|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКc-11 | 4 | КОМБІНАТОРНА  ОПТИМІЗАЦІЯ ЗА ДОПОМОГОЮ ЕВОЛЮЦІЙНИХ  МЕТОДІВ |  |  |
| Левицький О.О. | |
| 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 - 8





**Максимальне і мінімальне значення цільової функції**

Графік функції згідно варіанту показаний на рис.1.

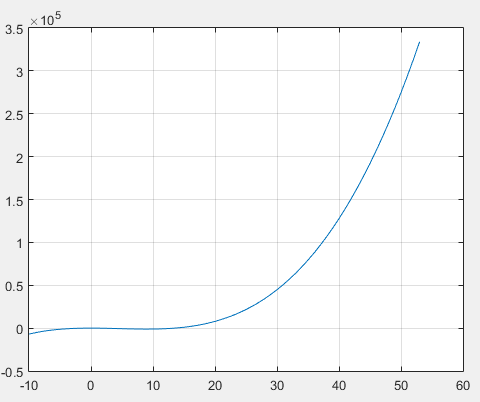
****

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

Для знаходження мінімуму та максимуму функції було вибрано інтервал [0;53]. Пошук мінімуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() і рівний: -1.1130e+03 при x = -8.9878.

Пошук максимуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fmnbnd() змінивши функцію на протилежну і рівний: 3,3386е5 при x = 52,99.

**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 | -1154,99 | -6090,05 | -6884,06 |
| xmin | -4,716 | -9,49 | -9,98 |
| fmax | 221422,38 | 329185,03 | 333849,44 |
| xmax | 46,91 | 52,77 | 52,999 |

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 десяти поколінь з розміром популяції 20 для знаходження мінімуму функції:

Початкова популяція

0 [50.187828] fitness = -278099.8

1 [-9.229362] fitness = 5679.911

2 [29.087982] fitness = -39769.82

3 [25.295048] fitness = -22770.371

4 [4.8658524] fitness = 628.3683

5 [44.965157] fitness = -191518.14

6 [-7.1462984] fitness = 3068.4883

7 [8.762579] fitness = 1110.9666

8 [31.14608] fitness = -51602.105

9 [29.593464] fitness = -42495.81

Нова популяція

0 [-9.229362] fitness = 5679.911

1 [-9.229362] fitness = 5679.911

2 [-9.229362] fitness = 5679.911

3 [-9.229362] fitness = 5679.911

4 [8.762579] fitness = 1110.9666

5 [8.762579] fitness = 1110.9666

6 [8.762579] fitness = 1110.9666

7 [-9.229362] fitness = 5679.911

8 [33.138386] fitness = -64993.816

9 [51.989265] fitness = -313043.6

Найкраще значення покоління 5679.911

...

Нова популяція

0 [-9.229362] fitness = 5679.911

1 [-9.229362] fitness = 5679.911

2 [-9.229362] fitness = 5679.911

3 [-9.229362] fitness = 5679.911

4 [-9.229362] fitness = 5679.911

5 [-9.229362] fitness = 5679.911

6 [-9.229362] fitness = 5679.911

7 [-9.229362] fitness = 5679.911

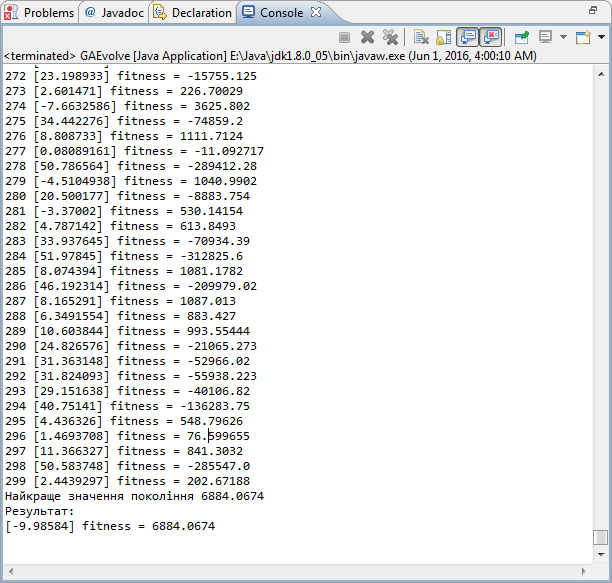
8 [-5.8654065] fitness = 1922.5592

9 [45.165993] fitness = -194463.4

Найкраще значення покоління 5679.911

Результат:

[-9.229362] fitness = 5679.911



*Рис.3. Результати виконання програми для знаходження мінімуму з розміром популяції 300*

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

Початкова популяція

0 [38.481655] fitness = 111425.91

1 [19.220238] fitness = 6382.414

2 [21.776987] fitness = 11850.668

3 [9.821054] fitness = -1082.8787

4 [40.811134] fitness = 136982.36

5 [44.735653] fitness = 188188.5

6 [-4.524658] fitness = -1048.5977

7 [14.478104] fitness = 616.0596

8 [37.6955] fitness = 103562.75

9 [41.40066] fitness = 144004.22

Нова популяція

0 [44.735653] fitness = 188188.5

1 [44.735653] fitness = 188188.5

2 [44.735653] fitness = 188188.5

3 [41.40066] fitness = 144004.22

4 [44.735653] fitness = 188188.5

5 [44.735653] fitness = 188188.5

6 [41.40066] fitness = 144004.22

7 [40.811134] fitness = 136982.36

8 [36.72101] fitness = 94328.33

9 [45.390507] fitness = 197790.89

Найкраще значення покоління 197790.89

...

Нова популяція

0 [45.390507] fitness = 197790.89

1 [45.390507] fitness = 197790.89

2 [45.390507] fitness = 197790.89

3 [45.390507] fitness = 197790.89

4 [45.390507] fitness = 197790.89

5 [45.390507] fitness = 197790.89

6 [45.390507] fitness = 197790.89

7 [45.390507] fitness = 197790.89

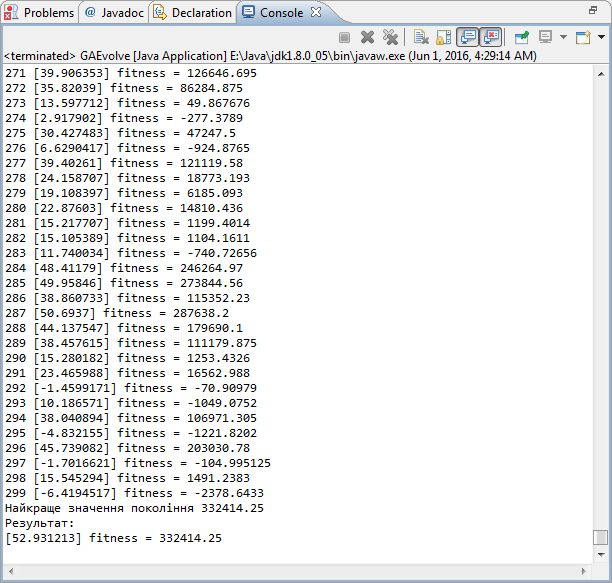
8 [48.379154] fitness = 245703.67

9 [6.2012196] fitness = -860.4087

Найкраще значення покоління 245703.67

Результат:

[48.379154] fitness = 245703.67



*Рис.4. Результати виконання програми для знаходження максимуму з розміром популяції 300*

Код програми на мові програмування Java:

**Коди Програми**

**GAEvolve.java**

**public** **class** GAEvolve {

**public** **float**[] best\_fitness;

**public** GAIndividual best\_ind;

**public** GAEvolve(**int** generations, **int** pop\_size, **int** genome\_size, **int** xrate,

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

best\_fitness = **new** **float**[generations];

GAPopulation gap = **new** GAPopulation(pop\_size, genome\_size, min\_range,

max\_range);

best\_fitness[0] = gap.ind[gap.best\_index].fitness;

**for** (**int** i = 1; i < generations; i++) {

gap = gap.generate(gap, xrate, mrate, min\_range, max\_range);

best\_fitness[i] = gap.ind[gap.best\_index].fitness;

