**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №1**

**по дисциплине «Оптимальное проектирование»**

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

Санкт-Петербург

2018

**Лабораторная работа №1**   
Исследование генетического алгоритма

Цель работы

Изучение генетического алгоритма в простейшем его варианте с целочисленным кодированием 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;

|  |  |  |
| --- | --- | --- |
| № | Функция *y*(***x***) | Поисковый интервал |
| (4) | 4(*x*1 – 5)2 + (*x*2 – 6)2 | *x*1∈ (0; 10); *x*2 ∈ (0; 10); |

Минимум функции в точке (5; 6).

Алгоритм генетический

Блок-схема классического ГА изображена на рис. 1.

НЕТ

Инициализация – выбор исходной популяции хромосом

Оценивание приспособленности популяции

Критерий остановки

Селекция хромосом

Применение генетических операторов

Создание новой популяции

Выбор «лучшей» хромосомы

ДА

Рис. 1

Классический ГА состоит из следующих шагов:

1. инициализация, или выбор исходной популяции хромосом;
2. оценка приспособленности хромосом в популяции;
3. проверка условия остановки алгоритма;
4. селекция хромосом;
5. применение генетических операторов;
6. формирование новой популяции;
7. выбор «наилучшей» хромосомы.

Спецификация

|  |  |  |
| --- | --- | --- |
| 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 Tutu | Количество особей участвующих в турнирном отборе | |
| float[,] inter | поисковый интервал | |
| Класс | | Назначение | |
| class Chromosom | | Работает с хромосомой | |
| class Individ | | Работает с особью содержащий хромосому | |
| class Population | | Работает с несколькими особями | |

График функции

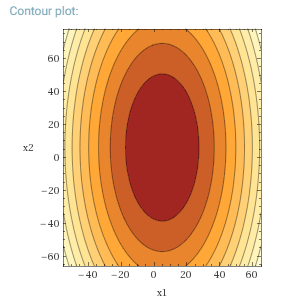
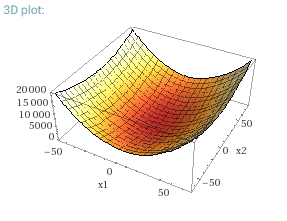


