# ГОУ ВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

### Кафедра АСУ

### 

Отчёт

о лабораторной работе №4

по курсу: «Разработка ГА решения задач комбинаторики»

# Вариант «2»

Выполнил:

ст. гр. ИСм-22

Мариничев И.И

Проверила:

Васяева Т.А

# 

# Донецк 2022

**Ход работы:**

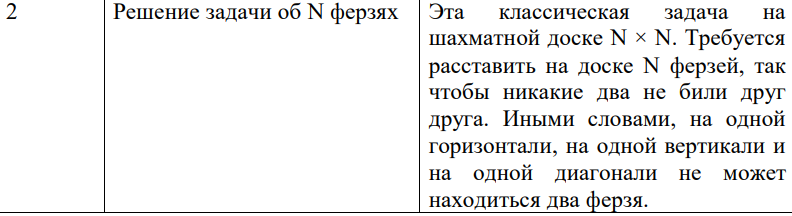
1. Разработать математическую постановку комбинаторной задачи – по варианту. Варианты заданий в табл. 4.2.

2. Разработать генетический алгоритм для решения поставленной задачи: представление особи; генетические операторы;функцию приспособленности.

3. Выполнить программную реализацию разработанного алгоритма на языке высокого уровня. Рекомендованный язык Python с применением DEAP. Предусмотреть возможность просмотра процесса поиска решения.

4. Выполнить тестирование программы на примере.

5. Исследовать зависимость числа поколений (генераций), скорости нахождения решения от основных параметров генетического алгоритма, например, число особей в популяции, вероятность кроссинговера, мутации; *Задание:*

**

Хромосома представляет из себя массив, где:

Индекс элемента – строка расположения ферзя;

Значение элемента – столбец.

Необходимо найти такую строку, что бы

Сумма: Y = 0;

Yi = if(|Индекс Xi – индекс Xi+1| == |значение Xi – значение Xi+1|) I else 0

**Алгоритм** кроссинговера следующий:

Для 2х родителей создаётся потомок, в которого входят совпадающие гены обоих родителей. Остальные гены рандомно заполняются оставшимися значениями.

Мутация:

2 случайных гена меняются местами.

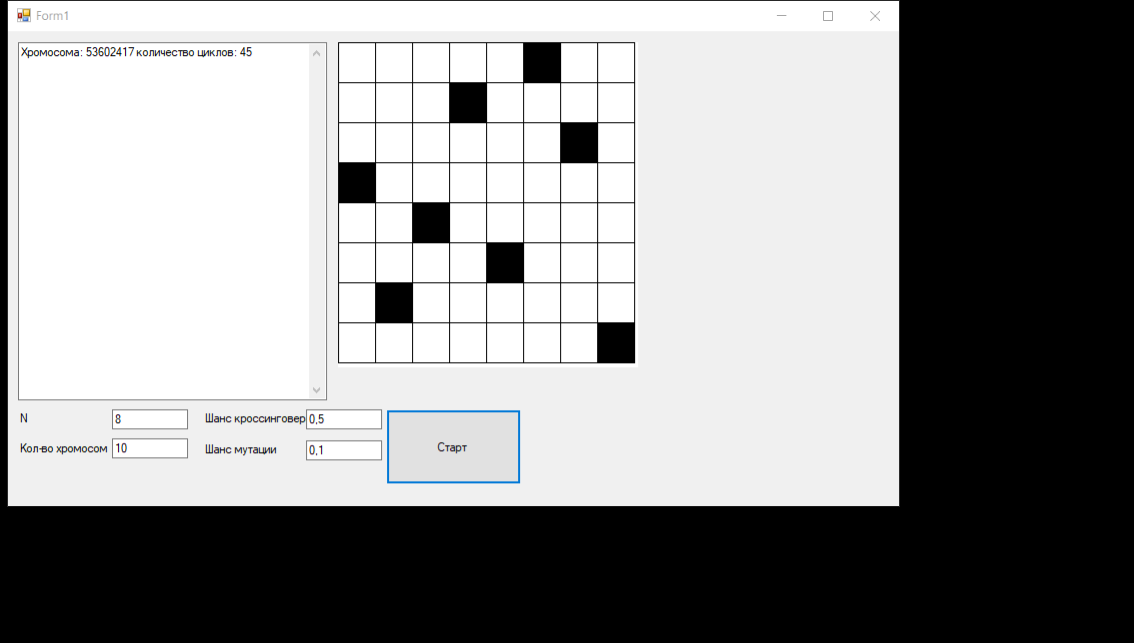
**Этап 1**

Для проверки работы алгоритма и сбора начальных данных, оставим лишь ограничение по количеству циклов.