System.***out***.println("Найкраще значення покоління " + best\_fitness[i]);

}

best\_ind = gap.ind[gap.best\_index];

}

**public** **static** **void** main(String[] args) {

**float**[] min = **new** **float**[] { -10f };

**float**[] max = **new** **float**[] { 53 };

GAEvolve gae = **new** GAEvolve(10, 20, 1, 80, 20, min, max);

System.***out***.println("Результат:");

System.***out***.println(gae.best\_ind);

}

}

**GAIndividual.java**

**import** java.util.Random;

**public** **class** GAIndividual {

**private** **static** Random *randg* = **new** Random();

**public** **int** genome\_size;

**public** **float**[] genome;

**public** **float** fitness;

**public** GAIndividual(**int** gsize, **float**[] min\_range, **float**[] max\_range) {

genome\_size = gsize;

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

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

genome[i] = *randg*.nextFloat() \* (max\_range[i] - min\_range[i])

+ min\_range[i];

}

evalFitness();

}

**public** GAIndividual(**float** d[]) {

genome\_size = d.length;

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

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

genome[i] = d[i];

}

evalFitness();

}

**public** GAIndividual mutate(**float**[] min\_range, **float**[] max\_range) {

**float** rate = 1.0f / (**float**) genome\_size;

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

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

result[i] = genome[i];

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

**if** ((**float**) Math.*random*() < rate)

result[i] = *randg*.nextFloat() \* (max\_range[i] - min\_range[i])

+ min\_range[i];

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

}

**public** **static** GAIndividual xover1p(GAIndividual f, GAIndividual m) {

Random rng = **new** Random();

**int** xpoint = 1 + rng.nextInt(1);

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

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

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

}

**for** (**int** i = xpoint; i < f.genome\_size; i++) {

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

}

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

}

**public** String toString() {

String s = "[";

s += genome[0] + "]";

s += " fitness = " + fitness;

**return** s;

}

**private** **void** evalFitness() {

**int** a = 12;

**int** b = -8;

**int** c = -40;

**int** d = 3;

fitness = (a + b \* genome[0] + c \* genome[0] \* genome[0] + d

\* genome[0] \* genome[0] \* genome[0]);

}

}

**GAPopulation.java**

**import** java.util.Random;

**public** **class** GAPopulation {

**private** **static** Random *randg* = **new** Random();

**public** **int** pop\_size;

**public** GAIndividual[] ind;

**public** **int** best\_index;

**public** GAPopulation(**int** psize, **int** gsize, **float**[] min\_range,

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

pop\_size = psize;

ind = **new** GAIndividual[pop\_size];

System.***out***.println("Початкова популяція");

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

{ind[i] = **new** GAIndividual(gsize, min\_range, max\_range);

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

evaluate();

}

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

**int** best = 0;

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

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

**if** (ind[i].fitness > best\_fitness) {

best = i;

best\_fitness = ind[i].fitness;

}

}

best\_index = best;

}

**public** GAPopulation(GAIndividual[] p) {

pop\_size = p.length;

ind = **new** GAIndividual[pop\_size];

System.***out***.println("Нова популяція");

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

ind[i] = p[i];

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

evaluate();

}

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

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

**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;

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

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

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

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

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

}

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

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

max\_range);

**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() {

**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) {

s\_index = tmp;

s\_fitness = ind[tmp].fitness;

}

}

**return** s\_index;

}

**public** String toString() {

String s = "best individual = " + ind[best\_index];

**return** s;

}

}

Код програми на мові програмування Matlab для знаходження екстремумів функції:

clc

a=12;

b=-8;

c=-40;

d=3;

x=-10:1:53;

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

plot(x,y)

grid on

minX=fminbnd(@(x) a+b\*x+c\*x^2+d\*x^3,-10,53)

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

maxX=fminbnd(@(x) -(a+b\*x+c\*x^2+d\*x^3),-10,53)

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

**Висновки**

Виконавши лабораторну роботу я реалізував за допомогою мови Java програмне забезпечення для пошуку оптимумів функції в якому використовується турнірний відбір та одноточкова мутація. 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.