Рис.2. График исследуемой функции

Результат тестирования

|  |  |  |
| --- | --- | --- |
| ------- + 0 + -------  Min 1,09556193055421  Min 1 0,91938133400471  Min 2 0,870804833408802  Min 3 0,870804833408802  Min 4 0,849280565964665  Min 5 0,849280565964665  Min 6 0,849280565964665  Min 7 0,854605191782445  Min 8 0,854605191782445  Min 9 0,854605191782445  Min 10 0,854605191782445  Min 11 0,854605191782445  Min 12 0,854605191782445  Min 13 0,854605191782445  Min 14 0,854605191782445  Min 15 0,854605191782445  Min 16 0,854605191782445  Min 17 0,854605191782445  Min 18 0,854605191782445  Min 19 0,854605191782445  Min 20 0,854605191782445  Min 21 0,854605191782445  Min 22 0,854605191782445  Min 23 0,854605191782445  Min 24 0,854605191782445  Min 25 0,854605191782445  Min 26 0,854605191782445  Min 27 0,854605191782445  Min 28 0,854605191782445  Min 29 0,854605191782445  Min 30 0,854605191782445  Min 31 0,854605191782445  Min 32 0,854605191782445  Min 33 0,854605191782445  Min 34 0,854605191782445  Min 35 0,854605191782445  Min 36 0,854605191782445  Min 37 0,854605191782445  Min 38 0,854605191782445  Min 39 0,854605191782445  Min 40 0,854605191782445  Min 41 0,854605191782445  Min 42 0,854605191782445  Min 43 0,854605191782445  Min 44 0,854605191782445  Min 45 0,854605191782445  Min 46 0,854605191782445  Min 47 0,854605191782445  Min 48 0,854605191782445  Min 49 0,854605191782445  Min 50 0,854605191782445  -------END-------  1) 5,40278957494466  2) 5,54651634533752  -------------  100011101111100100\_1010001  001100101101111111\_0110001 | ------- + 0 + -------  Min 0,184036447718843  Min 1 0,145676496389608  Min 2 0,138088280119358  Min 3 0,129756570110643  Min 4 0,129756570110643  Min 5 0,129756570110643  Min 6 0,129756570110643  Min 7 0,129756570110643  Min 8 0,129756570110643  Min 9 0,129756570110643  Min 10 0,129756570110643  Min 11 0,129756570110643  Min 12 0,129756570110643  Min 13 0,129756570110643  Min 14 0,129756570110643  Min 15 0,129756570110643  Min 16 0,129756570110643  Min 17 0,129756570110643  Min 18 0,129756570110643  Min 19 0,129756570110643  Min 20 0,129756570110643  Min 21 0,129756570110643  Min 22 0,129756570110643  Min 23 0,129756570110643  Min 24 0,129756570110643  Min 25 0,129756570110643  Min 26 0,129756570110643  Min 27 0,129756570110643  Min 28 0,129756570110643  Min 29 0,129756570110643  Min 30 0,129756570110643  Min 31 0,129756570110643  Min 32 0,129756570110643  Min 33 0,129756570110643  Min 34 0,129756570110643  Min 35 0,129756570110643  Min 36 0,129756570110643  Min 37 0,129756570110643  Min 38 0,129756570110643  Min 39 0,129756570110643  Min 40 0,129756570110643  Min 41 0,129756570110643  Min 42 0,129756570110643  Min 43 0,129756570110643  Min 44 0,129756570110643  Min 45 0,129756570110643  Min 46 0,129756570110643  Min 47 0,129756570110643  Min 48 0,129756570110643  Min 49 0,129756570110643  Min 50 0,129756570110643  -------END-------  1) 4,83440592391509  2) 5,85832792098307  -------------  011000011010000111\_1011110  011000100100111111\_0101001 | ------- + 0 + -------  Min 0,409836029220477  Min 1 0,413745390953046  Min 2 0,413745390953046  Min 3 0,400911615693697  Min 4 0,400911615693697  Min 5 0,400911615693697  Min 6 0,400911615693697  Min 7 0,400911615693697  Min 8 0,400911615693697  Min 9 0,400911615693697  Min 10 0,400911615693697  Min 11 0,400911615693697  Min 12 0,400911615693697  Min 13 0,400911615693697  Min 14 0,400911615693697  Min 15 0,400911615693697  Min 16 0,400911615693697  Min 17 0,400911615693697  Min 18 0,400911615693697  Min 19 0,400911615693697  Min 20 0,400911615693697  Min 21 0,400911615693697  Min 22 0,400911615693697  Min 23 0,400911615693697  Min 24 0,400911615693697  Min 25 0,400911615693697  Min 26 0,400911615693697  Min 27 0,400911615693697  Min 28 0,400911615693697  Min 29 0,400911615693697  Min 30 0,400911615693697  Min 31 0,400911615693697  Min 32 0,400911615693697  Min 33 0,400911615693697  Min 34 0,400911615693697  Min 35 0,400911615693697  Min 36 0,400911615693697  Min 37 0,400911615693697  Min 38 0,400911615693697  Min 39 0,400911615693697  Min 40 0,400911615693697  Min 41 0,400911615693697  Min 42 0,400911615693697  Min 43 0,400911615693697  Min 44 0,400911615693697  Min 45 0,400911615693697  Min 46 0,400911615693697  Min 47 0,400911615693697  Min 48 0,400911615693697  Min 49 0,400911615693697  Min 50 0,400911615693697  -------END-------  1) 4,82833191240823  2) 5,46799258792378  -------------  100101111010110011\_1011110  010010000110111111\_1010001 |

Минимум функции в точке (5; 6).

Тестирование проводилось при параметрах:

|  |  |  |  |
| --- | --- | --- | --- |
| Турнир | 5 | Мутация | 0 |
| Популяция | 70 | Инверсия | 0,07 |
| Итерации | 50 | Точка скрещивания | 0,75 |
| Длина гена | 25 | x1 (0;10) x2 (0;10) |  |

Анализ

Программа к 10 популяции находит минимум, который сохраняется к последней итерации. Однако сама популяция становиться однородной только к 15 популяции и после не претерпевает изменений, так как инверсия с вероятностью 0.07 мало валяет на улучшение минимума. Так же турнир в 5 особей ускоряет стабилизацию алгоритма.

