows the flow in Figure 1. Front generation and crowding distance calculations are implemented according to the descriptions in the lecture slides. Tournament selection (group size 10, epsilon 0.1) is used for selecting parents. Crossover and mutation rates are on a per-genome-basis; rate of 0.9 means that there is a 90% probability of an event to happen to the whole genome.

1.1.1 Genotype

The genotypes are represented as lists of all the city indices (zero-indexed), in the order that the salesman are visiting them. Different permutations of the list imply different routes for the salesman.

1.1.2 Crossover

For crossover, I used a customized version of the "ordered crossover" method.

The method creates a random number of intervals along the genome space. Even numbered intervals are copied from parent one to child one. Odd numbered intervals are copied from parent two to child two. The cities missing from each child is added in the order they appear in the opposite parent's genome.

Figure 2 shows an example of how the crossover method works.

By copying parts of the parents directly into the children, and then adding the missing cities, the crossover method ensures that all cities are part of the genome, and only once. The crossover method does therefore not yield infeasible offspring.

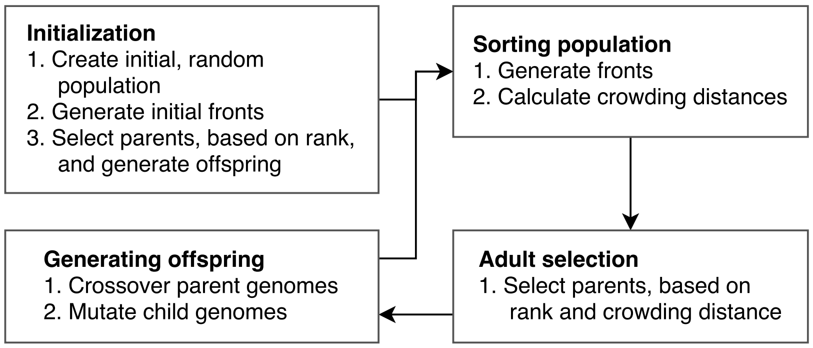

Initialization
1. Create initial, random population
2. Generate initial fronts
3. Select parents, based on rank, and generate offspring
Sorting population
1. Generate fronts
2. Calculate crowding distances
Adult selection
1. Select parents, based on rank and crowding distance
Generating offspring
1. Crossover parent genomes
2. Mutate child genomes

Figure 1: Flow of MOEA algorithm

1.1.3 Mutation

For mutation, I used the "reverse sequence mutation (RMS)" method.

When using RMS, a random interval along the genome is chosen. The gene order in the interval is reversed.

Figure 3 shows an example of how the mutation method works.

By reversing a subset of the original genome, the mutation method ensures that all cities are part of the genome, and only once. The mutation method does therefore not yield infeasible offspring.

