CIIT. Work #2 Alex Hreben AI-10

Task 1.

High-dimensional Test Function

1. In first time we generated started population (20 chromosomes with 20 genes)

```
-0.79 3.82 -2.9 -2.34 -2.75 -0.7 -1.19 4.32 -1.72 4.98 2.15 1.82 0.11 -0.76 3.86 -4.95 -1.6 1.46 -3.15 1.32
2.89 4.84 2.69 0.9 0.16 -4.65 4.1 -2.56 0.02 -3.57 3.58 2.2 0.16 3.96 -0.91 -0.49 -0.24 -2.42 2.22 -0.49
-4.6 -4.9 -2.07 -1.93 -0.14 -0.25 -2.95 2.99 3.26 -3.49 -3.48 -3.13 3.41 1.71 -0.64 -0.81 -3.01 -0.08 4.16 1.81
-1.36 -3.9 2 0.06 1.51 4.73 -4.03 0.22 1.35 0.11 -0.08 -4.67 3.81 4.05 3.73 2.49 -3.35 -3.07 1.59 -2.58
2.32 0.23 3.27 -2.84 -3.66 -4.65 1.4 -4.95 -0.03 0.55 -4.19 4.46 -2.78 -1.51 -0.75 3.73 -4.58 -0.73 -2.02 -3.65
4.99 -4.72 -0.63 3.71 -0.32 -4.74 -4.2 2.04 4.55 -1.86 0.37 3.95 -2.34 3.68 0.54 -0.84 -1.2 1.68 -1.66 -4.99
-1 0.11 -3.04 -1.09 -2.18 4.62 4.85 -2.7 0.85 4.2 1.74 2.23 1.97 3.5 0.55 -4.16 3.65 -1.28 -3.95 3.74
-4.04 2.74 -1.17 0.49 -4.16 -3.24 0.51 4.53 -1.45 0.93 -1.66 2.61 0.81 0.38 1.93 -1.23 -4.5 2.9 -2.14 1.79
1.96 2.73 -2 -2.39 1.87 0.21 -0.94 2.2 -3.79 -2.04 1.21 -4.55 -0.72 0.48 -2 -1.11 0.22 2.92 2.43 -3.5
-2.97 -3.92 2.11 2.96 -3.39 4.3 -0.86 -1.57 -3.36 3.73 -4.07 -2.45 4.35 -3.78 3.86 3.05 2.63 -4.11 3.07 0.24
-3.06 0.02 3.68 -4.2 -4.13 -4.55 3.8 0.26 1.68 4.47 1.29 -4.72 -3.68 -2.57 -0.1 0.46 -2.58 0.68 3.36 3.09
-4.02 -4.41 -1.39 1.83 -1.18 4.9 2.85 4.56 -4.17 -1.62 -1.35 -3.33 2.98 -1.5 0.65 1.19 -3.87 -2.5 2.4 0.36
-4.41 -0.41 -3.17 4.97 4.5 0.35 -1.74 4.16 -1.22 4.87 -4.18 -1.96 1.18 -1.05 -1.32 -1.83 2.98 0.23 0.24 1.25
-1.63 -4.81 -0.91 2.13 3.6 -4.84 2.68 1.15 -1.94 3.08 -4.44 3.51 1.77 2.98 2.17 -2.6 -4.35 -4.76 3.28 1.65
-1.64 -1.91 -0.55 1.67 3.62 1.66 -0.82 -2.24 -1.11 4.83 2.36 0 -3.84 1.94 -1.37 1.91 4.87 -4.01 -0.22 2.95
1.03 1.42 -0.09 -0.6 -0.09 -2.47 2.2 2.17 -0.86 -3.18 -3.37 -1.79 -4.01 -1.92 -1.72 0.86 2.97 4.29 -1.96 2.68
-3.83 2.55 2.64 3.85 2.72 3.17 1.02 0.56 -4.95 -2.51 -4.69 0.04 2.7 -3.25 2.34 0.9 4.18 2.68 -4.12 -4.95
3.35 -4.44 -2.8 -2.32 -3.99 -1.55 3.41 4.9 -2.77 4.74 1.1 -2.66 1.2 3.07 -2.11 0.57 -1.71 -0.79 0.99 -0.4
-0.11 -1.96 -2.58 -1.76 -0.92 2.98 -4.67 -1.43 0.88 -0.93 3.2 1.87 -2.67 -1.41 -2.85 -3.77 4.89 4 -0.68 -0.79
0.4 0.61 2.11 -4.59 4.88 0.3 -3.34 4.07 0.7 0.05 0.45 -3.21 -3.41 4.21 -2.36 -0.43 -0.86 -3.24 -1.46 -0.78
```