Если проводит турнир среди 2 особей, а не 5, и длиной гена 12, а не 24, то минимум стабилизируется к 20 показано. Что показано в примере.

|  |  |
| --- | --- |
| Ген 12 Мутация 0,00 Инверсия 0,07  Интервал (0;10) (0;10)  Целочисленная кодировка ------- + 0 + -------  Min 0,695144768404841  Min 1 0,132488322964513  Min 2 0,308217084773861  Min 3 0,102051252234402  Min 4 0,308217084773861  Min 5 0,0241516725033207  Min 6 0,308217084773861  Min 7 0,308217084773861  Min 8 0,162305202598243  Min 9 0,269556518640768  Min 10 0,0160414812429463  Min 11 0,0505813546106586  Min 12 0,0160414812429463  Min 13 0,162305202598243  Min 14 0,0160414812429463  Min 15 0,0160414812429463  Min 16 0,0160414812429463  Min 17 0,0160414812429463  Min 18 0,0160414812429463  Min 19 0,0160414812429463  Min 20 0,0160414812429463  … | Min 80 0,0160414812429463  Min 81 0,0160414812429463  Min 82 0,0160414812429463  Min 83 0,0160414812429463  Min 84 0,0160414812429463  Min 85 0,0160414812429463  Min 86 0,0160414812429463  Min 87 0,0160414812429463  Min 88 0,0160414812429463  Min 89 0,0160414812429463  Min 90 0,0160414812429463  Min 91 0,0160414812429463  Min 92 0,0160414812429463  Min 93 0,0160414812429463  Min 94 0,0160414812429463  Min 95 0,0160414812429463  Min 96 0,0160414812429463  Min 97 0,0160414812429463  Min 98 0,0160414812429463  Min 99 0,0160414812429463  Min 100 0,0160414812429463  -------END-------  1) 5,05982905982906  2) 5,95848595848596 |

Минимум функции в точке (5; 6).

Вывод

В ходе проделанной работы ознакомились с генетическим алгоритмом. Изучили генетический алгоритм в простейшем его варианте с целочисленным кодированием хромосом и протестировали его работу.

