**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №5**

**по дисциплине «Оптимальное проектирование»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3301 |  | Малинина Е.А. |
| Преподаватель |  | Каримов А.И. |

Санкт-Петербург

2018

**Лабораторная работа №5**   
Исследование схемы «генетический алгоритм – классический метод оптимизации»

Цель работы

Модификация программы, исполненной в лабораторной работе 4 для изучения схемы «генетический алгоритм – классический метод оптимизации». При выполнении данной лабораторной работы предлагается выбрать целочисленное кодирование хромосом.

Задание

Написать программу, в которой популяция 70 особей, турнирный отбор (размер 5), 1-точечное скрещивание (*pc* = 0,75), инверсия (*pI* = 0,07), условие остановки - 50 итераций. После останова генетического алгоритма лучшую особь используем в качестве стартовой точки для метода Хука-Дживса

Длина гена *m* = 12.

|  |  |  |
| --- | --- | --- |
| № | Функция *y*(***x***) | Поисковый интервал |
| (5) | (*x*2 – *x*12)2 + (1 – *x*1)2 | *x*1∈ (–5,12; 5,12); *x*2 ∈ (–5,12; 5,12); |
| (15) | [1.5 – x1(1 – x2)]2 + [2.25 – x1(1 – x22)]2 + + [2.625 – x1(1 – x23)]2 | *x1 ∈ (–5,12; 5,12); x2 ∈ (–5,12; 5,12);* |

Алгоритм

**Начало алгоритма: Установить точку t0 с начальными координатами, полученными другими методами.**

**Шаг 1. Выполнить шаг исследования по схеме, представленной на рисунке 1.**

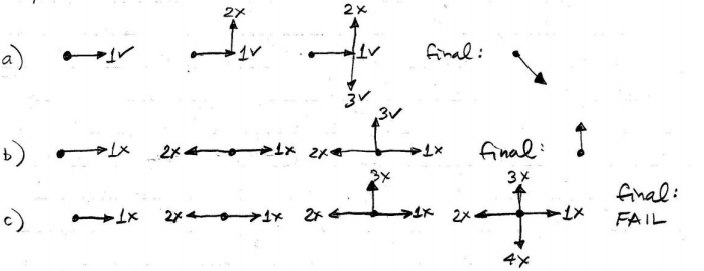


Рис.1.Алгоритм исследования

**Шаг 2. Если значение функции в шаге исследования лучше, чем в начальном то перейти в него.**

**Шаг 3. Переопределить t0 значением в наилучшей исследуемой точки и перейти в шаг 1.**

**Критерий остановки**

**Если ни одна исследуемая точка не лучше t0, то уменьшить шаг исследования.**

**Если шаг исследования достиг минимума, то t0 найденный минимум.**

**Шаг по образцу в данной интерпретации алгоритма не делается. Алгоритм требует больше шагов, но все равно находит минимум.**

Спецификация

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| class Chromosom | Работает с целочисленной хромосомой |
| class Individ | Работает с особью содержащий целочисленную хромосому |
| class Population | Работает с несколькими особями |
| class Parametrs | Содержит все переменные-параметры алгоритма |
| **Переменные** | **Назначение** |
| int LeGen | длина гена |
| float Mut | мутация |
| float Inv | инверсия |
| int point, point2 | Точки скрещивания |
| int Popul | количество особей популяции |
| Int Iter | количество итераций |
| Int Tuuuuuuuuutu | Количество особей участвующих в турнирном отборе |
| float[,] inter | поисковый интервал |
| **Функция** | **Назначение** |
| private void XD() | Функция метода Хука-Дживса |

График функции

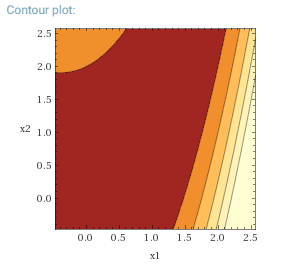
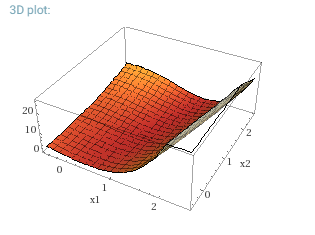


