Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Лабораторная работа №4

ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ МЕТОДА АНАЛИЗА ИЕРАРХИЙ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ВЫБОРА АЛЬТЕРНАТИВ

Выполнил:

ст. гр. И-32-д

Серба А.В.

Проверил:

Токарев А.И.

Севастополь

2014

1. ЦЕЛЬ РАБОТЫ

Исследовать применение аппарата метода анализа иерархий при принятии решений по выбору альтернатив.

1. ЗАДАНИЕ НА РАБОТУ
   1. Сформировать следующие матрицы:
   2. парных сравнений влияния характеристик альтернатив (решений) на общую цель принятия решений (матрицу *А1*);
   3. парных сравнений наличия рассматриваемых свойств (характеристик) у предлагаемых к анализу решений  – матрицы  (где , *m* – количество критериев (свойств, характеристик) рассматриваемых альтернатив);
   4. Реализовать процедуру, которая используя заданный в варианте метод, определяет вектор собственных значений *W* каждой из матриц парных сравнений, вычисляет собственное значение матрицы  и индекс согласованности (ИС) оценок в ней.
   5. Проверить выполнение условия согласованности оценок в каждой из матриц парных сравнений на каждом уровне. В случае плохой согласованности повторить шаги 1 и 2.
   6. Разработать процедуру, которая на основе векторов собственных значений  матриц  (, количество элементов в каждом из векторов равно количеству рассматриваемых решений (альтернатив), т.е. *n*) для каждого из решений сформировать вектор  весовых коэффициентов , каждый из которых соответствует наличию *j-*ой характеристики (свойства, критерия) у соответствующего *i-*го решения (количество элементов в каждом из этих векторов равно количеству свойств решений, влияющих на общую цель принятия решений, т.е. *m*);
   7. Разработать процедуру, которая на основе векторов весовых коэффициентов на первом уровне – , на втором уровне –  () выполняет расчет оценок для каждого решения, эта же процедура реализует определение на основе значений  () эффективного решения .
2. ВАРИАНТ ЗАДАНИЯ

**Вариант 3.**

В процессе дипломного проектирования возникает необходимость выбора темы дипломного проекта. (дипломный руководитель предлагает несколько тем на выбор).

Цель принятия решений состоит в выборе темы для дипломного проектирования из предлагаемого перечня.

Характеристиками (критериями), соответствующими свойствам решений, являются:

* сложность материала, положенного в основу темы дипломного проект;
* наличие знаний по материалу, на основе которого реализуется дипломный проект;
* возможность использования знаний, полученных при дипломном проектировании по выбранной теме, в дальнейшей деятельности;
* наличие свободного времени для реализации выбранной темы дипломного проекта.

Для реализации выбора необходимо сформировать требуемые матрицы парных сравнений и реализовать процедуру принятия решений. При этом для определения значений элементов собственных векторов матриц парных сравнений использовать третий из предложенных в Приложении А методов.

**Метод 3**

1. Выполнить суммирование элементов каждого столбца и определить величины, обратные каждой из полученных сумм (обратная величина для рассматриваемого значения – это результат деления единицы на само это значение)
2. Разделить каждую обратную величину на сумму всех обратных величин (нормализация обратных величин с тем, чтобы их сумма была равна 1).
3. ТЕКСТ ПРОГРАММЫ

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TPR4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

dataGridViewA.Columns.Add("Ai1", "Ai1");

dataGridViewA.Columns.Add("Ai2", "Ai2");

dataGridViewA.Columns.Add("Ai3", "Ai3");

dataGridViewA.Columns.Add("Ai4", "Ai4");

dataGridViewA.Rows.Add();

dataGridViewA.Rows.Add();

dataGridViewA.Rows.Add();

dataGridViewA.Rows.Add();

dataGridViewA.Rows[0].Cells[0].Value = "1";

dataGridViewA.Rows[0].Cells[1].Value = "2";

dataGridViewA.Rows[0].Cells[2].Value = "6";

dataGridViewA.Rows[0].Cells[3].Value = "24";

dataGridViewA.Rows[1].Cells[0].Value = "2";

dataGridViewA.Rows[1].Cells[1].Value = "1";

dataGridViewA.Rows[1].Cells[2].Value = "2";

dataGridViewA.Rows[1].Cells[3].Value = "6";

dataGridViewA.Rows[2].Cells[0].Value = (double)1 / 6 + "";

dataGridViewA.Rows[2].Cells[1].Value = (double)1 / 2 + "";

dataGridViewA.Rows[2].Cells[2].Value = "1";

dataGridViewA.Rows[2].Cells[3].Value = "2";

dataGridViewA.Rows[3].Cells[0].Value = (double)1 / 24 + "";

dataGridViewA.Rows[3].Cells[1].Value = (double)1 / 6 + "";

dataGridViewA.Rows[3].Cells[2].Value = (double)1 / 2 + "";

dataGridViewA.Rows[3].Cells[3].Value = "1";

