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

**Завдання:** Розробити за допомогою пакету Matlab програмне забезпечення для

вирішення задачі комівояжера.

В - 5





Код програми

**traveling\_salesman\_problem.m**

cities = 10;%кількість міст

locations = zeros(cities,2);

mas = zeros(cities,2);

x=cell(1,1);

for n = 1: cities%ненерування розташування міст

locations(n,1) = randi([0 20],1,1);

mas(1,n) = locations(n,1);

locations(n,2) = randi([0 20],1,1);

end

x{1}=mas(1,:);

plot(locations(:,1),locations(:,2),'bo')

distances = zeros(cities);

for count1=1:cities,%знаходження відстані між містами

for count2=1:count1,

x1 = locations(count1,1);

y1 = locations(count1,2);

x2 = locations(count2,1);

y2 = locations(count2,2);

distances(count1,count2)=sqrt((x1-x2)^2+(y1-y2)^2);

distances(count2,count1)=distances(count1,count2);

end;

end;

FitnessFcn = @(x) traveling\_fitness(x,distances);

my\_plot = @(options,state,flag) traveling\_s\_plot(options, ...

state,flag,locations);

options = gaoptimset('CreationFcn',@create\_permutations, ...

'CrossoverFcn',@crossover\_two\_point\_order, ...

'MutationFcn',@ivert\_with\_landslip, ...

'PlotFcn', my\_plot);

[x,fval,reason,output] = ga(FitnessFcn,cities,options)

displayEndOfDemoMessage(mfilename)

**create\_permutations.m**

function pop = create\_permutations(NVARS,FitnessFcn,options)

%функція для створення популяції

totalPopulationSize = sum(options.PopulationSize);

n = NVARS;

pop = cell(totalPopulationSize,1);

for i = 1:totalPopulationSize

pop{i} = randperm(n);

end

**crossover\_two\_point\_order.m**

function xoverKids = crossover\_two\_point\_order(parents,options,NVARS, ...

FitnessFcn,thisScore,thisPopulation)

%двохточкове впорядковуюче схрещування

nKids = length(parents)/2;

xoverKids = cell(nKids,1); % Normally zeros(nKids,NVARS);

index = 1;

for i=1:nKids

%вибір бітьків

parent1 = thisPopulation{parents(index)};

if(parents(index) == length(thisPopulation))

parent2 = thisPopulation{parents(index)-1};

else

parent2 = thisPopulation{parents(index)+1};

end

%генерування дві точки схрещування

p1 = randi([1 length(parent1)],1,1);

p2 = randi([p1 length(parent1)],1,1);

child = parent1;

child(p1:p2) = 0;

n = 0;

%копіювання в нащадок між точками схрещування генів 2

%батька в упорядкованому вигляді зліва направо

for j = 1:length(parent2)

if all(child ~= parent2(j))

child(p1+n) = parent2(j);

n = n+1;

end

end

xoverKids{i} = child;

index = index + 2;

end

**ivert\_with\_landslip.m**

function mutationChildren = ivert\_with\_landslip(parents ,options,NVARS, ...

FitnessFcn, state, thisScore,thisPopulation,mutationRate)

%інвертування з зсувом

mutationChildren = cell(length(parents),1);

for i=1:length(parents)

parent = thisPopulation{parents(i)};

p = ceil(length(parent) \* rand(1,2));%генерується вектор з 2 чисел,

%1ше число позиція гена який преміщатиметься на позицію 2го числа

child = parent;

%зсув елементів вправо по циклу

if(p(2)>p(1))

k1 = parent(p(2));

for j = (p(2)+1): length(parent)

k2 = parent(j);

child(j)=k1;

k1=k2;

end

for j = 1: p(1)

k2 = parent(j);

child(j)=k1;

k1=k2;

end

child(p(2)) = k1;

end

if(p(2)<p(1))

k1 = parent(p(1));

for j = p(2) : p(1)

k2 = parent(j);

child(j)=k1;

k1=k2;

end

end

child(p(2)) = parent(p(1));

mutationChildren{i} = child;

end

end

**traveling\_fitness.m**

function scores = traveling\_fitness(x,distances)

%функція для знаходження довжини шляху

scores = zeros(size(x,1),1);

for j = 1:size(x,1)

p = x{j};

f = distances(p(end),p(1));

for i = 2:length(p)

f = f + distances(p(i-1),p(i));

end

scores(j) = f;

end

end

**traveling\_s\_plot.m**

function state = traveling\_s\_plot(options,state,flag,locations)

%Функція для виведення графіків

[unused,i] = min(state.Score);

genotype = state.Population{i};

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

hold on;

plot(locations(genotype,1),locations(genotype,2));

hold off

end

Результати виконання програми

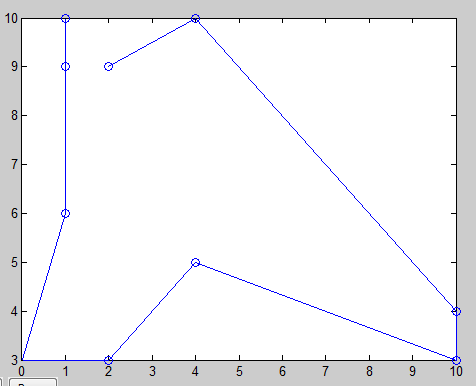


Рис.1. Рішення задачі комівояжера для 10 міст.

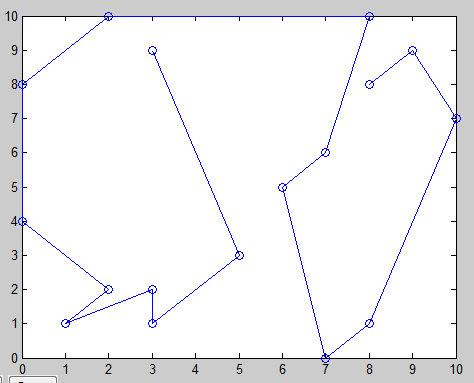


Рис.2. Рішення задачі комівояжера для 20 міст.

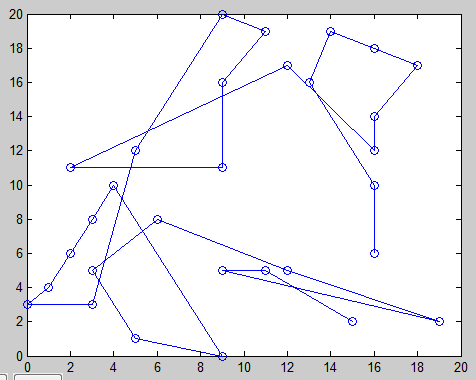


Рис.3. Рішення задачі комівояжера для 30 міст.

**Висновки:** виконавши лабораторну роботу я вивчив еволюційні оператори схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації. Реалізував за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера , в результаті програма дає результати близькі до оптимальних при кількості міст до 20, при більшій кількості шлях є неоптимальним.