Рис.2. График 1 исследуемой функции

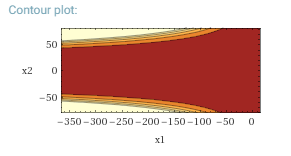
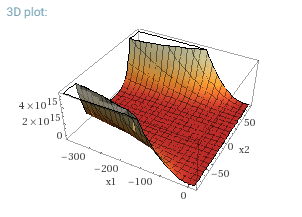


Рис.3. График 2 исследуемой функции

Результат тестирования

|  |  |
| --- | --- |
| Функция 5 | Функция 15 |
| Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-5,12;5,12) (-5,12;5,12)  Целочисленная кодировка  ------- + 0 + -------  Min 0,104391506774824  Min 1 0,148720072142224  Min 2 0,0650830369647859  Min 3 0,057949225480276  Min 4 0,00375076306540162  Min 5 0,00375076306540162  Min 6 0,00375076306540162  Min 7 0,00375076306540162  Min 8 0,00375076306540162  Min 9 0,00375076306540162  Min 10 0,00375076306540162  …  Min 49 0,00375076306540162  Min 50 0,00375076306540162  -------END-------  0,00375076306540162  1) 0,938979221991421  2) 0,876463960873484  метод Хука-Дживса  7,9507337727377E-12  0,999997221991421  0,999993960873484 | Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-5,12;5,12) (-5,12;5,12)  Целочисленная кодировка  ------- + 0 + -------  Min 0,0378511438441073  Min 1 0,0622642953746263  Min 2 0,205838408417942  Min 3 0,133716411087845  Min 4 0,122831380142969  Min 5 0,122831380142969  Min 6 0,122831380142969  Min 7 0,122831380142969  Min 8 0,122831380142969  Min 9 0,122831380142969  Min 10 0,122831380142969  ..  Min 49 0,122831380142969  Min 50 0,122831380142969  -------END-------  0,122831380142969  1) 3,51210736960573  2) 0,553885213504926  метод Хука-Дживса  6,39372286078952E-11  3,00001936960569  0,500005213504927 |

Анализ

Метод Хука-Дживса позволяет найти минимум с точностью до шага метода.

Так же метод неплохо работает с уплотнением сетки.

