**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления

Кафедра: Информационная безопасность (ИУ8)

**Методы оптимизации**

**Лабораторная работа №4 на тему:**

«Решение многокритериальной оптимизации»

Вариант 10

**Преподаватель:**

Коннова Н.С.

**Студент**:

Казакова М.К.

**Группа:**

ИУ8-34

Москва 2023

**Цель работы**

Изучить постановку задачи многокритериальной оптимизации (МКО); овладеть навыками решения задач МКО с помощью различных методов, выполнить сравнительный анализ результатов, полученных при помощи разных методов

**Постановка задачи**

Выбрать лучшую из альтернатив решения предложенной задачи по варианту из табл. с точки зрения указанных критериев следующими методами:

1) заменой критериев ограничениями;

2) формированием и сужением множества Парето;

3) методом взвешивания и объединения критериев;

4) методом анализа иерархий

**Ход работы**

1. **Вектор весов критериев**

|  |  |  |  |
| --- | --- | --- | --- |
| Цена | Удалённость | Пляж | Достопримечательности |
| 8 | 3 | 7 | 6 |

Нормализовав, получим вектор (0.36, 0.05, 0,32, 0.27)

1. **Метод замены критериев ограничениями**

Составим матрицу А оценок для альтернатив.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Цена | Удалённость | Пляж | Достопримечательности |
| А. Испания | 5 | 6 | 4 | 2 |
| B. Турция | 6 | 7 | 3 | 5 |
| C. Куба | 3 | 2 | 9 | 4 |
| D. Индонезия | 2 | 1 | 10 | 7 |

В данном случае под критерием «цена» подразумевается «дешевизна», т.е. чем ниже цена, тем больше баллов.

Выберем в качестве главного критерия цену (критерий 1).

Установим минимально допустимые уровни для остальных критериев:

Допустимое давление не менее

Долговечность не менее

Внешний вид не менее

Проведём нормировку матрицы (кроме столбца главного критерия) по формуле:

где и – минимальное и максимальное значение в столбце соответственно.

Нормированная матрица А:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| A | 5 | 0.83 | 0.14 | 1 |
| B | 6 | 1 | 0 | 0 |
| C | 3 | 0.17 | 0.86 | 0.38 |
| D | 2 | 0 | 1 | 0.25 |

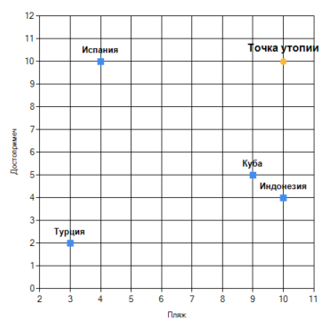
При заданных условиях приемлемыми являются следующие решения:

альтернатива А – Испания.

1. **Формирование и сужения множества Парето**

Выберем в качестве главных критериев для данного метода «Пляж» и «Достопримечательности». «Пляж» – по оси х, «Достопримечательности» – по оси у. Сформируем множество Парето графическим методом (см. рис. 1). Оба критерия максимизируются, поэтому точка утопии находится в

правом верхнем углу графика (10; 10).

******

*Рис. 1*

Рассчитаем Евклидово расстояния:

A.

B.

C.

D.

Исходя из графика можно сказать, что Евклидово расстояние до

точки утопии минимально для варианта C (Куба). А значит,

альтернатива C оптимальна.

1. **Взвешивание и объединение критериев**

Составим матрицу рейтингов альтернатив по критериям, используя шкалу 1÷10:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| A | 5 | 6 | 4 | 2 |
| B | 6 | 7 | 3 | 5 |
| C | 3 | 2 | 9 | 4 |
| D | 2 | 1 | 10 | 7 |

Нормализуем её:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| A | 0.31 | 0.38 | 0.15 | 0.48 |
| B | 0.38 | 0.44 | 0.12 | 0.10 |
| C | 0.19 | 0.12 | 0.35 | 0.24 |
| D | 0.12 | 0.06 | 0.38 | 0.19 |

Составим экспертную оценку критериев (по методу попарного сравнения):

Получим вектор весов критериев:

Нормализуем его и получим

Умножим нормализованную матрицу на нормализованный вектор весов критериев и получим значения объединенного критерия для всех альтернатив:

Как видно из полученной интегральной оценки, наиболее приемлемой является альтернатива A – Куба.

