МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра СТ

Звіт

з лабораторної роботи № 1

за темою «Оптимізація складних функцій за допомогою генетичних алгоритмів»

з дисципліни «Нейросистеми та Генетичні алгоритми»

|  |  |
| --- | --- |
| Виконав:  ст. гр. КСУАм-15-1  Ковтунець А.О | Перевірила:  Доц. каф. СТ  Імангулова З. А. |

Харків 2016

1 ОПТИМІЗАЦІЯ СКЛАДНИХ ФУНКЦІЙ ЗА ДОПОМОГОЮ ГЕНЕТИЧНИХ АЛГОРИТМІВ

1.1 Мета роботи

Вивчення принципів роботи генетичних алгоритмів. Набуття навичок реалізації генетичних алгоритмів для оптимізації складних багатопараметричних функцій.

1.2 Хід роботи

#### Таблиця 1.2 – Варіанти завдань

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Но-мер | Функція | Крос-совер | Вибір батьків | Механізм добору |
| 3 |  | Рівно-мірний | Інбридинг | Елітний |

Реалізували програмний продукт класичного генетичного алгоритму, який знаходить максимум функції підбираючи вхідні параметри. Результати зображено на рисунку 1.1.

Код програми:

package ua.nure;

import java.text.DecimalFormat;

import java.text.DecimalFormatSymbols;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Locale;

import java.util.Map;

import java.util.Random;

import java.util.TreeMap;

public class Main {

static Random rand = new Random();

static DecimalFormat df = new DecimalFormat("#.##");

static DecimalFormatSymbols otherSymbols = new DecimalFormatSymbols(Locale.getDefault());

public static void main(String[] args) {

otherSymbols.setDecimalSeparator('.');

df = new DecimalFormat("#.##", otherSymbols);

final float x1\_min = 0.5f, x1\_max = 1.1f, x2\_min = 1.0f, x2\_max = 4.6f;

final int n = 500, amount = 2;

float instance[][] = new float[n][amount];

//generate instance

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

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

if (j == 0)

instance[i][j] = Float.parseFloat(df.format( rand.nextFloat() \* (x1\_max - x1\_min) + x1\_min )) ;

if (j == 1)

instance[i][j] = Float.parseFloat(df.format( rand.nextFloat() \* (x2\_max - x2\_min) + x2\_min )) ;

}

}

String[][] s\_instance = get\_StringBinaryInstance(n, amount, x1\_min, x1\_max, x2\_min, x2\_max);

// parent's choice

long timet\_start, timer\_end;

timet\_start=System.nanoTime();

ArrayList<String> crossedChromosomes = new ArrayList<String>();

TreeMap<Float,Integer> treeMap = new TreeMap<Float,Integer>(Collections.reverseOrder());

int count=(int) (n\*0.9);

int parent1 = 0,parent2=0;

float[] best\_rezult=new float[3];

boolean mask[][]=new boolean[n][amount];

String chromosome1 = null,chromosome2= null;

//цыкл

for(int ii=0; ii<50; ii++){

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

parent1=rand.nextInt(499);

// mask[parent1][0]=true;

chromosome1 = s\_instance[parent1][0];

//вибір parent2

chromosome2=get\_SimilarParent(chromosome1, s\_instance, count);

//схрещування

crossedChromosomes=get\_crossedChromosomes(crossedChromosomes, chromosome1, chromosome2);

}

//велика хромосома)

StringBuffer long\_Chromosome = new StringBuffer();

for(String s: crossedChromosomes){

long\_Chromosome.append(s);

}

// мутація

for(int i=0; i<long\_Chromosome.length();i++){

if(rand.nextDouble()<=0.01 ) {long\_Chromosome.setCharAt(i, long\_Chromosome.charAt(i)==0 ? '1':'0'); }

}

//розбиваэм велику хромосому

crossedChromosomes.clear();

for(int i=0; i<=long\_Chromosome.length()-30; i+=30){

crossedChromosomes.add(long\_Chromosome.substring(i, i+30));

}

// функція

for(int i=0; i<crossedChromosomes.size()-1;i+=2){

treeMap.put((float) ((-2\*Math.pow(get\_float(crossedChromosomes.get(i+1)), 3)+6\*Math.pow(get\_float(crossedChromosomes.get(i+1)), 2)+10)\*Math.sin(Math.log(get\_float(crossedChromosomes.get(i)))\*Math.exp(get\_float(crossedChromosomes.get(i+1))))) , i) ;

}

if(best\_rezult[0]<treeMap.firstKey()){

best\_rezult[0]=treeMap.firstKey();

best\_rezult[1]=get\_float(crossedChromosomes.get(treeMap.get(treeMap.firstKey())));

best\_rezult[2]=get\_float(crossedChromosomes.get(treeMap.get(treeMap.firstKey()))+1);

}

// ннова популяція

int it=0;

for(Map.Entry<Float, Integer> entry : treeMap.entrySet()){

if(it>count)break;

s\_instance[it][0]=crossedChromosomes.get(entry.getValue());

s\_instance[it][1]=crossedChromosomes.get(entry.getValue()+1);

it+=2;

}

treeMap.clear();

crossedChromosomes.clear();

System.out.println("Iteration #"+ii);

System.out.println("Best rezult="+best\_rezult[0] + " x1=" + best\_rezult[1] + " x2=" + best\_rezult[2]);

}

System.out.println();

System.out.println("Best rezult="+best\_rezult[0] + " x1=" + best\_rezult[1] + " x2=" + best\_rezult[2]);

timer\_end=System.nanoTime();

System.out.println("Time"+String.format("%,12d", timer\_end-timet\_start) + " ns") ;

}

public static float[][] get\_FloatInstance (int n, int amount, float x1\_min, float x1\_max, float x2\_min, float x2\_max){

float instance[][] = new float[n][amount];

//generate instance

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

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

if (j == 0)

instance[i][j] = Float.parseFloat(df.format( rand.nextFloat() \* (x1\_max - x1\_min) + x1\_min )) ;

if (j == 1)

instance[i][j] = Float.parseFloat(df.format( rand.nextFloat() \* (x2\_max - x2\_min) + x2\_min )) ;

}

}

return instance;

}

public static String[][] get\_StringInstance (int n, int amount, float x1\_min, float x1\_max, float x2\_min, float x2\_max){

String instance[][] = new String[n][amount];

//generate instance

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

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

if (j == 0)

instance[i][j] = df.format( rand.nextFloat() \* (x1\_max - x1\_min) + x1\_min ) ;

if (j == 1)

instance[i][j] = df.format( rand.nextFloat() \* (x2\_max - x2\_min) + x2\_min ) ;

}

}

return instance;

}

public static String[][] get\_StringBinaryInstance (int n, int amount, float x1\_min, float x1\_max, float x2\_min, float x2\_max){

String instance[][] = new String[n][amount];

//generate instance

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

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

if (j == 0)

// WAT? генеруєм значення в діапазаоні; df.format обрізаєм лишні знаки(точність) отримуєм String(float); Float.parseFloat отримуєм Float(String); Float.floatToRawIntBits побітово переводимось в int(float); Integer.toBinaryString отримуєм двійковий код String->Int(Float)

instance[i][j] = Integer.toBinaryString(Float.floatToRawIntBits(Float.parseFloat(df.format( rand.nextFloat() \* (x1\_max - x1\_min) + x1\_min ))));

if (j == 1)

instance[i][j] = Integer.toBinaryString(Float.floatToRawIntBits(Float.parseFloat(df.format( rand.nextFloat() \* (x2\_max - x2\_min) + x2\_min ))));

}

}

return instance;

}

public static String get\_SimilarParent(String chromosome1, String[][] s\_instance, int count ){

String hromosome2 = null;

int max\_value=Integer.MIN\_VALUE, index = 0;

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

int i=0,sum=0;

for(char h: chromosome1.toCharArray()){

if(h==s\_instance[j][1].charAt(i)) sum++;

i++;

}

if(max\_value<sum) {index=j; max\_value =sum;}

}

hromosome2=s\_instance[index][1];

return hromosome2;

}

public static ArrayList<String> get\_crossedChromosomes(ArrayList<String> crossedChromosomes,String chromosome1, String chromosome2){

StringBuffer new\_chromosome1 = new StringBuffer(chromosome1);

StringBuffer new\_chromosome2 = new StringBuffer(chromosome2);

for(int i=1; i<chromosome1.length();i++){

if(i%2!=0) new\_chromosome1.setCharAt(i, chromosome1.charAt(i)); if (i%2==0) new\_chromosome1.setCharAt(i, chromosome2.charAt(i-1));

if(i%2==0) new\_chromosome2.setCharAt(i, chromosome2.charAt(i)); if (i%2!=0) new\_chromosome2.setCharAt(i, chromosome1.charAt(i-1));

}

crossedChromosomes.add(new\_chromosome1.toString());

crossedChromosomes.add(new\_chromosome2.toString());

return crossedChromosomes;

}

public static float get\_float(String crosedChromosome){

return Float.intBitsToFloat((Integer.parseInt(crosedChromosome,2)));

}

}

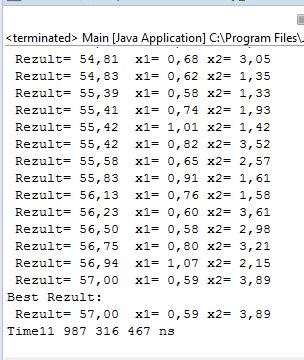


Рисунок 1.1 – Результати роботи програми

ВИСНОВКИ

В ході лабораторної роботи вивчили принципи генетичних алгоритмів. Набули навички реалізації генетичних алгоритмів для оптимізації складних багатопараметричних функцій.