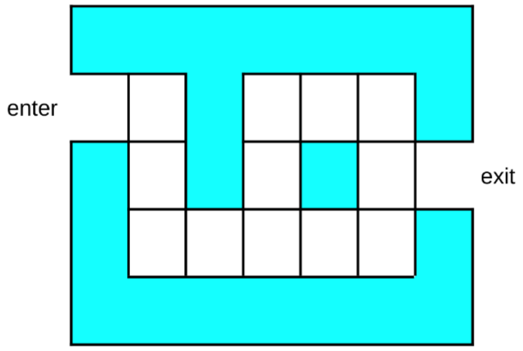
Evolutionary computation

Robot development using GA

General:
In this question, we will solve the problem of the robot. The robot can move through the maze from the starting point to the exit point. The robot can move one slot in each of the four directions (right, left, up and down). If there is a wall in the same direction, the robot will not move.

Goal:
Solve the robot's problem with genetic algorithm.



Tools:
1) Programming language – Python.
2) Evolutionary computation framework – DEAP.

Genetic algorithm parameters:

Representation	array (size = 14)
Recombination	single point crossover
Recombination probability	70%
Mutation	one-point mutation
Mutation probability	0.1%
Parent selection	tournament
Population size	100 individuals
Initialization	random
Termination condition	100 generations

Process of work:
1. From Phenotype to Genotype

To start with we need to define the genotype.

- In our task, the *phenotype* will represent the maze's map. A “W” in the maze represent a wall and “O” represent open way.
- The *genotype* will be encoded in the form of an array of size 14, but in case of big mazes it can be changed. The array consists options of robot’s move: forward, backward, left and right. Indexes in the array are step's numbers.

2. Initialization

Initialization of the initial *population* - at the first step, a set of candidates in the amount of 100 is generated for the solution, which gives new offspring as a result of the selection, mutation, and crossover operators.

3. Fitness Evaluation

The next step is the evaluation of candidates.

- To evaluate the fitness parameter, we will create a function (*evaluation()*) that will estimate how close the robot approached the exit, given the number of collisions with the wall. The lower this measure, the better a phenotype and zero is the best solution (optimal value).
- In case the robot moves to a wall, its fitness was decrease by adding negative value to is fitness. His location stays as before.
- In the fitness function, we take into account the number of steps the robot took and add the number of collisions with the wall (penalty), then multiply the resulting value by the remaining distance to the finish.

4. Parent selection

The selection operator selects representatives for the use of genetic operators. For selection, we use *Tournament selection*. Selected candidates are then passed on to the next generation. In a 3-way tournament selection, we select 3-individuals and run a tournament among them. Only the fittest candidate amongst those selected candidates is chosen and is passed on to the next generation. In this way many such tournaments take place and we have our final selection of candidates who move on to the next generation.

5. Crossover

We use *one-point crossover*, in this crossover, a random crossover point is selected and the tails of its two parents are swapped to get new offsprings. Resulting in two new representatives, with the genes of the first and second parent.

6. Mutation

The next step is the mutation operator (*one_point_mut*()). One gene of a random representative is mutated.

