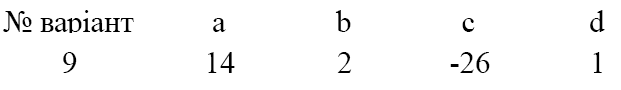
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКс-11 | 1 | Генетичний алгоритм пошуку |  |  |
| Гуменний Л.О. | |
| № залікової: | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р.З. | |

**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].



**Максимальне і мінімальне значення цільової функції**

Графік функції згідно варіанту показаний на рис.1.

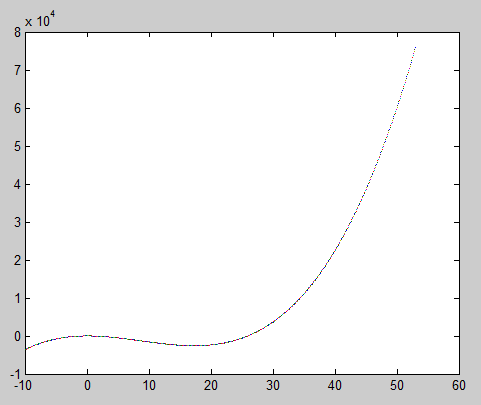
****

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

Для знаходження мінімуму функції було вибрано інтервал [0;53]. Пошук мінімуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() і рівний: 17.2948 при x = -2.5552e+003.

Для знаходження максимуму функції було вибрано інтервал [-10;10]. Пошук мінімуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() змінивши функцію на протилежну і рівний: 0.0385 при x = -14.0385. Графік функції на інтервалі [-10;10] показаний на рис.2.

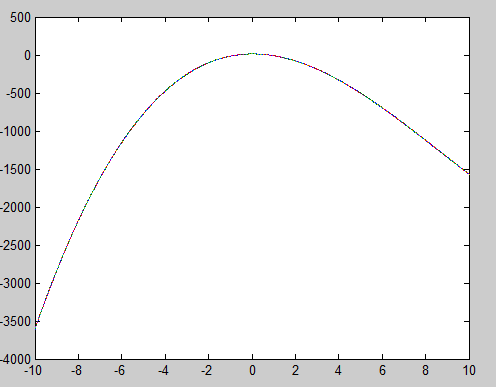


Рис.2. Графік функції на інтервалі[-10;10]

**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 | -2512.1626 | -2555.169 | -2555.2246 |
| xmin | 15.97067 | 17.248497 | 17.294273 |
| fmax | -12.889909 | 14.03714 | 14.038519 |
| xmax | 1.0796688 | 0.031248808 | 0.03855741 |

