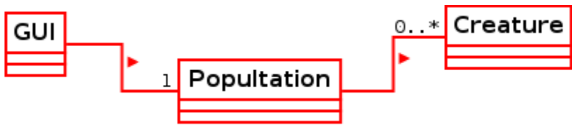
Project 2 IT3708:

Programming an Evolutionary Algorithm (EA)

a) Implementation

The project is implemented in Python, with the a GUI using the tkinter library for choosing all the settings. Other then the GUI, the program has two classes: Creature and Population. Creature mainly contains the fitness functions and Population the selection methods.

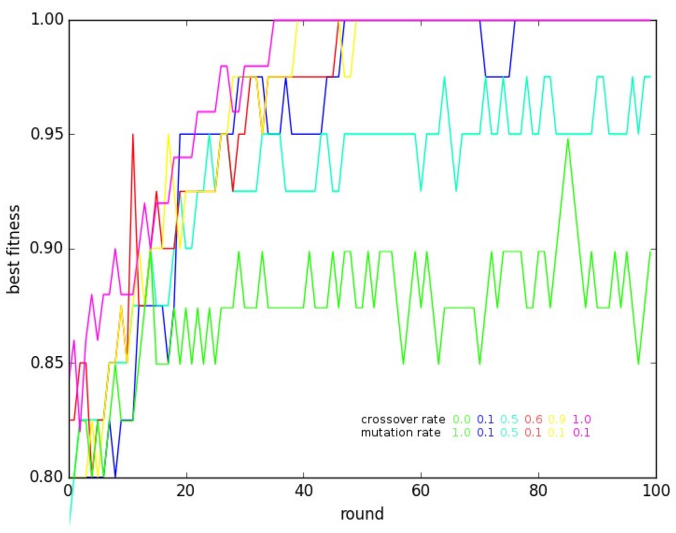


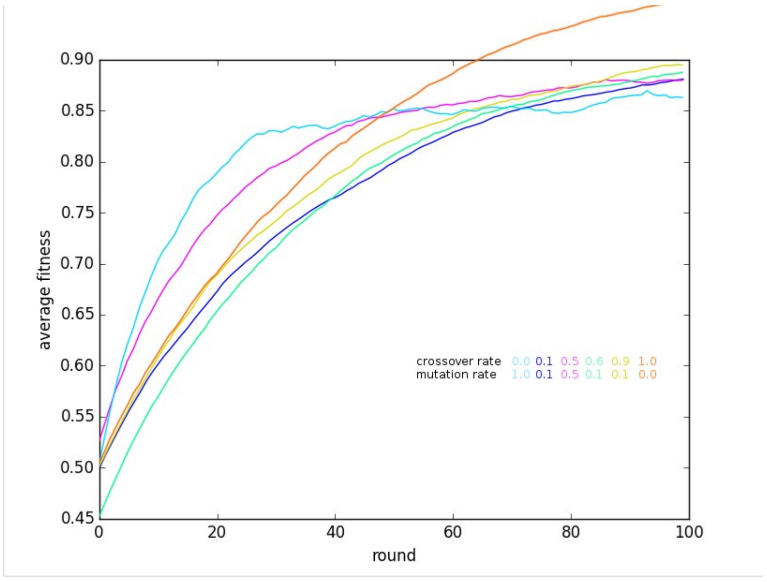
The GUI's start button runs a function that creates a population, with a first genereation of children and starts the evolutionary loop. Thanks to Python's dynamic type checking, the program is already very modular in terms of adding new genotypes and phenotypes. Indiviual genetic operators and selection mechanisms are own functions. Adding a new one means to implement a new function and add it to the GUI, so it can be used.

```
def fitness_count_ones(self):
    self.fitness = 0.0
    for i in range(self.len):
        self.fitness += self.genotype[i]
    self.fitness /= self.len
```

b) An analysis of the performance on the One-Max problem.

