INTRODUCTION TO ARTIFICIAL INTELLIGENCE

EXERCISE 2

Full names: Andreas Kalavas, Panagiota Chita

Emails: andreas.kalavas.stud@pw.edu.pl , panagiota.chita.stud@pw.edu.pl

Introduction:

In this report, we show our work on genetic algorithms, when used to optimize functions. Specifically, our algorithm tries to maximize the function $F(x) = x^T Ax + b^T x + c,$ where c is a constant, b is a d-dimensional vector, A is a dxd matrix, and x (the free variable) is again a d-dimensional vector. In our implementation, each iteration of the algorithm calls four functions, each one doing a different job as described further in the report. But first, we will analyse the user input format, and the population initialization and representation.

Input:

(Calling the def user_input())

The user defines the dimensionality, gives the constants of the function F, the search area, the population size, the crossover and mutation probabilities, as well as the number of the iterations, and the number of individuals to select to keep in each iteration as the best (depending on this number, there may be more individuals that stay the same).

For many of these parameters, some values are suggested, and every parameter is checked that is correct, in case of wrong input (for example negative probability), otherwise the user is asked to give another value until the input is acceptable.

Population initialization and representation:

The initialization is done randomly; as the genetic algorithms uses binary vectors, series of zeros and ones are generated randomly, with equal probability, and stored. The method of representation of the integer numbers we used is the signed binary; the first bit of the number represents the sign (0 for +, 1 for -). This is because we believe it behaves better when it comes to the crossover process (for example if we used the 2 complement representation, and tried to crossover 11111 with 00001 ,which represent -1 and 1, we could get 11001 and 00111 ,which represent -7 and 7, which is bad if we want the algorithm to converge for example at 0). In addition, we use later the fuctnion def ma2vec in order to decode the bits sequence into decimal.

The algorithm:

As mentioned before, in each iteration of the algorithm, four functions are called. These are discussed below:

1. **Function fitvalues:**

   This function takes as a parameter the current population, and returns the values of F(x), of the individuals of the population (which are also the fitness values, as we want to maximize the function), and also the scaled-normalized values.

   The function also calls function func (*def func)* which calculates the value for a given point (as binary vectors), and also ma2ve (*def ma2ve)* is called to transform the matrix of zeros and ones which represent a number, to a vector of integer numbers.

2. **Function roulette_wheel_selection:**

   This function selects some individuals with the roulette wheel selection process, and replaces them in the population at the places of the individuals that have been the longest in the population, because of the FIFO replacement policy (this is done with the use of an offset).

3. **Function crossover:**

   This function takes as parents the individuals that were selected in the roulette wheel selection process, and with a probability given from the user crossovers them, with the single-point procedure. The resulted children are added to the population, again with the FIFO policy. For every parent pair, we do an independent probability test to decide whether we will crossover them.

4. **Function mutation:**

   This function takes as parents again individuals selected in the roulette wheel selection process, and with a probability given from the user creates mutated children, which are again added to the population, with the FIFO policy. For every parent, we do an independent probability test to decide whether we will create a mutated child.

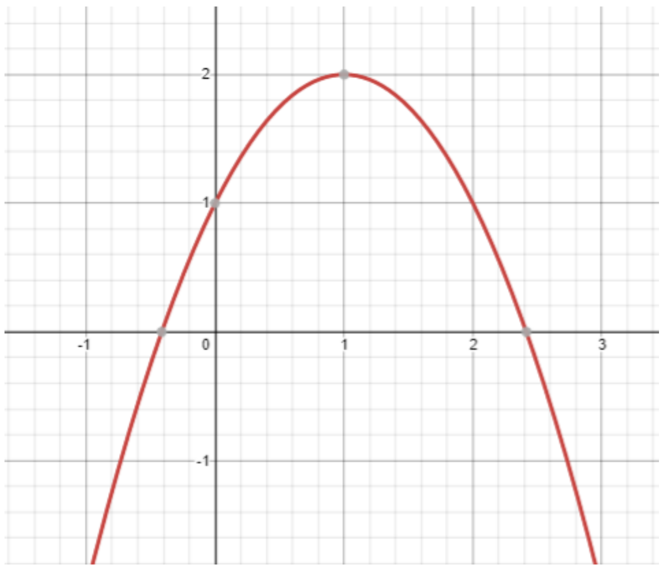
Output:

At the end, our program prints the individuals, their F(x) values, the best value and mean of values of the last population, as well as the best value of all the populations. On the next pages, a couple of test cases are presented.

