|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКм-12 | 3 | **Комбінаторна оптимізація за допомогою еволюційних методів** |  |  |
| Цедуляк Т. Б. | |
| № залікової: 1508517 | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р.З. | |

**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 - 10

Код програми

**traveling\_salesman\_problem.m**

cities = 40;%кількість міст

locations = zeros(cities,2);

mas = zeros(cities,2);

x=cell(1,1);

for n = 1: cities%генерування 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;

x;

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

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

state,flag,locations);

timeBegin = clock;

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

'CrossoverFcn',@OXcrossover, ...

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

'PlotFcn', my\_plot, ...

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

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

timeEnd = clock;

time=timeBegin-timeEnd %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

**OXcrossover.m**

function xoverKids = OXcrossover(parents,options,NVARS, ...

FitnessFcn,thisScore,thisPopulation)

nKids = length(parents)/2;

xoverKids = cell(nKids,1);

index = 1;

kidsIterator=1;

numberOfGenes=length(thisPopulation{parents(index)});

while kidsIterator<nKids

parent1 = thisPopulation{parents(index)};

index = index + 1;

parent2 = thisPopulation{parents(index)};

index = index + 1;

XorPoint=mod(ceil(rand(1)\*10),numberOfGenes );

if XorPoint==0

XorPoint=XorPoint+1;

end

child1=parent2;

l=XorPoint+1;

for firstPartIterator=1:XorPoint

if ismember(parent1(firstPartIterator), parent2((XorPoint+1):numberOfGenes))==0

child1(firstPartIterator)=parent1(firstPartIterator);

else

for secondPartIterator=l:numberOfGenes

if ismember(parent1(secondPartIterator), parent2((XorPoint+1):numberOfGenes))==0

child1(firstPartIterator)=parent1(secondPartIterator); l=l+1; break;

else

l=l+1;

end

end

end

end

xoverKids{kidsIterator} = child1;

kidsIterator=kidsIterator+1;

child2=parent1;

xoverKids{kidsIterator} = child2;

kidsIterator=kidsIterator+1;

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

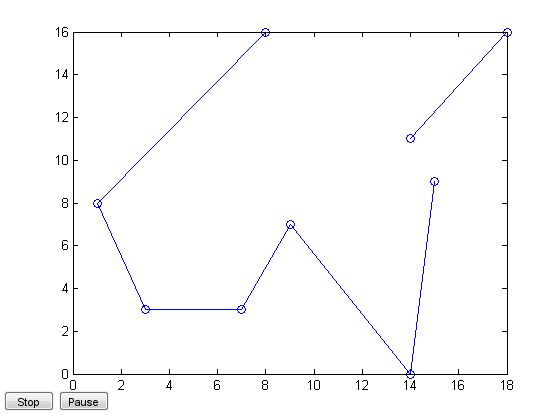


Рис.1. Рішення задачі комівояжера для 10 міст.

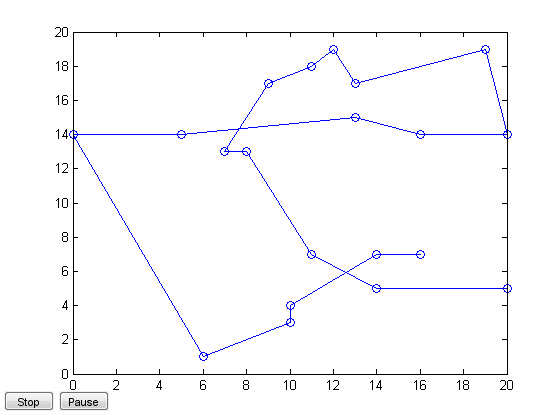


Рис.2. Рішення задачі комівояжера для 20 міст.

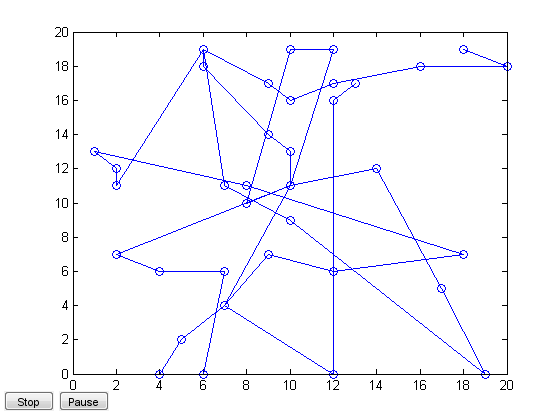


Рис.3. Рішення задачі комівояжера для 40 міст.

Таблиця порівняння кількості міст і популяції

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кількість міст | 10 | | | 20 | | | 40 | | |
| Популяція | 20 | 100 | 200 | 20 | 100 | 200 | 20 | 100 | 200 |
| Час виконання,c | 0.89 | 1.81 | 2.57 | 1.19 | 3.09 | 4.38 | 2.47 | 5.96 | 11.51 |
| Довжина шляху | 60.78 | 61.72 | 63.69 | 107.95 | 76.2 | 90.58 | 196.66 | 168.48 | 144.09 |
| Кількість  ітерацій | 51 | 51 | 51 | 53 | 74 | 66 | 122 | 115 | 135 |
| Кільк. оцінок функції | 1040 | 5200 | 10400 | 1080 | 7500 | 13400 | 2460 | 11600 | 27200 |

**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.