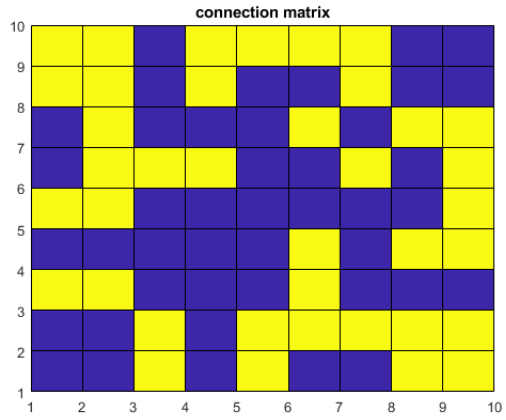
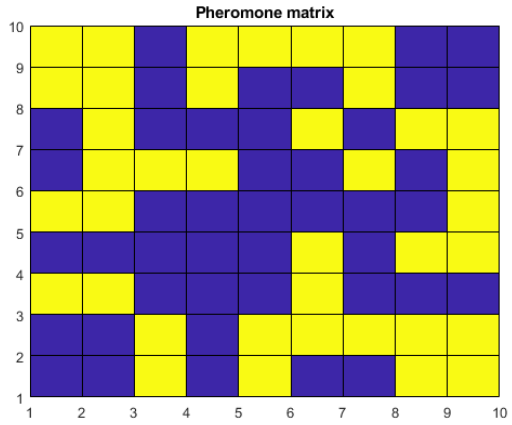
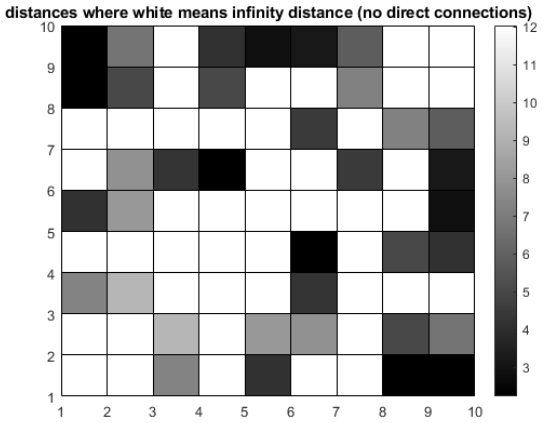
Home work 4 chap 15 Ant colony

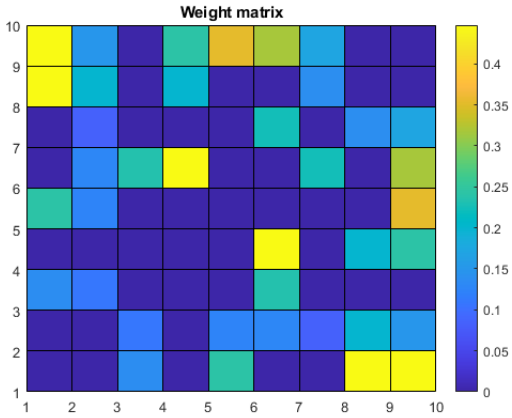
15.1

This task is basically to code some of the functions that we can later use when doing the ant colony algorithm, I will here show the plots that are asked for :







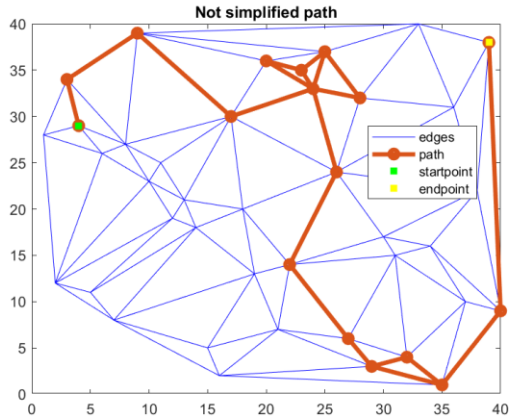


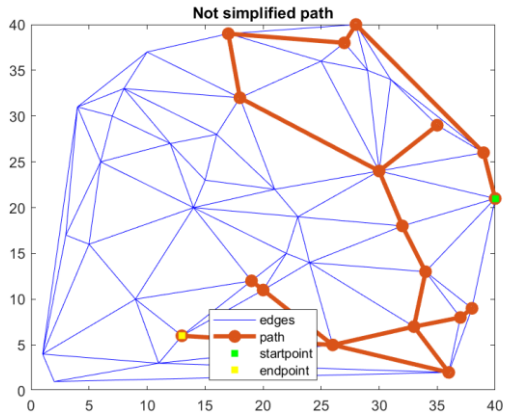
15.7

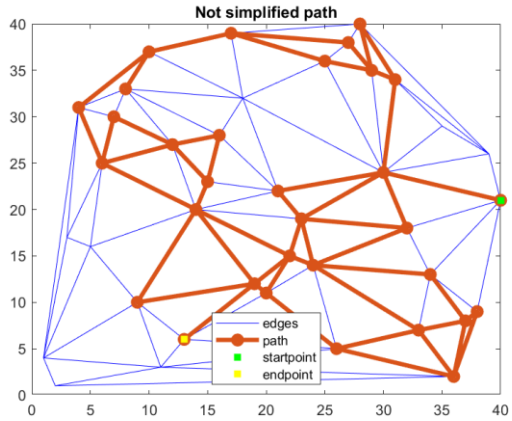
a) is just implementation

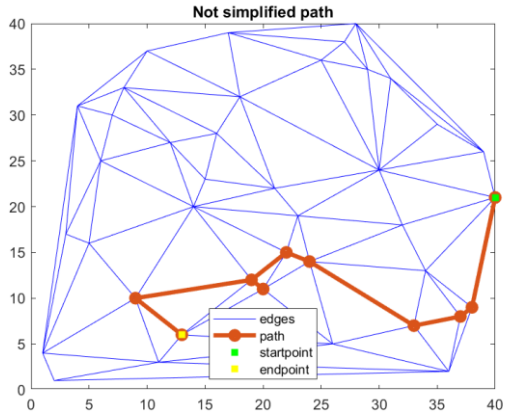
b)

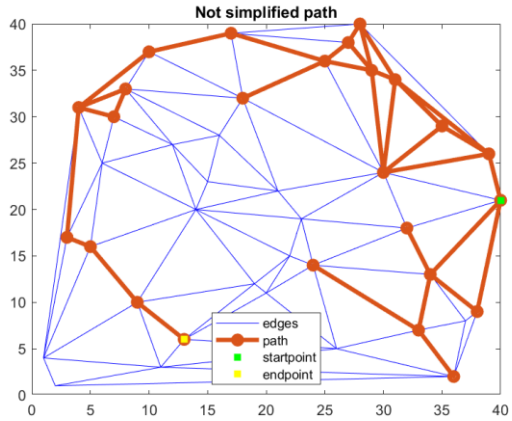
Not simplified paths for the ants that reached destination:

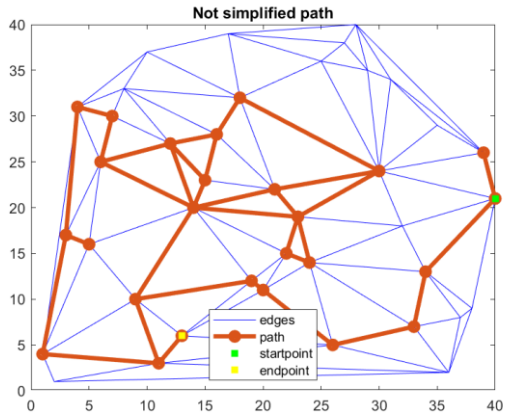

Not simplified path
edges
path
startpoint
endpoint

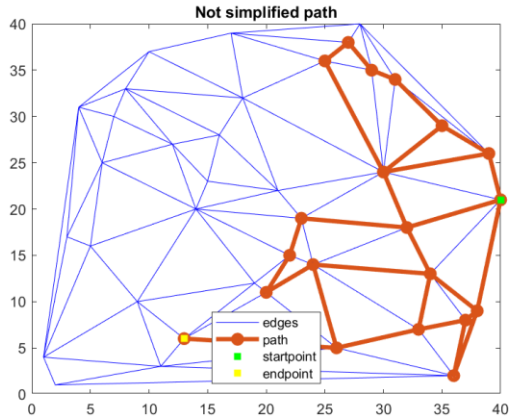

Not simplified path
edges
path
startpoint
endpoint

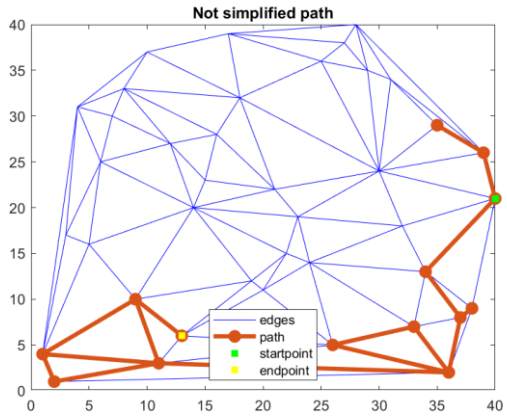

Not simplified path
edges
path
startpoint
endpoint

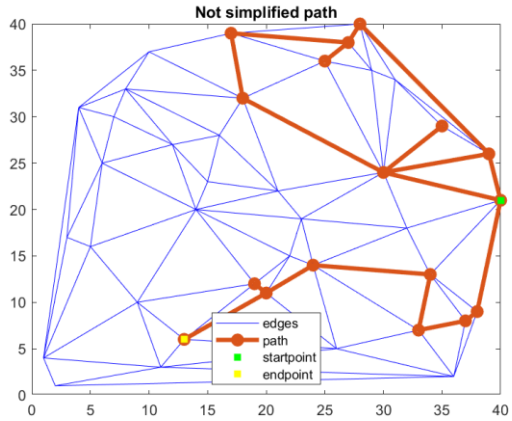

Not simplified path
edges
path
startpoint
endpoint

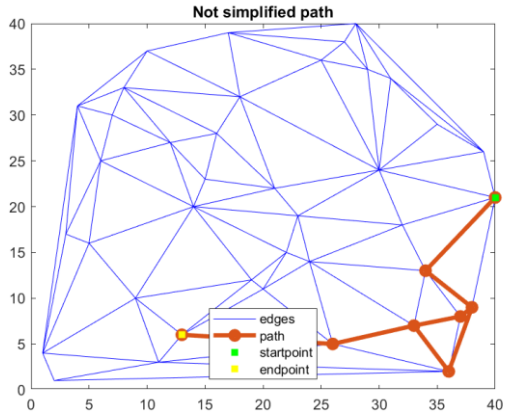

Not simplified path
edges
path
startpoint
endpoint


Not simplified path
edges
path
startpoint
endpoint

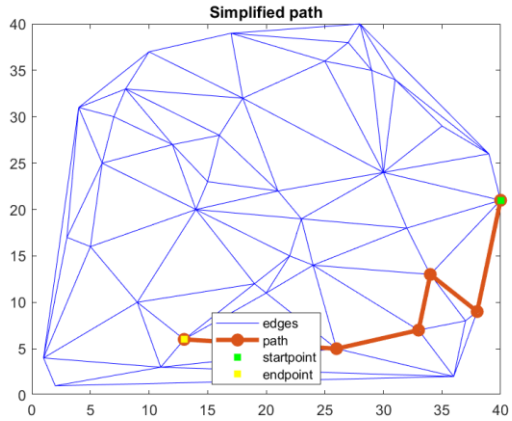


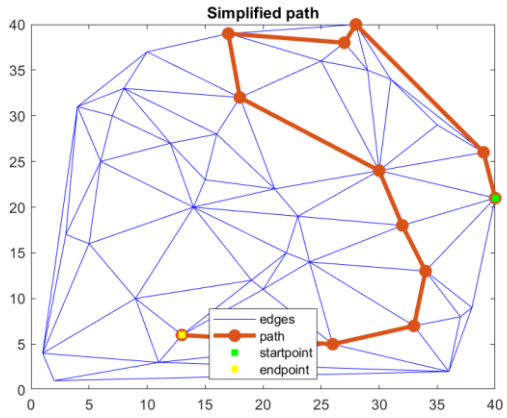


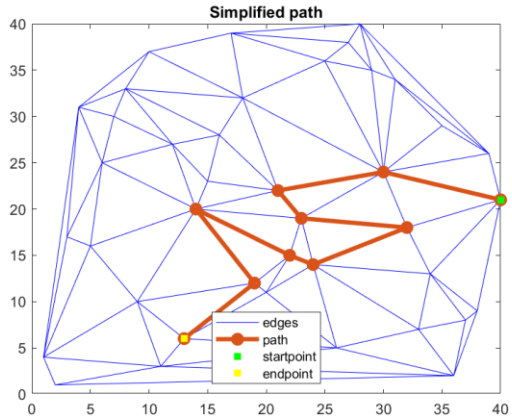


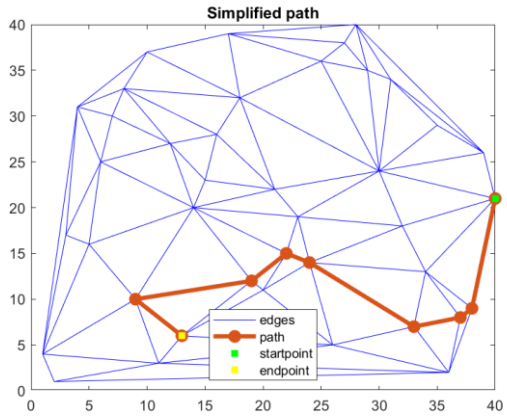


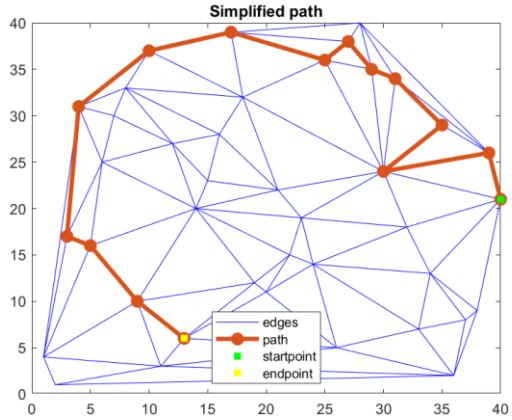
simplified paths for the ants that reached destination:

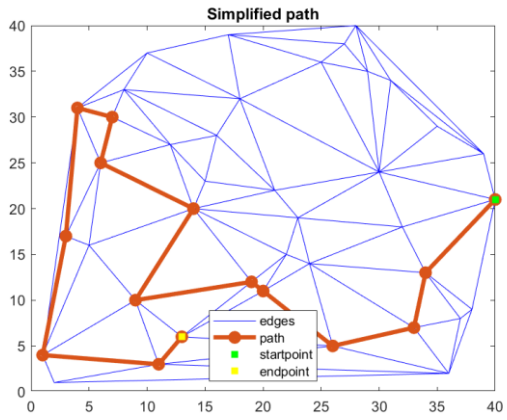


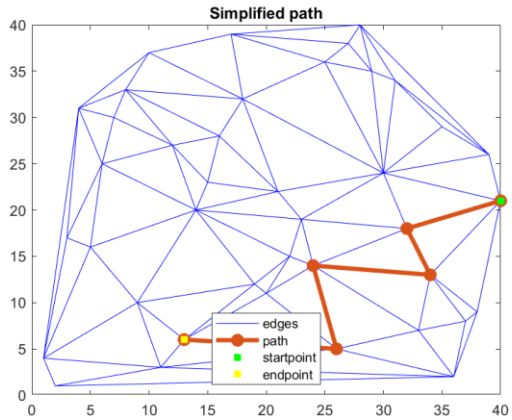


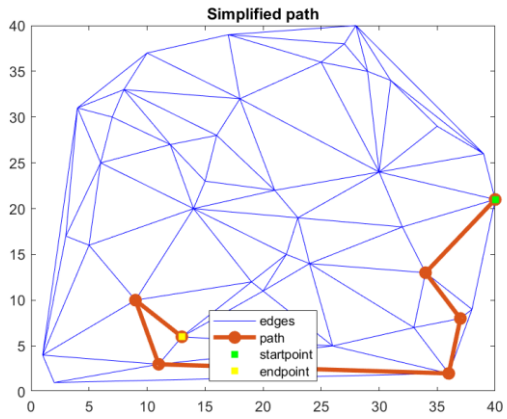

Simplified path
edges
path
startpoint
endpoint

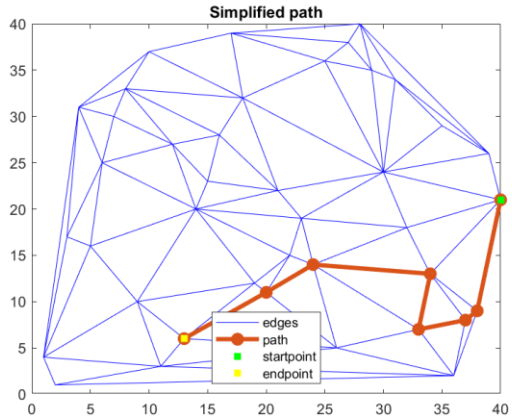

Simplified path
edges
path
startpoint
endpoint

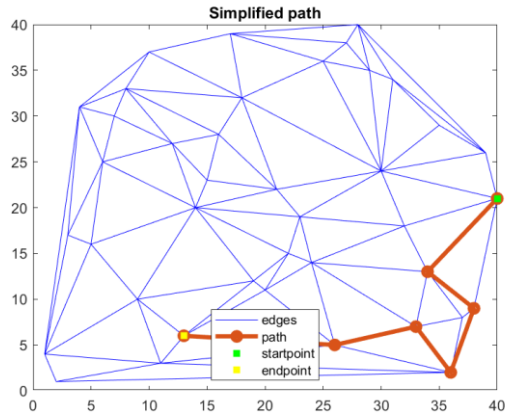

Simplified path
edges
path
startpoint
endpoint


Simplified path
edges
path
startpoint
endpoint

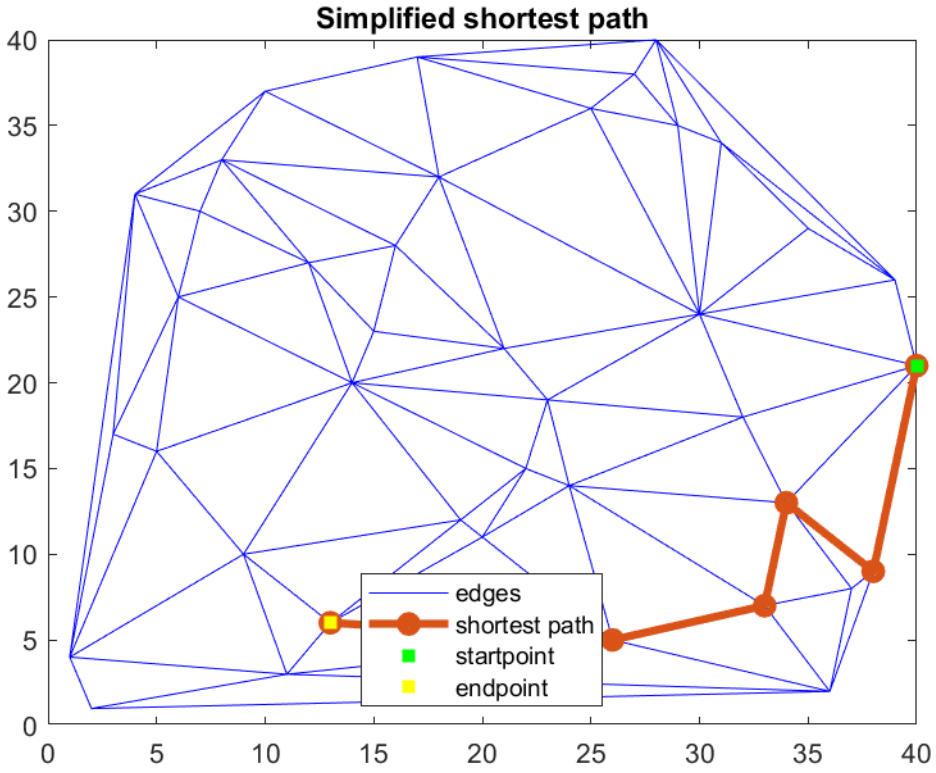

Simplified path
edges
path
startpoint
endpoint


Simplified path
edges
path
startpoint
endpoint

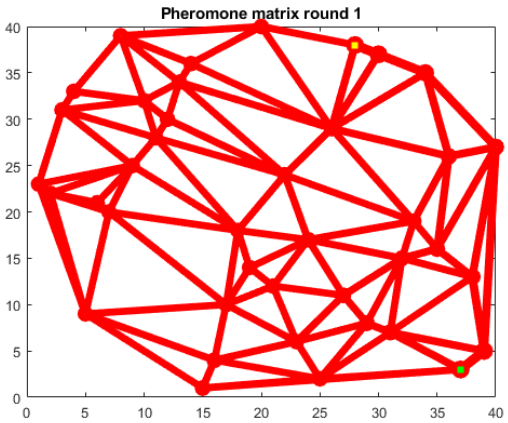


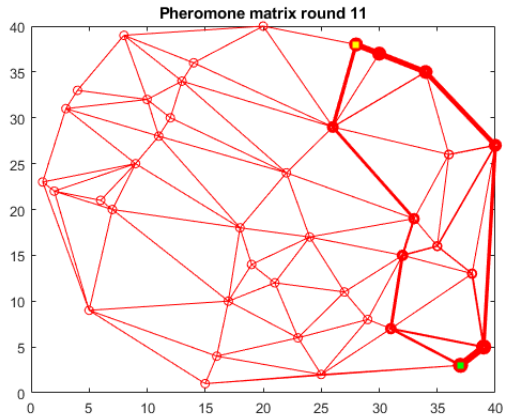


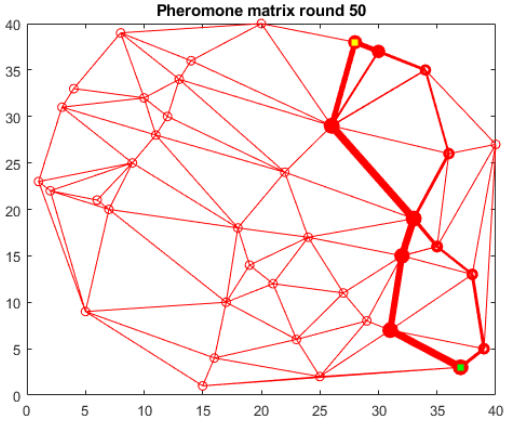
Shortest path:

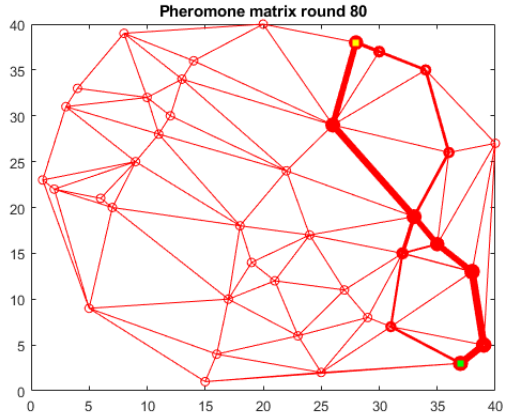


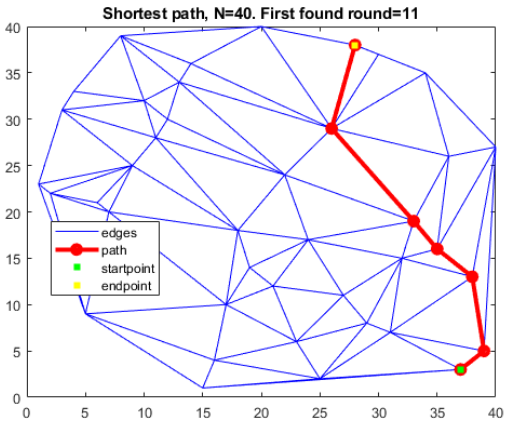
d-g)

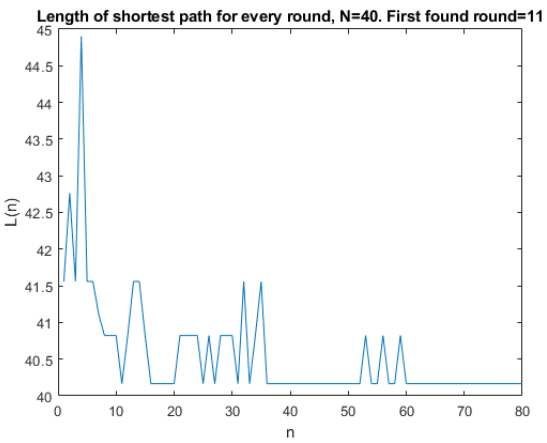

Pheromone matrix round 1


Pheromone matrix round 11


Pheromone matrix round 50

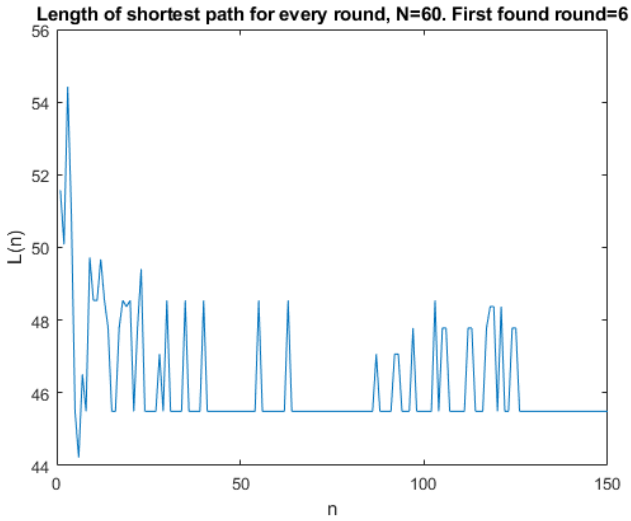

Pheromone matrix round 80

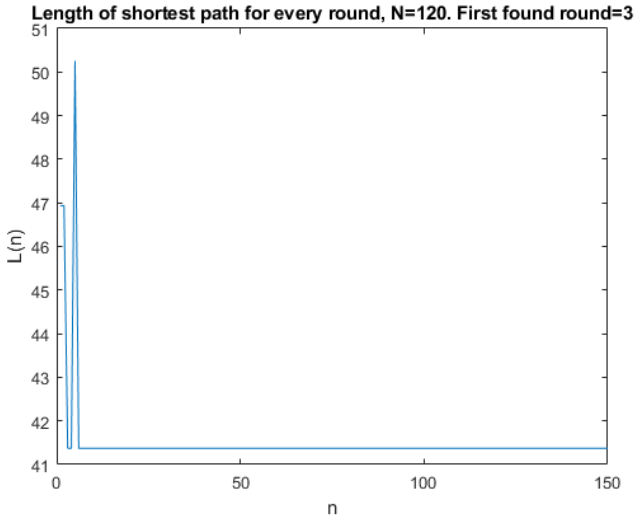

Shortest path, N=40. First found round=11
edges
path
startpoint
endpoint

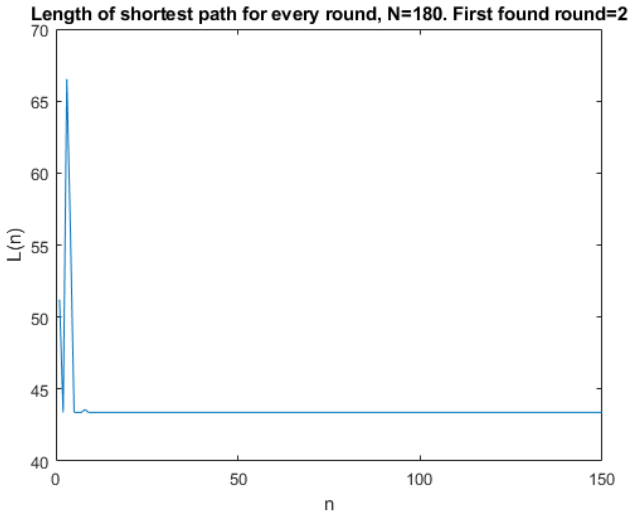

Length of shortest path for every round, N=40. First found round=11
L(n)
n

15.8

a)

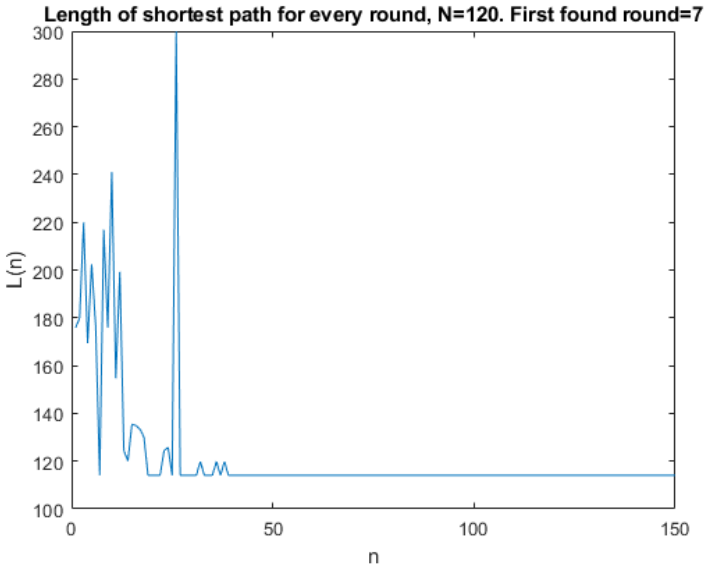






The three above plots are for the distance=40 between the starting point and end point.

When you increase N there is a possibility to increase the distance between the starting point and end point. I did so for N=120, I set the distance between the starting and end point to 100 and got this:

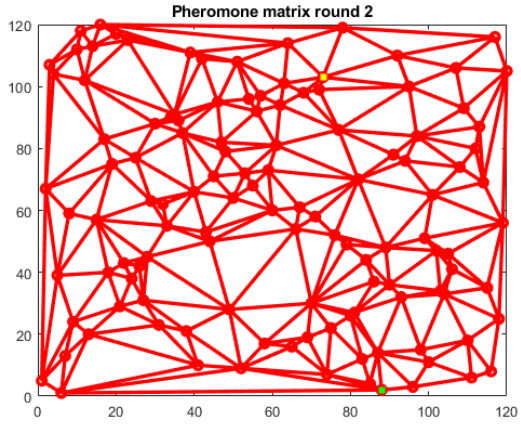


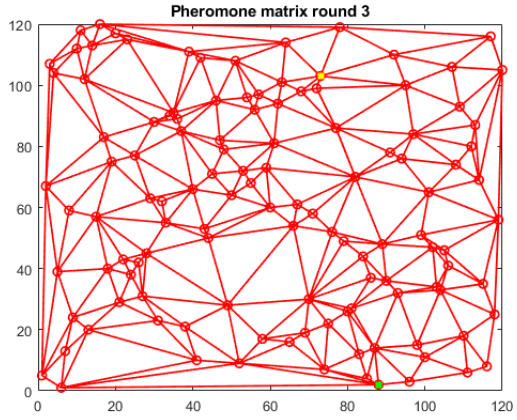
It gets a little more chaotic but not much.

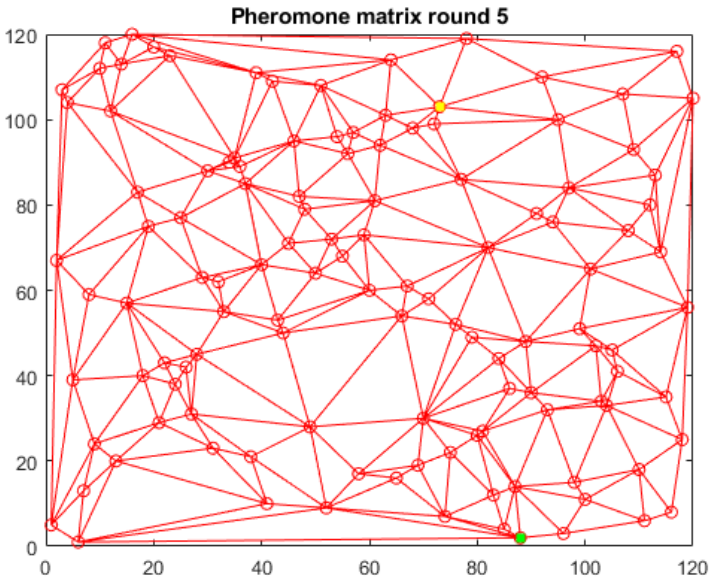
If we increase N and also the distance between starting point and ending point, the subsequent rounds after finding the shortest path is not hundred percent maintained.

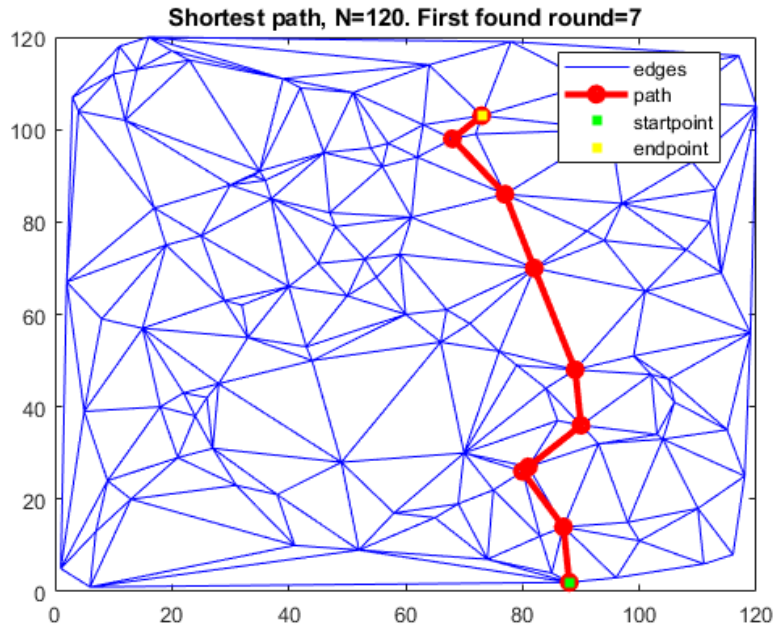
b)

This is for N=120 and the distance between starting and end point is 100:

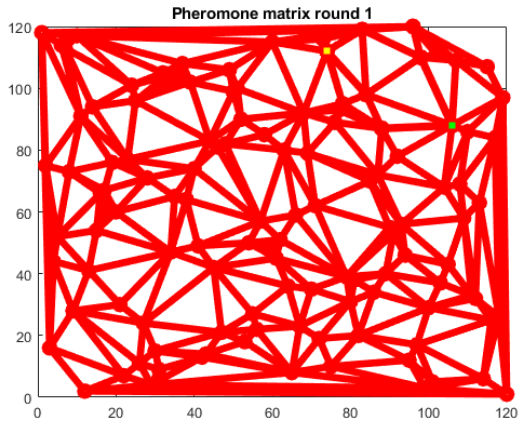


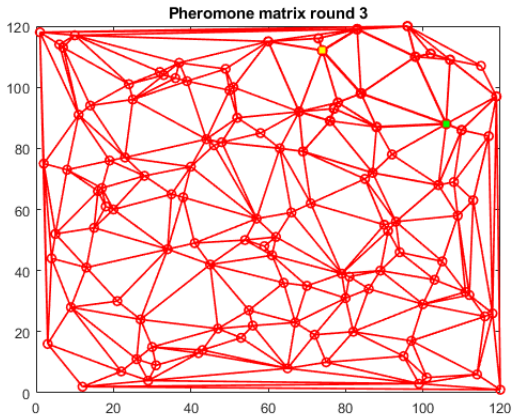


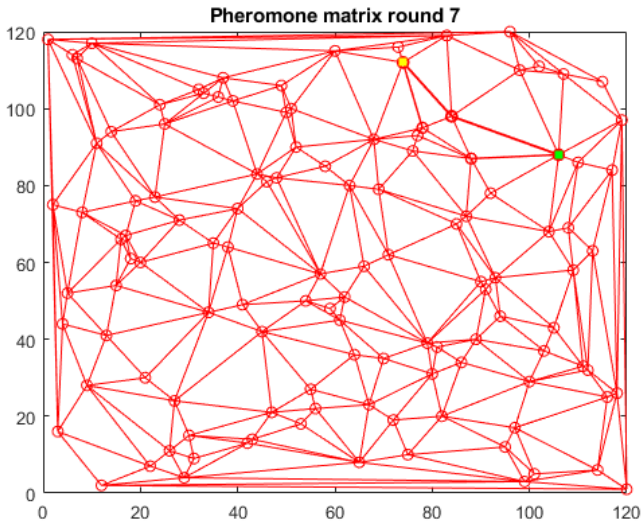


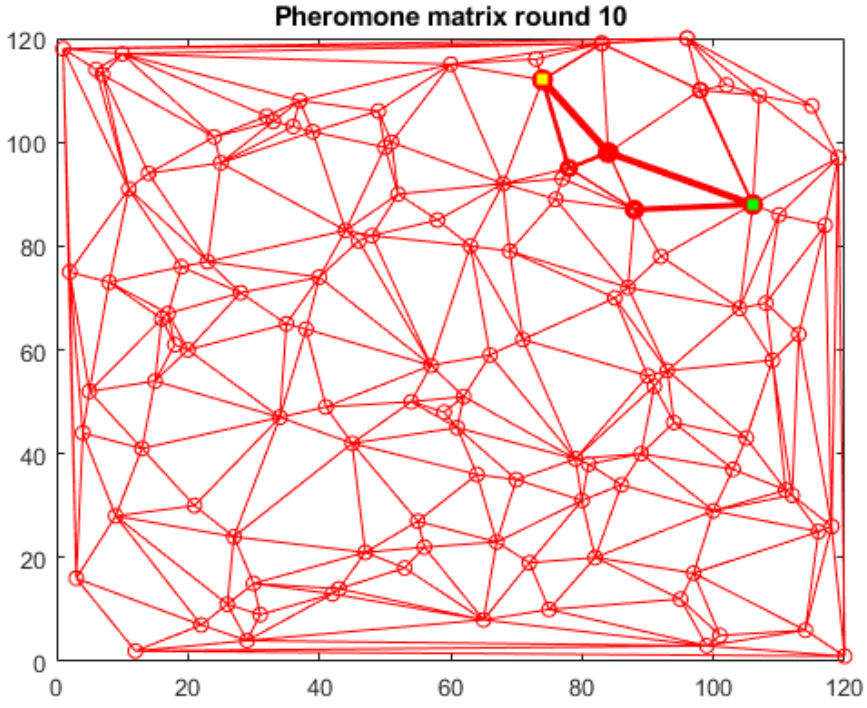


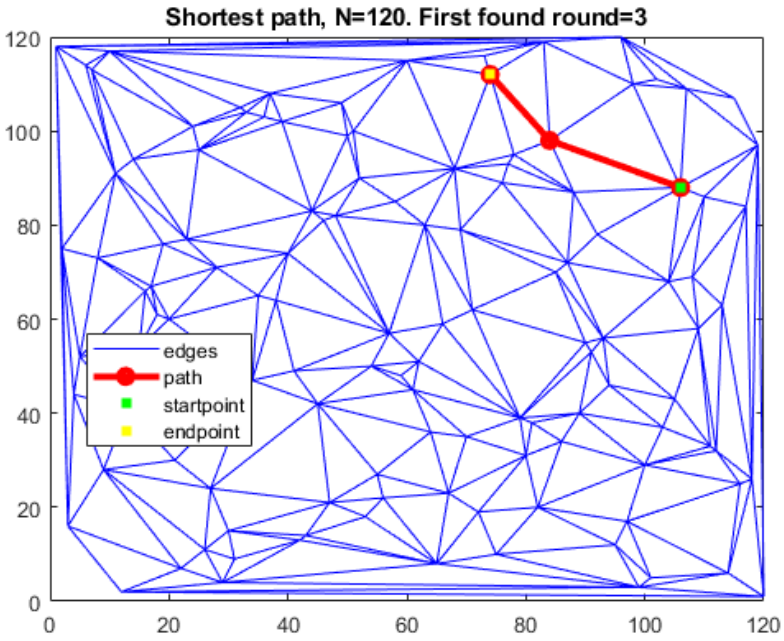
This is for N=120 and the distance between starting and end point is 40:


Pheromone matrix round 1
120
100
80
60
40
20
0
0
20
40
60
80
100
120

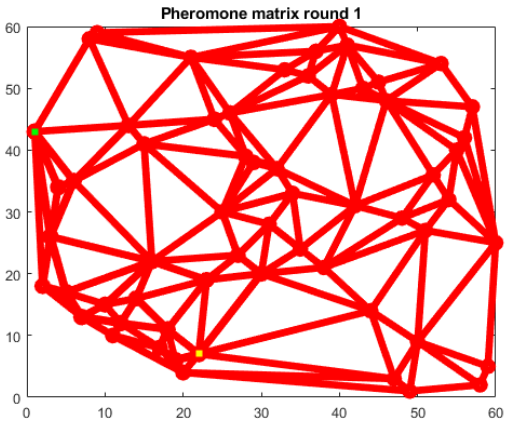

Pheromone matrix round 3
120
100
80
60
40
20
0
0
20
40
60
80
100
120

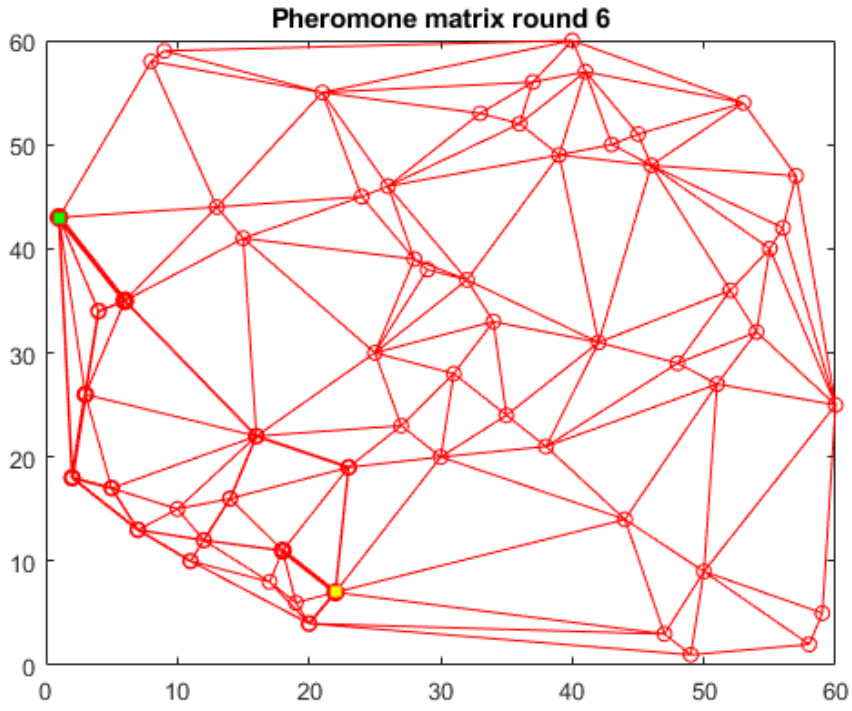

Pheromone matrix round 7
120
100
80
60
40
20
0
0
20
40
60
80
100
120

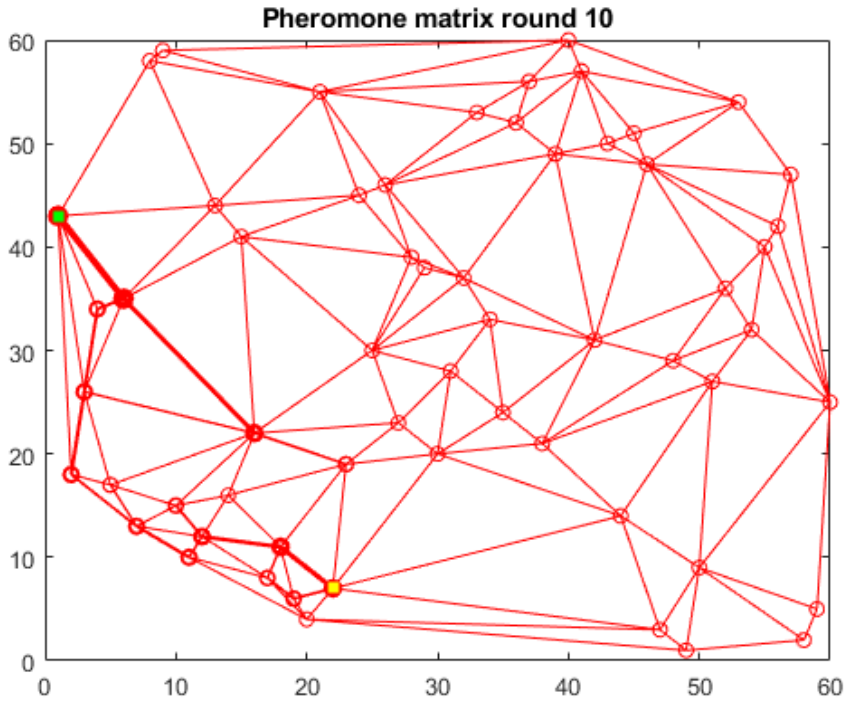

Pheromone matrix round 10
120
100
80
60
40
20
0
0
20
40
60
80
100
120

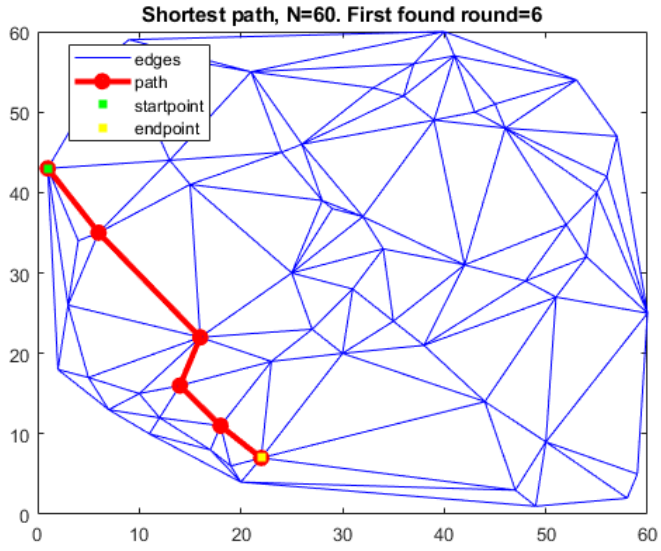


This is for N=60 and the distance between starting and end point is 40:

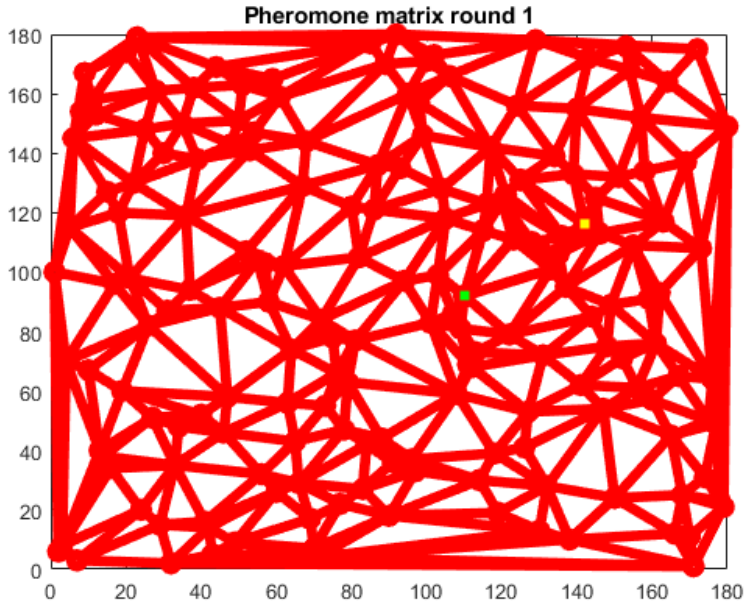

Pheromone matrix round 1
60
50
40
30
20
10
0
0
10
20
30
40
50
60

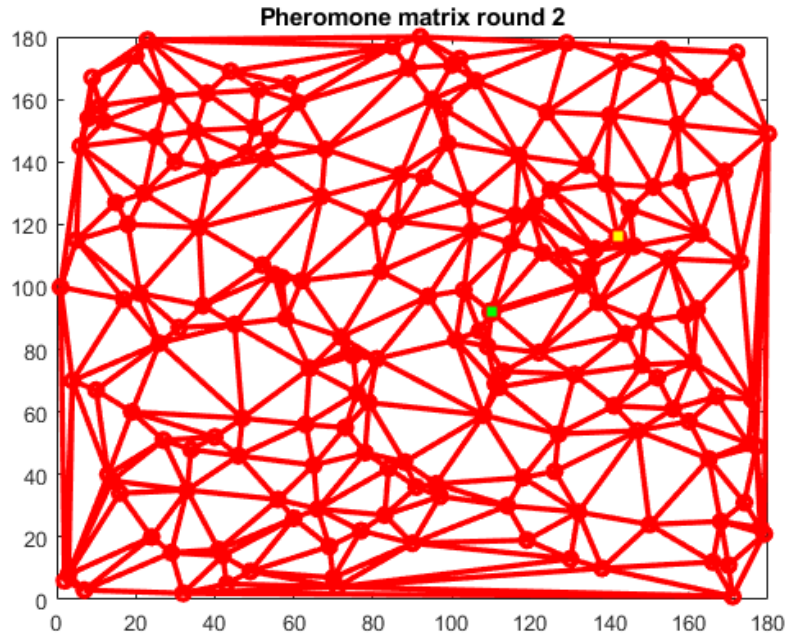

Pheromone matrix round 6
60
50
40
30
20
10
0
0
10
20
30
40
50
60

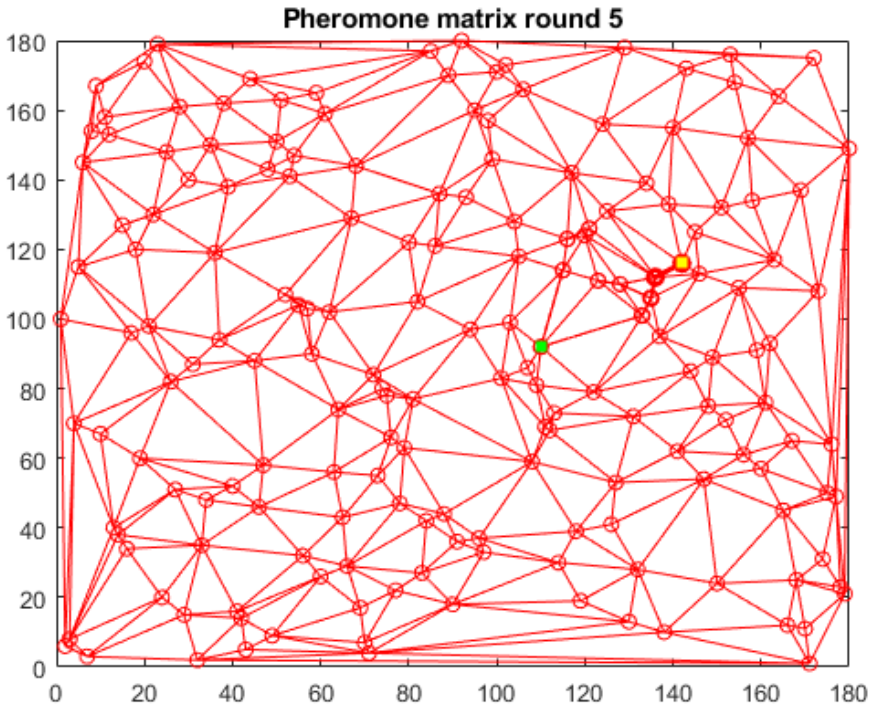


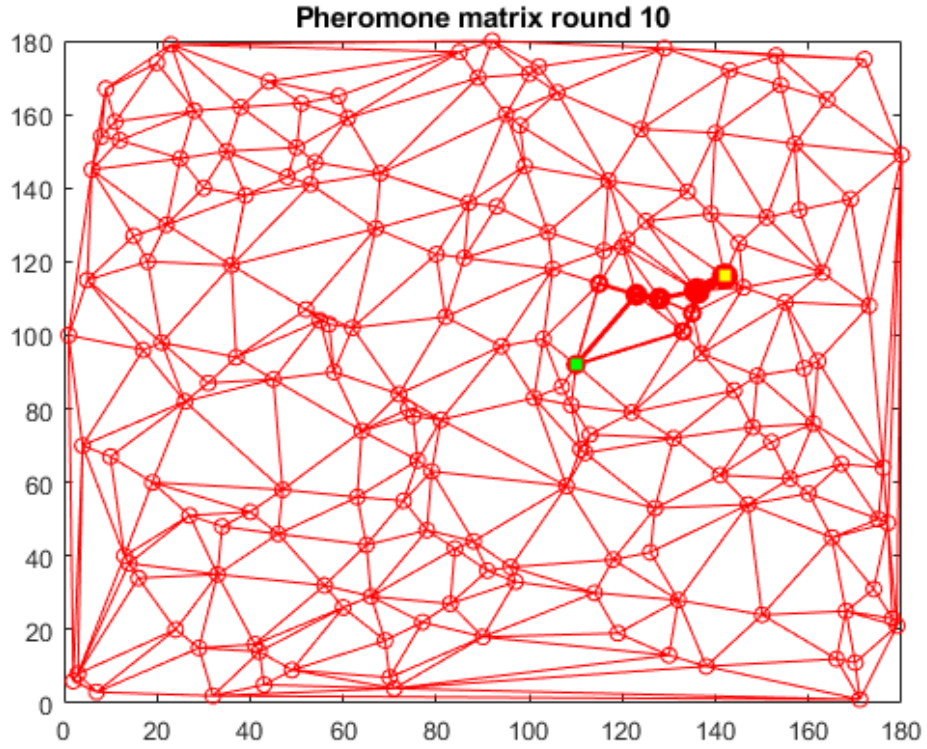


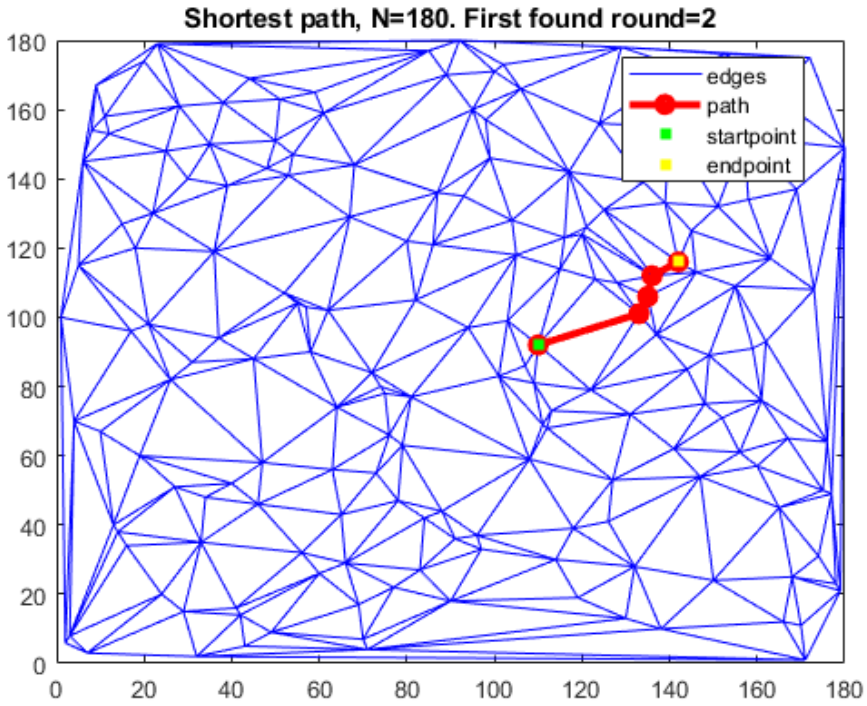
This is for N=180 and the distance between starting and end point is 40:


Pheromone matrix round 1
180
160
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120
140
160
180


Pheromone matrix round 2
180
160
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120
140
160
180


Pheromone matrix round 5
0
20
40
60
80
100
120
140
160
180


Pheromone matrix round 10
0
20
40
60
80
100
120
140
160
180



Question: Do the ants explore the full graph, or is the search, in practice, restricted only to a subset of the graph, especially after a few rounds? If the ants do not run over all the possible paths, how can we be sure that we have effectively found the minimum?

Answer: From my plots I can see that the pheromones are kind of similar on all edges, like there is no cluster where there is particular more pheromones. That might have to do

With that there is much more nodes that each ant can choose between than before.

Since there is more spread in the pheromones after all rounds than before (for instance N=40) there could be that there is a shorter path that never get found since the choice

of node at each step is based on the pheromone matrix. if the Pheromone matrix after some rounds converge to the shortest path found, we probably have found the global optimum

Though I found that the above is based on a starting point and end point with a distance between them of around 100. When I run for the distance 40 between starting and end point

which is similar for the last run with N=40 where the distance where around 30. Well with N= 120 and a distance between starting and end point of 40 I get that it's found the

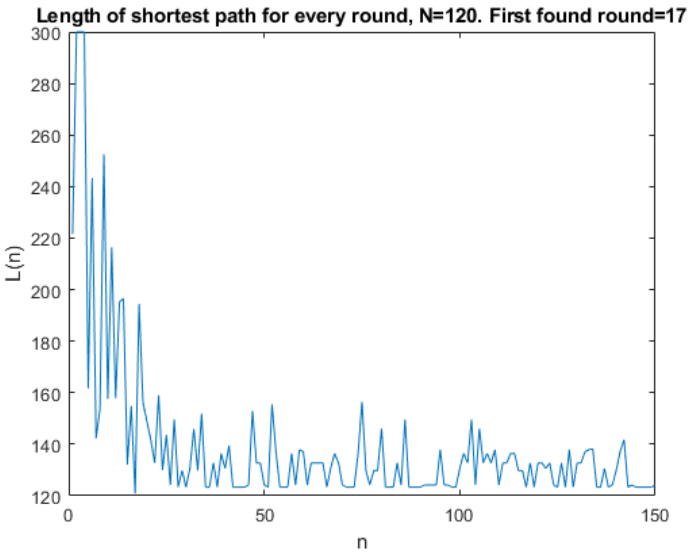
shortest path really fast. That might have to do with that the relative distance is shorter, I mean that there is less edges to choose from when the N is bigger with the same

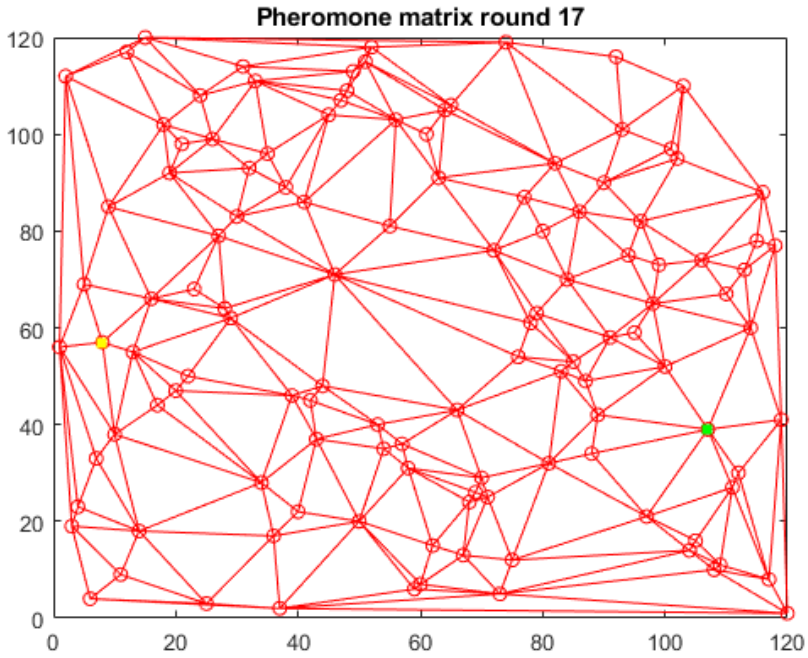
distance configuration between the starting and the end point. With this configuration I believe that we can be sure that the shortest path has been found.

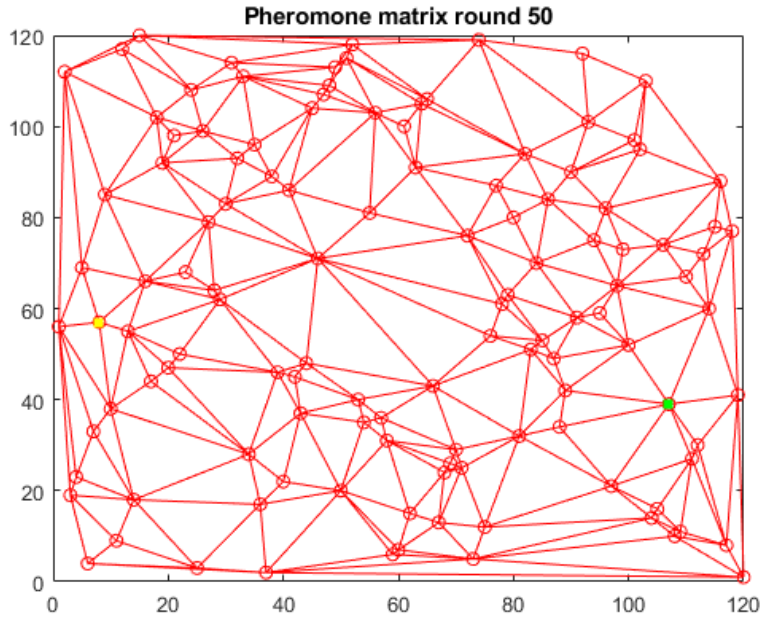
Comparing when I run for N=60 with the rest the same, I get comparing to N=120 and higher that it takes a little longer for it to find the shortest path.

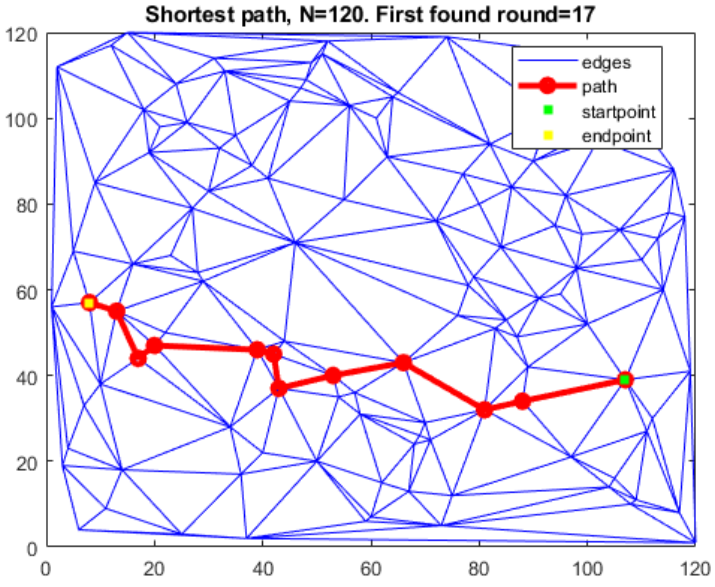
c)

for a lower alpha, alpha=0.5

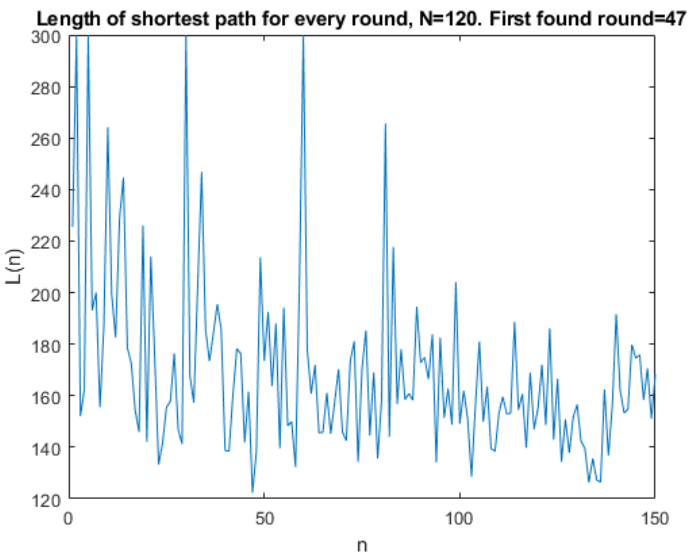


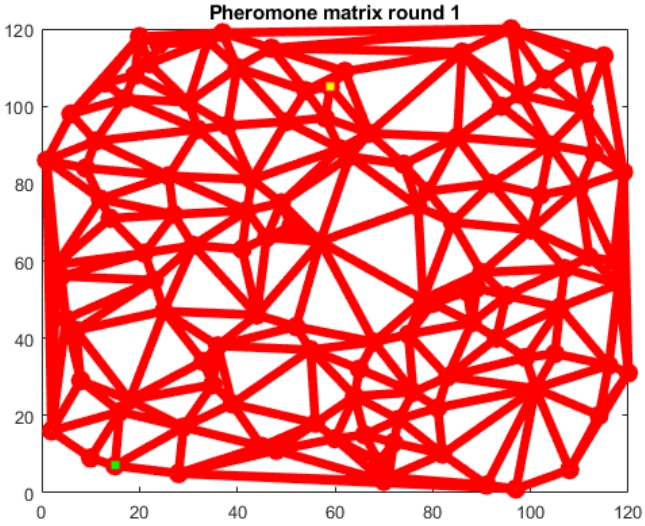


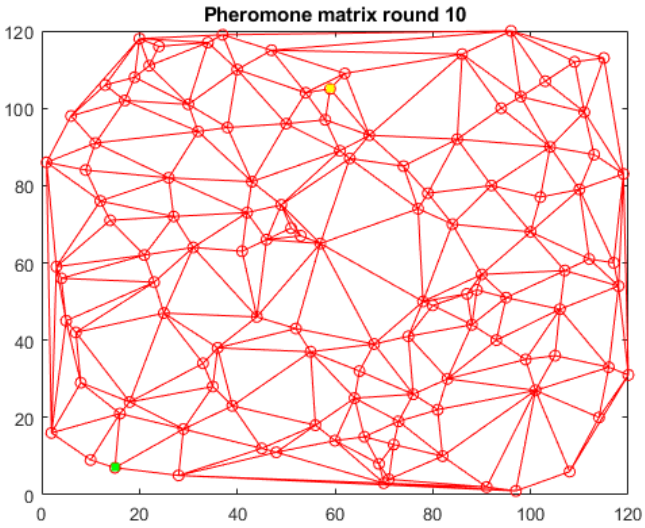

Pheromone matrix round 50
120
100
80
60
40
20
0
0
20
40
60
80
100
120

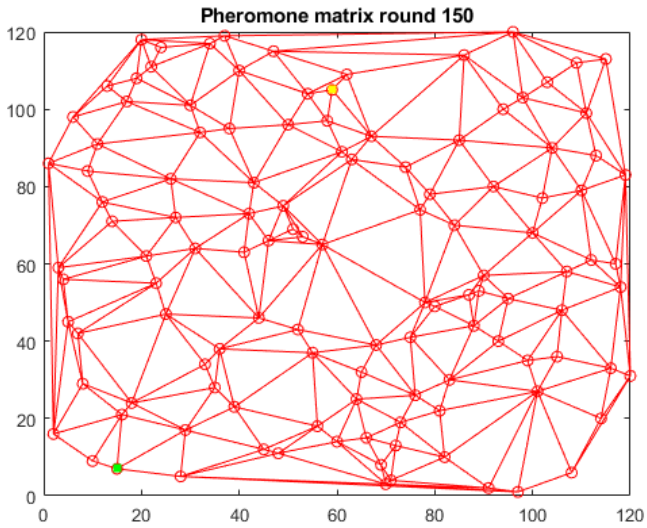

Shortest path, N=120. First found round=17
edges
path
startpoint
endpoint
120
100
80
60
40
20
0
0
20
40
60
80
100
120

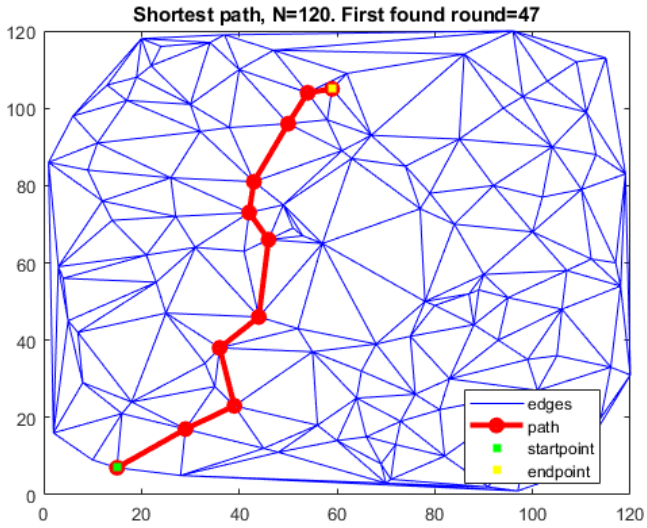
For alpha=0.1


Length of shortest path for every round, N=120. First found round=47
L(n)
n
300
280
260
240
220
200
180
160
140
120
0
50
100
150

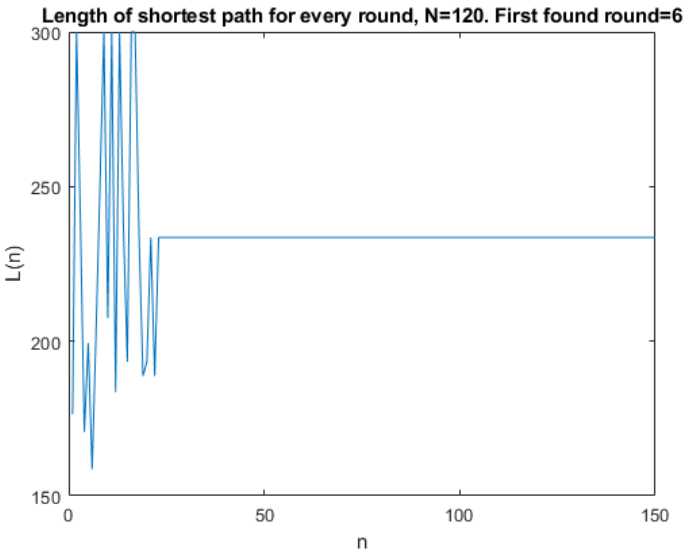

Pheromone matrix round 1
120
100
80
60
40
20
0
0
20
40
60
80
100
120


Pheromone matrix round 10
120
100
80
60
40
20
0
0
20
40
60
80
100
120

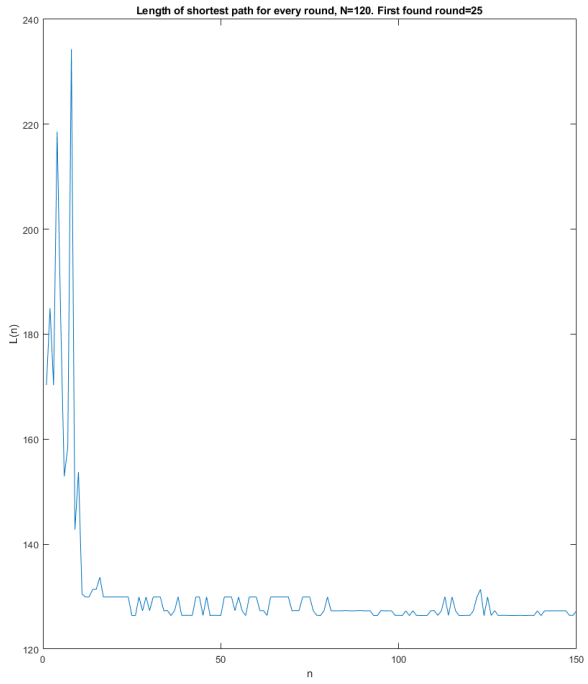




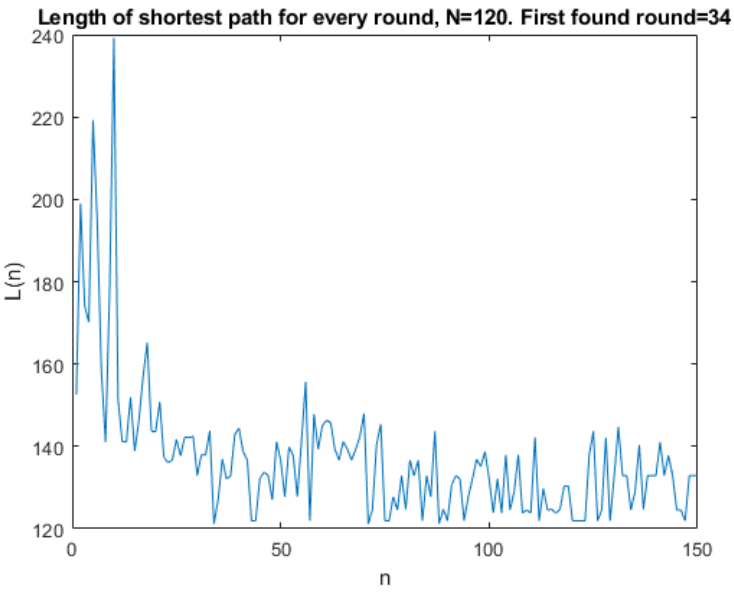
Increasing alpha, alpha=0.9



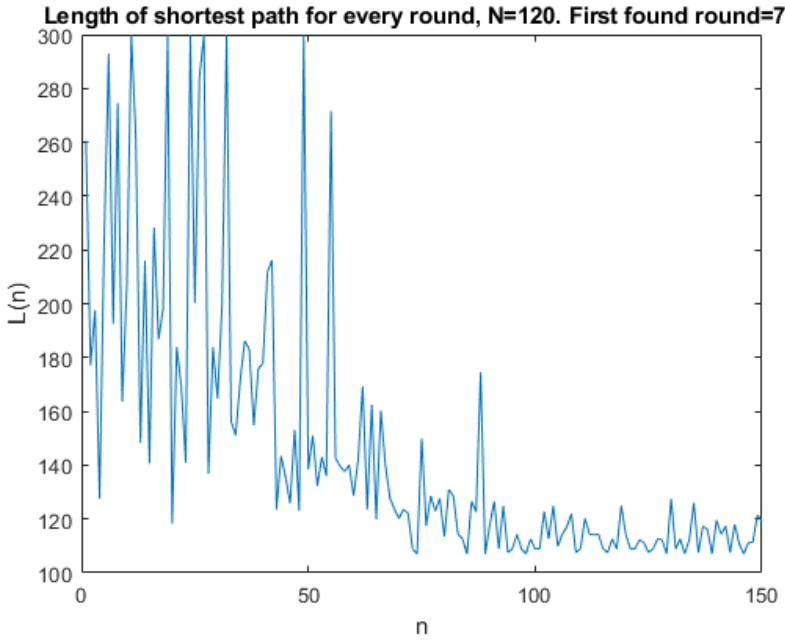
Decreasing beta, beta=0.5

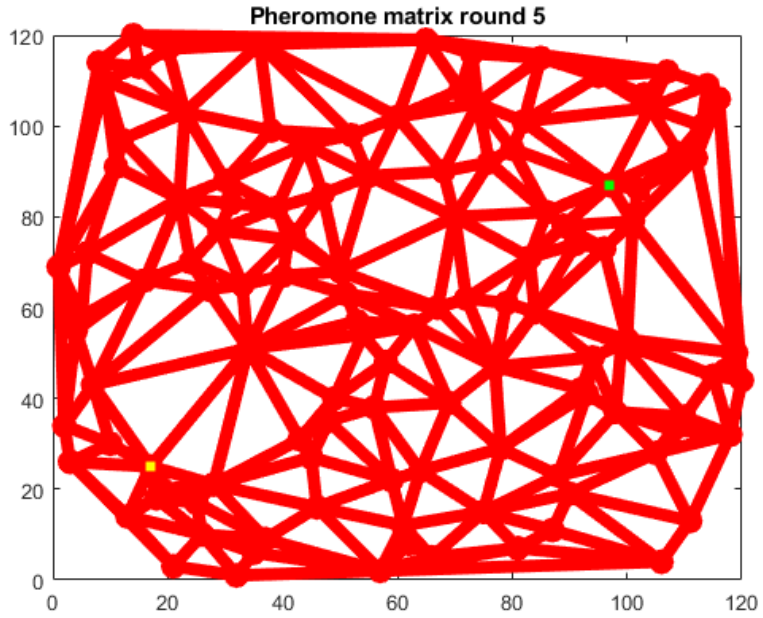


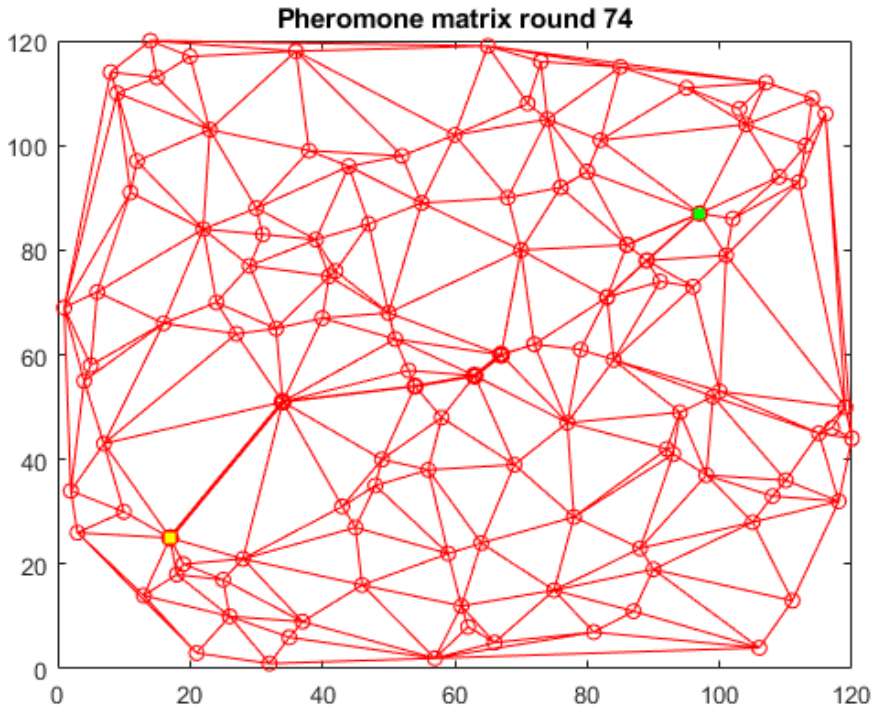
Decreasing beta and alpha to 0.5

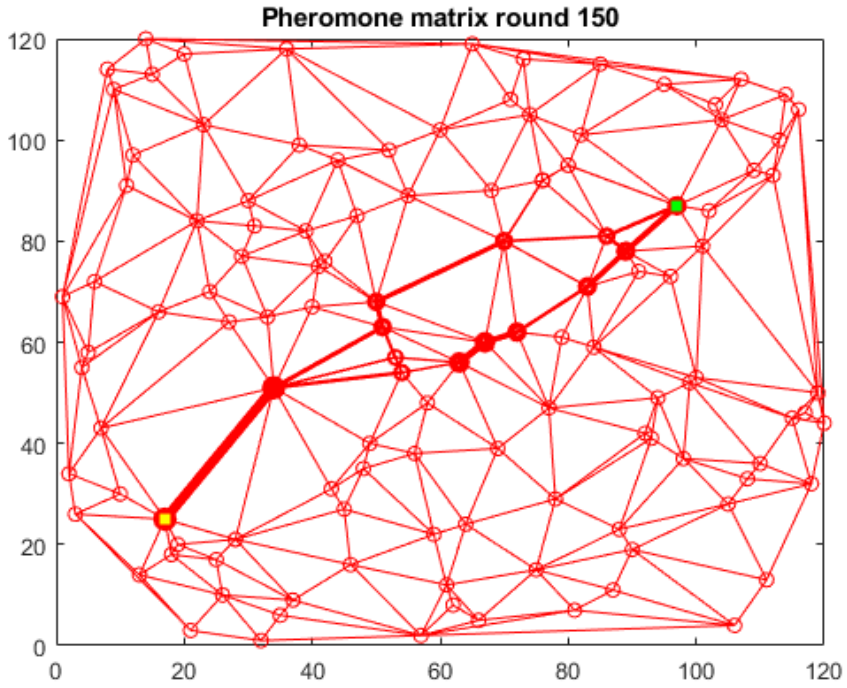


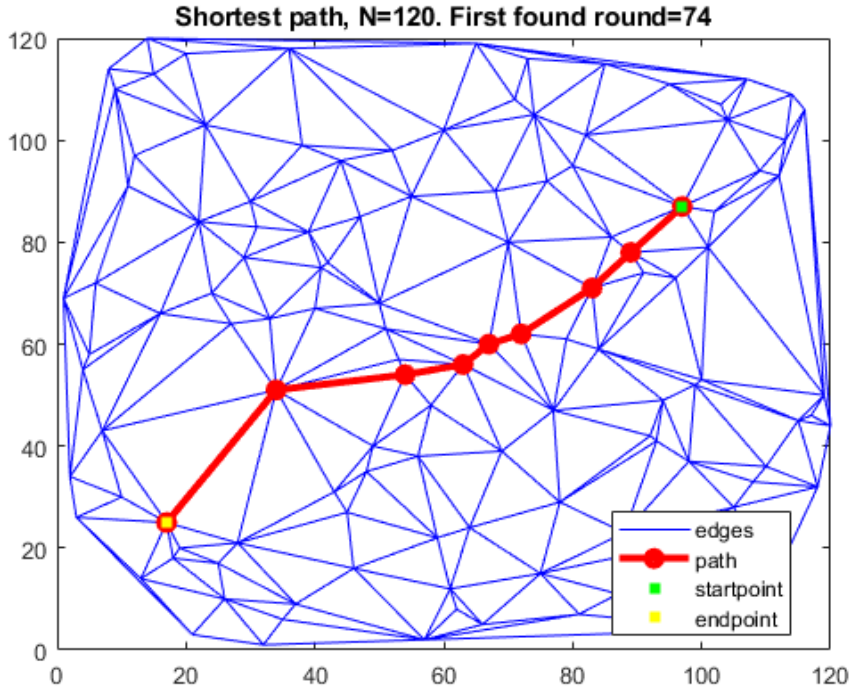
Decreasing rho, rho=0.1



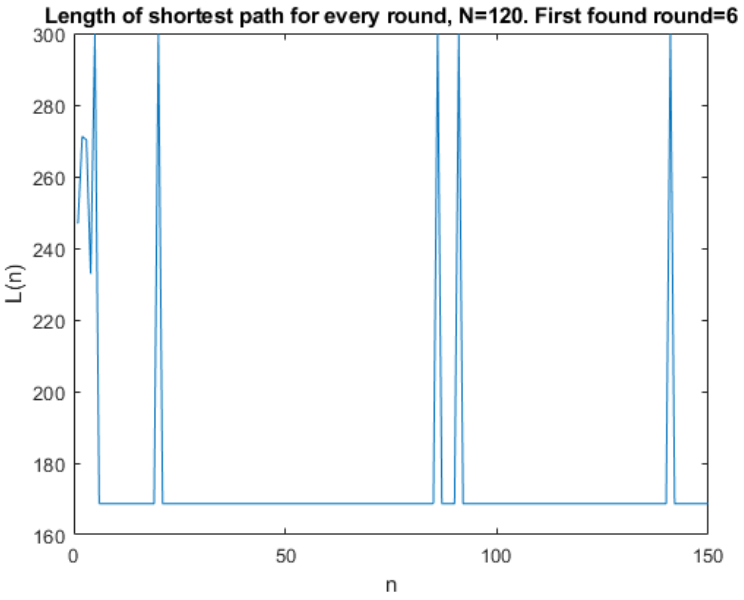

Pheromone matrix round 5
120
100
80
60
40
20
0
0
20
40
60
80
100
120


Pheromone matrix round 74
120
100
80
60
40
20
0
0
20
40
60
80
100
120





Increasing rho, rho=0.9



Question :

Do different choices give different shortest paths? Is some choice faster in finding the minimum?

Answer: One of the most interesting things or what make a significant difference was to decrease the rho. Since rho is a factor that decrease the pheromone level at every new round, it makes the pheromones decrease less fast and the shortest path found is found after more rounds than before. In the pheromone matrix plot, we can see more of a path of the pheromone level after a lot of rounds (150 rounds) which mirror the shortest path.

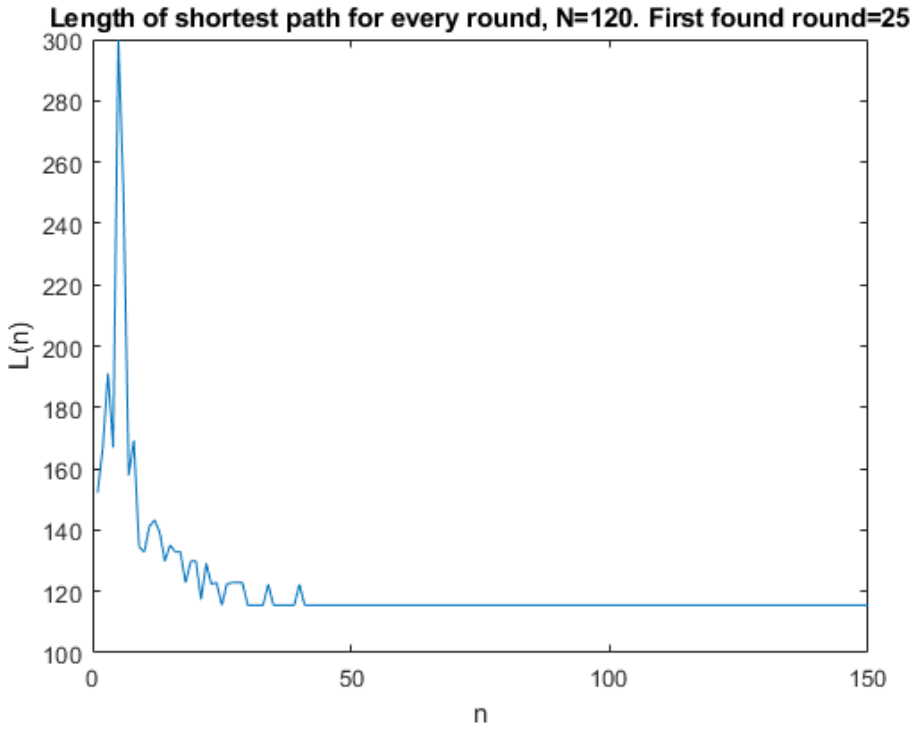
With a lower alpha we get a much noisier behavior of the shortest path length for every round. It shifts kind of chaotic.

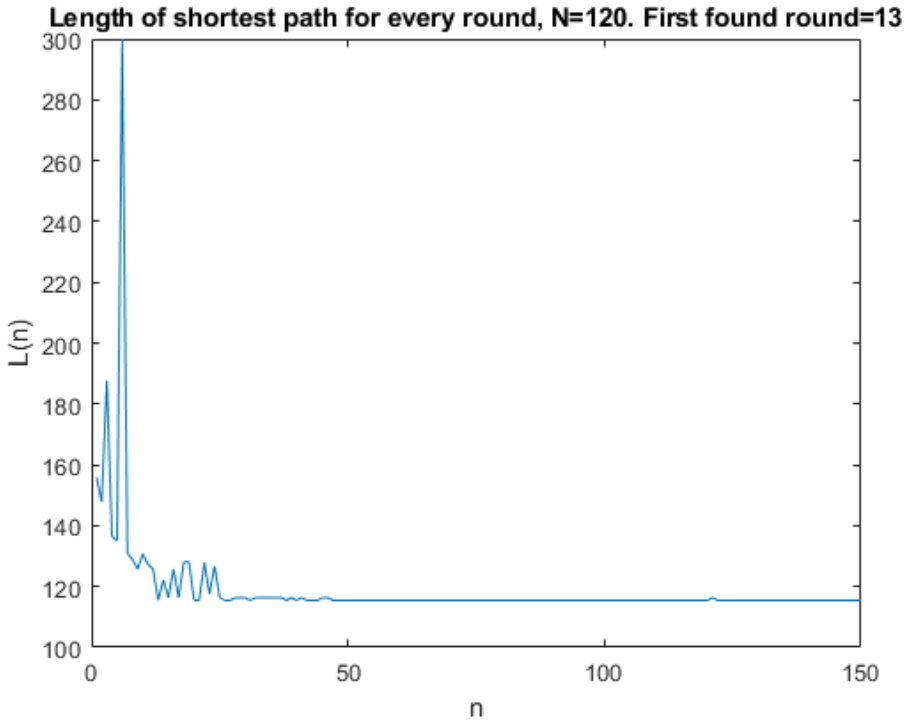
There is not so much difference when tuning the beta as comparing with changing rho.

Change alpha makes also a kind of big different.

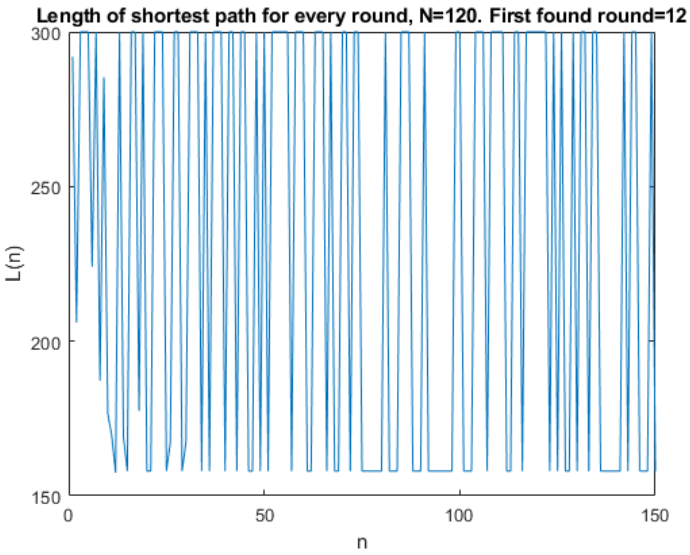
d)

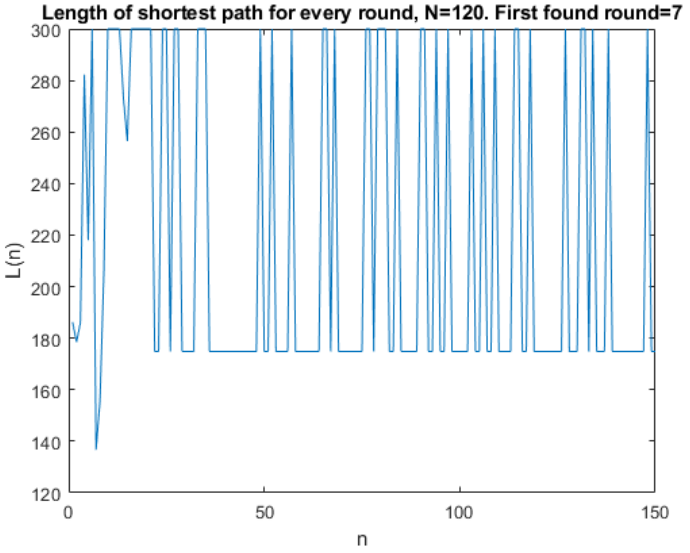
For the normal nr of ants which is 20 our results looks mostly like this:



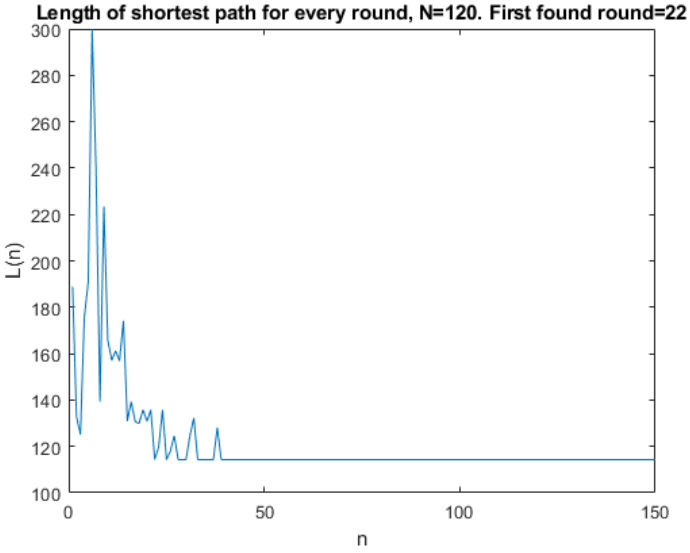


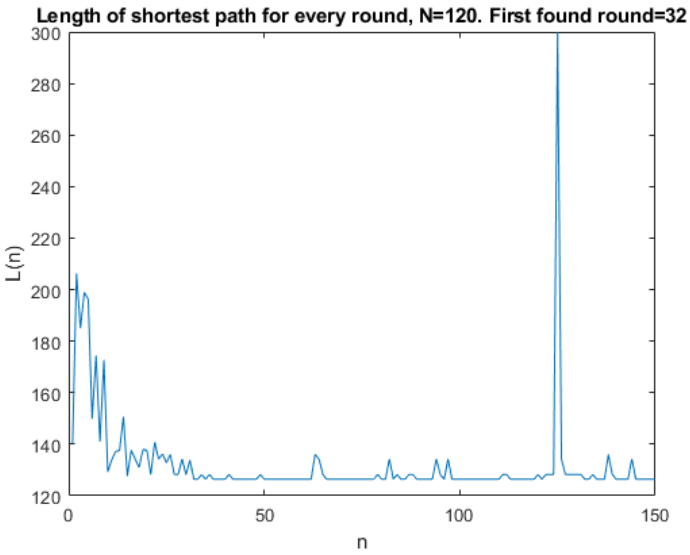
For the less nr of ants, ants=10 our results looks mostly like this:



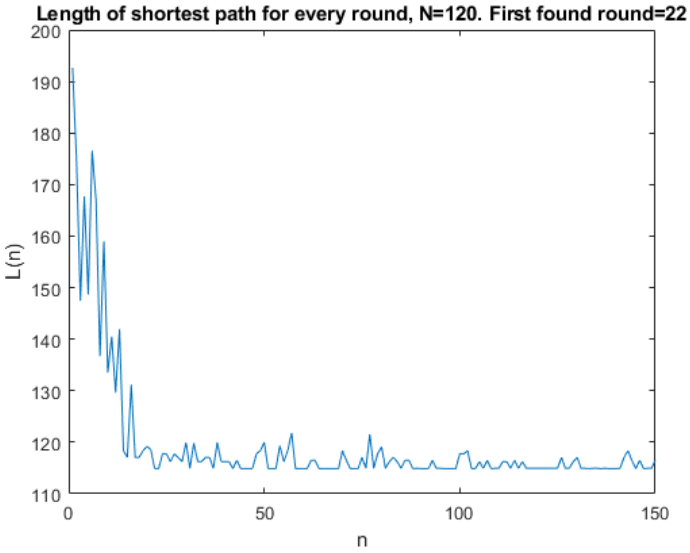


For the more nr of ants, ants = 0 our results looks mostly like this:

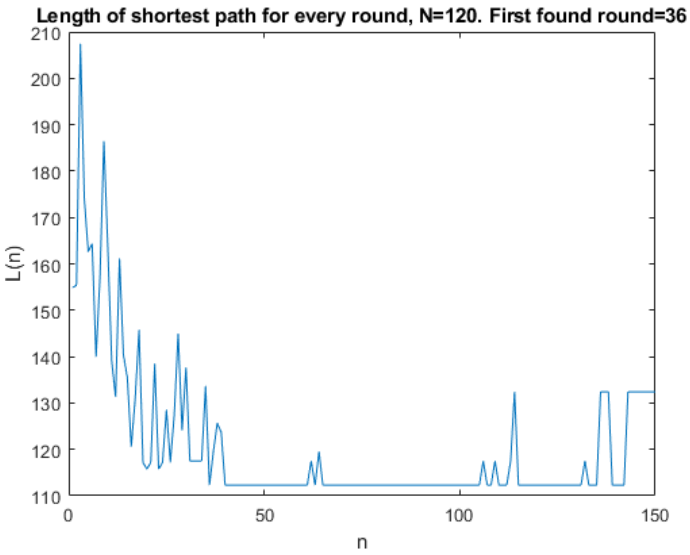




Ants=50:



Ants=70



Question: Try the simulation using a different number of ants A walking through the graph.

Does this make a difference to the algorithm? Do more ants result in a more or

less accurate algorithm? What about computational efficiency?

Answer: I would say that less ants gives more chaotic behavior when looking at plot of the length of the shortest path for every round. It also do not probably find the shortest

path since for the same configuration and a lot of different runs I get that the shortest path distance is higher for less ants(10 ants instead of 20 or 30)!

I also found that increasing the ants leads a little more to less fast find the shortest path. So I would say is less effective since it takes little longer for it to find

the shortest path. Maybe more ants would lead to a more accurate algorithm since there is more exploration, I mean there is more ants that explore trying to find the shortest path.

