**1. Постановка задачи**

С помощью генетического алгоритма решить задачу дискретной оптимизации.

Нахождение гамильтонова обхода в неориентированном графе. Гамильтонов путь – путь, который является простым путем (путём без петель), проходящим через каждую вершину графа ровно один раз

**2. Описание генетического алгоритма решения задачи**

1. Задаем количество вершин, граф смежности, количество особей в популяции.

2. Генерирование начальной популяции. Особенность данного шага заключается в том, чтобы сгенерировать особи, в которых отсутствуют одинаковые хромосомы.

3. Если поколение не первое, то перенести особей потомков в родительскую популяцию.

4. Подсчитываем приспособленность родительских особей. Для подсчета приспособленности особи необходимо соотнести хромосому и идущую за ней хромосому по порядку с соответствующими вершинами в графе смежности. Если между вершинами есть связь, увеличиваем приспособленность особи.

5. Генерируем первоначальные пары для скрещивания.

6. Сортируем особи.

7. Заменяем половину пар лучшими особями.

8.Скрещиваем особи кроссинговером. В результате получаются особи, с высокой долей вероятностью имеющие одинаковые хромосомы.

9. Мутация. Находим одинаковые хромосомы, заменяем их таким образом, чтобы в особи отсутствовали одинаковые хромосомы.

10. Проверяем, все ли особи одинаковы. Если да, то популяция выродилась.

11. Считаем приспособленность потомков. Если существует потомок с приспособленностью равной числу вершин, то останавливаем алгоритм и выводим ответ. Иначе, если популяция выродилась, то возвращаемся к шагу 2, если нет, то к шагу 3.

**3)Описание алгоритма**

На рисунке 1 изображена схема алгоритма.



Рисунок 1 –схема алгоритма



Рисунок 2 –схема алгоритма процедуры генерации особи

**4)Результаты выполнения программы для разных начальных данных.**

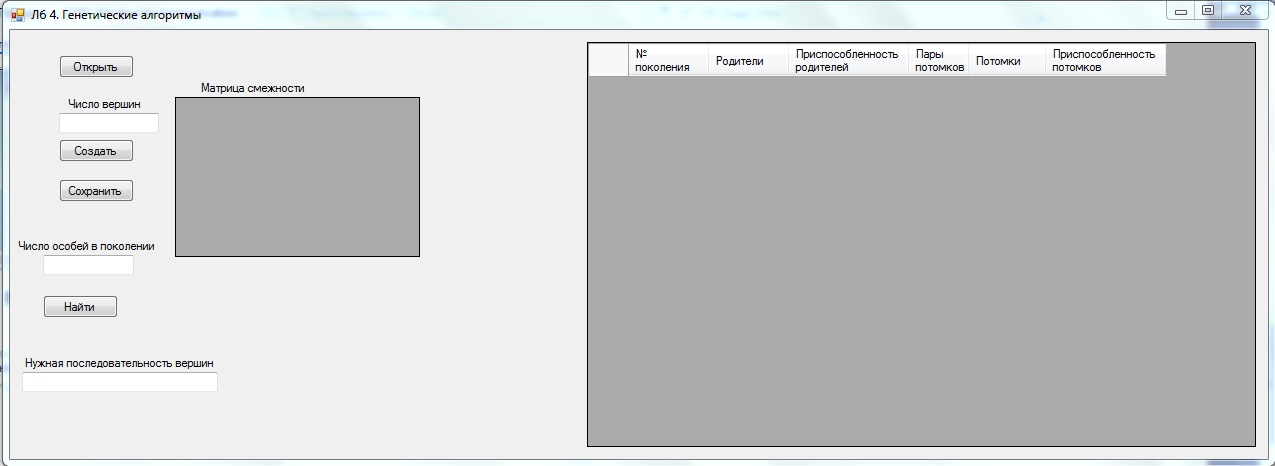


Рисунок 3 – экранная форма разработанной программы

На рисунках 4-6 представлены результаты выполнения программы для разных начальных данных.