|  |  |
| --- | --- |
| Функция 5 | Функция 15 |
| С уплотнением сетки в 2 раза после 30 итераций  Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-10;10) (-10;10)  Целочисленная кодировка  ------- + 0 + -------  Min 1,5836296626145  Min 1 1,41620220883965  Min 2 0,762150647925684  Min 3 0,58559046615128  Min 4 0,120462376606788  Min 5 0,120462376606788  Min 6 0,26852909557194  Min 7 0,120462376606788  Min 8 0,120462376606788  Min 9 0,120462376606788  Min 10 0,120462376606788  Min 11 0,120462376606788  Min 12 0,120462376606788  Min 13 0,120462376606788  Min 14 0,120462376606788  Min 15 0,120462376606788  …  Min 26 0,120462376606788  Min 27 0,120462376606788  Min 28 0,120462376606788  Min 29 0,120462376606788  Min 30 0,120462376606788  Min 31 0,117706128403552  Min 32 0,00356577192298061  Min 33 0,00356577192298061  Min 34 0,116266266386337  Min 35 0,116266266386337  Min 36 0,116266266386337  Min 37 0,116266266386337  …  Min 49 0,116266266386337  Min 50 0,116266266386337  -------END-------  1) 1,30859442404475  2) 1,56738230987682  -------------  000000000000001100\_001001  111111111111000000\_101001  0,116266266386337  6,08923507504474E-12  1,00000242404476  1,00000530987682 | С уплотнением сетки в 2 раза после 30 итераций  Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-10;10) (-10;10)  Целочисленная кодировка  ------- + 0 + -------  Min 0,00440133828139845  Min 1 0,37924140018289  Min 2 0,00440133828139845  Min 3 0,00440133828139845  Min 4 0,00440133828139845  Min 5 0,00440133828139845  Min 6 0,00440133828139845  Min 7 0,00440133828139845  Min 8 0,00440133828139845  Min 9 0,00440133828139845  Min 10 0,00440133828139845  Min 11 0,00440133828139845  Min 12 0,00440133828139845  Min 13 0,00440133828139845  Min 14 0,00440133828139845  Min 15 0,00440133828139845  …  Min 26 0,00440133828139845  Min 27 0,00440133828139845  Min 28 0,00440133828139845  Min 29 0,00440133828139845  Min 30 0,00440133828139845  Min 31 0,00393465325350854  Min 32 0,00393465325350854  Min 33 0,00393465325350854  Min 34 0,00393465325350854  Min 35 0,00393465325350854  Min 36 0,00393465325350854  Min 37 0,00393465325350854  …  Min 49 0,00393465325350854  Min 50 0,00393465325350854  -------END-------  1) 2,85644488671094  2) 0,463867811195124  -------------  111111111111000100\_100101  000000000000111110\_100001  0,00393465325350854  6,31297416052525E-11  2,99998088671099  0,499994811195123 |
| Без уплотнения сетки  Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-10;10) (-10;10)  Целочисленная кодировка------- + 0 + -------  Min 2,44223893996103  Min 1 0,784452643647066  Min 2 0,381985255070864  Min 3 0,334010129034838  Min 4 0,0571130165785851  Min 5 0,0571130165785851  Min 6 0,0571130165785851  Min 7 0,0571130165785851  Min 8 0,0571130165785851  Min 9 0,0571130165785851  Min 10 0,0571130165785851  Min 11 0,0571130165785851  Min 12 0,0571130165785851  Min 13 0,0571130165785851  Min 14 0,0571130165785851  Min 15 0,0571130165785851  Min 16 0,0571130165785851  Min 17 0,0571130165785851  …  Min 48 0,0571130165785851  Min 49 0,0571130165785851  Min 50 0,0571130165785851  -------END-------  1) 0,78876678876679  2) -0,51037851037851  -------------  100001010\_001  111010011\_110  1,32724719812533  4,89728170803339E-12  0,999997788766789  0,999995489621489 | Без уплотнения сетки  Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (-10;10) (-10;10)  Целочисленная кодировка------- + 0 + -------  Min 5,69071611112119  Min 1 0,154381339515384  Min 2 0,154381339515384  Min 3 0,0802626096855697  Min 4 0,0802626096855697  Min 5 0,0305979318861642  Min 6 0,0802626096855697  Min 7 0,0305979318861642  Min 8 0,0305979318861642  Min 9 0,0305979318861642  Min 10 0,0305979318861642  Min 11 0,0305979318861642  Min 12 0,0305979318861642  Min 13 0,0305979318861642  Min 14 0,0305979318861642  Min 15 0,0305979318861642  Min 16 0,0305979318861642  Min 17 0,0305979318861642  …  Min 48 0,0305979318861642  Min 49 0,0305979318861642  Min 50 0,0305979318861642  -------END-------  1) 3,26007326007326  2) 0,583638583638583  -------------  110110010\_101  111011100\_001  0,0305979318861642  3,0158116926521E-11  3,00001326007323  0,500003583638585 |

Вывод

В ходе проделанной работы ознакомились с методом Хука-Дживса. Написали программу с данным методом. Протестировали его работу. И убедились е его эффективной простоте с кучей условий.

