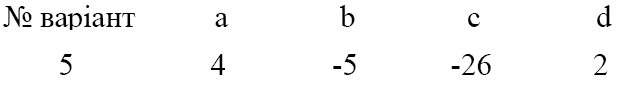
**Мета роботи**: реалізувати генетичний алгоритм пошуку максимального і мінімального значення цільової функції згідно варіанту.

**Завдання:** реалізувати генетичний алгоритм пошуку максимального і мінімального значення цільової функції

- f(x) = a + bx + cx2 + dx3 в інтервалі x = [-10, 53].

В - 5



**Максимальне і мінімальне значення цільової функції**

Графік функції згідно варіанту показаний на рис.1.

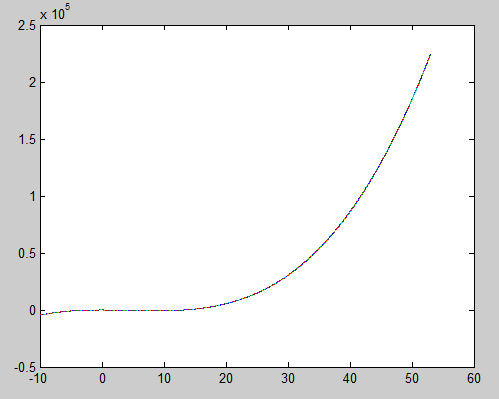
****

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

Для знаходження мінімуму функції я обрав інтервал [5;15]. Пошук мінімуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() і рівний: - 690.5349 при x = 8.76. Графік функції на інтервалі [5;15] показаний на рис.2.

Для знаходження максимуму функції я обрав інтервал [-10;10]. Пошук мінімуму цільової функції для перевірки реалізовано у Matlab за допомогою функції fminbnd() змінивши функцію на протилежну і рівний: 4.2386 при x = -0.0951. Графік функції на інтервалі [-10;10] показаний на рис.3.

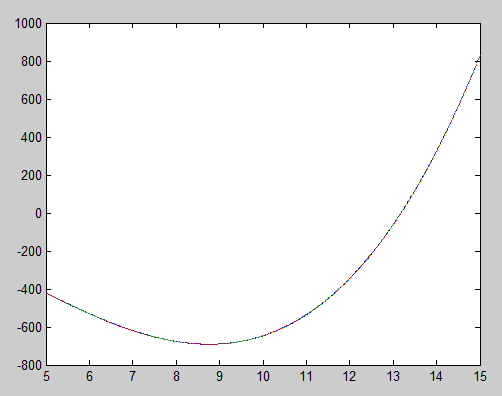
****

Рис.2. Графік функції на інтервалі[5;15]

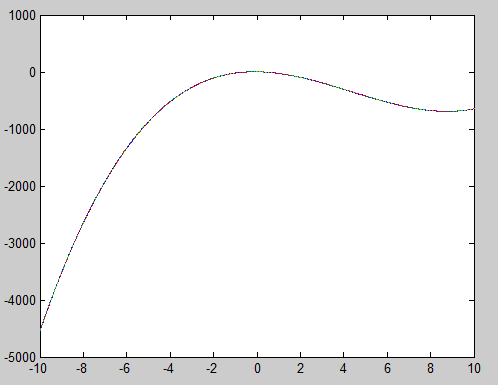


Рис.3. Графік функції на інтервалі[-10;10]

**Результати виконання програми**

Програмне забезпечення для пошуку оптимумів функції в якому використовується турнірний відбір, одно точковий кросинговер і точкова мутація.

Результати для десяти поколінь з розміром популяції 10:

Мінімум на інтервалі[5;15] - x = [8.796791] ; f = -690.5023.

Максимум на інтервалі[-10;10] – x = [-0.03215599]; f = 4.133829.

Отже похибка при знаходженні мінімуму - 0.004 %, максимуму 2,5 %, при збільшенні кількості поколінь і популяції похибки прямують до 0.