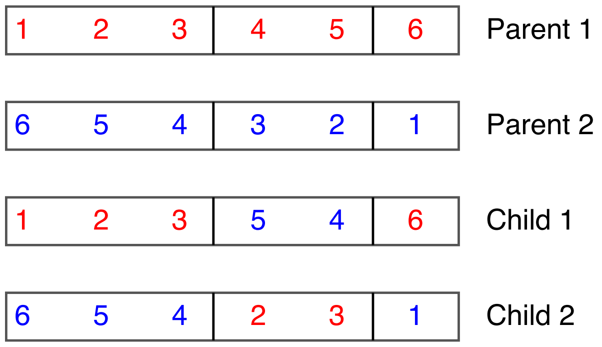


Figure 2: Visualisation of the crossover method

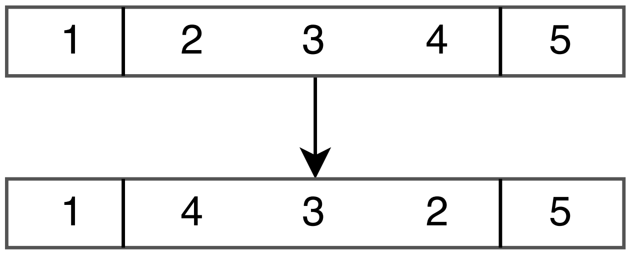
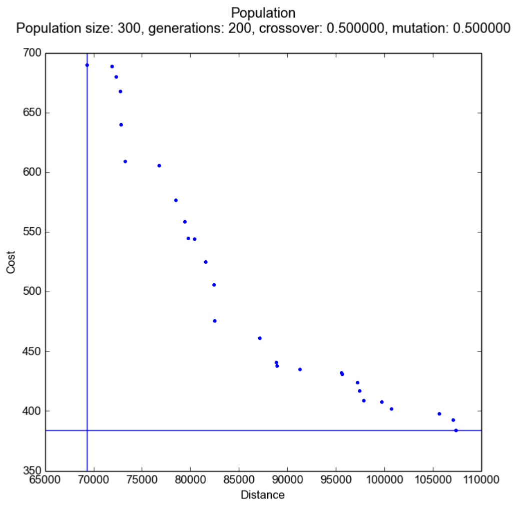


Figure 3: Visualisation of the mutation method

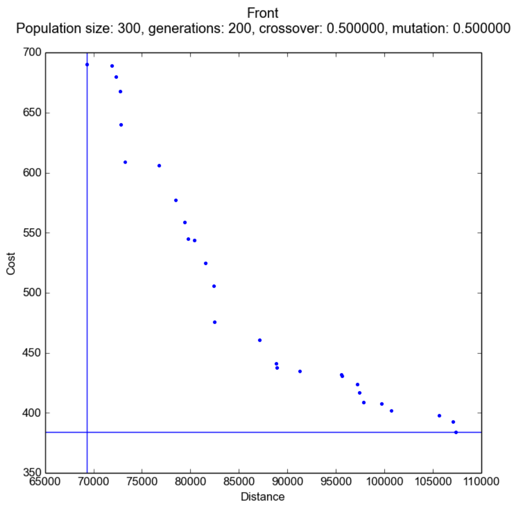
2 Parameters

#	Population	Generations	Crossover rate	Mutation rate	Distance	Cost	Pareto-front size
1	300	200	0.5	0.5	Max: 2199, min: 384	Max: 192349, min: 66404	300
2	100	500	0.5	0.5	Max: 2098, min: 294	Max: 180182, min: 86355	100
3	200	500	0.9	0.5	Max: 2255, min: 351	Max: 183000, min: 72406	200

