Project 5 - Multi-Objective Evolutionary Algorithms - Multi-Objective TSP

Aleksander Skraastad

May 11, 2016

1 Implementation

1.1 MOEA Design

I have written my MOEA in Java.

For my MOEA i have chosen to use a 48 element vector of shorts (16bit int) as my genotype. Each short represents the ID or index of a specific city in the cost and distance tables, hence no alterations to the genotype was necessary to represent the problem phenotype.

For mutation, I have opted to use a standard mutation rate of $[0.0, 1.0]$. The mutation itself may also have an intensity $[1, 48]$, which I have defined as the number of mutations occurring in the same individual on the current mutation pass. Each mutation is defined as a swap of two city IDs, selected by two random indexes in the genotype.

The crossover is implemented as a random $[1, 47]$ section of the genotype that is directly applied to the opposing genotype. For each element that is transferred between the genotypes, a full pass on the remainder genotype values is performed, and cross-checked with the previous value. This ensures the integrity of the genotype, and prevent duplicates.

For selection, I use crowding distance tournament selection. This is implemented such that K random individuals are selected from the total population, and sorted with the crowding operator. The crowding operator first assesses the rank of the individual, then crowding distance if the ranks are identical. The best suited is selected as tournament winner.

The evolution loop itself is quite simple, performing population merge (to form 2N pop size), then non-dominated sort, before adding each front and applying crowding distance until the standard population size of N is reached for the new population.

1.2 Mutation and crossover details

Since my implementation strategy for mutations consist of swaps, this prevents adding duplicate city IDs to the genome. It also ensures that no city ID is missing, securing a perfect one-visit round-trip.

The same goes for the crossover implementation. As each of the values being crossed over are transferred, an extra pass over the remainder of the genome is performed to check for duplicates, and when the potential duplicate is found, it is replaced by the previous value that was overwritten. This secures the integrity of the genome, and a perfect one-visit round-trip.

2 EA Choices

	A	B	C
Population size	1000	200	100
Generations	750	50000	10000
Crossover rate	1.0	1.0	1.0
Mutation rate	0.2	0.3	0.5
$Cost_{max}$	1409	1739	1340
$Cost_{min}$	405	377	360
$Distance_{max}$	111788	115745	117687
$Distance_{min}$	42236	39754	41756
No. solutions in ND-front	1000	200	100

Table 1: 3 Best MOEA Parameters and results

3 Final solutions

To save space and make the plots readable on paper, I have combined the "all members" and "only pareto" plots into one graph per configuration. The pareto front is the bold red plot to the bottom left.

Figure 1: Configuration A

For each dotted line, the line edges signal the mininum and maximum objective values for that rank of individuals.

