Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Лабораторная работа №2

ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ АППАРАТА ТЕОРИИ ПОЛЕЗНОСТИ ДЛЯ ОПИСАНИЯ БИНАРНЫХ ОТНОШЕНИЙ ПРИ ПРИНЯТИИ РЕШЕНИЙ

Выполнил:

ст. гр. И-32-д

Серба А.В.

Проверил:

Токарев А.И.

Севастополь

2014

1. ЦЕЛЬ РАБОТЫ

Исследовать применение аппарата теории полезности при принятии решений по выбору альтернатив.

1. ВАРИАНТ ЗАДАНИЯ

Вариант 3. Задача состоит в выборе одной из альтернатив, представляющих собой выставленные на продажу автомобили. Критериями (характеристиками) решений являются:  и . Используя метод, реализующий построение и исследование двумерной функции полезности, для заданных диапазонов значений критериев и их (критериев) дискретных оценок выполнить:

1. формирование линий безразличия ,
2. определение на их основе дискретных значений оценок одномерных функций полезности для каждого из критериев и  ,
3. аппроксимацию дискретных значений одномерных функций полезности с использованием полиномов второй степени,
4. вычисление коэффициента масштабирования *j* на основе выбираемых ЛПР по кривым безразличия решениям.

С использованием сформированных промежуточных решений выполнить для задаваемых характеристик альтернатив вычисление значений одномерных функций полезности, двумерной функции полезности и реализовать выбор эффективного решения.

Выполнить вывод исходных данных, всех промежуточных и конечных результатов.

Исходными данными для решаемой задачи являются: параметр "цена" изменяется в диапазоне , параметр "пробег" в диапазоне . Шаг дискретизации первого параметра задан равным 25, шаг дискретизации второго параметра задан равным 20. Соответственно, для первого критерия диапазон изменения его значений задан в виде , для второго критерия диапазон задан в виде . Выбор двух эквивалентных решений на одной из кривых безразличия, сформированных программно, выполнить самостоятельно.

Данные, на основании которых выбирается эффективное решение, имеют следующие значения:

|  |  |  |
| --- | --- | --- |
| Вариант | Цена | Пробег |
| 1 | 30 | 45 |
| 2 | 50 | 30 |
| 3 | 80 | 20 |
| 4 | 25 | 55 |

1. АНАЛИТИЧЕСКОЕ РЕШЕНИЕ

Аналитическое решение присутствует в тестовых примерах.

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 tpr2

{

public partial class Form1 : Form

{

SortedList<int, double> k1;

SortedList<int, double> k2;

List<int> ЗначенияЦены;

List<int> ЗначенияПробега;

public Form1()

{

InitializeComponent();

k1 = new SortedList<int, double>();

k2 = new SortedList<int, double>();

ЗначенияЦены = new List<int>();

ЗначенияПробега = new List<int>();

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

{

dataGridView1.Rows.Add();

dataGridView1.Rows[i].Cells[0].Value = i + 1;

}

dataGridView1.Rows[0].Cells[1].Value = 30;

dataGridView1.Rows[1].Cells[1].Value = 50;

dataGridView1.Rows[2].Cells[1].Value = 80;

dataGridView1.Rows[3].Cells[1].Value = 25;

dataGridView1.Rows[0].Cells[2].Value = 45;

dataGridView1.Rows[1].Cells[2].Value = 30;

dataGridView1.Rows[2].Cells[2].Value = 20;

dataGridView1.Rows[3].Cells[2].Value = 55;

}

double getj(double tochka1\_U1, double tochka1\_U2, double tochka2\_U1, double tochka2\_U2)

{

return ((tochka2\_U2 - tochka1\_U2) / (tochka1\_U1 - tochka1\_U2 - tochka2\_U1 + tochka2\_U2));

}

public class Splayn

{

public SplineTuple[] splines; // Сплайн

// Построение сплайна

// x - узлы сетки, должны быть упорядочены по возрастанию, кратные узлы запрещены

// y - значения функции в узлах сетки

// n - количество узлов сетки

public void BuildSpline(double[] x, double[] y, int n)

{

// Инициализация массива сплайнов

splines = new SplineTuple[n];

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

{

splines[i] = new SplineTuple();

}

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

{

splines[i].x = x[i];

splines[i].a = y[i];

}

splines[0].c = splines[n - 1].c = 0.0;

// Решение СЛАУ относительно коэффициентов сплайнов c[i] методом прогонки для трехдиагональных матриц

