Softcomputing
Genetic Algorithms for maxima search of multimodal functions - Report

Krystian Horecki 181079
Antoni Buszta 181013
Wroclaw University of Technology

17.01.2014 r.

Contents

1. Project description

1.1. Project goals

The goal of the project was to create genetic algorithm which will give possibility to find more than one local extremums of the function in case of multi modal functions. The main problem in this case is to find version of the algorithm which unlike normal genetic algorithm is able to find more than one result. There exists mechanisms which provide such functionality and some of them were used and will be described later in this paper. Partially the goal was also to check how used genetic algorithm parameters affects results of the computations especially in case of multi modal functions and multi modal optimization process.

1.2. Used technologies

To develop solution Python programming language was used with additional library called DEAP[1] which allows creation of genetic algorithms. All charts included in this paper were generated with Python library called pyGal[2].

1.3. Examined function

To determine utility of created solution and test quality of algorithm there was used the Shekel miltimodal function. Sheckel function is very useful for this kind of tests because it allows to have the number of maxima is given by the length of any of the arguments $\mathcal{A}$ or $\mathbf{C}$, $\mathcal{A}$ is a matrix of size M $\times$ N , where M is the number of maxima and N the number of dimensions and c is a M $\times$ 1 vector. The matrix $\mathcal{A}$ can be seen as the position of the maxima and the vector $\mathbf{C}$, the width of the maxima. Function equation look as follows:

$$f_{\text{Shekel}}(\mathbf{x}) = \sum_{i=1}^{M} \frac{1}{c_i + \sum_{j=1}^{N}(x_j - a_{ij})^2}$$

To check correctness of algorithm following matrix $\mathcal{A}$ was used:

$$\mathcal{A} = \begin{bmatrix} 0.5 & 0.5 & \\ & 0.25 & 0.25 \\ & 0.25 & 0.75 \\ & 0.75 & 0.25 \\ & 0.75 & 0.75 \\ & 0.35 & 0.35 \\ & 1.0 & 1.3 \\ & 1.4 & 0.2 \\ & 0.34 & 1.7 \\ & 1.5 & 1.3 \\ & 1.1 & 0.9 \\ & 0.9 & 1.7 \end{bmatrix} \text{ and } \mathbf{C} = \begin{bmatrix} 0.002 \\ 0.005 \\ 0.004 \\ 0.003 \\ 0.002 \\ 0.001 \\ 0.002 \\ 0.0023 \\ 0.0035 \\ 0.005 \\ 0.0055 \\ 0.0022 \end{bmatrix}, \text{ thus defining 12 maximums in } \mathbf{R}^2.$$

Function presented in above equation with given input can be also represented in graphical way which was presented on Figure 1.