За исходные данные примем:

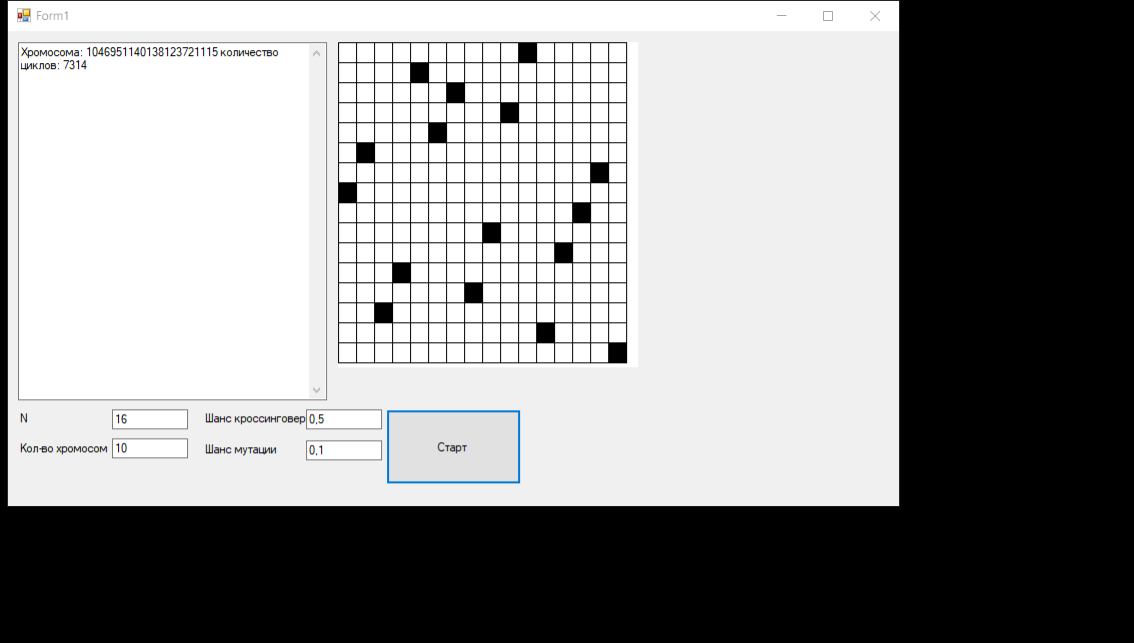
1. N = 8;
2. Кол-во хромосом = 10;
3. Шанс кроссинговера = 0,5;
4. Шанс мутации = 0,1.

Результаты:



Программа выполнила условия поиска.

Проверка при более сложных условиях:



При 16 ферзях количество циклов значительно увеличивается, однако скорость программы практически не падает.

При 17 ферзях, время выполнения заметно увеличилось.

При любом количестве ферзей свыше 17, время выполнения программы превышает допустимое время ожидания. Изменение параметров ГА, не способствует улучшению ситуации.

Листинг программного кода:

Форма1:

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

GeneticAlgorithm gen = new GeneticAlgorithm();

int N = Convert.ToInt32(textBox2.Text);

Chromosome best = gen.algorithm(N, Convert.ToInt32(textBox3.Text), Convert.ToDouble(textBox5.Text), Convert.ToDouble(textBox6.Text));

textBox1.Text = ("Хромосома: " + string.Join("", best.mass) + " количество циклов: " + best.colcikl);

Graphics g = pictureBox1.CreateGraphics();

g.Clear(Color.White);

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

{

for (int f = 0; f < N; f ++)

{

g.DrawRectangle(Pens.Black, ((pictureBox1.Width / N) \* f), ((pictureBox1.Height / N) \* j), pictureBox1.Width / N, pictureBox1.Height / N);

