Dawid Jantosz

Signal processing and identification in control of mechatronic devices

Topic: Genetic algorithm

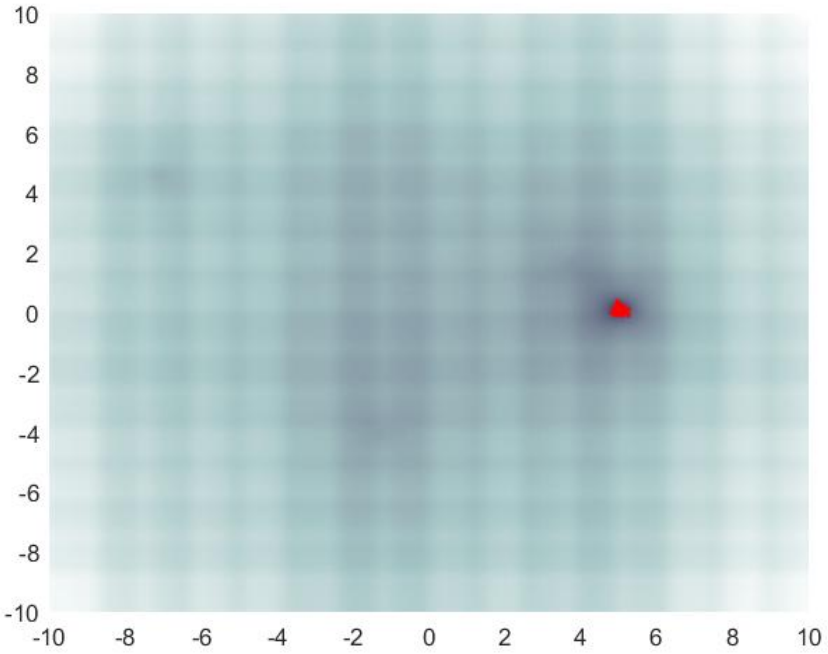
Introduction:

A genetic algorithm is an adaptive heuristic search algorithm inspired by "Darwin's theory of evolution in Nature." It is used to solve optimization problems in machine learning. It is one of the important algorithms as it helps solve complex problems that would take a long time to solve. The basic genetic the algorithm consists of the following steps:

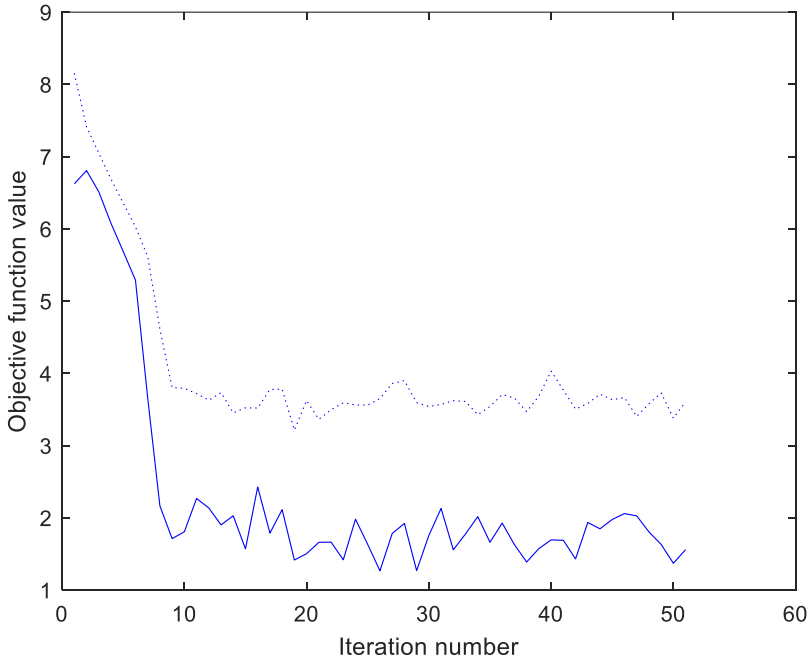
1. Generate randomly a population of solutions
2. Asses their value of objective function corresponding to each individual
3. Do a selection of subset of the best individuals
4. Mutate and crossover this subset creating new population

Implementation of genetic algorithm will start with the application of our basic random sampling code and preparing new metaparameters. We will also initialize population before starting while loop. Crucial is assessing fitness of whole population, instead of just one current solution. After implementation of the part of code enabling us to display current state of the population and store the best individuals, we can build the offspring population.

Ex 1. Configure and testing Genetic algorithm



(Figure 1 – Basic genetic algorithm)

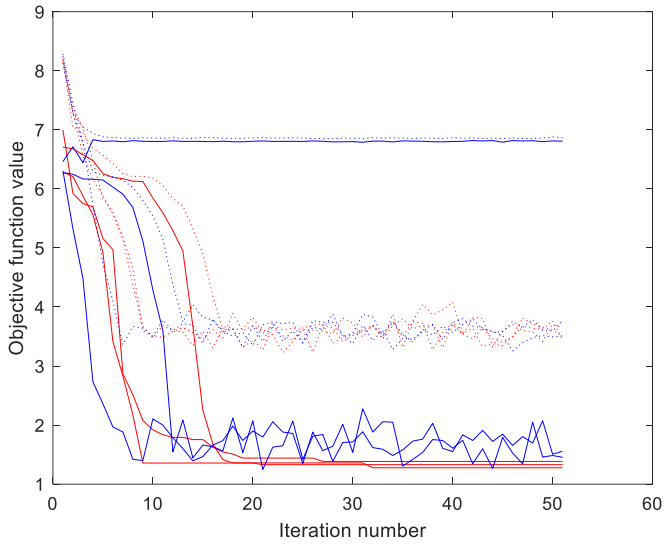


(convergence curve of basic genetic algorithm)

As we can see our genetic algorithm is working properly. Objective function is tending to decrease.

