Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Лабораторная работа №5

ИССЛЕДОВАНИЕ МЕТОДОВ РЕШЕНИЯ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ ПРИНЯТИЯ РЕШЕНИЙ НА ОСНОВЕ ПОСТРОЕНИЯ МНОЖЕСТВА ПАРЕТО

Выполнил:

ст. гр. И-32-д

Серба А.В.

Проверил:

Токарев А.И.

Севастополь

2014

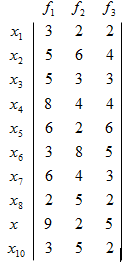
1. ЦЕЛЬ РАБОТЫ

Исследовать способы формирования множества Парето-оптимальных решений и определения эффективных решений в этом множестве.

1. ЗАДАНИЕ НА РАБОТУ
   1. Для первого и третьего вариантов в соответствии с заданием необходимо реализовать следующий порядок действий для выполнения лабораторной работы:
2. разработать процедуру определения на основе задаваемого множества решений Х и соответствующих им значений критериев и множества Р(Х), представляющего собой Парето-границу Х;
3. разработать процедуру определения координат идеальной точки (точки утопии);
4. разработать процедуру расчета расстояния до точки утопии для координат текущего рассматриваемого решения;
5. разработать процедуру определения эффективного решения, расстояние до которого от идеальной точки является минимальным.
6. ВАРИАНТ ЗАДАНИЯ

Вариант 3. Требуется для задаваемого множества Х в виде:  выполнить определение эффективных решений трехкритериальной задачи выбора с использованием метода идеальной точки. Значения критериев ,  и  для соответствующих решений  () сведены в матрицу, представленную ниже.

Таблица 1. Значения критериев и решений



1. 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 ТПР5

{

public partial class Решение : Form

{

public Решение(List<List<double>> значенияРешенийИКритериев, string направлениеПоиска)

{

InitializeComponent();

ЗаполнениеМатрицыList(значенияРешенийИКритериев, dataGridView, "f", "x");

List<double> значенияИдеальнойТочки = ВычислениеЗначенийИдеальнойТочки(значенияРешенийИКритериев, направлениеПоиска);

ЗаполнениеСтрокиList(значенияИдеальнойТочки, dataGridViewЗначенияТочкиУтопии, "f");

SortedList<int, List<double>> матрицаНесравнимыхМеждуСобойРешений;

if (направлениеПоиска == "Максимум")

{

матрицаНесравнимыхМеждуСобойРешений = ВычислениеМатрицыМножестваНесравнимыхМеждуСобойРешенийXMax(значенияРешенийИКритериев);

}

else

{

матрицаНесравнимыхМеждуСобойРешений = ВычислениеМатрицыМножестваНесравнимыхМеждуСобойРешенийXMin(значенияРешенийИКритериев);

}

ЗаполнениеМатрицыSortedList(матрицаНесравнимыхМеждуСобойРешений, dataGridViewНесравнимыеРешения, "f", "x");

SortedList<int,double> растояниеДоИдеальнойТочки = ВычислениеРастоянияДоИдеальнойТочки(значенияИдеальнойТочки, матрицаНесравнимыхМеждуСобойРешений);

ЗаполнениеСтолбцаSortedList(растояниеДоИдеальнойТочки, dataGridViewРастояниеДоИдеальнойТочки, "x");

dataGridViewРастояниеДоИдеальнойТочки.Rows[растояниеДоИдеальнойТочки.IndexOfValue(растояниеДоИдеальнойТочки.Values.Min())].Cells[0].Style.BackColor = Color.Red;

}

void ЗаполнениеМатрицыList(List<List<double>> матрица, DataGridView dataGridView, string nameColumns, string nameRows)

{

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

{

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

}

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

{

dataGridView.Rows.Add();

dataGridView.Rows[i].HeaderCell.Value = nameRows + (i + 1);

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

{

dataGridView.Rows[i].Cells[j].Value = матрица[i][j] + "";

}

}

}

void ЗаполнениеМатрицыSortedList(SortedList<int, List<double>> матрица, DataGridView dataGridView, string nameColumns, string nameRows)

{

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

{

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

}

int k = 0;

foreach (int i in матрица.Keys)

{

dataGridView.Rows.Add();

dataGridView.Rows[k].HeaderCell.Value = nameRows + (i + 1);

