CInte: Applied Computational Intelligence

Project 2:
Traveling Salesman Problem

Maxime Roedelé (101520)
Piotr Kasprowicz (101310)
Michael Jasinski (101893)

January 4, 2023

1 Single-Objective Optimization Problem

The goal was to approximate optimal solutions to a TSP with regard to 4 different costs.

1.1 Implementation and design choices:

The optimization was performed by using the DEAP library in Python to create a general algorithm. The implementation was subject to the following design-choices:

a. As individual solutions were lists of cities, a list was used to represent the individual's proposed sequence.

b. Mutations and crossovers had to make sure the individuals remained feasible by having them visit all cities only once. To this effect, the crossover and mutation-schemes **cx-Ordered** and **mutShuffleIndexes** were chosen. They made sure every list retained their summed values by swapping indexes, keeping them feasible w.r.t the TSP.

c. Evaluation of individuals was performed by calculating the cost of their data-sequence. The lower the overall cost, the better the individual was deemed.

d. Selection was done by a simple tournament-selector, pinning individual solutions of the population against each other till a new generation was formed. Additionally, elite selection was implemented to retain successful solutions between generations.

e. A heuristic initial solution was introduced to the first population in an attempt to speed up convergence in later generations.

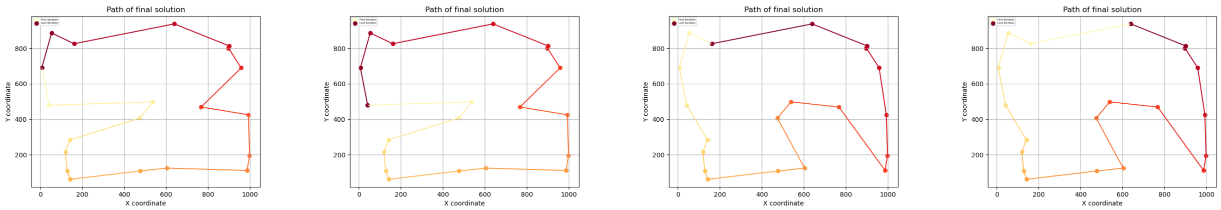
1.2 Results

After tuning of the parameters, the following values were used: population size = 40, probability of crossover = 0.7, probability of mutation = 0.15, probability of individual elements mutating = 0.05, tournament size = 3. After the algorithm was applied with 30 different seeds to a different number of cities with and without heuristics and elitism, the resulting means and standard deviations of distance and cost in Table 1 were obtained:

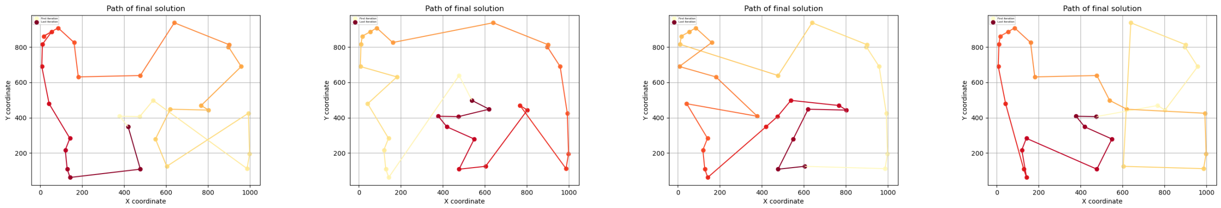
#	CityDistCar		CityDistPlane		CityCostCar		CityCostPlane		Using	Using
Cities	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Heuristics	Elitist
20	5009.11	706.14	5252.97	177.31	997.44	131.62	3989.76	526.48	NO	NO
	4721.93	718.03	5192.25	172.91	939.17	140.42	3756.69	561.68	YES	NO
	5060.62	635.08	5243.09	159.81	1003.85	127.13	4015.41	508.54	NO	YES
	4768.72	691.58	5190.40	174.56	948.70	141.12	3794.80	564.49	YES	YES
30	6709.39	893.68	7701.97	219.86	1397.61	171.39	5590.44	685.59	NO	NO
	6118.37	936.95	7537.42	233.03	1217.89	191.70	4871.58	766.83	YES	NO
	6853.94	877.02	7674.46	218.68	1337.51	181.71	5350.06	726.84	NO	YES
	6123.48	866.54	7541.86	218.74	1222.16	176.96	4888.65	707.85	YES	YES
50	11838.45	1187.84	12927.06	307.99	2347.94	245.59	9391.76	982.36	NO	NO
	9796.19	1346.70	12444.33	333.31	1928.81	271.37	7715.26	1085.49	YES	NO
	11553.89	1259.32	12915.07	297.93	2315.08	245.26	9260.34	981.07	NO	YES
	9414.82	1324.32	12354.49	332.86	1870.79	266.83	7483.16	1067.33	YES	YES

