|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКс-11 | 4 | реалізувати генетичний алгоритм пошуку максимального і мінімального значення цільової функції |  |  |
| Якимець А.Р. | |
| № 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 - 15



**Максимальне і мінімальне значення цільової функції**

Графік функції згідно варіанту показаний на рис.1.

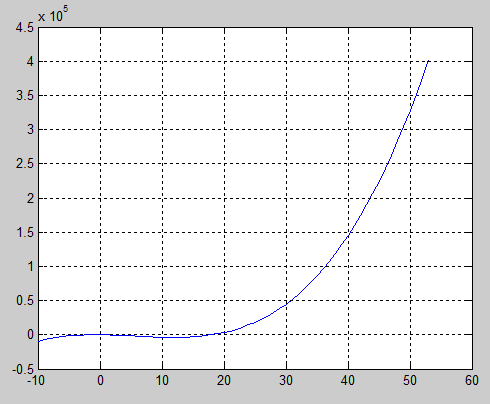
****

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: -3.8509e+003 при x =11.8422.

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: 72.0430 при x = -0.6755 . Графік функції на інтервалі [-10;10] показаний на рис.2.

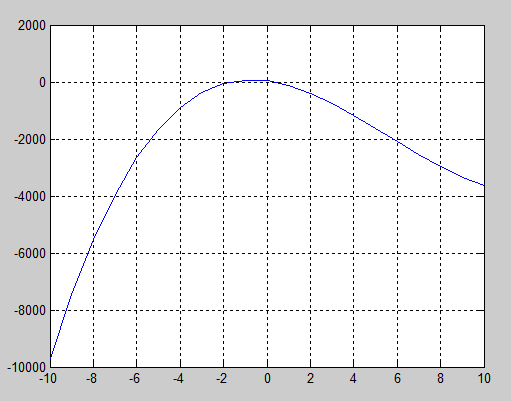


Рис.2. Графік функції на інтервалі[-10;10]

**Результати виконання програми**

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

Результати:

|  |  |  |  |
| --- | --- | --- | --- |
| Кількість поколінь, розмір популяції | 50 | 100 | 250 |
| fmin | -3847.7559 | -3850.8774 | -3850.438 |
| xmin | 12.045266 | 11.852991 | 11.764885 |
| fmax | 71.97658 | 71.59324 | 72.03872 |
| xmax | -0.70525646 | -0.7527714 | -0.68306065 |

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

Результат виконання програми для 10 поколінь з розміром популяції 50 для знаходження мінімуму функції:

Initial population

0 [30.752493] fitness = 50056.28

1 [35.162846] fitness = 87727.83

2 [51.999397] fitness = 376295.7

3 [39.59706] fitness = 139527.8

4 [14.169377] fitness = -3393.7188

5 [47.062004] fitness = 264065.06

6 [42.57522] fitness = 183200.03

7 [1.73703] fitness = -308.9478

8 [39.360558] fitness = 136378.33

9 [52.36243] fitness = 385584.3

10 [6.3920774] fitness = -2267.4824

11 [39.994976] fitness = 144929.97

12 [51.224243] fitness = 356952.5

13 [37.48338] fitness = 112962.5

14 [17.098413] fitness = -1195.0059

15 [3.7646606] fitness = -1058.5552

16 [43.16871] fitness = 192823.83

17 [4.52999] fitness = -1398.937

18 [29.060057] fitness = 38831.957

19 [25.584885] fitness = 20715.633

20 [36.609356] fitness = 102990.12

21 [48.076317] fitness = 285045.78

22 [41.365097] fitness = 164540.72

23 [32.454426] fitness = 63088.617

24 [47.9143] fitness = 281624.72

25 [14.448065] fitness = -3270.0986

26 [1.8450947] fitness = -341.0966

27 [34.516754] fitness = 81395.086

28 [13.278378] fitness = -3684.126

29 [13.873315] fitness = -3507.5273

30 [41.313927] fitness = 163779.69

31 [49.20695] fitness = 309670.3

32 [21.41858] fitness = 6549.7617

33 [33.422188] fitness = 71324.64

34 [40.59771] fitness = 153362.33

35 [14.634642] fitness = -3178.1348

36 [52.805588] fitness = 397123.6

37 [45.303528] fitness = 230104.0

38 [38.32533] fitness = 123121.78

39 [41.61623] fitness = 168308.22

40 [7.892425] fitness = -2925.6396

41 [30.637905] fitness = 49243.023

42 [48.612617] fitness = 296561.4

43 [15.872672] fitness = -2368.918

44 [34.004715] fitness = 76582.484

45 [0.79439586] fitness = -77.53808

46 [2.0870996] fitness = -416.84705

47 [39.80794] fitness = 142374.56

48 [26.015556] fitness = 22625.559

49 [5.229144] fitness = -1723.1007

New population

0 [14.420231] fitness = -3283.1787

1 [14.391711] fitness = -3296.4111

2 [15.383909] fitness = -2731.0762

3 [15.810425] fitness = -2418.2617

4 [5.842457] fitness = -2011.1614

5 [14.8943615] fitness = -3037.4932

6 [13.278378] fitness = -3684.126

7 [7.811535] fitness = -2892.611

8 [13.779565] fitness = -3539.8994

9 [13.842555] fitness = -3518.3408

10 [11.390446] fitness = -3835.9248

11 [5.3086424] fitness = -1760.3765

12 [14.392397] fitness = -3296.0967

13 [3.2783623] fitness = -854.8771

14 [1.7943043] fitness = -325.8543

15 [14.199087] fitness = -3381.3125

16 [15.044408] fitness = -2949.3994

17 [13.834733] fitness = -3521.0605

18 [15.904626] fitness = -2343.2188

19 [14.464872] fitness = -3262.12

20 [1.0205631] fitness = -124.505974

21 [12.479434] fitness = -3819.3535

22 [15.373306] fitness = -2738.2998

23 [30.62554] fitness = 49155.742

24 [12.779173] fitness = -3781.6602

25 [15.372672] fitness = -2738.7314

26 [13.278378] fitness = -3684.126

27 [14.434888] fitness = -3276.3125

28 [14.278737] fitness = -3347.1455

29 [12.447832] fitness = -3822.4507

30 [14.869392] fitness = -3051.662

31 [6.3920774] fitness = -2267.4824

32 [13.399715] fitness = -3653.579

33 [7.3764334] fitness = -2709.2668

34 [14.665511] fitness = -3162.1934

35 [4.52999] fitness = -1398.937

36 [49.20695] fitness = 309670.3

37 [51.999397] fitness = 376295.7

38 [21.41858] fitness = 6549.7617

39 [35.892426] fitness = 95235.49

40 [34.004715] fitness = 76582.484

41 [13.278378] fitness = -3684.126

42 [39.360558] fitness = 136378.33

43 [32.454426] fitness = 63088.617

44 [48.612617] fitness = 296561.4

45 [13.278378] fitness = -3684.126

46 [14.448065] fitness = -3270.0986

47 [13.278378] fitness = -3684.126

48 [5.229144] fitness = -1723.1007

49 [15.872672] fitness = -2368.918

Best in generation -3835.9248

New population

0 [9.280605] fitness = -3425.2827

1 [12.44808] fitness = -3822.4268

2 [14.46487] fitness = -3262.121

3 [12.3384285] fitness = -3831.9033

4 [12.38666] fitness = -3827.977

5 [13.264212] fitness = -3687.5137

.

.

.