All genes (-5;5)

2. Then we calculate fitness function and sort our population.
   Our fitness function:

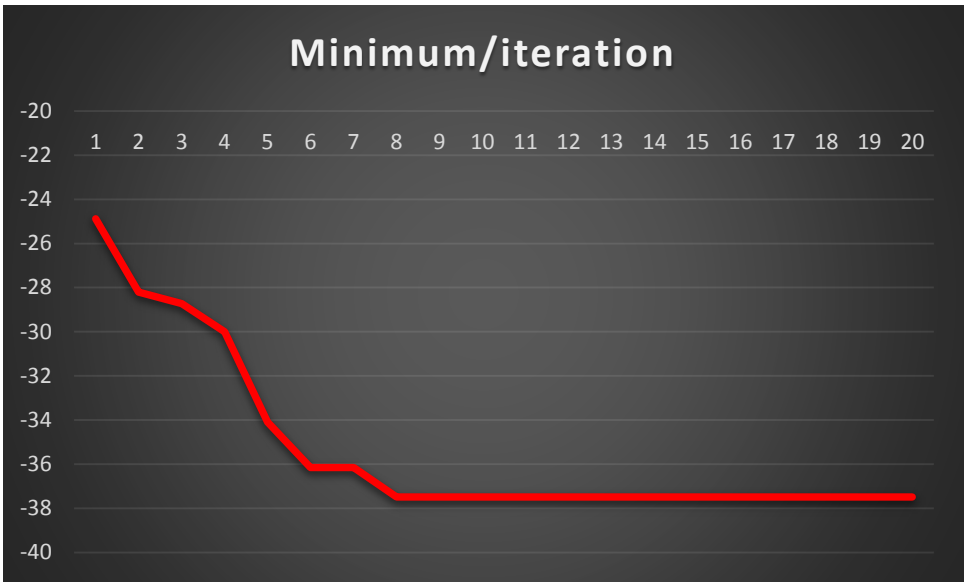
$$y = x1 \sin(\ /\ x1\ /) + x2 \sin(\ /\ x2/) + \cdots + x20 \sin(/\ x20/)$$