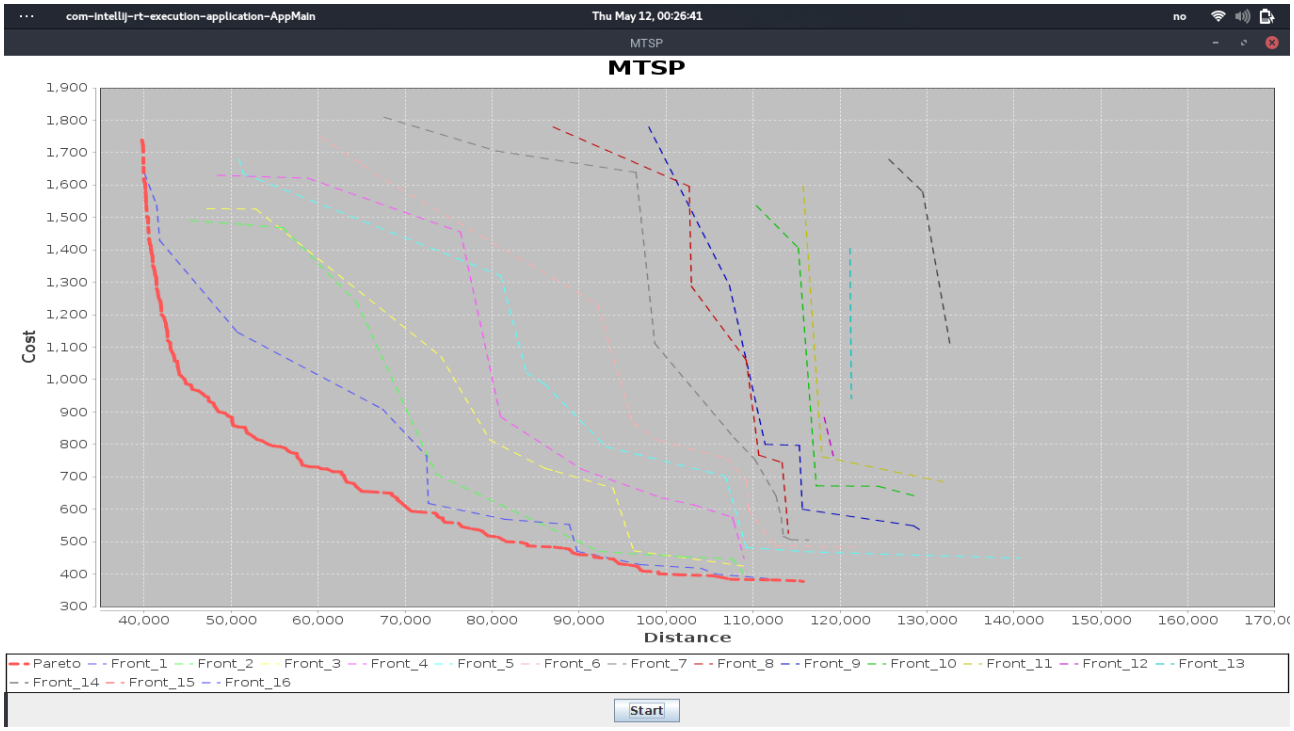

Figure 2: Configuration B

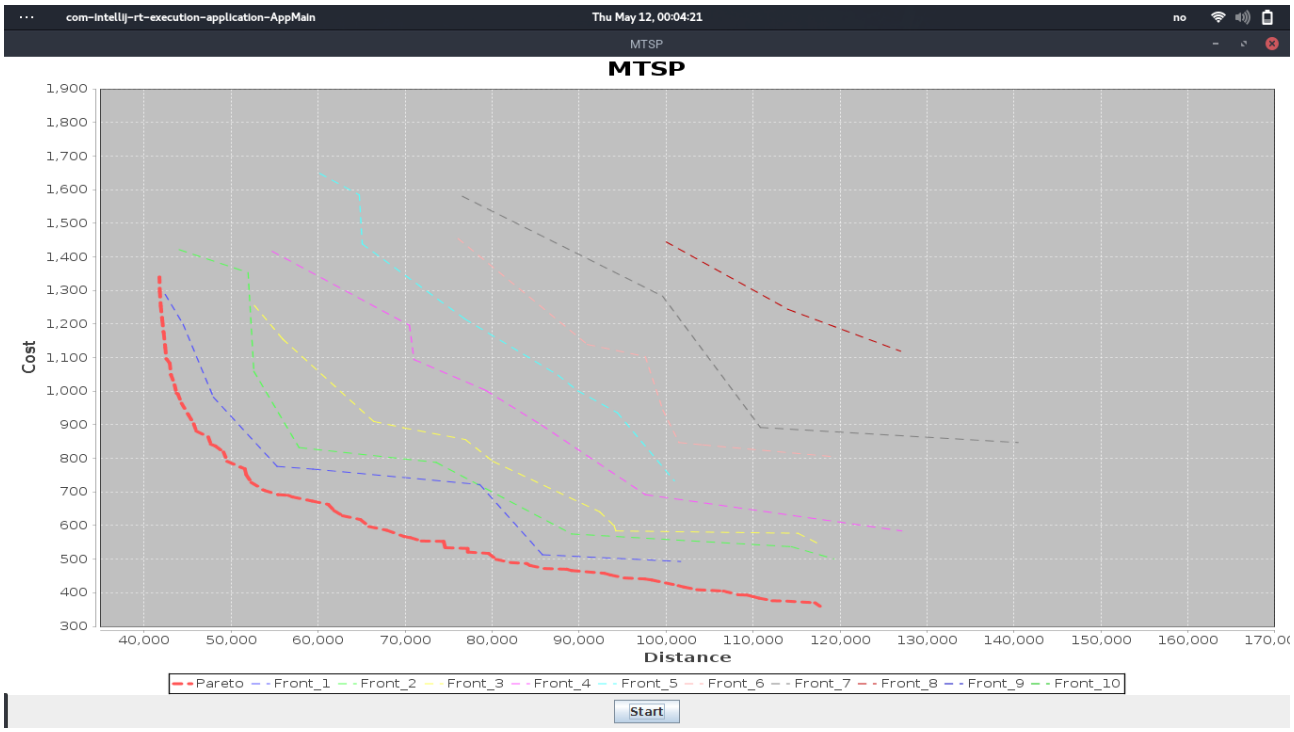

Figure 3: Configuration C