Code

```
clear all
close all
clc

N = 10;

% a)
Mmatrix = Matrix(N);
figure(1)
pcolor(Mmatrix)
title('connection matrix')


% b)
randomVertices = GenRandomVertices(N);
distances = Distances(Mmatrix, randomVertices);
figure(2)
pcolor(distances)
c = gray;
colormap(c)
grid on
colorbar
title('distances where white means infinity distance (no direct connections)')

% c)
weightMatrix = WeightMatrix(distances);
figure(3)
pcolor(weightMatrix)
title('Weight matrix')
colorbar
% d)
pheromoneMatrix = PheromoneMatrix(Mmatrix);
figure(4)
pcolor(pheromoneMatrix)
title('Pheromone matrix')
% e)
path = GenPath(2, Mmatrix);

% f)
pathLength = GetPathLengt(path, distances);

% g)
sPath = SimplifyPath(path, distances);

% delaunay plot
%
% figure(2)
```

```
% plot(x,y,'.','markersize',12)
% grid on
% hold on
% DT = delaunay(x,y);
% triplot(DT,x,y)

figure(5)
x = randomVertices(:,1);
y = randomVertices(:,2);
tri = delaunayTriangulation(x,y);
triplot(tri)

    clear all
close all
clc

%%% 15.7
N = 40;
maxSteps = 80;
nAnts = 20;
alpha = 0.8;
beta = 1.0;
rho = 0.5;
iterations = 80;
%
randomVertices = GenRandomVertices(N);
xVertices = randomVertices(:,1);
yVertices = randomVertices(:,2);
tri = delaunayTriangulation(xVertices, yVertices);
% triplot(tri)

Edges = edges(tri);

Mmatrix = zeros(N);
for i = 1:length(Edges)
    Mmatrix(Edges(i,1), Edges(i,2)) = 1;
end
Mmatrix = Mmatrix + Mmatrix';


% Calculating the distance
distanceMatrix = Distances(Mmatrix, randomVertices);
% figure(1)
% pcolor(distances)
% c = gray;
% colormap(c)
% grid on
% colorbar
% title('distances where white means infinity distance (no direct connections)')
```

```
% Calculating the weightmatrix
weightMatrix = WeightMatrix(distanceMatrix);
% figure(3)
% pcolor(weightMatrix)

% calculating the pheromone matrix
pheromoneMatrix = PheromoneMatrix(Mmatrix);
% figure(4)
% pcolor(pheromoneMatrix)

% Generate the path
path = GenPath(2, Mmatrix);

% Calculate the length of the path
pathLength = GetPathLengt(path, distanceMatrix);

% simplify the path
sPath = SimplifyPath(path, distanceMatrix);

%%%%%
% to avoid that starting point is to close to end point
distanceCondition = 0;
while distanceCondition < 10
    s0 = randi(N);
    t0 = randi(N);
    distanceCondition = pdist2(randomVertices(s0,:), randomVertices(t0,:));
end

%%% running the algorithm
pheromoneMatrices = {};
shortestPaths = {};
shortestPathDistances = [];
for iteration = 1:iterations
    % for all ants append its path to a cell array
    pheromoneMatrix = pheromoneMatrix .* (1-rho);
    paths = {};
    for i = 1:nAnts
        paths{i,1} = [];
        paths{i,1} = [paths{i,1},s0];
        for step = 1:maxSteps
            nextNode = BranchDecision(alpha, beta, paths{i,1}(end), weightMatrix, Mmatrix, pheron
            paths{i,1} = [paths{i,1},nextNode];
            if nextNode == t0   % break when reach target
                break
            end
        end
```

```
    end

    %%% Check which ants that arrive to the destination
    shortestPathDistance = 100;
    nReachedDestination = 0;
    arrivals = {};    % append all the ants that reach the destination
    arrivalsSimplified = {};    % append the simplified paths that reached
                                % The destination
    j = 1;  % Just a index to append correct
    for i = 1:nAnts
        iPath = paths{i,1};
        if ~isempty(find(iPath==t0))
            nReachedDestination = nReachedDestination+1;
            arrivals{j,1} = iPath;
            iPathSimplified = SimplifyPath(iPath,distanceMatrix);
            arrivalsSimplified{j,1} = iPathSimplified;
            j = j+1;

            iPathDistance = GetPathLengt(iPathSimplified,distanceMatrix);
            if iPathDistance < shortestPathDistance
                shortestPathDistance = iPathDistance;
                shortestPath = iPathSimplified;
            end
            % Update the pheromones
            for j = 1:length(iPathSimplified)-1
                pheromoneMatrix = UpdatePheromones(pheromoneMatrix, j, iPathSimplified, iPathDis
            end

        end
    end
    shortestPaths{iteration} = shortestPath;
    pheromoneMatrices{iteration} = pheromoneMatrix;
    shortestPathDistances(end+1) = shortestPathDistance;
end


%%% Plot all ants that reached the destination
% for k = 1:length(arrivals)
%     arrivalPath = arrivals{k};
%     figure(k)
%     triplot(tri)
%     hold on
%     plotx = randomVertices(arrivalPath,1);
%     ploty = randomVertices(arrivalPath,2);
%     plot(plotx,ploty,'-o','LineWidth',3)
%     scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
%     scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
%     legend('edges','path','startpoint','endpoint','Location','best')
```

```
%     title('Not simplified path')

%     %%% Saving the paths figures in a certain folder
%     folder = 'C:\Users\axelq\OneDrive\Skrivbord\MpCas\Simulation of complex systems\simulation-
%     baseFileName = sprintf('Figure %d.png', k);
%     fullFileName = fullfile(folder, baseFileName);
%     saveas(figure(k),fullFileName)
% end

%%% Plot the above paths when they are simplified
% j = 1;
% for k = length(arrivals)+1:length(arrivalsSimplified)+length(arrivals)
%     arrivalPathSimplified = arrivalsSimplified{j};
%     figure(k)
%     triplot(tri)
%     hold on
%     plotx = randomVertices(arrivalPathSimplified,1);
%     ploty = randomVertices(arrivalPathSimplified,2);
%     plot(plotx,ploty,'-o','LineWidth',3)
%     scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
%     scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
%     legend('edges','path','startpoint','endpoint','Location','best')
%     title('Simplified path')
%
% %     %%% Saving the Simplified paths figures in a certain folder
% %     folder = 'C:\Users\axelq\OneDrive\Skrivbord\MpCas\Simulation of complex systems\simulatio
% %     baseFileName = sprintf('SimplifiedFigure %d.png', j);
% %     fullFileName = fullfile(folder, baseFileName);
% %     saveas(figure(k),fullFileName)
% %     j = j+1;
% end




% iPheromoneMatrix = 1;   %iterator
% for h = k+1:length(pheromoneMatrices)+k
%     figure(h)
%     triplot(tri)
%     hold on
%     pher = pheromoneMatrices{iPheromoneMatrix};
%     for i = 1:N
%         for j = 1:N
%             if pher(i, j) > 0.1
%                 iPlotx = randomVertices(i,1);
%                 iPloty = randomVertices(i,2);
%                 jPlotx = randomVertices(j,1);
%                 jPloty = randomVertices(j,2);
```

```
%                 plot([randomVertices(i,1),randomVertices(j,1)],[randomVertices(i,2),randomVerti
%             end
%         end
%     end
%     iPheromoneMatrix = iPheromoneMatrix + 1;
% end

%%
clc
figure(1)
plot(1:iterations,shortestPathDistances)
ylabel('L(n)')
xlabel('n')
title('Length of shortest path for every round')
index = find(shortestPathDistances==min(shortestPathDistances));
firstShortestPathIndex = index(1);
firstShortestPath = shortestPaths{firstShortestPathIndex};

figure(2)
triplot(tri)
hold on
plotx = randomVertices(firstShortestPath,1);
ploty = randomVertices(firstShortestPath,2);
plot(plotx,ploty,'-o','Color','r','LineWidth',3)
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
legend('edges','path','startpoint','endpoint','Location','best')
title('Shortest path')

iPheromoneMatrix = 1;   %iterator
for h = 3:length(pheromoneMatrices)+2
    figure(h)
    triplot(tri)
    hold on
    pher = pheromoneMatrices{iPheromoneMatrix};
    for i = 1:N
        for j = 1:N
            if pher(i, j) > 0.3
                iPlotx = randomVertices(i,1);
                iPloty = randomVertices(i,2);
                jPlotx = randomVertices(j,1);
                jPloty = randomVertices(j,2);
                plot([randomVertices(i,1),randomVertices(j,1)],[randomVertices(i,2),randomVertice
                title('Pheromone matrix')
            end
        end
    end
    iPheromoneMatrix = iPheromoneMatrix + 1;
```

```
end

    clear all
close all
clc

%%% 15.7
N = 120;
maxSteps = 80;
nAnts = 20;
alpha = 0.8;
beta = 0.5;
rho = 0.5;
iterations = 150;
%
randomVertices = GenRandomVertices(N);
xVertices = randomVertices(:,1);
yVertices = randomVertices(:,2);
tri = delaunayTriangulation(xVertices, yVertices);

Edges = edges(tri);

Mmatrix = zeros(N);
for i = 1:length(Edges)
    Mmatrix(Edges(i,1), Edges(i,2)) = 1;
end
Mmatrix = Mmatrix + Mmatrix';


% Calculating the distance
distanceMatrix = Distances(Mmatrix, randomVertices);

% Calculating the weightmatrix
weightMatrix = WeightMatrix(distanceMatrix);

% calculating the pheromone matrix
pheromoneMatrix = PheromoneMatrix(Mmatrix);

%%%%%
% to avoid that starting point is to close to end point
distanceCondition = 0;
while distanceCondition < 100 || distanceCondition > 110
    s0 = randi(N);
    t0 = randi(N);
    distanceCondition = pdist2(randomVertices(s0,:), randomVertices(t0,:));
end

%%% running the algorithm
pheromoneMatrices = {};
```

```
shortestPaths = {};
shortestPathDistances = [];
for iteration = 1:iterations
    % for all ants append its path to a cell array
    pheromoneMatrix = pheromoneMatrix .* (1-rho);
    paths = {};
    for i = 1:nAnts
        paths{i,1} = [];
        paths{i,1} = [paths{i,1},s0];
        for step = 1:maxSteps
            nextNode = BranchDecision(alpha, beta, paths{i,1}(end), weightMatrix, Mmatrix, pheron
            paths{i,1} = [paths{i,1},nextNode];
            if nextNode == t0   % break when reach target
                break
            end
        end
    end

    %%% Check which ants that arrive to the destination
    shortestPathDistance = 300;
    nReachedDestination = 0;
    arrivals = {};    % append all the ants that reach the destination
    arrivalsSimplified = {};    % append the simplified paths that reached
                                % The destination
    j = 1;  % Just a index to append correct
    for i = 1:nAnts
        iPath = paths{i,1};
        if ~isempty(find(iPath==t0))
            nReachedDestination = nReachedDestination+1;
            arrivals{j,1} = iPath;
            iPathSimplified = SimplifyPath(iPath,distanceMatrix);
            arrivalsSimplified{j,1} = iPathSimplified;
            j = j+1;

            iPathDistance = GetPathLengt(iPathSimplified,distanceMatrix);
            if iPathDistance < shortestPathDistance
                shortestPathDistance = iPathDistance;
                shortestPath = iPathSimplified;
            end
            % Update the pheromones
            for j = 1:length(iPathSimplified)-1
                pheromoneMatrix = UpdatePheromones(pheromoneMatrix, j, iPathSimplified, iPathDist
            end

        end
    end
    shortestPaths{iteration} = shortestPath;
    pheromoneMatrices{iteration} = pheromoneMatrix;
```

```
    shortestPathDistances(end+1) = shortestPathDistance;
end


%%% Plot all ants that reached the destination
% for k = 1:length(arrivals)
%     arrivalPath = arrivals{k};
%     figure(k)
%     triplot(tri)
%     hold on
%     plotx = randomVertices(arrivalPath,1);
%     ploty = randomVertices(arrivalPath,2);
%     plot(plotx,ploty,'-o','LineWidth',3)
%     scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
%     scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
%     legend('edges','path','startpoint','endpoint','Location','best')
%     title('Not simplified path')

%     %%% Saving the paths figures in a certain folder
%     folder = 'C:\Users\axelq\OneDrive\Skrivbord\MpCas\Simulation of complex systems\simulation-
%     baseFileName = sprintf('Figure %d.png', k);
%     fullFileName = fullfile(folder, baseFileName);
%     saveas(figure(k),fullFileName)
% end

%%% Plot the above paths when they are simplified
% j = 1;
% for k = length(arrivals)+1:length(arrivalsSimplified)+length(arrivals)
%     arrivalPathSimplified = arrivalsSimplified{j};
%     figure(k)
%     triplot(tri)
%     hold on
%     plotx = randomVertices(arrivalPathSimplified,1);
%     ploty = randomVertices(arrivalPathSimplified,2);
%     plot(plotx,ploty,'-o','LineWidth',3)
%     scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
%     scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
%     legend('edges','path','startpoint','endpoint','Location','best')
%     title('Simplified path')
%
% %     %%% Saving the Simplified paths figures in a certain folder
% %     folder = 'C:\Users\axelq\OneDrive\Skrivbord\MpCas\Simulation of complex systems\simulatio
% %     baseFileName = sprintf('SimplifiedFigure %d.png', j);
% %     fullFileName = fullfile(folder, baseFileName);
% %     saveas(figure(k),fullFileName)
% %     j = j+1;
% end
```

```
% iPheromoneMatrix = 1;   %iterator
% for h = k+1:length(pheromoneMatrices)+k
%     figure(h)
%     triplot(tri)
%     hold on
%     pher = pheromoneMatrices{iPheromoneMatrix};
%     for i = 1:N
%         for j = 1:N
%             if pher(i, j) > 0.1
%                 iPlotx = randomVertices(i,1);
%                 iPloty = randomVertices(i,2);
%                 jPlotx = randomVertices(j,1);
%                 jPloty = randomVertices(j,2);
%                 plot([randomVertices(i,1),randomVertices(j,1)],[randomVertices(i,2),randomVerti
%             end
%         end
%     end
%     iPheromoneMatrix = iPheromoneMatrix + 1;
% end

% index = find(shortestPathDistances==min(shortestPathDistances));
% firstShortestPathIndex = index(1)
% firstShortestPath = shortestPaths{firstShortestPathIndex};
% clc
% figure(1)
% plot(1:iterations,shortestPathDistances)
% ylabel('L(n)')
% xlabel('n')
% msg1 = sprintf('Length of shortest path for every round, N=%d. First found round=%d',N,firstSho
% title(msg1)
%
%
% figure(2)
% triplot(tri)
% hold on
% plotx = randomVertices(firstShortestPath,1);
% ploty = randomVertices(firstShortestPath,2);
% plot(plotx,ploty,'-o','Color','r','LineWidth',3)
% scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
% scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
% legend('edges','path','startpoint','endpoint','Location','best')
% msg2 = sprintf('Shortest path, N=%d. First found round=%d',N, firstShortestPathIndex);
% title(msg2)
```

```
% find the first shorthes path
index = find(shortestPathDistances==min(shortestPathDistances));
firstShortestPathIndex = index(1)
firstShortestPath = shortestPaths{firstShortestPathIndex};

% Plot the shortest paths length as fun of nr of rounds
figure(1)
plot(1:iterations,shortestPathDistances)
ylabel('L(n)')
xlabel('n')
msg1 = sprintf('Length of shortest path for every round, N=%d. First found round=%d',N,firstShort
title(msg1)


% Plot the shortest path found
figure(2)
triplot(tri)
hold on
plotx = randomVertices(firstShortestPath,1);
ploty = randomVertices(firstShortestPath,2);
plot(plotx,ploty,'-o','Color','r','LineWidth',3)
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
legend('edges','path','startpoint','endpoint','Location','best')
msg2 = sprintf('Shortest path, N=%d. First found round=%d',N, firstShortestPathIndex);
title(msg2)



% Plot the phermoneMatrix at round 1
pheromoneMatrix = pheromoneMatrices{1};
figure(3)
for i = 1:N
    for j = 1:N
        if Mmatrix(i,j) == 1
        plot([randomVertices(i,1),randomVertices(j,1)], ...
            [randomVertices(i,2),randomVertices(j,2)],'-o','Color','r','LineWidth',pheromoneMatri
        hold on
        end
    end
end
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
msg3 = sprintf('Pheromone matrix round %d',1);
title(msg3)

% Plot the phermoneMatrix at first round when the shortest path is found
pheromoneMatrix = pheromoneMatrices{firstShortestPathIndex};
```

```
figure(4)
for i = 1:N
    for j = 1:N
        if Mmatrix(i,j) == 1
        plot([randomVertices(i,1),randomVertices(j,1)], ...
            [randomVertices(i,2),randomVertices(j,2)],'-o','Color','r','LineWidth',pheromoneMatri
        hold on
        end
    end
end
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
msg4 = sprintf('Pheromone matrix round %d',firstShortestPathIndex);
title(msg4)


% Plot the phermoneMatrix at round 50
pheromoneMatrix = pheromoneMatrices{50};
figure(5)
for i = 1:N
    for j = 1:N
        if Mmatrix(i,j) == 1
        plot([randomVertices(i,1),randomVertices(j,1)], ...
            [randomVertices(i,2),randomVertices(j,2)],'-o','Color','r','LineWidth',pheromoneMatri
        hold on
        end
    end
end
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
msg5 = sprintf('Pheromone matrix round %d',50);
title(msg5)

pheromoneMatrix = pheromoneMatrices{150};
figure(6)
for i = 1:N
    for j = 1:N
        if Mmatrix(i,j) == 1
        plot([randomVertices(i,1),randomVertices(j,1)], ...
            [randomVertices(i,2),randomVertices(j,2)],'-o','Color','r','LineWidth',pheromoneMatri
        hold on
        end
    end
end
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
msg6 = sprintf('Pheromone matrix round %d',150);
title(msg6)
```

```
%% subsection to plot different pheromone matrixes from all the stored ones from the simulation
pheromoneMatrix = pheromoneMatrices{80};
figure(7)
for i = 1:N
    for j = 1:N
        if Mmatrix(i,j) == 1
        plot([randomVertices(i,1),randomVertices(j,1)], ...
            [randomVertices(i,2),randomVertices(j,2)],'-o','Color','r','LineWidth',pheromoneMatri
        hold on
        end
    end
end
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
msg7 = sprintf('Pheromone matrix round %d',80);
title(msg7)

pheromoneMatrix = pheromoneMatrices{10};
figure(8)
for i = 1:N
    for j = 1:N
        if Mmatrix(i,j) == 1
        plot([randomVertices(i,1),randomVertices(j,1)], ...
            [randomVertices(i,2),randomVertices(j,2)],'-o','Color','r','LineWidth',pheromoneMatri
        hold on
        end
    end
end
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
msg8 = sprintf('Pheromone matrix round %d',10);
title(msg8)


pheromoneMatrix = pheromoneMatrices{5};
figure(9)
for i = 1:N
    for j = 1:N
        if Mmatrix(i,j) == 1
        plot([randomVertices(i,1),randomVertices(j,1)], ...
            [randomVertices(i,2),randomVertices(j,2)],'-o','Color','r','LineWidth',pheromoneMatri
        hold on
        end
    end
end
scatter(randomVertices(s0,1),randomVertices(s0,2),'g','square','filled')
```

```
scatter(randomVertices(t0,1),randomVertices(t0,2),'y','square','filled')
msg9 = sprintf('Pheromone matrix round %d',5);
title(msg9)

    function distanceMatrix = Distances(Mmatrix, vertices)

    N = length(vertices);
    distanceMatrix = zeros(N,N);

%     % Taking out the upper triagular part of Mmatrix
%     upperOfMmatrix = triu(Mmatrix);


%     % Find all indexes of ones
    for i = 1:N
        for j = 1:N
            if Mmatrix(i, j) == 1
                distance = pdist2(vertices(i,:), vertices(j,:));
                distanceMatrix(i, j) = distance;
            else
                distanceMatrix(i, j) = inf;
            end
        end
    end




%     % Generate random distances for all ones
%     randomDistances = randi(50, length(indexOfOnes), 1);
%     upperOfMmatrix(indexOfOnes) = randomDistances;


%
%     % Make the distance matrix by adding the inverse of upperOfMmatrix
%     distanceMatrix = upperOfMmatrix + upperOfMmatrix';
%
%     % Find all indexes of zeros in distance matrix
%     indexOfZeros = find(distanceMatrix == 0);
%
%     % Set all the zeros to inf distance
%     distanceMatrix(indexOfZeros) = inf;
end

    function simplifyPath = SimplifyPath(path, distanceMatrix)

    simplifyPath = [];
    i = 1;
    while i < length(path)+1
```

```
        if ismember(path(i), path(i+1:end))
            k = find(path(i+1:end) == path(i));
            i = k(1) + i;   % Find the next index of repeated vertices
        else
            simplifyPath(end+1) = path(i);
            i = i+1;
        end
    end

end

    function nextNode = BranchDecision(alpha, beta, i, weightMatrix, Mmatrix, pheromoneMatrix)

    possibleNextNode = find(Mmatrix(i,:) == 1);
    nextProbability = zeros(1,length(possibleNextNode));

    for j = 1:length(nextProbability)
        Nominator = pheromoneMatrix(i, possibleNextNode(j))^alpha * weightMatrix(i, possibleNextN
        nextProbability(j) = Nominator;
    end
    nextProbability = nextProbability./sum(nextProbability);
    nextNode = randomChoice(nextProbability, possibleNextNode);

end

    function pheromoneMatrix = PheromoneMatrix(Mmatrix)

    pheromoneMatrix = Mmatrix;

end

    function randomVertices = GenRandomVertices(N)

    x = randperm(N,N);
    y = randperm(N,N);
    randomVertices = cat(2,x',y');

end

    function newPheromoneMatrix = UpdatePheromones(pheromoneMatrix, j, path, pathDistance)

    pheromoneMatrix(path(j), path(j+1)) = pheromoneMatrix(path(j), path(j+1)) + 1/pathDistance;
    pheromoneMatrix(path(j+1), path(j)) = pheromoneMatrix(path(j+1), path(j)) + 1/pathDistance;
    newPheromoneMatrix = pheromoneMatrix;
end

    function pathLength = GetPathLengt(path, distanceMatrix)

    sum = 0;
```

```
    for i = 1:length(path)-1
        distance = distanceMatrix(path(i), path(i+1));
        sum = sum + distance;
    end
pathLength = sum;
end

    function node = randomChoice(Probabilities,nodes)
    x = cumsum([0 Probabilities(:).'/sum(Probabilities(:))]);
    x(end) = 1e3*eps + x(end);
    [a a] = histc(rand,x);
    node = nodes(a);

end

    function weightMatrix = WeightMatrix(distanceMatrix)

    % Find index of inf distances
    indexOfinf = find(distanceMatrix == inf);

    % Make all inf to one
    distanceMatrix(indexOfinf) = 1;

    % Make all the non inf distances to weights
    tmpMatrix = 1./distanceMatrix;

    % Find index of all ones in tmpWeigth matrix and set these weigths to
    % zero
    indexOfOnes = find(tmpMatrix == 1);
    tmpMatrix(indexOfOnes) = 0;
    weightMatrix = tmpMatrix;

end
```