|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САПР** | | Тема | оцінка | підпис |
| КНм-14 | 5 | ПРОГРАМУВАННЯ ГА ДЛЯ ЗАДАЧІ КОМІВОЯЖЕРА (TSP) |  |  |
| Малоїд Т.В. | |
| № залікової: | |
| **Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні** | | Викладач: | |
| к.т.н. Кривий Р.З. | |

**Завдання:** Розробити за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера (TSP)

**Проблема**. У вас є безліч міст (представлені у вигляді точок на площині з X і Y координати). Мета полягає в тому, щоб знайти найкоротший маршрут, який відвідує кожне місто рівно один раз, повертаючись в кінці своєї відправної точки.

**Варіант 5.**

1(5). Турнірна селекція

**Теоретичні відомості:**

При турнірної селекції формується випадкове підмножина з елементів популяції у тому числі вибирається один елемент з найбільшим значенням цільової функції. Турнірний відбір реалізує n турнірів, аби вибрати n особин. Кожен турнір побудований на вибірці k елементів з популяції, і вибору кращої особини у тому числі. Найбільш поширений турнірний відбір з k=2.

У середовищі Matlab, турнірна селекція вже реалізована у фреймворку ga для генетичних алгоритмів, та задається таким чином:

options = gaoptimset(...

'SelectionFcn', {@selectiontournament,2}, ...

);

**Виконання лабораторного завдання:**

Основний код програми: lab5\_maloid.m

%% Варіант 5

clear, clc, close all

dbstop if error

%% Масив відстаней між містами

n = 20; % кількість міст

coord = randi(100, 2, n); % випадкові координати міст

distance = zeros(n, n);

for i=1:1:n

for j=1:1:n

distance(i,j) = int32(pdist([coord(1,i),coord(2,i);coord(1,j),coord(2,j),],'euclidean'));

end

end

disp('Coordinates')

disp(coord)

disp('Array of distance')

disp(distance)

%% Стартова популяція

populationSize = 30;

startPopulation = zeros(populationSize,n);

aa = randsample(n,n);

for i=1:1:populationSize

startPopulation (i,:) = randsample(n,n);

end

disp('StartPopulation')

disp(startPopulation)

%% Параметри ГА

options = gaoptimset(...

'EliteCount', populationSize/2, ...

'PopulationSize', populationSize, ...

'InitialPopulation', startPopulation, ...

'MutationFcn', @MutationFcn, ...

'CrossoverFcn', @CrossoverFcn, ...

'SelectionFcn', {@selectiontournament,2}, ...

'PlotFcns', {@gaplotbestf} ...

);

%% Запуск методу

[xx,fval,exitflag,output,population,scores] = ga(@(x)optim\_function(x, distance), n, options);

%% Відображення результатів

disp('Best child:');

disp(xx);

fprintf('f(x) = %d\n', fval);

figure

plot(coord(1,:),coord(2,:),'bo');

grid on

hold on

for i=1:1:n

if (i==n)

plot([coord(1,xx(i)),coord(1,xx(1))],[coord(2,xx(i)),coord(2,xx(1))],'r--')

text((coord(1,xx(i))+coord(1,xx(1)))/2,(coord(2,xx(i))+coord(2,xx(1)))/2, num2str(distance(xx(i),xx(1))))

else

plot([coord(1,xx(i)),coord(1,xx(i+1))],[coord(2,xx(i)),coord(2,xx(i+1))])

text((coord(1,xx(i))+coord(1,xx(i+1)))/2,(coord(2,xx(i))+coord(2,xx(i+1)))/2, num2str(distance(xx(i),xx(i+1))))

end

text(coord(1,xx(i))+2,coord(2,xx(i))+2,num2str(xx(i)))

end

disp('Last population:');

for i=1:1:n

fprintf('\t%d', population(i,:));

fprintf('\t=>\t%d\n', scores(i));

end;

Функції мутації та кроссоверу аналогічні функціям 3 лабораторної роботи:

Схрещення: двохточечне впорядковуюче.

Код CrossoverFcn.m

%% Двохточечне впорядковуюче схрещення (вар5)

function [xoverKids] = CrossoverFcn( parents, nvars, thisPopulation )

% parents - індекси батьків в поточній популяції, що беруть участь у схрещуванні.