Код программы

|  |  |
| --- | --- |
| class Chromosom  { private static int length = 15;  private bool[] chrom;  private static int \_seed = Environment.TickCount;  private static Random rnd1 = new Random(\_seed);  //возможность измененния длины гена  public static int Length  { get  { return length; }  set  { length = value; }  }  //возможность обращаться к к конкретному значению в хромосоме (переопределение)  public bool this[int i]  { get { return chrom[i]; }  set { chrom[i] = value; }  }  public Chromosom()  { int L = Chromosom.Length;  chrom = new bool[L];  Randomize(); }  public Chromosom(int L)  { Chromosom.Length = L;  // int L = BinaryChromosome.Length;  chrom = new bool[L];  Randomize();  }  public string Prin(int point, int point2)  { string tectik=" ";  char a;  for (int i = 0; i < Chromosom.Length; i++)  { if (i == point || i == point2 && i!=0)  tectik = tectik + '\_';  a = (chrom[i] ? '1' : '0');  tectik = tectik + a ; }  return tectik;  }  //рандом работает классно  public void Randomize()  { for (int i = 0; i < Chromosom.Length; i++)  { if (rnd1.NextDouble() > 0.5)  chrom[i]=true; }  }  // мутация работает  public void Mutate(double p)  { for (int i = 0; i < Chromosom.Length; i++)  { if (rnd1.NextDouble() < p)  chrom[i] = !chrom[i]; }  }  // инвенсия не работает  public void Invers(double p)  {for (int i = 0; i < Chromosom.Length; i++)  {  if (rnd1.NextDouble() < p)  { chrom[i] = !chrom[i];  break; }  } } | // однототечный кроссенговер  public static void PointCrossOver(int point, Chromosom chromA, Chromosom chromB, out Chromosom chromC, out Chromosom chromD)  { chromC = new Chromosom();  chromD = new Chromosom();  for (int i = 0; i < point; i++)  { chromC[i] = chromA[i];  chromD[i] = chromB[i]; }  for (int i = point; i < Chromosom.Length; i++)  {  chromC[i] = chromB[i];  chromD[i] = chromA[i];  }  }  // двухточеный кроссенговер  public static void PointCrossOver(int point, int point2, Chromosom chromA, Chromosom chromB, out Chromosom chromC, out Chromosom chromD)  {  chromC = new Chromosom();  chromD = new Chromosom();  int i = 0;  for (i = 0; i < point; i++)  {  chromC[i] = chromA[i];  chromD[i] = chromB[i];  }  for ( i = point; i < point2; i++)  {  chromC[i] = chromB[i];  chromD[i] = chromA[i];  }  for ( i = point2; i < Chromosom.Length; i++)  {  chromC[i] = chromA[i];  chromD[i] = chromB[i];  }  }  } |
| class Individ  { private Chromosom xrom\_1= new Chromosom(), xrom\_2 = new Chromosom();  // переделать в вектор?  /\*интервал\*/  public double Min = -10;  public double Max = 10;  public double Fank\_out = 0;  float Mut\_ind=0, Inv\_ind=0;  int point\_ind=1, point2\_ind=3, L\_ind=25;  public Individ ( int L, float Mut, float Inv, int point, int point2, float[,] inter0) {  xrom\_1 = new Chromosom(L);  xrom\_2 = new Chromosom(L);  Min = inter0[0,0];  Max = inter0[0,1];  Set\_chrom(L, Mut, Inv, point, point2);  }  public Individ(float[,] inter0)  { xrom\_1 = new Chromosom();  xrom\_2 = new Chromosom();  Min = inter0[0, 0];  Max = inter0[0, 1];  Set\_chrom(L\_ind, Mut\_ind, Inv\_ind, point\_ind, point2\_ind); }  public string Prin\_Ind(int point, int point2)  { return "1)" + xrom\_1.Prin( point, point2) + Environment.NewLine + "2)" + xrom\_2.Prin( point, point2) + Environment.NewLine;  }  private void Set\_chrom(int L, float Mut, float Inv, int point, int point2)  {  Mut\_ind = Mut;  Inv\_ind = Inv;  point\_ind = point;  point2\_ind = point2;  L\_ind = L;  }  public void Mutate(double p, int i)// 1 - 1 хром 2- 2 хром 3> - обе  { switch (i)  { case 1:  xrom\_1.Mutate(p);  break;  case 2:  xrom\_2.Mutate(p);  break;  default:  xrom\_1.Mutate(p);  xrom\_2.Mutate(p);  break;  }  } | public void Invers(double p, int i)// 1 - 1 хром 2- 2 хром 3> - обе  {  switch (i)  {  case 1:  xrom\_1.Invers(p);  break;  case 2:  xrom\_2.Invers(p);  break;  default:  xrom\_1.Invers(p);  xrom\_2.Invers(p);  break;  }  }  public static Individ[] PointCrossOver(int L, float Mut, float Inv, int point, int point2, Individ A, Individ B, float[,] inter)  {  Individ[] C = { new Individ(L, Mut, Inv, point, point2, inter), new Individ(L, Mut, Inv, point, point2, inter) };  if (point2 == 0)  {  Chromosom.PointCrossOver(point, A.xrom\_1, B.xrom\_1, out C[0].xrom\_1, out C[1].xrom\_1);  Chromosom.PointCrossOver(point, A.xrom\_2, B.xrom\_2, out C[0].xrom\_2, out C[1].xrom\_2);  }  else  {  Chromosom.PointCrossOver(point, point2, A.xrom\_1, B.xrom\_1, out C[0].xrom\_1, out C[1].xrom\_1);  Chromosom.PointCrossOver(point, point2, A.xrom\_2, B.xrom\_2, out C[0].xrom\_2, out C[1].xrom\_2);  }  return C;  }  // сделать фенотип ???  public double Fenotip\_1  { get  { return Genot\_in\_Fenotip(xrom\_1); } }  public double Fenotip\_2  { get { return Genot\_in\_Fenotip(xrom\_2); }}    double Genot\_in\_Fenotip(Chromosom ch)  { Int32 gen = 0;  for (int i = 0; i < Chromosom.Length; i++)  { if (ch[i])  gen += 1 << i;  }  return Min + gen\*(Max - Min) / (Math.Pow(2.0, Chromosom.Length) - 1);  }  public string Prin\_Fen()  { return "1) " + Fenotip\_1 + " 2) " + Fenotip\_2 + Environment.NewLine;  }  } |
