# ГОУ ВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

### Кафедра АСУ

### 

Отчёт

о лабораторной работе №5

по курсу: «Оптимизация многомерных функций с помощью ЭС»

# Вариант «2»

Выполнил:

ст. гр. ИСм-22

Мариничев И.И

Проверила:

Васяева Т.А

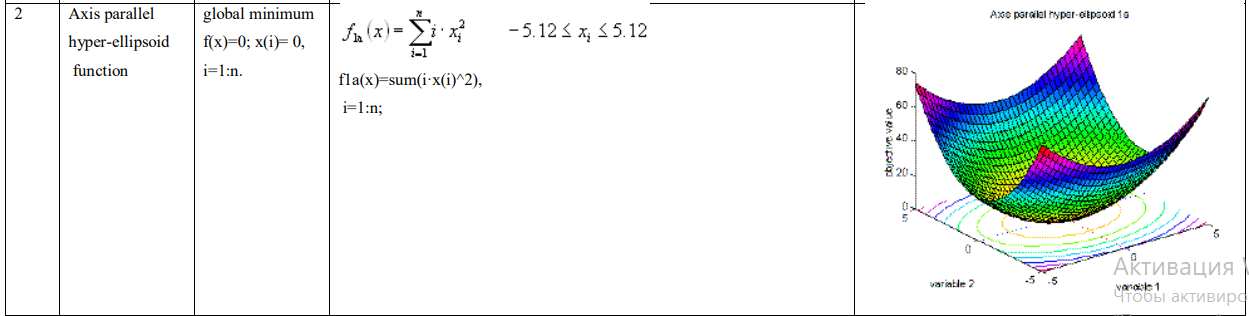
# 

# Донецк 2022

**Ход работы:**

1. Разработать ЭС нахождения оптимума функции из лабораторной работы 2.
2. Выполнить программную реализацию разработанного алгоритма на языке высокого уровня. Рекомендованный язык Python с применением DEAP. Предусмотреть возможность просмотра процесса поиска решения.
3. Для n=2 вывести на экран график данной функции с указанием найденного экстремума. Предусмотреть возможность пошагового просмотра процесса поиска решения (с выводом всех точек популяции и лучшего решения другим цветом).
4. Исследовать зависимость числа поколений (генераций), точности нахождения решения от основных параметров ЭС. Сравнить результат с полученным в лабораторной работе №2.