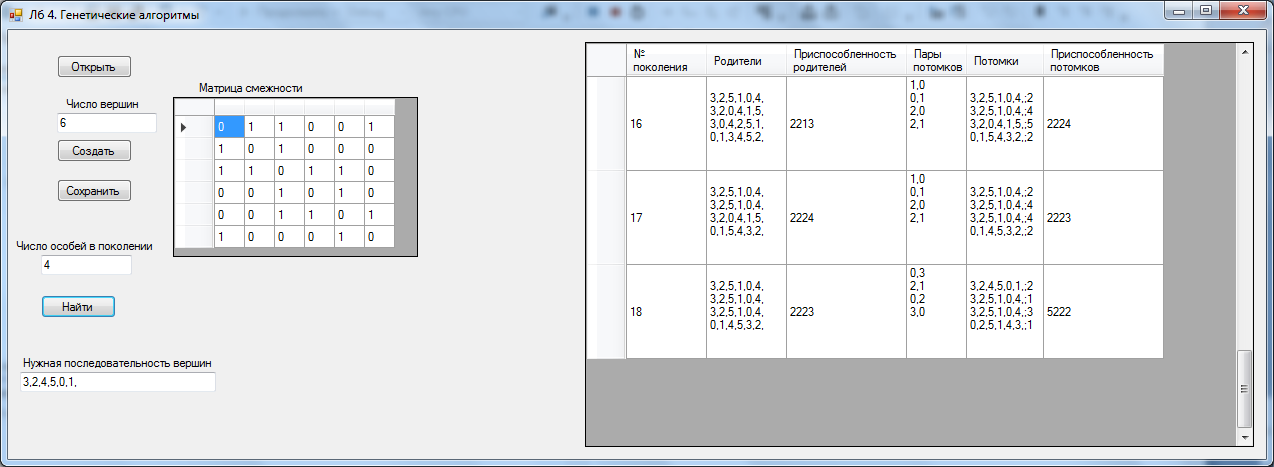


Рисунок 4 – выполнение программы при 6 вершинах и 4 особей в одном поколении

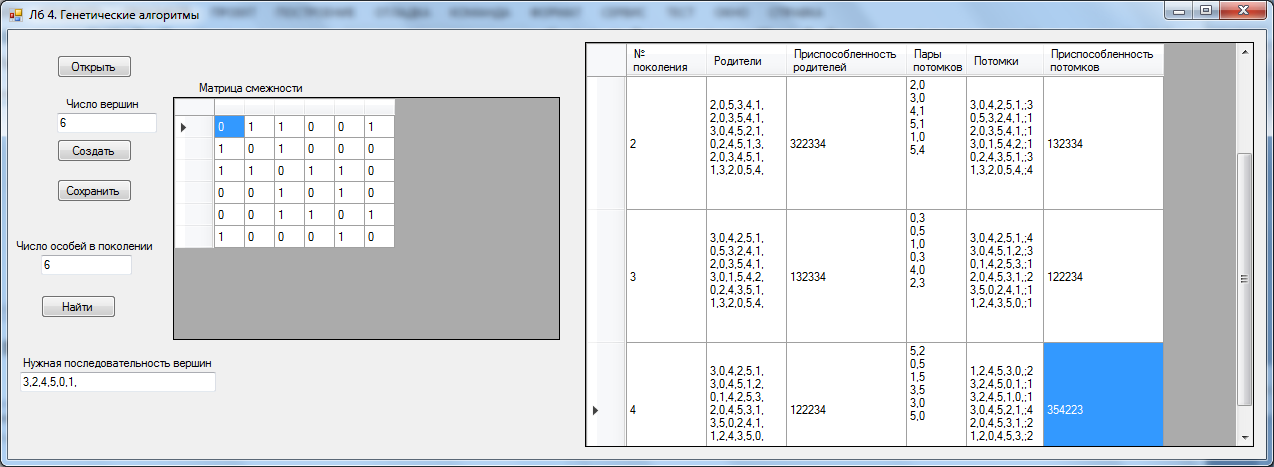


Рисунок 5 – выполнение программы при 6 вершинах и 6 особей в одном поколении

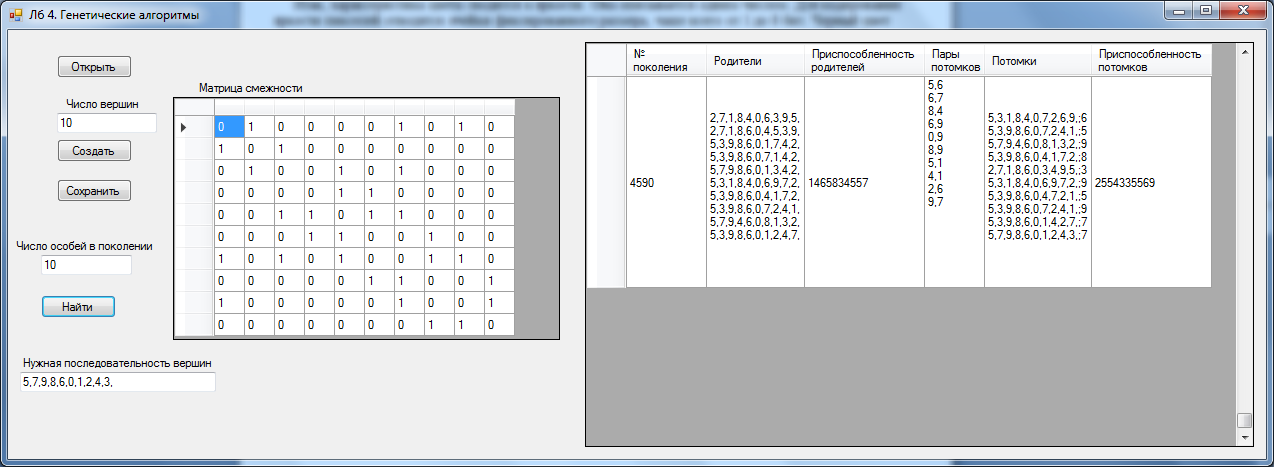


Рисунок 6 – выполнение программы при 10 вершинах и 10 особей в одном поколении

Приложение А

Листинг 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.Threading;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

int chromosome = Convert.ToInt32(textBox1.Text);

int count\_species = Convert.ToInt32(textBox2.Text);

int[] z\_c = new int[count\_species];

bool fatal = false;

bool finish=false;

int n\_generation = 0;

int[,] generation\_chield = new int[count\_species, chromosome];

while (!finish)

{

n\_generation++;

dataGridView1.Rows.Add();

dataGridView1.Rows[n\_generation - 1].Cells[0].Value = n\_generation.ToString();

int[,] generation\_parent = new int[count\_species, chromosome];

int[,] pair = new int[count\_species, 2];

int[,] z\_p = new int[count\_species,2];

//Генерация поколения

if ((n\_generation == 1)||(fatal==true))

{

for (int i = 0; i < count\_species; i++)

{

string species = "";

int[] buff = new int[chromosome];

buff = Generate(chromosome);

Thread.Sleep(1000);

for (int j = 0; j < chromosome; j++)

{

generation\_parent[i, j] = buff[j];

species += generation\_parent[i, j].ToString() + ',';

}

dataGridView1.Rows[n\_generation - 1].Cells[1].Value += species + "\n";

}

}

else

for (int i = 0; i < count\_species; i++)

{

