|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САПР** | | Тема | оцінка | підпис |
| СПКс-11 | 4 | Оптимізація цільової функції за допомогою генетичного алгоритму |  |  |
| Тураш Ю.Ю. | |
| № залікової: 1508500 | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р.З. | |

**Мета роботи**: реалізувати генетичний алгоритм пошуку максимального і мінімального значення цільової функції згідно варіанту.

**Завдання:** реалізувати генетичний алгоритм пошуку максимального і мінімального значення цільової функції f(x) = a + bx + cx2 + dx3 на інтервалі

x = [-10, 53].

**Варіант 13**

Розробити програму, яка реалізовує генетичний алгоритм пошуку максимального і мінімального значення цільової функції:

*f(x) = 62- x - 86x2 +2x3* в інтервалі *x = [-10, 53]*.





**Максимальне і мінімальне значення цільової функції**

Графік функції згідно варіанту показаний на рис.1.

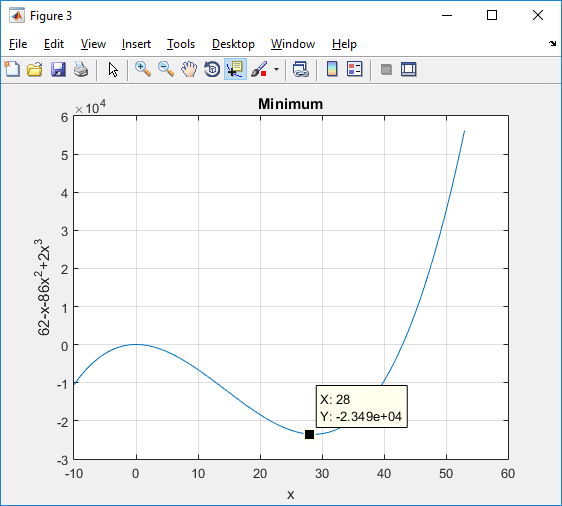


Рис.1. Локальний мінімум на інтервалі [-10;53]

Пошук мінімуму та максимуму функції відбувався інтервалі [-10;53]. Для пошуку максимуму та мінімуму використовувалась функція fminbnd().

Мінімум: -2.349e+04, при x = 28.

Максимум: 62.0029, при х = 0

Графік функції на інтервалі [-10;10] показаний на рис.2.

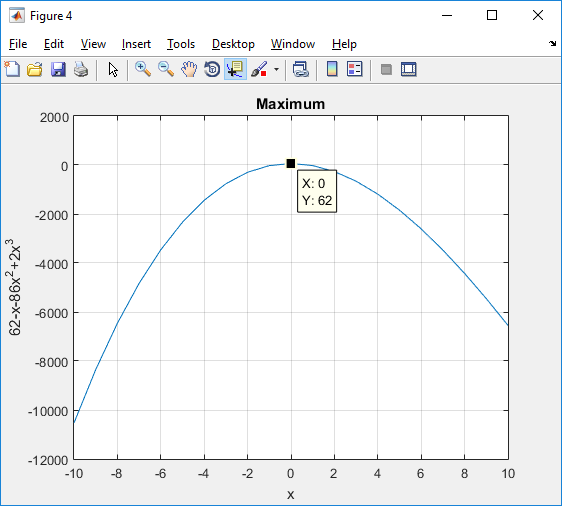


Рис.2. Локальний максимум на інтервалі[-10;10]

**Результати виконання програми**

Для виконання індивідуального завдання була вибрано використовувати турнірний відбір, одно точковий кросинговер і точкова мутація.

Результати:

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість поколінь, розмір популяції | 10 | 100 | 300 |
| fmin | 23275,14766 | 23515,19013 | 23524,29878 |
| fmax | 46970,1011 | 54192,20574 | 55944,22962 |

Похибка при знаходженні мінімуму і максимуму залежать від кількості поколінь і розміру популяції, при їхньому збільшенні похибки зменшуються.