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

Початкова популяція

0 [11.449142] fitness = -459.82715

1 [13.487187] fitness = 113.80664

2 [8.308405] fitness = -685.25977

3 [9.914709] fitness = -652.15076

4 [12.642687] fitness = -173.43774

5 [11.255337] fitness = -494.31372

6 [12.529922] fitness = -206.25293

7 [14.658211] fitness = 643.28613

8 [6.807999] fitness = -604.0243

9 [12.356955] fitness = -254.16357

Нова популяція

0 [8.308405] fitness = -685.25977

1 [8.308405] fitness = -685.25977

2 [8.308405] fitness = -685.25977

3 [8.308405] fitness = -685.25977

4 [8.308405] fitness = -685.25977

5 [8.308405] fitness = -685.25977

6 [9.914709] fitness = -652.15076

7 [11.396473] fitness = -469.51318

8 [12.047099] fitness = -332.8296

9 [8.308405] fitness = -685.25977

Найкраще значення покоління -685.25977

Нова популяція

0 [8.308405] fitness = -685.25977

1 [8.308405] fitness = -685.25977

2 [8.308405] fitness = -685.25977

3 [8.308405] fitness = -685.25977

4 [8.308405] fitness = -685.25977

5 [8.308405] fitness = -685.25977

6 [8.308405] fitness = -685.25977

7 [5.4177165] fitness = -468.19363

8 [10.707354] fitness = -575.229

9 [8.308405] fitness = -685.25977

Найкраще значення покоління -685.25977

Нова популяція

0 [8.308405] fitness = -685.25977

1 [8.308405] fitness = -685.25977

2 [8.308405] fitness = -685.25977

3 [8.308405] fitness = -685.25977

4 [8.308405] fitness = -685.25977

5 [8.308405] fitness = -685.25977

6 [8.308405] fitness = -685.25977

7 [8.812399] fitness = -690.46655

8 [5.4316893] fitness = -469.73798

9 [8.308405] fitness = -685.25977

Найкраще значення покоління -690.46655

Нова популяція

0 [8.308405] fitness = -685.25977

1 [8.812399] fitness = -690.46655

2 [8.308405] fitness = -685.25977

3 [8.308405] fitness = -685.25977

4 [8.812399] fitness = -690.46655

5 [8.812399] fitness = -690.46655

6 [8.812399] fitness = -690.46655

7 [12.705692] fitness = -154.55078

8 [6.8803] fitness = -609.7967

9 [8.812399] fitness = -690.46655

Найкраще значення покоління -690.46655

Нова популяція

0 [8.812399] fitness = -690.46655

1 [8.812399] fitness = -690.46655

2 [8.812399] fitness = -690.46655

3 [8.812399] fitness = -690.46655

4 [8.812399] fitness = -690.46655

5 [8.812399] fitness = -690.46655

6 [8.812399] fitness = -690.46655

7 [12.62011] fitness = -180.10864

8 [13.9276085] fitness = 294.2329

9 [8.812399] fitness = -690.46655

Найкраще значення покоління -690.46655

Нова популяція

0 [8.812399] fitness = -690.46655

1 [8.812399] fitness = -690.46655

2 [8.812399] fitness = -690.46655

3 [8.812399] fitness = -690.46655

4 [8.812399] fitness = -690.46655

5 [8.812399] fitness = -690.46655

6 [8.812399] fitness = -690.46655

7 [6.390701] fitness = -567.815

8 [14.338398] fitness = 482.62988

9 [8.812399] fitness = -690.46655

Найкраще значення покоління -690.46655

Нова популяція

0 [8.812399] fitness = -690.46655

1 [8.812399] fitness = -690.46655

2 [8.812399] fitness = -690.46655

3 [8.812399] fitness = -690.46655

4 [8.812399] fitness = -690.46655

5 [8.812399] fitness = -690.46655

6 [8.812399] fitness = -690.46655

7 [10.9164915] fitness = -547.16504

8 [8.796791] fitness = -690.5023

9 [8.812399] fitness = -690.46655

Найкраще значення покоління -690.5023

Нова популяція

0 [8.812399] fitness = -690.46655

1 [8.812399] fitness = -690.46655

2 [8.796791] fitness = -690.5023

3 [8.796791] fitness = -690.5023

4 [8.812399] fitness = -690.46655

5 [8.796791] fitness = -690.5023

6 [8.796791] fitness = -690.5023

7 [12.762708] fitness = -137.11719

8 [7.0159664] fitness = -620.1935

9 [8.796791] fitness = -690.5023

Найкраще значення покоління -690.5023

Нова популяція

0 [8.796791] fitness = -690.5023

1 [8.796791] fitness = -690.5023

2 [8.796791] fitness = -690.5023

3 [8.796791] fitness = -690.5023

4 [8.796791] fitness = -690.5023

5 [8.796791] fitness = -690.5023

6 [8.796791] fitness = -690.5023

7 [11.788918] fitness = -391.5735

8 [7.652707] fitness = -660.5805

9 [8.796791] fitness = -690.5023

Найкраще значення покоління -690.5023

**Результат:**

**[8.796791] fitness = -690.5023**

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

Початкова популяція

0 [-9.184466] fitness = -3692.7935

1 [-3.7437296] fitness = -446.62518

2 [-8.182565] fitness = -2791.6177

3 [-6.659254] fitness = -1706.3091

4 [-1.0603342] fitness = -22.31464

5 [-5.0355053] fitness = -885.4503