// Вычисление прогоночных коэффициентов - прямой ход метода прогонки

double[] alpha = new double[n - 1];

double[] beta = new double[n - 1];

alpha[0] = beta[0] = 0.0;

for (int i = 1; i < n - 1; ++i)

{

double h\_i = x[i] - x[i - 1], h\_i1 = x[i + 1] - x[i];

double A = h\_i;

double C = 2.0 \* (h\_i + h\_i1);

double B = h\_i1;

double F = 6.0 \* ((y[i + 1] - y[i]) / h\_i1 - (y[i] - y[i - 1]) / h\_i);

double z = (A \* alpha[i - 1] + C);

alpha[i] = -B / z;

beta[i] = (F - A \* beta[i - 1]) / z;

}

// Нахождение решения - обратный ход метода прогонки

for (int i = n - 2; i > 0; --i)

splines[i].c = alpha[i] \* splines[i + 1].c + beta[i];

// Освобождение памяти, занимаемой прогоночными коэффициентами

beta = null;

alpha = null;

// По известным коэффициентам c[i] находим значения b[i] и d[i]

for (int i = n - 1; i > 0; --i)

{

double h\_i = x[i] - x[i - 1];

splines[i].d = (splines[i].c - splines[i - 1].c) / h\_i;

splines[i].b = h\_i \* (2.0 \* splines[i].c + splines[i - 1].c) / 6.0 + (y[i] - y[i - 1]) / h\_i;

}

}

// Вычисление значения интерполированной функции в произвольной точке

public double f(double x)

{

SplineTuple s;

int n = splines.Length;

//BuildSpline(myx,y,n);

if (x <= splines[0].x) // Если x меньше точки сетки x[0] - пользуемся первым эл-тов массива

s = splines[1];

else if (x >= splines[n - 1].x) // Если x больше точки сетки x[n - 1] - пользуемся последним эл-том массива

s = splines[n - 1];

else // Иначе x лежит между граничными точками сетки - производим бинарный поиск нужного эл-та массива

{

int i = 0, j = n - 1;

while (i + 1 < j)

{

int k = i + (j - i) / 2;

if (x <= splines[k].x)

j = k;

else

i = k;

}

s = splines[j];

}

double dx = (x - s.x);

// Вычисляем значение сплайна в заданной точке по схеме Горнера (в принципе, "умный" компилятор применил бы схему Горнера сам, но ведь не все так умны, как кажутся)

return s.a + (s.b + (s.c / 2.0 + s.d \* dx / 6.0) \* dx) \* dx;

}

}

public class SplineTuple

{

public double a, b, c, d, x;

}

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

{

List<int> ПромежуточныеЗначенияЦены = new List<int>();

List<int> ПромежуточныеЗначенияПробега = new List<int>();

for (int i = 0; i < dataGridView1.Rows.Count-1; i++)

{

ПромежуточныеЗначенияЦены.Add(int.Parse(dataGridView1.Rows[i].Cells[1].Value.ToString()));

ПромежуточныеЗначенияПробега.Add(int.Parse(dataGridView1.Rows[i].Cells[2].Value.ToString()));

}

for (int i = int.Parse(textBoxНижнийПределЦены.Text), j = 0; i <= int.Parse(textBoxВерхнийПределЦены.Text); i += int.Parse(textBoxШагЦены.Text), j++)

{

dataGridViewK1.Columns.Add("k1" + j, "k1[" + i + "]");

ЗначенияЦены.Add(i);

k1.Add(i, (double)1 / i);

}

dataGridViewK1.Rows.Add();

for (int i = int.Parse(textBoxНижнийПределЦены.Text), j = 0; i <= int.Parse(textBoxВерхнийПределЦены.Text); i += int.Parse(textBoxШагЦены.Text), j++)

{

dataGridViewK1.Rows[0].Cells[j].Value = k1[i];

}

textBoxLimitK1.Text += "[" + k1.Values.Min() + "," + k1.Values.Max() + "]";

for (int i = int.Parse(textBoxНижнийПределПробега.Text), j = 0; i <= int.Parse(textBoxВерхнийПределПробега.Text); i += int.Parse(textBoxШагПробега.Text), j++)

{

dataGridViewK2.Columns.Add("k2" + j, "k2[" + i + "]");

ЗначенияПробега.Add(i);

k2.Add(i, (double)1 / i);

}

dataGridViewK2.Rows.Add();

for (int i = int.Parse(textBoxНижнийПределПробега.Text), j = 0; i <= int.Parse(textBoxВерхнийПределПробега.Text); i += int.Parse(textBoxШагПробега.Text), j++)