1. **Метод анализа иерархий:**

Для каждого из критериев составим и нормализуем матрицу попарного сравнения альтернатив:

• Цена

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | Сумма по строке | Нормированная сумма по строке |
| A | 1 | 0.83 | 1.67 | 2.50 | 11 | 0.31 |
| B | 1.20 | 1 | 2 | 3 | 13.2 | 0.38 |
| C | 0.60 | 0.50 | 1 | 1.50 | 6.6 | 0.19 |
| D | 0.40 | 0.33 | 0.67 | 1 | 4.4 | 0.13 |

Коэффициент согласованности: 0,4

• Удалённость

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | Сумма по строке | Нормированная сумма по строке |
| A | 1 | 0.86 | 3 | 6 | 16.86 | 0.38 |
| B | 1.17 | 1 | 3.50 | 7 | 19.67 | 0.44 |
| C | 0.33 | 0.29 | 1 | 2 | 5.62 | 0.13 |
| D | 0.17 | 0.14 | 0.50 | 1 | 2.81 | 0.06 |

Коэффициент согласованности: 1,5

• Пляж

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | Сумма по строке | Нормированная сумма по строке |
| A | 1 | 1.33 | 0.44 | 0.40 | 7.18 | 0.15 |
| B | 0.75 | 1 | 0.33 | 0.30 | 5.38 | 0.12 |
| C | 2.25 | 3 | 1 | 0.90 | 16.15 | 0.35 |
| D | 2.50 | 3.33 | 1.11 | 1 | 17.94 | 0.38 |

Коэффициент согласованности: 0,1

• Достопримечательности

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | Сумма по строке | Нормированная сумма по строке |
| A | 1 | 5 | 2 | 2.50 | 20.5 | 0.48 |
| B | 0.20 | 1 | 0.40 | 0.50 | 4.1 | 0.1 |
| C | 0.50 | 2.50 | 1 | 1.25 | 10.25 | 0.24 |
| D | 0.40 | 2 | 0.80 | 1 | 8.2 | 0.19 |

Коэффициент согласованности: 0,003

• Оценка приоритетов критериев

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | Сумма по строке | Нормированная сумма по строке |
| A | 1 | 8 | 1.14 | 1.33 | 10.48 | 0.36 |
| B | 0.12 | 1 | 0.14 | 0.17 | 2.43 | 0.05 |
| C | 0.88 | 7 | 1 | 1.17 | 17.04 | 0.32 |
| D | 0.75 | 6 | 0.86 | 1 | 14.61 | 0.27 |

Коэффициент согласованности: 0

Составим матрицу ( – альтернатива, - критерий) и умножим на столбец оценки приоритетов:

Оценив полученный вектор, можем сделать вывод, что оптимальным вариантом является A – Испания.

**Вывод**

В ходе выполнения работы были изучены и разобраны различные методы решения многокритериальных задач: метод замены критериев ограничениями, метод Парето, метод взвешивания и метод анализа иерархий. Также в ходе работы в различных методах были получены различные оптимальные решения. Это связано с тем, что в первых двух методах выделяются главные критерии, которые сильно влияют на ответ.

**Приложения**

**Приложение А**

**Файл ‘initialdata.cs’:**

public class initialdata

{

public static int[] matrix01 = new int[] { 8, 1, 7, 6 };

static int[,] GetMatrix()

{

int[,] matrix0 = new int[4, 4]

{

{ 5, 6, 4, 10 },

{ 6, 7, 3, 2 },

{ 3, 2, 9, 5 },

{ 2, 1, 10, 4 }

};

return matrix0;

}

public static int[,] matrix0 = GetMatrix();

public static void Criteria()

{

Console.WriteLine();

Console.WriteLine(" Критерии");

Console.WriteLine("1. Цена");

Console.WriteLine("2. Удалённость");

Console.WriteLine("3. Пляж");

Console.WriteLine("4. Достопримечательности");

Console.WriteLine();