*Задание:*



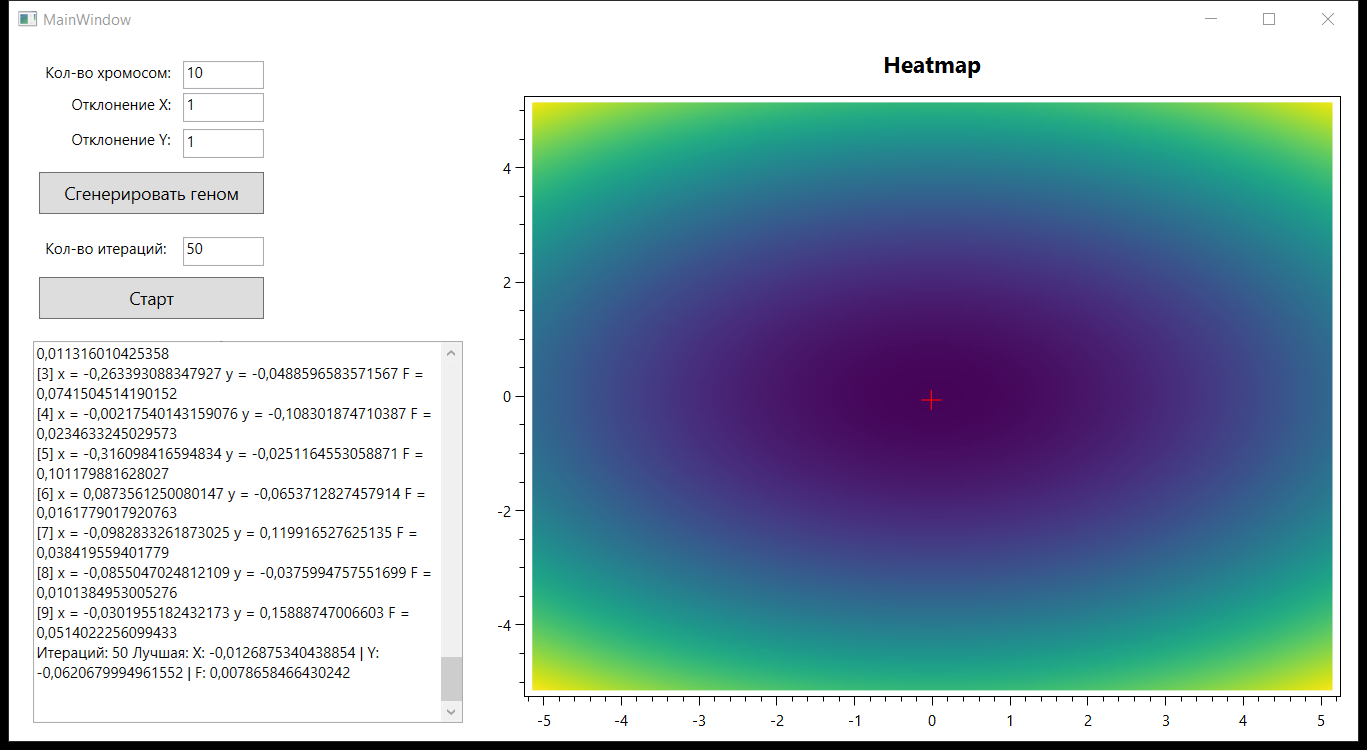
Для реализации задания был выбран алгоритм (1+1), он же «Двукратная эволюционная стратегия».

Принцип работы алгоритма, следующий:

В цикле перебираются все элементы популяции. Для каждого элемента создаётся потомок, к значениям X и Y которого прибавляется (или вычитается) случайное число в неком заданном диапазоне (для данной лабораторной был взят диапазон [0; 1]) и вычисляется значение его целевой функции. Если значение его ЦФ лучше чем значение у родительского элемента, то родительский элемент заменяется потомком, иначе потомок уничтожается.

**Проверка:**

Для проверки были взяты следующие параметры: кол-во хромосом = 10; диапазон отклонения(диапазон из которого берётся случайное число для вычисления параметров потомка) для X и Y = 1 ([0;1]).



Исходная популяция

[0] x = 0,635545852480245 y = 4,10599293661583 F = 34,122274521683

[1] x = -1,52526675533749 y = 3,52348882600828 F = 27,1563856889481

[2] x = 2,3740973555176 y = 0,12871939426694 F = 5,66947561839658

[3] x = -1,40804036444427 y = 2,86262199773575 F = 18,3717870717456

[4] x = 3,53831046984452 y = -3,21830165297645 F = 33,2345720401133

[5] x = 3,93797867719921 y = 3,51172300263854 F = 40,1720729565969

[6] x = 3,74785290177346 y = 1,96125796170964 F = 21,7394669580706

[7] x = 1,97598332735523 y = -2,38218496195142 F = 15,2541204958809

[8] x = -3,1213772573701 y = -0,652152564829287 F = 10,5936019184541

[9] x = -1,8891733822083 y = 3,74917302067819 F = 31,6815727460068

Среднее значение: 21,6262877760682

Минимальное значение: 5,66947561839658

Финальный геном

[0] x = -0,0126875340438854 y = -0,0620679994961552 F = 0,0078658466430242

[1] x = -0,256473050292802 y = -0,0918977781440578 F = 0,0826688287821228

[2] x = 0,049718984593506 y = 0,0664982443239996 F = 0,011316010425358

[3] x = -0,263393088347927 y = -0,0488596583571567 F = 0,0741504514190152

[4] x = -0,00217540143159076 y = -0,108301874710387 F = 0,0234633245029573

[5] x = -0,316098416594834 y = -0,0251164553058871 F = 0,101179881628027

[6] x = 0,0873561250080147 y = -0,0653712827457914 F = 0,0161779017920763

[7] x = -0,0982833261873025 y = 0,119916527625135 F = 0,038419559401779

[8] x = -0,0855047024812109 y = -0,0375994757551699 F = 0,0101384953005276

[9] x = -0,0301955182432173 y = 0,15888747006603 F = 0,0514022256099433

Среднее значение: 0,041678252550483

Минимальное значение: 0,0078658466430242

Итераций: 50 Лучшая: X: -0,0126875340438854 | Y: -0,0620679994961552 | F: 0,0078658466430242

Известный оптимум: F(0;0) = 0.