{

dataGridViewK2.Rows[0].Cells[j].Value = k2[i];

}

textBoxLimitK2.Text += "[" + k2.Values.Min() + "," + k2.Values.Max() + "]";

double[] u1 = new double[k1.Count],

u2 = new double[k1.Count],

u = new double[k1.Count];

Splayn splayn\_cost = new Splayn();

Splayn splayn\_mileag = new Splayn();

double[] x\_cost = new double[ЗначенияЦены.Count];

double[] y\_cost = new double[ЗначенияЦены.Count];

double[] x\_mileage = new double[ЗначенияПробега.Count];

double[] y\_mileage = new double[ЗначенияПробега.Count];

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

{

x\_cost[i] = ЗначенияЦены[i];

y\_cost[i] = ЗначенияЦены.Count-1-i;

}

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

{

x\_mileage[i] = ЗначенияПробега[i];

y\_mileage[i] = ЗначенияПробега.Count-1-i;

}

splayn\_cost.BuildSpline(x\_cost, y\_cost, ЗначенияЦены.Count);

splayn\_mileag.BuildSpline(x\_mileage, y\_mileage, ЗначенияПробега.Count);

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

{

u1[i] = splayn\_cost.f(ПромежуточныеЗначенияЦены[i]);

u2[i] = splayn\_mileag.f(ПромежуточныеЗначенияПробега[i]);

}

double k = getj(ПромежуточныеЗначенияЦены[0], ПромежуточныеЗначенияПробега[0], ПромежуточныеЗначенияЦены[3], ПромежуточныеЗначенияПробега[3]);

int max = 0;

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

{

u[i] = u1[i] \* k + (1 - k) \* u2[i];

dataGridView1.Rows[i].Cells[5].Value = u[i];

if (u[i] > u[max])

{

max = i;

}

}

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

{

if (i == max)

{

dataGridView1.Rows[i].Cells[5].Style.BackColor = Color.Red;

}

}

for (int i = 0; i < k1.Count; i++)

{

dataGridView1.Rows[i].Cells[3].Value = u1[i].ToString();

dataGridView1.Rows[i].Cells[4].Value = u2[i].ToString();

dataGridView1.Rows[i].Cells[5].Value = u[i].ToString();

}

#region Создание графика кривых безразличия

Grafig КривыеБезразличия = new Grafig();

КривыеБезразличия.Visible = true;

КривыеБезразличия.chart.ChartAreas[0].AxisX.Minimum = k1.Values.Min() - 0.005;

КривыеБезразличия.chart.ChartAreas[0].AxisX.Maximum = k1.Values.Max() + 0.005;

КривыеБезразличия.chart.ChartAreas[0].AxisX.Title = "k1";

КривыеБезразличия.chart.ChartAreas[0].AxisY.Minimum = k2.Values.Min() - 0.01;

КривыеБезразличия.chart.ChartAreas[0].AxisY.Maximum = k2.Values.Max() + 0.01;

КривыеБезразличия.chart.ChartAreas[0].AxisY.Title = "k2";

for (int j = k2.Values.Count - 2; j >= 0; j--)

{

КривыеБезразличия.chart.Series.Add("Кривая безразличия " + (k2.Values.Count - j - 1));

КривыеБезразличия.chart.Series["Кривая безразличия " + (k2.Values.Count - j - 1)].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

for (int i = k1.Values.Count - 1; i >= 0; i--)

{

if (i + j < k2.Values.Count)

{

КривыеБезразличия.chart.Series["Кривая безразличия " + (k2.Values.Count - j - 1)].Points.AddXY(Math.Round(k1.Values[k2.Values.Count - 1 - i], 3), Math.Round(k2.Values[j + i], 3));

}

}

}

КривыеБезразличия.chart.Series.Add("Возможные значения");

КривыеБезразличия.chart.Series["Возможные значения"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.FastPoint;

foreach (int j in k2.Keys)

{

foreach (int i in k1.Keys)

{

КривыеБезразличия.chart.Series["Возможные значения"].Points.AddXY(Math.Round(k1[i], 3), Math.Round(k2[j], 3));

}

}

КривыеБезразличия.chart.Titles.Add("Кривые безразличия");

#endregion

#region График функции полезности критерия K1

Grafig ГрафикФункцииПолезностиКритерияК1 = new Grafig();

ГрафикФункцииПолезностиКритерияК1.Visible = true;