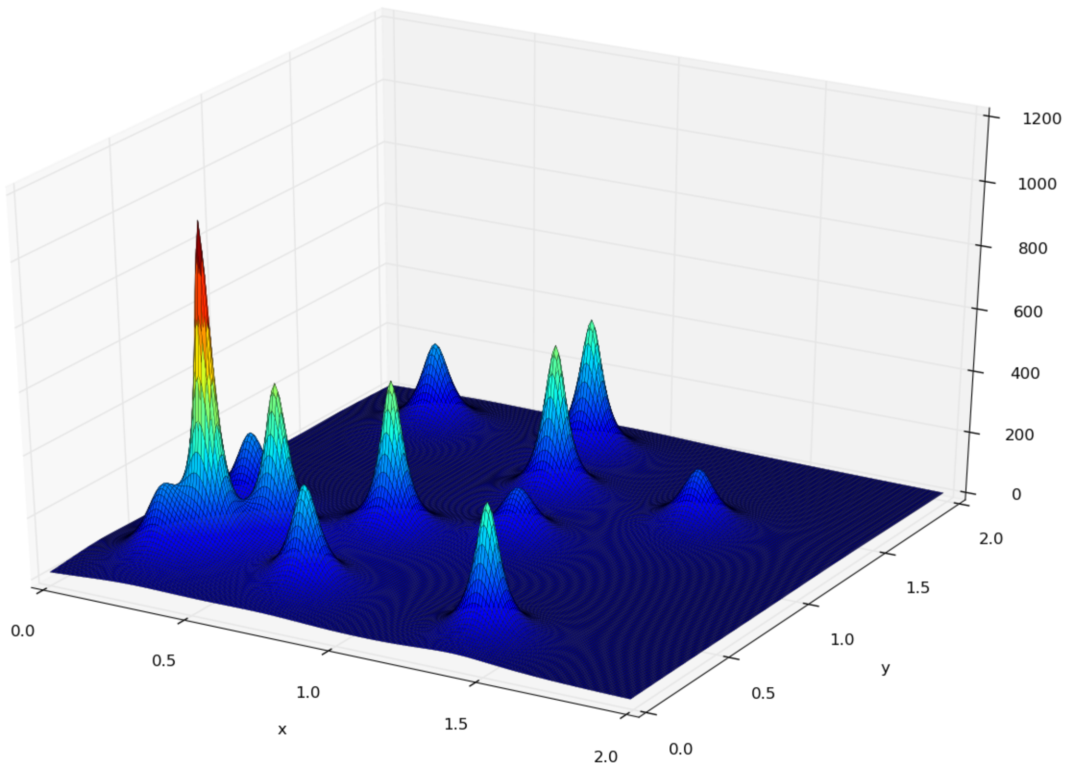

Figure 1: Chart of Scheckel function with used A and C input data.

The reson that this kind of function with those parameters was used was to have benchmark which will contain many extremums with different values are scattered in $\mathbf{R}^2$.

2. Genetic algorithm

2.1. Description

To be able to maintain more than one stable solution and find more than one local extremum it was required to develop modified genetic algorithm. Genetic algorithm was develop with niching[3, 4] mechanism which allows to maintain subpopulations of individuals in many niches which gave more than one best solution.

2.2. Data representation

To represent individuals in population a vector with two float values was used. This vector represents point in $\mathbf{R}^2$ and define one individual. As a population the vector of individuals is used with length according to population size. Population is generated by creation of random coordinates values for each individual which are within given area of examined space, this action is repeated to create as many individuals as it is required. Whole population is later divided into number of subpopulations by choosing and taking random individuals.

2.3. Niching method

We used niching mechanism using fitness sharing method which simulate situation when in one niche is only limited number of resources which can be used by individuals. Individuals which are occupying the same niche have to share the resources and in case of having too less of them individuals are moving to another niche instead of focusing in only one point. To create genetic algorithm with niching it is necessary to define sharing radius ϕ_{share} which defines niche size. The sharing function is defined as:

$$sh(d_{ij}) = \begin{cases} 1 - (\frac{d_{ij}}{\phi_{share}})^{\alpha}, & \text{if } d_{ij} < \phi_{share} \\ 0, & \text{otherwise} \end{cases}$$

To calculate ϕ_{share} the following equation was used:

$$\phi_{share} = \frac{\sqrt{2 * |range_{max} - range_{min}|}}{radiusfactor}$$

, where radius factor is parameter given by user. To calculate shared fitness value equation is defined as:

$$f_{share}(i) = \frac{f_{raw}(i)}{\sum_{i=1}^{\pi} sh(d_{ij})}$$

,
where π is the representatives vector size. Representatives which are used to calculate fitness value are taken from every other subpopulation than currently examined individual and each of them is the best individual from its subpopulation.

2.4. Mutation method

Mutation is performed by adding random mutationΔ value to x and y coordinate. Value of mutationΔ for each coordinate is calculated with equation defined as

$$mutation\Delta = random(-80, 80) * (range_{max} - range_{min}) * 0.0001$$

Later value of new coordinates for given individual is calculated with equation defined as

$$individual[x] = \begin{cases} individual[x] + mutation\Delta, & \text{if individual[x] + mutation}\Delta <= range_{max} \text{ or } >= range_{min} \\ range_{max}, & \text{if individual[x] + mutation}\Delta > range_{max} \\ range_{min}, & \text{if individual[x] + mutation}\Delta < range_{min} \end{cases}$$

$$individual[y] = \begin{cases} individual[y] + mutation\Delta, & \text{if individual[y] + mutation}\Delta <= range_{max} \text{ or } >= range_{min} \\ range_{max}, & \text{if individual[y] + mutation}\Delta > range_{max} \\ range_{min}, & \text{if individual[y] + mutation}\Delta < range_{min} \end{cases}$$

2.5. Crossing method

The method of crossing which was used is simply setting y coordinate from first individual to second one and vice versa. This method can be described with equation defined as

$$\begin{aligned}
individual_{first_{new}}[x] &= individual_{first}[x] \\
individual_{first_{new}}[y] &= individual_{second}[y] \\
individual_{second_{new}}[x] &= individual_{second}[x] \\
individual_{second_{new}}[y] &= individual_{first}[y]
\end{aligned}$$

2.6. Evaluation method

During the evaluation of given individual 3 values are taken into account and those are:

- value of function in given point,
- absolute value of derivative of function on X axis in given point,
- absolute value of derivative of function on Y axis in given point.

