|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКм-12 | 4 | **Генетичні алгоритми пошуку пошуку** |  |  |
| Купець І.В. | |
| № залікової: | |
| 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].

В - 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N варіанту | a | b | c | d |
| 6 | 50 | -63 | -25 | 1 |

**Максимальне і мінімальне значення цільової функції**

Графік функції згідно варіанту показаний на рис.1.

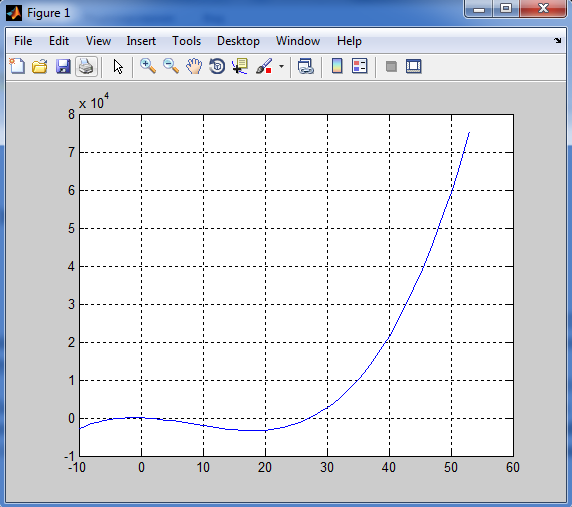


Рис.1. Графік функції на інтервалі [-10;53]

Для знаходження мінімуму функції було вибрано інтервал [-10;53]. Пошук мінімуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() і рівний: -3.3527e+03.

Для знаходження максимуму функції було вибрано інтервал [-10;10]. Пошук максимуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() , змінивши функцію на протилежну і рівний: 87.8872. Графік функції на інтервалі [-10;10] показаний на рис.2.

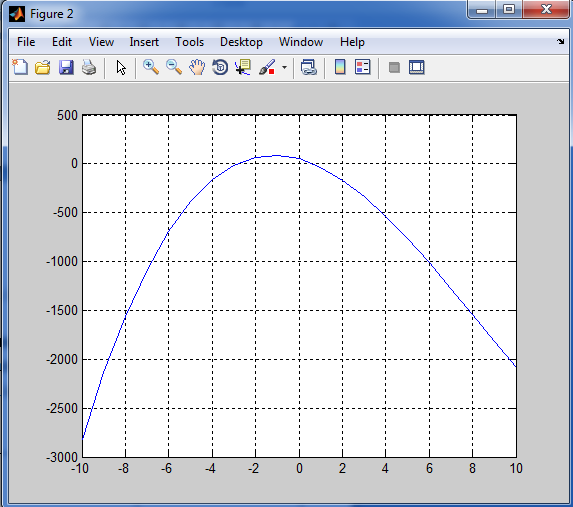


Рис.2. Графік функції на інтервалі[-10;10]

**Результати виконання програми**

Для виконання індивідуального завдання була вибрано використовувати турнірний відбір, рівномірне схрещування і класична мутація обміну.

Результати:

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість поколінь, розмір популяції | 10 | 100 | 300 |
| fmin | -33519.73 | -33526.73 | -33526.83 |
| fmax | 87.79712 | 87.88631 | 87.88721 |

Отже похибка при знаходженні мінімуму і максимуму залежать від кількості поколінь і розміру популяції, при їхньому збільшенні похибки прямують до 0.

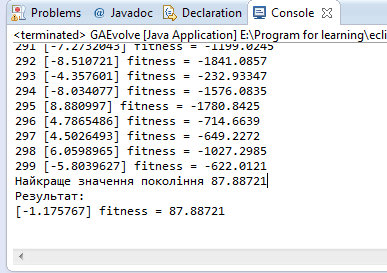


Рис.3. Результати виконання програми

Результат виконання програми для десяти поколінь з розміром популяції 30 для знаходження максимуму функції:

Початкова популяція

0 [9.643394] fitness = -1985.6223

1 [0.121644974] fitness = 41.968227

2 [-1.0329428] fitness = 87.299

. . .

27 [-5.846746] fitness = -636.1337

28 [-5.3915906] fitness = -493.7905

29 [2.0432386] fitness = -174.56447

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.0329428] fitness = 87.299

. . .

27 [-3.2025194] fitness = -37.490017

28 [-5.513774] fitness = -530.30286

29 [6.5257416] fitness = -1147.8535

Найкраще значення покоління 87.39345

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.3081512] fitness = 87.39345

. . .

27 [-4.1476526] fitness = -190.1256

28 [8.284954] fitness = -1619.2805

29 [-0.11850262] fitness = 57.11293

Найкраще значення покоління 87.39345

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.3081512] fitness = 87.39345

. . .

