

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

*Кафедра “Системи автоматизованого проектування”*



**Звіт**

до лабораторної роботи №3  
з курсу: «Методи нечіткої логіки та еволюційні алгоритми при  
автоматизованому проектуванні»  
на тему:  
**«Комбінаторна оптимізація  
за допомогою еволюційних  
методів»**

Виконав:

ст. гр. КНСП-11

Гуль О. В.

Перевірив:

Кривий Р. З.

**Мета:** ознайомитися з основними теоретичними відомостями, вивчити еволюційні оператори схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації.

### **Теоретичні відомості**

При використанні методів еволюційного пошуку для розв’язку задач комбінаторної оптимізації, як правило, застосовуються негомологічні числові хромосоми, тобто такі хромосоми, гени яких можуть приймати значення в заданому інтервалі. При цьому інтервал однаковий для всіх генів, але в хромосомі не може бути двох генів з однаковим значенням.

Комбінаторні задачі оперують із дискретними структурами або розміщенням об’єктів, незначні зміни яких часто викликають стрибкоподібну зміну показників якості (фітнесс- функції). Традиційні оператори еволюційні оператори, що генерують нових нащадків, не можуть бути застосовані при використанні негомологічних хромосом, оскільки внаслідок виконання таких операторів генеруються нащадки, що містять однакові гени і тому не можуть бути інтерпретовані при розв’язку комбінаторної задачі. Тому для розв’язку задач комбінаторної оптимізації були розроблені спеціальні генетичні оператори, що не створюють неприпустимих рішень.

### **Завдання**

Розробити за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера. Параметри еволюційного методу обрати з таблиці 1 відповідно до варіанту.

№	Еволюційні оператори	
	Схрещування	Мутація
3	Циклове	Мутація золотого перетину

### **Хід роботи**

Для виконання завдання була використана функція `ga` пакету Matlab, і реалізовано власні функції мутації та схрещування, згідно з варіантом.

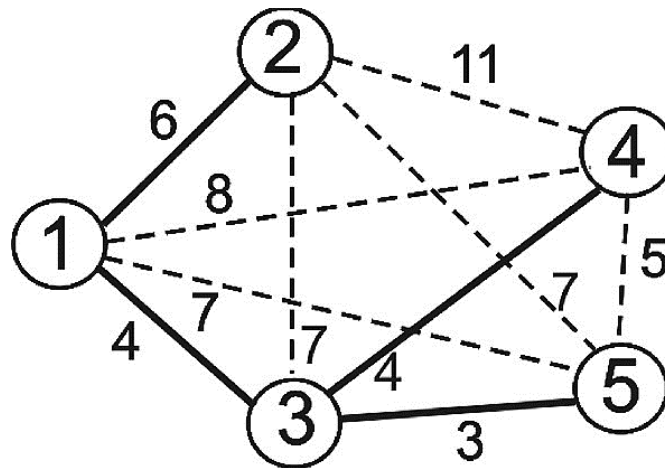


Рис. 1. Графічне представлення задачі комівояжера

### Функція для схрещування

```
function [xoverKids] = CrossoverFcn( parents, options, nvars, FitnessFcn, ...
    unused, thisPopulation )
%% Реалізація функції для схрещування потомків
(циклове схрещування)
% 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)-1
    p1 = thisPopulation(parents(i), :);
    p2 = thisPopulation(parents(i+1), :);

    % генеруємо цикл
    t = randi(nvars); % початок циклу (індекс)
    cycle = zeros(1, nvars);
    for j = 1:1:nvars
        cycle(1, j) = t;
        nv = p2(t);
        t = find(p1==nv);
        if (p1(cycle(1,1)) == nv)
            break; % цикл замкнувся
        end;
    end;

    % елементи, що не попали в цикл успадковуються
    від іншого батька
    child = p2;
    for j = 1:1:nvars
        if (cycle(1, j) ~= 0)
            child(1, cycle(1, j)) = p1(cycle(1, j));
        end;
    end;

    ret(i/2, :) = child;
end;
```

```

        end;
    end;
    ret((i+1)/2, :) = child;
end;
xoverKids = ret;

end

