Прізвище: Сливка

Ім’я: Наталія

Група: КНСП-11

Дата прийняття роботи

у системі Git: 02.06.2018

Дисципліна: Методи нечіткої логіки та еволюційні алгоритми при автоматизованому проектуванні

Перевірив: Кривий Р.З.

**Звіт до лабораторної роботи № 5**

**«Рішення задачі комівояжера за допомогою генетичного алгоритму»**

**МЕТА РОБОТИ**

Ознайомитися з основними теоретичними відомостями про задачу комівояжера, обрати середовище розробки та мову програмування, реалізувати вирішення задачі комівояжера для [10; 50] міст за допомогою генетичного алгоритму.

**КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

Задача комівояжераполягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Зазвичай задано, що маршрут повинен проходити через кожне місто тільки один раз, в такому випадку розв'язок знаходиться серед гамільтонових циклів.

Існує маса різновидів узагальненої постановки задачі, зокрема геометрична задача комівояжера (коли матриця відстаней відображає відстані між точками на площині), трикутна задача комівояжера (коли на матриці вартостей виконується нерівність трикутника), симетрична та асиметрична задачі комівояжера.   
  
Прості методи розв'язання задачі комівояжера: повний лексичний перебір, жадібні алгоритми (метод найближчого сусіда), метод включення найближчого міста, метод найдешевшого включення, метод мінімального кістяка дерева. На практиці застосовують різні модифікації ефективніших методів: метод гілок і меж і метод генетичних алгоритмів, а так само алгоритм мурашиної колонії. 

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Розробити програмне забезпечення для розв’язку задачі Комівояжера.

Варіант 8.

**Код головного файлу програми:**

function main()

clear; clc;

[names, x, y] = textread('cities.csv', '%s %d %d', 'delimiter', ',');

global WAY\_MATRIX;

WAY\_MATRIX = getWayMatrix(x, y, 1000);

population\_size = ceil(length(x)^(1/2));

elite\_count = floor(0.3\*population\_size);

nvars = length(x);

global RET;

RET = struct('generation', 0, 'population', struct, 'fvals', struct);

options = gaoptimset(...

'Generations', (300), ...

'CreationFcn', @CreationFcn, ...

'PopulationSize', population\_size, ...

'MutationFcn', @MutationFcn, ...

'CrossoverFcn', @CrossoverFcn, ...

'OutputFcns', {@OutputFcn}, ...

'SelectionFcn', @selectiontournament, ...

'PlotFcns', {@gaplotbestf, @gaplotdistance} ...

);

%SelectionFcn ?????? ? ?????????

[xval,fval,exitflag,output,population,scores] = ga(@FitnessFcn, nvars, options);

XX = x; YY = y; NNames = names;

for i = 1:1:nvars

XX(i) = x(xval(i));

YY(i) = y(xval(i));

NNames(i) = names(xval(i));

end

XR = [XX(nvars), XX(1)];

YR = [YY(nvars), YY(1)];

figure

plot(XX, YY, '-\*', XR, YR, '--')

text(XX+1,YY+1,NNames)

disp('Start population:');

PrintIter(RET.population.s0, RET.fvals.s0, nvars, population\_size);

disp('Generation 1:');

PrintIter(RET.population.s1, RET.fvals.s1, nvars, population\_size);

disp('Generation 2:');

PrintIter(RET.population.s2, RET.fvals.s2, nvars, population\_size);

disp('Result:');

PrintIter(population, scores, nvars, population\_size);

fprintf('Best:\n');

PrintOne(xval, fval, nvars);

PrintOneStr(NNames, fval, nvars);

end

function way\_matrix = getWayMatrix(X, Y, M)

ret = zeros(length(Y), length(X));

for i = 1:1:length(Y)

for j = 1:1:length(X)

if (i == j)

ret(i,j) = M;

else

v = ((X(i) - X(j)).^2 + (Y(i)-Y(j)).^2).^(1/2);

ret(i,j) = v;

end

end

end

way\_matrix = ret;

end

function PrintIter(curp, fval, nvars, psize)

for i = 1:1:psize

PrintOne(curp(i,:), fval(i), nvars);

end

fprintf('\n');

end

function PrintOne(x, fval, nvars)

fprintf('[');

for j=1:1:nvars

fprintf('%3d, ', x(j));

end

fprintf('] => %.2f\n', fval);

end

function PrintOneStr(x, fval, nvars)

fprintf('[');

for j=1:1:nvars

fprintf('%3s, ', char(x(j)));

end

fprintf('] => %.2f\n', fval);

end

**Отримані результати:**

Початкові дані:

Start population:

[ 9, 10, 2, 1, 8, 5, 7, 6, 4, 3, ] => 251.06

[ 3, 8, 10, 9, 6, 1, 7, 4, 5, 2, ] => 205.81

[ 6, 9, 2, 7, 5, 4, 1, 10, 8, 3, ] => 256.78

[ 2, 6, 5, 1, 7, 10, 9, 3, 4, 8, ] => 255.04

Отримані результати:

Generation 1:

[ 3, 8, 10, 9, 6, 1, 7, 4, 5, 2, ] => 205.81

[ 3, 8, 10, 9, 6, 1, 7, 4, 5, 2, ] => 205.81

[ 3, 8, 10, 9, 6, 1, 7, 4, 5, 2, ] => 205.81

[ 3, 8, 10, 9, 6, 2, 7, 4, 5, 1, ] => 251.80

Generation 2:

[ 3, 8, 10, 9, 6, 1, 7, 4, 5, 2, ] => 205.81

[ 3, 8, 10, 9, 6, 1, 7, 4, 5, 2, ] => 205.81

[ 3, 8, 10, 9, 6, 1, 7, 4, 5, 2, ] => 205.81

[ 3, 8, 10, 4, 6, 1, 7, 9, 5, 2, ] => 205.42

Result:

[ 7, 8, 4, 9, 6, 1, 5, 10, 2, 3, ] => 147.94

[ 7, 8, 4, 9, 6, 1, 5, 10, 2, 3, ] => 147.94

[ 7, 8, 4, 9, 6, 1, 5, 10, 2, 3, ] => 147.94

[ 7, 6, 4, 9, 8, 1, 5, 10, 2, 3, ] => 183.99

Best:

[ 7, 8, 4, 9, 6, 1, 5, 10, 2, 3, ] => 147.94

[ G, K, D, L, F, A, E, M, B, C, ] => 147.94

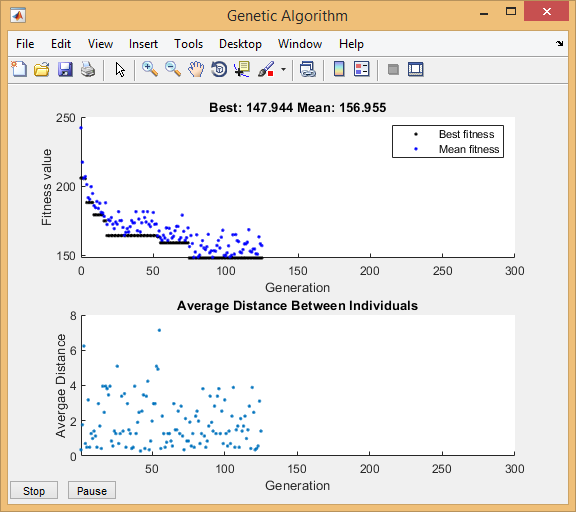


Рис. 1 Сходимість функції

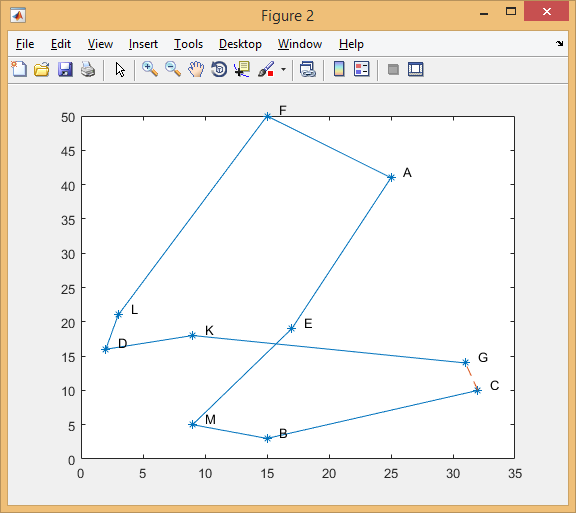


Рис. 2 Графічне представлення маршруту

**Висновки.**

У ході виконання даної лабораторної роботи я ознайомилась із принципом побудови та розв’язання задачі Комівояжера. Як результат, розробила програмне забезпечення для розв’язку задачі Комівояжера за допомогою генетичного алгоритму.