|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| КНм-14 | 3 | Комбінаторна оптимізація за допомогою еволюційних методів |  |  |
| Євтух Олеся | |
|  | |
| Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні | | Викладач: | |
| Кривий Р.З. | |

**1.Мета**

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

1. **Короткі теоретичні відомості**

**Впорядковуючий оператор схрещування**

Впорядковуючий оператор схрещування (Order crossover, OX) був запропонований Д. Девісом у 1985 році для негомологічних числових хромосом. Схрещування може виконуватися по одній або по двох точках. Точки схрещування вибираються випадково.

При одноточечному схрещуванні в хромосому першого нащадка копіюється хромосома першого батька, а потім гени нащадка, що розташовані правіше точки схрещування, перевпорядковуються у послідовність, що відповідає другому батькові. При цьому другий батько переглядається від початку до кінця, зліва направо, і елементи, яких не вистачає у нащадку, додаються, починаючи від точки схрещування, один по одному.

Крок 1. Випадковим чином вибрати точку схрещування.

Крок 2. Скопіювати в хромосому першого нащадка сегмент хромосоми першого батька, що розташований лівіше точки схрещування.

Крок 3. Інші гени в нащадку копіюються із другого батька в упорядкованому вигляді зліва направо, крім елементів, які вже увійшли до нащадка.

Для створення другого нащадка застосовується аналогічний порядок дій.

При двохточечному впорядковуючому схрещуванні змінюється частина хромосоми, що перебуває між точками схрещування.

Також застосовується **позиційно впорядковуюче схрещування**, що виконується в наступній послідовності.

Крок 1. Випадковим чином вибрати деяку кількість позицій (генів) у першій батьківській хромосомі.

Крок 2. Скопіювати в хромосому першого нащадка обрані на попередньому кроці гени першої батьківської хромосоми. При цьому копіювання генів відбувається в ті ж позиції, на яких вони розташовувалися в першій батьківській хромосомі.

Крок 3. Інші гени в нащадку копіюються із другого батька в упорядкованому вигляді зліва направо, крім елементів, що вже ввійшли в нащадка.

Для створення другого нащадка застосовується аналогічна послідовність дій за винятком того, що перший батько тепер уважається другим, а другий - першим.

В цьому випадку кожна хромосома представляє собою впорядковану послідовність чисел і застосування стандартних операторів схрещування неминуче привело б до утворення нелегальних рішень.

**Мутація обміну**

Мутація обміну використовується для бінарних і числових негомологічних хромосом.

1. При класичній мутації обміну в хромосомі випадковим чином вибираються два гени, які міняються місцями.

Крок 1. Створити хромосому нащадка як копію батьківської хромосоми H = {h1,h2,...,hL}.

Крок 2. Вибрати два числа y1 і y2 випадковим чином із множини Y = {0, 1, 2, .... L+1}, причому y1  y2.

Крок 3. Сформувати нову хромосому H шляхом обміну елементів, розташованих на позиціях y1 і y2.

Таким чином, після застосування класичної мутації обміну одержуємо хромосому H':



1. **Індивідуальне завдання**

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

Розробити за допомогою пакету Matlab програмне забезпечення для вирішення задачі комівояжера.

**Варіант №6**

|  |  |  |
| --- | --- | --- |
| **№** | **Схрещування** | **Мутація** |
| 6 | позиційно впорядковуюче | класична мутація обміну |