string species = "";

for (int j = 0; j < chromosome; j++)

{

generation\_parent[i, j] = generation\_chield[i, j];

species += generation\_parent[i, j].ToString() + ',';

}

dataGridView1.Rows[n\_generation - 1].Cells[1].Value += species + "\n";

}

for (int i = 0; i < count\_species; i++)

{

z\_p[i,0] = 0;

for (int j = 0; j < chromosome - 1; j++)

{

int x = generation\_parent[i, j];

int x1 = generation\_parent[i, j + 1];

if (Convert.ToInt32(dataGridView2.Rows[x].Cells[x1].Value) == 1)

z\_p[i,0]++;

}

dataGridView1.Rows[n\_generation - 1].Cells[2].Value += z\_p[i,0].ToString();

}

Random r = new Random();

for (int i = 0; i < count\_species; i++)

{

string species = "";

pair[i, 0] = r.Next(count\_species);

int buff = r.Next(count\_species);

while (buff == pair[i, 0])

buff = r.Next(count\_species);

pair[i, 1] = buff;

species = pair[i, 0].ToString() + "," + pair[i, 1].ToString();

dataGridView1.Rows[n\_generation - 1].Cells[3].Value += species + "\n";

}

for (int i = 0; i < count\_species; i++)

z\_p[i, 1] = i;

for (int i = 0; i < count\_species-1; i++)

{

int i1 = i;

for (int j = i + 1; j < count\_species; j++)

if (z\_p[j, 0] < z\_p[i1, 0]) i1 = j;

if (i1 > i)

{

int b = z\_p[i1, 0];

z\_p[i1, 0] = z\_p[i, 0];

z\_p[i, 0] = b;

b = z\_p[i1, 1];

z\_p[i1, 1] = z\_p[i, 1];

z\_p[i, 1] = b;

}

}

int c = count\_species-1;

for (int i = count\_species-1 ; i>0; i=i-2)

{

string species = "";

pair[c, 0] = z\_p[i,1];

pair[c, 1] = z\_p[i - 1,1];

c--;

dataGridView1.Rows[n\_generation - 1].Cells[3].Value += species + "\n";