6 [-6.6049385] fitness = -1673.5145

7 [-3.1974373] fitness = -311.2052

8 [7.779024] fitness = -666.7712

9 [-1.2449942] fitness = -33.93481

Нова популяція

0 [-3.1974373] fitness = -311.2052

1 [-1.0603342] fitness = -22.31464

2 [-1.0603342] fitness = -22.31464

3 [-3.1974373] fitness = -311.2052

4 [-1.0603342] fitness = -22.31464

5 [-1.0603342] fitness = -22.31464

6 [-1.2449942] fitness = -33.93481

7 [-8.30163] fitness = -2890.5833

8 [-0.21579266] fitness = 3.8481376

9 [-1.0603342] fitness = -22.31464

Найкраще значення покоління 3.8481376

Нова популяція

0 [-0.21579266] fitness = 3.8481376

1 [-0.21579266] fitness = 3.8481376

2 [-1.0603342] fitness = -22.31464

3 [-0.21579266] fitness = 3.8481376

4 [-0.21579266] fitness = 3.8481376

5 [-0.21579266] fitness = 3.8481376

6 [-0.21579266] fitness = 3.8481376

7 [-6.7442846] fitness = -1758.4309

8 [7.1201363] fitness = -627.77576

9 [-0.21579266] fitness = 3.8481376

Найкраще значення покоління 3.8481376

Нова популяція

0 [-0.21579266] fitness = 3.8481376

1 [-0.21579266] fitness = 3.8481376

2 [-0.21579266] fitness = 3.8481376

3 [-0.21579266] fitness = 3.8481376

4 [-0.21579266] fitness = 3.8481376

5 [-0.21579266] fitness = 3.8481376

6 [-0.21579266] fitness = 3.8481376

7 [-5.8752766] fitness = -1269.7302

8 [-4.802141] fitness = -793.04395

9 [-0.21579266] fitness = 3.8481376

Найкраще значення покоління 3.8481376

Нова популяція

0 [-0.21579266] fitness = 3.8481376

1 [-0.21579266] fitness = 3.8481376

2 [-0.21579266] fitness = 3.8481376

3 [-0.21579266] fitness = 3.8481376

4 [-0.21579266] fitness = 3.8481376

5 [-0.21579266] fitness = 3.8481376

6 [-0.21579266] fitness = 3.8481376

7 [-1.128005] fitness = -26.31279

8 [8.913231] fitness = -689.9186

9 [-0.21579266] fitness = 3.8481376

Найкраще значення покоління 3.8481376

Нова популяція

0 [-0.21579266] fitness = 3.8481376

1 [-0.21579266] fitness = 3.8481376

2 [-0.21579266] fitness = 3.8481376

3 [-0.21579266] fitness = 3.8481376

4 [-0.21579266] fitness = 3.8481376

5 [-0.21579266] fitness = 3.8481376

6 [-0.21579266] fitness = 3.8481376

7 [-8.704218] fitness = -3241.25

8 [-6.798985] fitness = -1792.4688

9 [-0.21579266] fitness = 3.8481376

Найкраще значення покоління 3.8481376

Нова популяція

0 [-0.21579266] fitness = 3.8481376

1 [-0.21579266] fitness = 3.8481376

2 [-0.21579266] fitness = 3.8481376

3 [-0.21579266] fitness = 3.8481376

4 [-0.21579266] fitness = 3.8481376

5 [-0.21579266] fitness = 3.8481376

6 [-0.21579266] fitness = 3.8481376

7 [-0.03215599] fitness = 4.1338296

8 [-2.3104215] fitness = -147.9034

9 [-0.21579266] fitness = 3.8481376

Найкраще значення покоління 4.1338296

Нова популяція

0 [-0.03215599] fitness = 4.1338296

1 [-0.03215599] fitness = 4.1338296

2 [-0.21579266] fitness = 3.8481376

3 [-0.03215599] fitness = 4.1338296

4 [-0.03215599] fitness = 4.1338296

5 [-0.21579266] fitness = 3.8481376

6 [-0.21579266] fitness = 3.8481376

7 [8.497307] fitness = -688.7135

8 [5.5503407] fitness = -482.7443

9 [-0.03215599] fitness = 4.1338296

Найкраще значення покоління 4.1338296

Нова популяція

0 [-0.03215599] fitness = 4.1338296

1 [-0.03215599] fitness = 4.1338296

2 [-0.03215599] fitness = 4.1338296

3 [-0.03215599] fitness = 4.1338296

4 [-0.03215599] fitness = 4.1338296

5 [-0.03215599] fitness = 4.1338296

6 [-0.03215599] fitness = 4.1338296

7 [-8.325739] fitness = -2910.8833

8 [-7.36382] fitness = -2167.6716

9 [-0.03215599] fitness = 4.1338296

Найкраще значення покоління 4.1338296

Нова популяція

0 [-0.03215599] fitness = 4.1338296

1 [-0.03215599] fitness = 4.1338296

2 [-0.03215599] fitness = 4.1338296

3 [-0.03215599] fitness = 4.1338296

4 [-0.03215599] fitness = 4.1338296

5 [-0.03215599] fitness = 4.1338296

6 [-0.03215599] fitness = 4.1338296

7 [-4.492905] fitness = -679.7659

8 [6.030199] fitness = -533.0409

9 [-0.03215599] fitness = 4.1338296

Найкраще значення покоління 4.1338296

**Результат:**

**[-0.03215599] fitness = 4.1338296**

Код програми

**GAEvolve.class**

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

**public** **float**[] best\_fitness;//best\_fitness[i] є найкращим значенням для і-го покоління даної еволюції

**public** GAIndividual best\_ind; //найкращий індивідум(х) даної еволюції

**public** GAEvolve(**int** generations, **int** pop\_size, **int** genome\_size, **int** xrate,**int** mrate, **float**[] min\_range, **float**[] max\_range){

// xrate: частота схрещування

