|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САП** | | Тема | оцінка | підпис |
| КНС-13 | 3(номер лаб.) | **КОМБІНАТОРНА**  **ОПТИМІЗАЦІЯ ЗА ДОПОМОГОЮ ЕВОЛЮЦІЙНИХ** **МЕТОДІВ** |  |  |
| Писаренко В.Б. | |
|  | |
| **Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні** | | Викладач: | |
| асистент каф. САПР  Кривий Р.З. | |

**1.МЕТА**

Ознайомитися з основними теоретичними відомостями за темою роботи. Вивчити роботу функції ga пакету Matlab. Розробити за допомогою пакету Matlab програмне забезпечення, що реалізує 2 методи еволюційного пошуку. Основні еволюційні оператори для реалізації еволюційних методів обрати з таблиці 1 відповідно до варіанту

**2. РЕЗУЛЬТАТ ВИКОНАННЯ**

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 №4**



Жадібний оператор схрещування (поглинаюче схрещування, greedy

crossover) був запропонований в 1985 році Д. Грефенстеттом у співавторстві з

іншими вченими для розв’язку задачі комівояжера. Це евристичний оператор

схрещування, орієнтований на використання знань про об'єкт.

Ідея побудови “жадібного” алгоритму полягає в наступному. На

кожному кроці послідовно вибираються кращі елементи із множини наявних,

тобто рішення, що поліпшують цільову функцію, причому таким чином, щоб

не порушувати діючих обмежень. Генерація нащадків відбувається за рахунок

вибору кращих ділянок батьківських хромосом і їхнього наступного

сполучення.

Схема роботи жадібного оператора схрещування може змінюватися

залежно від характеру розв'язуваних задач.

Послідовність виконання жадібного оператора схрещування наведена

нижче.

Крок 1. Обчислити значення цільової функції у відібраних для

схрещування хромосом: f(H1) і f(H2), H1 = {h11, h12, …, h1n},

H2 = {h21, h22, …, h2n}, i,hvi 1,n, v 1,2 .

Крок 2. Встановити: j = 1. Випадковим чином вибрати початкову точку

для генерації хромосоми-нащадка: pj = rand(1,n).

Крок 3. Встановити temp = pj; j = j + 1.

Крок 4. Визначити наступну точку хромосоми-нащадка:

H – номер гена хромосоми Hv , що відповідає значенню temp.

Крок 5. У випадку, якщо хромосома-нащадок складена повністю

(j = n), перейти на крок 8.

Крок 6. Виконати перевірку на передчасне замикання циклу: pj = pd,

d 1, j 1 . У випадку передчасного замикання циклу збільшити шлях за

рахунок включення гена, обраного випадковим образом із числа ще не

включених.

Крок 7. Виконати перехід на крок 3.

Крок 8. Кінець.

Практика показує, що застосування жадібного оператора схрещування

підвищує швидкість збіжності розв’язку, але в той же час це сприяє

зменшенню розмаїтості популяції, що веде до її швидкого виродження, а

також зниженню можливостей виходу з локальних оптимумів.

Код програми на мові Matlab :

function varnewout = tsp\_ga(varargin)

% Ініціалізація початкових змінних

defaultConfig.xy = 10\*rand(50,2);

defaultConfig.matrixd = [];

defaultConfig.legSize = 100;

defaultConfig.quantIter = 1e4;

defaultConfig.getProg = true;

defaultConfig.getResult = true;

defaultConfig.getWaitbar = false;

% Інтерпретація вхідних данних

if ~nargin

userConfig = struct();

elseif isstruct(varargin{1})

userConfig = varargin{1};

end

configStruct = get\_config(defaultConfig,userConfig);

% Конфігурація програми

xy = configStruct.xy;

matrixd = configStruct.matrixd;

legSize = configStruct.legSize;

quantIter = configStruct.quantIter;

getProg = configStruct.getProg;

getResult = configStruct.getResult;

getWaitbar = configStruct.getWaitbar;

if isempty(matrixd)

nPoints = size(xy,1);

a = meshgrid(1:nPoints);

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

end

% Перевірка вхідних даних

[N,dims] = size(xy);

[nr,nc] = size(matrixd);

n = N;

% Перевірка вх даних

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

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

getProg = logical(getProg(1));

getResult = logical(getResult(1));

getWaitbar = logical(getWaitbar(1));

% Створення популяції

pop = zeros(legSize,n);

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

for k = 2:legSize

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

end

% Запусе ГА

globalMin = Inf;

totalDist = zeros(1,legSize);

distHistory = zeros(1,quantIter);

tmpPop = zeros(4,n);

newPop = zeros(legSize,n);

if getProg

figure('Name','TSP\_GA | Current Best Solution','Numbertitle','off');

hAx = gca;

end

if getWaitbar

end

for iter = 1:quantIter

% Обчислення кожного члена

for p = 1:legSize

d = matrixd(pop(p,n),pop(p,1)); % Closed Path

for k = 2:n

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

end

totalDist(p) = d;

end

% Знаходження найкращого шляху

[minDist,index] = min(totalDist);

distHistory(iter) = minDist;

if minDist < globalMin

globalMin = minDist;

optRoute = pop(index,:);

if getProg

% Графік найкращого шляху

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('Full Distance = %1.4f, Iteration = %d',minDist,iter));

drawnow;

end

end

% Оператори генетичного алгоритму

randomOrder = randperm(legSize);

for p = 4:4:legSize

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 % Мутація

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;

% Оновлення

if getWaitbar && ~mod(iter,ceil(quantIter/325))

waitbar(iter/quantIter,hWait);

end

end

if getWaitbar

close(hWait);

end

% Повернення вих даних

if nargout

resultStruct = struct( ...

'xy', xy, ...

'matrixd', matrixd, ...

'legSize', legSize, ...

'quantIter', quantIter, ...

'getProg', getProg, ...

'getResult', getResult, ...

'getWaitbar', getWaitbar, ...

'optRoute', optRoute, ...

'minDist', minDist);

varnewout = {resultStruct};

end

end

% Зміна початкової конфігурації вхідних параметрів

function config = get\_config(defaultConfig,userConfig)

% Ініціалізація вхідної конфігурації

config = defaultConfig;

defaultFields = fieldnames(defaultConfig);

userFields = fieldnames(userConfig);

nUserFields = length(userFields);

% Зміна усієї стандартної конфігурації разом з даними користувача

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

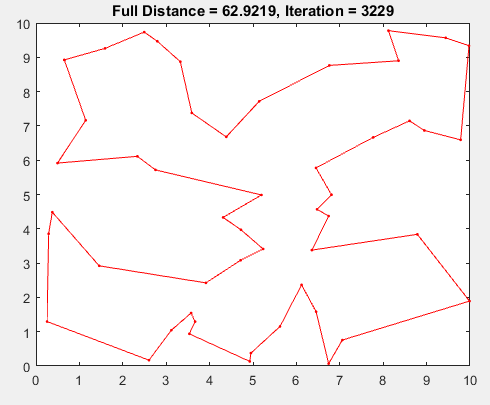


Рис 1 : Результат роботи з програми

**4.ВИСНОВОК.**

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