}

g.FillRectangle(Brushes.Black, ((pictureBox1.Width / N))\*best.mass[j], (pictureBox1.Height / N)\*j, pictureBox1.Width / N, pictureBox1.Height / N);

}

}

}

Класс1:

namespace Лаба\_4\_ИИ

{

class Chromosome

{

public int[] mass;

public int hits;

public int colcikl;

public Chromosome(int[] mass2)

{

mass = new int[mass2.Length];

mass2.CopyTo(mass, 0);

hits = mass2.Length;

}

public Chromosome(int n)

{

mass = new int[n];

mass.Initialize();

hits = n;

}

public Chromosome()

{

}

}

class GeneticAlgorithm

{

int colN, colH;

public List<Chromosome> chromosomes;

public StringBuilder textAll;

private Random random;

public GeneticAlgorithm()

{

chromosomes = new List<Chromosome>();

textAll = new StringBuilder();

random = new Random();

}

public void init(int n, int m)

{

colN = n;

colH = m;

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

{

chromosomes.Add(new Chromosome(chrString(n)));

}

}

public int[] chrString(int n)

{

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

int[] mass = new int[n];

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

{

list.Add(i);

}

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

{

int tmp = list[random.Next(0, list.Count - 1)];

mass[i] = tmp;

list.Remove(tmp);

}

return mass;

}

public Chromosome algorithm(int colN, int colChr, double crosschans, double mutchans)

{

init(colN, colChr);

List<Chromosome> newgen = new List<Chromosome>();

Chromosome best = new Chromosome();

bool b = true;

int count = 0;

while (b)

{

for (int j = 0; j < chromosomes.Count; j++)

{

if (random.NextDouble() <= crosschans)

{

newgen.Add(new Chromosome(crossingover(j)));

}

if (random.NextDouble() <= mutchans)

{

mutation(chromosomes[j]);

}

}

chromosomes.AddRange(newgen);

newgen.Clear();

for (int j = 0; j < chromosomes.Count; j++)

{

chromosomes[j].hits = crossCheck(chromosomes[j].mass);

if (chromosomes[j].hits == 0)

{

b = false;

best = chromosomes[j];

}

}

Chromosome tmp;

int tmpi;

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

{

tmp = chromosomes[k];

tmpi = k;

for (int l = 0; l < chromosomes.Count; l++)

{

if(tmp.hits > chromosomes[l].hits)

{

chromosomes[tmpi] = chromosomes[l];

chromosomes[l] = tmp;

tmpi = l;

}

}

}

for (int i = colChr; i < chromosomes.Count;)

{

chromosomes.RemoveAt(i);

}

count++;

}

best.colcikl = count;

return best;

}

public int[] crossingover(int p1num)

{

int p2num;

crossingerr: p2num = random.Next(0, chromosomes.Count - 1);

if (p2num == p1num)

{

goto crossingerr;

}

int[] child = new int[chromosomes[p1num].mass.Length];

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

tmplist.AddRange(chromosomes[p1num].mass.ToList());

for (int i = 0; i < chromosomes[p1num].mass.Length; i++)

{

if (chromosomes[p1num].mass[i] == chromosomes[p2num].mass[i])

{

child[i] = chromosomes[p1num].mass[i];

tmplist.Remove(child[i]);

}

else child[i] = -1;

}

for (int i = 0; i < chromosomes[p1num].mass.Length; i++)

{

if (child[i]==-1)

{

int tmp = tmplist[random.Next(0, tmplist.Count - 1)];

child[i] = tmp;

tmplist.Remove(tmp);

}

}

return child;

}

public void mutation(Chromosome chromosome)

{

int i, j;

int tmp;

i = random.Next(0, chromosome.mass.Length - 1);

muterror: j = random.Next(0, chromosome.mass.Length - 1);

if (j == i) goto muterror;

tmp = chromosome.mass[i];

chromosome.mass[i] = chromosome.mass[j];

chromosome.mass[j] = tmp;

}

public int crossCheck(int[] mass)

{

int sum = 0;

for(int i=0; i< mass.Length-1; i++)

{

for (int j = i+1; j < mass.Length; j++)

{

if (j - i == Math.Abs(mass[j] - mass[i])) sum++;

}

}

return sum;

}

}

}