Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Лабораторная работа № 5

“ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ ТЕОРИИ ВАЖНОСТИ КРИТЕРИЕВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ВЫБОРА АЛЬТЕРНАТИВ”

Выполнил:

ст. гр. ИС-17-2о Горбенко К. Н.

Проверил

Кротов К.В.

Севастополь

2020

1 ЦЕЛЬ

Исследовать применение аппарата теории важности критериев при принятии решений по выбору альтернатив.

2 ПОСТАНОВКА ЗАДАЧИ

Для второго варианта задания предусматривается следующий порядок действий по выполнению лабораторной работы: 1) на основе информации Θ о количественной важности критериев сформировать N-модель в виде вектора, каждый i-ый элемент которого соответствует i-му критерию и определяет число повторений исходных скалярных оценок li k в формируемом векторе ( ) l Θ xK (при n ,1l =); 2) разработать процедуру определения доминируемых решений, выполняющую для каждого решения l x сравнение его значений скалярных оценок i lk вектора ( ) lxK с такими же скалярными оценками hi k решений h x ; тем самым должны быть определены решения h x , доминируемые текущим рассматриваемым решением l x (при n ,1h = и l h ≠); результатом выполнения процедуры является множество Θ X не сравнимых между собой с использованием отношения предпочтения □решений; 3) разработать процедуру, использующую информацию Θ о важности критериев, входными данными для которой будет являться сформированный вектор значений, интерпретируемый как N-модель; разрабатываемая процедура реализует формирование векторов ( ) l Θ xK (n ,1l =), представляющих собой модификацию исходных векторных оценок ( ) lxK (n ,1l =) по соответствующему виду N-модели; таким образом, результатом реализации процедуры являются модифицированные с учетом информации Θ о количественной важности критериев векторные оценки ( ) l Θ xK (n ,1l =); 4) разработать процедуру, упорядочивающую по убыванию скалярные оценки li k (n ,1i =) для каждой сформированной векторной оценки ( ) l Θ xK (n ,1l =); 5) для модифицированных векторных оценок ( ) l Θ xK каждого решения l x (n ,1l =) проконтролировать выполнение условия доминирования им других решений h x для их векторных оценок ( ) h Θ xK (при n ,1h = и l h ≠) (т.е. выполняется поэлементное сравнение оценок li k и hi k из соответствующих векторов ( ) l Θ xK и ( ) h Θ xK); при выполнении условия

