**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 - 9





Код програми

**traveling\_salesman\_problem.m**

cities = 20;%кількість міст

locations = zeros(cities,2);

mas = zeros(cities,2);

x=cell(1,1);

for n = 1: cities

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);

c = clock;

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

'CrossoverFcn',@crossover\_pmx, ...

'MutationFcn',@invert\_classic, ...

'PlotFcn', my\_plot, ...

'Generations',500,'PopulationSize',100);

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

lastPopul = population(length(population))

c2 = clock;

time=c-c2%знаходження часу роботи оптимізації

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\_pmx.m**

function xoverKids = crossover\_pmx(parents,options,NVARS, ...

FitnessFcn,thisScore,thisPopulation)

%схрещування pmx

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;

for j = p1:p2

item1 = parent1(j);

item2 = parent2(j);

for k = 1:length(parent2)

if(child(k) == item1) pos1 = k;

else if (child(k) == item2) pos2 = k;

end

end

end

if(item1 ~= item2)

child(pos1)=item2;

child(pos2)=item1;

end

end

xoverKids{i} = child;

index = index + 2;

end

**invert\_classic.m**

function mutationChildren = invert\_classic(parents ,options,NVARS, ...

FitnessFcn, state, thisScore,thisPopulation,mutationRate)

%класичне інвертування

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

for i=1:length(parents)

parent = thisPopulation{parents(i)};

%вибір двох порядкових номер генів між якми роводитимесь інвертування

p1 = ceil((length(parent) -1) \* rand);

p2 = p1 + ceil((length(parent) - p1- 1) \* rand);

child = parent;

child(p1:p2) = fliplr(child(p1:p2));

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

При запуску для 10 міст з координатами(рис.1) :

Locations10=[0 14;7 0; 13 12;18 5; 18 9;4 16; 6 11;16 12;13 12;6 19]

При запуску для 20 міст з координатами(рис.2) :

Locations20=[14,2;16,0;11,8;5,18;4,7;6,8;7,5;0,19;13,8;6,6;20,15;8,19;17,6;0,5;20,18;0,13;6,10;3,18;12,11;6,18;]

При запуску для 30 міст з координатами(рис.3) :

Locations30=[5,17;16,17;15,3;11,3;0,14;14,3;20,4;20,15;3,9;12,4;15,13;8,2;13,0;11,20;12,6;12,1;9,10;16,3;13,20;15,5;20,7;2,12;5,13;19,18;0,8;16,14;14,9;9,12;2,10;2,7;]

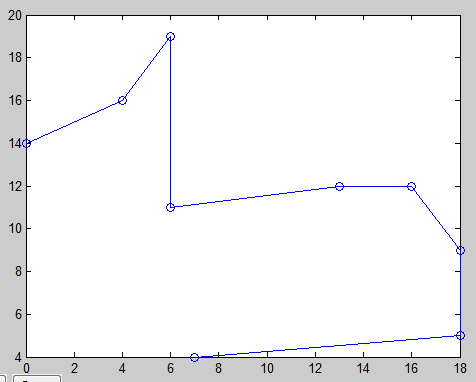


Рис.1. Рішення задачі комівояжера для 10 міст.

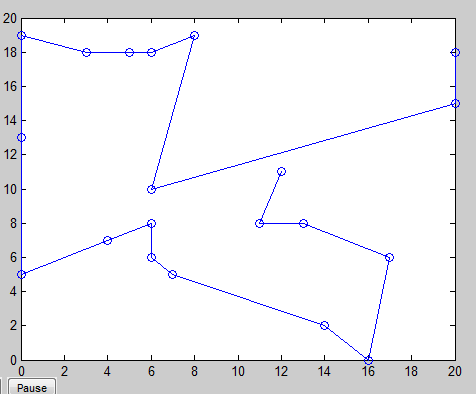


Рис.2. Рішення задачі комівояжера для 20 міст.

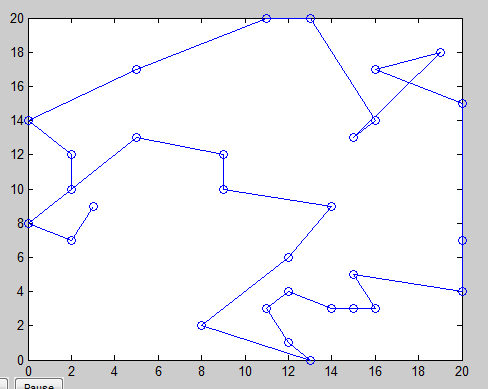


Рис.3. Рішення задачі комівояжера для 30 міст.

Таблиця порівняння залежності кількості міст і популяції

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кількість міст | 10 | | | 20 | | | 30 | | |
| Популяція | 20 | 50 | 100 | 20 | 50 | 100 | 20 | 50 | 100 |
| Час виконання,c | 1.662 | 1.726 | 1.68 | 1.22 | 1.712 | 2.14 | 1.79 | 2.5 | 2.8 |
| Мін. довжинна | 58.41 | 58.41 | 58.41 | 102.34 | 99.27 | 92.1 | 166.27 | 141.96 | 124.25 |
| Кількість  ітерацій | 51 | 51 | 51 | 61 | 72 | 69 | 85 | 98 | 83 |
| Кільк. оцінок функції | 1040 | 2600 | 5200 | 1240 | 3650 | 7000 | 1720 | 4950 | 8400 |

**Висновки:** виконавши лабораторну роботу я вивчив еволюційні оператори схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації. Реалізувала за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера з pmx схрещуванням і класичним інвертуванням. В результаті програма коректно працює для кількість міст до 15, з більшою кількістю міст шлях комівояжера не оптимальний.