% nvars - кількість змінних (генів)

% thisPopulation - поточна популяція (матриця)

result = zeros(length(parents)/2, nvars);

for i = 1:2:length(parents)-1

parent1 = thisPopulation(parents(i), :);

parent2 = thisPopulation(parents(i+1), :);

separator = randi([2, nvars-1],1,1); %випадковий індекс умовного поділу

gene1 = randi([1, separator-1],1,1); %випадковий перший ген

gene2 = randi([separator+1, nvars],1,1); %випадковий другий ген

child = zeros(1, nvars); %ініціалізація нащадка

child(1,1:gene1) = parent1(1:gene1); %копіювання генів зліва від 1го гена

child(1,gene2:nvars) = parent1(gene2:nvars);%копіювання генів справа від 2го гена

nextGene = gene1+1; %наступний ген після 1го

inarr = ismember(parent2(1,:), child); %перевірка наявності генів нащадка в другого батька

for j = 1:1:nvars

if (inarr(j) == 0)

child(1, nextGene) = parent2(j); %копіювання гену нащадку

nextGene = nextGene + 1;

end

end

result((i+1)/2,:) = child;

end;

xoverKids = result;

end

%% Кінець

Мутація: інвертування із зсувом.

Код MutationFcn.m

%% Мутація інвертування із зсувом

function [ mutationChildren ] = MutationFcn( parents, nvars, thisPopulation)

% parents - номер особини в популяції, що мутує

% nvars - кількість змінних

% thisPopulation - поточна популяція

mutant = thisPopulation(parents, :); %особина, що мутує

offset = randi(nvars); %випадковий крок зміщення

mutationChildren = circshift(mutant',offset)'; %зміщення вправо на крок

end

%% Кінець

Оптимізуюча функція:

Обчислення довжини шляху, враховуючи повернення в місто з якого починається шлях.

Код optim\_function.m

%% Оптимізуюча функція

function [ result ] = optim\_function( way, array)

sum = 0;

for i = 1:1:length(way)-1

sum = sum + array(way(i), way(i+1));

end

result = sum + array(way(i+1), way(1));

end

**Результати виконання:**

Прогін 1 (10 міст)

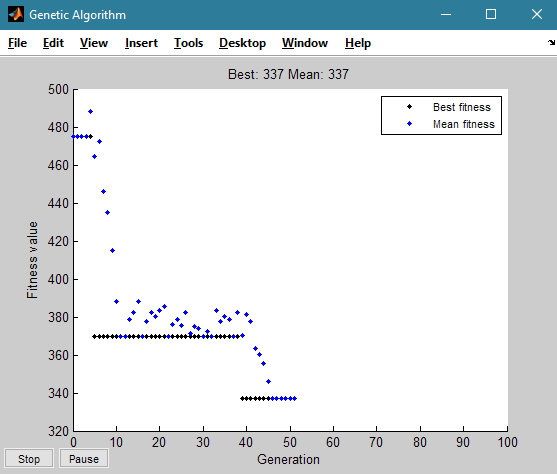


Рис.1 Залежність фітнес функції від генерації

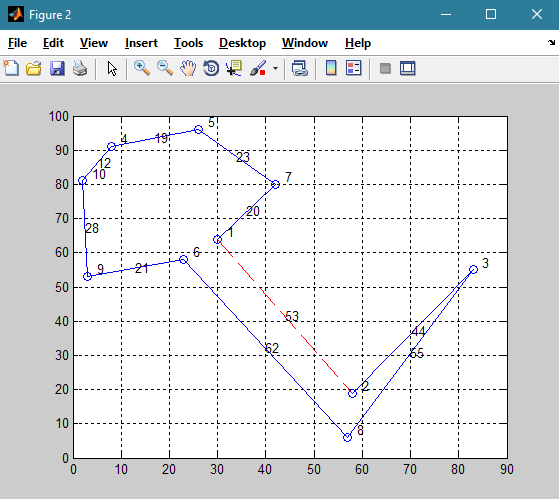
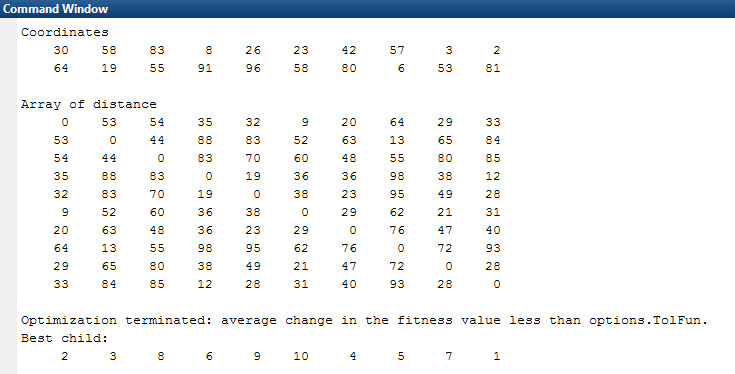