}

}

**Файл ‘Methods.cs’:**

public class Methods

{

// Метод для вывода матрицы

public static void PrintMatrix<T>(T[,] matrix)

{

int rows = matrix.GetLength(0);

int cols = matrix.GetLength(1);

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

{

for (int j = 0; j < cols; j++)

{

if (matrix[i, j] is double)

{

double value = (double)(object)matrix[i, j];

if (value % 1 == 0)

{

if (value == 10)

{

Console.Write("10 ");

}

else

{

Console.Write(" " + value + " ");

}

}

else

{

Console.Write(value.ToString("F2") + " ");

}

}

else

{

int value = (int)(object)matrix[i, j];

if (value == 10)

{

Console.Write("10 ");

}

else

Console.Write(value + " ");

}

}

Console.WriteLine();

}

}

public static void PrintMatrix<T>(T[] matrix)

{

int cols = matrix.GetLength(0);

for (int j = 0; j < cols; j++)

{

if (matrix[j] is double)

{

double value = (double)(object)matrix[j];

if (value % 1 == 0)

{

if (value == 10)

{

Console.Write("10 ");

}

else

{

Console.Write(" " + value + " ");

}

}

else

{

Console.Write(value.ToString("0.##") + " ");

}

}

else

{

int value = (int)(object)matrix[j];

if (value == 10)

{

Console.Write("10 ");

}

else

Console.Write(value + " ");

}

}

Console.WriteLine();

}

public static double MaxinColumn(double[,] matrix, int column)

{

double max = 0;

for (int i = 0; i < 4; i++)

{

if (matrix[i, column] > max)

max = matrix[i, column];

}

return max;

}

public static double MininColumn(double[,] matrix, int column)

{

double min = 10;

for (int i = 0; i < 4; i++)

{

if (matrix[i, column] < min)

min = matrix[i, column];

}

return min;

}

public static void WriteCountry(int i)

{

switch (i)

{

case 0:

Console.WriteLine("Испания");

break;

case 1:

Console.WriteLine("Турция");

break;

case 2:

Console.WriteLine("Куба");

break;

case 3:

Console.WriteLine("Индонезия");

break;

default:

Console.WriteLine("arror in country");

break;

}

}

// Метод для нормирования значений в столбцах

public static void NormalizeColumns(double[,] matrix)

{

for (int j = 0; j < 4; j++)

{

double sum = 0;

for (int i = 0; i < 4; i++)

{

sum += matrix[i, j];

}

for (int i = 0; i < 4; i++)

{

matrix[i, j] /= sum;

}

}

}

public static void NormalizeColumns(double[] matrix)

{

double sum = 0;

for (int i = 0; i < 4; i++)

{

sum += matrix[i];

}

for (int i = 0; i < 4; i++)

{

matrix[i] /= sum;

}

}

// Умножение матриц

public static double[] MatrixMultiplication(double[,] matrix1, double[] matrix2)

{

double[] result = new double[4];

for (int i = 0; i < 4; i++) // по строке

{

double sum = 0;

for (int j = 0; j < 4; j++) // по элементам в строке

{

sum += matrix1[i, j] \* matrix2[j];

}

result[i] = sum;

}

return result;

}

// Метод нахождения индекса строки, в которой максимальный элемент

public static int IndexMaxStr(double[,] matrix, int column)

{

int IndexMax = 0;

for (int i = 0; i < 4; i++)

{

if (matrix[i, column] > matrix[IndexMax, column])

IndexMax = i;

}

return IndexMax;

}

// Выводит промежуточную нормированную матрицу и возвращает преобразованный столбец

public static void PairwiseComparison(double[] matrix)

{

double[,] matrix1 = new double[4, 4]; // Промежуточная таблица

for (int i = 0; i < 4; i++) // по элементам данного массива или по строке

{

for (int j = 0; j < 4; j++) // по столбцам

{

matrix1[i, j] = matrix[i] / matrix[j];

}

}

PrintMatrix(matrix1);

for (int i = 0; i < 4; i++) // по строке

{

for (int j = 0; j < 4; j++) // по столбцам

{

matrix[i] += matrix1[i, j];

}

}

Console.WriteLine("\nСумма по строкам");