Having listed above parameters it is possible to determine how much given individual fit to maximum definition and promote it or not during selection. Later those evaluation results are used during selection which is performed by tournament method. During tournament random 3 individuals are used during each tournament and later as a result of selection, new population of individuals is returned.

2.7. Result analysis

To analyze results gained from the algorithm vectors of real function extremums points and gained results found by algorithm were sorted by value of the function in given point. Later by iteration through both vectors till end of smaller of them, the absolute error value was calculated by differing real extremum value with calculated one. As a quality metric the average error value was used.

3. Performed tests

3.1. Description

To check the best set of parameters which gives the best results of searching multiple maxima several tests were performed. Following tests were performed:

- tests of average error value for different number of generations,
- tests of average error value for different population size,
- tests of average error value for different subpopulations number,
- tests of average error value for different crossing probability,
- tests of average error value for different mutation probability,
- tests of average error value for different ray distance factor.

In case of each testcase the default set of parameters was used with following parameters:

Maximum X value	2.0
Maximum Y value	2.0
Minimum Y value	2.0
Minimum Y value	2.0
Population size	150
Subpopulation number	12
Crossing probability	0.6
Mutation probability	0.1
Ray distance factor	25
Number of generations	200

To make sure that gained results are accurate test for each parameter value was performed 5 times and average value was taken as a result.

3.2. Number of generations test

In order to check how average error value changes along with number of generations the testcase was prepared, where number of generations is increased by 5 from 5 to 100. Figure 2 presents result of the test. It can be clearly seen that average error is going down with increasing number of generations but in case of values bigger than 50, there is no significant improvement. This can be caused by limited resolution of mutationΔ value and function containing maximums in form of high peaks.