Ex.2 Comparison of genetic algorithm with and without elite succession

Selection is one of the basic steps in the operation of a genetic algorithm. It specifies which individuals will have a chance to pass its genes. During the lecture, we learned about such methods of making a selection as: n-best, Roulette, tournament and elite. In elite succession, each individual is assigned a fitness value via the fitness function. Using these scores, a percentage of the best, most fit individuals are used as parents. Elite succession is process that prevents from deterioration of result. We choose best individuals and pass the without alteration to the next population.



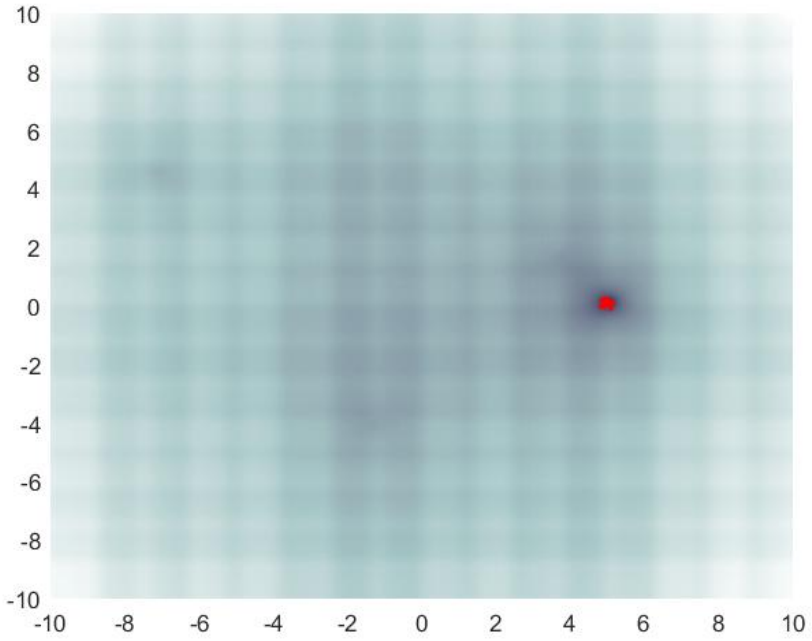
(Figure 2 – convergence curve of GA with(red) and without elite succesion(blue))

Blue plot is basic genetic alghoritm and the red one is genetic alghoritm with elite succesion. As we can see elite succesion gives us better result. Due to the best result of a given population not being carried over to the next one in the algorithm without elite succesion, we can see that one of the runs of said algorithm (marked in blue) got stuck in a local minimum and was not able to leave it.

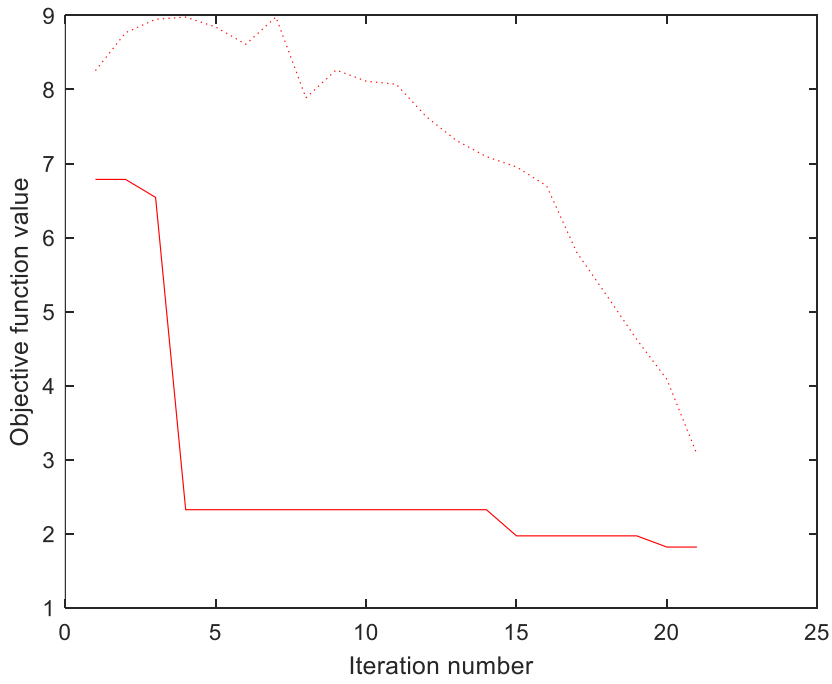
Ex.3 Many minima

Similarly, as in the 1+1 algorithm case, exploration vs exploitation balance is crucial in configuration of GA. In this exercise we have to balance exploitation/exploration metaparameters to obatin proper convergence curves. We should pay attention to avoid dead iterations, too long exploration/ exploitation and charp changes between

our metaparameters. Note that we have to increase number of objective funtions checks in case of many- minima, due to complexity of the problem.



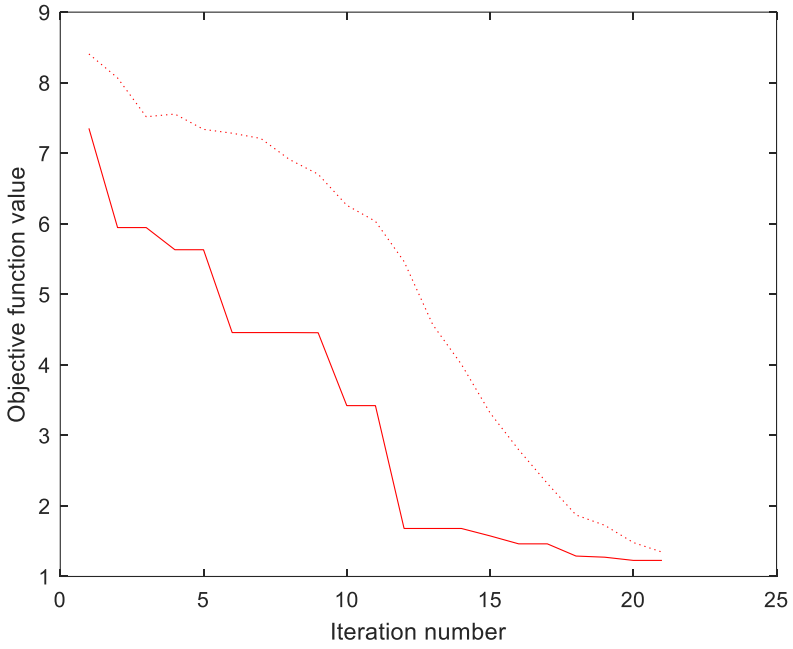
(Figure 3 – Genetic algorithm, manyminima case)