Figure 1: Results after execution of 30 runs

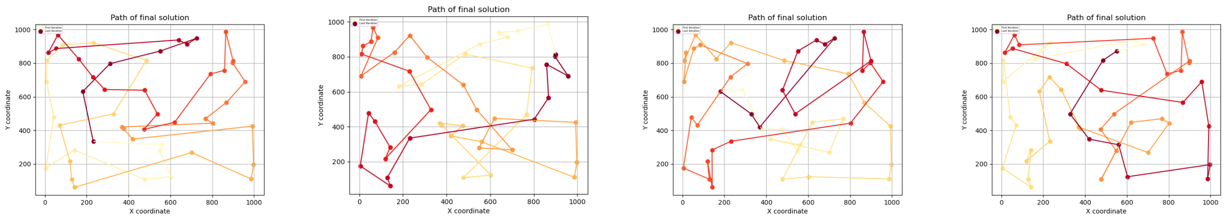
Figures 2-5 show the paths of the solutions and the convergence curves for the different numbers of cities and the different objectives. All images except the rightmost row have elitist selection **enabled**. As the costs-objectives of both cars and planes were close to identical in most cases, they are assumed the same and summarized by the figures labeled **Cost**:



(a) Distance by Car (b) Distance by Plane (c) Cost (d) Cost, no elite s.

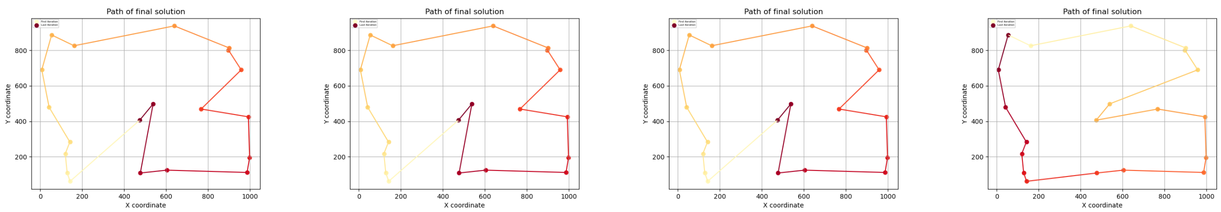


(e) Distance by Car (f) Distance by Plane (g) Cost (h) Cost, no elite s.

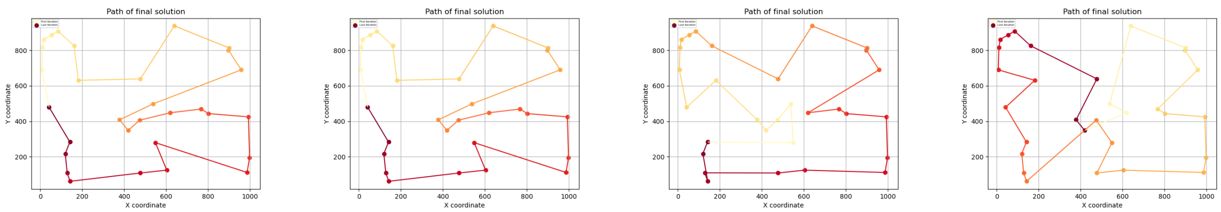


(i) Distance by Car (j) Distance by Plane (k) Cost (l) Cost, no elite s.