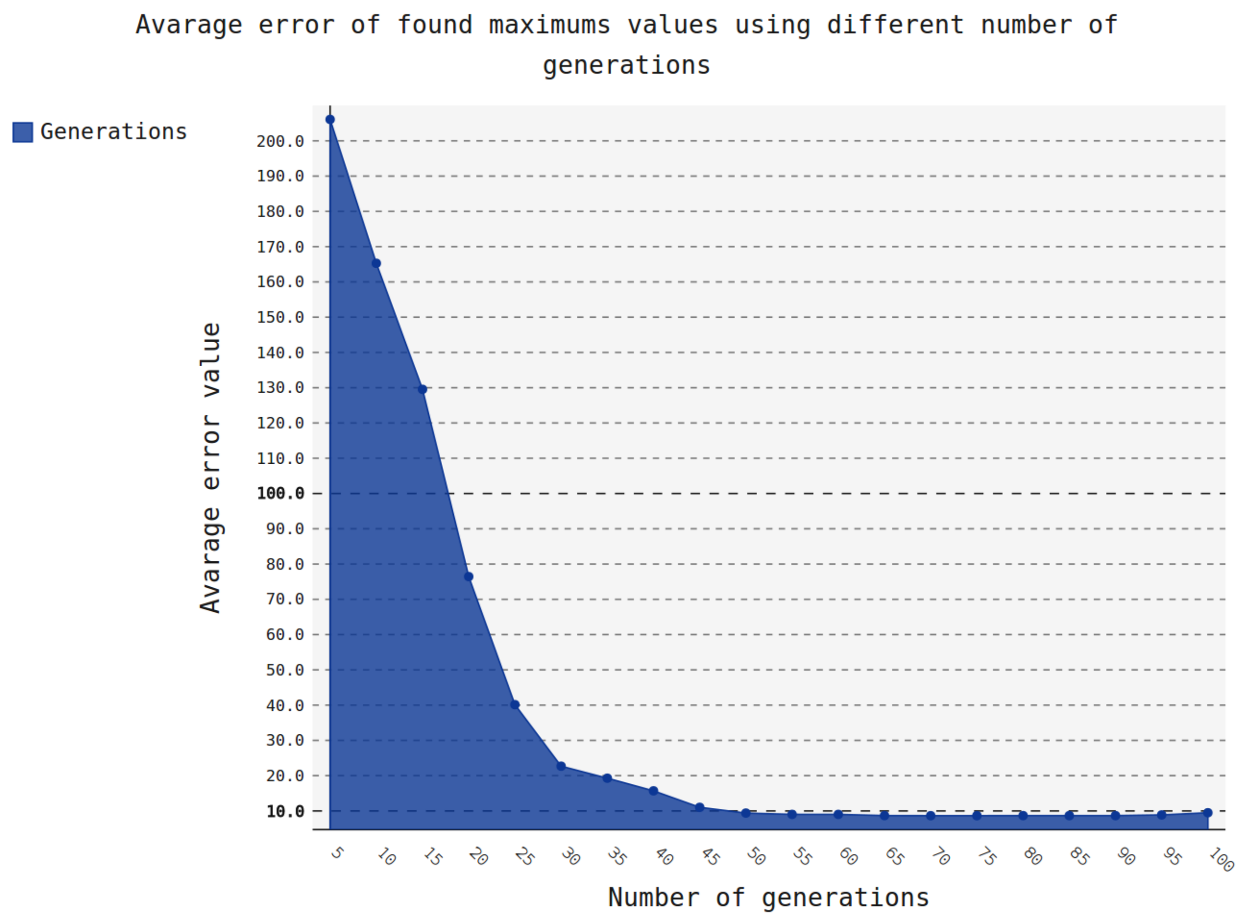

Figure 2: Results of the algorithm test with different number of generations.

3.3. Number of individuals in population test

To check how size of population impact on gained results there was performed test with different population size. Number of individual in population was increased by 15 from 15 to 450. As it can be seen on Figure 3 average error is going down in range of population size between 15 and 75. The best results can be obtained using population with size equal to 150. For population size great than 150 average error is increasing which is most likely caused by limited number of resources in each niche.