3. Ten chromosomes with smallest fitness we cross with help one-point crossover.
4. Repeat 2-3 while fitness no change.

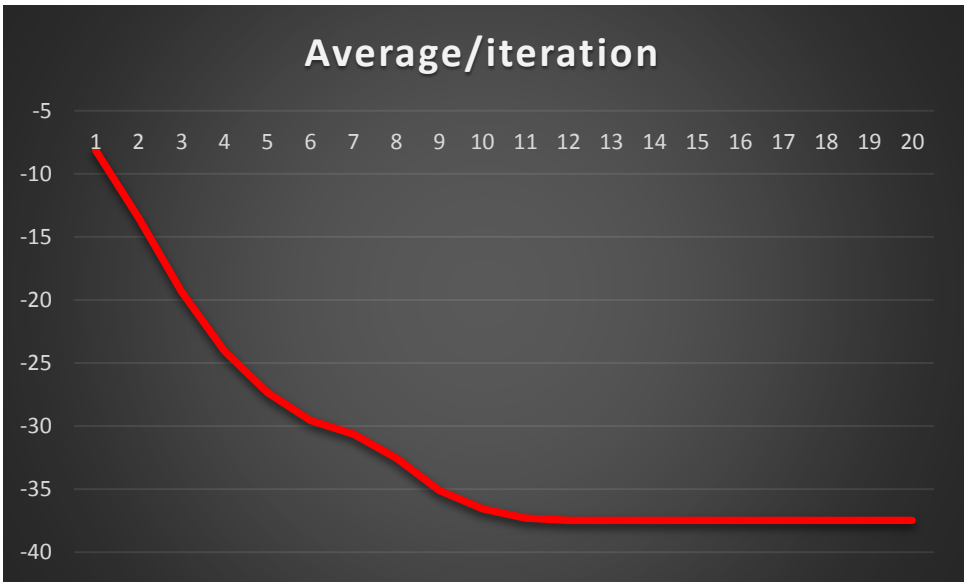
Code:

```
import java.util.Arrays;
import java.util.Random;
import java.math.*;

public class siitT {
	public static int chromosoms = 20;
	public static int genes = 20;
	public static float tmp = 0.0f;
	public static float[][] population = new float[chromosoms][genes];
	public static float[] resultY = new float[chromosoms];
	public static float sum = 0f;
	public static float lastY = 0;
	public static float[] best = new float[genes];
	public static float []xz = new float[chromosoms];

	public static void cross(float[] a, float[] b, int n){
		float[] arr1 = new float[a.length];
		float[] arr2 = new float[b.length];
		float[] end1 = new float[a.length];
		float[] end2 = new float[b.length];

		for (int i = n; i < a.length; i++) {
```

```
                end1[i] = a[i];
                end2[i] = b[i];
            }
            for (int i = 0; i < a.length; i++) {
                arr1[i] = a[i];
                arr2[i] = b[i];
                if(i>n){
                    arr1[i] = end1[i];
                    arr2[i] = end2[i];
                }
            }
        }

        public static void generatePop(){
            Random rnd = new Random();
            for (int i = 0; i < chromosoms; i++) {
                for (int j = 0; j < genes; j++) {
                    population[i][j] = rnd.nextInt(1000)/100f - 5;
                }
            }
        }
        public static void main(String[] args) {

            generatePop();
            for (int i = 0; i < chromosoms; i++) {
                for (int j = 0; j < genes; j++) {
                System.out.print(population[i][j] + " ");
                }
                System.out.println();
            }

            for (int i = 0; i < population.length; i++) {
                for (int j = 0; j < genes; j++) {
                    float a = (float) (population[i][j]*Math.sin(
Math.abs(population[i][j])));
                    sum =+ a;
                }
                resultY[i] = (float)sum;
            }

            for (int i = 0; i < resultY.length; i++) {
                xz[i] = resultY[i];
            }
            Arrays.sort(xz);
            lastY = xz[9];

            for (int i = 0; i < resultY.length; i++) {
                if (resultY[i] <= lastY) {
                    best[i] = resultY[i];
                }
                }
            System.out.println("*************");
            for (int i = 0; i < resultY.length; i++) {
                System.out.print(resultY[i] + " ");
            }
            System.out.println();
            for (int i = 0; i < best.length; i++) {
                System.out.print(best[i] + " ");
            }
                }
}
```



Picture 1. Minimum value of fitness function by iteration.



Picture 2. Average value of fitness function by iteration.

Task 2.

2-D version of Schwefel's function

Exercise 6 *1. Minimize*

$y = x \sin(|x|)$

in the following way!

(1) Represent value of x by a 10-bit binary chromosome.

(2) Create a population of 20 chromosomes at random, with fitness being y.

(3) Evolve this population till fitness dosen't change.

2. Show

(1) the graph of fitness vs generation.

(2) all 20 points (*x, y*) *in the 1st, an intermediate, and final generation.*

1. Generate start population (20 chromosomes with 10 binary genes).
2. Calculate fitness with.

   Our population with fitness:

Chromosomes	Fitness
1001010011	5
1111101100	7
1100000001	3
0101111011	7
0000011010	3
1111100111	8
0100010100	3
1100110000	4
1001000111	5
1111001110	7
0101000000	2
0011000001	3
1011111101	8
1111100101	7
1011100011	6
0011001101	5
0000000001	1
0001011100	4
1100001011	5
0100010001	3

3. Then we write our result in (-5;5)

```
0.23097293042006534
-2.7713049103236975
4.735867173522031
2.012991264589042
1.4063699329594572
-0.05279364293534352
0.9838294872267892
-4.534921774217892
-2.3633582159065183
-3.5639294200686673
```