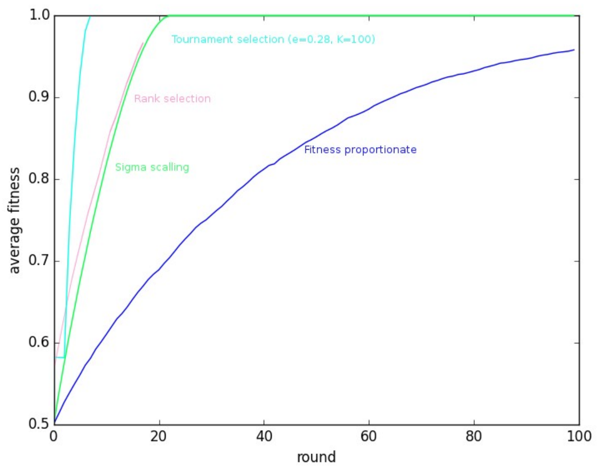
To consistently make the algorithm terminate with a fitness of 1.0 before 100 rounds, a population of 12,000 was needed to find the solution of the 40-bit One-Max problem where the adult selection protocol is full-generational replacement and the parent selection mechanism is fitness-proportionate. The most successfull settings are a crossover-rate of 1 and a mutation rate of 0.





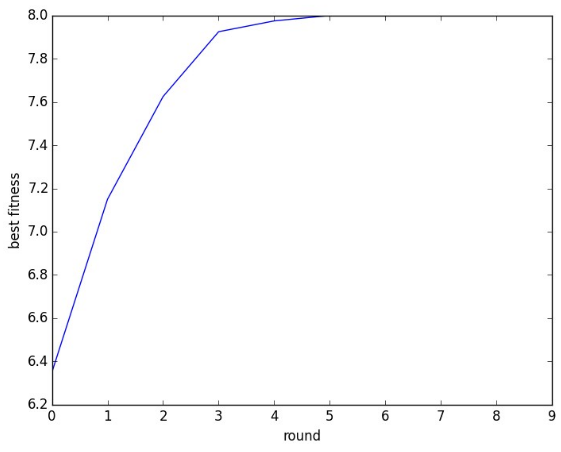
This is also shown by the graphes which show a comparison of the average and best fitness values in runs with different settings.

The next graph shows the average fitness values for runs with the same settings but different parent selection methods. The tournament selection method with parameters e=0.28, K=100 clearly showed the best result.

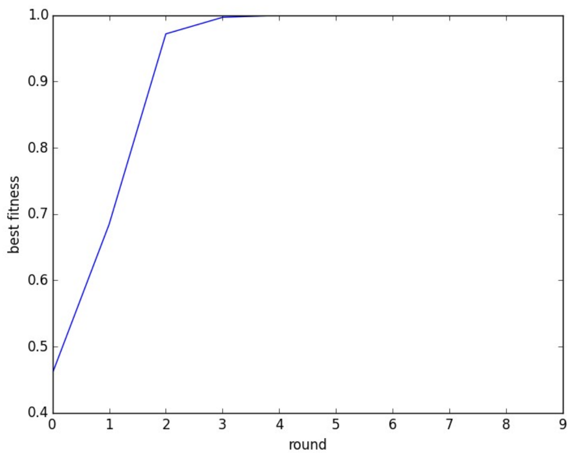


If the searched string is changed from a series of ones to a random one, the difficulty to find it is not changed. The EA gives a string a good fitness if it is close to the target. It does not matter how the target looks like. With a series of eight runs each searching for a random string and a string of ones, the EA found the string after 10.5 respectively 11.125 rounds.

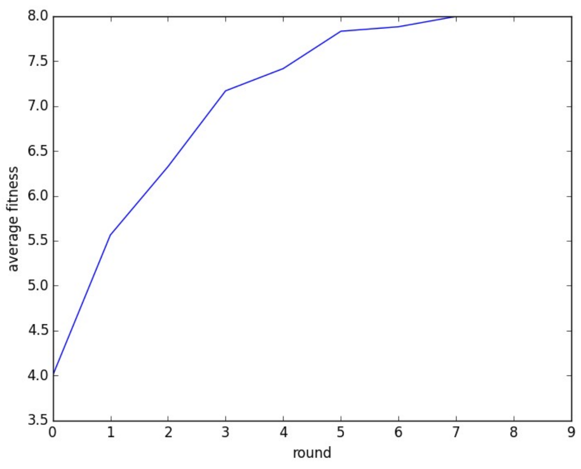
In the graphs below, the averaged results of 8 runs of the One-Max Problem and the LOLZ Problem are seen.



