Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

*Кафедра програмного забезпечення автоматизованих систем*

**Лабораторна робота №7**

Тема:

**«**Генетичні алгоритми**»**

***Виконав*:**

ст. групи ПІТ-15-3

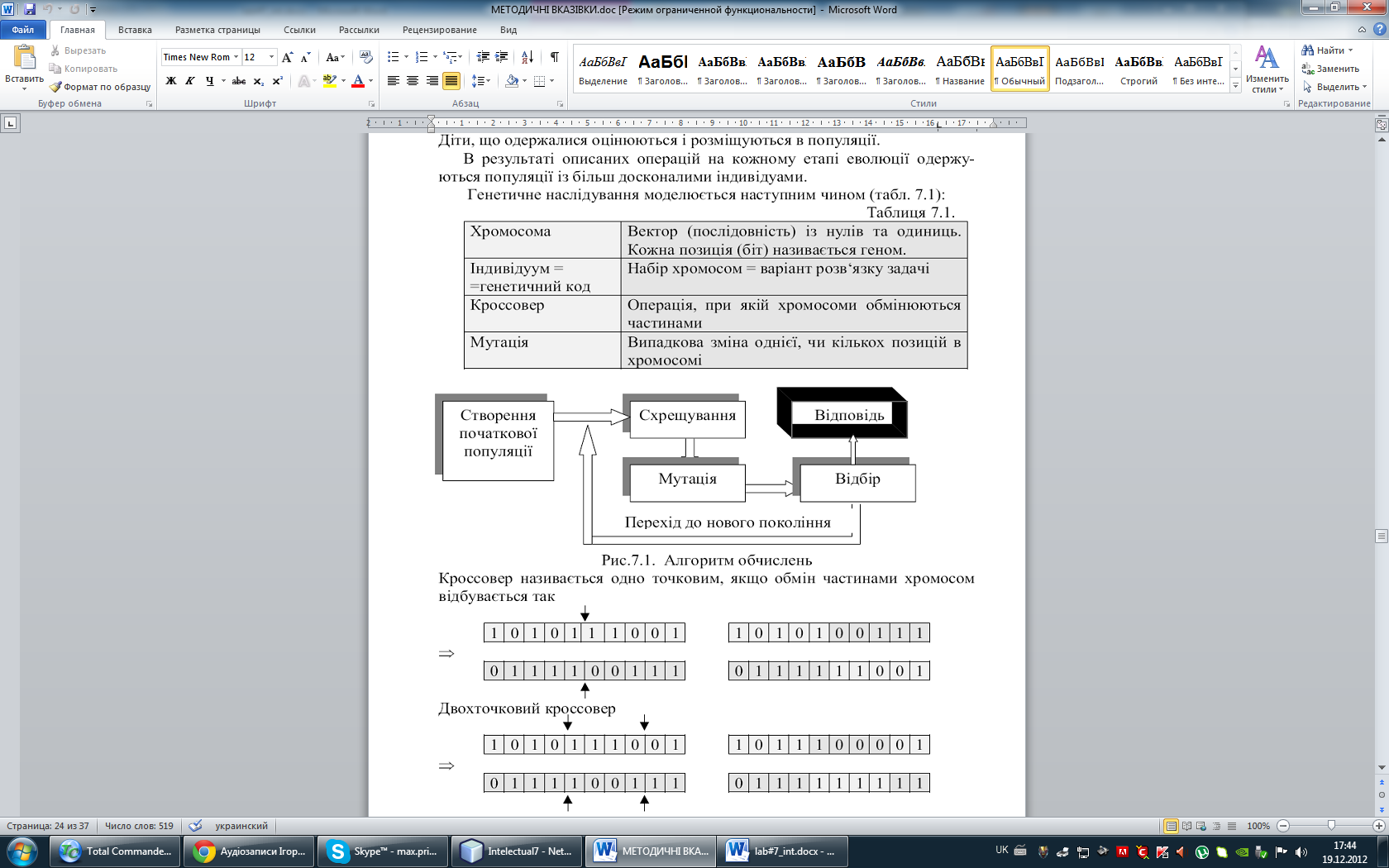
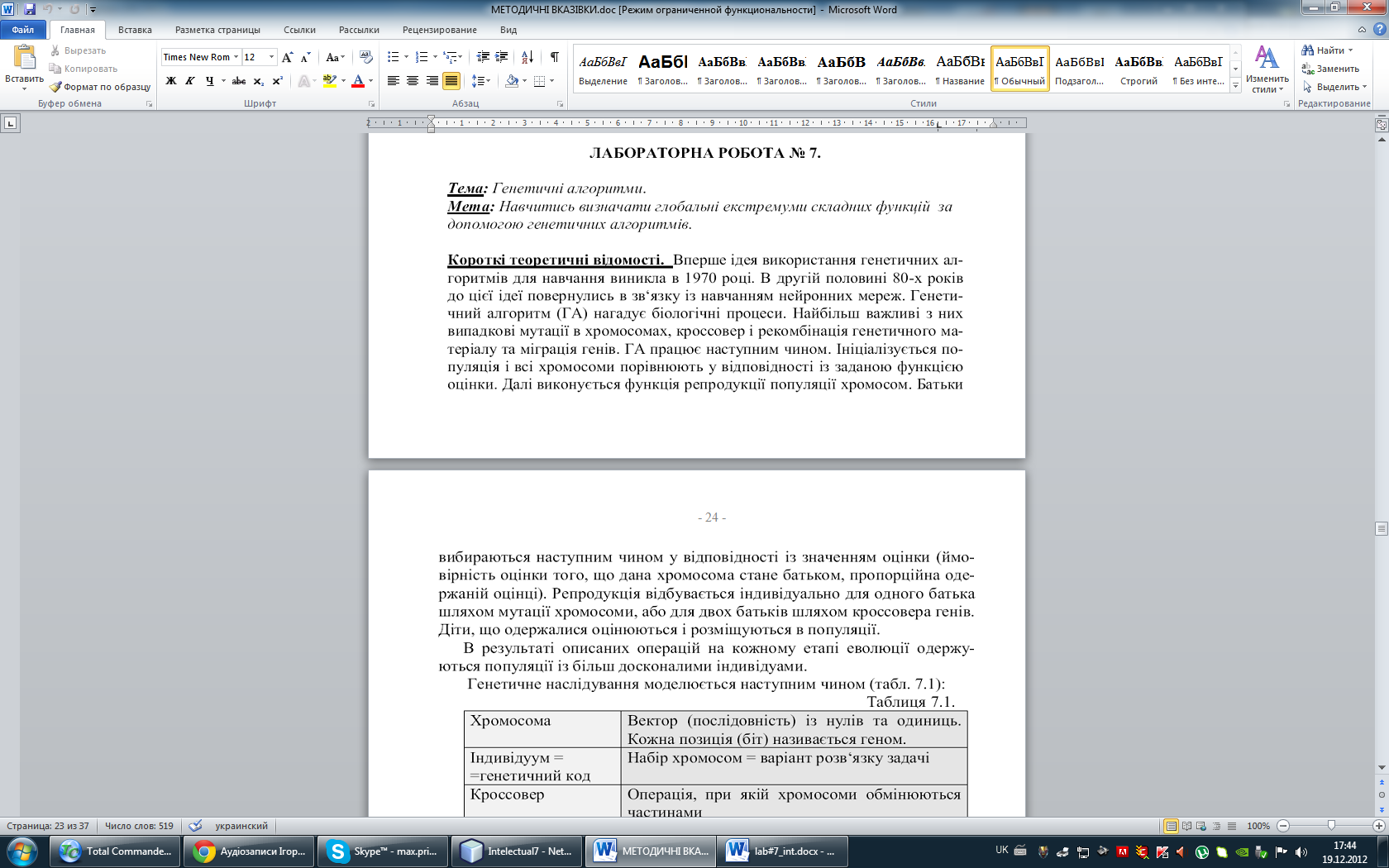
Свирид О. Б.