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

{

dataGridView.Rows[k].Cells[j].Value = матрица[i][j] + "";

}

k++;

}

}

void ЗаполнениеСтрокиList(List<double> вектор, DataGridView dataGridView, string nameColumns)

{

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

{

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

}

dataGridView.Rows.Add();

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

{

dataGridView.Rows[0].Cells[i].Value = вектор[i] + "";

}

}

void ЗаполнениеСтолбцаSortedList(SortedList<int,double> вектор, DataGridView dataGridView, string nameRows)

{

dataGridView.Columns.Add("Растояние", "Растояние");

int k = 0;

foreach (int i in вектор.Keys)

{

dataGridView.Rows.Add(вектор[i]);

dataGridView.Rows[k].HeaderCell.Value = nameRows + (i + 1);

k++;

}

}

List<double> ВычислениеЗначенийИдеальнойТочки(List<List<double>> значенияРешенийИКритериев, string направлениеПоиска)

{

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

if (направлениеПоиска == "Максимум")

{

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

{

значенияИдеальнойТочки.Add(0);

}

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

{

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

{

if (значенияРешенийИКритериев[i][j] > значенияИдеальнойТочки[j])

{

значенияИдеальнойТочки[j] = значенияРешенийИКритериев[i][j];

}

}

}

}

else

{

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

{

значенияИдеальнойТочки.Add(double.MaxValue);

}

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

{

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

{

if (значенияРешенийИКритериев[i][j] < значенияИдеальнойТочки[j])

{

значенияИдеальнойТочки[j] = значенияРешенийИКритериев[i][j];

}

}

}

}

return значенияИдеальнойТочки;

}

public SortedList<int, List<double>> ВычислениеМатрицыМножестваНесравнимыхМеждуСобойРешенийXMax(

List<List<double>> матрицаСкалярныхОценок)

{

SortedList<int, List<double>> матрицаМножестваНесравнимыхМеждуСобойРешенийX = new SortedList<int, List<double>>();

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

{

матрицаМножестваНесравнимыхМеждуСобойРешенийX.Add(i, матрицаСкалярныхОценок[i]);

}

for (int k = 0; k < матрицаСкалярныхОценок.Count; k++)

{

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

{

bool kяОценкаБольшеИлиРавнаiйОценки = true;

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

{

if (матрицаСкалярныхОценок[k][j] < матрицаСкалярныхОценок[i][j] && i != k)

{

kяОценкаБольшеИлиРавнаiйОценки = false;

}

}

bool однаjяkйОценкиБольшеОднойjйiйОценки = false;

if (kяОценкаБольшеИлиРавнаiйОценки == true)

{

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

{

if (матрицаСкалярныхОценок[k][j] > матрицаСкалярныхОценок[i][j] && i != k)

{

однаjяkйОценкиБольшеОднойjйiйОценки = true;

break;

}

}

}

if (kяОценкаБольшеИлиРавнаiйОценки == true && однаjяkйОценкиБольшеОднойjйiйОценки == true)

{

матрицаМножестваНесравнимыхМеждуСобойРешенийX.Remove(i);

}

}

}

return матрицаМножестваНесравнимыхМеждуСобойРешенийX;

}

public SortedList<int, List<double>> ВычислениеМатрицыМножестваНесравнимыхМеждуСобойРешенийXMin(

List<List<double>> матрицаСкалярныхОценок)

{

SortedList<int, List<double>> матрицаМножестваНесравнимыхМеждуСобойРешенийX = new SortedList<int, List<double>>();

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

{

матрицаМножестваНесравнимыхМеждуСобойРешенийX.Add(i, матрицаСкалярныхОценок[i]);

}

for (int k = 0; k < матрицаСкалярныхОценок.Count; k++)

{

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

{

bool kяОценкаБольшеИлиРавнаiйОценки = true;

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

{

if (матрицаСкалярныхОценок[k][j] > матрицаСкалярныхОценок[i][j] && i != k)

{

kяОценкаБольшеИлиРавнаiйОценки = false;

}

}

bool однаjяkйОценкиБольшеОднойjйiйОценки = false;

if (kяОценкаБольшеИлиРавнаiйОценки == true)

{

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

{

if (матрицаСкалярныхОценок[k][j] < матрицаСкалярныхОценок[i][j] && i != k)

{

однаjяkйОценкиБольшеОднойjйiйОценки = true;

break;