}

for (int i = 0; i < count\_species; i++)

{

string species = "";

int buffer = r.Next(1, chromosome);

for (int j = 0; j < buffer; j++)

generation\_chield[i, j] = generation\_parent[pair[i, 0], j];

for (int j = buffer; j < chromosome; j++)

generation\_chield[i, j] = generation\_parent[pair[i, 1], j];

for (int i1 = 0; i1 < chromosome; i1++)

for (int j = 0; j < chromosome; j++)

if ((generation\_chield[i, i1] == generation\_chield[i, j]) && (i1 != j))

{

int n = j;

while (n != 0)

{

int number = r.Next(chromosome);

generation\_chield[i, j] = number;

for (int j1 = 0; j1 < j; j1++)

if (generation\_chield[i, j] != generation\_chield[i, j1])

n--;

else

{

n = j;

}

}

}

for (int j = 0; j < chromosome; j++)

species += generation\_chield[i, j].ToString() + ",";

dataGridView1.Rows[n\_generation - 1].Cells[4].Value += species + ";" + buffer.ToString() + "\n";

}

for (int i = 0; i < count\_species; i++)

{

z\_c[i] = 0;

for (int j = 0; j < chromosome - 1; j++)

{

int x = generation\_chield[i, j];

int x1 = generation\_chield[i, j + 1];

if (Convert.ToInt32(dataGridView2.Rows[x].Cells[x1].Value) == 1)

z\_c[i]++;

}

dataGridView1.Rows[n\_generation - 1].Cells[5].Value += z\_c[i].ToString();

}

for (int i = 0; i < count\_species; i++)

{

if (z\_c[0] != z\_c[i])

{

fatal = false;

break;

}

else

fatal = true;

}

for (int i = 0; i < count\_species; i++)

{

if (z\_c[i] == chromosome - 1)

{

string species = "";

for (int j = 0; j < chromosome; j++)

species += generation\_chield[i, j].ToString() + ",";

textBox3.Text = species;

finish = true;

break;

}

}

}

}

int[] Generate(int n)

{

int[] a = new int[n];

Random r = new Random();

a[0] = r.Next(n);

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

{

bool flag = true;

while (flag)

{

int number = r.Next(n);

int j;

for (j = 0; j < i + 1; ++j)

if (number == a[j]) break;

if (j == i + 1)

{

a[i + 1] = number;

flag = false;

}

if ((i + 1) == n) break;

}

}

return a;

}

private void button2\_Click(object sender, EventArgs e)

{

dataGridView2.RowCount = Convert.ToInt32(textBox1.Text);

dataGridView2.ColumnCount = dataGridView2.RowCount;

for (int i = 0; i < dataGridView2.ColumnCount; i++)

{

dataGridView2.Columns[i].Width = 30;

for (int j = 0; j < dataGridView2.ColumnCount; j++)

dataGridView2.Rows[j].Cells[i].Value = "0";

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

string filename = openFileDialog1.FileName;

StreamReader f = new StreamReader(filename);

string s = "";

int j = 0;

int z = 0;

while (true)

{

s = f.ReadLine();

if (s == null) break;

if (z == 0)

{

int l = s.Length;

textBox1.Text = l.ToString();

dataGridView2.RowCount = l;

dataGridView2.ColumnCount = dataGridView2.RowCount;

for (int i = 0; i < dataGridView2.ColumnCount; i++)

dataGridView2.Columns[i].Width = 30;

z++;

}

for (int i=0;i<dataGridView2.ColumnCount;i++)

dataGridView2.Rows[j].Cells[i].Value=s[i].ToString();

j++;

}

f.Close();

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string filename = saveFileDialog1.FileName;

StreamWriter f = new StreamWriter(filename);

string s = "";

for (int j = 0; j < dataGridView2.RowCount; j++)

{

for (int i = 0; i < dataGridView2.RowCount; i++)

s += dataGridView2.Rows[j].Cells[i].Value.ToString();

f.WriteLine(s);

s = "";

}

f.Close();

}

}

}

}

Филиал федерального государственного бюджетного образовательного учреждения высшего образования

«Национальный исследовательский университет «МЭИ»

в городе Смоленске

Отчет

по лабораторной работе №3

Тема: «Использование генетических алгоритмов для решения

задачи дискретной оптимизации»

По курсу: «Системы искусственного интеллекта»

Бригада: Батулев А.И.

Восканян К.К.

Группа: ВМ-16(маг)

Преподаватели: Зернов М.М.

доц. Сеньков А.В.

Смоленск

2016 г.