Код программы

|  |
| --- |
| private void XD()  {  //test\_Ind;  VeshIndivid t0, t1, t2, t3, t4;  t0 = new VeshIndivid(Aa);  t1 = new VeshIndivid(Aa);  t2 = new VeshIndivid(Aa);  t3 = new VeshIndivid(Aa);  t4 = new VeshIndivid(Aa);  double f, f0, alf = 0.1;  int pr = 0;  VeshPopulation F = new VeshPopulation(Aa);  // t0.xrom[0]= test\_Ind  /////для 4  t0.xrom[0].chrom = x1;  t0.xrom[1].chrom = x2;  f0 = t0.Fank\_out = F.Fank\_(t0, Aa);  MtextBox.Text = MtextBox.Text + Environment.NewLine  + f0 + Environment.NewLine;  ////  for (int i = 0; i < 1000 && pr != 1; i++)  {  t1.xrom[0].chrom = t0.Fenotip\_1(Aa) + alf;  t1.xrom[1].chrom = t0.Fenotip\_2(Aa);  t1.Fank\_out = F.Fank\_(t1, Aa);  if (Math.Abs(t1.Fank\_out) < Math.Abs(t0.Fank\_out))  {  t2.xrom[0].chrom = t0.xrom[0].chrom + alf;  t2.xrom[1].chrom = t0.xrom[1].chrom + alf;  t2.Fank\_out = F.Fank\_(t2, Aa);  if (Math.Abs(t2.Fank\_out) < Math.Abs(t1.Fank\_out))  {  t0.xrom[0].chrom = t2.xrom[0].chrom;  t0.xrom[1].chrom = t2.xrom[1].chrom;  f0 = t0.Fank\_out = t2.Fank\_out;  }  else  {  t3.xrom[0].chrom = t0.xrom[0].chrom + alf;  t3.xrom[1].chrom = t0.xrom[1].chrom - alf;  t3.Fank\_out = F.Fank\_(t3, Aa);  if (Math.Abs(t3.Fank\_out) < Math.Abs(t1.Fank\_out))  {  t0.xrom[0].chrom = t3.xrom[0].chrom;  t0.xrom[1].chrom = t3.xrom[1].chrom;  f0 = t0.Fank\_out = t3.Fank\_out;  }  else  {  t0.xrom[0].chrom = t1.xrom[0].chrom;  t0.xrom[1].chrom = t1.xrom[1].chrom;  f0 = t0.Fank\_out = t1.Fank\_out;  }  }  }  else  {  t2.xrom[0].chrom = t0.xrom[0].chrom - alf;  t2.xrom[1].chrom = t0.xrom[1].chrom;  t2.Fank\_out = F.Fank\_(t2, Aa);  if (Math.Abs(t2.Fank\_out) < Math.Abs(t0.Fank\_out))  {  t3.xrom[0].chrom = t0.xrom[0].chrom - alf;  t3.xrom[1].chrom = t0.xrom[1].chrom + alf;  t3.Fank\_out = F.Fank\_(t3, Aa);  if (Math.Abs(t3.Fank\_out) < Math.Abs(t2.Fank\_out))  {  t0.xrom[0].chrom = t3.xrom[0].chrom;  t0.xrom[1].chrom = t3.xrom[1].chrom;  f0 = t0.Fank\_out = t3.Fank\_out;  }  else  {  t4.xrom[0].chrom = t0.xrom[0].chrom - alf;  t4.xrom[1].chrom = t0.xrom[1].chrom - alf;  t4.Fank\_out = F.Fank\_(t4, Aa);  if (Math.Abs(t4.Fank\_out) < Math.Abs(t2.Fank\_out))  {  t0.xrom[0].chrom = t4.xrom[0].chrom;  t0.xrom[1].chrom = t4.xrom[1].chrom;  f0 = t0.Fank\_out = t4.Fank\_out;  }  else  {  t0.xrom[0].chrom = t2.xrom[0].chrom;  t0.xrom[1].chrom = t2.xrom[1].chrom;  f0 = t0.Fank\_out = t2.Fank\_out;  }  }  }// 2+  else //2-  {  t3.xrom[0].chrom = t0.xrom[0].chrom;  t3.xrom[1].chrom = t0.xrom[1].chrom + alf;  t3.Fank\_out = F.Fank\_(t3, Aa);  if (Math.Abs(t3.Fank\_out ) < Math.Abs(t0.Fank\_out))  {  t0.xrom[0].chrom = t3.xrom[0].chrom;  t0.xrom[1].chrom = t3.xrom[1].chrom;  f0 = t0.Fank\_out = t3.Fank\_out;  }  else  {  t4.xrom[0].chrom = t0.xrom[0].chrom;  t4.xrom[1].chrom = t0.xrom[1].chrom - alf;  t4.Fank\_out = F.Fank\_(t4, Aa);  if (Math.Abs(t4.Fank\_out) < Math.Abs(t0.Fank\_out))  {  t0.xrom[0].chrom = t4.xrom[0].chrom;  t0.xrom[1].chrom = t4.xrom[1].chrom;  f0 = t0.Fank\_out = t4.Fank\_out;  }  else pr = 1;//  }  }  //els -1  }  if (pr == 1)  { alf = alf / 10;  if (alf > 0.000001)  pr = 0;  }  }  MtextBox.Text = MtextBox.Text + Environment.NewLine  + f0 + Environment.NewLine;  MtextBox.Text = MtextBox.Text +  t0.xrom[0].Prin() + Environment.NewLine + t0.xrom[1].Prin() + Environment.NewLine;  } |