27 [5.3481913] fitness = -849.03973

28 [-3.944497] fitness = -151.8458

29 [-5.5965433] fitness = -555.7413

Найкраще значення покоління 87.39345

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.3081512] fitness = 87.39345

. . .

27 [5.311064] fitness = -839.9707

28 [7.2816963] fitness = -1348.2262

29 [-1.3336678] fitness = 87.182175

Найкраще значення покоління 87.39345

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.3081512] fitness = 87.39345

. . .

27 [-9.993774] fitness = -2815.4138

28 [7.414364] fitness = -1383.8365

29 [-9.235062] fitness = -2287.975

Найкраще значення покоління 87.39345

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.3081512] fitness = 87.39345

. . .

27 [-2.0752134] fitness = 64.13875

28 [8.76116] fitness = -1748.4127

29 [-5.083083] fitness = -407.04446

Найкраще значення покоління 87.39345

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.3081512] fitness = 87.39345

. . .

27 [-5.2539015] fitness = -454.11716

28 [-9.878004] fitness = -2730.9058

29 [-6.7164936] fitness = -957.6327

Найкраще значення покоління 87.39345

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.3081512] fitness = 87.39345

. . .

27 [-9.426895] fitness = -2415.498

28 [-2.5508537] fitness = 31.434376

29 [-8.614039] fitness = -1901.5336

Найкраще значення покоління 87.39345

Нова популяція

0 [-1.3081512] fitness = 87.39345

1 [-1.3081512] fitness = 87.39345

2 [-1.3081512] fitness = 87.39345

. . .

27 [-7.899252] fitness = -1505.2006

28 [3.3659286] fitness = -407.15616

29 [6.9715977] fitness = -1265.4482

Найкраще значення покоління 87.39345

Результат:

[-1.3081512] fitness = 87.39345

Результат виконання програми для десяти поколінь з розміром популяції 30 для знаходження мінімуму функції:

Початкова популяція

0 [17.452446] fitness = 3348.398

1 [39.36022] fitness = -19817.55

2 [10.250298] fitness = 2145.4995

. . .

27 [12.384445] fitness = 2665.1245

28 [13.576748] fitness = 2910.9612

29 [19.843952] fitness = 3230.5308

Нова популяція

0 [17.452446] fitness = 3348.398

1 [12.384445] fitness = 2665.1245

2 [13.576748] fitness = 2910.9612

. . .

27 [5.314946] fitness = 840.91785

28 [17.451038] fitness = 3348.3667

29 [2.2683868] fitness = 209.87567

Найкраще значення покоління -86.501236

Нова популяція

0 [17.778118] fitness = 3352.58

1 [17.452446] fitness = 3348.398

2 [17.778118] fitness = 3352.58

. . .

27 [30.26912] fitness = -2970.7168

28 [34.536892] fitness = -9249.75

29 [7.891222] fitness = 1512.5342

Найкраще значення покоління -9249.75

Нова популяція

0 [17.778118] fitness = 3352.58

1 [17.778118] fitness = 3352.58

2 [17.778118] fitness = 3352.58

. . .

27 [43.08577] fitness = -30909.719

28 [48.383575] fitness = -51742.1

29 [27.585018] fitness = -279.17188

Найкраще значення покоління -51742.1

Нова популяція

0 [17.778118] fitness = 3352.58

1 [17.778118] fitness = 3352.58

2 [17.778118] fitness = 3352.58

. . .

27 [4.962531] fitness = 756.0965

28 [16.279753] fitness = 3286.7544

29 [5.0639744] fitness = 780.26654

Найкраще значення покоління 115.38188

Нова популяція

0 [17.778118] fitness = 3352.58

1 [17.778118] fitness = 3352.58

2 [17.778118] fitness = 3352.58

. . .

27 [4.1873217] fitness = 578.72375

28 [33.916702] fitness = -8170.51

29 [27.205326] fitness = 31.708984

Найкраще значення покоління -11706.875

Нова популяція

0 [17.778118] fitness = 3352.58

1 [17.778118] fitness = 3352.58

2 [17.778118] fitness = 3352.58

. . .

27 [2.8432312] fitness = 308.23807

28 [11.448263] fitness = 2447.3682

29 [24.238922] fitness = 1924.2041

Найкраще значення покоління -68359.875

Нова популяція

0 [17.778118] fitness = 3352.58

1 [17.778118] fitness = 3352.58

2 [17.778118] fitness = 3352.58

. . .

27 [-5.1859865] fitness = 435.11853

28 [42.485962] fitness = -28936.543

29 [9.933571] fitness = 2062.5073

Найкраще значення покоління -28936.543

Нова популяція

0 [17.778118] fitness = 3352.58

1 [17.778118] fitness = 3352.58

2 [17.778118] fitness = 3352.58

. . .