//21

dataGridViewA21.Columns.Add("Ai1", "Ai1");

dataGridViewA21.Columns.Add("Ai2", "Ai2");

dataGridViewA21.Columns.Add("Ai3", "Ai3");

dataGridViewA21.Rows.Add();

dataGridViewA21.Rows.Add();

dataGridViewA21.Rows.Add();

dataGridViewA21.Rows[0].Cells[0].Value = "1";

dataGridViewA21.Rows[0].Cells[1].Value = "3";

dataGridViewA21.Rows[0].Cells[2].Value = "8";

dataGridViewA21.Rows[1].Cells[0].Value = "3";

dataGridViewA21.Rows[1].Cells[1].Value = "1";

dataGridViewA21.Rows[1].Cells[2].Value = "3";

dataGridViewA21.Rows[2].Cells[0].Value = (double)1 / 8 + "";

dataGridViewA21.Rows[2].Cells[1].Value = (double)1 / 3 + "";

dataGridViewA21.Rows[2].Cells[2].Value = "1";

//22

dataGridViewA22.Columns.Add("Ai1", "Ai1");

dataGridViewA22.Columns.Add("Ai2", "Ai2");

dataGridViewA22.Columns.Add("Ai3", "Ai3");

dataGridViewA22.Rows.Add();

dataGridViewA22.Rows.Add();

dataGridViewA22.Rows.Add();

dataGridViewA22.Rows[0].Cells[0].Value = "1";

dataGridViewA22.Rows[0].Cells[1].Value = "4";

dataGridViewA22.Rows[0].Cells[2].Value = "9";

dataGridViewA22.Rows[1].Cells[0].Value = "4";

dataGridViewA22.Rows[1].Cells[1].Value = "1";

dataGridViewA22.Rows[1].Cells[2].Value = "4";

dataGridViewA22.Rows[2].Cells[0].Value = (double)1 / 9 + "";

dataGridViewA22.Rows[2].Cells[1].Value = (double)1 / 4 + "";

dataGridViewA22.Rows[2].Cells[2].Value = "1";

//23

dataGridViewA23.Columns.Add("Ai1", "Ai1");

dataGridViewA23.Columns.Add("Ai2", "Ai2");

dataGridViewA23.Columns.Add("Ai3", "Ai3");

dataGridViewA23.Rows.Add();

dataGridViewA23.Rows.Add();

dataGridViewA23.Rows.Add();

dataGridViewA23.Rows[0].Cells[0].Value = "1";

dataGridViewA23.Rows[0].Cells[1].Value = "5";

dataGridViewA23.Rows[0].Cells[2].Value = "7";

dataGridViewA23.Rows[1].Cells[0].Value = "5";

dataGridViewA23.Rows[1].Cells[1].Value = "1";

dataGridViewA23.Rows[1].Cells[2].Value = "5";

dataGridViewA23.Rows[2].Cells[0].Value = (double)1 / 7 + "";

dataGridViewA23.Rows[2].Cells[1].Value = (double)1 / 5 + "";

dataGridViewA23.Rows[2].Cells[2].Value = "1";

//24

dataGridViewA24.Columns.Add("Ai1", "Ai1");

dataGridViewA24.Columns.Add("Ai2", "Ai2");

dataGridViewA24.Columns.Add("Ai3", "Ai3");

dataGridViewA24.Rows.Add();

dataGridViewA24.Rows.Add();

dataGridViewA24.Rows.Add();

dataGridViewA24.Rows[0].Cells[0].Value = "1";

dataGridViewA24.Rows[0].Cells[1].Value = "6";

dataGridViewA24.Rows[0].Cells[2].Value = "11";

dataGridViewA24.Rows[1].Cells[0].Value = "6";

dataGridViewA24.Rows[1].Cells[1].Value = "1";

dataGridViewA24.Rows[1].Cells[2].Value = "6";

dataGridViewA24.Rows[2].Cells[0].Value = (double)1 / 11 + "";

dataGridViewA24.Rows[2].Cells[1].Value = (double)1 / 6 + "";

dataGridViewA24.Rows[2].Cells[2].Value = "1";

}

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

{

List<List<double>> Aij = Считывание(dataGridViewA);

List<double> Wi = ВычислениеWi(Aij);

ЗаполнениеВектора(Wi, dataGridViewWi, "W1");