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

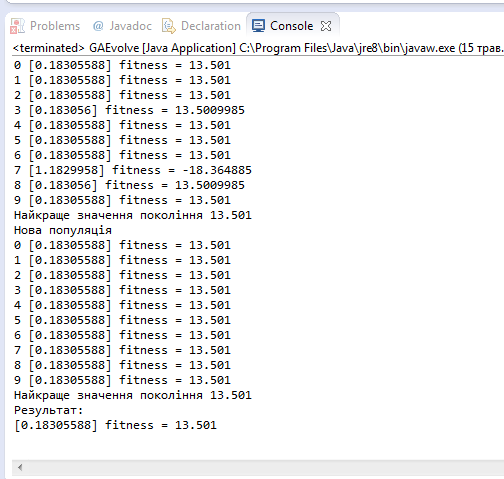


Рис.3. Результати виконання програми

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

Початкова популяція

0 [9.278826] fitness = -1407.0786

1 [1.0854399] fitness = -13.182953

2 [5.442657] fitness = -584.0748

3 [5.9458423] fitness = -683.0838

4 [4.0921807] fitness = -344.68274

5 [4.8657537] fitness = -476.63358

6 [5.5129557] fitness = -597.63025

7 [7.5866623] fitness = -1030.6512

8 [0.01876235] fitness = 14.028379

9 [1.0004288] fitness = -9.020157

10 [9.942979] fitness = -1553.5566

11 [6.170333] fitness = -728.6344

12 [5.0346317] fitness = -507.3507

13 [6.734378] fitness = -846.2627

14 [1.0924488] fitness = -13.540881

15 [9.539635] fitness = -1464.8901

16 [2.301486] fitness = -106.92422

17 [6.2658496] fitness = -748.2483

18 [3.4437191] fitness = -246.61203

19 [7.993726] fitness = -1120.6073

20 [7.7684937] fitness = -1070.7251

21 [7.512809] fitness = -1014.43384

22 [7.382257] fitness = -985.86

23 [1.0470253] fitness = -11.260948

24 [9.911582] fitness = -1546.6943

25 [1.8547165] fitness = -65.3497

26 [2.744639] fitness = -155.69437

27 [8.310876] fitness = -1191.1777

28 [0.8207923] fitness = -1.3216488

29 [3.2803397] fitness = -223.91714

Нова популяція

0 [0.01876235] fitness = 14.028379

1 [0.04799831] fitness = 14.036208

2 [0.5705142] fitness = 6.8640757

3 [1.0004288] fitness = -9.020157

4 [0.8224938] fitness = -1.3874949

5 [0.01876235] fitness = 14.028379

6 [1.0812595] fitness = -12.970531

7 [1.0241085] fitness = -10.146454

8 [2.244639] fitness = -101.19984

9 [3.2487056] fitness = -219.62178

10 [2.6749501] fitness = -147.5492

11 [0.022674322] fitness = 14.031993

12 [1.078981] fitness = -12.855089

13 [0.8207923] fitness = -1.3216488

14 [0.82076037] fitness = -1.3204136

15 [1.3227457] fitness = -26.531218

16 [0.8207923] fitness = -1.3216488

17 [1.0187621] fitness = -9.889909

18 [1.2582012] fitness = -22.651608

19 [1.1135147] fitness = -14.630096

20 [0.01876235] fitness = 14.028379

21 [10.0] fitness = -1566.0

22 [6.2658496] fitness = -748.2483

23 [7.382257] fitness = -985.86

24 [5.5129557] fitness = -597.63025

25 [3.4437191] fitness = -246.61203

26 [4.0854397] fitness = -343.60107

27 [1.0470253] fitness = -11.260948

28 [5.5129557] fitness = -597.63025

29 [0.01876235] fitness = 14.028379

Найкраще значення покоління 14.036208

Нова популяція

0 [0.04799831] fitness = 14.036208

1 [0.04799831] fitness = 14.036208

2 [0.04799831] fitness = 14.036208

3 [0.04799819] fitness = 14.036208

4 [0.04799819] fitness = 14.036208

5 [0.04799831] fitness = 14.036208

6 [0.04799831] fitness = 14.036208

7 [0.04799819] fitness = 14.036208

8 [0.04799819] fitness = 14.036208

9 [0.04799831] fitness = 14.036208

10 [0.04799819] fitness = 14.036208

11 [0.04799831] fitness = 14.036208

12 [0.04799831] fitness = 14.036208

13 [0.04799831] fitness = 14.036208

14 [0.04799819] fitness = 14.036208

15 [0.04799819] fitness = 14.036208

16 [0.04799831] fitness = 14.036208

17 [0.04799831] fitness = 14.036208

18 [0.04799831] fitness = 14.036208

19 [0.04799819] fitness = 14.036208

20 [0.04799819] fitness = 14.036208

21 [0.04799819] fitness = 14.036208

22 [0.04799819] fitness = 14.036208

23 [0.04799831] fitness = 14.036208

24 [0.04799819] fitness = 14.036208

25 [0.04799831] fitness = 14.036208

26 [0.04799819] fitness = 14.036208

27 [0.04799819] fitness = 14.036208

28 [0.04799831] fitness = 14.036208

29 [0.04799819] fitness = 14.036208

Найкраще значення покоління 14.036208

**Результат:**

**[0.04799831] fitness = 14.036208**

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

Початкова популяція

0 [9.494883] fitness = -1454.9927

1 [23.060877] fitness = -1502.916

2 [23.116497] fitness = -1480.6318

3 [22.437338] fitness = -1734.6904

4 [9.517969] fitness = -1460.0999

5 [35.84837] fitness = 12741.891

6 [12.440836] fitness = -2059.7297

7 [27.703022] fitness = 1376.4043

8 [37.654533] fitness = 16613.852

9 [27.283651] fitness = 1024.1152

10 [39.607674] fitness = 21440.492

11 [16.69447] fitness = -2546.1118

12 [1.2049391] fitness = -19.589535

