НУЛП, САПР, СПК		Тема	Оцінка:	Підпис:
КНМ-14	4	Використання генетичних алгоритмів з бітовим представленням хромосом		
Ласка Г. Л.				
№11341784				
Методи нечіткої логіки та еволюційні алгоритми			**Викладач:** Кривий Р. З.	

Мата: навчитися застосовувати генетичні алгоритми з побітовим представленням хромосом.

Хід роботи

Для виконання завдання була використана функція ga пакету MatLab, і окремо реалізовані функції для побітової мутації і побітового схрещування.

Цільові функції для пошуку мінімуму та максимуму:

```
function  [output_args] = FitnessFcn(
input_args )
% input_args = [x1]
% варіант 3
a = 10; b = -20; c = -40; d = 1;
x = input_args(1);

f = a + b*x + c*(x^2) + d*(x^3);
output_args = f;

end
```

```
function  [output_args] =
MaxFitnessFcn( input_args )

output_args =
(-1)*FitnessFcn( input_args );

end
```

Побітова мутація

```
function [ mutationChildren ] = MutationFcn( parents, options, nvars, ...
FitnessFcn, state, thisScore, thisPopulation )
% parents – номер особини в популяції, що мутує
% nvars – кількість змінних
% state – інформація про поточну популяцію
% thisScore – оцінки поточної популяції
% thisPopulation – поточна популяція

% маска мутації. змінює випадковий біт на протилежний
 mask = zeros(1, 6);
 mask(randi(6)) = 1;

 mutant = thisPopulation(parents, :)+10;
 for i=1:1:nvars
    dm = mutant(i);
    if dm > 63
       dm = de2bi(dm);
       dm = dm(1:6);          %відтинаємо лишні біти
    else
        dm = de2bi(dm, 6);
    end

    dm = bitxor(dm, mask);
    mutant(i) = bi2de(dm)-10;
 end

 mutationChildren = mutant;
end
```

Побітове схрещування

```
function [ xoverKids ] = CrossoverFcn( parents, options, nvars, FitnessFcn,
...
    unused,thisPopulation )
% parents - індекси батьків в поточній популяції, що беруть участь у
%           схрещуванні. вектор з парною кількістю елементів
% nvars   - кількість змінних (генів)
% unused  - вектор-стовбець із оцінкою кожної особини
% thisPopulation - поточна популяція (матриця)

ret = zeros(length(parents)/2, nvars);
for i = 1:2:length(parents)
    p1 = thisPopulation(i, :);
    p2 = thisPopulation(i+1, :);

    c = thisPopulation(i, :);
    for j = 1:1:nvars
        p1_bit = toBitArr(p1(j)+10);
        p2_bit = toBitArr(p2(j)+10);

        c_bit = [p1_bit(1:3), p2_bit(4:6)];
        c(j) = bi2de(c_bit)-10;
    end
    ret((i+1)/2,:) = c;
end;

xoverKids = ret;

end

function [bitVal] = toBitArr(decVal)
    if decVal > 63
        dm = de2bi(decVal);
        dm = dm(1:6);          %відтинаємо лишні біти
    else
        dm = de2bi(decVal, 6);
    end
    bitVal = dm;
end
```

Результати кожної ітерації зберігаються в глобальну змінну, після чого виводяться на екран.

Функція для збереженя везультатів кожної ітерації

```
function [ state,options,optchanged ] = OutputFcn( options,state,flag )
 global RET;
 ci = state.Generation;
 RET.generation = ci;
 key = strcat('s',num2str(ci));
 RET.population(:).(key) = state.Population;
 RET.fvals(:).(key) = state.Score;
 optchanged  = false;
end
```

Результати виконання:

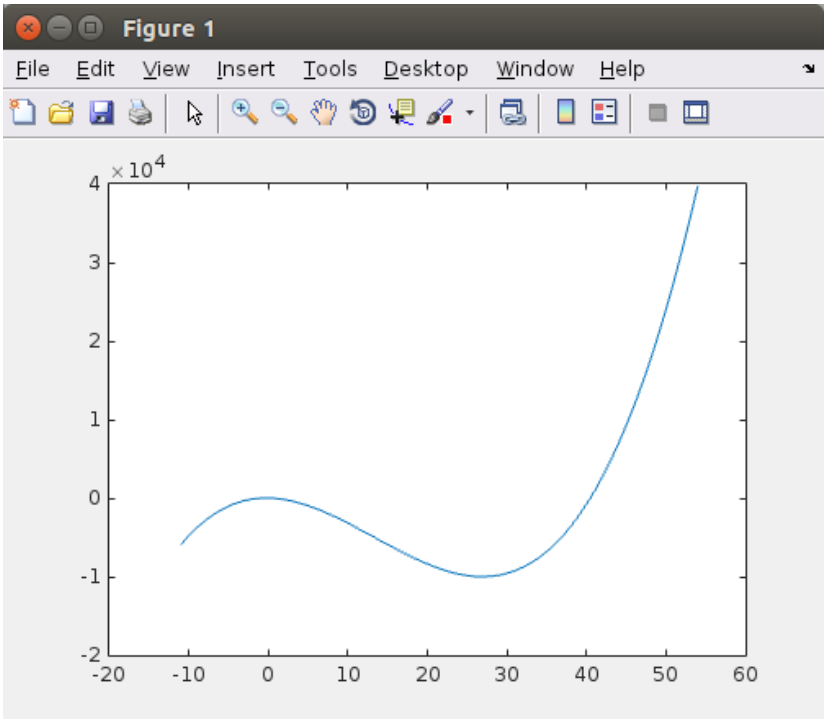

Рис. 1. Графік функції

```
Command Window
New to MATLAB? See resources for Getting Started.
#Пошук мінімуму
Optimization terminated: average change in the fitness value less than options.TolFun.
Початкова популяція:
[ 42 ]=>2698    [ 47 ]=>14533   [ -2 ]=>-118    [ 48 ]=>17482   [ 30 ]=>-9590
Покоління 1:
[ 25 ]=>-9865   [ 30 ]=>-9590   [  7 ]=>-1747   [ -4 ]=>-614    [ 50 ]=>24010
Покоління 2:
[ 25 ]=>-9865   [ 30 ]=>-9590   [ 23 ]=>-9443   [-10 ]=>-4790   [ 33 ]=>-8273
Результат:
[ 27 ]=>-10007  [ 27 ]=>-10007  [ 27 ]=>-10007  [ 27 ]=>-10007  [ 27 ]=>-10007
f(27) = -10007
```

```
Command Window
New to MATLAB? See resources for Getting Started.
#Пошук максимуму
Optimization terminated: average change in the fitness value less than options.TolFun.
Початкова популяція:
[ 42 ]=>2698    [ 47 ]=>14533   [ -2 ]=>-118    [ 48 ]=>17482   [ 30 ]=>-9590
Покоління 1:
[ 51 ]=>27601   [ 51 ]=>27601   [ 48 ]=>17482   [ 47 ]=>14533   [ 50 ]=>24010
Покоління 2:
[ 51 ]=>27601   [ 51 ]=>27601   [ 51 ]=>27601   [ 50 ]=>24010   [ 51 ]=>27601
Результат:
[ 53 ]=>35467   [ 53 ]=>35467   [ 53 ]=>35467   [ 53 ]=>35467   [ 53 ]=>35467
f(53) = 35467
```

Висновок: якщо вхідні дані цілі числа, то побітове представлення хромосоми є хорошим варіантом для зберігання цієї умови під час виконання генетичного алгоритму.