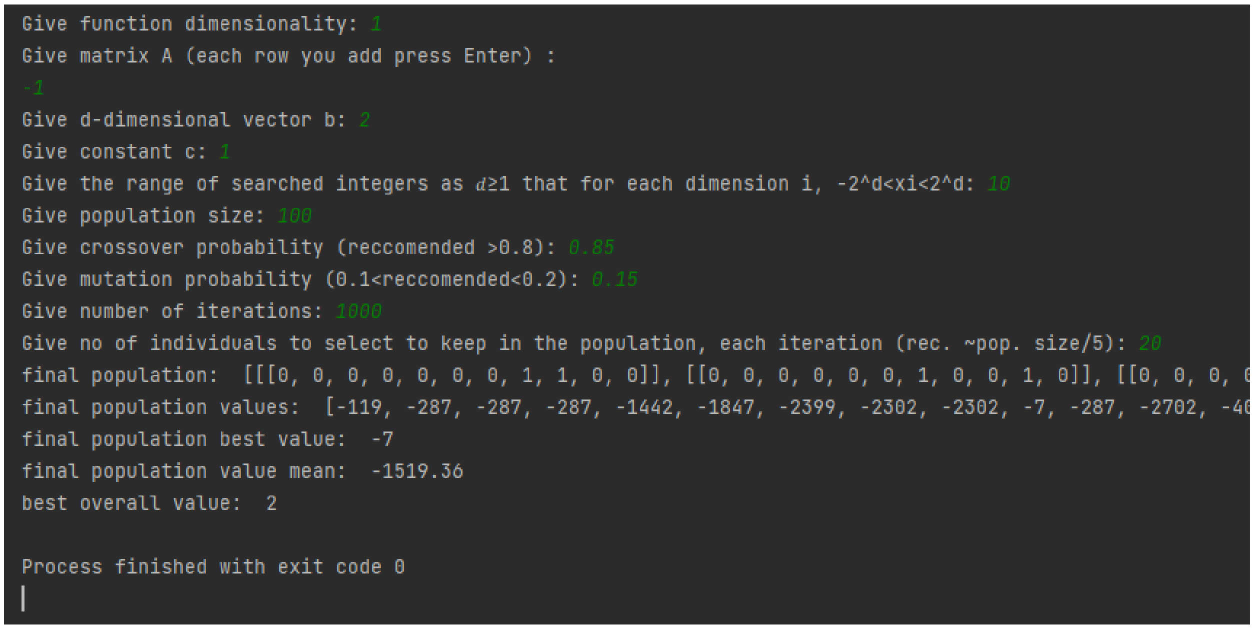
Test 1:

$$F(x) =- x^2 + 2x + 1$$

The function is shown on the picture. As we can see, the maximum is observed at the point x=1, and its value is F(1)=2.



Below we can see the terminal, as we run the algorithm. The algorithm managed to find the maximum, but the maximum is not in the final population.



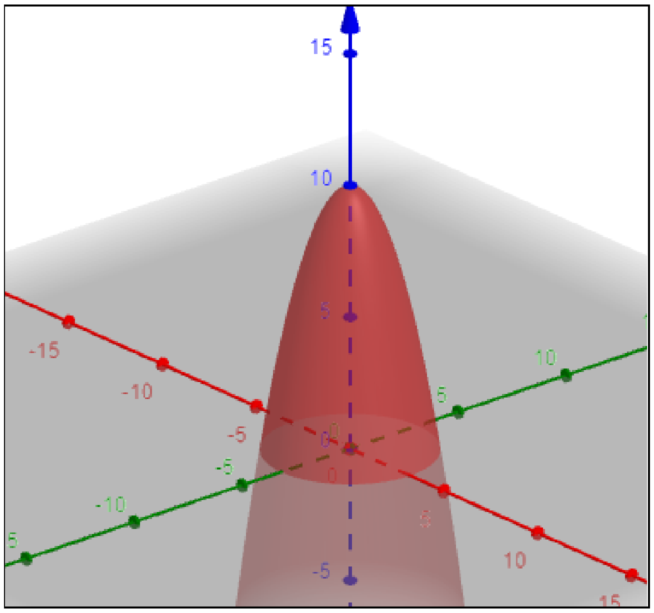
```
Give function dimensionality: 1
Give matrix A (each row you add press Enter) :
-1
Give d-dimensional vector b: 2
Give constant c: 1
Give the range of searched integers as d≥1 that for each dimension i, -2^d<xi<2^d: 10
Give population size: 100
Give crossover probability (reccomended >0.8): 0.85
Give mutation probability (0.1<reccomended<0.2): 0.15
Give number of iterations: 1000
Give no of individuals to select to keep in the population, each iteration (rec. ~pop. size/5): 20
final population:  [[[0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0]], [[0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0]], [[0, 0, 0, 0
final population values:  [-119, -287, -287, -287, -1442, -1847, -2399, -2302, -2302, -7, -287, -2702, -40
final population best value:  -7
final population value mean:  -1519.36
best overall value:  2

Process finished with exit code 0
```