13 [29.379894] fitness = 2990.211

14 [6.9638867] fitness = -895.24225

15 [50.24516] fitness = 61323.234

16 [42.62348] fitness = 30300.14

17 [48.357048] fitness = 52390.53

18 [24.542166] fitness = -814.9971

19 [11.486011] fitness = -1877.8358

20 [33.019733] fitness = 7733.674

21 [2.1208382] fitness = -89.1657

22 [9.026643] fitness = -1350.9407

23 [50.247906] fitness = 61336.87

24 [51.608036] fitness = 68321.39

25 [19.261703] fitness = -2447.4736

26 [47.038513] fitness = 46658.35

27 [15.716618] fitness = -2494.686

28 [18.911469] fitness = -2483.3447

29 [28.85912] fitness = 2452.9316

Нова популяція

0 [18.942394] fitness = -2480.4854

1 [15.954351] fitness = -2511.124

2 [0.7105217] fitness = 2.653876

3 [17.449154] fitness = -2554.6035

4 [16.880257] fitness = -2550.8467

5 [18.973885] fitness = -2477.5117

6 [3.8279915] fitness = -303.24197

7 [20.708927] fitness = -2213.71

8 [30.692688] fitness = 4496.09

9 [16.69447] fitness = -2546.1118

10 [18.72382] fitness = -2499.4468

11 [16.69447] fitness = -2546.1118

12 [21.708935] fitness = -1964.8662

13 [7.9685097] fitness = -1115.0111

14 [12.444502] fitness = -2060.3916

15 [15.716618] fitness = -2494.686

16 [11.954886] fitness = -1969.4083

17 [4.9720764] fitness = -495.89856

18 [14.880707] fitness = -2418.444

19 [12.6945] fitness = -2104.7966

20 [24.690592] fitness = -734.86816

21 [28.85912] fitness = 2452.9316

22 [29.379894] fitness = 2990.211

23 [18.911469] fitness = -2483.3447

24 [28.85912] fitness = 2452.9316

25 [18.911469] fitness = -2483.3447

26 [9.026643] fitness = -1350.9407

27 [16.69447] fitness = -2546.1118

28 [15.716618] fitness = -2494.686

29 [15.716618] fitness = -2494.686

Найкраще значення покоління -2554.6035

Нова популяція

0 [17.383965] fitness = -2555.0176

1 [17.383965] fitness = -2555.0176

2 [17.383965] fitness = -2555.0176

3 [17.383965] fitness = -2555.0176

4 [17.383965] fitness = -2555.0176

5 [17.383965] fitness = -2555.0176

6 [17.383965] fitness = -2555.0176

7 [17.383965] fitness = -2555.0176

8 [17.383965] fitness = -2555.0176

9 [17.383965] fitness = -2555.0176

10 [17.383965] fitness = -2555.0176

11 [17.383965] fitness = -2555.0176

12 [17.383965] fitness = -2555.0176

13 [17.383965] fitness = -2555.0176

14 [17.383965] fitness = -2555.0176

15 [17.383965] fitness = -2555.0176

16 [17.383965] fitness = -2555.0176

17 [17.383965] fitness = -2555.0176

18 [17.383965] fitness = -2555.0176

19 [17.383965] fitness = -2555.0176

20 [17.383965] fitness = -2555.0176

21 [17.383965] fitness = -2555.0176

22 [9.384209] fitness = -1430.4744

23 [17.383965] fitness = -2555.0176

24 [53.0] fitness = 75963.0

25 [18.383965] fitness = -2523.2256

26 [17.383965] fitness = -2555.0176

27 [17.383965] fitness = -2555.0176

28 [17.383965] fitness = -2555.0176

29 [17.383965] fitness = -2555.0176

Найкраще значення покоління -2555.0176

Нова популяція

0 [17.383965] fitness = -2555.0176

1 [17.383965] fitness = -2555.0176

2 [17.383965] fitness = -2555.0176

3 [17.383965] fitness = -2555.0176

4 [17.383965] fitness = -2555.0176

5 [17.383965] fitness = -2555.0176

6 [17.383965] fitness = -2555.0176

7 [17.383965] fitness = -2555.0176

8 [17.383965] fitness = -2555.0176

9 [17.383965] fitness = -2555.0176

10 [17.383965] fitness = -2555.0176

11 [17.383965] fitness = -2555.0176

12 [17.383965] fitness = -2555.0176

13 [17.383965] fitness = -2555.0176

14 [17.383965] fitness = -2555.0176

15 [17.383965] fitness = -2555.0176

16 [17.383965] fitness = -2555.0176

17 [17.383965] fitness = -2555.0176

18 [17.383965] fitness = -2555.0176

19 [17.383965] fitness = -2555.0176

20 [17.383965] fitness = -2555.0176

21 [17.383965] fitness = -2555.0176

22 [17.383965] fitness = -2555.0176

23 [20.383965] fitness = -2278.7285

24 [17.383965] fitness = -2555.0176

25 [17.383965] fitness = -2555.0176

26 [33.383965] fitness = 8310.117

27 [17.383965] fitness = -2555.0176

28 [17.383965] fitness = -2555.0176

29 [17.383965] fitness = -2555.0176

Найкраще значення покоління -2555.0176

**Результат:**

**[17.383965] fitness = -2555.0176**

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.