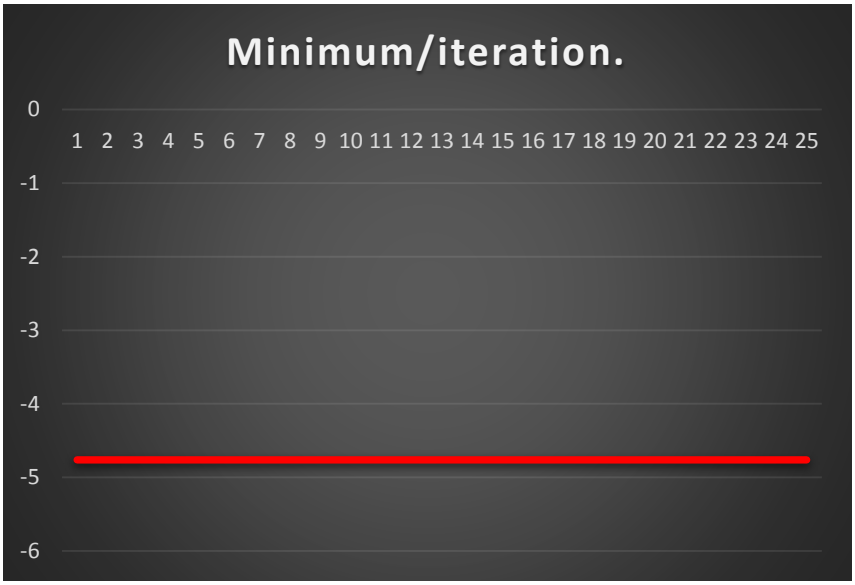
```
0.1890317564253989
-1.627000119664935
-0.5296445079106711
-3.8625831556710133
-0.29653330076398865
-0.6795759385619088
0.27795814161988114
0.10297741660542299
-0.0664506837444192
1.249746150523185
```

4. Ten chromosomes with smallest fitness we cross with help one-point crossover.
5. Repeat 3-4 while fitness don't change.

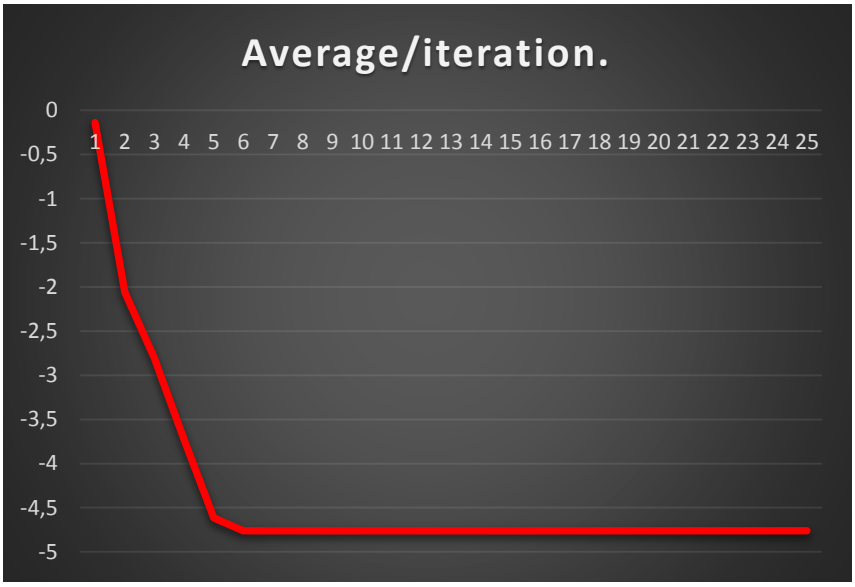
Code:

```
package siitTT;

import java.util.Arrays;
import java.util.Random;

public class siitTT {

	public static int chromososm = 20;
	public static int genes = 10;
	public static int[][] population = new int[chromososm][genes];
	public static double numb = 0d;
	public static double tmp = 0d;
	public static int generation = 0;
	public static int[] fit = new int[population.length];

	public static double converte(int[] k) {
		double numb1 = 0;
		for (int i = 9; i >= 0; i--) {
			double tmp1 = k[i] * Math.pow(2, i);
			numb1 = numb1 + tmp1;
		}
		return numb1/1023*5;
	}

	public static double from2To10(int[] k){
		for (int i = 9; i >= 0; i--) {
			tmp = k[i] * Math.pow(2, i);
			numb = numb + tmp;
		}
		numb = numb * Math.sin(Math.abs(numb))/1023*5;
		if(k[0] == 0)
			return -(numb);
		else {
			return numb;
		}
	}


	public static int fitness(int[] a){
		int f = 0;
		for (int i = 0; i < a.length; i++) {
			f = f + a[i];
		}
		return f;
	}

	public static void cross(int[] a, int[] b,int n){
		int[] arr1 = new int[a.length];
		int[] arr2 = new int[b.length];
		int[] end1 = new int[a.length];
		int[] end2 = new int[b.length];
```