hl xx □ , процедура реализует исключение решения h x из множества Θ X : h x\XX □□ □ ; 6) результатом выполнения разрабатываемой программы является определение множества не сравнимых решений Θ X , сформированного на основе информации Θ о количественной важности критериев; 7) выполнить вывод множества Θ X , полученного в результате исключения из него доминируемых решений h x при учете дополнительной информации Θ о количественной важности критериев.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

```
using System;
using System.Collections.Generic;
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.Shapes;
namespace Lab3_WpfApp
{
    public partial class Window1 : Window
    {
        TPR3 tpr;
        public Window1(TPR3 _tpr)
        {
            InitializeComponent();
            tpr = _tpr;
            Write("Множество не доминируемых решений : ");
            foreach (int i in tpr.PMaxX) Write(" " + (i+1) + "; ");
            WriteLn();
            WriteLn("N модель:");
            for(int i = 0; i < tpr.PNX; i++)
            {
                Write("Kp(X" + (i+1) + ") = { ");
                for (int j = 0; j < tpr.NCount; j++) Write(tpr.PValueP[i][j].ToString() +
"; ");
                WriteLn(" }");
            }
            WriteLn("N модель после сортировки:");
            for (int i = 0; i < tpr.PNX; i++)
            {
                Write("Kp(X" + (i + 1) + ") = { ");
                for (int j = 0; j < tpr.NCount; j++) Write(tpr.PValuePU[i][j].ToString()
+ "; ");
                WriteLn(" }");
            }
            Write("Множество не доминируемых решений выведенное с использованием кол
важности : ");
            foreach (int i in tpr.PMaxXP) Write(" " + (i + 1) + "; ");
            WriteLn();
        }

        private void Write(string str)
        {
            TB1.Text += str;
        }
        private void WriteLn(string str)
        {
            TB1.Text += str + '\n';
        }
        private void WriteLn()
        {
            TB1.Text += '\n';
        }
    }
}
using System;
```

```
using System.Collections.Generic;
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 Lab3_WpfApp
{
    public partial class MainWindow : Window
    {
        private TextBox[][] textBoxs;
        private int NX;
        private int NK;
        TPR3 tpr;
        public MainWindow()
        {
            InitializeComponent();
        }
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                NX = int.Parse(TBNX.Text);
                NK = int.Parse(TBNK.Text);
            }
            catch (Exception)
            {
                MessageBox.Show("Неверный ввод", "Ошибка");
                return;
            }
            if (textBoxs != null)
                if (textBoxs.Length != 0)
                {
                    foreach (TextBox[] tbl in textBoxs) foreach (TextBox tb in tbl)
MyGrid.Children.Remove(tb);
                    MyGrid.ColumnDefinitions.Clear();
                    MyGrid.RowDefinitions.Clear();
                }

            if (MyGrid.ColumnDefinitions.Count != NK) for (int i = 0; i < NK; i++)
MyGrid.ColumnDefinitions.Add(new ColumnDefinition());
            if (MyGrid.RowDefinitions.Count != NX + 1) for (int i = 0; i <= NX; i++)
MyGrid.RowDefinitions.Add(new RowDefinition());

            textBoxs = new TextBox[NX + 1][];
            for (int i = 0; i <= NX; i++) textBoxs[i] = new TextBox[NK];

            for (int i = 0; i <= NX; i++) for (int j = 0; j < NK; j++)
                {
                    textBoxs[i][j] = new TextBox();
                    MyGrid.Children.Add(textBoxs[i][j]);
                    Grid.SetColumn(textBoxs[i][j], j);
                    Grid.SetRow(textBoxs[i][j], i);
                    textBoxs[i][j].Text = (i+1) + " X " + (j+1);
                    textBoxs[i][j].Margin = new Thickness(5);
                }
        }
        private void Button_Click_1(object sender, RoutedEventArgs e)
```

```
        {
            int[][] _values = new int[NX][];
            for (int i = 0; i < NX; i++) _values[i] = new int[NK];
            int[] _vazh = new int[NK];
            try
            {
                for (int i = 0; i < NX; i++) for (int j = 0; j < NK; j++) _values[i][j] =
int.Parse(textBoxs[i][j].Text);
                for (int i = 0; i < NK; i++) _vazh[i] = int.Parse(textBoxs[NX][i].Text);
            }
            catch (Exception)
            {
                MessageBox.Show("Неверный ввод", "Ошибка");
                return;
            }
            tpr = new TPR3(NK, NX);
            tpr.SetValue(_values);
            tpr.SetVazh(_vazh);
            tpr.OprMaxX();
            tpr.OprMaxXP();
            Window1 w1 = new Window1(tpr);
            w1.ShowDialog();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Lab3_WpfApp
{
    public class TPR3
    {
        private int NK; // Количество критериев
        private int NX; // Количество решений