PrintMatrix(matrix);

}

public static double ConsistencyCoefficient(double[] matrix, int[] matrixweight)

{

double[,] matrix1 = new double[4, 4]; // Промежуточная таблица

for (int i = 0; i < 4; i++) // по элементам данного массива или по строке

{

for (int j = 0; j < 4; j++) // по столбцам

{

matrix1[i, j] = matrix[i] / matrix[j];

}

}

double[] matrix2 = new double[4]; // Матрица весов (нормированная)

Array.Copy(matrixweight, matrix2, matrixweight.Length);

double sum = matrixweight.Sum();

for (int i = 0; i < 4; i++)

{

matrix2[i] = matrixweight[i] / sum;

}

double LambdaMax = 0;

for (int j = 0; j < 4; j++) // по столбцам

{

double SuminColumn = 0;

for (int i = 0; i < 4; i++)

{

SuminColumn += matrix1[i, j];

}

LambdaMax += SuminColumn \* matrix2[j];

}

return (LambdaMax - 4) / (3 \* 0.9);

}

}

**Файл ‘Program1’:**

// Нормирование матрицы

class Program1

{

static void Main()

{

Program(initialdata.matrix01);

}

static void Program(int[] matrix)

{

Console.WriteLine("таблица весов");

Methods.PrintMatrix(matrix);

double[] matrix1 = new double[4];

Array.Copy(matrix, matrix1, matrix.Length);

double sum = matrix.Sum();

for (int i = 0; i < 4; i++)

{

matrix1[i] = matrix[i] / sum;

}

Console.WriteLine("нормированная таблица весов");

Methods.PrintMatrix(matrix1);

}

}

**Файл ‘Program2’:**

// Замена критериев ограничениями

class Program2

{

static void Main()

{

Console.WriteLine("Начальная таблица");

Methods.PrintMatrix(initialdata.matrix0);

initialdata.Criteria();

Program(initialdata.matrix0, 1);

}

static void Program(int[,] matrix, int SelectedColumn)

{

double[,] matrix1 = new double[4, 4];

Array.Copy(matrix, matrix1, matrix.Length);

for (int j = 1; j < 4; j++)

{

if ((SelectedColumn - 1) != j)

{

double Amax = Methods.MaxinColumn(matrix1, j);

double Amin = Methods.MininColumn(matrix1, j);

for (int i = 0; i < 4; i++)

matrix1[i, j] = (matrix1[i, j] - Amin) / (Amax - Amin);

}

}

Console.WriteLine("нормированная матрица (кроме " + SelectedColumn + " столбца)");

Methods.PrintMatrix(matrix1);

double A2 = 0.3 \* Methods.MaxinColumn(matrix1, 1);

double A3 = 0.3 \* Methods.MaxinColumn(matrix1, 2);

double A4 = 0.3 \* Methods.MaxinColumn(matrix1, 3);

int IndexStr = 0;

for (int i = 0; i < 4; i++)

{

if ((matrix1[i, 1] >= A2) && (matrix1[i, 2] >= A3) && (matrix1[i, 3] >= A4)) // Проверяем минимально требуемые значения

if (matrix1[i, 0] >= matrix1[IndexStr, 0]) // Если по главному критерию результат лучше

IndexStr = i; // то k присваиваем значение данной строки

}

Console.WriteLine();

Methods.WriteCountry(IndexStr);

}

}

**Файл ‘Program3’:**

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace Program3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

// Считываем значения x и y из текстовых полей

int x = int.Parse(textBox1.Text);

int y = int.Parse(textBox2.Text);

// Выбираем столбцы x и y из matrix0

int[] columnX = new int[4];

int[] columnY = new int[4];

for (int i = 0; i < 4; i++)

{

columnX[i] = initialdata.matrix0[i, x - 1];

columnY[i] = initialdata.matrix0[i, y - 1];

}

// Строим график

Chart chart1 = new Chart();

chart1.Size = new Size(500, 500);

chart1.ChartAreas.Add(new ChartArea());