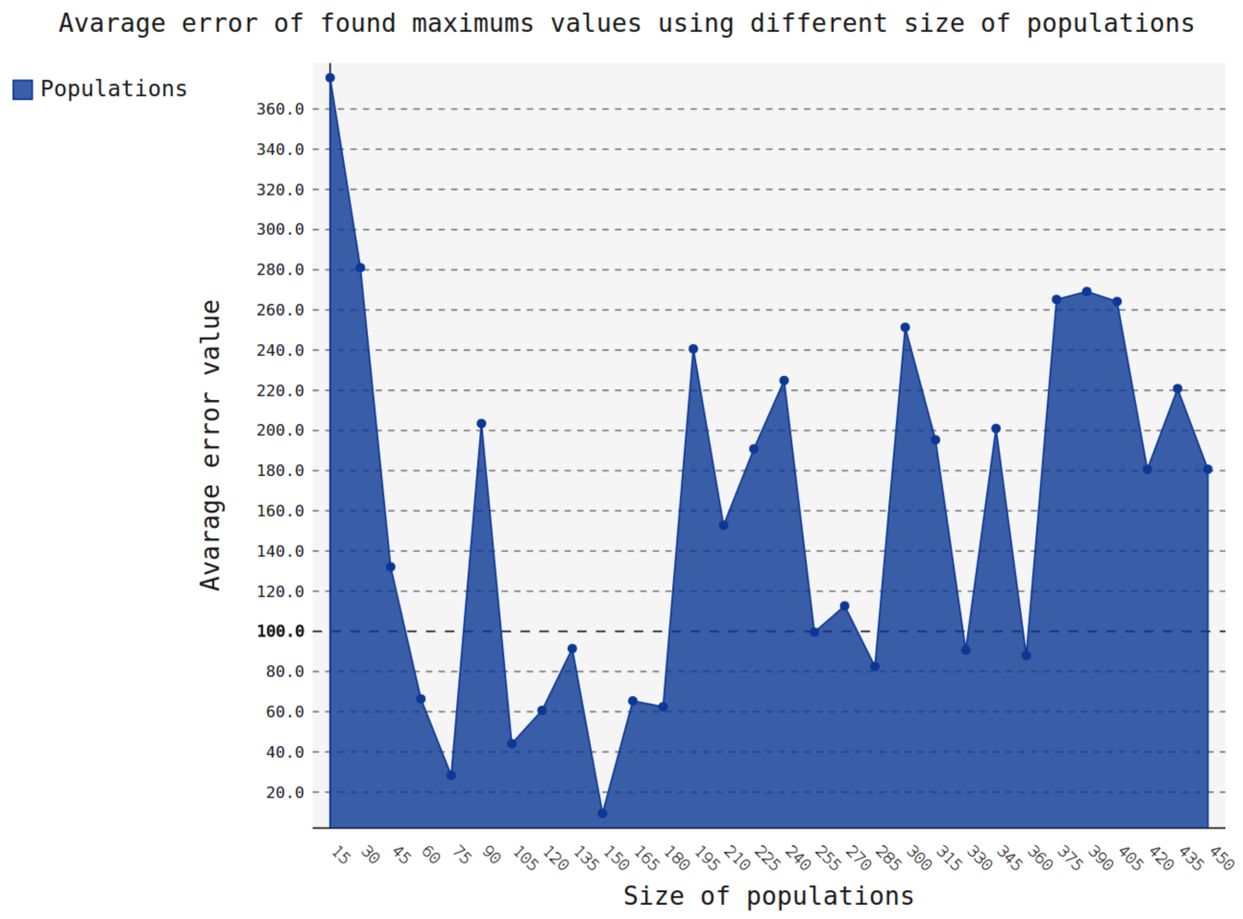

Figure 3: Results of the algorithm test with different sizes of population.

3.4. Number of subpopulations test

It is very important to have proper number of subpopulations to get best results so test which was performed used different number of subpopulations. Number of subpopulations was increased by 1 from 1 to 40. Results of the test can be seen on Figure 4 and it is clear that in case of 12 local maximas, the best results are gained with 12 subpopulations.

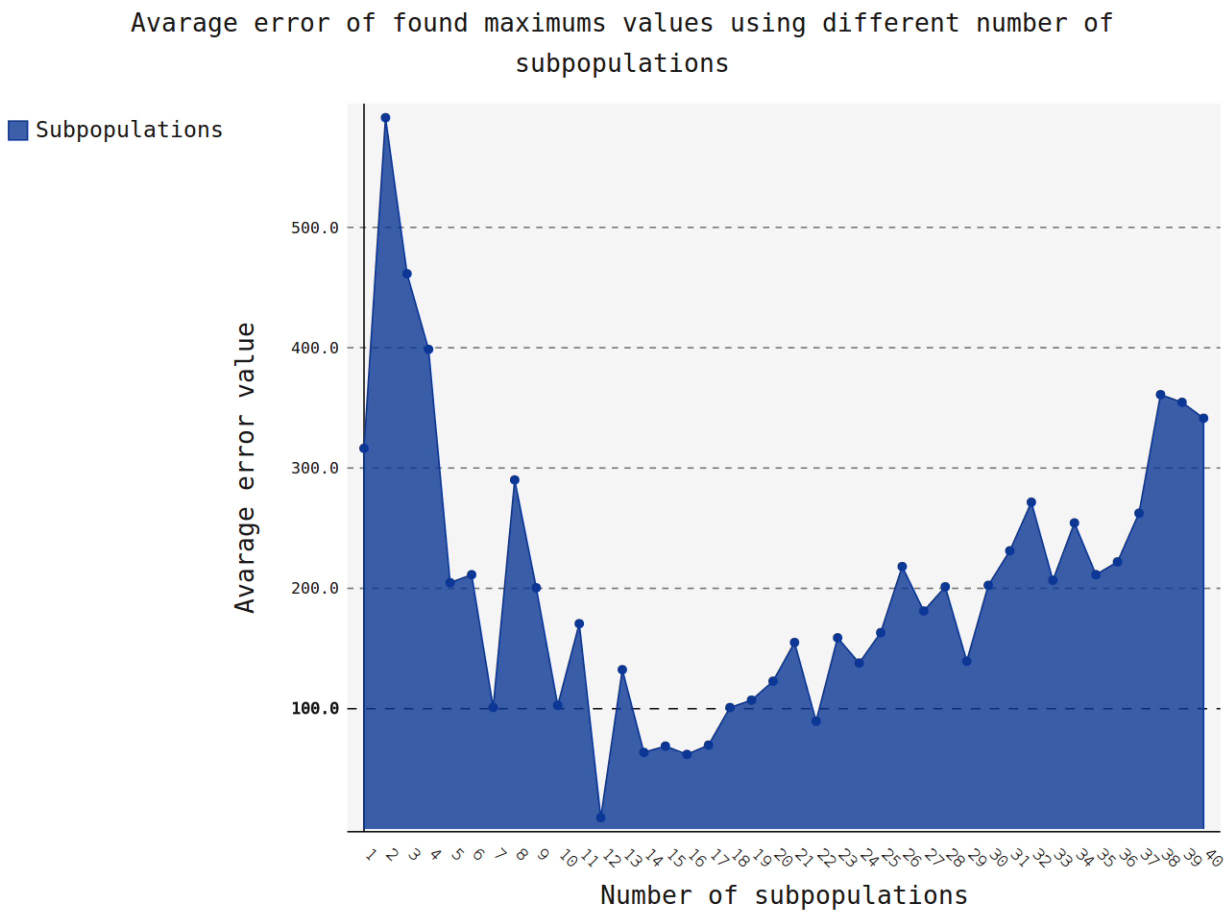

Figure 4: Results of the algorithm test with different number of subpopulation.