***Перевірив:***

Дитко Т.В.

Івано-Франківськ

2015



**Хід роботи**

class Lab7 {

long BEGIN;

final boolean allow = true;

double err = 0.02;

LinkedList<Gener> population = new LinkedList<Gener>();

final Random rand;

final int populationSize = 10;

final int maxParents = 10;

public Lab7() {

rand = new Random();

for (int i = 0; i < populationSize; i++) {

Gener c = new Gener();

c.random();

population.add(c);

}

Collections.sort(population);

print();

}

void print() {

for (Gener c : population) {

jTextArea1.append(c+"\n");

}

}

public double func(double x1, double x2)

{

return (20 + x1\*x1 + x2\*x2 - 10\*Math.cos(2\*3.14\*x1) - 10\*Math.cos(2\*3.14\*x2));

}

double produceNextGen(double x1, double x2) {

LinkedList<Gener> newpopulation = new LinkedList<Gener>();

while (newpopulation.size() < populationSize \* (1.0-(maxParents/100.0)) ) {

int size = population.size();

int i = rand.nextInt(size);

int j, k, l;

j = k = l = i;

while (j == i)

j = rand.nextInt(size);

while (k == i || k == j)

k = rand.nextInt(size);

while (l == i || l == j || k == l)

l = rand.nextInt(size);