Результат виконання програми для десяти поколінь з розміром популяції 10 для знаходження мінімуму функції:

Початкова популяцiя

0 [24,15874] Fitness = 21955,36241

1 [14,8959] Fitness = 12424,81776

2 [33,44668] Fitness = 21345,6692

3 [52,55577] Fitness = -52797,56822

4 [3,81221] Fitness = 1080,83876

Нова популяцiя

0 [33,44668] Fitness = 21345,6692

1 [24,15874] Fitness = 21955,36241

2 [33,44668] Fitness = 21345,6692

3 [2,41151] Fitness = 412,48588

4 [24,15874] Fitness = 21955,36241

Найкраще значення поколiння 21955,3624062268

Нова популяцiя

0 [24,15874] Fitness = 21955,36241

1 [24,15874] Fitness = 21955,36241

2 [24,15874] Fitness = 21955,36241

3 [31,77924] Fitness = 22633,9216

4 [24,15874] Fitness = 21955,36241

Найкраще значення поколiння 22633,9216021907

Нова популяцiя

0 [24,15874] Fitness = 21955,36241

1 [31,77924] Fitness = 22633,9216

2 [24,15874] Fitness = 21955,36241

3 [12,79229] Fitness = 9837,3312

4 [31,77924] Fitness = 22633,9216

Найкраще значення поколiння 22633,9216021907

Нова популяцiя

0 [31,77924] Fitness = 22633,9216

1 [31,77924] Fitness = 22633,9216

2 [31,77924] Fitness = 22633,9216

3 [17,04606] Fitness = 15037,82543

4 [31,77924] Fitness = 22633,9216

Найкраще значення поколiння 22633,9216021907

Нова популяцiя

0 [31,77924] Fitness = 22633,9216

1 [31,77924] Fitness = 22633,9216

2 [31,77924] Fitness = 22633,9216

3 [52,46961] Fitness = -52150,375

4 [31,77924] Fitness = 22633,9216

Найкраще значення поколiння 22633,9216021907

Нова популяцiя

0 [31,77924] Fitness = 22633,9216

1 [31,77924] Fitness = 22633,9216

2 [31,77924] Fitness = 22633,9216

3 [30,34213] Fitness = 23275,14766

4 [31,77924] Fitness = 22633,9216

Найкраще значення поколiння 23275,1476570039

Нова популяцiя

0 [30,34213] Fitness = 23275,14766

1 [30,34213] Fitness = 23275,14766

2 [31,77924] Fitness = 22633,9216

3 [-2,4354] Fitness = 474,53659

4 [30,34213] Fitness = 23275,14766

Найкраще значення поколiння 23275,1476570039

Нова популяцiя

0 [30,34213] Fitness = 23275,14766

1 [30,34213] Fitness = 23275,14766

2 [30,34213] Fitness = 23275,14766

3 [45,3503] Fitness = -9684,15002

4 [30,34213] Fitness = 23275,14766

Найкраще значення поколiння 23275,1476570039

Нова популяцiя

0 [30,34213] Fitness = 23275,14766

1 [30,34213] Fitness = 23275,14766

2 [30,34213] Fitness = 23275,14766

3 [52,64912] Fitness = -53502,75065

4 [30,34213] Fitness = 23275,14766

Найкраще значення поколiння 23275,1476570039

Результат:

[30,34213] Fitness = 23275,14766

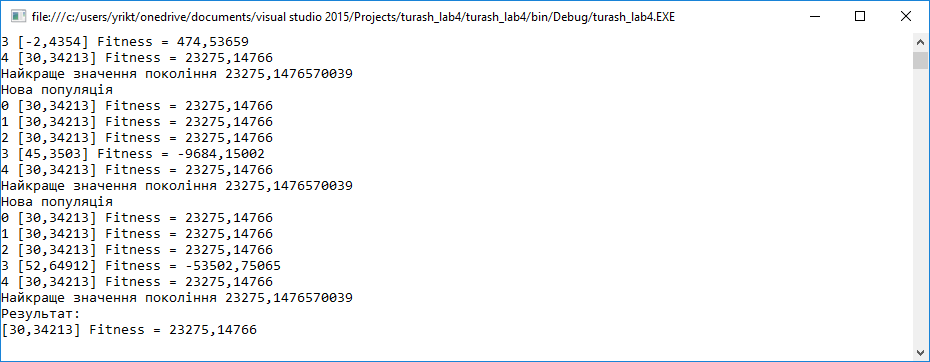


Рис.3. Результати виконання програми для знаходження мінімуму

Результат виконання програми для десяти поколінь з розміром популяції 10 для знаходження максимуму функції:

Початкова популяцiя

0 [14,92322] Fitness = -12458,45915

1 [26,96576] Fitness = -23283,63103

2 [-1,94] Fitness = -274,33086

3 [7,52141] Fitness = -3959,68759

4 [0,83242] Fitness = 2,73036

Нова популяцiя

0 [0,83242] Fitness = 2,73036

1 [0,83242] Fitness = 2,73036

2 [0,83242] Fitness = 2,73036

3 [51,76308] Fitness = 46970,1011

4 [-1,94] Fitness = -274,33086

Найкраще значення поколiння 46970,1010969288

Нова популяцiя

0 [51,76308] Fitness = 46970,1011

1 [51,76308] Fitness = 46970,1011

2 [51,76308] Fitness = 46970,1011

3 [35,38381] Fitness = -19044,54205

4 [51,76308] Fitness = 46970,1011

Найкраще значення поколiння 46970,1010969288

Нова популяцiя

0 [51,76308] Fitness = 46970,1011

1 [51,76308] Fitness = 46970,1011

2 [51,76308] Fitness = 46970,1011

3 [43,33512] Fitness = 1277,3426

4 [51,76308] Fitness = 46970,1011

Найкраще значення поколiння 46970,1010969288

Нова популяцiя

0 [51,76308] Fitness = 46970,1011

1 [51,76308] Fitness = 46970,1011

2 [51,76308] Fitness = 46970,1011

3 [12,46059] Fitness = -9433,94655

4 [51,76308] Fitness = 46970,1011

Найкраще значення поколiння 46970,1010969288

Нова популяцiя

0 [51,76308] Fitness = 46970,1011

1 [51,76308] Fitness = 46970,1011

2 [51,76308] Fitness = 46970,1011

3 [40,9605] Fitness = -6822,56113

4 [51,76308] Fitness = 46970,1011

Найкраще значення поколiння 46970,1010969288

Нова популяцiя

0 [51,76308] Fitness = 46970,1011

1 [51,76308] Fitness = 46970,1011

2 [51,76308] Fitness = 46970,1011

3 [21,3233] Fitness = -19671,38306

4 [51,76308] Fitness = 46970,1011

Найкраще значення поколiння 46970,1010969288

Нова популяцiя

0 [51,76308] Fitness = 46970,1011

1 [51,76308] Fitness = 46970,1011

2 [51,76308] Fitness = 46970,1011

3 [-4,40639] Fitness = -1774,50708

4 [51,76308] Fitness = 46970,1011

Найкраще значення поколiння 46970,1010969288

Нова популяцiя

0 [51,76308] Fitness = 46970,1011

1 [51,76308] Fitness = 46970,1011

2 [51,76308] Fitness = 46970,1011

3 [1,90777] Fitness = -239,02515

4 [51,76308] Fitness = 46970,1011

Найкраще значення поколiння 46970,1010969288

Нова популяцiя

0 [51,76308] Fitness = 46970,1011

1 [51,76308] Fitness = 46970,1011

2 [51,76308] Fitness = 46970,1011

3 [3,46055] Fitness = -888,4645

4 [51,76308] Fitness = 46970,1011

Найкраще значення поколiння 46970,1010969288

Результат:

[51,76308] Fitness = 46970,1011

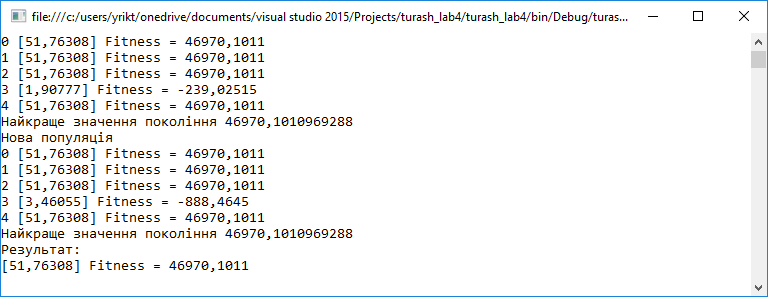


Рис.4. Результати виконання програми для знаходження максимуму

Код програми на мові програмування C#:

class GAEvolve

{

public double[] BestFitness;//є найкращим значенням для і-го покоління даної еволюції

public GAIndividual BestInd; //найкращий індивідум(х) даної еволюції

public GAEvolve(int generations, int popSize, int genomeSize, int xrate, int mrate, double[] minRange, double[] maxRange)