3.5. Crossing probability test

To check how crossing probability of two individuals is affecting results, test with changing crossing probability was performed. Probability was increased by 0.05 from 0.05 to 1.0. Figure 5 presents results of the test which show that best results are reached using probability equal to 0.6.

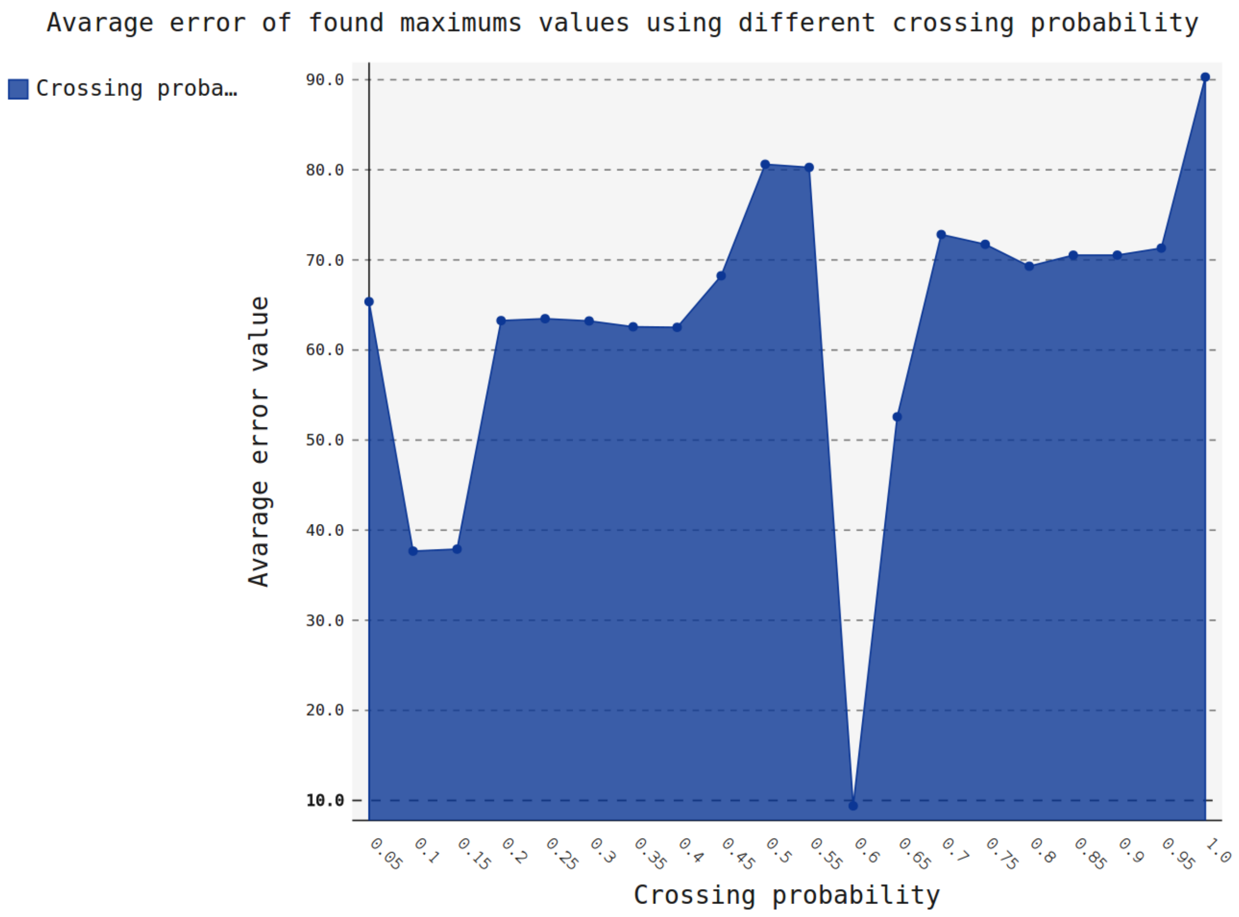

Figure 5: Results of the algorithm test with different crossing probabilities.