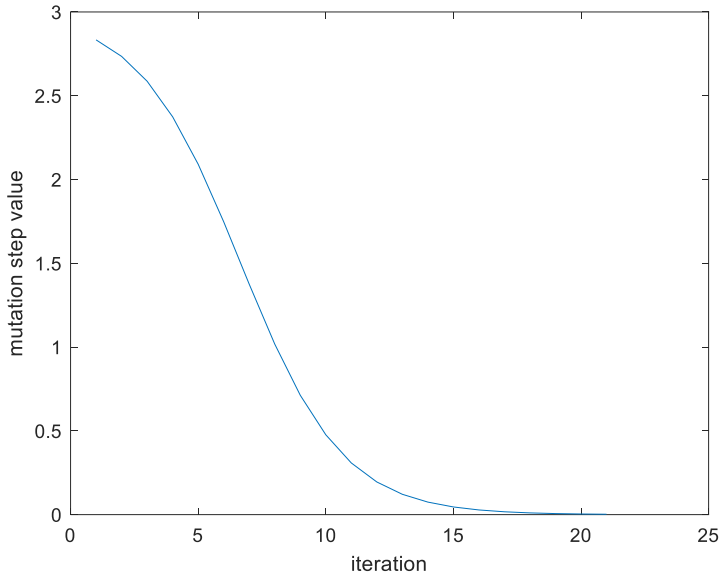
(Convergence curve – Genetic algorithm, manyminima case)

As we can see, our genetic algorithm for many-minima is not well optimized. Initial step is to high and exploration phase is too weak. We have to increase value of P1 and deacrese initial step value. The population has very high diversity till the end. It means

that we could have focused stronger on the neighborhood of the best minimum candidate.

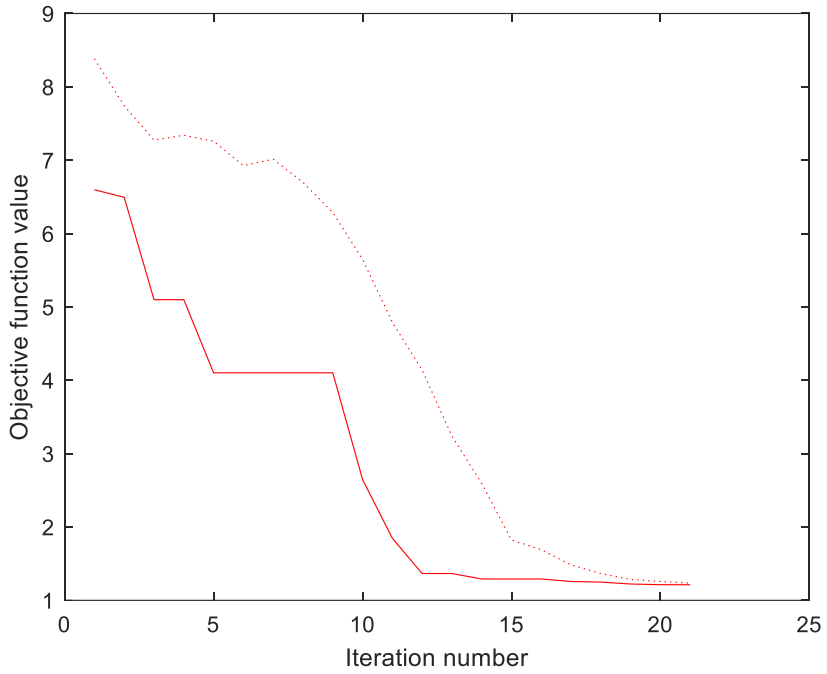


(Figure 4 – GA Step=3,P1=3, P2=2)

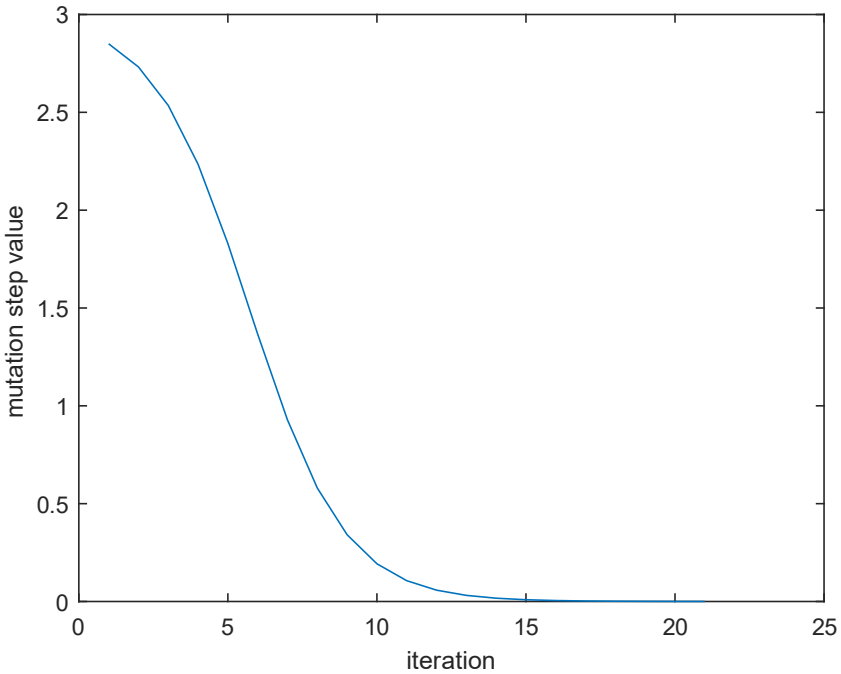


(Figure 5– GA Step=3,P1=3, P2=2)

With above parameters, we almost got satysfying solution. We can improve it by deacreasing value of exploration.