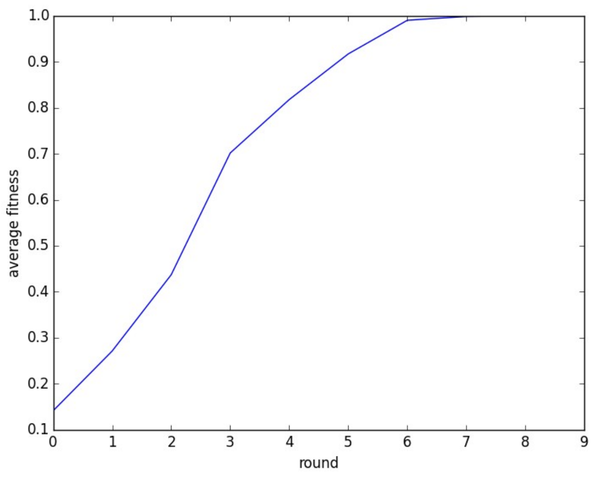
Graph 2: One-Max Problem: Best fitness



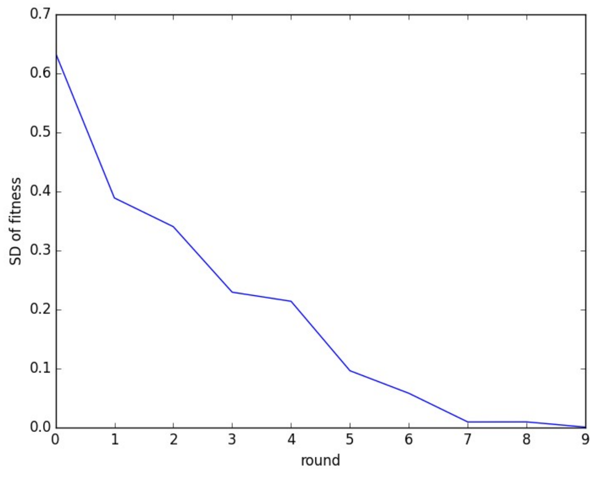
Graph 1: LOLZ-Prefix Problem: Best fitness



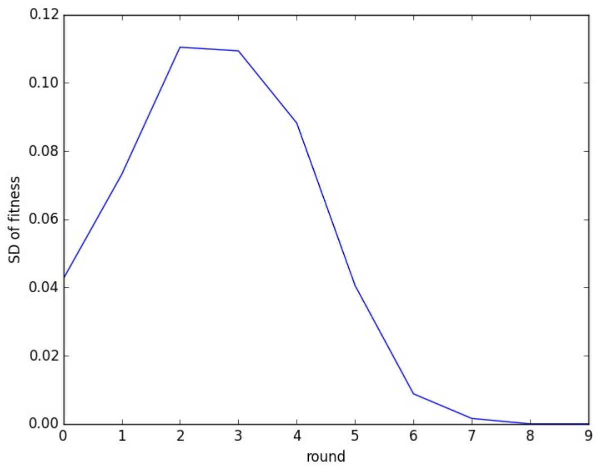
Graph 3: One-Max Problem: Average fitness



Graph 4: LOLZ-Prefix Problem: Average fitness



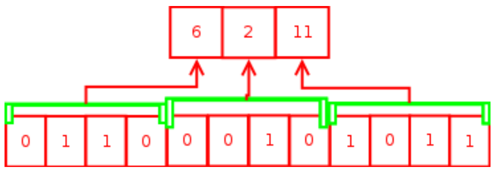
Graph 5: One-Max Problem: SD of fitness



Graph 6: LOLZ-Prefix Problem: SD of fitness

In Graph one and two, it can be seen, that the best fitness rises faster in the LOLZ-Prefix Problem, since there are more ways for a g-type to score a high fitness. It can be seen, that the standart deviation has a peak, when the average fitnesses of the population reaches 0.5, which is when the genotypes with long sequences of zeros are about to get extinct.

c) Surprising Sequences



The genotype is represented as a binary string of lenght $L \cdot \left\lceil \frac{\ln(S)}{\ln(2)} \right\rceil$. When translated to the phenotype, each $\left\lceil \frac{\ln(S)}{\ln(2)} \right\rceil$ bits are translated into one symbol. If for example a mutation generates a symbol that is bigger than S, its ones are flipped until the phenotype is valid.