48 [-0.6986542] fitness = 72.00281

49 [-0.6986542] fitness = 72.00281

Best in generation 72.00281

New population

0 [-0.6986542] fitness = 72.00281

1 [-0.6986542] fitness = 72.00281

2 [-0.6986542] fitness = 72.00281

3 [-0.6986542] fitness = 72.00281

4 [-0.6986542] fitness = 72.00281

5 [-0.6986542] fitness = 72.00281

6 [-0.6986542] fitness = 72.00281

7 [-0.6986542] fitness = 72.00281

8 [-0.6986542] fitness = 72.00281

9 [-0.6986542] fitness = 72.00281

10 [-0.6986542] fitness = 72.00281

11 [-0.6986542] fitness = 72.00281

12 [-0.6986542] fitness = 72.00281

13 [-0.6986542] fitness = 72.00281

14 [-0.6986542] fitness = 72.00281

15 [-0.6986542] fitness = 72.00281

16 [-0.6986542] fitness = 72.00281

17 [-0.6986542] fitness = 72.00281

18 [-0.6986542] fitness = 72.00281

19 [-0.6986542] fitness = 72.00281

20 [-0.6986542] fitness = 72.00281

21 [-0.6986542] fitness = 72.00281

22 [-0.6986542] fitness = 72.00281

23 [-0.6986542] fitness = 72.00281

24 [-0.6986542] fitness = 72.00281

25 [-0.6986542] fitness = 72.00281

26 [-0.6986542] fitness = 72.00281

27 [-0.6986542] fitness = 72.00281

28 [-0.6986542] fitness = 72.00281

29 [-0.6986542] fitness = 72.00281

30 [-0.6986542] fitness = 72.00281

31 [-0.6986542] fitness = 72.00281

32 [-0.6986542] fitness = 72.00281

33 [-0.6986542] fitness = 72.00281

34 [-0.6986542] fitness = 72.00281

35 [1.9099026] fitness = -360.8812

36 [0.55415726] fitness = -34.09344

37 [6.3744793] fitness = -2259.345

38 [7.1691856] fitness = -2618.951

39 [5.7133045] fitness = -1950.5099

40 [-0.8658781] fitness = 69.29464

41 [9.735868] fitness = -3555.0413

42 [7.557701] fitness = -2786.7534

43 [-5.2345657] fitness = -1868.0498

44 [-0.33917522] fitness = 63.69708

45 [-0.6986542] fitness = 72.00281

46 [-0.6986542] fitness = 72.00281

47 [-0.6986542] fitness = 72.00281

48 [-0.6986542] fitness = 72.00281

49 [-0.6986542] fitness = 72.00281

Best in generation 72.00281

Results:

[-0.6986542] fitness = 72.00281

Код програми

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

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

//creating new population with p, xrate - % of population is mating

// mrate - % of population mutated, rest - recreation

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

|| xrate + mrate > 100)

System.***err***.println("error: xrate and/or mrate set incorrectly");

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

**int** newg\_index = 0;

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

//xn: amount of sons to mate

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

// mn: amount of sons to mutate

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

// select to parents for cross-over:

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

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

//System.out.println(p1+ " "+ p2);

newg[newg\_index++] = GAIndividual.*xover1p*(p.ind[p1], p.ind[p2]);

}

// mutation

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

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

max\_range);

// recreation:

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

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

//for (int i = 0; i < p.pop\_size; i++) {

// System.out.println(i + " " + p.ind[i]);}

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

}

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

//tournament selection of size pop\_size/10

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

**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) {//< for min//>for max

s\_index = tmp;

s\_fitness = ind[tmp].fitness;

}

}

**return** s\_index;

}

**private** **void** evaluate() {

//evaluation

**int** best = 0;

// index of best individual

**float** best\_fitness = ind[0].fitness;

**float** sum = ind[0].fitness;

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

sum += ind[i].fitness;

**if** (ind[i].fitness > best\_fitness) {//< for min//>for max

best = i;

best\_fitness = ind[i].fitness;

}

}

best\_index = best;

}

**Висновки:** виконавши лабораторну роботу я вивчив еволюційні оператори схрещування та мутації в генетичних алгоритмах. Реалізував за допомогою програмної мови Java програмне забезпечення для пошуку оптимумів функції в якому використав турнірний відбір, одно-точковий кросинговер і точкову мутацію. В результаті виконання програми похибка знаходження оптимуму є невеликою, і прямує до 0 при збільшенні кількості популяції і кількості поколінь.