        private int[][] Value; // Матрциа оценок
        private int[] Vazh; // Важность критериев
        private int[] MaxX; // Наиболее предпочтительные решения
        private int[] MaxXP; // Наиболее предпочтительные решение с использование доп
информации
        private int[][] ValueP; // Оценки в N модели
        private int[][] ValuePU; // Упорядоченные оценки N модели
        // свойства для доступа к полям из вне
        public int PNK
        {
            get
            {
                return NK;
            }
        }
        public int PNX
        {
            get
            {
                return NX;
            }
        }
        public int[][] PValue
        {
            get
            {
                return Value;
            }
        }
```

```
        public int[] PVazh
        {
            get
            {
                return Vazh;
            }
        }
        public int[] PMaxX
        {
            get
            {
                return MaxX;
            }
        }
        public int[] PMaxXP
        {
            get
            {
                return MaxXP;
            }
        }
        public int[][] PValueP
        {
            get
            {
                return ValueP;
            }
        }
        public int[][] PValuePU
        {
            get
            {
                return ValuePU;
            }
        }
        public int NCount
        {
            get
            {
                int temp = 0;
                for (int i = 0; i < NK; i++) temp += Vazh[i];
                return temp;
            }
        }
        public TPR3(int _nk, int _nx) // Конструктор
        {
            NK = _nk;
            NX = _nx;
            Vazh = new int[_nk];
            Value = new int[_nx][];
            for (int i = 0; i < _nx; i++) Value[i] = new int[_nk];
        }
        public void SetValue(int[][] _value) // Инициализация оценок
        {
            if (_value.Length == 0) return;
            if (Value.Length == 0) return;
            if (_value.Length != Value.Length) return;
            if (_value[0].Length != Value[0].Length) return;

            for (int i = 0; i < NX; i++) for (int j = 0; j < NK; j++) Value[i][j] =
_value[i][j];
        }
        public void SetVazh(int[] _vazh) // Инициализация важности
        {
            if (Vazh.Length == 0) return;
```

```
            if (Vazh.Length != _vazh.Length) return;

            for (int i = 0; i < NK; i++) Vazh[i] = _vazh[i];
        }
        public void OprMaxX() // Определение предпочтительных решений
        {
            if (Value.Length == 0) return;
            List<int> temp = new List<int>();
            for (int i = 0; i < NX; i++) temp.Add(i);
            bool btemp1 = false;
            bool btemp2 = false;

            for (int i = 0; i < NX; i++) for(int j = 0; j < NX; j++)
                {
                    for (int k = 0; k < NK; k++)
                    {
                        if (Value[i][k] > Value[j][k]) btemp1 = true;
                        if (Value[i][k] < Value[j][k]) btemp2 = true;
                    }

                    if (!btemp1) if (btemp2) temp.Remove(i);
                    btemp1 = false;
                    btemp2 = false;
                }
            MaxX = new int[temp.Count];
            for (int i = 0; i < temp.Count; i++) MaxX[i] = temp[i];
        }

        public void OprMaxXP() // Определение предпочтительных решений с учетом доп
информации
        {
            ValueP = new int[NX][];
            ValuePU = new int[NX][];
            for(int i = 0; i < NX; i++)
            {
                ValueP[i] = new int[NCount];
                ValuePU[i] = new int[NCount];
            }
            int temp = 0;
            for(int i = 0; i < NX; i++)
            {
                temp = 0;
                for(int j = 0; j < NK; j++) for(int t = 0; t < Vazh[j]; t++)
                    {
                        ValueP[i][temp] = Value[i][j];
                        ValuePU[i][temp++] = Value[i][j];
                    }}
            for (int i = 0; i < NX; i++) Array.Sort(ValuePU[i]);
            List<int> temp2 = new List<int>();
            for (int i = 0; i < NX; i++) temp2.Add(i);
            bool btemp1 = false;
            bool btemp2 = false;
            for (int i = 0; i < NX; i++) for (int j = 0; j < NX; j++){
                    for (int k = 0; k < NCount; k++)
                    {
                        if (ValuePU[i][k] > ValuePU[j][k]) btemp1 = true;
                        if (ValuePU[i][k] < ValuePU[j][k]) btemp2 = true;
                    }
                    if (!btemp1) if (btemp2) temp2.Remove(i);
                    btemp1 = false;
                    btemp2 = false;
                }
            MaxXP = new int[temp2.Count];
            for (int i = 0; i < temp2.Count; i++) MaxXP[i] = temp2[i];
        }}}
```