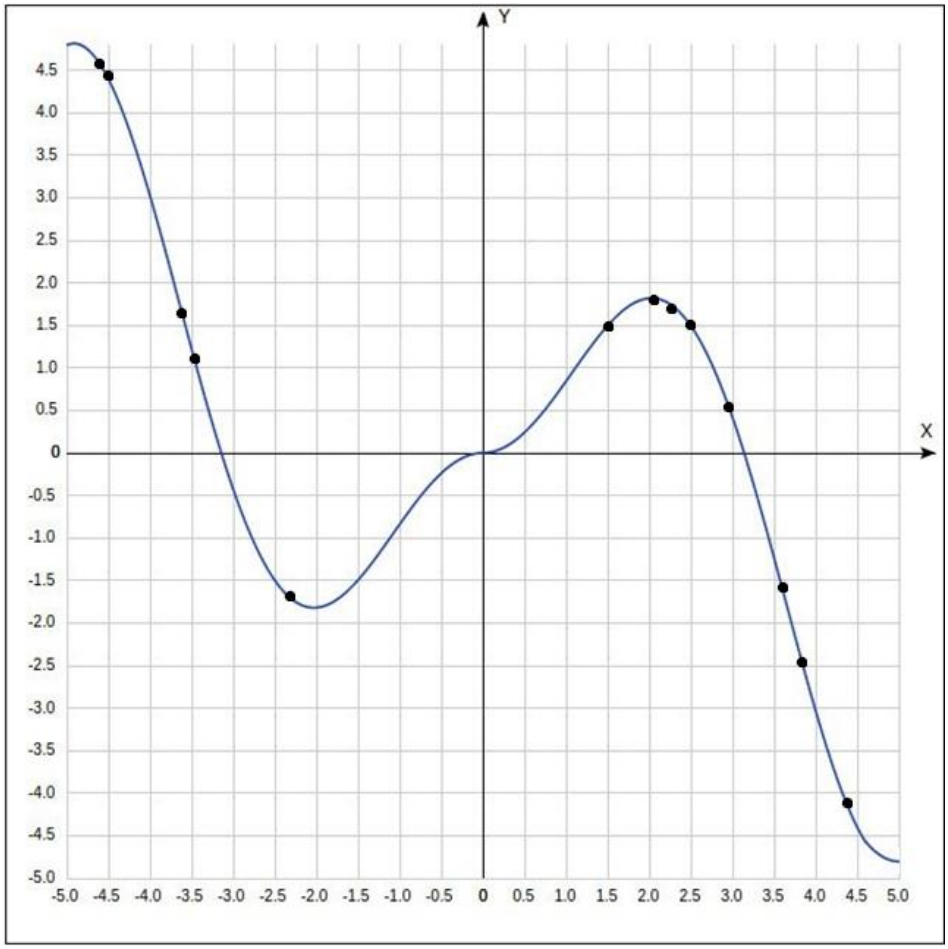
```
		for (int i = n; i < a.length; i++) {
			end1[i] = a[i];
			end2[i] = b[i];
		}
		for (int i = 0; i < a.length; i++) {
			arr1[i] = a[i];
			arr2[i] = b[i];
			if(i>n){
				arr1[i] = end1[i];
				arr2[i] = end2[i];
			}
		}
	}

	public static void generatePop(){
		Random rnd = new Random();
		for (int i = 0; i < chromososm; i++) {
			for (int j = 0; j < genes; j++) {
				population[i][j] = rnd.nextInt(2);
			}
		}
	}

	public static void main(String[] args) {

		generatePop();
		for (int i = 0; i < chromososm; i++) {
			for (int j = 0; j < genes; j++) {
				System.out.print(population[i][j] + " ");
			}
			System.out.println(" ");
		}

		for (int i = 0; i < population.length; i++) {
			System.out.print(fitness(population[i]) + " " );
			System.out.println(/*converte(population[i]) +*/ " " +
from2To10(population[i]) + " " );
		}
		for (int i = 0; i < population.length; i++) {
			for (int j = 0; j < population.length/2; j++) {
				System.out.print(population[i][j]);
			}
			fit[i] = fitness(population[i]);
			System.out.println("      " + fitness(population[i]));
		}
		System.out.println("_____________________________");

		Arrays.sort(fit);
		int q = fit[9];
		int[] minB = new int[20];
		for (int i = 0; i < fit.length; i++) {
			if(q > fit[i])
				minB[i] = i;
			else
				minB[i] = 0;
		}
		for (int i = 0; i < population.length; i++) {
			if (fitness(population[i]) >= minB[i]) {

			}
		}
		for (int i = 0; i < population.length; i++) {
			for (int j = 0; j < population.length/2; j++) {
				System.out.print(population[i][j]);
			}
			System.out.println();
		}}}
```