Gener c1 = population.get(i);

Gener c2 = population.get(j);

Gener c3 = population.get(k);

Gener c4 = population.get(l);

int f1 = c1.result();

int f2 = c2.result();

int f3 = c3.result();

int f4 = c4.result();

Gener w1, w2;

if (f1 > f2)

w1 = c1;

else

w1 = c2;

if (f3 > f4)

w2 = c3;

else

w2 = c4;

Gener child1, child2;

Gener[] childs = newChilds(w1,w2);

child1 = childs[0];

child2 = childs[1];

double Mutation = 0.01;

boolean m1 = rand.nextFloat() <= Mutation;

boolean m2 = rand.nextFloat() <= Mutation;

if(m1)mutate(child1);

if(m2)mutate(child2);

boolean isChild1Good = child1.result() >= w1.result();

boolean isChild2Good = child2.result() >= w2.result();

newpopulation.add( isChild1Good ? child1 : w1);

newpopulation.add( isChild2Good ? child2 : w2);

}

int j = (int)(populationSize\*maxParents/100.0);

for (int i = 0; i < j; i++) {

newpopulation.add( population.get(i));

}

population=newpopulation;

Collections.sort(population);

return func(x1,x2);

}

Gener newChild(Gener c1, Gener c2, int zmish)

{

Gener child = new Gener();

for (int i = 0; i < zmish; i++) {

child.genotype[i] = c1.genotype[i];

}

for (int j = zmish; j < Gener.SIZE; j++) {

child.genotype[j] = c2.genotype[j];

}

return child;

}

Gener[] newChilds(Gener c1, Gener c2)

{

Gener child1 = new Gener();

Gener child2 = new Gener();

for (int i = 0; i < Gener.SIZE; i++) {

boolean b = rand.nextFloat() >= 0.5;

if(b){

child1.genotype[i] = c1.genotype[i];

child2.genotype[i] = c2.genotype[i];

}

else

{

child1.genotype[i] = c2.genotype[i];

child2.genotype[i] = c1.genotype[i];

}

}

return new Gener[]{child1,child2} ;

}

void mutate(Gener c) {

int i = rand.nextInt(Gener.SIZE);

c.genotype[i] = !c.genotype[i];

}

public void start() {

GA ga = new GA();

ga.run();

}

void run() {

double fridge1 = 0, fridge2 = 0;

double extremum = func(0,0);

double mFridge1 = -5.12;

double mFridge2 = -5.12;

for(int i = 0; i < 512; i++)

{

mFridge2+=err;

mFridge2 = -5.12;

for(int j = 0; j < 512; j++)

{

mFridge2+=err;

if(produceNextGen(mFridge1, mFridge2)>extremum){

extremum = produceNextGen(mFridge1, mFridge2);

fridge1 = mFridge1;

fridge2 = mFridge2;

}

}

}

jTextArea1.append("Фінальний екстремум функції: "+extremum+ " при x1="+fridge1+", x2 = "+fridge2);

}

public class Gener implements Comparable<Gener> {

public static final int SIZE = 10;

public boolean[] genotype;

public Gener() {

genotype = new boolean[SIZE];

}

void random()

{

for (int i = 0; i < genotype.length; i++) {

genotype[i] = 0.5 > rand.nextFloat();

}

}

private String gene()

{

StringBuilder sb = new StringBuilder();

for (int i = 0; i < genotype.length; i++) {

sb.append(genotype[i] == true ? 1 : 0); }

return sb.toString();

}

int result() {

int sum = 0;

for (int i = 0; i < genotype.length; i++) {

if (genotype[i])

sum++;

}

return sum;

}

public int compareTo(Gener o) {

int f1 = this.result();

int f2 = o.result();

if (f1 < f2) return 1;

else if (f1 > f2) return -1;

else return 0;

}

public String toString()

{

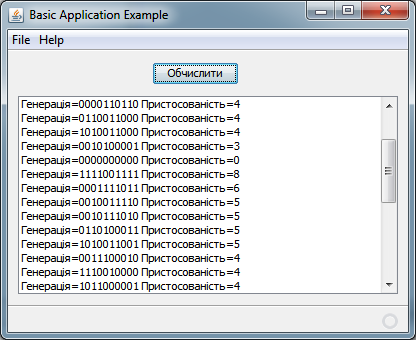
return "Генерація="+gene()+" Пристосованість="+result();

}

}

}

**Результати виконання програми**



**Висновок:** під час даної лабораторної роботи я вивчив принципи роботи і навчання генетичного алгоритму і реалізував його програмно.