|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| СПКc-11 | 3 | **Комбінаторна оптимізація за допомогою еволюційних методів** |  |  |
| Ленько Ю. С. | |
| № залікової: 1508516 | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р.З. | |

1. **Мета роботи**

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

1. **Індивідуальне завдання**

Розробити за допомогою пакету Matlab програмне забезпечення для

вирішення задачі комівояжера.

**Варіант - 9**

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

'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

**PMX.m**

function xoverKids = PMX(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

**classic\_invert.m**

function mutationChildren = classic\_invert(parents ,options,NVARS, ...

FitnessFcn, state, thisScore,thisPopulation,mutationRate)

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

numberOfGenes=length(thisPopulation{parents(1)});

for i=1:length(parents)

parent = thisPopulation{parents(i)};

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

if XorPoint==0

XorPoint=XorPoint+1;

end

child = parent;

child(XorPoint) = parent(XorPoint+1);

child(XorPoint+1) = parent(XorPoint);

mutationChildren{i} = child;

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

1. **Результати виконання** **індивідуального** **завдання**

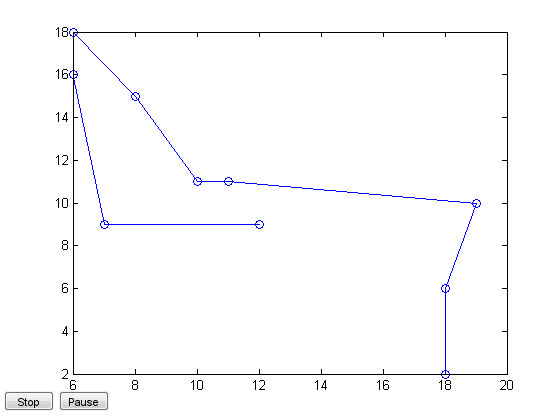


Рис.1. Рішення задачі комівояжера для 10 міст.

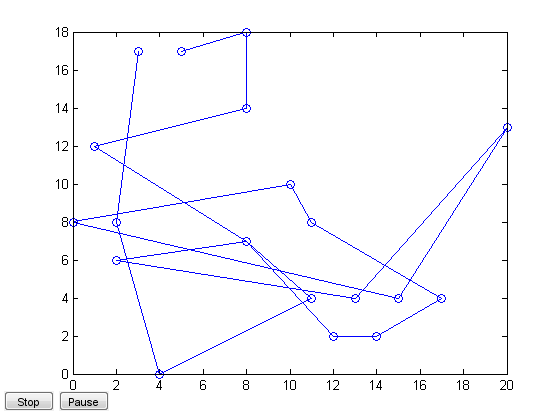


Рис.2. Рішення задачі комівояжера для 20 міст.

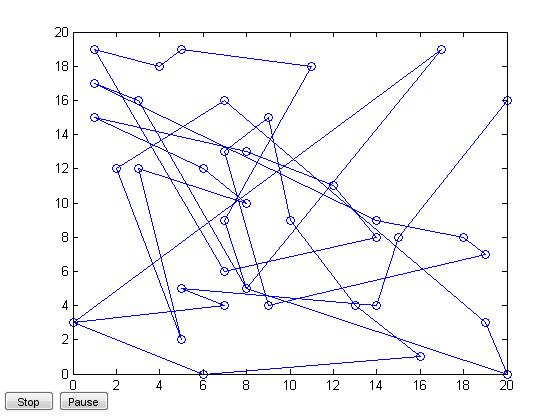


Рис.3. Рішення задачі комівояжера для 40 міст.

Таблиця порівняння кількості міст і популяції

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Кількість міст | 10 | | | 20 | | | 40 | | |
| Популяція | 20 | 100 | 200 | 20 | 100 | 200 | 20 | 100 | 200 |
| Час виконання,c | 1.25 | 1.6 | 2.22 | 1.17 | 1.31 | 1.81 | 1.04 | 1.61 | 1.99 |
| Довжина шляху | 70.38 | 48.55 | 57.81 | 140.79 | 107.15 | 112.63 | 322.81 | 255.47 | 243.93 |
| Кількість  ітерацій | 51 | 51 | 51 | 51 | 51 | 51 | 51 | 51 | 51 |
| Кільк. оцінок функції | 1040 | 5200 | 10400 | 1040 | 5200 | 10400 | 1040 | 5200 | 10400 |

**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.