// Добавление грида

chart1.ChartAreas[0].AxisX.MajorGrid.Enabled = true;

chart1.ChartAreas[0].AxisY.MajorGrid.Enabled = true;

chart1.ChartAreas[0].AxisX.Interval = 1;

chart1.ChartAreas[0].AxisY.Interval = 1;

chart1.ChartAreas[0].AxisX.Title = GetAxisTitle(x);

chart1.ChartAreas[0].AxisY.Title = GetAxisTitle(y);

// Добавление точек

Series series = chart1.Series.Add("Data");

chart1.Series["Data"].ChartType = SeriesChartType.Point;

for (int i = 0; i < 4; i++)

{

chart1.Series["Data"].Points.AddXY(columnX[i], columnY[i]);

chart1.Series["Data"].Points[i].Label = WriteCountry(i); // Добавляем подпись для каждой точки

}

chart1.Series["Data"].MarkerSize = 10;

chart1.Series["Data"].Font = new Font("Arial", 10, FontStyle.Bold);

// Добавление точки на график

chart1.Series.Add("Marker");

chart1.Series["Marker"].ChartType = SeriesChartType.Point;

chart1.Series["Marker"].Points.AddXY(10, 10);

chart1.Series["Marker"].MarkerSize = 10;

chart1.Series["Marker"].Label = "Точка утопии";

chart1.Series["Marker"].Font = new Font("Arial", 12, FontStyle.Bold);

// Добавление графика на форму

Controls.Add(chart1);

// Все расстояния в квадрате

double MinDistance = 200;

double DistanceToPoint = 0;

int IndexStr = 0;

for (int i = 0; i < 4; i++)

{

DistanceToPoint = Math.Pow(10 - initialdata.matrix0[i, x - 1], 2) + Math.Pow(10 - initialdata.matrix0[i, y - 1], 2);

if (DistanceToPoint < MinDistance)

{

MinDistance = DistanceToPoint;

IndexStr = i;

}

}

myLabel.Text = "Ответ: " + WriteCountry(IndexStr);

// Массив строк для заголовков столбцов таблицы

string[] columnHeaders = { "Страна", GetAxisTitle(x), GetAxisTitle(y) };

// Установите массивы данных и заголовков столбцов в DataGridView

dataGrid1.ColumnCount = 3;

dataGrid1.RowCount = 4;

for (int i = 0; i < 3; i++) dataGrid1.Columns[i].HeaderText = columnHeaders[i];

for (int i = 0; i < 4; i++)

{

dataGrid1.Rows[i].Cells[0].Value = WriteCountry(i);

dataGrid1.Rows[i].Cells[1].Value = columnX[i].ToString();

dataGrid1.Rows[i].Cells[2].Value = columnY[i].ToString();

}

// Отображение таблицы

dataGrid1.RowHeadersVisible = false;

dataGrid1.Visible = true;

}

private string GetAxisTitle(int axis)

{

switch (axis)

{

case 1:

return "Цена";

case 2:

return "Удалённость";

case 3:

return "Пляж";

case 4:

return "Достопримеч";

default:

return "";

}

}

public class initialdata

{

static int[,] GetMatrix()

{

int[,] matrix0 = new int[4, 4]

{

{ 5, 6, 4, 10 },

{ 6, 7, 3, 2 },

{ 3, 2, 9, 5 },

{ 2, 1, 10, 4 }

};

return matrix0;

}

public static int[,] matrix0 = GetMatrix();

}

private string WriteCountry(int i)

{

switch (i)

{

case 0:

return "Испания";

case 1:

return "Турция";

case 2:

return "Куба";

case 3:

return "Индонезия";

default:

return "arror in country";

}

}

}

}

**Файл ‘Program4’:**

// Методом взвешивания и объединения критериев

class Program4

{

static void Main()

{

Program(initialdata.matrix0, initialdata.matrix01);

}

static void Program(int[,] matrix, int[] matrix0)

{

// Общая матрица

double[,] matrix1 = new double[4, 4];

Array.Copy(matrix, matrix1, matrix.Length);

Methods.NormalizeColumns(matrix1);

Console.WriteLine("Нормированная матрица");

Methods.PrintMatrix(matrix1);

// Матрица весов

double[] matrixAlfa = new double[4];