(Figure 6– GA Step=3,P1=3, P2=1.6)



(Figure 7– GA Step=3,P1=3, P2=1.6)

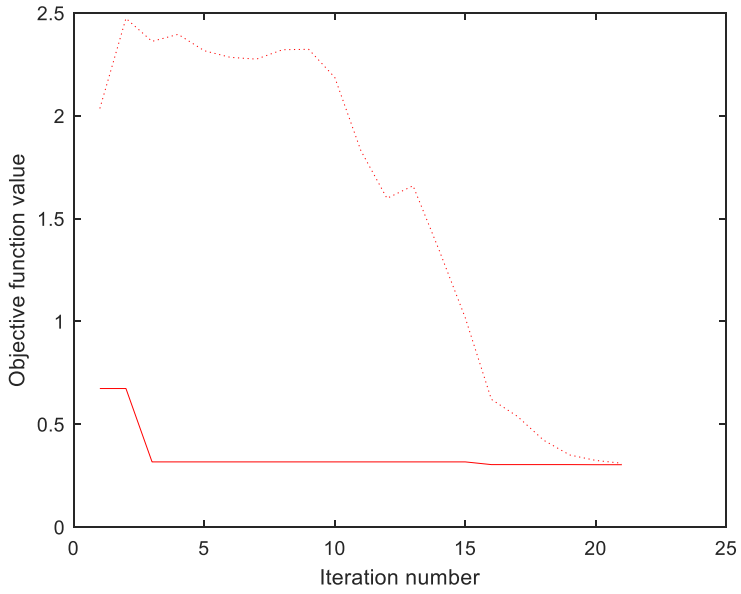
```
InitialStep = 3;
P1 = 3.0;
P2 = 1.6;
```

Conclusions:

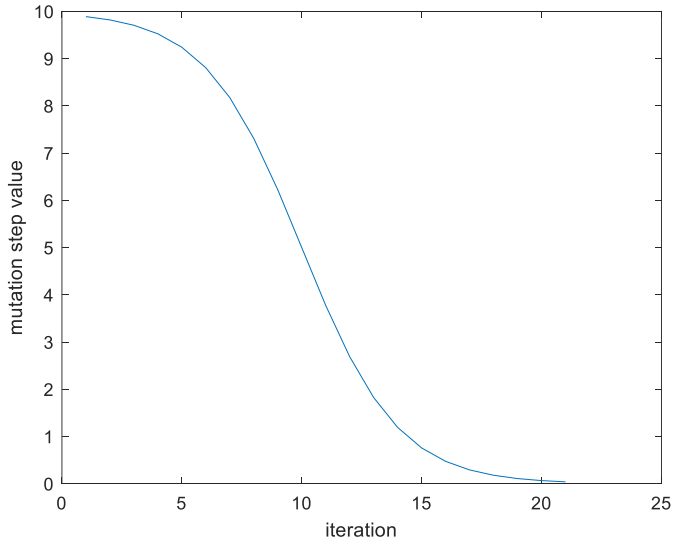
As we can see, our genetic algorithm for many-minima is well optimized, there is no sharp changes between exploration and exploitation. The population has very low diversity till the end. It means that we obtained satisfying solution.

Ex 3.2 Few-minima

In this exercise we have to balance exploration/exploitation parameters as for previous algorithm, but this time we will use few_minima type of function. In this type of function we can suspect that length of mutation step will be important, because relatively high step won't be able to improve our solution.

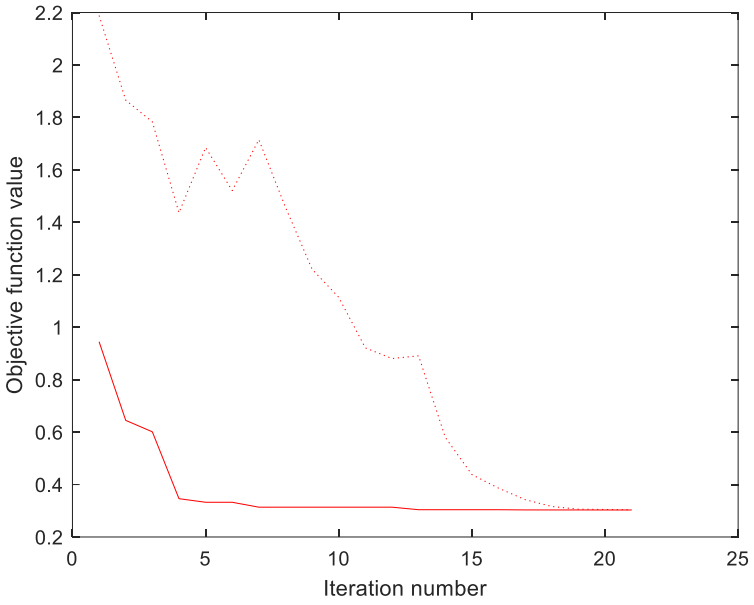


(Figure 8– GA Step=10,P1=2, P2=2)