//21

List<List<double>> Aij21 = Считывание(dataGridViewA21);

List<List<double>> Wi21=new List<List<double>>();

Wi21.Add(ВычислениеWi(Aij21));

ЗаполнениеВектора(Wi21[0], dataGridViewWi21, "W21");

Wi21.Add(ПроизведениеМатрицыНаВектор(Wi21[0], Aij21));

ЗаполнениеВектора(Wi21[1], dataGridViewWi21, "W21");

Wi21.Add(ДелениеВектораНаВектор(Wi21[1], Wi21[0]));

ЗаполнениеВектора(Wi21[2], dataGridViewWi21, "W21");

double Lmax21 = ВычислениеLmax(Wi21[2]);

textBoxLmax21.Text = Lmax21+"";

double Is21 = ВычислениеIS(Lmax21, Wi21[2].Count);

textBoxIS21.Text = Is21 + "";

//22

List<List<double>> Aij22 = Считывание(dataGridViewA22);

List<List<double>> Wi22 = new List<List<double>>();

Wi22.Add(ВычислениеWi(Aij22));

ЗаполнениеВектора(Wi22[0], dataGridViewWi22, "W22");

Wi22.Add(ПроизведениеМатрицыНаВектор(Wi22[0], Aij21));

ЗаполнениеВектора(Wi22[1], dataGridViewWi22, "W22");

Wi22.Add(ДелениеВектораНаВектор(Wi22[1], Wi22[0]));

ЗаполнениеВектора(Wi22[2], dataGridViewWi22, "W22");

double Lmax22 = ВычислениеLmax(Wi22[2]);

textBoxLmax22.Text = Lmax22 + "";

double Is22 = ВычислениеIS(Lmax22, Wi22[2].Count);

textBoxIS22.Text = Is22 + "";

//23

List<List<double>> Aij23 = Считывание(dataGridViewA23);

List<List<double>> Wi23 = new List<List<double>>();

Wi23.Add(ВычислениеWi(Aij23));

ЗаполнениеВектора(Wi23[0], dataGridViewWi23, "W23");

Wi23.Add(ПроизведениеМатрицыНаВектор(Wi23[0], Aij23));

ЗаполнениеВектора(Wi23[1], dataGridViewWi23, "W23");

Wi23.Add(ДелениеВектораНаВектор(Wi23[1], Wi23[0]));

ЗаполнениеВектора(Wi23[2], dataGridViewWi23, "W23");

double Lmax23 = ВычислениеLmax(Wi23[2]);

textBoxLmax23.Text = Lmax23 + "";

double Is23 = ВычислениеIS(Lmax23, Wi23[2].Count);

textBoxIS23.Text = Is23 + "";

//24

List<List<double>> Aij24 = Считывание(dataGridViewA24);

List<List<double>> Wi24 = new List<List<double>>();

Wi24.Add(ВычислениеWi(Aij24));

ЗаполнениеВектора(Wi24[0], dataGridViewWi24, "W24");

Wi24.Add(ПроизведениеМатрицыНаВектор(Wi21[0], Aij21));

ЗаполнениеВектора(Wi24[1], dataGridViewWi24, "W24");

Wi24.Add(ДелениеВектораНаВектор(Wi24[1], Wi24[0]));

ЗаполнениеВектора(Wi24[2], dataGridViewWi24, "W24");

double Lmax24 = ВычислениеLmax(Wi24[2]);

textBoxLmax24.Text = Lmax24 + "";

double Is24 = ВычислениеIS(Lmax24, Wi24[2].Count);

textBoxIS24.Text = Is24 + "";

List<List<double>> W2ij=new List<List<double>>();

W2ij.Add(Wi21[2]);

W2ij.Add(Wi22[2]);

W2ij.Add(Wi23[2]);

W2ij.Add(Wi24[2]);

List<double> D = ВычиеслениеD(W2ij,Wi);

ЗаполнениеВектора(D, dataGridViewDi, "D");

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

{

if (D[i] == D.Max())

{

richTextBoxОтвет.Text += "D[" + (i+1) + "] = " + D[i] + "\n";

}

}

}

public List<double> ВычиеслениеD(List<List<double>> wij,List<double> wj)

{

List<double> D = new List<double>();

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

{

D.Add(0);

for (int j = 0; j < wij[i].Count; j++)

{

D[i] += wij[i][j] \* wj[j];

}

}

return D;

}

public double ВычислениеLmax(List<double> Wi2)

{

double результат=0;

for (int i = 0; i < Wi2.Count; i++)

{

результат += Wi2[i] / Wi2.Count;

}

return результат;

}

public double ВычислениеIS(double Lmax,int n)

