**INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY**

**CAMPUS ESTADO DE MÉXICO**



**Applied Computer Science**

**Masters in Nanotechnology**

Manuel Valenzuela Rendón

**Simulated Annealing**

Antonio Osamu Katagiri Tanaka (A01212611)

Bruno González Soria (A01169284)

**Due date: April 29, 2019, 11:59PM**

**MATLAB Script and Implemented Functions**

%% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% \* AUTHOR(S) :

% \* Bruno González Soria (A01169284)

% \* Antonio Osamu Katagiri Tanaka (A01212611)

% \*

% \* FILENAME :

% \* HW03.m

% \*

% \* DESCRIPTION :

% \* Computación Aplicada (Ene 19 Gpo 1)

% \* Homework on Simulated Annealing

% \*

% \* NOTES :

% \*

% \*

% \* START DATE :

% \* 25 Apr 2019

% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

warning('off')

clc;

close all;

%% \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Problem:

% Using simulated annealing in the global optimization toolbox, solve the

% traveling salesman problem for 70 cities with coordinates generated in the

% following way:

% N = 70;

% rng(123);

% coordinates = rand(N,2);

% Upload to Blackboard a pdf file that contains the following:

% A script of your solution

% A plot of the best route found

N = 70;

rng(123);

coordinates = rand(N,2);

distances = TSPtable(coordinates);

objFcnTSP(distances);

route0 = randperm(N)';

TSPplot(route0,coordinates,'b',2)

xlabel('x')

ylabel('y')

title(sprintf('TSP with %d cities',N))

options = optimoptions(@simulannealbnd,'DataType','custom',...

'AnnealingFcn',@TSPinversion);

r0 = randperm(N);

TSPplot(r0,coordinates,'-')

title(sprintf('initial random route, cost=%f',objFcnTSP(r0)))

r = simulannealbnd(@objFcnTSP,r0,[],[],options);

TSPplot(r,coordinates,'-')

title(sprintf('cost=%f',objFcnTSP(r)))

function distances = TSPtable(coordinates)

[nCities,nx] = size(coordinates);

if nx ~= 2

error('coordinates should be an (nCities) x 2 matrix')

end

distances = zeros(nCities);

for i=1:nCities

for j=i: nCities

distances(i,j) = ...

sqrt( (coordinates(i,1)-coordinates(j,1))^2 + ...

(coordinates(i,2)-coordinates(j,2))^2);

distances(j,i) = distances(i,j);

end

end

end

function f = objFcnTSP(varargin)

% f = objFcnTSP(distances)

% Loads distance matrix.

% f = objFcnTSP(route)

% Evaluates a route given a distances matrix.

persistent distances

if isempty(varargin)

clear distances

else

[n,m] = size(varargin{1});

if n==m

distances = varargin{1};

else

route = varargin{1};

n = length(route);

% Initialize with distance between last a first city

f = distances(route(n), route(1));

for i=2:n

% Add distance from city i to city i-1

f = f + distances(route(i-1), route(i));

end

end

end

end

function neighbor = TSPinversion(optimValues,varargin)

route = optimValues.x;

n = length(route);

m1 = floor(rand\*n)+1;

m2 = mod(floor(rand\*(n-1))+m1, n)+1;

n1 = min([m1 m2]);

n2 = max([m1 m2]);

neighbor = route;

neighbor(n1:n2) = route(n2:-1:n1);

end

function TSPplot(route, coordenates, s, varargin)

% plotTSP(route, coordenates, s, flag=0)

%

% Plots a TSP route given their coordinates. flag specifies how

% the cities are enumerated:

% 0: no numbers

% 1: order in which they are visited

% 2: original numbers

if length(varargin)>=1

flag = varargin{1};

else

flag = 1;

end

n = length(route);

xy = [];

for i=1:n

xy = [xy;coordenates(route(i),:)];

end

xy = [xy; xy(1,:)];

axis([0 1 0 1])

if flag==0

plot(xy(:,1),xy(:,2),s)

axis([0 1 0 1])

end

if flag==1

% Order in which they are visited

plot(xy(:,1),xy(:,2),s,...

xy(:,1),xy(:,2),'.r',...

xy(1,1),xy(1,2),'ok',...

xy(2,1),xy(2,2),'sb')

axis([0 1 0 1])

for i=1:n

text(coordenates(route(i),1)+0.01,coordenates(route(i),2),...

sprintf('%d',i), 'FontSize', 8)

end

end

if flag==2

% Original numbers

plot(xy(:,1),xy(:,2),s,...

xy(:,1),xy(:,2),'.r',...

xy(1,1),xy(1,2),'ok',...

xy(2,1),xy(2,2),'sb')

axis([0 1 0 1])

for i=1:n

text(coordenates(route(i),1)+0.01,coordenates(route(i),2),...

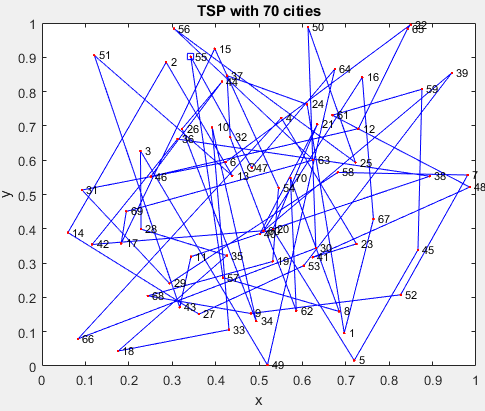
sprintf('%d',route(i)), 'FontSize', 8)

end

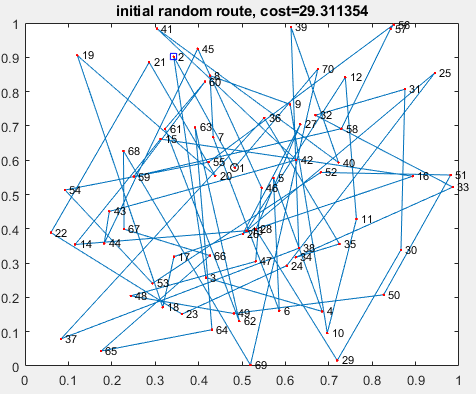
end

end

**Results**

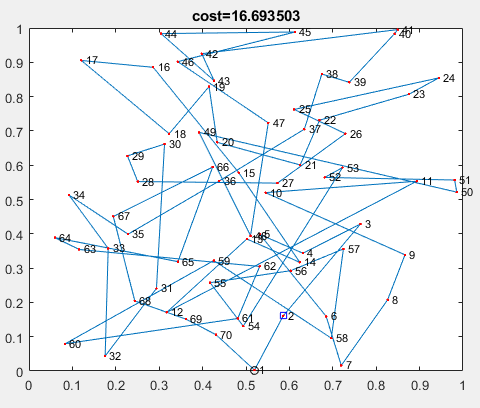


**Figure 1.** Traveling Salesman problem for 70 cities.



**Figure 2.** Random root taken with random price.

Optimization terminated: change in best function value less than options. FunctionTolerance.



**Figure 3.** Optimized route with lowest cost.