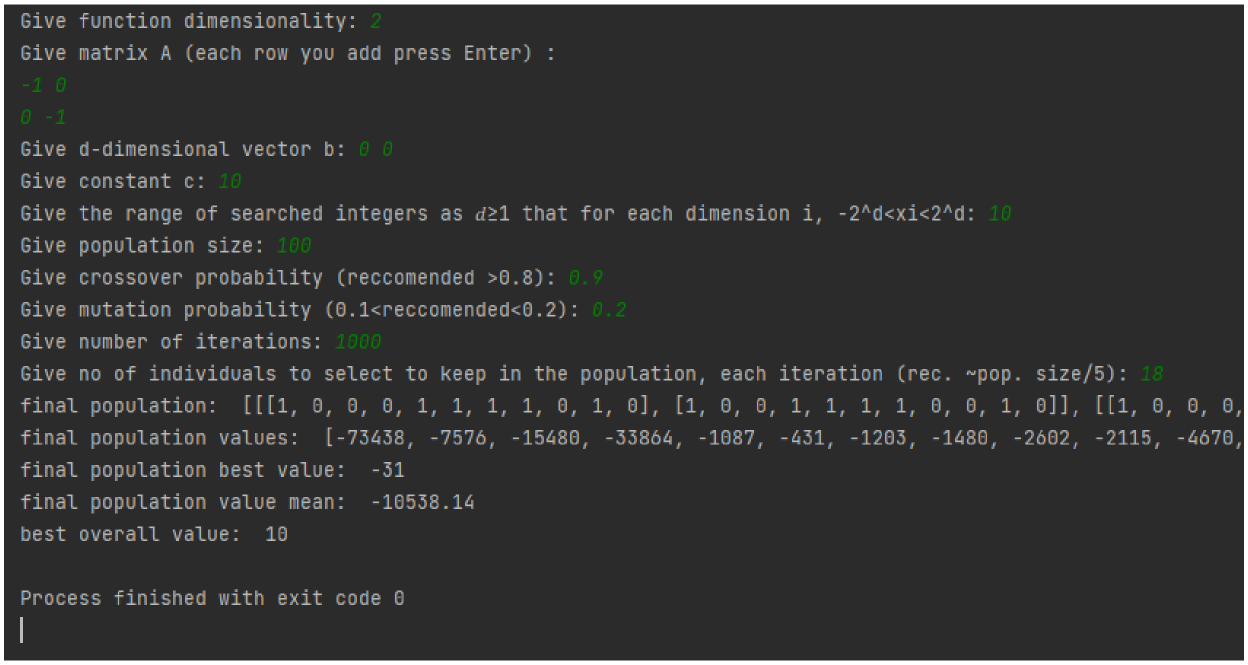
Test 2:

$$F(x) = 10 - x^2 - y^2 = x^T \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \end{bmatrix} x + 10$$

The function is shown on the picture. As we can see, the maximum is observed at the point x=(0,0), and its value is F(0,0)=10.

Below we can see the terminal, as we run the algorithm. The algorithm managed to find the maximum, but again the maximum is not in the final population.





```
Give function dimensionality: 2
Give matrix A (each row you add press Enter) :
-1 0
0 -1
Give d-dimensional vector b: 0 0
Give constant c: 10
Give the range of searched integers as d≥1 that for each dimension i, -2^d<xi<2^d: 10
Give population size: 100
Give crossover probability (reccomended >0.8): 0.9
Give mutation probability (0.1<reccomended<0.2): 0.2
Give number of iterations: 1000
Give no of individuals to select to keep in the population, each iteration (rec. ~pop. size/5): 18
final population:  [[[1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0], [1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0]], [[1, 0, 0, 0,
final population values:  [-73438, -7576, -15480, -33864, -1087, -431, -1203, -1480, -2602, -2115, -4670,
final population best value:  -31
final population value mean:  -10538.14
best overall value:  10

Process finished with exit code 0
|
```

Notes:

Genetic algortihms are algorithms used in optimization problems and they are based in the principles of Biological Evolution. Main elements of these algorithms are population, chromosomes and genes. The size of the population is an important factor, however, it is not always decisive and the proper size of a population is yet to be determined by the scientific society. We should bare in mind that with the increase of the population, the memory and computational needs are increasing as well .The options vary and some support that the diversity of the population is equally important, if not more. We have also noticed that as the iterations of the algorithm increase, the closer to our solution we reach. In general, the genetic algorithm uses tehcniques, such as crossover and mutation, in order to avoid being trapped in a local minimum/maximum and to increase the diversity and the search area of the algorithm.