

Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Лабораторная работа № 3

“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 $l_i k$ в формируемом векторе $() l \Theta xK$ (при $n, l l =$); 2) разработать процедуру определения доминируемых решений, выполняющую для каждого решения $l x$ сравнение его значений скалярных оценок $i l k$ вектора $() l xK$ с такими же скалярными оценками $h_i k$ решений $h x$; тем самым должны быть определены решения $h x$, доминируемые текущим рассматриваемым решением $l x$ (при $n, l h =$ и $l h \neq$); результатом выполнения процедуры является множество ΘX не сравнимых между собой с использованием отношения предпочтения \sqsubset 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 $() l \Theta xK (n, l l =)$, представляющих собой модификацию исходных векторных оценок $() l xK (n, l l =)$ по соответствующему виду N -модели; таким образом, результатом реализации процедуры являются модифицированные с учетом информации Θ о количественной важности критериев векторные оценки $() l \Theta xK (n, l l =)$; 4) разработать процедуру, упорядочивающую по убыванию скалярные оценки $l_i k (n, l i =)$ для каждой сформированной векторной оценки $() l \Theta xK (n, l l =)$; 5) для модифицированных векторных оценок $() l \Theta xK$ каждого решения $l x (n, l l =)$ проконтролировать выполнение условия доминирования им других решений $h x$ для их векторных оценок $() h \Theta xK$ (при $n, l h =$ и $l h \neq$) (т.е. выполняется поэлементное сравнение оценок $l_i k$ и $h_i k$ из соответствующих векторов $() l \Theta xK$ и $() h \Theta xK$); при выполнении условия

$h l x x \sqsubset$, процедура реализует исключение решения $h x$ из множества $\Theta X : h x \backslash \Theta X \rightarrow \rightarrow \square$; 6) результатом выполнения разрабатываемой программы является определение множества не сравнимых решений ΘX , сформированного на основе информации Θ о количественной важности критериев; 7) выполнить вывод множества ΘX , 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[][] textBoxes;
        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 textBoxes) 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());

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

                for (int i = 0; i <= NX; i++) for (int j = 0; j < NK; j++)
                {
                    textBoxes[i][j] = new TextBox();
                    MyGrid.Children.Add(textBoxes[i][j]);
                    Grid.SetColumn(textBoxes[i][j], j);
                    Grid.SetRow(textBoxes[i][j], i);
                    textBoxes[i][j].Text = (i+1) + " X " + (j+1);
                    textBoxes[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(textBoxes[i][j].Text);
            for (int i = 0; i < NK; i++) _vazh[i] = int.Parse(textBoxes[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; // Матрица 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

MainWindow

8

5

Применить размер

Расчитать

3	5	5	4	4
4	4	4	5	4
5	4	3	3	5
3	5	3	5	3
4	2	4	5	5
3	5	3	5	3
5	3	4	3	4
4	5	3	4	3
4	1	6	2	1

Рисунок 1 — Выполнение программы

Window1

Множество не доминируемых решений : 1; 2; 3; 4; 5; 6; 7; 8;

N модель:

Kp(X1) = { 3; 3; 3; 3; 5; 5; 5; 5; 5; 5; 4; 4; 4; }

Kp(X2) = { 4; 4; 4; 4; 4; 4; 4; 4; 4; 4; 5; 5; 4; }

Kp(X3) = { 5; 5; 5; 5; 4; 3; 3; 3; 3; 3; 3; 3; 5; }

Kp(X4) = { 3; 3; 3; 3; 5; 3; 3; 3; 3; 3; 3; 5; 5; 3; }

Kp(X5) = { 4; 4; 4; 4; 2; 4; 4; 4; 4; 4; 4; 5; 5; 5; }

Kp(X6) = { 3; 3; 3; 3; 5; 3; 3; 3; 3; 3; 3; 5; 5; 3; }

Kp(X7) = { 5; 5; 5; 5; 3; 4; 4; 4; 4; 4; 4; 3; 4; }

Kp(X8) = { 4; 4; 4; 4; 5; 3; 3; 3; 3; 3; 3; 4; 4; 3; }

N модель после сортировки:

Kp(X1) = { 3; 3; 3; 3; 4; 4; 5; 5; 5; 5; 5; 5; }

Kp(X2) = { 4; 4; 4; 4; 4; 4; 4; 4; 4; 4; 4; 5; 5; }

Kp(X3) = { 3; 3; 3; 3; 3; 3; 3; 3; 4; 5; 5; 5; 5; }

Kp(X4) = { 3; 3; 3; 3; 3; 3; 3; 3; 3; 3; 5; 5; 5; }

Kp(X5) = { 2; 4; 4; 4; 4; 4; 4; 4; 4; 4; 4; 5; 5; 5; }

Kp(X6) = { 3; 3; 3; 3; 3; 3; 3; 3; 3; 3; 5; 5; 5; }

Kp(X7) = { 3; 3; 3; 4; 4; 4; 4; 4; 4; 5; 5; 5; 5; }

Kp(X8) = { 3; 3; 3; 3; 3; 3; 4; 4; 4; 4; 4; 4; 5; }

Множество не доминируемых решений выведенное с использованием кол важности : 1; 2; 5; 7;

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