The fitness function for both globally- and locally-surprising sequences is

```
self.fitness = 1.0 / (1.0 + fails)
```

where fails is the number of reoccurrences of sequences.

Best locally surprising sequences found

S	L	Population size	Rounds	Sequence
3	11	1000	1	1,1,2,2,0,0,2,1,0,1
5	27	100	54	2,3,4,4,0,1,3,0,4,1,1,4,3,1,0,0,2,2,0,3,3,2,4,2,1,2
10	70	1000	15	2,9,7,1,6,1,1,5,0,9,3,2,0,5,3,3,4,6,6,0,0,4,8,9,4,0,7,8,0,8,4,7,5,7,9,5,5,4,1,0,2,4,2,5,6,5,1,8,5,9,6,4,4,3,8,2,2,6,9,2,3,7,7,2,8,1,4,9,0,6
15	120	1000	15	5,7,14,11,11,5,10,2,3,1,7,2,0,5,5,3,8,11,1,3,0,9,4,4,1,8,0,14,4,0,10,8,13,7,7,12,8,6,3,2,11,12,0,12,9,0,0,11,13,10,9,13,13,9,14,13,11,8,5,11,10,6,0,6,7,5,8,8,4,9,7,9,6,6,11,7,6,12,11,0,4,3,13,14,1,13,8,9,3,6,14,14,9,1,0,8,12,6,10,12,12,3,11,6,9,9,12,1,10,4,12,5,0,7,13,2,6,5,2,14
20	200	1000	30	17,1,4,5,0,4,2,17,19,8,12,16,11,5,9,15,5,4,18,4,17,8,10,9,3,2,14,14,17,0,18,8,3,7,17,12,14,13,5,16,16,10,10,2,4,6,18,19,0,14,4,14,15,0,17,16,7,0,2,9,16,12,9,18,16,4,16,1,8,7,4,7,7,16,0,0,12,5,18,18,3,11,0,15,9,7,9,10,5,17,18,12,19,9,12,15,18,14,10,4,9,0,16,3,8,1,16,19,2,18,6,19,15,14,7,13,6,15,19,17,17,5,12,3,16,9,6,9,8,6,4,19,11,13,10,14,2,5,1,17,7,14,19,12,18,11,16,13,12,1,15,16,14,11,19,14,9,2,1,11,2,0,13,17,3,9,17,13,19,13,11,18,9,1,19,7,18,1,3,10,15,12,13,1,0,8,11,4,4,15,13,3,19,4,11,17,10,11,3,15

Best globally surprising sequences found

S	L	Population size	Rounds	Sequence
3	7	100	3	0,1,2,2,0,2,1
5	12	1000	2	1,2,3,0,4,0,3,3,2,1,4,2
10	24	1000	49	1,9,6,0,0,2,3,8,1,3,4,7,2,5,9,4,6,7,6,4,1,0,8,5
15	36	1000	19	2,10,12,9,14,12,8,6,4,14,15,7,3,14,5,11,13,12,2,4,9,6,11,1,2,0,13,10,8,14,7,1,7,15,6
20	42	1000	15	11,8,8,14,1,2,17,9,5,18,20,13,10,6,4,7,12,20,15,0,17,16,11,16,13,7,15,9,13,19,10,12,5,20,6,19,9,14,2,4,1,8

d) Difficulty

The LOLZ problem is harder than the simpler One-Max Problem, since the best fitness string is the same for both, but in LOLZ, a string with up to z leading zeros gives a high fitness value but is very far from the best fitness string, which contains only of ones. The Surprising Sequences Problem again is harder, because the correctness of a bit depends on the complete sequence.