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

Practice work №1
«Evolutionary Computation»
Subject: «All One Problem»

Made by:
Alexandra Letun
Checked by:
Pr. 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#)

```
using System.Collections.Generic;

namespace Exercise1
{
  public class Conditional
  {
    public List<int> populationMax = new List<int>();
    public bool Condit(int preview, int iter)
    {
      int count = 0;
      for (int i = iter; i < 10 + iter; i++)
      {
        if ((populationMax[i] - 1) <= preview && preview <= (populationMax[i] + 1))
        {
          count = 0;
        }
        else
          count++;
      }
      if (count == 0)
        return false;
      return true;
    }
  }
  class Program
  {
    static void Main(string[] args)
    {
      Conditional c = new Conditional();
      List<Population> population = new List<Population>();
      Population pop = new Population();
      int i = -1;
      int preview = 0;
      int average = 0;
      while (i <= 500)
      {
        i++;
        pop.Cross();
        pop.Mutation();
        preview = pop.GetMax();
        average = pop.GetAverage();
        using (System.IO.StreamWriter file = new System.IO.StreamWriter(@"Maximum.txt", true))
        {
          file.WriteLine(preview);
        }
        using (System.IO.StreamWriter file = new System.IO.StreamWriter(@"Average.txt", true))
        {
          file.WriteLine(average);
        }
        pop.GetAverage();
        c.populationMax.Add(preview);
      }
      int i1 = 0;
      while (c.Condit(preview, i1))
      {
        pop.Cross();
        pop.Mutation();
        preview = pop.GetMax();
        average = pop.GetAverage();
        using (System.IO.StreamWriter file = new System.IO.StreamWriter(@"Maximum.txt", true))
        {
          file.WriteLine(preview);
```