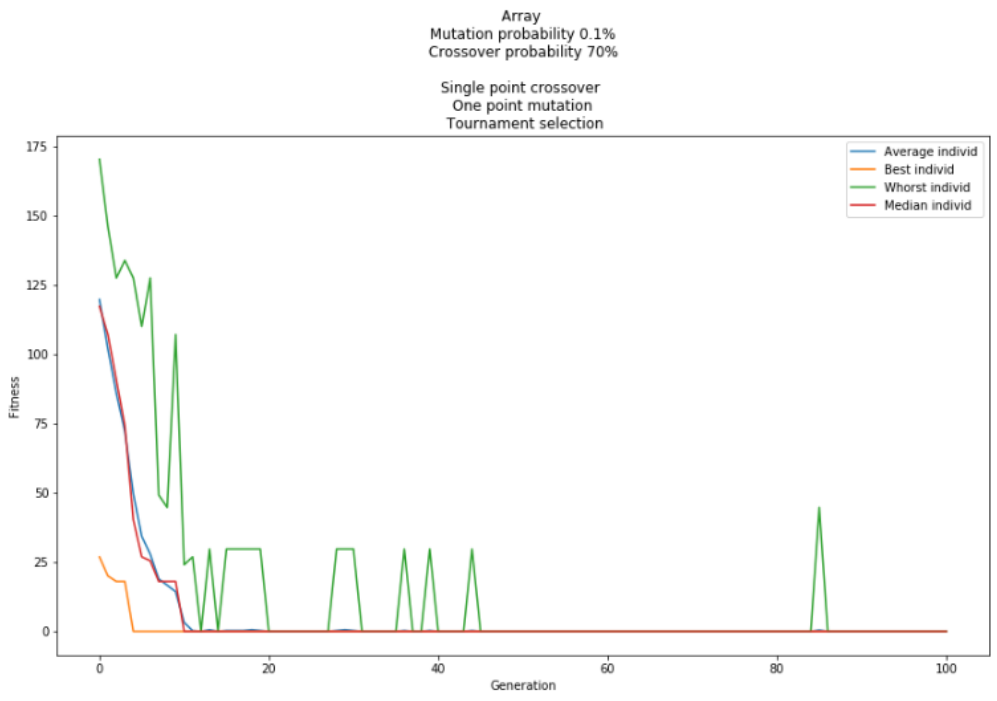
7. Final

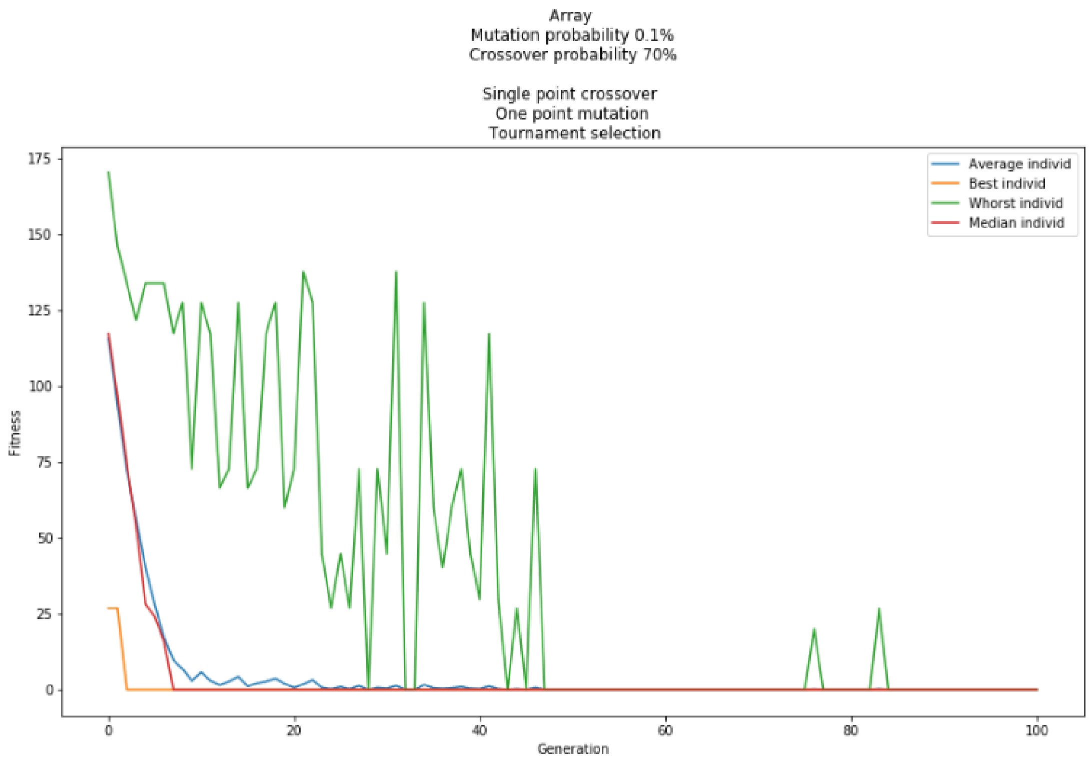
The final stage of each iteration is the formation of a new population. Stop occurs on the hundredth generation.