```

## Функція мутації

```

function [ mutationChildren ] = MutationFcn( parents, options, nvars, ...
FitnessFcn, state, thisScore, thisPopulation )
% Проводить мутацію методом золотого січення

% parents – номер особини в популяції, що мутує
% nvars – кількість змінних
% state – інформація про поточну популяцію
% thisScore – оцінки поточної популяції
% thisPopulation – поточна популяція

k = 0.62;          % k=62%
t = ceil(k*nvars); % точка розриву

mutant = thisPopulation(parents, :);
d = mutant(t);
d1 = mutant(t+1);
mutant(t) = d1;
mutant(t+1) = d;

mutationChildren = mutant;

end

```

## Точка запуску програми

```

%% точка запуску програми
%% Варіант 3
%% Схрещування: цеклове
%% Мутація: золотого перетину

% Задача: знайти найвигідніший маршрут,
% який проходить через кожне місто по одному разу
% одною особою є послідовність обходу міст
% значення генів не можуть повторюватися
% а довжина хромосоми рівна кількості міст
% для матриці з 5 міст можливо всього 5! = 120 різних
% способів обходу

```

```

% тому розмір популяції візьмемо рівний кількості
міст (5)

startPopulation = [
    1, 2, 3, 4, 5;
    2, 3, 4, 5, 1;
    3, 4, 5, 1, 2;
    4, 5, 1, 2, 3;
    5, 1, 2, 3, 4
];

options = gaoptimset(...
    'EliteCount', 0, ...
    'PopulationSize', 5, ...
    'InitialPopulation', startPopulation, ...
    'MutationFcn', @MutationFcn, ...
    'CrossoverFcn', @CrossoverFcn, ...
    'TimeLimit', 3 ...
);
[x, fval, exitflag, output, population, scores] = ga(@optim_function, 5, options);

disp('Найращей потомок:'); disp(x);
fprintf('f(x) = %d\n', fval);
disp('Остання популяція:');
for i=1:1:5
    for j=1:1:5
        fprintf(' %t%d', population(i, j));
    end;
    fprintf(' %t=>%t%d\n', scores(i));
end;

```

## Результат виконання

```

>> main
Optimization terminated: average change in the fitness value less than options.TolFun.
Найращей потомок:
    2    1    3    5    4

f(x) = 18
Остання популяція:
    2    1    3    5    4    =>    18
    2    1    3    5    4    =>    18
    2    1    3    5    4    =>    18
    2    1    3    5    4    =>    18
    2    1    3    4    5    =>    19
>>

```

New to MATLAB? See resources for [Getting Started](#).

```
>> main
Optimization terminated: average change in the fitness value less than options.TolFun.
Найращей потомок:
    1     2     3     5     4

f(x) = 21
Остання популяція:
    1     2     3     4     5    =>    22
    1     2     3     5     4    =>    21
    1     2     3     4     5    =>    22
    1     2     3     5     4    =>    21
    1     2     3     4     5    =>    22
fx >>
```

New to MATLAB? See resources for [Getting Started](#).

```
>> main
Optimization terminated: average change in the fitness value less than options.TolFun.
Найращей потомок:
    1     2     3     5     4

f(x) = 21
Остання популяція:
    1     2     3     4     5    =>    22
    1     2     3     5     4    =>    21
    1     2     3     5     4    =>    21
    1     2     3     5     4    =>    21
    1     2     3     5     4    =>    21
fx >>
```

## Висновок

На відміну від класичних методів розв'язання задачі комівояжера, використовуючи генетичні алгоритми, ми зразу отримуємо декілька оптимальних варіантів. Але такий підхід не гарантує, що результат є найоптимальніший.