Министерство науки и высшего образования Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Институт прикладной математики и компьютерных наук

**ОТЧЕТ**

по лабораторной работе №2

по теме «Применение генетического алгоритма для решения задач оптимизации»

дисциплины «Интеллектуальные системы»

Выполнили студенты:

Павловский Владимир,

Эккерманн Илья,

Качан Павел,

Пестунов Игорь,

Раданович (Байгужинова) Алина

Томск – 2021

**Цель работы:** разработать программу для нахождения минимума функции y(x) = , через вещественное кодирование.

**Постановка задачи:**

1.Провести оценку популяции с помощью функции приспособленности.

2.Определить критерий оценки особей, задав функцию приспособленности.

3.Реализовать селекцию особей для скрещивания.

4.Выполнить операцию скрещивания, отобранных в результате селекции особей с применением оператора кроссинговера.

5.Реализовать оператор мутации.

6.Определить параметры генетического алгоритма.

**Метод решения задачи:**

При решении задачи был использован генетический алгоритм. В начале популяция формируется случайным образом.

Для оценки качества закодированных решений использовали функцию приспособленности. Далее, по результатам оценки особей, наиболее приспособленные из них, выбирались для скрещивания (селекцией усечения).

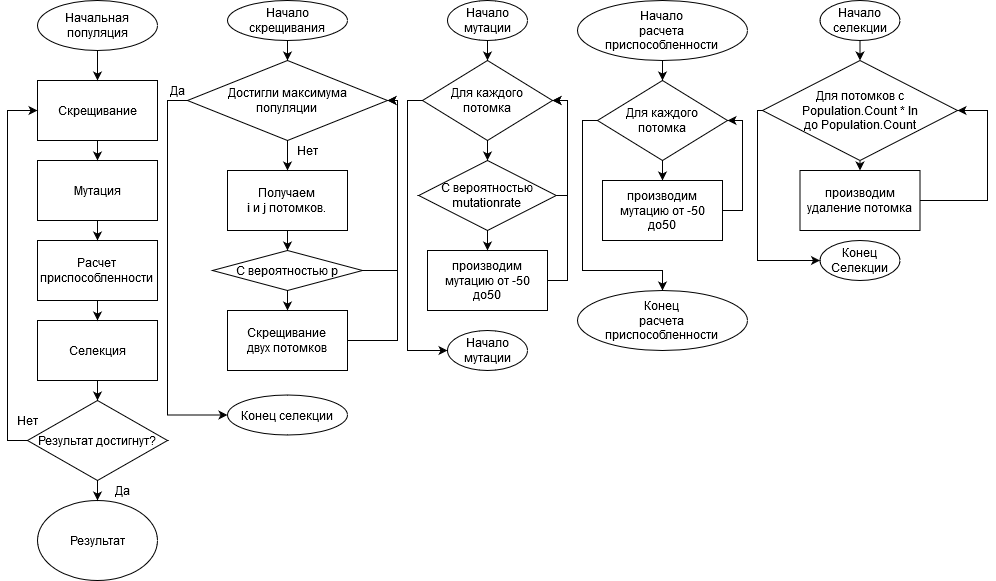
В результате, было создано потомство, посредством применения генетического оператора кроссинговера (вещественное кодирование).

Созданные потомки сформировали новую популяцию, причем часть потомков мутировали, что выражается в случайном изменении их генотипов (в результате обмена хромосомной информацией между родительскими особями).

**Параметры Генетического Анализа:**

1. Размер популяции: 1000.
2. Тип селекции: усечением.
3. Тип оператора мутации: вещественное кодирование.
4. Мутации: 40%.
5. Разрыв поколений: 0.3
6. Коэффициент элитарности: нет.

**Структурная схема алгоритма**

****

**Листинг программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab2

{

class Program

{

static GenAlg GA = new GenAlg();

static void Main(string[] args)

{

GA.Func = (x) => { return x \* x + 4; };

GA.Init();

var result = GA.Process();

Console.WriteLine("iter: {0}", GA.wag);

Console.WriteLine("Rezyltat: x = {0}; y = {1}.", result.X, result.Y);

Console.ReadLine();

Console.ReadLine();

}

}

}

namespace Lab2

{

public class Potomok: IComparable

{

public double X;

public double Y;

public int Compare(Potomok x, Potomok y)

{

var dlt = x.Y - y.Y;

if (dlt > 0)

return 1;

if (dlt < 0)

return -1;

return 0;

}

public int CompareTo(object obj)