3 Results

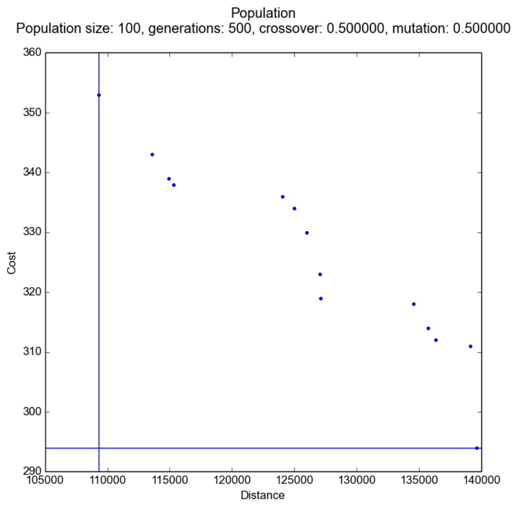


(a) Population of configuration 1

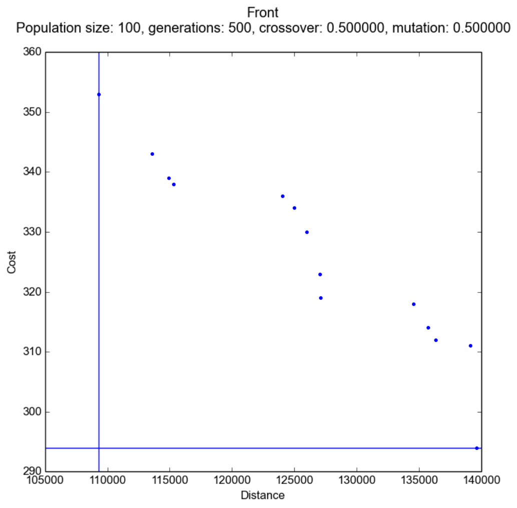


(b) Pareto-front of configuration 1

Figure 4: Results for configuration 1

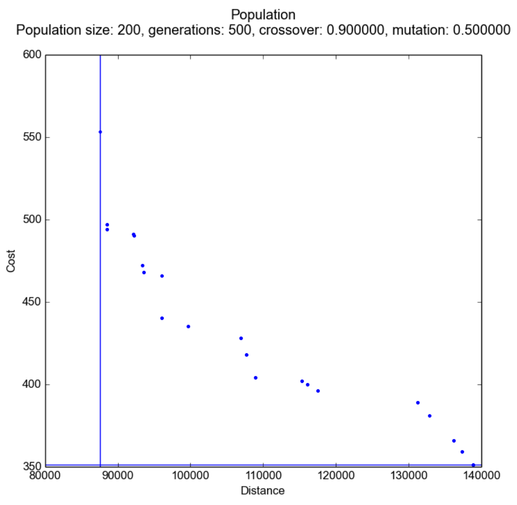


(a) Population of configuration 2

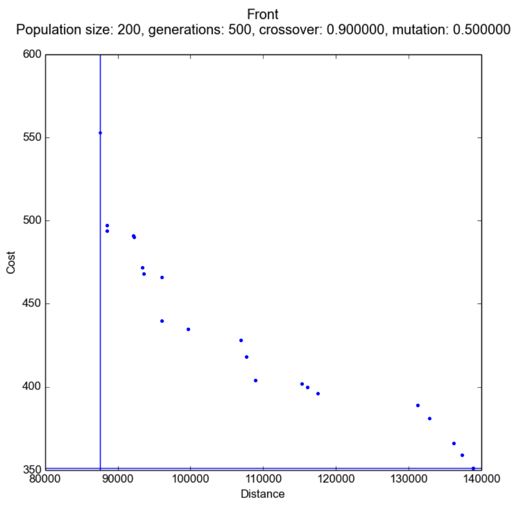


(b) Pareto-front of configuration 2

Figure 5: Results for configuration 2



(a) Population of configuration 3



(b) Pareto-front of configuration 3

Figure 6: Results for configuration 3

As can be seen in Figures 4, 5, and 6, the final population converges towards the pareto front. This happens when the simulation is run for a certain amount of generations. The population is initially spread, but converges towards the front, as these solutions are among the best the algorithm can find with given configurations.

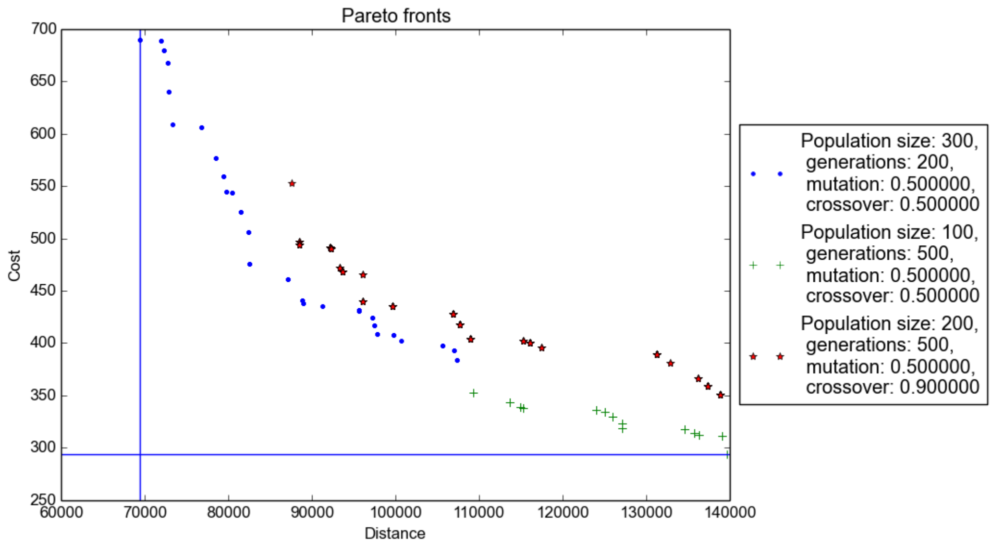


Figure 7: Comparison of the pareto-fronts of the different configurations