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

if (i == j)

matrixAlfa[i] += 0;

else

{

if (matrix0[i] > matrix0[j])

matrixAlfa[i] += 1;

if (matrix0[i] < matrix0[j])

matrixAlfa[i] += 0;

if (matrix0[i] == matrix0[j])

matrixAlfa[i] += 0.5;

}

}

}

Methods.NormalizeColumns(matrixAlfa);

Console.WriteLine("\nMatrixAlfa:");

Methods.PrintMatrix(matrixAlfa);

// Запись в столбик реультата перемножения матриц

double[,] matrix2 = new double[4, 1];

for (int i = 0; i < 4; i++)

matrix2[i, 0] = Methods.MatrixMultiplication(matrix1, matrixAlfa)[i];

Console.WriteLine("\nРезультат перемножения");

Methods.PrintMatrix(matrix2);

Console.WriteLine();

Methods.WriteCountry(Methods.IndexMaxStr(matrix2, 0));

}

}

**Файл ‘Program5’:**

// Метод анализа иерархий

class Program5

{

static void Main()

{

Program(initialdata.matrix0, initialdata.matrix01);

}

static void Program(int[,] matrix, int[] matrix0)

{

// Общая матрица

double coef = 0;

double[,] matrix1 = new double[4, 4];

for (int j = 0; j < 4; j++)

{

Console.WriteLine("\nНормированный " + (j+1) + " столбец");

double[] column = new double[4];

for (int i = 0; i < 4; i++)

{

column[i] = matrix[i, j];

}

Methods.PairwiseComparison(column);

coef = Methods.ConsistencyCoefficient(column, matrix0);

Methods.NormalizeColumns(column);

Console.WriteLine("нормированная сумма");

Methods.PrintMatrix(column);

for (int i = 0; i < 4; i++)

{

matrix1[i, j] = column[i];

}

Console.WriteLine("коэффициент согласованности: {0: 0.000}", coef);

}

// Матрица весов

double[] matrix2 = new double[4];

Array.Copy(matrix0, matrix2, matrix0.Length);

Console.WriteLine("\nМатрица весов");

Methods.PairwiseComparison(matrix2);

coef = Methods.ConsistencyCoefficient(matrix2, matrix0);

Methods.NormalizeColumns(matrix2);

Console.WriteLine("нормированная сумма");

Methods.PrintMatrix(matrix2);

Console.WriteLine("коэффициент согласованности: " + coef);

// Результат

double[,] matrix3 = new double[4, 1];

for (int i = 0; i < 4; i++)

matrix3[i, 0] = Methods.MatrixMultiplication(matrix1, matrix2)[i];

Console.WriteLine();

Methods.PrintMatrix(matrix3);

Console.WriteLine();

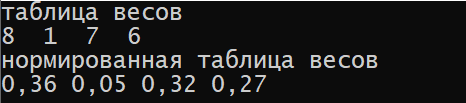
Methods.WriteCountry(Methods.IndexMaxStr(matrix3, 0));