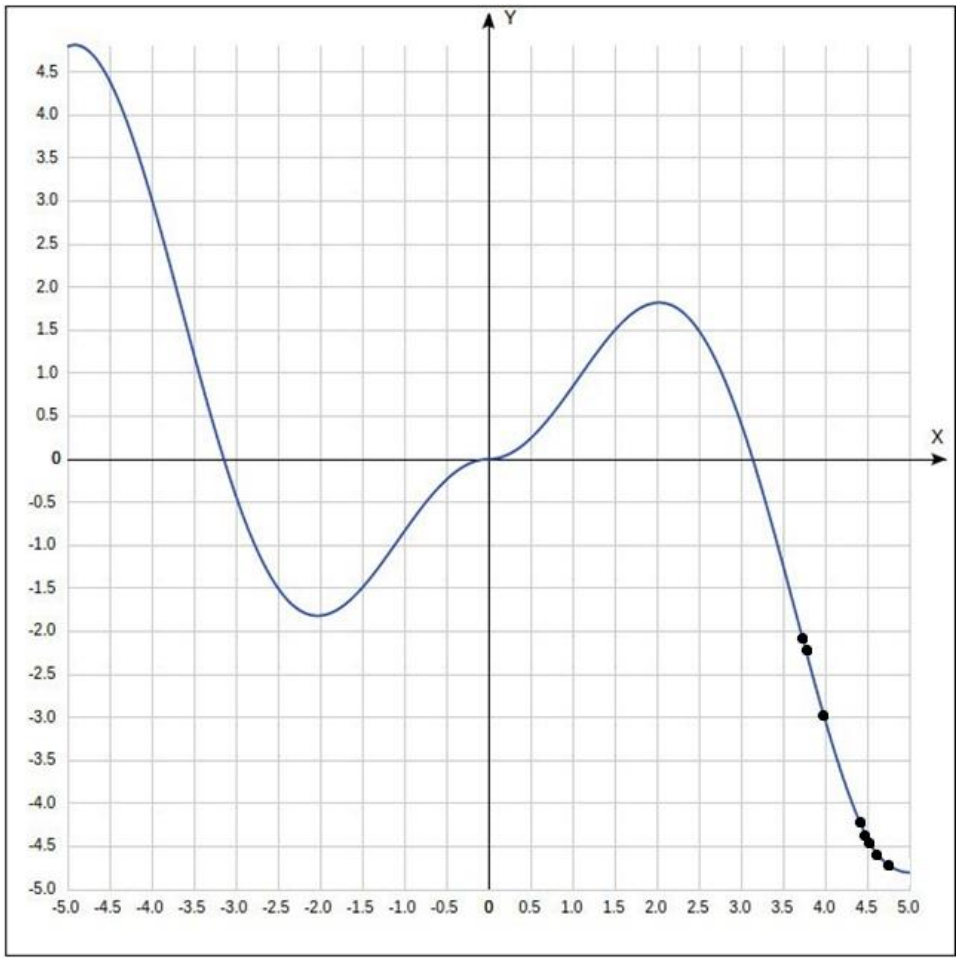
Picture 1. Minimum value of fitness function by iteration.