ГрафикФункцииПолезностиКритерияК1.chart.ChartAreas[0].AxisX.Minimum = k1.Keys.Min();

ГрафикФункцииПолезностиКритерияК1.chart.ChartAreas[0].AxisX.Maximum = k1.Keys.Max();

ГрафикФункцииПолезностиКритерияК1.chart.ChartAreas[0].AxisX.Title = "k1";

ГрафикФункцииПолезностиКритерияК1.chart.ChartAreas[0].AxisY.Minimum = 0;

ГрафикФункцииПолезностиКритерияК1.chart.ChartAreas[0].AxisY.Maximum = k1.Count - 1;

ГрафикФункцииПолезностиКритерияК1.chart.ChartAreas[0].AxisY.Title = "U(k1)";

ГрафикФункцииПолезностиКритерияК1.chart.Titles.Add("График функции полезности критерия K1");

ГрафикФункцииПолезностиКритерияК1.chart.Series.Add("График функции полезности критерия K1");

ГрафикФункцииПолезностиКритерияК1.chart.Series["График функции полезности критерия K1"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

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

{

ГрафикФункцииПолезностиКритерияК1.chart.Series["График функции полезности критерия K1"].Points.AddXY(ЗначенияЦены[i], ЗначенияЦены.Count - 1 - i);

}

ГрафикФункцииПолезностиКритерияК1.chart.Series.Add("Значения функции полезности через шаг дискретизации");

ГрафикФункцииПолезностиКритерияК1.chart.Series["Значения функции полезности через шаг дискретизации"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;

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

{

ГрафикФункцииПолезностиКритерияК1.chart.Series["Значения функции полезности через шаг дискретизации"].Points.AddXY(ЗначенияЦены[i], ЗначенияЦены.Count - 1 - i);

}

ГрафикФункцииПолезностиКритерияК1.chart.Series.Add("Промежуточные значения функции полезности");

ГрафикФункцииПолезностиКритерияК1.chart.Series["Промежуточные значения функции полезности"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;

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

{

ГрафикФункцииПолезностиКритерияК1.chart.Series["Промежуточные значения функции полезности"].Points.AddXY(ПромежуточныеЗначенияЦены[i], u1[i]);

}

#endregion

#region График функции полезности критерия K2

Grafig ГрафикФункцииПолезностиКритерияК2 = new Grafig();

ГрафикФункцииПолезностиКритерияК2.Visible = true;

ГрафикФункцииПолезностиКритерияК2.chart.ChartAreas[0].AxisX.Minimum = k2.Keys.Min();

ГрафикФункцииПолезностиКритерияК2.chart.ChartAreas[0].AxisX.Maximum = k2.Keys.Max();

ГрафикФункцииПолезностиКритерияК2.chart.ChartAreas[0].AxisX.Title = "k2";

ГрафикФункцииПолезностиКритерияК2.chart.ChartAreas[0].AxisY.Minimum = 0;

ГрафикФункцииПолезностиКритерияК2.chart.ChartAreas[0].AxisY.Maximum = k2.Count - 1;

ГрафикФункцииПолезностиКритерияК2.chart.ChartAreas[0].AxisY.Title = "k2";

ГрафикФункцииПолезностиКритерияК2.chart.Titles.Add("График функции полезности критерия K2");

ГрафикФункцииПолезностиКритерияК2.chart.Series.Add("График функции полезности критерия K2");

ГрафикФункцииПолезностиКритерияК2.chart.Series["График функции полезности критерия K2"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

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

{

ГрафикФункцииПолезностиКритерияК2.chart.Series["График функции полезности критерия K2"].Points.AddXY(ЗначенияПробега[i], ЗначенияПробега.Count - 1 - i);

}

ГрафикФункцииПолезностиКритерияК2.chart.Series.Add("Значения функции полезности через шаг дискретизации");

ГрафикФункцииПолезностиКритерияК2.chart.Series["Значения функции полезности через шаг дискретизации"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;

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

{

ГрафикФункцииПолезностиКритерияК2.chart.Series["Значения функции полезности через шаг дискретизации"].Points.AddXY(ЗначенияПробега[i], ЗначенияПробега.Count - 1 - i);

}

ГрафикФункцииПолезностиКритерияК2.chart.Series.Add("Промежуточные значения функции полезности");

ГрафикФункцииПолезностиКритерияК2.chart.Series["Промежуточные значения функции полезности"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;

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

{

ГрафикФункцииПолезностиКритерияК2.chart.Series["Промежуточные значения функции полезности"].Points.AddXY(ПромежуточныеЗначенияПробега[i], u2[i]);

}

#endregion

}

}

}

1. ТЕСТОВЫЕ ПРИМЕРЫ

На рисунках 1-4 изображены тестовые примеры работы программы.