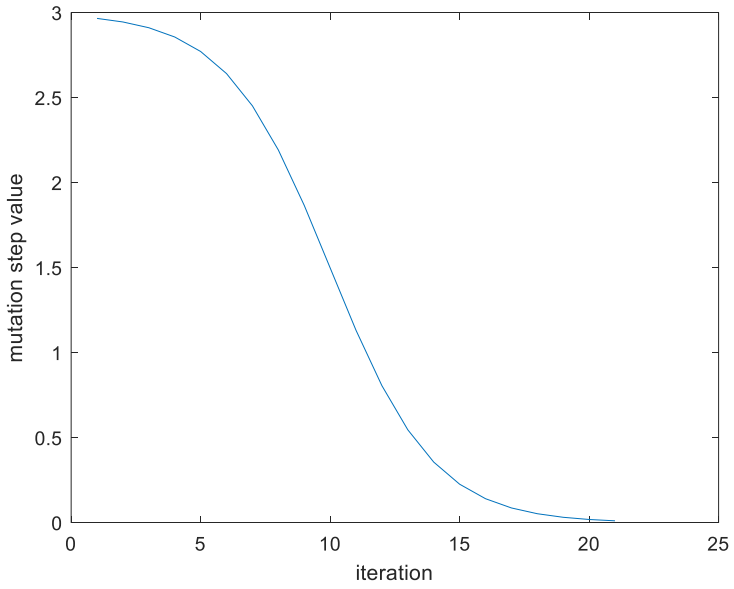


(Figure 9– GA Step=10,P1=2, P2=2)

In this case we are not sure if we have actually hit the bottom here. With lower step we will be surer that our solution improves.



(Figure 10– GA Step=3,P1=2, P2=2)



(Figure 11– GA Step=3,P1=2, P2=2)

Choosen parameters:

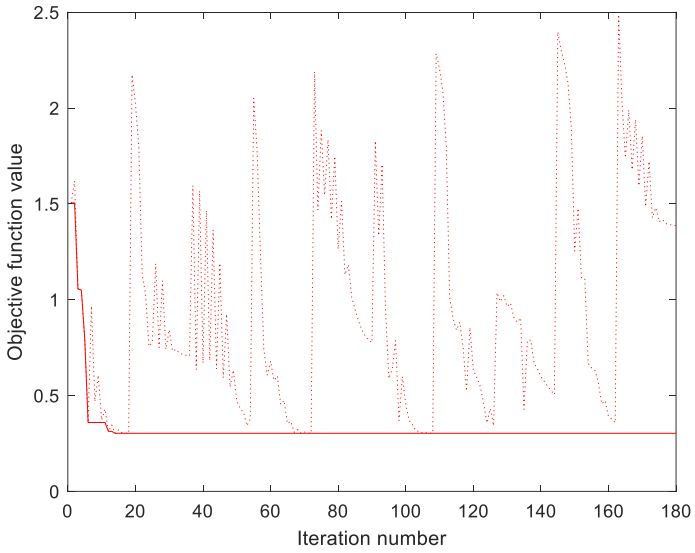
```
InitialStep = 3; % Exploration/exploitation balance parameters:
P1 = 2;
P2 = 2;
```

As we can see, our genetic algorithm for many-minima is well optimized, there is no sharp changes between exploration and exploitation. The population has very low diversity till the end. It means that we obtained satisfying solution.

Ex.4 Comparison of multistart gradient, 1+1 algorithm and genetic algorithm

Ex.4.1 Multistart gradient algorithm for fewminima type

In practical applications we often have many local minima, what causes that we are never sure if we have the best possible solution, but in this case we are using few_minima type of function, which indicates that we will have small number of minima.



(Figure 12– Multistart gradient alghoritm for fewminima type)

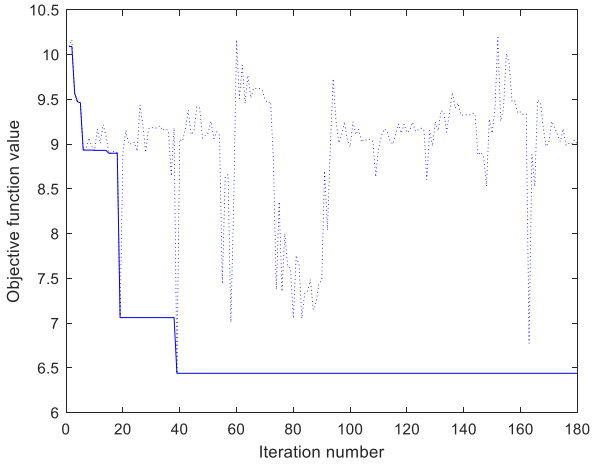
Chosen parameters:

```
InitialStep = 2
P1 = 2;
P2 = 2;
```

As we know multistart gradient algorithm for manyminima type of function is sensitive of local minima. If our number of minimas is limited, this method works well in terms of getting satisfying results.

Ex.4.2 Multistart gradient algorithm for manyminima type

In this case we will observe the performance of Multistart gradient algorithm for manyminima type. By this point we've learned that characteristics of gradient algorithm will cause that algorithm will be sensitive to local minima and the result may not be satisfying.