And finally, the overlayed plots of each pareto front from the three configurations:

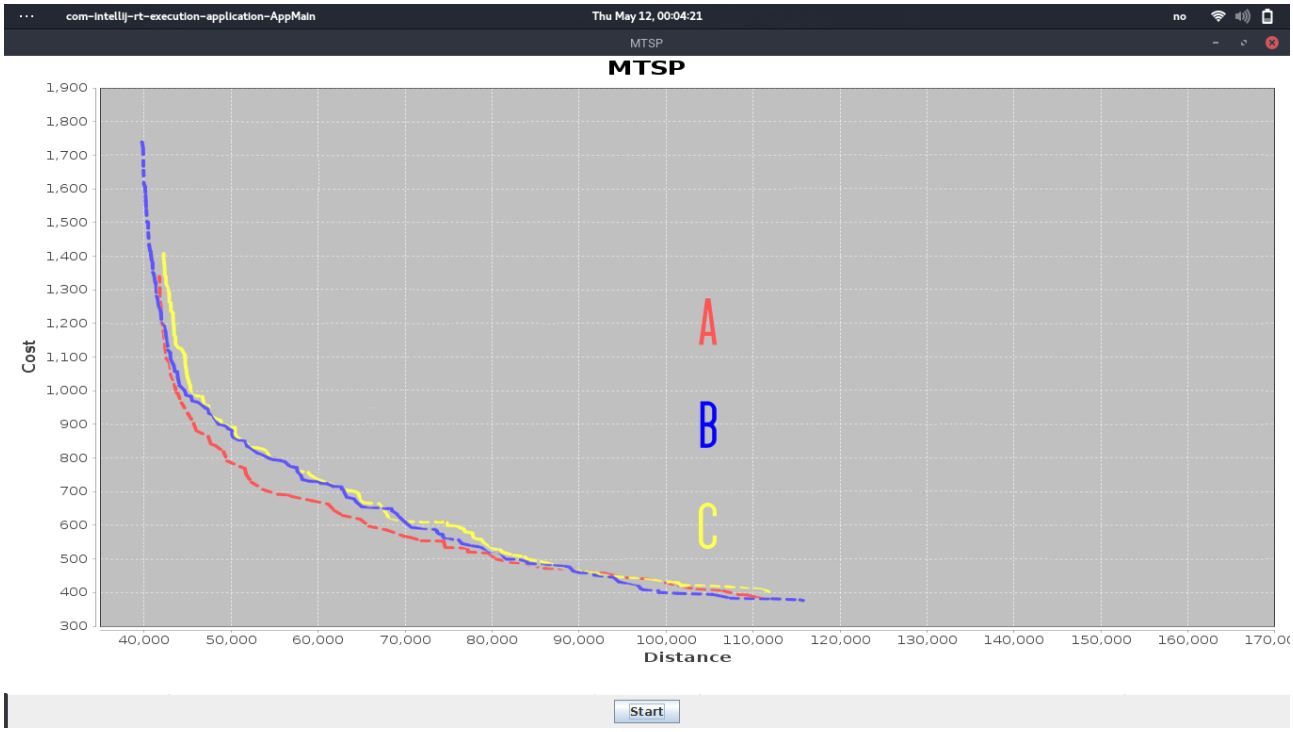

Figure 4: Pareto fronts from all three configurations