// mrate: частота мутації

// xrate і mrate можуть змінюватися під час покоління генерації

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;

//if(i<11)

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

}

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

}

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

//для пошуку мінімуму

//float[] min = new float[]{5f};

//float[] max = new float[]{15f};

//для пошуку максимуму

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

**float**[] max = **new** **float**[]{10};

// GAEvolve.evolveAndMakeMFile("ga.m",1,300,400,2,70,20,min,max);

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

System.*out*.println("Результат:");

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

}

}

**GAIndividual.class**

**import** java.util.Random;

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

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

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

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

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

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

// створити випадкового індивідума довжиною gsize

// і-й ген повине бути в діапазоні між min\_range[i] і max\_range[i]

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];

}

///////////

//System.out.println(genome[0]);

evalFitness();// оцінити придатність цього нового індивідума

}

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

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

// min\_range[i] <= d[i] <= max\_range[i]

genome\_size = d.length;

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

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

genome[i] = d[i];

}

evalFitness();// оцінити придатність цієї нового індивідума

}

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

// rate це шанс кожного гена мутувати

**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];

// застосування точкової мутації

**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) {

// одноточковий кросинговер

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() {

fitness = (4 - 5 \* genome[0] - 26 \* genome[0] \* genome[0] + 2 \* genome[0]

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

}

}

**GAPopulation.Population**

**import** java.util.Random;

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

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

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

**public** GAIndividual[] ind;

**public** **int** best\_index; // індекс найкращого індивідума в масиві ind

// best\_fitness = ind[best\_index].fitness

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

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

// створення рандомної популяції кількісью pop\_size

// psize: довжина популяції

// gsize: довжина геному

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]+" ");}

//System.out.println();

evaluate();

}

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

// Створення популяції з такимиж індивідумами як в 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) {

//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++) {

// select to parents for cross-over:

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

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

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

}

// мутація:

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

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

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;

}

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

//оцінювання

**int** best = 0;

// індекс найкрощого індивідума

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

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

// сума придатності особин даної популяції

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

sum += ind[i].fitness;

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

best = i;

best\_fitness = ind[i].fitness;

}

}

best\_index = best;

}

**public** String toString() {

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

**return** s;

}

}

**Висновки:** виконавши лабораторну роботу я вивчив еволюційні оператори схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації. 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.