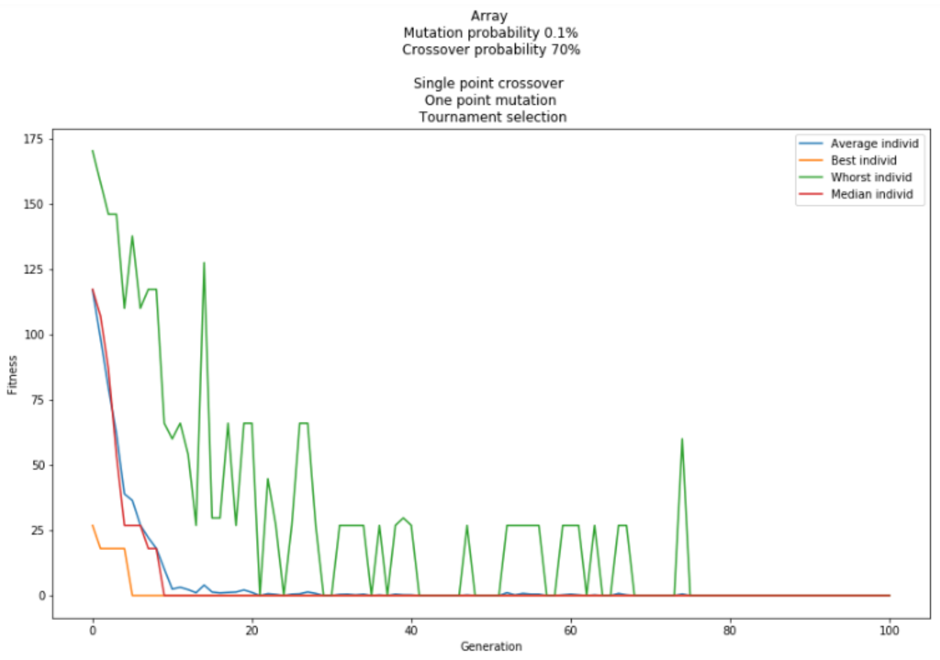
Results:

During this experiment, we made three runs:

Average time for termination: 0.76s



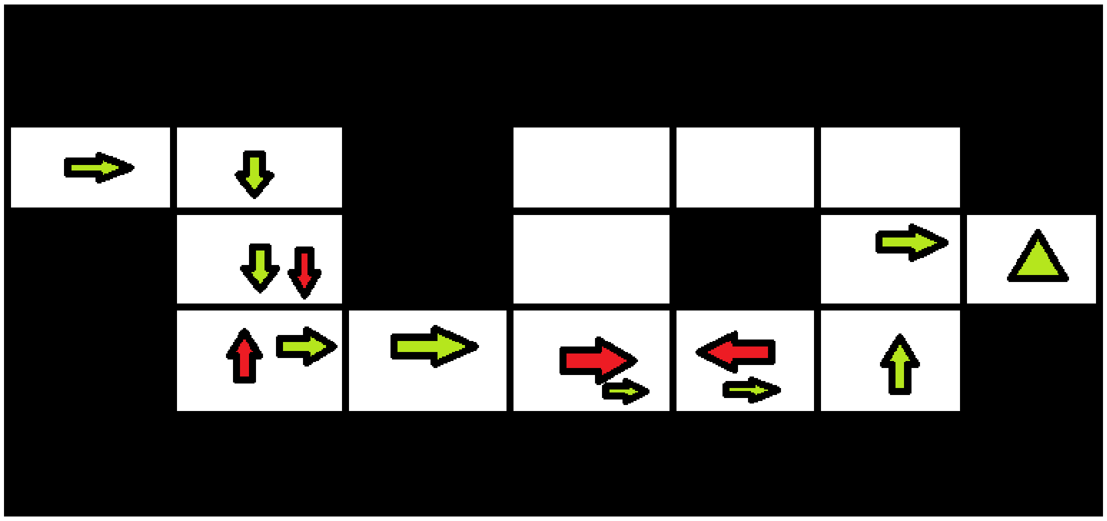




Conclusions:

- We tested the experiment also with other parameters GA, namely, other values of the probabilities of mutation and crossover. We chose the best option and entered it in the table.
- In all three runs, the correct solutions were found - which can be seen on the graph as convergence to a suitability value of 0.
- The algorithm finished with the optimal path quite fast.
- As we see after the 20th generation, the entire population becomes approximately the same (the same fitness value). Individuals come to one certain value of fitness function, which does not change. In the future, a mutation of only one gene with a 0.1% probability of a mutation cannot lead to strong changes in the population as a whole, only in point individuals does fitness deteriorate. The crossover in this case does not affect, since the entire population is the same.
- The optimal solution that the algorithm found:

['right', 'down', 'down', 'up', 'down', 'right', 'right', 'right', 'left', 'right', 'right', 'up', 'right', 'u
p']



- The developed genetic algorithm is a powerful tool for solving the robot problem and can be applied on a larger maze.

Future Work:

- In this work, our task was to reach the exit from the maze. In future work, we propose to optimize the fitness function to find the finish in a minimum step's number.