Погрешность: 0 - 0,0078658466430242 = -0,0078658466430242

Вывод: уже к 50й итерации алгоритмом было найдено решение с погрешностью в 0,007. Это гораздо быстрее и точнее, чем использование генетического алгоритма. Отсюда можем сделать вывод: для задач такого типа, где необходимо найти экстремум функции с ОДНИМ локальным минимумом, Метод Эволюционных стратегий подходит лучше генетического алгоритма.

Код программы:

Windows forms:

using Newtonsoft.Json;

using OxyPlot;

using OxyPlot.Axes;

using OxyPlot.Series;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace gen\_2

{

public partial class MainWindow : Window

{

private PlotModel model;

private int sizeMap = 200;

public double startPoint = -5.12, endPoint = 5.12;

private ScatterSeries point = new ScatterSeries();

GeneticAlgorithm geneticAlgorithm = new GeneticAlgorithm();

public MainWindow()

{

InitializeComponent();

initHeatMap();

point.ColorAxisKey = "ColorAxis";

point.MarkerSize = 8;

point.MarkerType = MarkerType.Plus;

}

private void initHeatMap()

{

model = new PlotModel { Title = "Heatmap" };

model.Axes.Add(new LinearColorAxis { Palette = OxyPalettes.Viridis(255) });

var data = new double[sizeMap, sizeMap];

double xValue = startPoint, yValue = startPoint, max = endPoint;

double det = (Math.Abs(xValue) + max) / sizeMap;

for (int y = 0; y < sizeMap; ++y)

{

xValue = startPoint;

for (int x = 0; x < sizeMap; ++x)

{

data[x, y] = getFunction(xValue, yValue);

xValue += det;

}

yValue += det;

}

var heatMapSeries = new HeatMapSeries

{

X0 = startPoint,

X1 = endPoint,

Y0 = startPoint,

Y1 = endPoint,

Interpolate = true,

RenderMethod = HeatMapRenderMethod.Bitmap,

Data = data

};

var axis1 = new LinearColorAxis();

axis1.Key = "ColorAxis";

axis1.IsAxisVisible = true;

model.Axes.Add(axis1);

axis1.Palette.Colors.Clear();

axis1.Palette.Colors.Add(OxyColor.FromArgb((byte)255, 255, 0, 0));

model.Series.Add(heatMapSeries);

testPlotView.Model = model;

}

public double getFunction(double x, double y)

{

return (x \* x) + (2 \* y \* y);

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

int kolChromosome = Convert.ToInt32(textBlock2.Text);

geneticAlgorithm.CreateGenome(kolChromosome, startPoint, endPoint, startPoint, endPoint);

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

int kolIteration = Convert.ToInt32(textBlock1.Text);

double xMut = Convert.ToDouble(textBlock3.Text);

double yMut = Convert.ToDouble(textBlock4.Text);

Chromosome components = geneticAlgorithm.Evolution(kolIteration, xMut, yMut);

double x = components.xVal;

double y = components.yVal;

initHeatMap();

var random = new Random();

textBox1.Clear();

textBox1.AppendText(geneticAlgorithm.TextAll + Environment.NewLine);

while (model.Series.Count > 1)

{

model.Series.Remove(model.Series.Last());

}

model.Series.Add(components.Point);

testPlotView.Model.InvalidatePlot(true);

}

}

}

Class1:

using OxyPlot;

using OxyPlot.Series;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace gen\_2

{

class Chromosome

{

public double xVal;

public double yVal;

public double funcVal;

private ScatterSeries point;

public Chromosome()

