Wrocław University of Science and Technology

Faculty Of Electronics

Computer Science

Internet Engineering

# Softcomputing

Genetic Algorithms for maxima search of multimodal functions

Authors:
Bogusław Matysik
Paweł Zaborowski

## Spis treści

**1 Introduction** **2**
1.1 Function . . . . . . 2

**2 Theory** **2**
2.1 Genetic algorithm in general . . . . . . 2

**3 Implementation** **4**
3.1 Fitness function . . . . . . 4
3.2 Selection function . . . . . . 4
3.3 Reproduction function . . . . . . 5
3.4 Termination function . . . . . . 6
3.5 Technology . . . . . . 7

**4 Experiments** **7**
4.1 First function . . . . . . 7
4.2 Second function . . . . . . 7

**5 Plots** **8**
5.1 First function . . . . . . 8
5.2 Second function . . . . . . 10

**6 Results & conclusions** **10**

# 1 Introduction

## 1.1 Function

In our project we chosen two different functions. One more complex than the second one. We compared accuracy of the results of the both of them. They are more wisely described in the next sections. Both of these functions were tested on the same variables' restrictions. It is worth mentioning that genetic algorithm is heuristic algorithm. We cannot be sure about the result since they are not perfectly exact. Every time it is some kind of estimation.

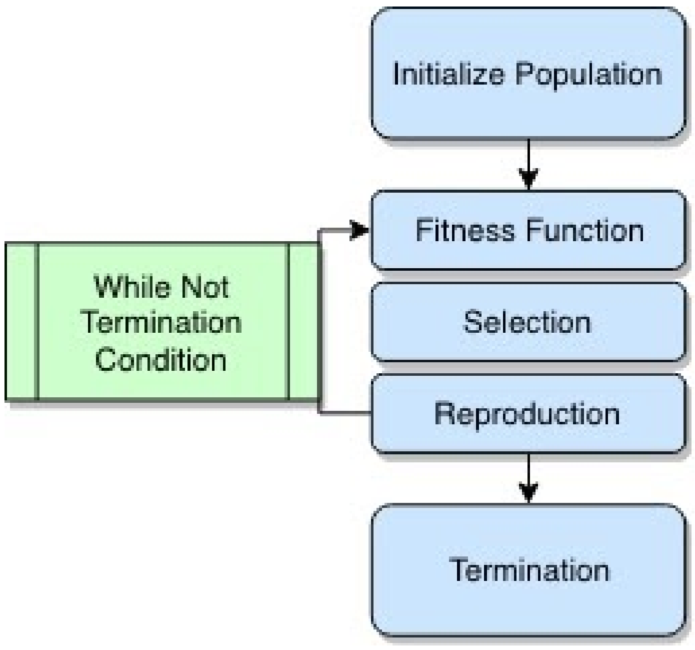
# 2 Theory

## 2.1 Genetic algorithm in general

Common characteristic of evolution algorithm which distinguish them from other traditional optimization methods are:

- usage of the genetic operators that are adapted to the form of solutions
- processing a population of solutions leading to a parallel search of solution spaces from different points
- quality of current solutions is enough to direct solution process in proper way
- purposeful introduction of random elements

Algorithm process:



The most commonly used chromosome coding:

- a vector of genes each of which can be a single or multi-bit integer or a real number
- using tree data structures

For each population we need to calculate probability of being selected p:

$$p_s = \frac{f(x)}{\sum f(x)}$$

Chromosomes with the best value of fitness function have greater chance to be chosen.

Methods of selection:

- Roulette wheel method
- Ranking list
- Tournament

Crossing's aim is to concatenate, in different combinations, characteristics coming from different individuals from population. "Child" has set of characteristics that are combination of parents' characteristics.

Mutation is about increasing variety of individuals. It introduces to genotype random changes. In that way we can protect of premature convergence of the algorithm.

Crossover operation

## 3 Implementation

In this paragraph we described each of the stage of our implementation of genetic algorithm. We tested our results with different amount of initial population size and most of all iterations of algorithm. At the begging we started with initial population size equals 100 and 100 of iterations.

The algorithm begins by initializing a population of individuals using default values. Then, it runs each member of that population through a fitness function. After that it selects the fittest members of the population to reproduce using a method defined in the reproduction function. The evaluation and reproduction is repeated until a desired number of iterations have passed. At end, the algorithm presents the best member or members of the population according to the fitness function.