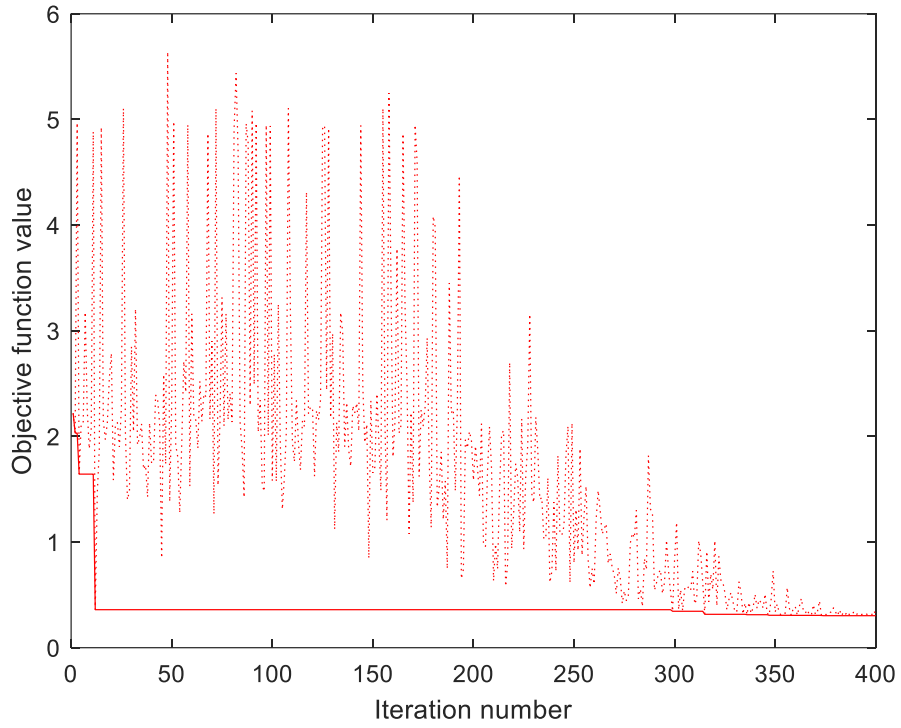


(Figure 13– Multistart gradient alghoritm for fewminima type)

As we though Multistart gradient algorithm for manyminima type isn't the best choice. Properties of this algorithm doesn't allow us to obtain satisfying solution.

Ex.4.3 1+1 alghoritm for fewminima type

To obtain algorithm 1+1 we have to generate a random starting point. Next step will be checking a value in this point, then generating a new point by adding to him a small random value. After that, the new point is being either preserved or discarded, depending on whether it yields a value lower than its predecessor.



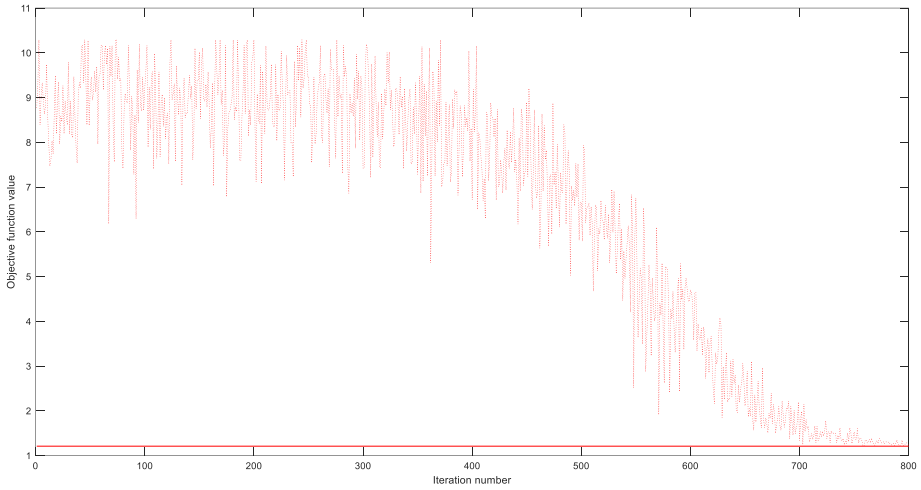
(Figure 14– 1+1 alghoritm for fewminima type)

```
InitialStep = 8; % Exploration/exploitation balance parameters:
P1 = 2;
P2 = 45;
```

As we can see, our algorithm worked properly. With well determined parameters balance between exploration and exploitation is preserved and solution is satisfying.

Ex.4.4 1+1 algorithm for manyminima type

This time we will repeat procedure from the previous exercise, but in this case using 1+1 algorithm for manyminima type of function.



(Figure 15– 1+1 alghoritm for manyminima type)

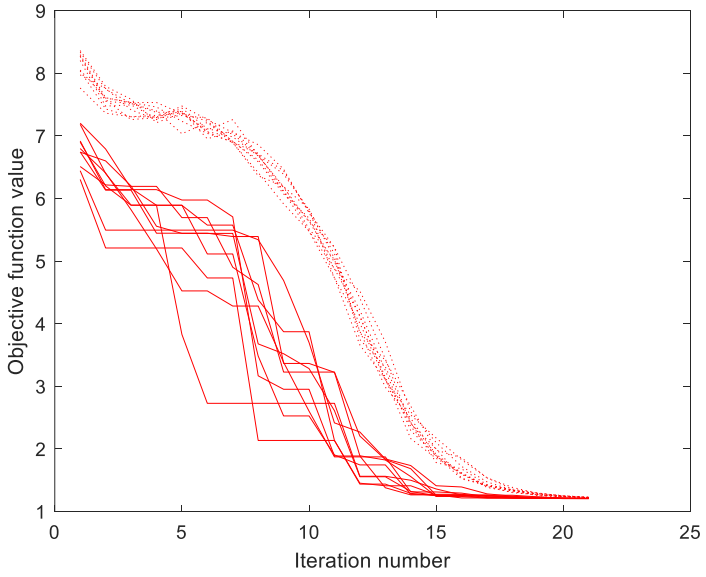
Choosen paramets:

```
InitialStep = 10; % Exploration/exploitation balance parameters:
P1 = 2;
P2 = 45;
```

In this case our obtained solution was less precise in comparison with 1+1 algorithm for manyminima type. We can see this phenomena due to sensitivity of 1+1 algorithm to local minima.

Ex.4.5 Genetic algorithm for manyminima type

Genetic algorithm is in fact really similar to 1+1 algorithm, so we can suspect that our solution, respectively for Genetic algorithm for manyminima type and for Genetic algorithm for fewminima type will also be similar.



