НУЛП, САПР, СПК		Тема	Оцінка:	Підпис:
КНСП-11	4	Використання генетичних алгоритмів з бітовим представленням хромосом		
Янчук Н. Ю.				
Варіант 10				
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]
% в а р і а н т  10
a = 26; b = -86; c = -59; d = 3;
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 - н о м е р  о с о б и н и  в  п о п у л я ц і ї, щ о  м у т у є
% nvars - к і л ь к і с т ь  з м і н н и х
% state - і н ф о р м а ц і я  п р о  п о т о ч н у  п о п у л я ц і ю
% thisScore - о ц і н к и  п о т о ч н о ї  п о п у л я ц і ї
% thisPopulation - п о т о ч н а  п о п у л я ц і я

% м а с к а  м у т а ц і ї.  з м і н ю є  в и п а д к о в и й  б і т  н а
п р о т и л е ж н и й
 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);          %в і д т и н а є м о  л и ш н і  б і т и
    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 - і н д е к с и  б а т ь к і в  в  п о т о ч н і й  п о п у л я ц і ї, щ о
б е р у т ь  у ч а с т ь  у
%          с х р е щ у в а н н і. в е к т о р  з  п а р н о ю  к і л ь к і с т ю
е л е м е н т і в
% nvars   - к і л ь к і с т ь  з м і н н и х  (г е н і в)
% unused  - в е к т о р - с т о в б е ц ь  і з  о ц і н к о ю  к о ж н о ї  о с о б и н и
% thisPopulation - п о т о ч н а  п о п у л я ц і я  (м а т р и ц я)

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);          %в і д т и н а є м о  л и ш н і  б і т и
```

```
    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:

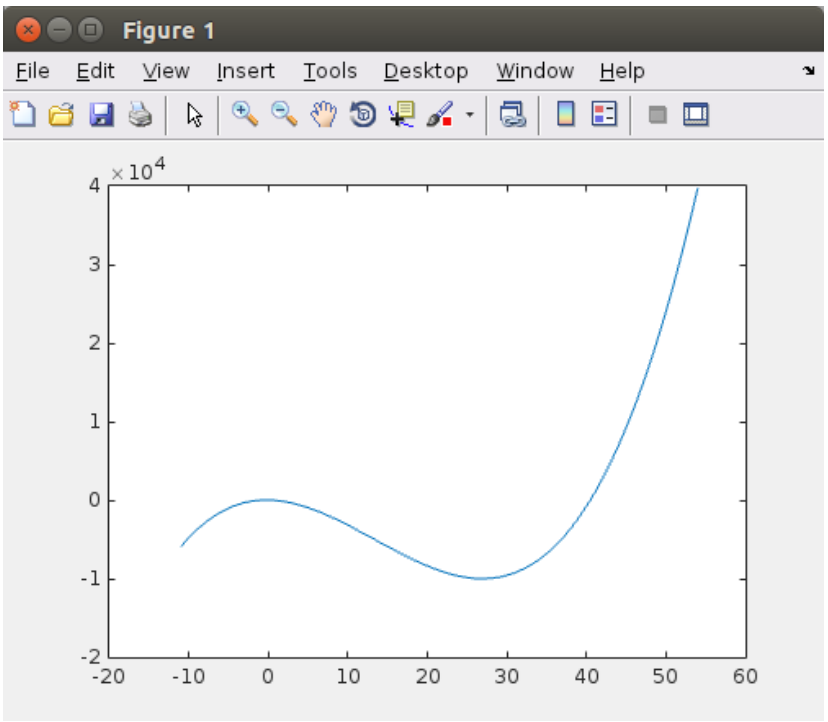

Рис. 1. Графік функції

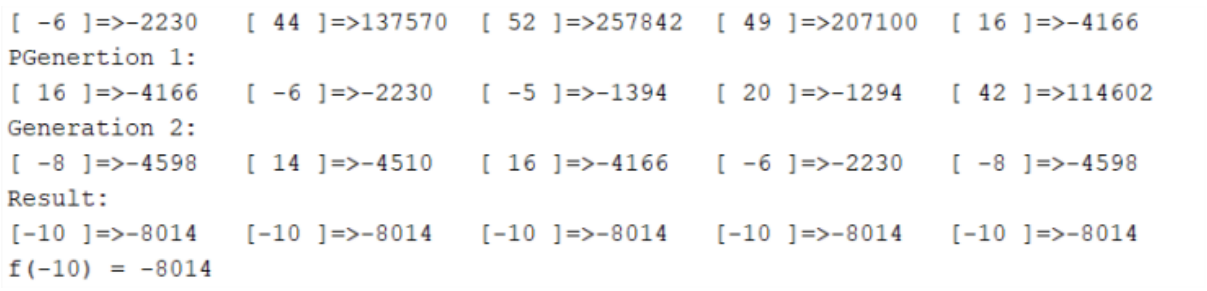

Рис. 2. Результат пошуку мінімуму

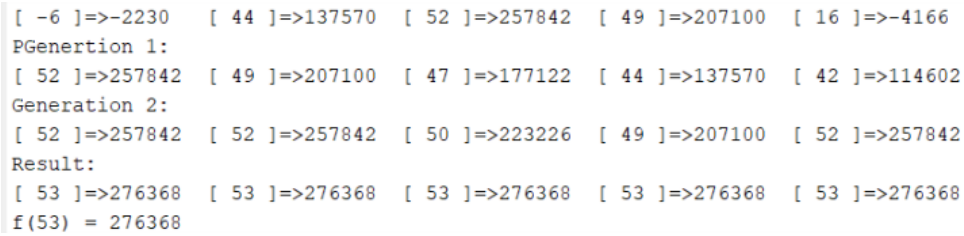

Рис. 3. Результат пошуку максимуму

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