### 3.1 Fitness function

We prepared two functions:

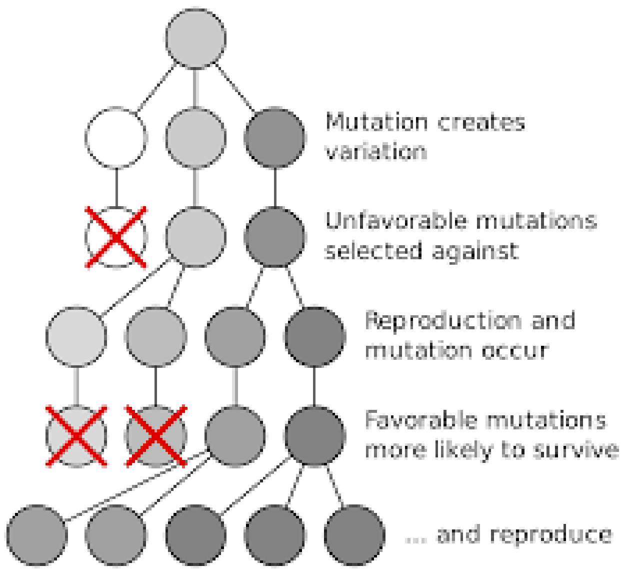
1.
$$20 * x^2 + y^2 - 10 * (cos(2 * \pi * x) + sin(2 * \pi * y)$$

2.
$$sin(3 * x) * cos(3 * y)/10$$

The variable restriction for both of them is set to <-1;1>. The aim is to maximize them, so found best input.

### 3.2 Selection function

The selection function takes the population and the results of the fitness function to determine who should reproduce in the next iteration. The selection function calculates a threshold from scores from fitness function. For example, it may calculate the average score and only keep the top half of the population as in our case, because we want to find maximum value. It passes the selected population and average score into the reproduction function and deletes the rejected individuals.


Mutation creates variation
Unfavorable mutations selected against
Reproduction and mutation occur
Favorable mutations more likely to survive
... and reproduce

```
def evaluate_generation(population):
        scores = []
        total = 0
        for individual in population:
                r = fitness_function(individual[0], individual[1])
                scores.append(r)
                total += r
        else:
                print("error: Wrong number of arguments received")
        avg = total / len(scores)
        return scores, avg
```

indivudal[] is two item array because we have two variables in our fitness function.

## 3.3 Reproduction function

The reproduction function determines how to expand the population based on the existing members. Modifying the behavior and hyperparameters of the reproduction function. The simplest reproduction method is mutation, where each new member is based on a single individual and this is the method we used in out algorithm.

Besides that, crossover is a more complex method of reproduction, where each new individual is based on some combination of existing individuals. Crossover still mutates the new organism's attributes, but does so by applying a function of multiple organisms' attributes.

In our process we decided to confine just with mutation method.

```
def mutate(individual):
        new = []
        for attribute in individual:
                new.append(attribute +
                        random.normalvariate(0, attribute + .1))
# Random factor of normal distribution
        return new
```

It modifies each attribute of an individual by a random factor. In particular, we use a normal variate distribution that has a higher probability of a smaller modification as it has a mean near zero. When creating new individuals, it is important to consider how different they should be from the parents. If the difference is too small, the algorithm will require many iterations to reach a max, but if the difference is too big, the output will be imprecise and may miss the maximum.

## 3.4 Termination function

After all of the iterations have run, the termination function takes the final population and evaluates it on the fitness function to return the best individual(score). In this case, the best individual is the estimated maximum value of the function and its attributes are the estimated optimal inputs to given function. The inputs may be greater or less than the actual best inputs, but the estimated maximum value will by definition be less than or equal to the true global maximum.

```
def find_best(population):
        best = None
        val = None
        for individual in population:
                r = fitness_function(individual[0], individual[1])
                try:
                        if r > val:
```

```
                best = individual
                val = r
        except:
            best = individual
            val = r
    return best, val
```

Running the simpler function on 100 individuals with 100 iterations got us within one thousand of the best answer almost instantly.

### 3.5 Technology

Technologies we used to implement algorithm:

- python 3.6
- libraries: numpy, math, matplotlib, mpl_toolkits

To compare out results we used WolframAlpha tool.

## 4 Experiments

### 4.1 First function

$$20 * x^2 + y^2 - 10 * (cos(2 * \pi * x) + sin(2 * \pi * y)$$

| | |
|---|---|
| population: 100 | |
| 100 | The optimal input is [5.847e+90, -0.1] with a value of 6.838e+182 |
| 150 | The optimal input is [-2.232e+126, -0.1] with a value of 9.971e+253 |
| 170 | The optimal input is [8.163e+138, -0.1] with a value of 1.332e+279 |

| | |
|---|---|
| population: 120 | |
| 100 | |
| 150 | The optimal input is [-9.071e+92, -0.1] with a value of 1.646e+187 |
| 170 | |

the bigger population the bigger scores to choose.