Рис.2 Оптимальний маршрут



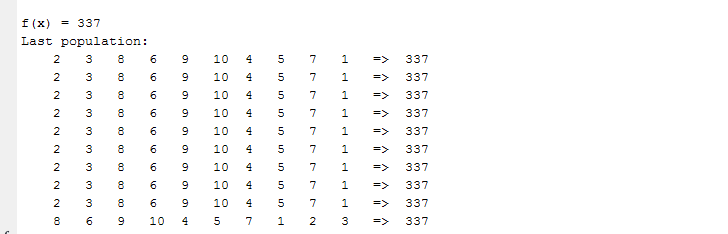


Рис.3 Вивід програми

Прогін 2 (15 міст)

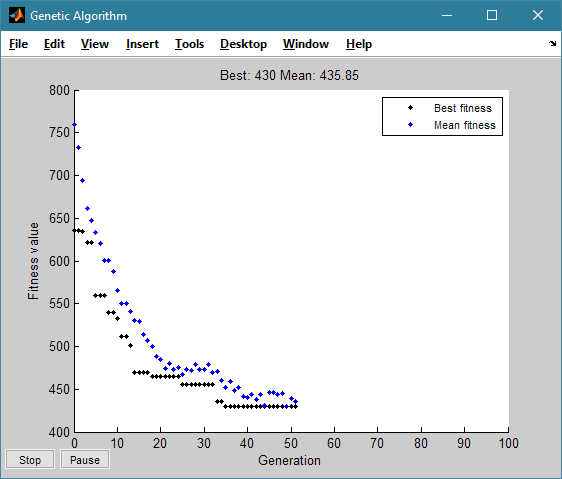


Рис.4 Залежність фітнес функції від генерації

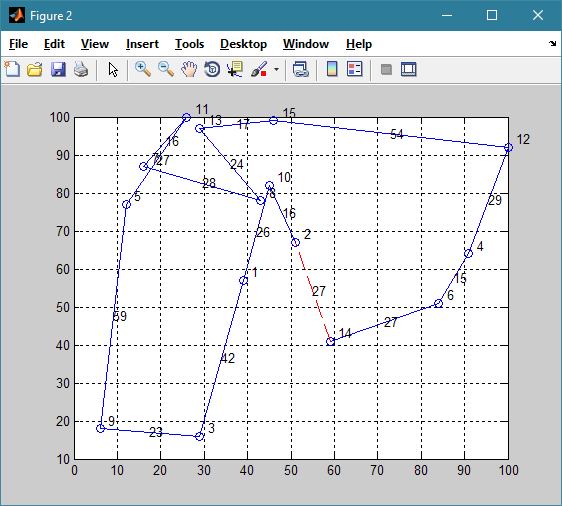
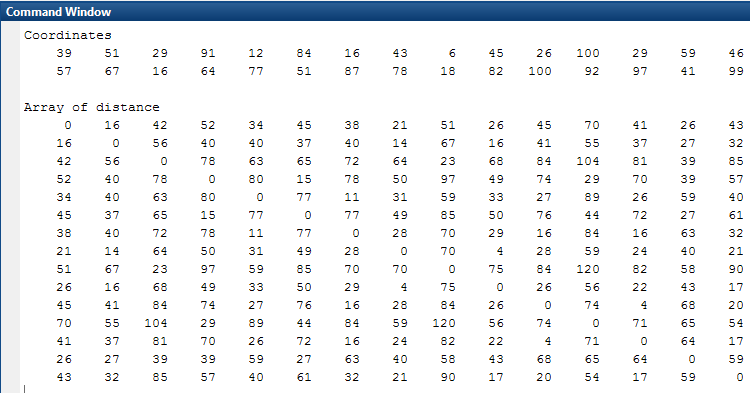
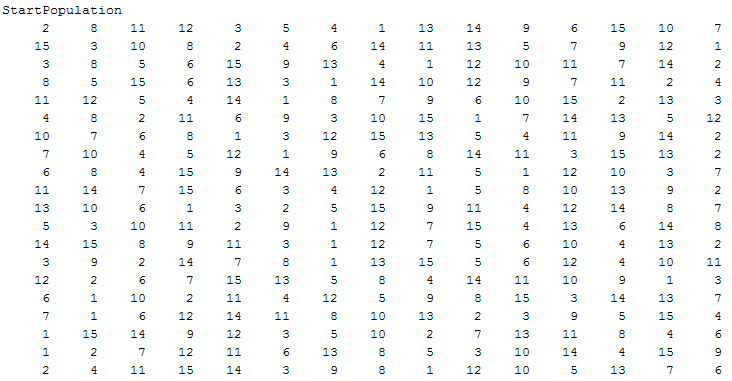