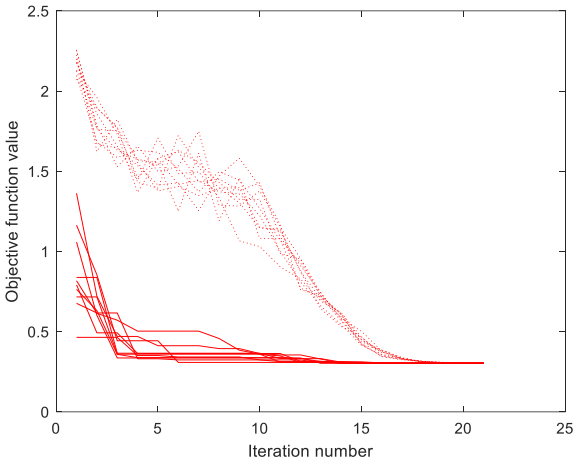
(Figure 16– Genetic alghoritm for manyminima type)

Choosen parameters:

```
InitialStep = 3; % Exploration/exploitation balance parameters:
P1 = 3.5;
P2 = 1.6;
```

With proper exploration/exploitation parameters we obtained solution that is well-optimized. The population has very low diversity till the end. It means that we have focused stronger on the neighborhood of the best minimum candidate. What is more there is no steep changes from exploration to exploitation and dead iterations.

Ex.4.6 Genetic algorithm for fewminima type



(Figure 17– Genetic alghoritm for fewminima type)

Choosen parameters:

```
InitialStep = 3; % Exploration/exploitation balance parameters:
P1 = 2;
P2 = 2;
```

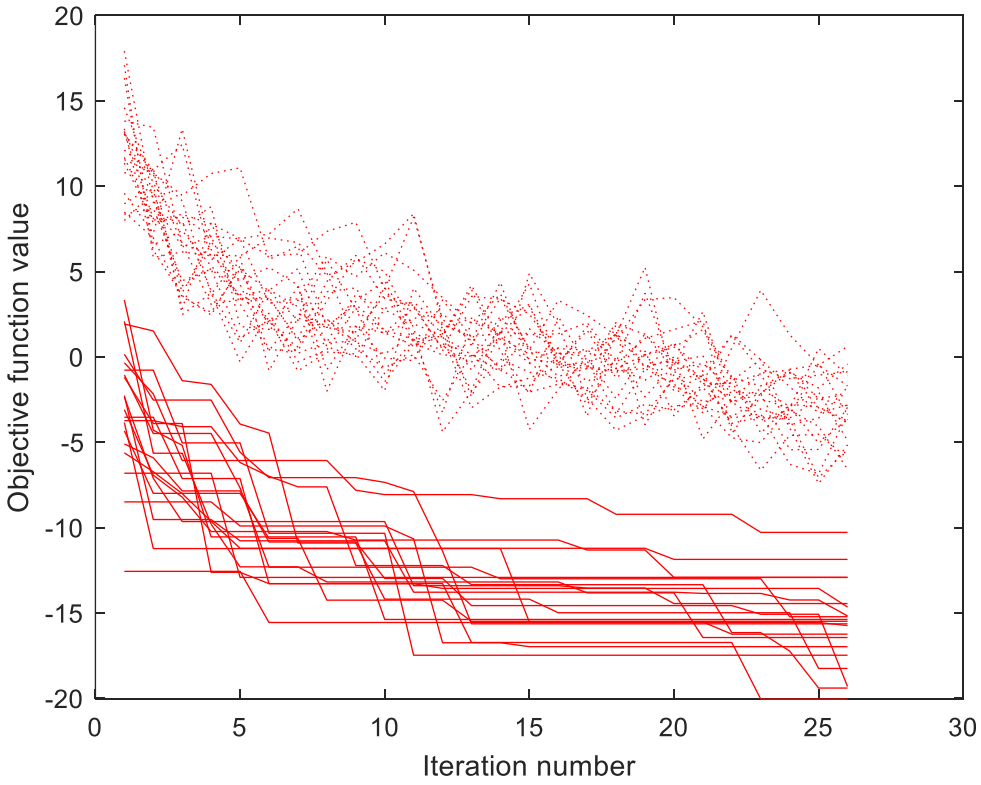
Conclusions:

In multistart gradient algorithm for manyminima function we can not obtain proper solution. This method is not suitable for problems with large numbers of manyminima. Other methods works relatively well.

Ex.5 Genetic algorithm for multidimensional type

In this exercise we have to configure our Genetic algorithm for multidimensional type. We can adjust parameters by the usage of convergence curve or by the use of histograms. In this exercises I used multiple convergence curves and parameters that are stored in the table below.

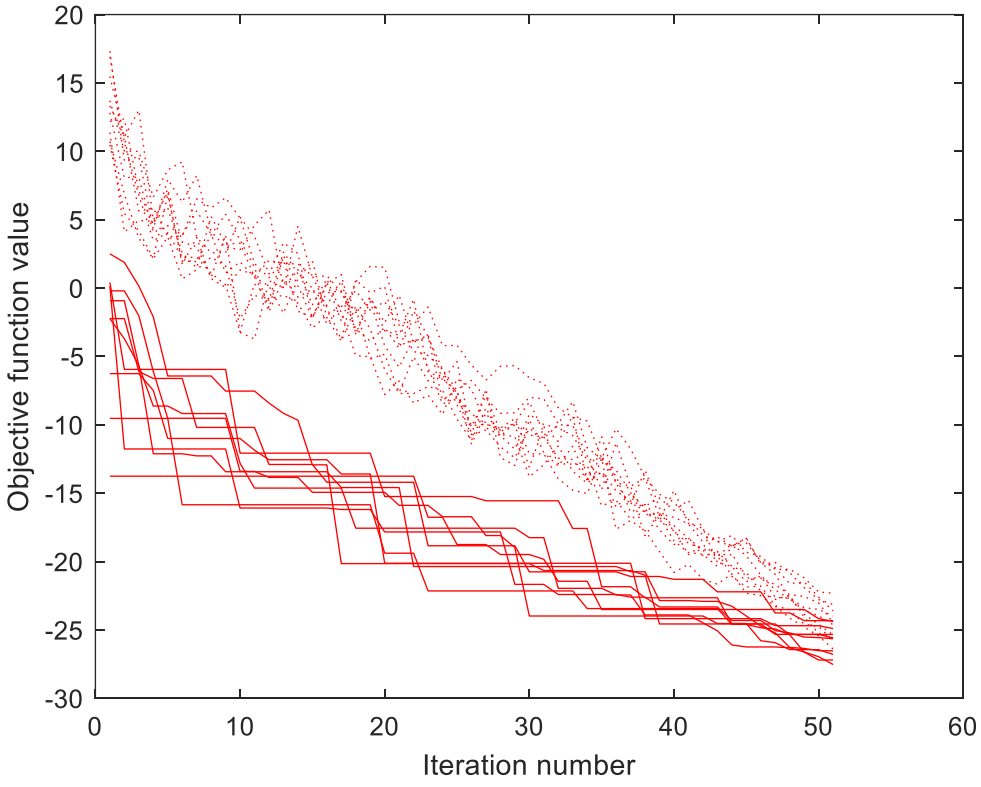
Solution for basic parameters:



(Figure 18– Genetic alghoritm for multidimensional type-Initial result)

Too weak exploitation – the population has very high diversity till the end. It means that we could have focused stronger on the neighborhood of the best minimum candidate.

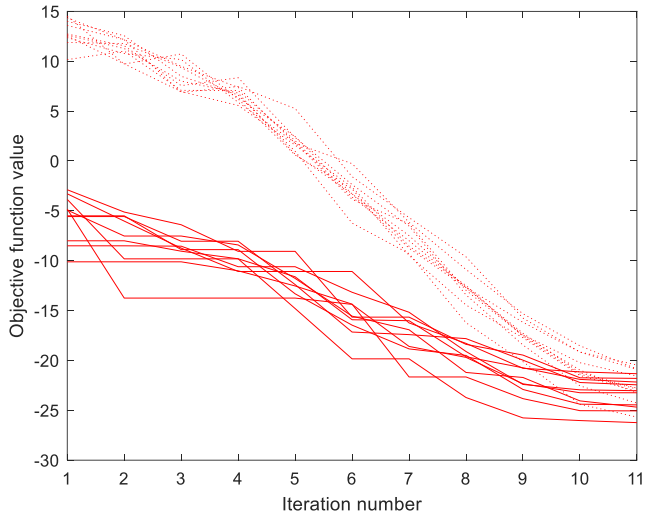
p1	2	3	3,5	4	4,2
p2	10	8	5,5	5	5
initial step	2	2	2	2	2
n	10	10	10	10	10
psize	20	20	20	20	20
maxsteps	50	50	50	50	50
mean	-18,8703	-21.4678	-25.0121	-26.0632	-25.8232
std	1,8429	2,2180	1,6045	1,2823	1,1424
results	-19.68125	-19.28264	-24.14181	-24.06607	-24.32519
	-14.46103	-20.89654	-25.19850	-28.00924	-27.53089
	-18.16790	-20.03826	-23.62693	-26.83235	-24.38274
	-17.86014	-21.04528	-24.85101	-23.74596	-25.33340
	-18.93829	-19.92215	-26.87400	-25.73238	-27.20203
	-19.40835	-26.80089	-23.93983	-26.19563	-24.89854
	-20.39919	-21.51269	-27.32190	-26.53954	-25.59520
	-20.49313	-20.90371	-22.79657	-26.33334	-26.79262
	-18.53235	-20.57220	-27.21271	-26.79049	-25.64719
	-20.76098	-23.70331	-24.15735	-26.38712	-26.52375



(Figure 19– Genetic alghoritm for multidimensional type-final result)

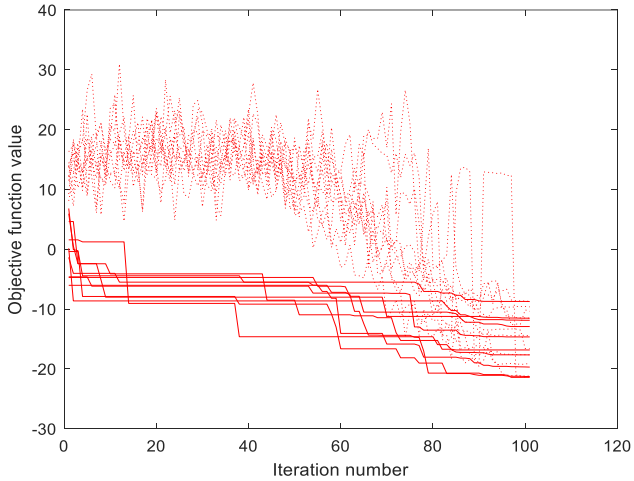
As we can see, we obtained result with the smallest std value and the best repeatability. We increased p1 values while decreasing p2 to obtain the optimal solution.

In the example before we optimized our genetic algorithm only changes the values corresponding to exploration and exploitation. What will be the results when we change the number of iterations and max steps to extreme values? We have to remember that to obtain fair solution i.e. solution that doesn't favor any approach we should use the same number of objective function checks. At first I will choose p_size equal to 100 and 10 iterations.



(Figure 20– Genetic alghoritm for multidimensional type-focus on large population and small number of iterations)

In this case after configuration of P1 and P2 parameters, which led to P1 equal to 3 and P2 equal to 0.6 we obtained solution with relatively high value of std -1.5970. In case with large population and small number of iterations it is very hard to obtain good exploitation that is necessary to lower final result.



(Figure 21– Genetic alghoritm for multidimensional type-focus on low population and high number of iterations)

Our function with low population and high number of iterations has troubles with hitting minima. In multidimensional space adjusting parameters P1 and P2 is difficult, when we have low population. It is troublesome to provide any progress.