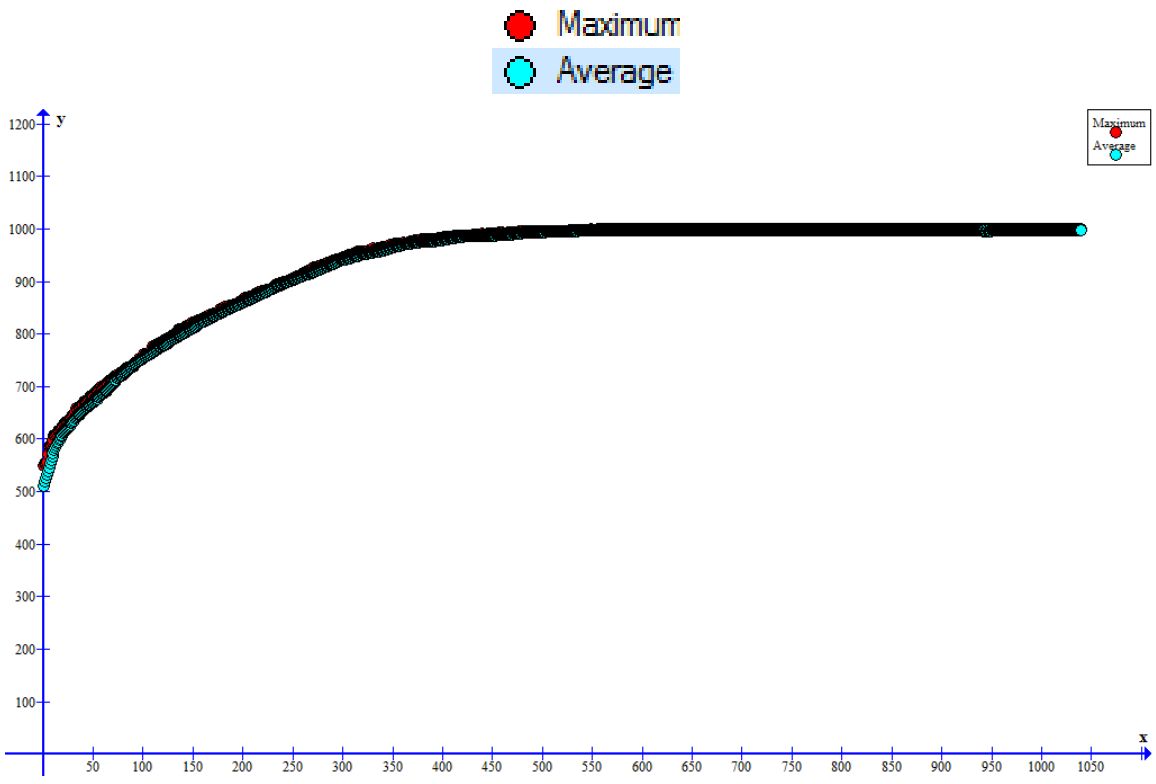
```
            }
            using (System.IO.StreamWriter file = new System.IO.StreamWriter(@"Average.txt", true))
            {
                file.WriteLine(average);
            }
            c.populationMax.Add((pop.GetMax()));
            i1++;
        }
        using (System.IO.StreamWriter file = new System.IO.StreamWriter(@"Maximum.txt", true))
        {
            file.WriteLine(i + i1 - 1);
        }
    }
  }
}

using System;
using System.Linq;

namespace Exercise1
{
  public class Population
  {
    Random rand = new Random();
    const int countChromosomes = 100;
    const int countGenes = 1000;
    int[,] parents = new int[countChromosomes, countGenes];
    int[,] childs = new int[countChromosomes, countGenes];
    int[,] halfOfPopulation = new int[countChromosomes / 2, countGenes];
    int[] sum = new int[countChromosomes];
    int max;
    int average;
    int[] keys = new int[countChromosomes];
    public Population()
    {
      int[] genes = new int[countGenes];
      for (int i = 0; i < countChromosomes; i++)
      {
        int s = 0;
        for (int j = 0; j < countGenes; j++)
        {
          genes[j] = rand.Next(0, 2);
          if (genes[j] == 1)
            s += 1;
          parents[i, j] = genes[j];
        }
        sum[i] = s;
      }
    }
    public void Cross()
    {
      int[,] reserv = new int[countChromosomes, countGenes];
      for (int i = 0; i < countChromosomes; i++)
        keys[i] = i;
      Array.Sort(sum, keys);
      for (int i = 0, n = countChromosomes - 1; i < countChromosomes / 2; i++, n--)
      {
        for (int j = 0; j < countGenes; j++)
        {
          halfOfPopulation[i, j] = parents[keys[n], j];
        }
      }
      int k = 0;
      for (int i = 0; i < countChromosomes / 2; i++)
      {
        int a = rand.Next(0, countChromosomes / 2);
        int b = rand.Next(0, countChromosomes / 2);
        for (int j = 0; j < countGenes; j++)
        {
          childs[0, j] = halfOfPopulation[a, j];
          childs[1, j] = halfOfPopulation[b, j];
          reserv[0, j] = childs[0, j];
        }
        int c = rand.Next(0, countGenes);
        for (int j = countGenes - c; j < countGenes; j++)
        {
          childs[0, j] = childs[1, j];
          childs[1, j] = reserv[0, j];
        }
        for (int j = 0; j < countGenes; j++)
        {
          parents[k, j] = childs[0, j];
          parents[k+1, j] = childs[1, j];
        }
        k += 2;
      }
    }
    public void Mutation()
    {
      for (int i = 0; i < countChromosomes; i++)
      {
        int s = 0;
        for (int j = 0; j < countGenes; j++)
```

```
            {
                int value = rand.Next(0, 1000);
                if (value == 0)
                    if (parents[i, j] == 1)
                        parents[i, j] = 0;
                    else
                        parents[i, j] = 1;
                if (parents[i, j] == 1)
                    s += 1;
            }
            sum[i] = s;
        }
    }
    public int GetMax()
    {
        max = sum.Max();
        return max;
    }
    public int GetAverage()
    {
        average = (int)sum.Average();
        return average;
    }
  }
}
```

Results

Maximum = 1000
Average = 997
Iterations = 1039

Graph (intermediate & maximum) - red
Graph (intermediate average) – blue



Describtion

At first i Create 100 random binary-chromosomes each with 1000 genes. Then I defined fitness - is the number of “1” in one chromosome – the more the better. After that selected 2 chromosomes at random from the better half of the population and created a child chromosome by a onr-point-crossover. Then I gived the child a mutation with a probability of 1/1000 = 0.001 and repeated cycle 100 times and created the next generation. After that repeated until the fitness value does not change any more.