Рис.5 Оптимальний маршрут





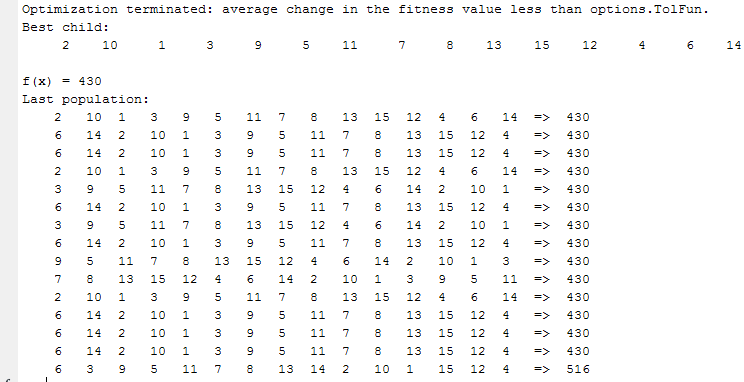


Рис.6 Вивід програми

Прогін 3 (20 міст)

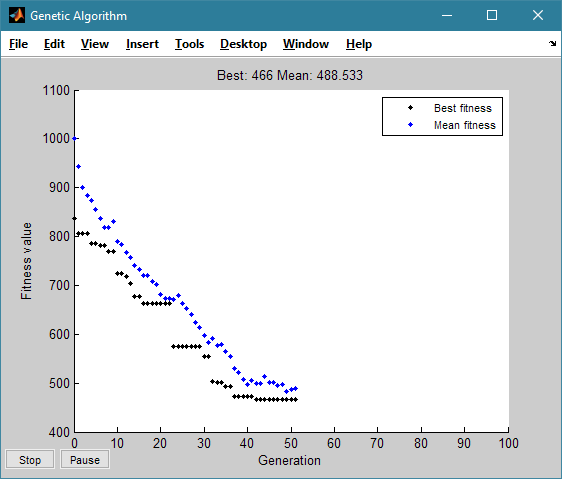


Рис.7 Залежність фітнес функції від генерації

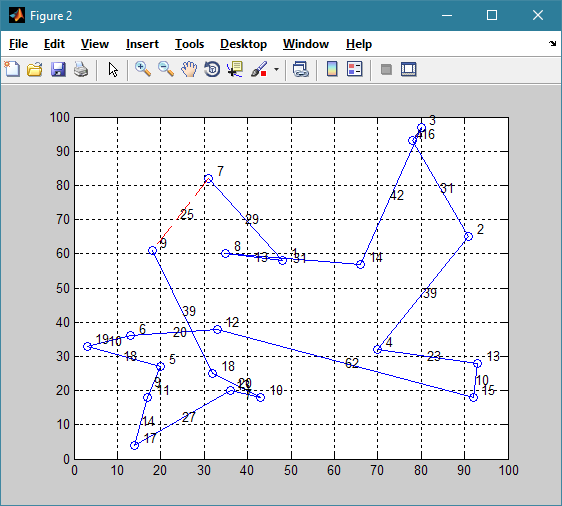
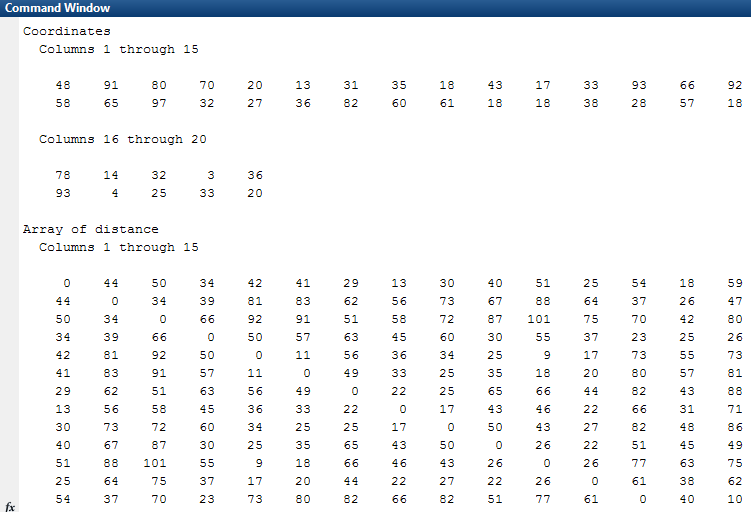
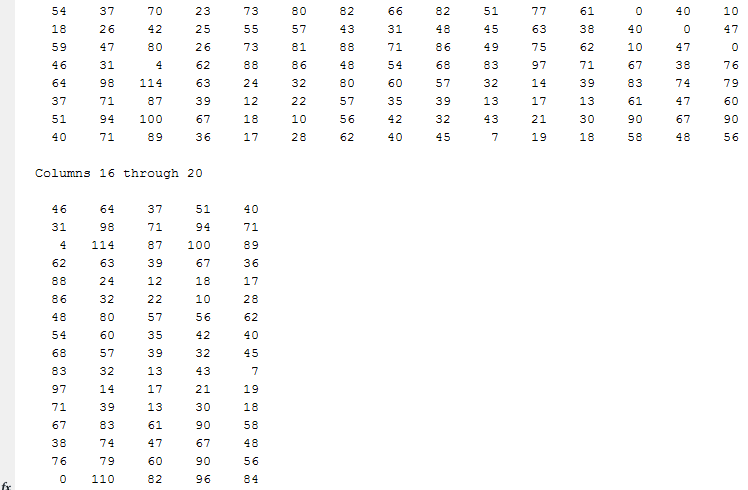
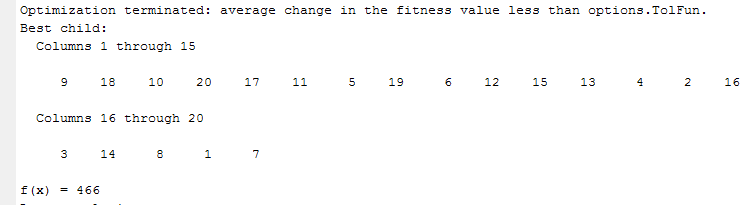


Рис.8 Оптимальний маршрут