{

return (Lmax - n) / (n - 1);

}

List<List<double>> Считывание(DataGridView dataGridView)

{

List<List<double>> матрица = new List<List<double>>();

for (int i = 0; i < dataGridView.Rows.Count - 1; i++)

{

матрица.Add(new List<double>());

for (int j = 0; j < dataGridView.Rows[i].Cells.Count; j++)

{

матрица[i].Add(double.Parse(dataGridView.Rows[i].Cells[j].Value + ""));

}

}

return матрица;

}

public void ЗаполнениеВектора(List<double> вектор, DataGridView dataGridView, string textColumn)

{

if (dataGridView.Columns.Count == 0)

{

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

{

dataGridView.Columns.Add((i+1) + textColumn, (i+1) + textColumn);

}

}

dataGridView.Rows.Add();

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

{

dataGridView.Rows[dataGridView.Rows.Count - 1].Cells[i].Value = вектор[i];

}

}

public List<double> ВычислениеWi(List<List<double>> A)

{

//1

List<double> Wi = new List<double>();

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

{

Wi.Add(0);

for (int j = 0; j < A[i].Count; j++)

{

Wi[i] += A[i][j];

}

Wi[i] = 1 / Wi[i];

}

//2

double sumObr = 0;

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

{

sumObr += Wi[i];

}

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

{

Wi[i] = Wi[i] / sumObr;

}

return Wi;

}

public List<double> ПроизведениеМатрицыНаВектор(List<double> вектор, List<List<double>> матрица)

{

List<double> результат = new List<double>();

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

{

результат.Add(0);

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

{

результат[i] += вектор[i] \* матрица[j][i];

}

}

return результат;

}

public List<double> ДелениеВектораНаВектор(List<double> вектор1, List<double> вектор2)

{

List<double> результат = new List<double>();

for (int i = 0; i < вектор1.Count; i++)

{

результат.Add(вектор1[i] / вектор2[2]);

}

return результат;

}

}

}

1. ТЕСТОВЫЕ ПРИМЕРЫ

На рисунке изображен тестовый пример работы программы.

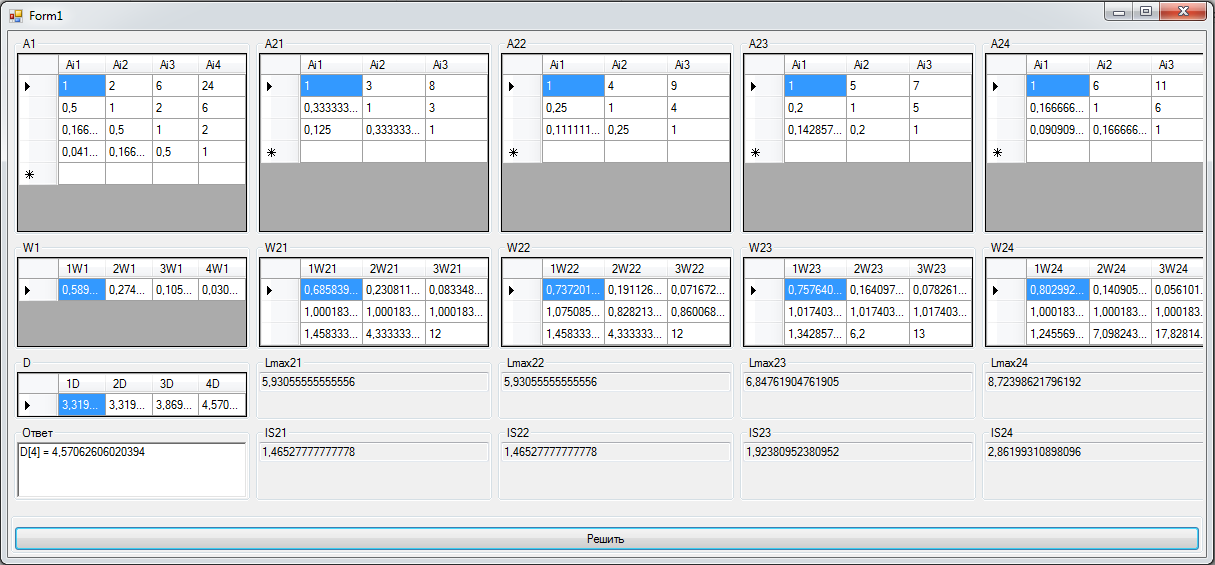


Рисунок 1 – Главное меню, задание значений и вывод результатов

ВЫВОДЫ

В данной лабораторной работе было исследовано применение аппарата метода анализа иерархий при принятии решений по выбору альтернатив.