4 ПРИМЕРЫ ВЫПОЛНЕНИЯ

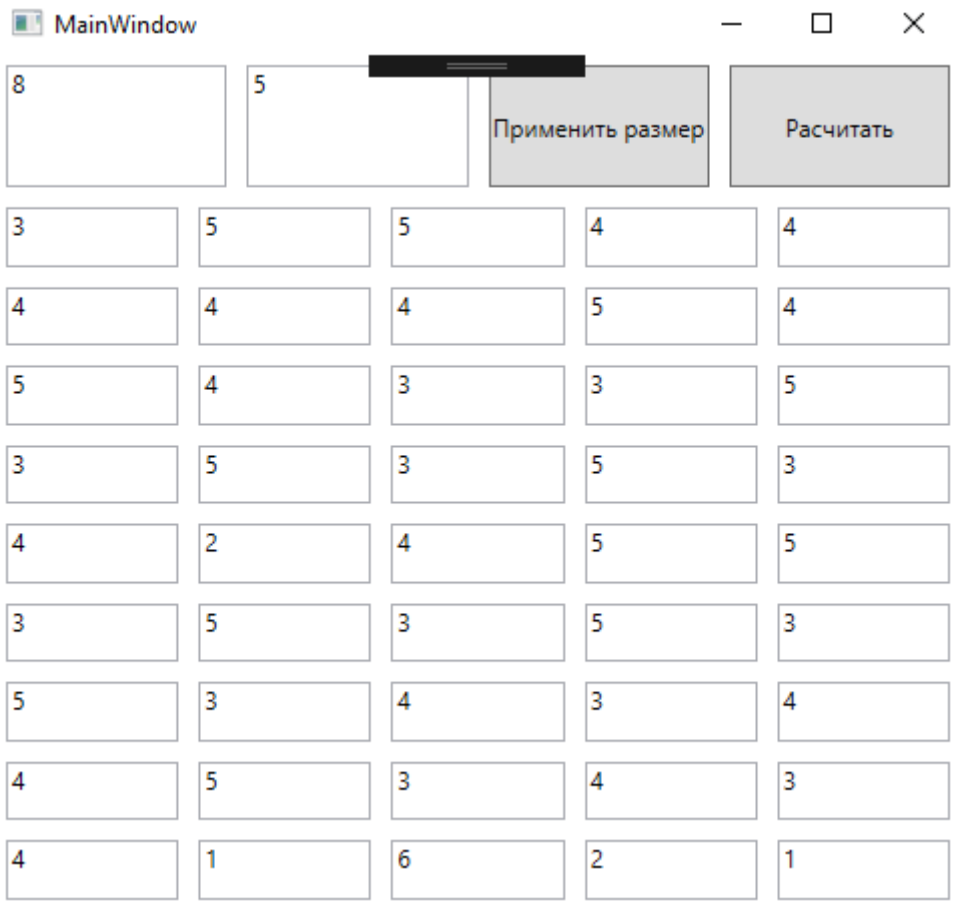

Рисунок 1 — Выполнение программы

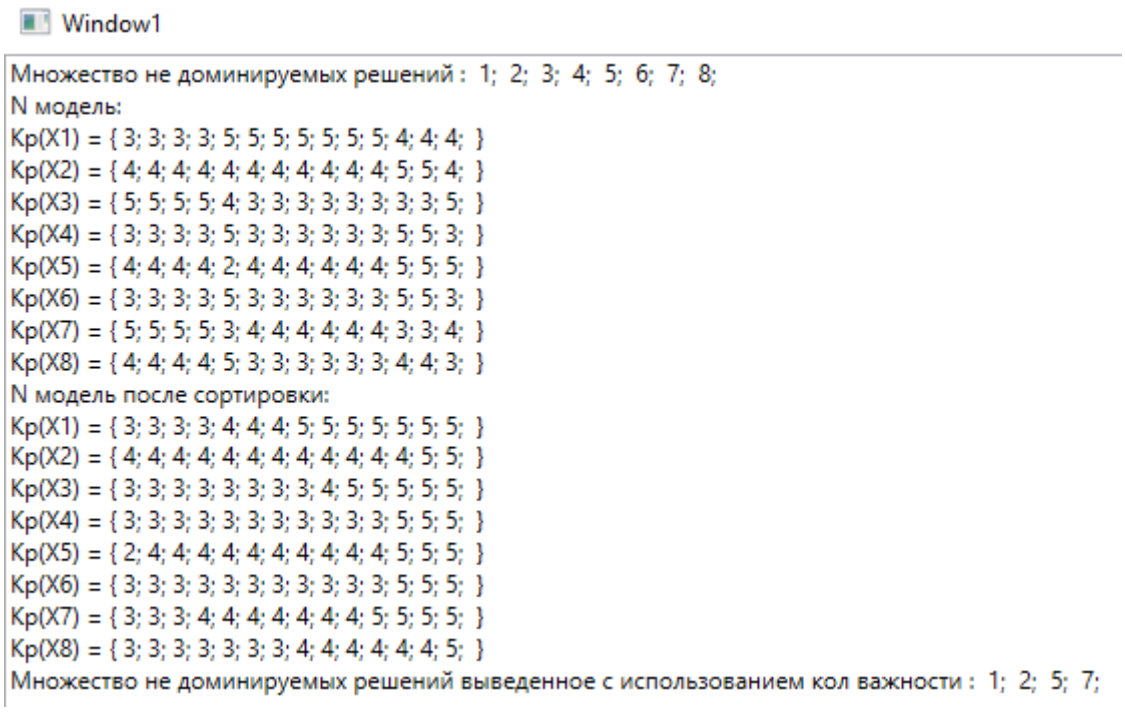

Рисунок 2 — Результат обработки

ВЫВОДЫ

В ходе работы была написана и протестирована программа реализующая определение не доминируемых отношений, использующая простое сравнение скалярных оценок векторов критериев, а так же количественную теорию важности критериев