{

BestFitness = new double[generations];

var gap = new GAPopulation(popSize, genomeSize, minRange, maxRange);

BestFitness[0] = gap.Ind[gap.BestIndex].Fitness;

for (var i = 1; i < generations; i++)

{

gap = gap.Generate(gap, xrate, mrate, minRange, maxRange);

BestFitness[i] = gap.Ind[gap.BestIndex].Fitness;

Console.WriteLine("Найкраще значення поколiння " + BestFitness[i]);

}

BestInd = gap.Ind[gap.BestIndex];

}

static void Main(string[] args)

{

double[] min = { -10f };

double[] max = { 53 };

var gae = new GAEvolve(10, 5, 1, 70, 20, min, max);

Console.WriteLine("Результат:");

Console.WriteLine(gae.BestInd);

Console.ReadKey();

}

}

class GAIndividual

{

private static Random randg = new Random(); // Рандомний генератор

public int GenomeSize;

public double[] Genome;

public double Fitness;

public GAIndividual(int gsize, double[] minRange, double[] maxRange)

{

// створити випадкового індивідума довжиною gsize

// і-й ген повине бути в діапазоні між minRange[i] і maxRange[i]

GenomeSize = gsize;

Genome = new double[GenomeSize];

for (var i = 0; i < GenomeSize; i++)

{

Genome[i] = (double)randg.NextDouble() \* (maxRange[i] - minRange[i]) + minRange[i];

}

EvalFitness();// оцінити придатність цього нового індивідума

}

public GAIndividual(double[] d)

{

// Створити індивідума, що його ген є таки же як d []

GenomeSize = d.Length;

Genome = new double[GenomeSize];

for (var i = 0; i < GenomeSize; i++)

{

Genome[i] = d[i];

}

EvalFitness();// оцінити придатність цієї нового індивідума

}

public GAIndividual Mutate(double[] minRange, double[] maxRange)

{

// rate це шанс кожного гена мутувати

var rate = 1.0f / (double)GenomeSize;

var result = new double[GenomeSize];

for (var i = 0; i < GenomeSize; i++)

result[i] = Genome[i];

// застосування точкової мутації

for (var i = 0; i < GenomeSize; i++)

result[i] = randg.NextDouble() \* (maxRange[i] - minRange[i]) + minRange[i];

return new GAIndividual(result);

}

public static GAIndividual Xover1P(GAIndividual f, GAIndividual m)

{

// одноточковий кросинговер

var rng = new Random();

int xpoint = 1 + rng.Next(1);

double[] child = new double[f.GenomeSize];

for (var i = 0; i < xpoint; i++)

{

child[i] = f.Genome[i];

}

for (int i = xpoint; i < f.GenomeSize; i++)

{

child[i] = m.Genome[i];

}

return new GAIndividual(child);

}

public override string ToString()

{

var s = "[";

s += Math.Round(Genome[0], 5) + "]";

s += " Fitness = " + Math.Round(Fitness, 5);

return s;

}

private void EvalFitness() => Fitness = (62 - Genome[0] - 86 \* Genome[0] \* Genome[0] + 2 \* Genome[0] \* Genome[0] \* Genome[0]);

}

class GAPopulation

{

private static readonly Random randg = new Random(); // Рандомний генератор

public int PopSize;

public GAIndividual[] Ind;

public int BestIndex; // індекс найкращого індивідума в масиві Ind

public GAPopulation(int psize, int gsize, double[] minRange,

double[] maxRange)

{

// створення рандомної популяції кількісью PopSize

PopSize = psize;