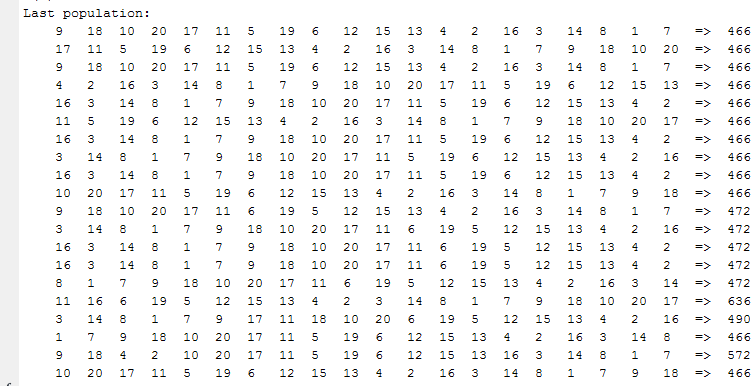


Рис.9 Вивід програми

**Висновки.**

Виконуючи лабораторну роботу я ознайомився з основними теоретичними відомостями та вивчив еволюційні оператори селекції, що використовуються при розв’язуванні задач комбінаторної оптимізації.

Також я розробив за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера, використовуючи функцію селекції згідно індивідуального варіанту.