3.6. Mutation probability test

To check how mutation probability of two individuals is affecting results, test with changing mutation probability was performed. Probability was increased by 0.05 from 0.05 to 1.0. Figure 6 presents results of the test which show that best results are reached using probability equal to 0.1.

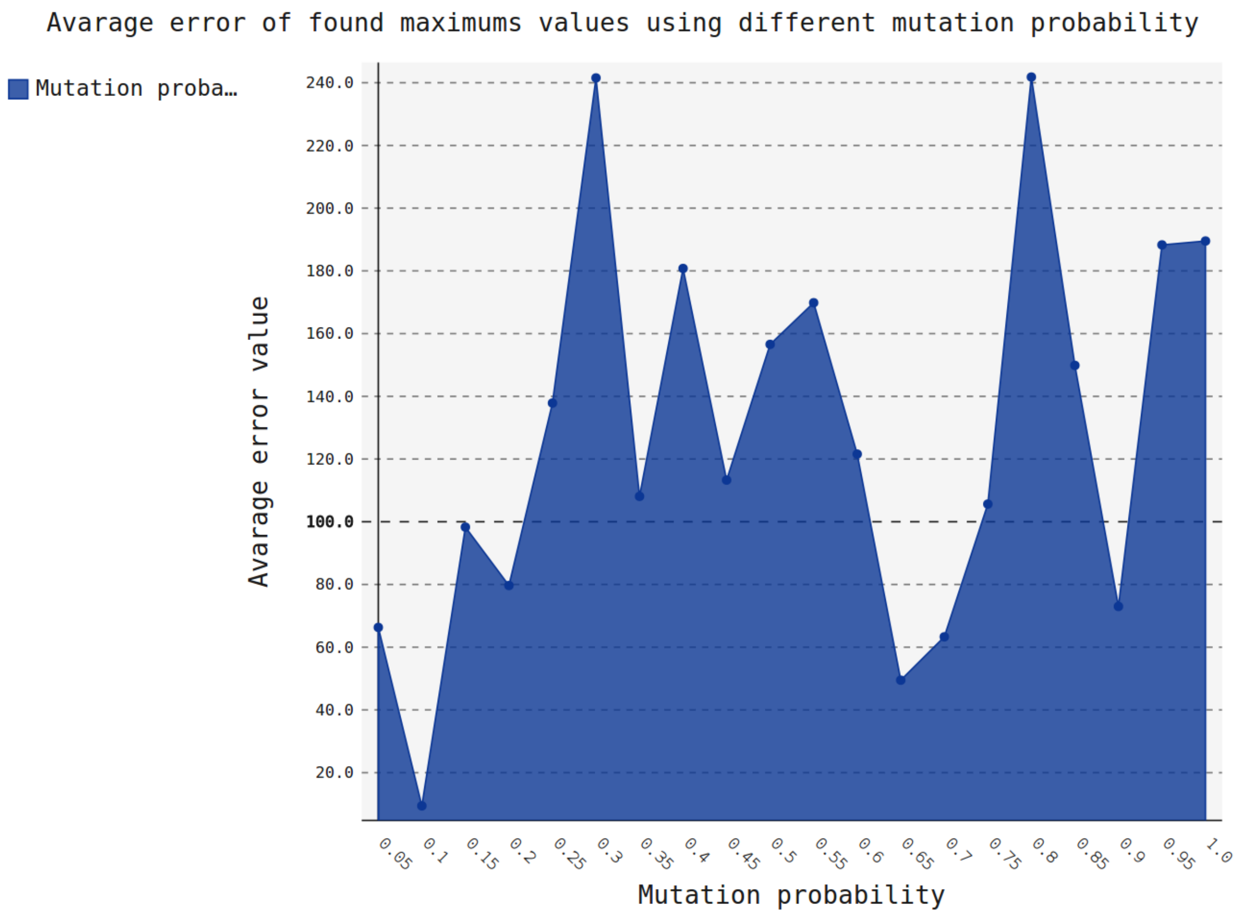

Figure 6: Results of the algorithm test with different mutation probabilities.

3.7. Ray distance used for niching test

In case of using niching mechanism it is very important to properly define size of niches to gain best results of computations. The ray distance factor which is used to multiply ray of the niche was used as a test parameter and it was checked how it affect work of the algorithm. Test checked the value of the average error for different niche ray values. Figure 7 presents results of the test which shows that error is decreasing along with niche ray but there is no significant improvement for values smaller than 0.02.