{

var y = obj as Potomok;

var dlt = Y - y.Y;

if (dlt > 0)

return 1;

if (dlt < 0)

return -1;

return 0;

}

}

}

namespace Lab2

{

public class GenAlg

{

readonly int razmer = 1000; //размер популяции

readonly int max\_iter = 100; //максимум итераций

readonly double mp = 0.4; //мутация

readonly double ln = 0.3; //порог отсечения

readonly double p = 0.7; //вероятность скрещивания

readonly double dlt = 0.0001;

Random rnd = new Random((int)DateTime.Now.Ticks);

public int wag { get; private set; }

public Func<double, double> Func;

public List<Potomok> Popylyaciya = new List<Potomok>();

private double func(double x)

{

return Func(x);

}

public void Init()

{

GenPop();

Ocenka();

}

public Potomok Process()

{

int i = 0;

for (; i < max\_iter; i++)

{

Skrewivanie();

Mytaciya();

Ocenka();

Selekciya();

if (Math.Abs(Popylyaciya.First().Y - Popylyaciya.Last().Y) <= dlt)

break;

}

wag = i;

return Popylyaciya.First();

}

public void GenPop()

{

for(int i = 0; i < razmer; i++)

{

var potom = new Potomok()

{

X = (rnd.NextDouble() - 0.5) \* 100000

};

Popylyaciya.Add(potom);

}

}

public void Ocenka() //оценка популяции

{

foreach(var potom in Popylyaciya)

{

potom.Y = func(potom.X);

}

Popylyaciya.Sort();

}

public void Selekciya() //селекция

{

for(int i = (int)(Popylyaciya.Count \* ln); i < Popylyaciya.Count; i++)

Popylyaciya.RemoveAt(i);

}

public void Skrewivanie() //скрещивание

{

var old\_razmer = Popylyaciya.Count;

while (Popylyaciya.Count < razmer)

{

int i = (int)(rnd.NextDouble() \* old\_razmer);

int j = (int)(rnd.NextDouble() \* old\_razmer);

if(p > rnd.NextDouble())

{

var deti = SkrewPot(Popylyaciya[i], Popylyaciya[j]);

Popylyaciya.Add(deti.Item1);

Popylyaciya.Add(deti.Item2);

}

}

}

public (Potomok, Potomok) SkrewPot(Potomok x, Potomok y)

{

var lambda = rnd.NextDouble();

var zx = new Potomok() { X = lambda \* x.X + (1 - lambda) \* y.X };

var zy = new Potomok() { X = lambda \* y.X + (1 - lambda) \* x.X };

return (zx, zy);

}

public void Mytaciya() //мутация

{

foreach(var potom in Popylyaciya)

{

if(mp > rnd.NextDouble())

{

potom.X += (rnd.NextDouble() - 0.5) \* 100;

}

}

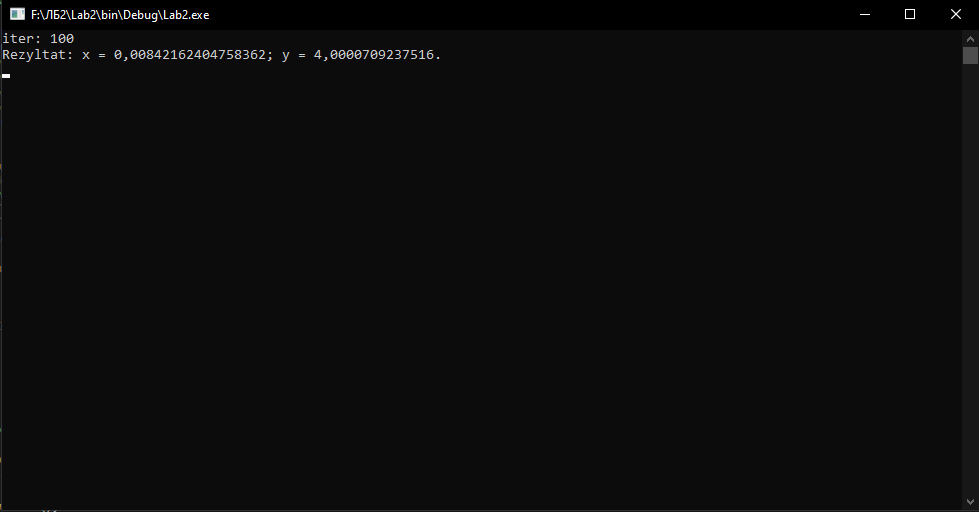
}

}

}

**Результат работы алгоритма**

При ln = 0.3

****

**Ссылка на GitHub:**