**Код програми на мові Matlab :**

function varargout = lab3(varargin)

defaultConfig.xy = 10\*rand(20,2);

defaultConfig.dmat = [];

defaultConfig.popSize = 40;

defaultConfig.numIter = 1e4;

defaultConfig.showProg = true;

defaultConfig.showResult = true;

defaultConfig.showWaitbar = false;

if ~nargin

userConfig = struct();

elseif isstruct(varargin{1})

userConfig = varargin{1};

else

try

userConfig = struct(varargin{:});

catch

error('Expected inputs');

end

end

configStruct = get\_config(defaultConfig,userConfig);

xy = configStruct.xy;

dmat = configStruct.dmat;

popSize = configStruct.popSize;

numIter = configStruct.numIter;

showProg = configStruct.showProg;

showResult = configStruct.showResult;

showWaitbar = configStruct.showWaitbar;

if isempty(dmat)

nPoints = size(xy,1);

a = meshgrid(1:nPoints);

dmat = reshape(sqrt(sum((xy(a,:)-xy(a',:)).^2,2)),nPoints,nPoints);

end

[N,dims] = size(xy);

[nr,nc] = size(dmat);

if N ~= nr || N ~= nc

error('Invalid XY or DMAT inputs!')

end

n = N;

popSize = 4\*ceil(popSize/4);

numIter = max(1,round(real(numIter(1))));

showProg = logical(showProg(1));

showResult = logical(showResult(1));

showWaitbar = logical(showWaitbar(1));

pop = zeros(popSize,n);

pop(1,:) = (1:n);

for k = 2:popSize

pop(k,:) = randperm(n);

end

globalMin = Inf;

totalDist = zeros(1,popSize);

distHistory = zeros(1,numIter);

tmpPop = zeros(4,n);

newPop = zeros(popSize,n);

if showProg

figure('Name','Lab\_3','Numbertitle','off');

hAx = gca;

end

if showWaitbar

hWait = waitbar(0,'Searching for near-optimal solution ...');

end

for iter = 1:numIter

for p = 1:popSize

d = dmat(pop(p,n),pop(p,1));

for k = 2:n

d = d + dmat(pop(p,k-1),pop(p,k));

end

totalDist(p) = d;

end

[minDist,index] = min(totalDist);

distHistory(iter) = minDist;

if minDist < globalMin

globalMin = minDist;

optRoute = pop(index,:);

if showProg

rte = optRoute([1:n 1]);

if dims > 2, plot3(hAx,xy(rte,1),xy(rte,2),xy(rte,3),'r.-');

else plot(hAx,xy(rte,1),xy(rte,2),'r.-'); end

title(hAx,sprintf('Total Distance = %1.4f, Iteration = %d',minDist,iter));

drawnow;

end

end

randomOrder = randperm(popSize);

for p = 4:4:popSize

rtes = pop(randomOrder(p-3:p),:);

dists = totalDist(randomOrder(p-3:p));

[ignore,idx] = min(dists);

bestOf4Route = rtes(idx,:);

routeInsertionPoints = sort(ceil(n\*rand(1,2)));

I = routeInsertionPoints(1);

J = routeInsertionPoints(2);

for k = 1:4

tmpPop(k,:) = bestOf4Route;

switch k

case 2

tmpPop(k,I:J) = tmpPop(k,J:-1:I);

case 3

tmpPop(k,[I J]) = tmpPop(k,[J I]);

case 4

tmpPop(k,I:J) = tmpPop(k,[I+1:J I]);

otherwise

end

end

newPop(p-3:p,:) = tmpPop;

end

pop = newPop;

if showWaitbar && ~mod(iter,ceil(numIter/325))

waitbar(iter/numIter,hWait);

end

end

if showWaitbar

close(hWait);

end

if nargout

resultStruct = struct( ...

'xy', xy, ...

'dmat', dmat, ...

'popSize', popSize, ...

'numIter', numIter, ...

'showProg', showProg, ...

'showResult', showResult, ...

'showWaitbar', showWaitbar, ...

'optRoute', optRoute, ...

