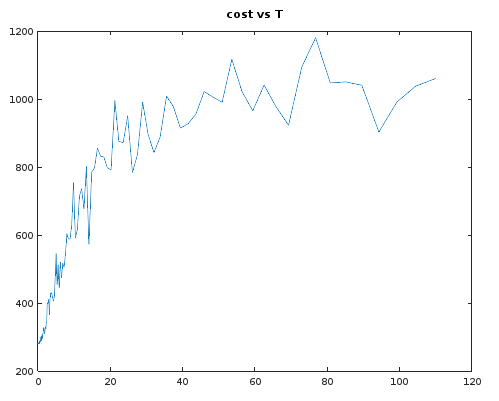
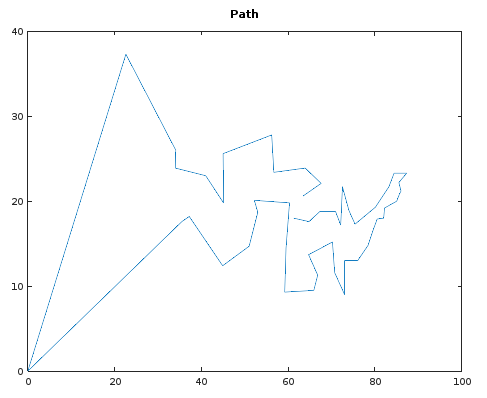
Final Minimum cost = 281.74





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

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fourier Transfrom and Filtering \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*%

% Jonathan Shipley

% Scientific Modeling

% 3/28/17

% Description: This program determines the shortest path between all the capitals of each

% state in the US using simulated annealing.

% Important Parameters: x and y are preset. T, "temperature", to determine criteria for

% random factors.

% Input: x and y are set

% Output: Outputs a graph of the cooling schedule, the minimum distance for the path and

% the x and y values for the shortest path.

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

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

% clear all previous variables

clear;

% Starting 50 city co-ordinates for the Traveling Salesman Problem.

y=[11.6 37.3 12.4 13.7 17.6 18.7 20 18 9 13 0 23 18.8 18.8 20.6 18 17.2 9.5 23.3 17.9 21.2...

21.7 23.9 11.3 17.6 25.6 19.8 18.2 22.2 19.2 14.7 21.7 14.8 27.8 18.9 14.5 23.9 19.3 20.8...

13 23.4 15.2 9.3 19.8 23.3 16.5 26 17.3 22.1 20.1];

x=[70.7 22.6 44.9 64.7 35.5 53 85 82 73 73 0 41 67.3 70.9 63.4 61.3 72.1 65.9 87.3 80.5 86 72.5...

63.9 66.8 64.8 45 60.3 37.2 85.5 82.2 51 83.2 78.4 56.2 74 59.5 34 80.1 85.6 76 56.7 70.2...

59.2 45.1 84.4 79.5 34.1 75.4 67.6 52.2];

N = 50; % number of states

distance = 0;

% find path length between all 50 cities

for k=1:N-1

distance += ((x(k) - x(k+1))^2 + (y(k) - y(k+1))^2)^0.5;

end

distance += ((x(50) - x(1))^2 + (y(50) - y(1))^2)^0.5; % make it round trip

cost(1) = distance;

% Define "temperature"

T(1) = 110;

TInitial = T(1);

costOld = distance;

count = 0;

n = 1;

while T(n) > 0.000001\*TInitial

count = 0;

for p=1:1000

xtemp = x;

ytemp = y;

% swap two cities

city1 = randi(50);

city2 = randi(50);

xtemp([city1 city2]) = xtemp([city2 city1]);

ytemp([city1 city2]) = ytemp([city2 city1]);

distance = 0;

% find path length between all 50 cities

for k=1:N-1

distance += ((xtemp(k) - xtemp(k+1))^2 + (ytemp(k) - ytemp(k+1))^2)^0.5;

end

distance += ((xtemp(50) - xtemp(1))^2 + (ytemp(50) - ytemp(1))^2)^0.5; % make it round trip

costNew = distance;

dcost = costNew - costOld;

if dcost < 0 %keep swapped coordinates

x = xtemp;

y = ytemp;

costOld = costNew;

end

if dcost > 0 && exp(-dcost / T(n)) > rand % if meets this condition then keep the swap

x = xtemp;

y = ytemp;

costOld = costNew;

end

end

n+=1;

T(n) = 0.95 \* T(n-1);

cost(n) = costOld;

%distance

end

min(T)

n

min(cost)

plot(T, cost)

title('cost vs T')

figure

plot(x, y)

title('Path')