| class Population  {  private int Namber = 10, nIters = 50, fanc = 4, Tuutu = 2; // популяция, итерации  // private double pMut, pInv; // мутация 0- нет /1 всегда есть //инверсия 0- нет /1 всегда есть  float[,] inter;  public Individ[] Popul\_all;  private static int \_seed = Environment.TickCount;  private static Random rnd2 = new Random(\_seed);  public Population(int L, float Mut, float Inv, int point, int point2, int Namber\_0, int nIters\_0, int fanc\_0, float[,] inter, int tuuuutu)  {Set\_pop(Namber\_0, nIters\_0, fanc\_0, tuuuutu);  Popul\_all = Popul\_0(L, Mut, Inv, point, point2, inter); }  private void Set\_pop(int Namber0, int nIters0, int fanc0, int tuuuutu)  {  Namber = Namber0;  nIters = nIters0;  fanc = fanc0;  Tuutu = tuuuutu; }  public double Fank\_(double Fenotip\_01, double Fenotip\_02)  { double x1 = Fenotip\_01, x2 = Fenotip\_02;  return 4 \* (x1 - 5) \* (x1 - 5) + (x2 \* x2 - 6) \* (x2 \* x2 - 6);  }  Individ[] Popul\_0(int L, float Mut, float Inv, int point, int point2, float[,] inter) // новая популяция (прям новая первая)  {Individ[] Ind\_popul = new Individ[Namber]; //(L, Mut, Inv, point, point2, inter);  for (int i = 0; i < Namber; i++)  {Ind\_popul[i] = new Individ(L, Mut, Inv, point, point2, inter);  Ind\_popul[i].Fank\_out = Fank\_(Ind\_popul[i].Fenotip\_1, Ind\_popul[i].Fenotip\_2);}  return Ind\_popul; }  public string Print(int point, int point2)  { string a = "";  for (int i = 0; i < Namber; i++)  { a = a  Popul\_all[i].Fank\_out + Environment.NewLine; }  return a; }  public void Select(int L, float Mut, float Inv, int point, int point2, float[,] inter)  {  Individ[] Ind\_popul = new Individ[Namber];  Individ f, g; | Individ[] C = { new Individ(L, Mut, Inv, point, point2, inter), new Individ(L, Mut, Inv, point, point2, inter) };  for (int i = 0; 2 \* i + 1 <= Namber; i++)  { f = Turnir(Rand());  g = Turnir(Rand());  C = Individ.PointCrossOver(L, Mut, Inv, point, point2, f, g, inter);  Ind\_popul[2 \* i] = C[0];  Ind\_popul[2 \* i + 1] = C[1];  Ind\_popul[2 \* i].Fank\_out = Fank\_(Ind\_popul[i].Fenotip\_1, Ind\_popul[i].Fenotip\_2);  Ind\_popul[2 \* i + 1].Fank\_out = Fank\_(Ind\_popul[i + 1].Fenotip\_1, Ind\_popul[i + 1].Fenotip\_2); }  Popul\_all = Ind\_popul; Mut\_all(Mut, Inv,3); }  public Individ Turnir(Individ[] Tur)  { if (Tur.Length > 2)  { int i;  Individ[] Tur2 = new Individ[Tur.Length - 1];  for (i = 0; i < Tur2.Length - 1; i++)  Tur2[i] = Tur[i];  if (Tur[Tur.Length - 1].Fank\_out > Tur[Tur.Length - 2].Fank\_out)  Tur2[i] = Tur[Tur.Length - 2];  else Tur2[i] = Tur[Tur.Length - 1];  return Turnir(Tur2);  }  else {  if (Tur[1].Fank\_out > Tur[0].Fank\_out)  return Tur[0];  else return Tur[1];  }  }  private Individ[] Rand()  {  Individ[] D = new Individ[Tuutu];  for (int i = 0; i <= Tuutu-1; i++)  {  D[i] = Popul\_all[rnd2.Next(0, Popul\_all.Length-1)];  }  return D;  }  private void Mut\_all(double mut0, double inv0, int i0)  {  for (int i = 0; i <= Popul\_all.Length-1; i++)  {  Popul\_all[i].Mutate(mut0, i0);  Popul\_all[i].Invers(inv0, i0);  }  }  } |
| Population test = new Population(LeGen, Mut, Inv, point, point2, Popul, Iter, fanc, inter, Tuuuuuuuuutu);  // создание популяции  Individ test\_Ind = test.Turnir(test.Popul\_all);  MtextBox.Text = MtextBox.Text + "------- + 0 + -------" + Environment.NewLine;  // MtextBox.Text = MtextBox.Text + test.Print( point, point2) + Environment.NewLine;  MtextBox.Text = MtextBox.Text + test\_Ind.Fank\_out + Environment.NewLine;  MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen() + Environment.NewLine;  for (int i = 0; i < Iter; i++)  {  test.Select(LeGen, Mut, Inv, point, point2, inter);  test\_Ind = test.Turnir(test.Popul\_all);  MtextBox.Text = MtextBox.Text + "-------" + i +"-------"+ Environment.NewLine+  // test.Print(point, point2)+ Environment.NewLine + "-----" + Environment.NewLine +  test\_Ind.Fank\_out + Environment.NewLine;  MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen() + Environment.NewLine;  }  test\_Ind = test.Turnir(test.Popul\_all);  MtextBox.Text = MtextBox.Text + "-------END-------" +Environment.NewLine +  test.Print(point, point2) + Environment.NewLine + "-----" +  Environment.NewLine + test\_Ind.Fank\_out + Environment.NewLine;  MtextBox.Text = MtextBox.Text + test\_Ind.Prin\_Fen()+Environment.NewLine; | |