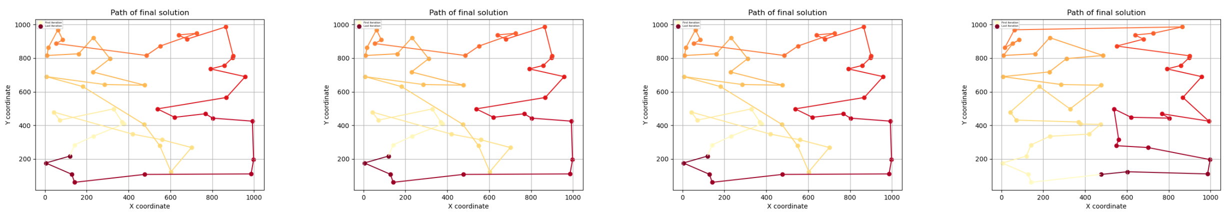
Figure 2: Sol. for 20 (a-d), 30 (e-h) and 50 (i-l) cities, with **no heuristics** and elite selection



(a) Distance by Car (b) Distance by Plane (c) Cost (d) Cost, no elite s.

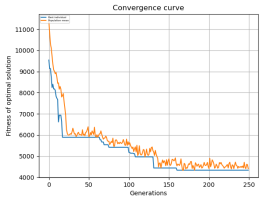


(e) Distance by Car (f) Distance by Plane (g) Cost (h) Cost, no elite s.

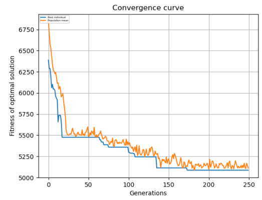


(i) Distance by Car (j) Distance by Plane (k) Cost (l) Cost, no elite s.

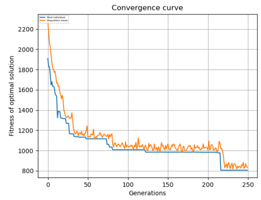
Figure 3: Solutions for 20 (a-d), 30 (e-h) and 50 (i-l) cities, with **heuristics** and elite selection



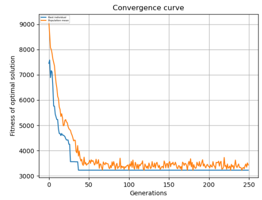
(a) Distance by Car



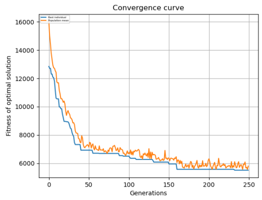
(b) Distance by Plane



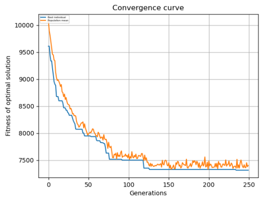
(c) Cost



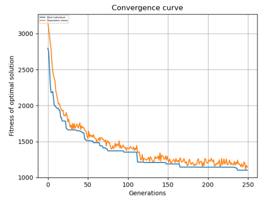
(d) Cost, no elite s.



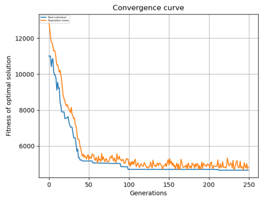
(e) Distance by Car



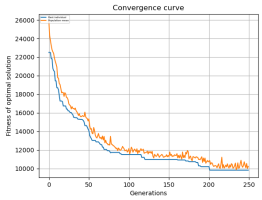
(f) Distance by Plane



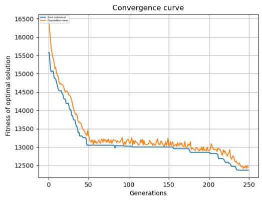
(g) Cost



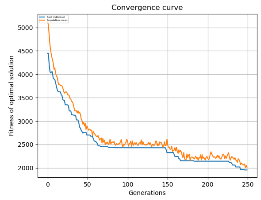
(h) Cost, no elite s.



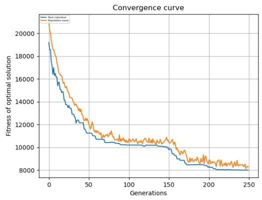
(i) Distance by Car



(j) Distance by Plane

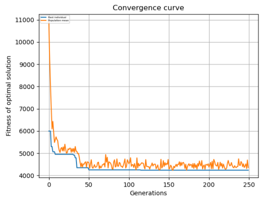


(k) Cost

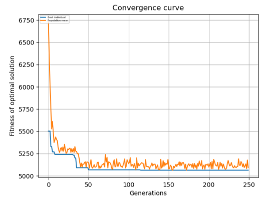


(l) Cost, no elite s.