### 4.2 Second function

$$sin(3 * x) * cos(3 * y)/10$$

population: 150

| | |
|---|---|
| 100 | The optimal input is [3.074e+97, -0.1] with a value of 1.890e+196 |
| 150 | The optimal input is [-4.447e+140, -0.1] with a value of 3.955e+282 |
| 170 | |

population: 100

| | |
|---|---|
| 100 | The optimal input is [0.541, 0.012] with a value of 0.333 |
| 150 | The optimal input is [0.537, 0.002] with a value of 0.333 |
| 170 | The optimal input is [0.513, -0.012] with a value of 0.333 |

population: 120

| | |
|---|---|
| 100 | The optimal input is [0.524, 0.002] with a value of 0.333 |
| 150 | The optimal input is [0.512, 0.006] with a value of 0.333 |
| 170 | |

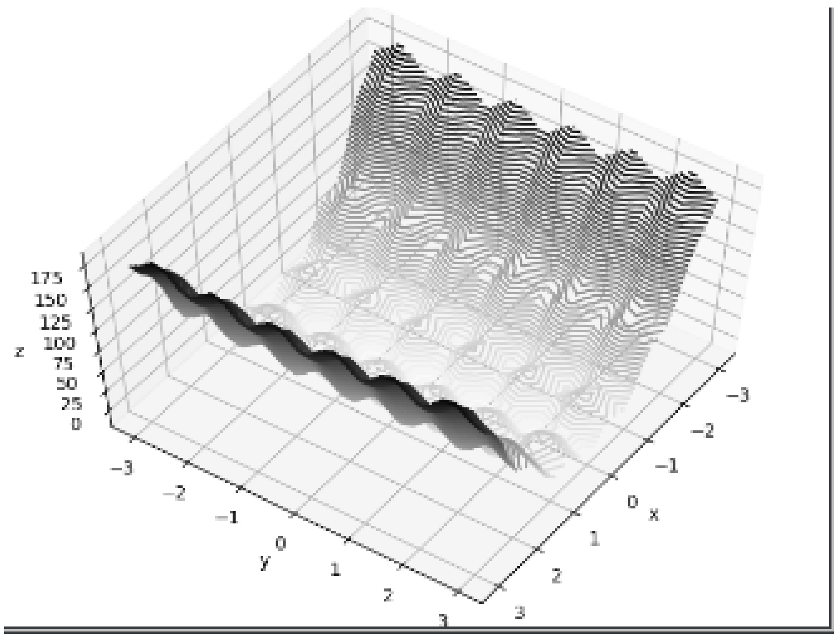
population: 150

| | |
|---|---|
| 100 | The optimal input is [0.512, 0.006] with a value of 0.333 |
| 150 | The optimal input is [0.516, -0.014] with a value of 0.333 |
| 170 | The optimal input is [0.527, 0.001] with a value of 0.333 |

## 5 Plots

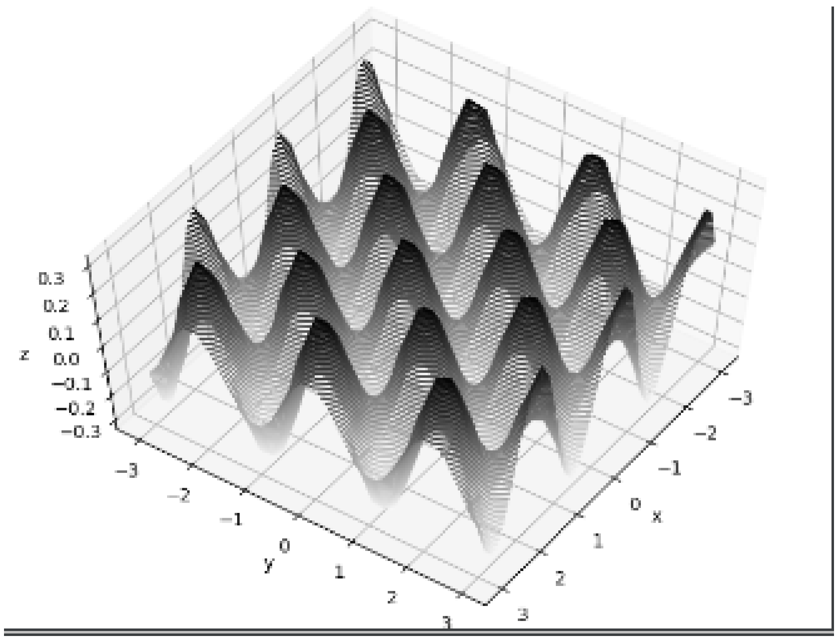
### 5.1 First function

$$20 * x^2 + y^2 - 10 * (cos(2 * \pi * x) + sin(2 * \pi * y)$$


175
150
125
100
75
50
25
0
z
−3
−2
−1
0
1
2
3
y
−3
−2
−1
0
1
2
3
x

### 5.2 Second function

$$sin(3 * x) * cos(3 * y)/10$$



## 6 Results & conclusions

To compare our results with the correct ones we used scipy.optimize library. In scipy.optimize.fmin() function we put our function in parameter, with two variables and then negate it we get maximum:

```
max = opt.fmin(lambda vec: −fitness_function(vec[0], vec[1]), [−1.0, 1
```

This lib uses the downhill simplex algorithm.

Results:

- function 1):
  Current function value: 0.333333
  [-0.52363581, 1.04718372]

- function 2):
  Current function value: 0.333333
  [-0.52363581, 1.04718372]