MINISTRY OF EDUCATION OF REPUBLIC OF BELARUS
ESTABLISHMENT OF EDUCATION
"BREST STATE TECHNICHNICAL UNIVERSITY"

Practice work №1
«Evolutionary Computation»

Made by:
Ivan Bakunovich
Checked by:
Prof. Akira

2016

Task

1. Create 100 random binary-chromosomes each with 1000 genes.
2. Fitness is the number of “1” in one chromosome – the more the better.
3. Select 2 chromosomes at random from the better half of the population.
4. Create a child chromosome by a onr-point-crossover.
5. Give the child a mutation with a probability of 1/1000 = 0.001.
6. Repeat from 2. to 5. 100 times and create the next generation.
7. Repeat 6. until the fitness value does not change any more.
8. Show the result:

(1) Desplay the best chromosome in the 1st, an intermediate & final generation.
(2) Desplay the best and average fitness vs. generation.

Program code (C#)

File 1

```
namespace SIIT_Lab1
{
   class Program
   {
      static void Main(string[] args)
      {
         AllGenericStuffDoneHere obj = new AllGenericStuffDoneHere();
         obj.GenerateParent();
         while(obj.Uslovie() == false)
         {
            obj.CountOnes();
            obj.Delete_Bad_Genes();
            obj.picture();
            obj.Cross();
            obj.mutation();
            obj.Preparation();
         }
         Console.WriteLine("An excellent gene!\n");
      }
   }
}
```

File 2

```
namespace SIIT_Lab1
{
   class AllGenericStuffDoneHere
   {
      int[,] parent;
      int[,] child;
      List<int> max_generation_gen;
      List<int> average_generation_gen;
      int[] numof1_in_parent;
      int[,] fortime;
      Random rand;

      public void GenerateParent()
      {
         rand = new Random();
         parent = new int[100,1000];
```

```
    int j = 0;
    while(j < 100)
    {
        for (int i = 0; i < 1000; i++)
            parent[j, i] = rand.Next(0, 2);
        j++;
    }
    max_generation_gen = new List<int>();
    average_generation_gen = new List<int>();
}
// Count the number of 1 elements in every single gen
public void CountOnes()
{
    numof1_in_parent = new int[100];
    for (int i = 0; i < 100; i++)
    {
        int count = 0;
        for (int k = 0; k < 1000; k++)
            if (parent[i,k] == 1)
                count++;
        numof1_in_parent[i] = count;
    }
}
// delete the worst half of our parents
public void Delete_Bad_Genes()
{
    int[] keys = new int[100];
    for (int i = 0; i < 100; i++)
        keys[i] = i;
    Array.Sort(numof1_in_parent, keys);
    average_generation_gen.Add((int)numof1_in_parent.Average());
    fortime = new int[50,1000];
    for (int i = 99; i > 49; i--)
    {
        for (int j = 0; j < 1000; j++)
            fortime[99 - i, j] = parent[keys[i], j];
    }

    max_generation_gen.Add(numof1_in_parent[99]);
}

public void Cross()
{
    child = new int[100,1000];
    int p = 0;
    for (int i = 0; i < 50; i++)
    {
        int first_parent = rand.Next(0, 50);
        int second_parent = rand.Next(0, 50);
        int[] first_parent_mass = new int[1000];
        int[] second_parent_mass = new int[1000];
        for(int j = 0; j < 1000; j++)
        {
            first_parent_mass[j] = fortime[first_parent,j];
            second_parent_mass[j] = fortime[second_parent,j];
        }
        int num = rand.Next(1, 1000);
        for(int j = num; j < 1000; j++)
        {
            int k = first_parent_mass[j];
            first_parent_mass[j] = second_parent_mass[j];
            second_parent_mass[j] = k;
        }
        for(int k = 0; k < 1000; k++)
        {
            child[p, k] = first_parent_mass[k];
```

```
                child[p + 1, k] = second_parent_mass[k];
            }
            p += 2;
        }
    }

    public void mutation()
    {
        for(int i=0; i < 100; i++)
        {
            for (int j = 0; j < 1000; j++)
            {
                int num = rand.Next(0, 1000);
                if (num == 0)
                {
                    if (child[i,j] == 1) child[i,j] = 0;
                    else child[i,j] = 1;
                }
            }
        }
    }
    // make child become parents for next generation
    public void Preparation()
    {
        parent = child;
    }
    // condition of the end of the evolution
    public bool Uslovie()
    {

        int[] last20 = new int[20];
        if (max_generation_gen.Count < 20) return false;
        int k = max_generation_gen.Count;
        if (max_generation_gen[k - 1] == 1000) return true;
        return false;
    }
    // show out the result of every iteration
    public void picture()
    {
        int k = max_generation_gen.Count, k1 = average_generation_gen.Count;
        Console.WriteLine("Generation " + k);
        Console.WriteLine("Max = " + max_generation_gen[k - 1]);
        Console.WriteLine("Average = " + average_generation_gen[k1 - 1]);
    }
  }
}
```

Description

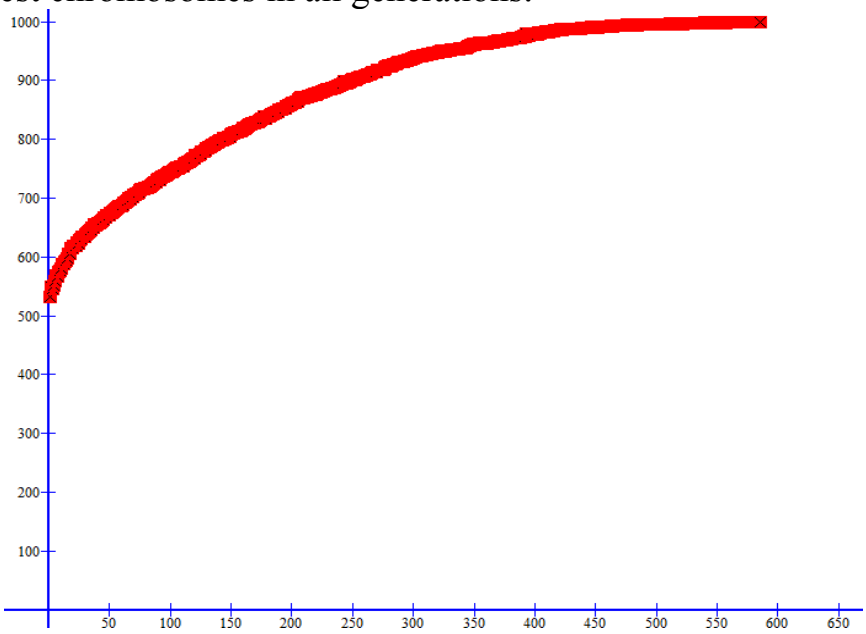
At first I Create 100 random binary-chromosomes each with 1000 genes. After that fitness of each chromosome was defined (it's the number of "1" elements in each chromosome – the more the better) and for making next population we choose 50 best chromosomes. After that were selected 2 random chromosomes from the better half of the population and created a child chromosomes by a one-point-crossover. Next step is mutation of children with the possibility 0,001. The cycle of one-point-crossover and mutation repeated 50 times and, as result, we create next generation. The whole cycle repeated until the fitness value does not change any more.

RESULTS

Best chromosomes (number of '1' elements in chromosome):
1st generation – 547;

Intermediate generation – 886;
Final chromosome – 1000.

Graphic of best chromosomes in all generations:



Graphic of average chromosomes in all generations: