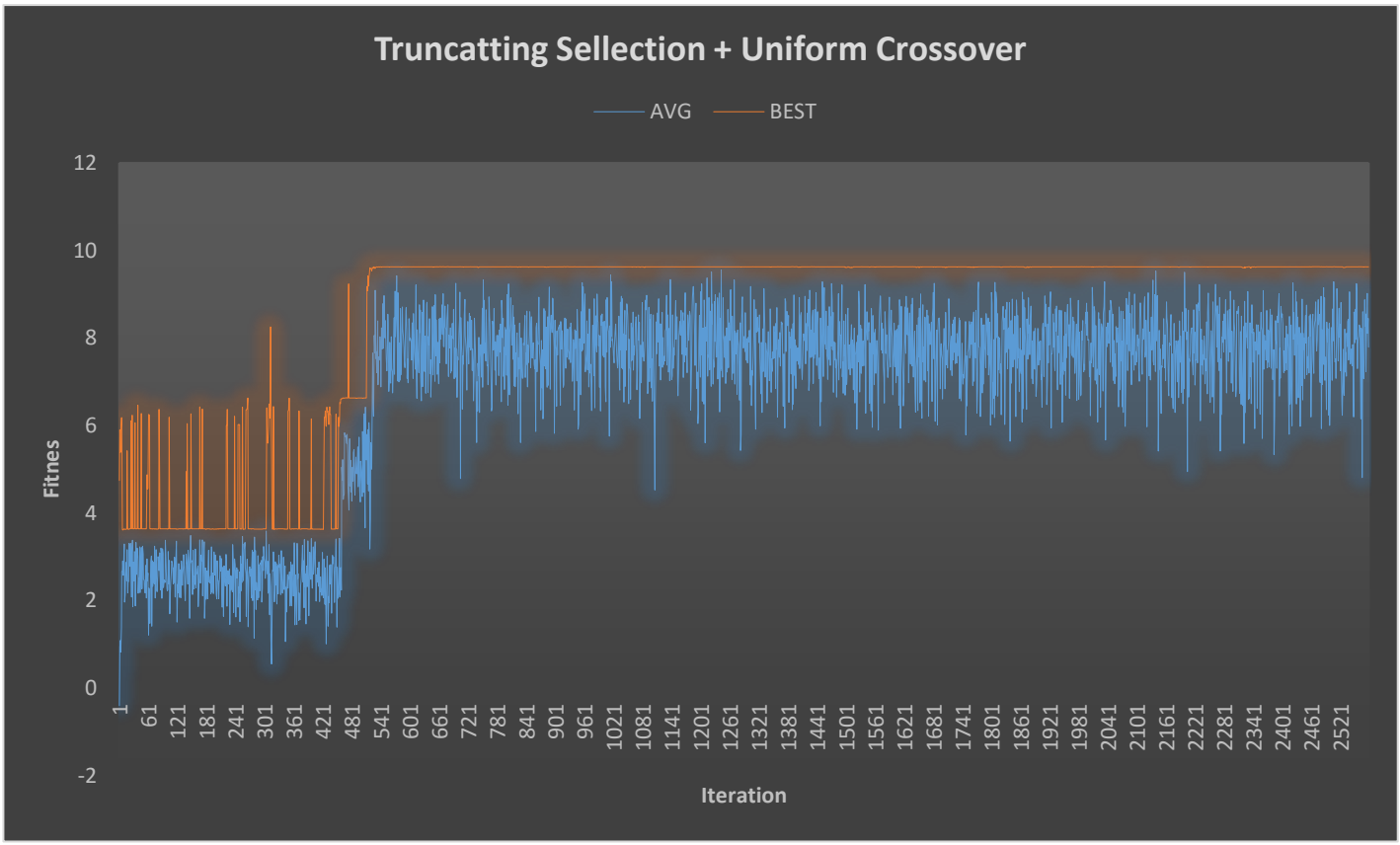
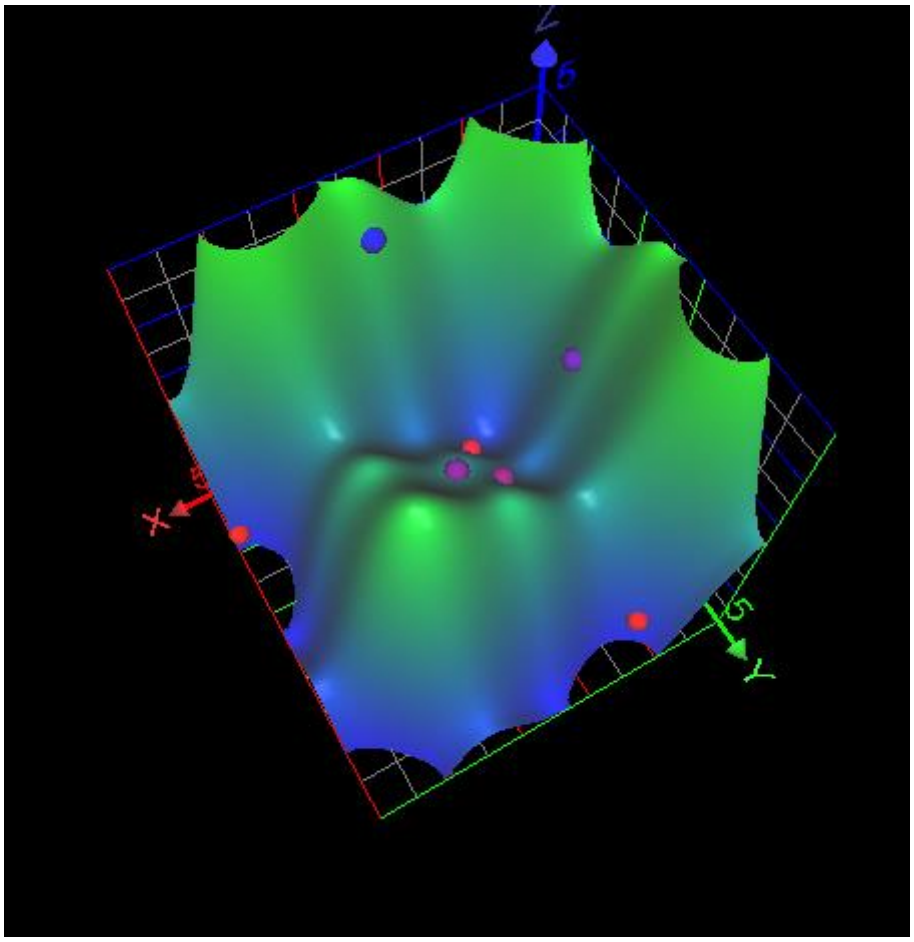
Student: Kirill Tsibikov

1. Maximization Function

Brest 2016

Function: y=x1*sin(abs(x1))+x2*sin(abs(x2))





Listing

```
      package javaapplication1;
      import java.util.Map;
      import java.util.Random;
      class GeneticAlgo{
          boolean[][] popln;
          public enum SelectionType {
            TOURNEY, ROULETTE_WHEEL, TRUNCATING
      }
          public enum CrossingType {
            ONE_POINT_RECOMBINATION, TWO_POINT_RECOMBINATION,
ELEMENTWISE_RECOMBINATION, ONE_ELEMENT_EXCHANGE
      }
          private SelectionType slctp;
          private CrossingType crstp;
          private int genomLength; //Длина генома в битах
          private int generationCount; //Кол-во поколений
          private int individualCount; //Кол-во
Геномов(Индивидов,Особей) в поколении
          private int[] chosenchromosom;
          private SelectionType selectionType; //Тип Селекции
          private CrossingType crossingType; //Тип Скрещивания
          public GeneticAlgo(String s, String c){
              slctp=SelectionType.valueOf(s);
```

```
            crstp =CrossingType.valueOf(c);
            popln=new  boolean[100][1000];
            genomLength=1000;
            generationCount=100;
            individualCount=100;
            chosenchromosom=new int[100];
        }
        public boolean[][] run(){
            this.generateFirstGeneration();
            float ftnsmax=0;
            for(int i=0; i<100000000; i++)
            {
                this.selection();
                for(int j=0; j<100; j++)
                {
                    ftnsmax=(ftnsmax<=fitnes(j))?fitnes(j):ftnsmax;
                }
    //            System.out.print(avgfitnes());
    //            System.out.print(":");
    //            System.out.print(ftnsmax);
    //            System.out.println();
                if(i==0||i==500||i==800)
                {
                    System.out.println("The " + i + " population");
                    for(int j=0; j<100; j++)
                    {
                        System.out.print(j+1);
                        System.out.print(":");
                        System.out.print(this.chosenchromosom[j]);
                        System.out.println();
                    }
                }
                if(ftnsmax==1000)break;
            }

            return (new boolean[100][1000]);
        }
        private void generateFirstGeneration() {
        Random rnd=new Random();
            for(int i=0; i<100; i++)
            {
                for(int j=0; j<1000; j++)
                {
                    popln[i][j]=rnd.nextBoolean();
                }
            }
        } //генерация первого поколения
        private void selection(){
            boolean[][] genomListOffsprings=new boolean[100][1000];
            Random rndd=new Random();
            switch(this.slctp)
            {
                case ROULETTE_WHEEL:{
                    float[] wheel = new float[this.individualCount];
```

```
                wheel[0] = fitnes(0);//Значение ФитнессФункции для 1-
ого генома
                     this.chosenchromosom[0]=0;
                for (int i=1;i<this.individualCount;i++){
                           wheel[i] = wheel[i-1] + fitnes(i);//Значение
ФитнессФункции для i-ого генома
                                      this.chosenchromosom[i]=0;
                     }
                float all = wheel[this.individualCount-1];

                for (int i=0;i<this.individualCount;i++){
                          float index = Math.abs(rndd.nextFloat())*all;
                          int l = 0;
                          int r = individualCount-1;
                          int c = 0;
                          while (l < r){
                     c = (l+r) >> 1;
                     if (index <= wheel[c])
                               r = c;
                           else
                        l = c + 1;
                          }
                          int a=l;
                          index = Math.abs(rndd.nextFloat())*all;

                          l = 0;
                          r = individualCount-1;
                          c = 0;
                          while (l < r){
                     c = (l+r) >> 1;
                     if (index <= wheel[c])
                               r = c;
                           else
                        l = c + 1;
                          }
                          this.chosenchromosom[l]++;
                          this.chosenchromosom[a]++;
                          genomListOffsprings[i] = this.crossing(l,a);
                }
                     popln=genomListOffsprings;
                     break;
                 }
                 case TOURNEY:
                 {
                     for (int i=0;i<this.individualCount;i++){
                          int index1 = rndd.nextInt(individualCount);
                          int index2 = rndd.nextInt(individualCount);
                          int index3 = rndd.nextInt(individualCount);
                          int index4 = rndd.nextInt(individualCount);
                          float fr1 = fitnes(index1);
                          float fr2 = fitnes(index2);
                          index1=(fr1>fr2)?index1:index2;
                          float fr3 = fitnes(index3);
                          float fr4 = fitnes(index4);
```

```
                        index2=(fr3>fr4)?index3:index4;
                        genomListOffsprings[i] =
this.crossing(index1, index2);
                }
                    popln=genomListOffsprings;
                    break;
                }
                case TRUNCATING:
                {
                    int percent=(Math.abs(rndd.nextInt())+10)%50+1;
                    this.sort();
                    for(int i=0; i<this.individualCount; i++)
                    {
                        genomListOffsprings[i] =
this.crossing((Math.abs(rndd.nextInt()))%50,(Math.abs(rndd.nextInt()))
%50);
                    }
                    popln=genomListOffsprings;
                    break;
                }
                default:
                    break;

        }
        } //Процедура селекци
        private boolean[] crossing(int a, int b) {
            boolean[] vec=new boolean[1000];
            switch(crstp)
            {
                case ONE_POINT_RECOMBINATION:
                {
                 for(int i=0; i<genomLength; i++)
                 {
                     Random rndd=new Random();

vec[i]=(rndd.nextBoolean())?popln[b][i]:popln[a][i];
                 }
                    break;
                }
                default:
                    break;
            }


        return vec;
        } //Процедура скрещивания
        private float fitnes(int nomber){
            float ftns=0.f;
            for(int j=0; j<genomLength; j++)
                {
                    ftns+=(this.popln[nomber][j])?1:0;
                }
            return ftns;
        } //Фитнес функция
```

```
        private float avgfitnes(){
            float ftns=0.f;
            for(int i=0; i<100; i++)
            for(int j=0; j<genomLength; j++)
                {
                    ftns+=(this.popln[i][j])?1:0;
                }
            return ftns/100;
        } //Фитнес функция
        private void sort()
        {
            for(int i=0; i<this.individualCount; i++)
            {
                boolean[] amiba;
                amiba = new boolean[1000];
                amiba=popln[i];
                double fit=fitnes(i);
                for(int j=i; j<this.individualCount; j++)
                {
                    if(fitnes(j)>fit){
                        fit=fitnes(j);
                        amiba=popln[j];
                        popln[j]=popln[i];
                        popln[i]=amiba;
                    }
                }
            }
        }
    }
    public class JavaApplication1 {
        public static void main(String[] args) {
            GeneticAlgo p=new
GeneticAlgo("ROULETTE_WHEEL","ONE_POINT_RECOMBINATION" );
            p.run();
          }
    }
```