Evolutionary Computing Practical Assignment 1

Setup:

For this assignment we used Python. Most of the cases were run on a laptop with CPU: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz.

The ones with population over 1000 were run in Google Collab.

The results we observed are:

Counting Ones

	MINIMUM POPULATION	AVERAGE ITERATIONS	AVERAGE FITNESS EVALUATIONS	TIME	ITERATIONS SD	FITNESS EVAL SD
UX	20	16.8	134	0.31 seconds	1.4	112.0
2X	60	20.5	4920	0.40 seconds	2.13	512.5

Non-deceptive Trap Function (tightly linked)

	MINIMUM POPULATION	AVERAGE ITERATIONS	AVERAGE FITNESS EVALUATIONS	TIME	ITERATIONS SD	FITNESS EVAL SD
UX	130	54.75	23998	6.61 seconds	24.62	15818.83
2X	120	18.6	8928	1.36 seconds	24.035	10296.0

Non-deceptive Trap Function (not linked)

	MINIMUM POPULATION	AVERAGE ITERATIONS	AVERAGE FITNESS EVALUATIONS	TIME	ITERATIONS SD	FITNESS EVAL SD
UX	190	37.85	28766	9.57 seconds	16.98	50016.49
2X	1200	39.45	189360.0	34.47 seconds	16.26	78071.59

Deceptive Trap Function (tightly linked)

	MINIMUM POPULATION	AVERAGE ITERATIONS	AVERAGE FITNESS EVALUATIONS	TIME	ITERATIONS SD	FITNESS EVAL SD
UX	1280, with 14 successes	516.85	2646272.0	1326.017	104.45	534791.96
2X	180	23.05	16596.0	4.75	4.57	30501.75

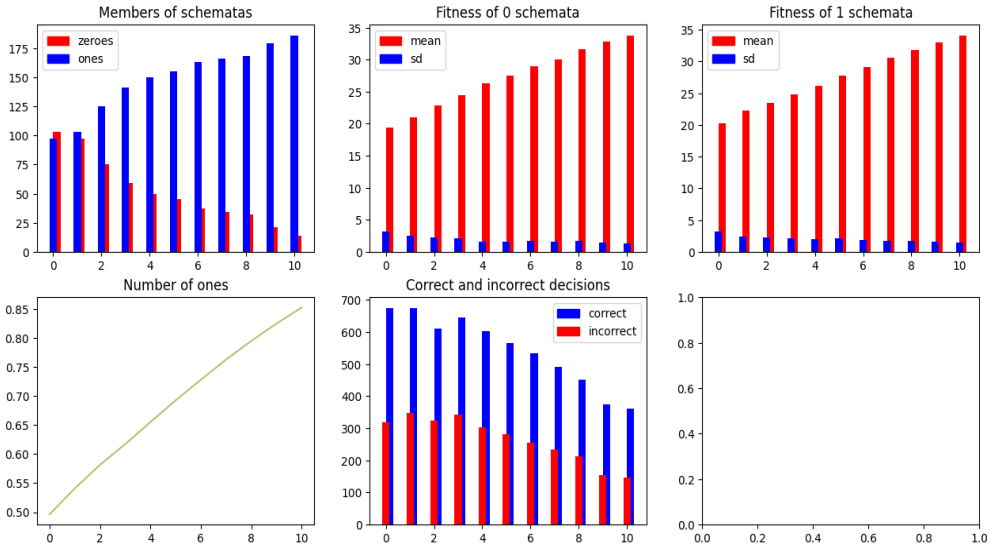
Deceptive Trap Function (not linked)

	MINIMUM POPULATION	AVERAGE ITERATIONS	AVERAGE FITNESS EVALUATIONS	TIME	ITERATIONS SD	FITNESS EVAL SD
UX	1280, with 16 successes	486.3	2489856.0	1378.62	157.74	833177.92
2X	1280, less than 10	153.85	787712.0	256.64	151.08	800808.10

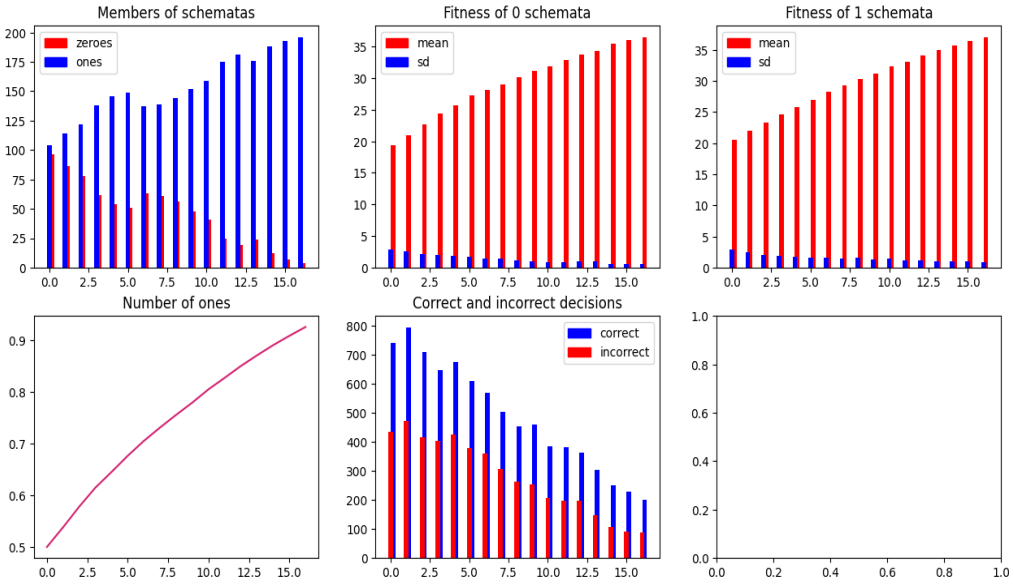
Counting Ones plots

These plots are produced when running the code using the first fitness function (counting ones) with population size 200 for only one run. In all of the cases the solution is quickly found.

Uniform crossover



2-point crossover



Conclusions:

- From this assignment we conclude that there is no distinction between Uniform and 2-point crossover meaning that both perform equally good under different circumstances.
- As expected, “Counting Ones” method was the least cost expensive overall, because using this we found the optimal solution within the smallest population and the least amount of time.
- Deceptive trap Function was the worst cost expensive wise and performed the worst during our computations. This conclusion was made by the fact that three cases using Deceptive trap functions failed to find the optimal solution with at least 19 successes before we reached the population limit.
- For Trap functions which are tightly linked the crossover method doesn’t really have any significant difference, however when they aren’t tightly linked the UX crossover shows a lot better results with a lot smaller population. The situation seems to be reversed when we look at the deceptive trap functions. There when they are tightly linked the 2 point crossover shows remarkably better results.
- The uniform crossover does not distinguish between tightly linked or not as is seen in both the trap functions and the deceptive trap functions, which makes sense since this crossover method doesn’t aim at preserving any patterns.
- In the last case (deceptive, not linked, 2point cross) we observe that all the members of the population become the same relatively quickly and thus it rarely finds a solution before terminating, which happens in less time and with less evaluations than in the UX case.

Group Members:

Liliya Spasova, student number 7473885

Z. Damian Parathyras, student number 7546149