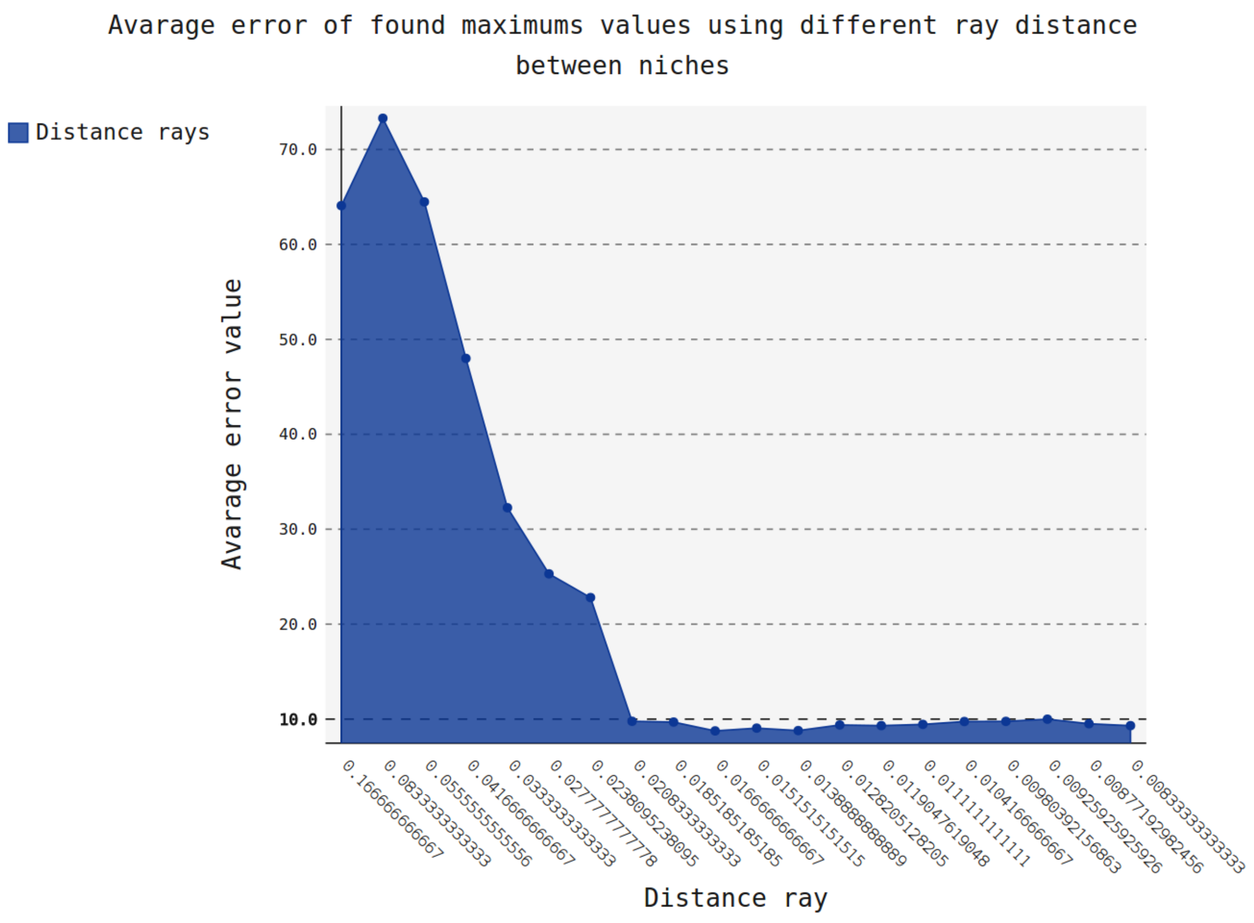

Figure 7: Results of the algorithm test with different ray distance factor.

4. Final remarks

Using genetic algorithms with niching mechanism is a very good solution for searching extremums in multi modal functions. Unfortunately the biggest problem on case of this kind of algorithm is to properly define input parameters especially distance radius factor and number of subpopulations. Those two parameters depends on type of the function and number of extremums which are searched so it is really hard to define universal values equation.

List of Figures

References

[1] Distributed evolutionary algorithm in python. http://code.google.com/p/deap/, 2013.

[2] A python svg charts creator. http://pygal.org/, 2013.

[3] Thomas Grüninger and David Wallace. Multimodal optimization using genetic algorithms. http://cadlab.mit.edu/publications/96-senin-tr9602-ga/dontindex/96-senin.pdf.

[4] Xin Yao. Niching and speciation. http://www.cs.bham.ac.uk/p̃kl/teaching/2009/ec/lecture_notes/l06-niching.pdf.