27 [27.400837] fitness = -126.31055

28 [33.157635] fitness = -6929.8066

29 [21.8266] fitness = 2836.8848

Найкраще значення покоління -6929.8066

Нова популяція

0 [17.778118] fitness = 3352.58

1 [17.778118] fitness = 3352.58

2 [17.778118] fitness = 3352.58

. . .

27 [7.04628] fitness = 1285.3188

28 [22.27716] fitness = 2704.7295

29 [29.239098] fitness = -1832.0488

Найкраще значення покоління -31529.29

Результат:

[43.269558] fitness = -31529.29

Код програми

**public** GAPopulation generate(GAPopulation p, **int** xrate, **int** mrate,

**float**[] min\_range, **float**[] max\_range) {

//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)

System.*err*.println("error: xrate і/чи mrate неправилно встановлені");

GAIndividual[] newg = **new** GAIndividual[p.pop\_size];

**int** newg\_index = 0;

**int** xn = xrate \* p.pop\_size / 100;

//xn: Кількість нащадків, які будуть схрешення

**int** mn = mrate \* p.pop\_size / 100;

// mn: кількість нащадків які будуть створенні мутацією

// схрещування:

**for** (**int** i = 0; i < xn; i++) {

**int** p1 = p.tr\_select();

**int** p2 = p.tr\_select();

newg[newg\_index++] = GAIndividual.*uniform*(p.ind[p1], p.ind[p2]);

}

// мутація:

**for** (**int** i = 0; i < mn; i++){

**int** n = (**int**)(Math.*random*() \* p.pop\_size);

newg[newg\_index++] = p.ind[p.tr\_select()].mutate(p.ind[n],max\_range);

}

// відтворення:

**for** (**int** i = newg\_index; i < p.pop\_size; i++)

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

**return** **new** GAPopulation(newg);

}

**public** **int** tr\_select() {

//турнірна вибірка розміром pop\_size/10

//вона повертає індекс вибраного особи в ind []

**int** s\_index = *randg*.nextInt(pop\_size);

// індекс вибраного індивідума

**float** s\_fitness = ind[s\_index].fitness;

**int** tr\_size = Math.*min*(10, pop\_size);

**for** (**int** i = 1; i < tr\_size; i++) {

**int** tmp = *randg*.nextInt(pop\_size);

**if** (ind[tmp].fitness < s\_fitness) {//< для min//>для max

s\_index = tmp;

s\_fitness = ind[tmp].fitness;

}

}

**return** s\_index;

}

**public** **static** GAIndividual uniform(GAIndividual f, GAIndividual m) {

// рівномірне схрещення

**float**[] child = **new** **float**[f.genome\_size];

Random random = **new** Random();

**for** (**int** k = 0; k < f.genome\_size; k++) {

String fs = *floatToBinary*(f.genome[k]);

String ms = *floatToBinary*(m.genome[k]);

**int**[] maska = **new** **int**[fs.length()];

**for**(**int** i = 0; i < fs.length(); i++){

maska[i] = random.nextInt(2);

}

String childs = "";

**for** (**int** i = 0; i < fs.length(); i++) {

**if**(maska[i] == 0) childs += fs.charAt(i);

**else** childs += ms.charAt(i);

}

child[0] = *binaryToFloat*(childs);

}

**return** **new** GAIndividual(child);

}

**public** GAIndividual mutate(GAIndividual gaIndividual, **float**[] max\_reg) {

**float**[] result = **new** **float**[genome\_size];

**for** (**int** i = 0; i < genome\_size; i++)

result[i] = genome[i];

// класична мутація обміну

**for** (**int** i = 0; i < genome\_size; i++){

String string = *floatToBinary*( gaIndividual.genome[i]);

**int** n = string.indexOf(".");

**int** pp1 = (**int**)(Math.*random*()\*n);

**int** pp2 = (**int**)(Math.*random*()\*n);

**if**(pp1 > pp2) { **int** q = pp1; pp1 = pp2; pp2 = q;}

**char** [] charmas = string.toCharArray();

**char** c = charmas[pp1];

charmas[pp1] = charmas[pp2];

charmas[pp2] = c;

String end = "";

**for**(**int** j = 0; j < charmas.length; j++) {

end += charmas[j];}

result[i] = *binaryToFloat*(end);

**if**(result[i] > max\_reg[i]) result[i] = max\_reg[i];

}

**return** **new** GAIndividual(result);

}

**Висновки:** виконавши лабораторну роботу я реалізував за допомогою програмної мови Java програмне забезпечення для пошуку оптимумів функції, в якому використав турнірний відбір, рівномірне схрещування і класичну мутацію обміну. Програма показує результати за короткий період часу з невеликою похибкою навіть при невеликій кількості поколінь і розміру популяції, але при збільшенні цих параметрів похибка прямує до 0.