

1 exercise – Roman Rudski

Conditions

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 by roulette method
4. Create a child chromosome by a uniform-crossover.
5. Repeat from 2. to 5. 100 times and create the next generation.
6. Repeat 6. until the fitness value does not change any more.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static Genetic.Constants;
namespace Genetic {
public static class Constants {
public const int CHROM= 100;
public const int GENES = 1000;
public const int GENERAT = 1000;
public const int FINTESS = 100;
public const int MUT = 10000;
public const int MUTATION = 10;
} }
namespace Genetic {
public class Chromosome : IComparable {
public List Genes { get; set; }
public double Fitness { get; set; }
public double Probability { get; set; }
//for roulette method
public int Count { get; set; }
readonly Random _rnd;
public Chromosome(Random rnd) { _rnd = rnd; Genes = new List();
for (int i = 0; i < GENES_NUMBER; i++)
Genes.Add(_rnd.Next(2));
Fitness = Genes.Sum();
} public int CompareTo(Chromosome compareChromosome)
{ if (compareChromosome == null) return -1;
else return compareChromosome.Fitness.CompareTo(Fitness); }
public Chromosome CreateChild(Chromosome parent2)
{ int chooseParent; Chromosome child = new Chromosome(_rnd);
child.Genes.Clear();
for(int i = 0; i < GENES_NUMBER; i++)
{ chooseParent = _rnd.Next(2);
if (chooseParent == 1) child.Genes.Add(Genes.ElementAt(i));
else child.Genes.Add(parent2.Genes.ElementAt(i)); }
child.Fitness = child.Genes.Sum(); return child; } } }
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static Genetic.Constants;
namespace Genetic {
class Program { static void Main(string[] args) { List chromosomes = new List();
List childChromosomes = new List();
int generationCount = 0; // increment if fitness hasn't change
int fitnessChangeCount = 0;
double parentsFitness, childsFitness;
int i;
Random rnd = new Random(DateTime.Now.TimeOfDay.Milliseconds);
//1-st generation
for (i = 0; i < CHROMOSOMES_NUMBER; i++)
chromosomes.Add(new Chromosome(rnd));
double fitnessSumm = chromosomes.Select(x => x.Fitness).Sum();
foreach (var chr in chromosomes) chr.Probability = chr.Fitness / fitnessSumm; chromosomes.Sort();
//1-st generation
Console.WriteLine("Calculating...");
Chromosome mother, father;
while (generationCount != GENERATIONS_NUMBER) {
chromosomes.Sort();
parentsFitness = chromosomes.Average(x => x.Fitness);
if (generationCount % 100 == 0)
Console.WriteLine($"Generation {generationCount}: - Average fitness: {parentsFitness}");
for (i = 0; i < CHROMOSOMES_NUMBER; i++) {
mother = ChooseParent(chromosomes, rnd);
father = ChooseParent(chromosomes, rnd);
childChromosomes.Add(father.CreateChild(mother));

childsFitness = childChromosomes.Average(x => x.Fitness);
if ((int)childsFitness == (int)parentsFitness)
fitnessChangeCount++; chromosomes.Clear();
chromosomes.AddRange(childChromosomes);
childChromosomes.Clear();
fitnessSumm = chromosomes.Select(x => x.Fitness).Sum();
foreach (var chr in chromosomes) chr.Probability = chr.Fitness / fitnessSumm; generationCount++; } }
public static Chromosome ChooseParent(List chromosomes, Random rnd) {
double probabilitySum, parentProbability;
Chromosome parent = null;
probabilitySum = 0.0;
parentProbability = rnd.NextDouble();
for (int j = 0; j < GENES_NUMBER; j++) {
if (probabilitySum < parentProbability) probabilitySum += chromosomes.ElementAt(j).Probability;
else { parent = chromosomes.ElementAt(j - 1); break; } } return parent; } }

```

Graph

Roulette selection and uniform crossover

Population:800

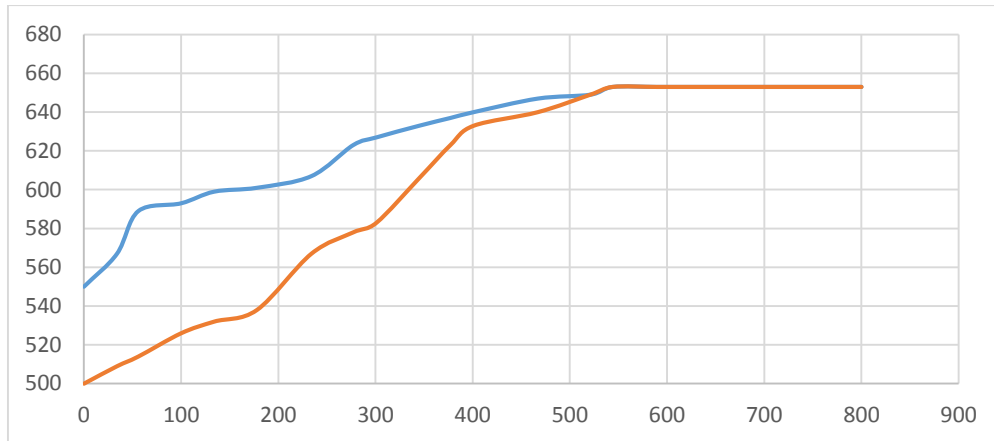
X:population

Y:Fitnes

Max fitnes : 652

Blue- best

red- average



roulette selection and one-point crossover.

Population: 800

X : population

Y: fitness

Red - best

Blue-average

MAX FINTESS:995