Ind = new GAIndividual[PopSize];

Console.WriteLine("Початкова популяцiя");

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

{

Ind[i] = new GAIndividual(gsize, minRange, maxRange);

Console.WriteLine(i + " " + Ind[i] + " ");

}

Evaluate();

}

public GAPopulation(GAIndividual[] p)

{

// Створення популяції з такимиж індивідумами як в p

PopSize = p.Length;

Ind = new GAIndividual[PopSize];

Console.WriteLine("Нова популяцiя");

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

{

Ind[i] = p[i];

Console.WriteLine(i + " " + Ind[i] + " ");

}

Evaluate();

}

public GAPopulation Generate(GAPopulation p, int xrate, int mrate,

double[] minRange, double[] maxRange)

{

//Створення нової популяції з р, xrate відсотків індивідумів нового населення є

//схрещування, mrate відсотків з них створюються в результаті мутації, а інші по відтворення.

if (xrate < 0 || xrate > 100 || mrate < 0 || mrate > 100

|| xrate + mrate > 100)

Console.WriteLine("error: xrate i/чи mrate неправильно встановленi");

var newg = new GAIndividual[p.PopSize];

int newg\_index = 0;

int xn = xrate\*p.PopSize/100;

//xn: Кількість нащадків, які будуть схрешення

var mn = mrate\*p.PopSize/100;

// mn: кількість нащадків які будуть створенні мутацією

// схрещування:

for (var i = 0; i < xn; i++)

{

// select to parents for cross-over:

var p1 = p.tr\_select();

var p2 = p.tr\_select();

newg[newg\_index++] = GAIndividual.Xover1P(p.Ind[p1], p.Ind[p2]);

}

// мутація:

for (var i = 0; i < mn; i++)

newg[newg\_index++] = p.Ind[p.tr\_select()].Mutate(minRange,

maxRange);

// відтворення:

for (var i = newg\_index; i < p.PopSize; i++)

newg[i] = p.Ind[p.tr\_select()];

return new GAPopulation(newg);

}

public int tr\_select()

{

//турнірна вибірка розміром PopSize/10

//вона повертає індекс вибраного особи в Ind []

var sIndex = randg.Next(PopSize);

// індекс вибраного індивідума

var sFitness = Ind[sIndex].Fitness;

var trSize = Math.Min(10, PopSize);

for (var i = 1; i < trSize; i++)

{

var tmp = randg.Next(PopSize);

if (Ind[tmp].Fitness > sFitness)

{

sIndex = tmp;

sFitness = Ind[tmp].Fitness;

}

}

return sIndex;

}

private void Evaluate()

{

//оцінювання

var best = 0;

// індекс найкрощого індивідума

var bestFitness = Ind[0].Fitness;

var sum = Ind[0].Fitness;

// сума придатності особин даної популяції

for (int i = 1; i < PopSize; i++)

{

sum += Ind[i].Fitness;

if (Ind[i].Fitness > bestFitness)

{

best = i;

bestFitness = Ind[i].Fitness;

}

}

BestIndex = best;

}

public string toString() => "best individual = " + Ind[BestIndex];

}

Код програми на мові програмування Matlab для знаходження екстремумів функції:

clc

a=62;

b=-1;

c=-86;

d=2;

x=-10:1:53;

y=a+b\*x+c\*x.^2+d\*x.^3;

figure

plot(x,y)

title('Minimum')

xlabel('x')

ylabel('62-x-86x^2+2x^3')

grid on

minX=fminbnd(@(x) a+b\*x+c\*x^2+d\*x^3,0,53);

minY=a+b\*minX+c\*minX^2+d\*minX^3;

x=-10:1:10;

y=a+b\*x+c\*x.^2+d\*x.^3;

figure

plot(x,y)

title('Maximum')

xlabel('x')

ylabel('62-x-86x^2+2x^3')

grid on

maxX=fminbnd(@(x) -(a+b\*x+c\*x^2+d\*x^3),-10,10);

maxY=a+b\*maxX+c\*maxX^2+d\*maxX^3;

**Висновок.** Під час виконання даної роботи здобув навички використання генетичних алгоритмів для знаходження екстремумів функції.