}

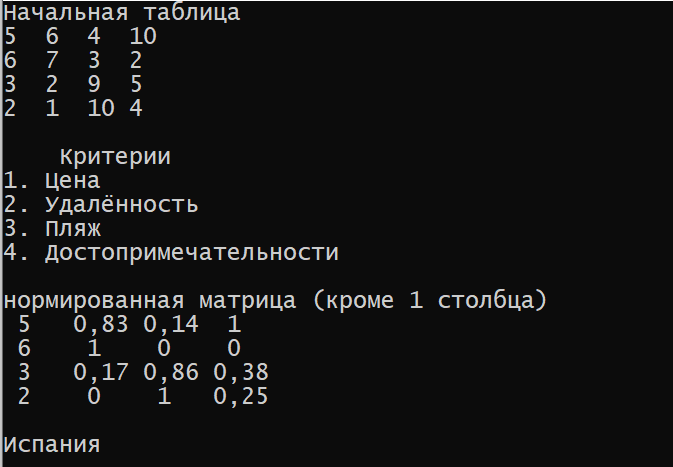
}

**Приложение В**

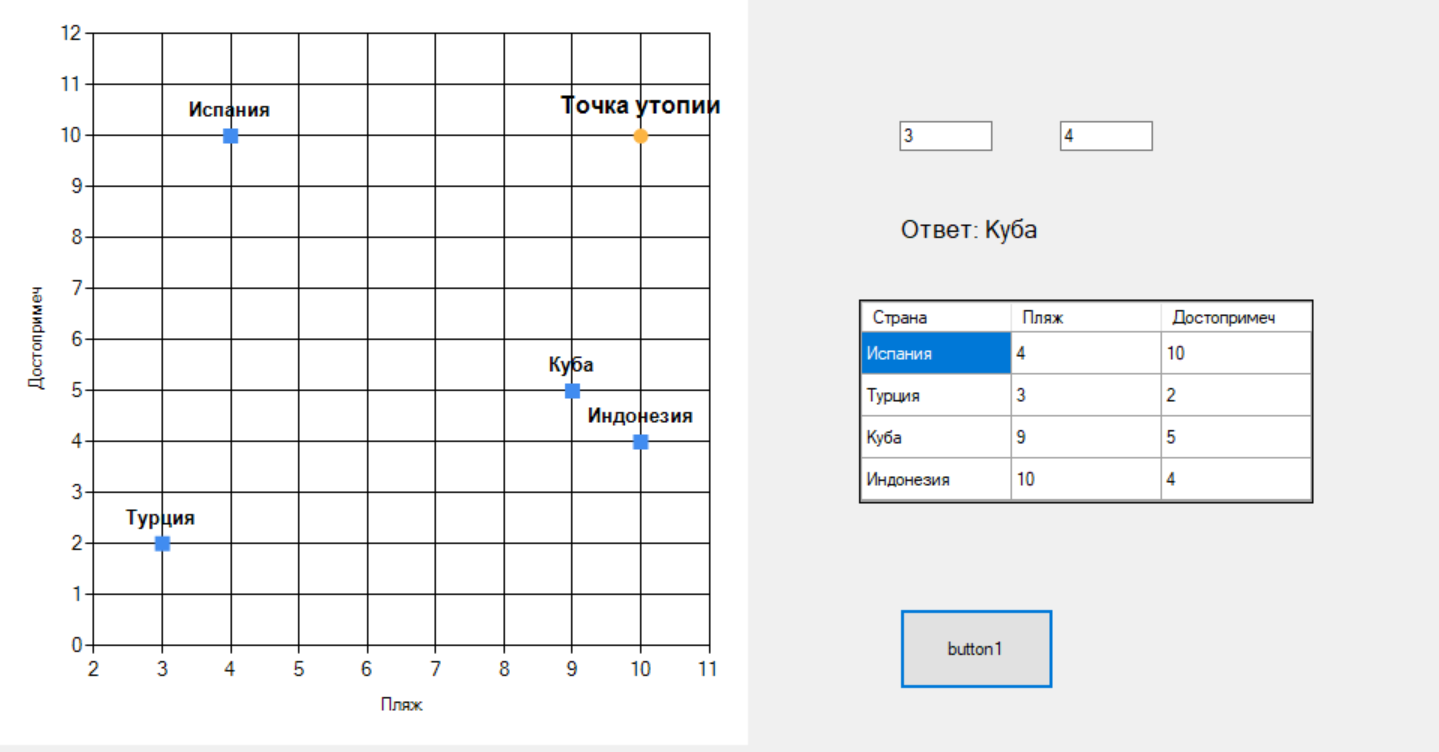
Пункт 1



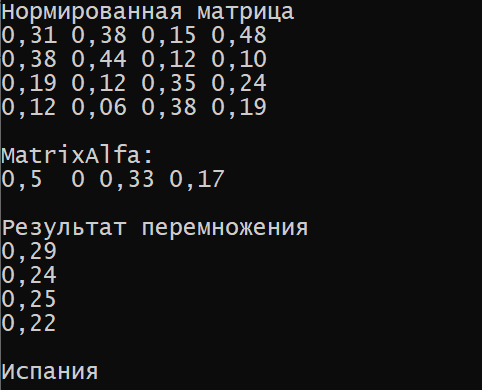
Пункт 2



Пункт 3



Пункт 4



Пункт 5