'minDist', minDist);

varargout = {resultStruct};

end

end

function config = get\_config(defaultConfig,userConfig)

config = defaultConfig;

defaultFields = fieldnames(defaultConfig);

userFields = fieldnames(userConfig);

nUserFields = length(userFields);

for i = 1:nUserFields

userField = userFields{i};

isField = strcmpi(defaultFields,userField);

if nnz(isField) == 1

thisField = defaultFields{isField};

config.(thisField) = userConfig.(userField);

end

end

end

**Результати виконання:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тест** | **Кількість міст** | **Розмір популяції** | **Відстань між містами** | **Кінцева ітерація** |
| Тест № 1 | 20 | 40 | 35,8208 | 257 |
| Тест № 2 | 100 | 200 | 80,7191 | 8860 |
| Тест № 3 | 30 | 120 | 44,7396 | 1916 |

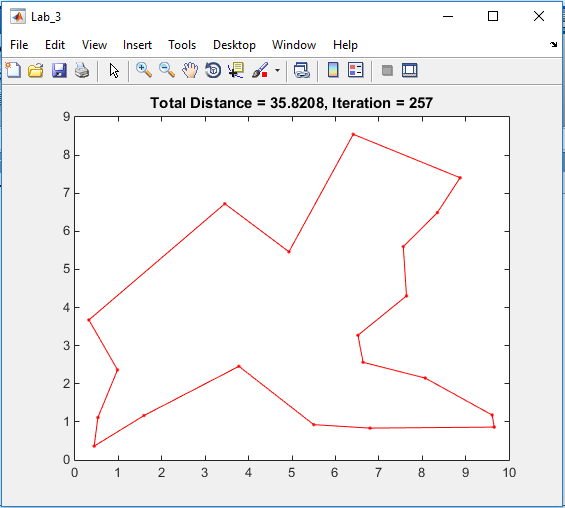


Рис. 1 Результат роботи програми для Тест № 1

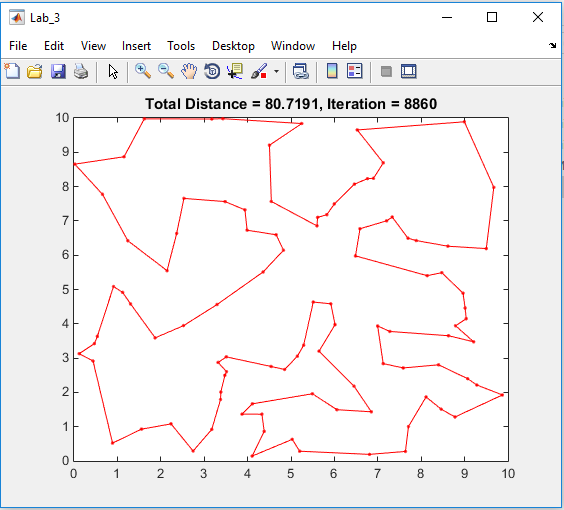


Рис. 2 Результат роботи програми для Тест № 2

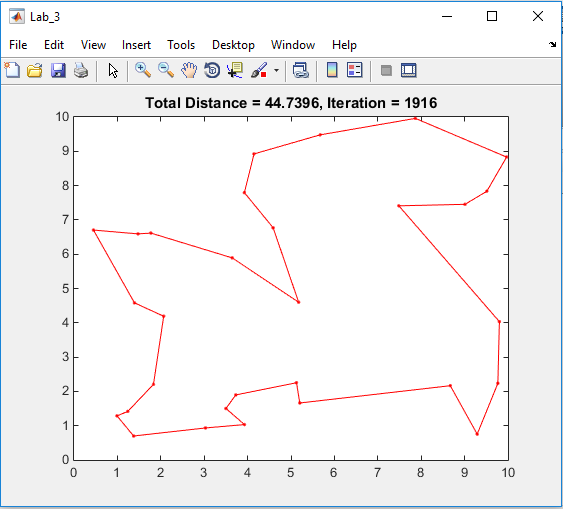


Рис. 3 Результат роботи програми для Тест № 3

**4. Висновок**

У результаті проведення лабораторної роботи я вивчила роботу функції пакету Matlab. Розробила за допомогою пакету Matlab програмне забезпечення, що реалізує методи еволюційного пошуку.