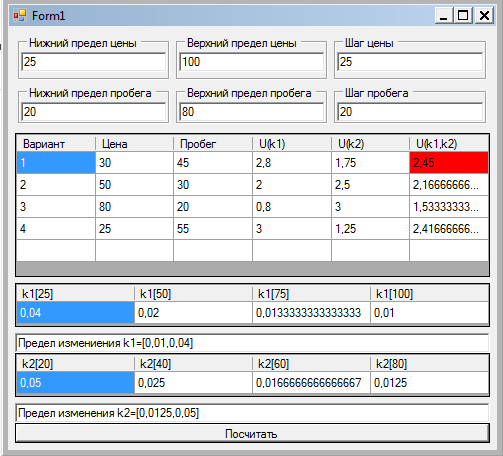


Рисунок 1 – Главное меню, задание значений и вывод результатов

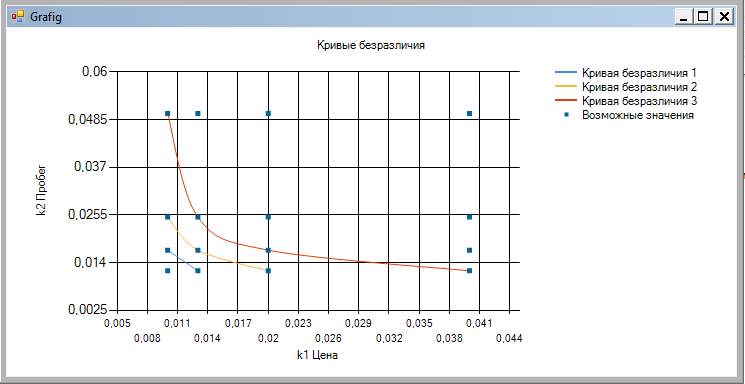


Рисунок 2 – Кривые безразличия и возможные их значения

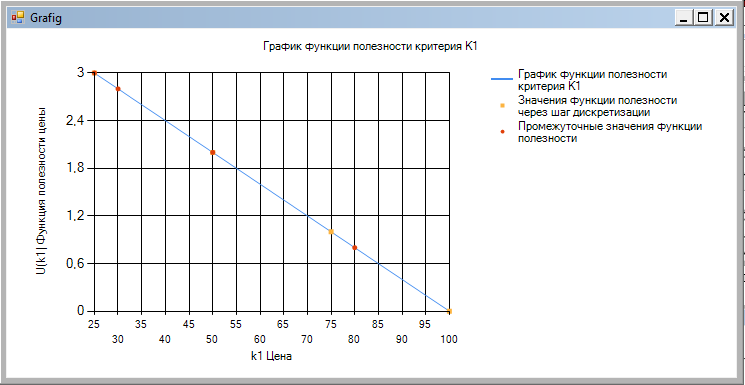


Рисунок 3 – График функции полезности, промежуточные значения и значения через шаг дискретизации функции полезности критерия k1(Цена)

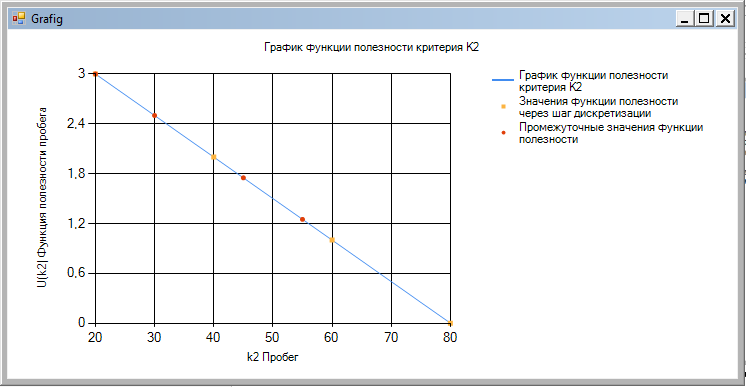


Рисунок 4 – График функции полезности, промежуточные значения и значения через шаг дискретизации функции полезности критерия k2(Пробег)

ВЫВОДЫ

В данной лабораторной работе были исследовано применение аппарата теории полезности при принятии решений по выбору альтернатив.