}

}

}

if (kяОценкаБольшеИлиРавнаiйОценки == true && однаjяkйОценкиБольшеОднойjйiйОценки == true)

{

матрицаМножестваНесравнимыхМеждуСобойРешенийX.Remove(i);

}

}

}

return матрицаМножестваНесравнимыхМеждуСобойРешенийX;

}

public SortedList<int,double> ВычислениеРастоянияДоИдеальнойТочки(List<double> значенияИдеальнойТочки, SortedList<int, List<double>> множествоНесравнимыххМеждуСобойРешений)

{

SortedList<int,double> растояниеДоИдеальнойТочки = new SortedList<int,double>();

foreach (int i in множествоНесравнимыххМеждуСобойРешений.Keys)

{

растояниеДоИдеальнойТочки.Add(i,0);

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

{

растояниеДоИдеальнойТочки[i] += Math.Pow(значенияИдеальнойТочки[j] - множествоНесравнимыххМеждуСобойРешений[i][j], 2);

}

растояниеДоИдеальнойТочки[i] = Math.Sqrt(растояниеДоИдеальнойТочки[i]);

}

return растояниеДоИдеальнойТочки;

}

}

}

1. ТЕСТОВЫЕ ПРИМЕРЫ

На рисунках 1-4 изображены тестовые примеры работы программы.

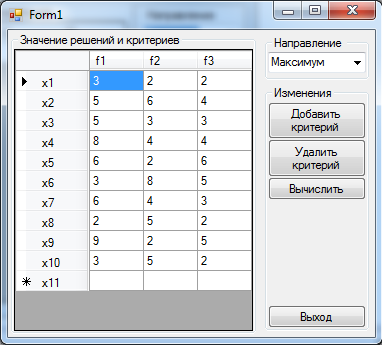


Рисунок 1 – Главное меню, задание значений

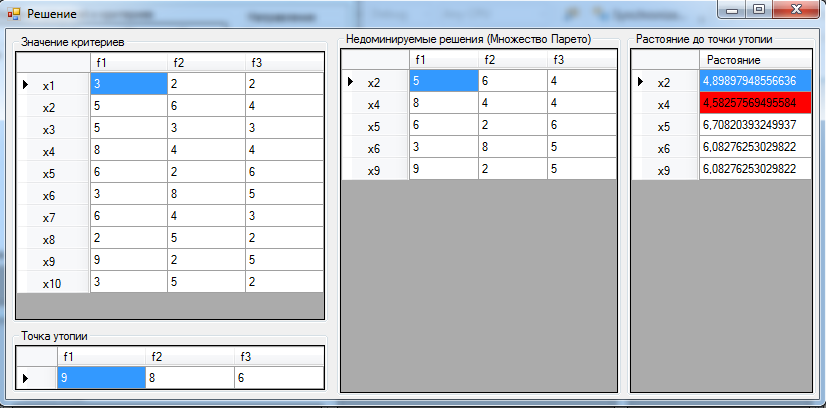


Рисунок 2 – Решение на максимум

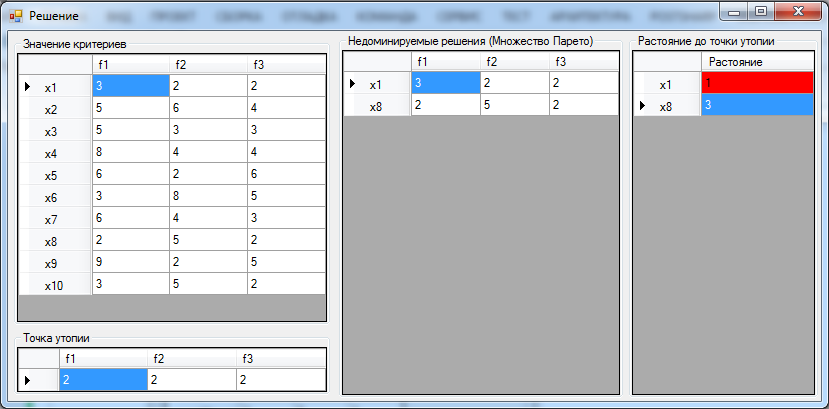


Рисунок 3 – Решение на минимум

ВЫВОДЫ

В данной лабораторной работе были исследованы способы формирования множества Парето-оптимальных решений и определения эффективных решений в этом множестве.