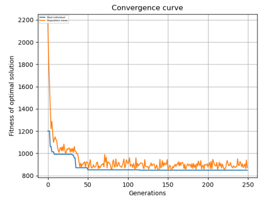
Figure 4: Convergence curves for 20 (a-d), 30 (e-h) and 50 (i-l) cities, with **no heuristics** and elite selection



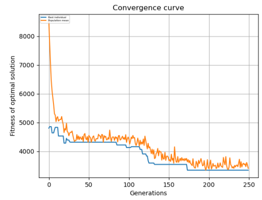
(a) Distance by Car



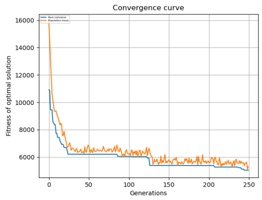
(b) Distance by Plane



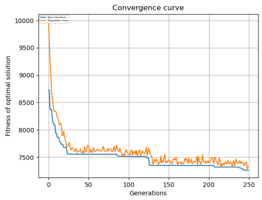
(c) Cost



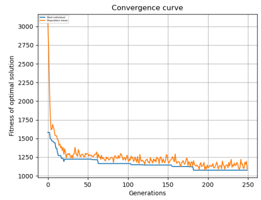
(d) Cost, no elite s.



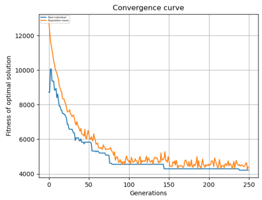
(e) Distance by Car



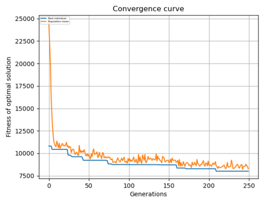
(f) Distance by Plane



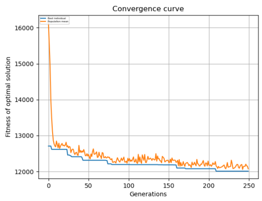
(g) Cost



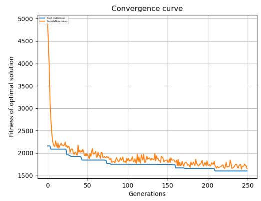
(h) Cost, no elite s.



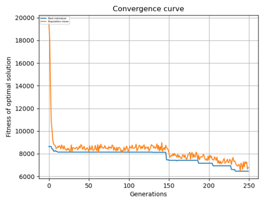
(i) Distance by Car



(j) Distance by Plane



(k) Cost



(l) Cost, no elite s.

Figure 5: Convergence curves for 20 (a-d), 30 (e-h) and 50 (i-l) cities, with **heuristics** and elite selection.

2 Multi-objective Optimization Problem

2.1 Implementation and design

The attempt to solve the problem was performed by using *pymoo* library. The goal ws to generate an optimal Pareto font for the minimizing the cost and time required to visit n cities either by plane or car. The following design-choices were made:

a. An individual in population was represented by 2 lists. One representing a sequence of cities in which they should be visited, and the second one representing a form of transport from city A to B.

b. Operators had to be made custom to fit the 2 lists. Sampling used Random Permutation, mutation used Inverse Mutation and crossover Edge Recombination Operator.

c. Applied algorithm was NSGA-II.

d. Hard constraint, which doesn't allow to more than 3 flight in a row, was handled by using Repair operator, which creating new individual as long as criteria are not meet.

2.2 Results

Unfortunately, in regard to problem with creating individual as 2d array with *pymoo* tasks ended up in a failure. No results were obtained.

3 Final Remarks

It is evident, upon inspection of Table 1 and Figures 2-5, that the use of a heuristic solution in the initial population yields improved convergence on the algorithm in all cases. A good example to consider is the visible difference between Figure 2e, showing a solution achieved without any initial heuristics, and Figure 3e, where such a solution is present in the first population. The path taken with an initial heuristic solution presents a far more logical path between the 30 cities; a statement that is easily validated by visual inspection and numerical analysis. Figure 4e and 5e further show that the convergence towards a better solution is significantly better. The same is evident in most other cases, though visual inspection proves challenging with more cities. One can then safely assume that the heuristic solution with elitism sets a sort of benchmark for future individuals and offspring, where less qualified solution are deemed redundant more quickly.

The inclusion of elitism also showed promising results. Except for with smaller amounts of cities, where the exercise becomes somewhat redundant, the selection-scheme always provides an improvement in convergence, evident from the same results.