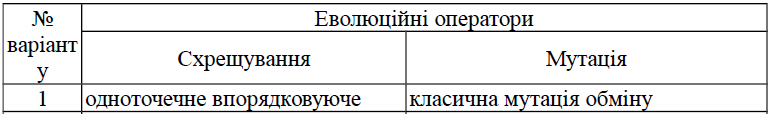
**Мета роботи**: Ознайомитися з основними теоретичними відомостями, вивчити еволюційні оператори схрещування та мутації, що використовуються при розв’язуванні задач комбінаторної оптимізації.

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

вирішення задачі комівояжера.

В – 1



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\_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);

child = parent1;

n = 0;

%копіювання в нащадок між точками схрещування генів 2

%батька в упорядкованому вигляді зліва направо

for j = p1+1:length(parent2)

if all(child ~= parent2(j))

child(p1+1+n) = parent2(j);

n = n+1;

end

end

xoverKids{i} = child;

index = index + 2;

end

**invert\_classic.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));

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

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

child = parent;

%заміна генів р1 р2

tmp = child(p2);

child(p2) = child(p1);

child(p1) = tmp;

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) :

Location = [17 19; 2 19; 13 2; 5 11; 20 20; 3 20; 20 10; 16 2; 8 19; 16 20;]

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

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

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

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

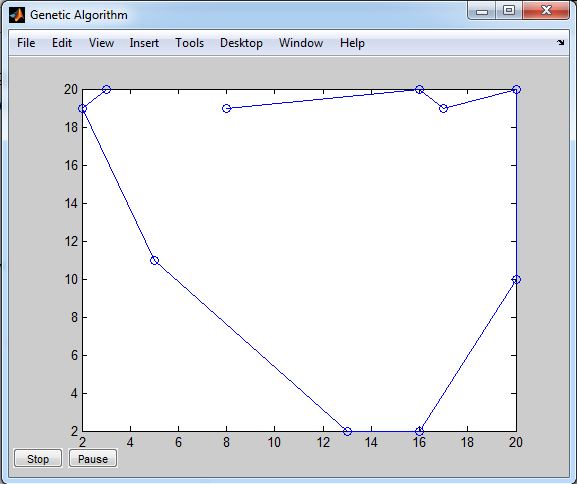


Рис.1. Рішення задачі комівояжера для 10 міст.

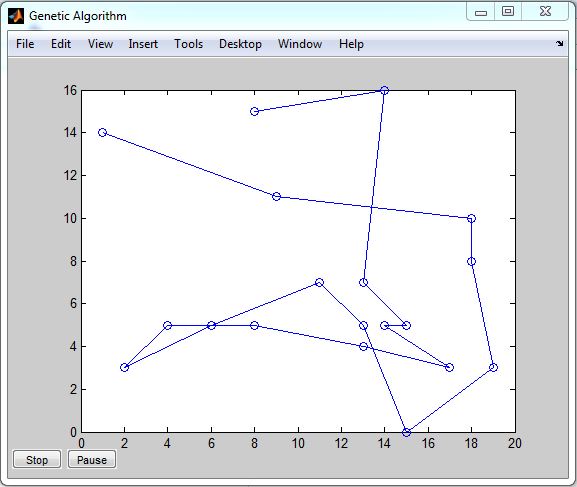


Рис.2. Рішення задачі комівояжера для 20 міст.

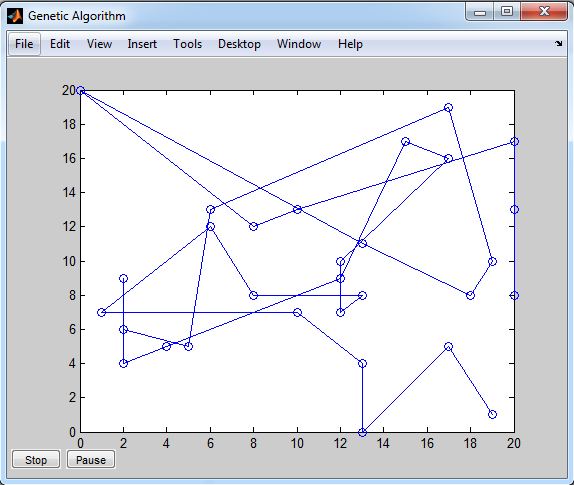


Рис.3. Рішення задачі комівояжера для 30 міст.

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 | 2.74 | 1.72 | 1.57 | 1.67 | 2.24 | 1.84 | 1.67 | 2.81 | 4.03 |
| Мін. довжинна | 61.68 | 61.93 | 64.25 | 93.46 | 103.1 | 110.4 | 184.0 | 157.46 | 116.13 |
| Кількість  ітерацій | 51 | 51 | 51 | 73 | 86 | 54 | 65 | 98 | 124 |
| Кільк. оцінок функції | 1040 | 2600 | 5200 | 1480 | 4350 | 5500 | 1320 | 4950 | 12500 |

**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 з одноточечне впорядковуюче схрещуванням і класична мутація обміну. В результаті програма коректно працює для кількість міст до 15, з більшою кількістю міст шлях комівояжера не оптимальний.