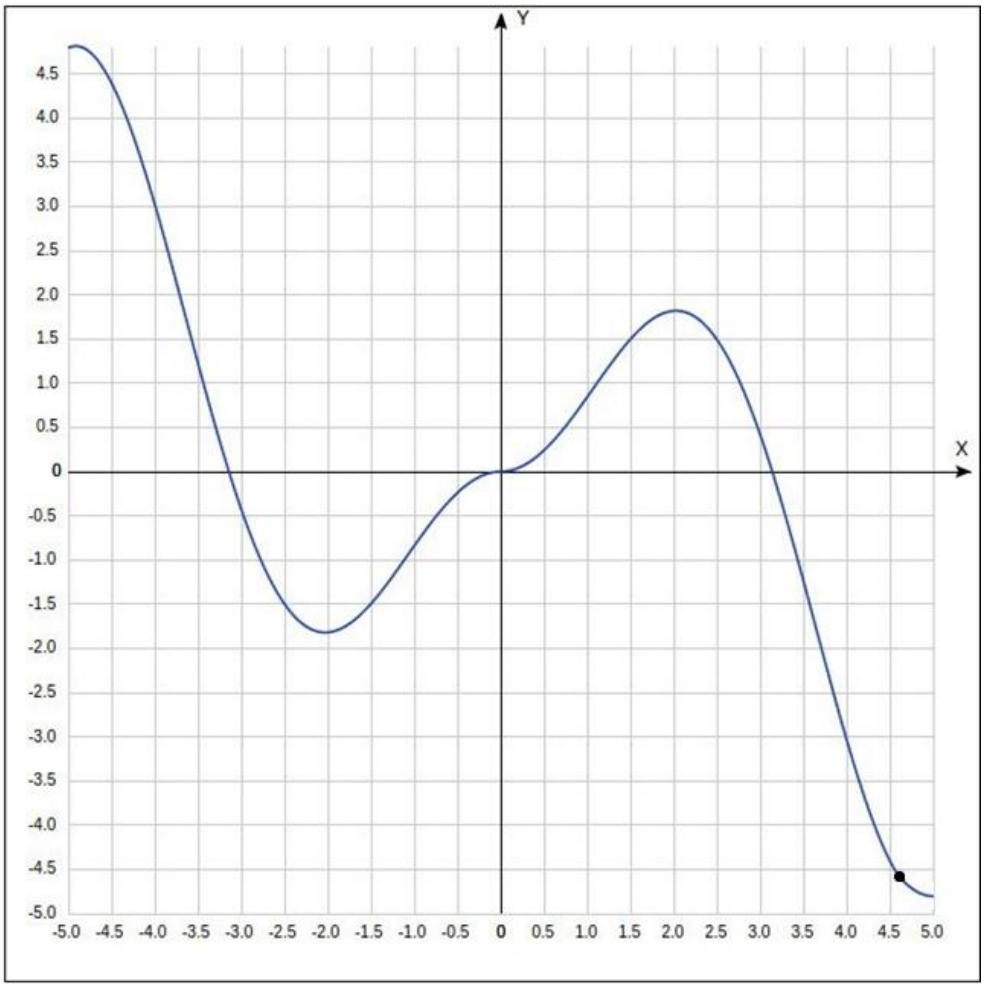
Picture 2. Average value of fitness function by iteration.



Picture 3. Iteration #5



Picture 4. Iteration #11



Picture 5. All dots in minimum (13 iteration)