{

}

public Chromosome(double xval, double yval, double fun)

{

xVal = xval;

yVal = yval;

funcVal = fun;

}

public Chromosome(Chromosome c)

{

xVal = c.xVal;

yVal = c.yVal;

funcVal = c.funcVal;

}

public ScatterSeries Point

{

get

{

this.point = new ScatterSeries();

this.point.ColorAxisKey = "ColorAxis";

this.point.MarkerSize = 8;

this.point.MarkerType = MarkerType.Plus;

point.Points.Add(new ScatterPoint(xVal, yVal, double.NaN, 1));

return point;

}

set => point = value;

}

public void ReCreateFull(double xval, double yval, double fun)

{

xVal = xval;

yVal = yval;

funcVal = fun;

}

}

class FitnessFunc

{

public static readonly double minInt = -5.12;

public static readonly double maxInt = 5.12;

public static double Func(double x, double y)

{

return (x \* x) + (2 \* y \* y);

}

}

class GeneticAlgorithm

{

private Random random = new Random();

private StringBuilder textAll = new StringBuilder();

List<Chromosome> genome;

public string TextAll { get { return textAll.ToString(); } }

public GeneticAlgorithm() { }

public Chromosome Evolution(int iterationsCount, double xMut, double yMut)

{

textAll.Clear();

double minInterval = FitnessFunc.minInt;

double maxInterval = FitnessFunc.maxInt;

int currentCount = 0;

textAll.Append("Исходная популяция" + Environment.NewLine);

printGenome(genome);

Chromosome best = genome.Find(g => g.funcVal == genome.Min(v => v.funcVal));

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

{

foreach (Chromosome chr in genome)

{

Chromosome c = new Chromosome();

if (random.Next(0, 2) == 0)

c.xVal = chr.xVal + random.NextDouble() \* xMut;

else

c.xVal = chr.xVal - random.NextDouble() \* xMut;

if (random.Next(0, 2) == 0)

c.yVal = chr.yVal + random.NextDouble() \* yMut;

else

c.yVal = chr.yVal - random.NextDouble() \* yMut;

c.funcVal = FitnessFunc.Func(c.xVal, c.yVal);

if (c.funcVal < chr.funcVal)

{

chr.xVal = c.xVal;

chr.yVal = c.yVal;

chr.funcVal = c.funcVal;

}

}

//textAll.Append("сокращённая популяция" + Environment.NewLine);

//printGenome(genome);

textAll.Append("Среднее значение: " + genome.Average(g => g.funcVal) + Environment.NewLine + "Минимальное значение: " + genome.Min(g => g.funcVal) + Environment.NewLine);

currentCount++;

best = genome.Find(g => g.funcVal == genome.Min(v => v.funcVal));

}

textAll.Append("Финальный геном" + Environment.NewLine);

printGenome(genome);

textAll.Append("Итераций: " + currentCount + " Лучшая: " + "X: " + best.xVal + " | " + "Y: " + best.yVal + " | F: " + best.funcVal + Environment.NewLine);

return best;

}

//private void Reproduction()

//{

// Chromosome cr;

// for (int i = 0; i < genome.Count; i++)

// {

// for (int j = 0; j < genome.Count; j++)

// {

// if (genome[i].funcVal < genome[j].funcVal)

// {

// cr = genome[i];

// genome[i] = genome[j];

// genome[j] = cr;

// }

// }

// }

//}

public void CreateGenome(int count, double xMin, double xMax, double yMin, double yMax)

{

genome = new List<Chromosome>();

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

{

genome.Add(new Chromosome((random.NextDouble() \* (xMax - xMin) + xMin), (random.NextDouble() \* (yMax - yMin) + yMin), 0));

genome[i].funcVal = FitnessFunc.Func(genome[i].xVal, genome[i].yVal);

}

}

public void printGenome(List<Chromosome> genome)

{

int i = 0;

foreach (Chromosome c in genome)

{

textAll.Append("[" + i++ + "] " + "x = " + c.xVal + " y = " + c.yVal + " F = " + c.funcVal + Environment.NewLine);

}

}

}

}