Name: Ishaan Malhotra

Table of Contents

Roll no. : 1610110152

Instructer: Prof. Vijay Chakka

Lab 3

Aim: Plotting data in graphs and few properties

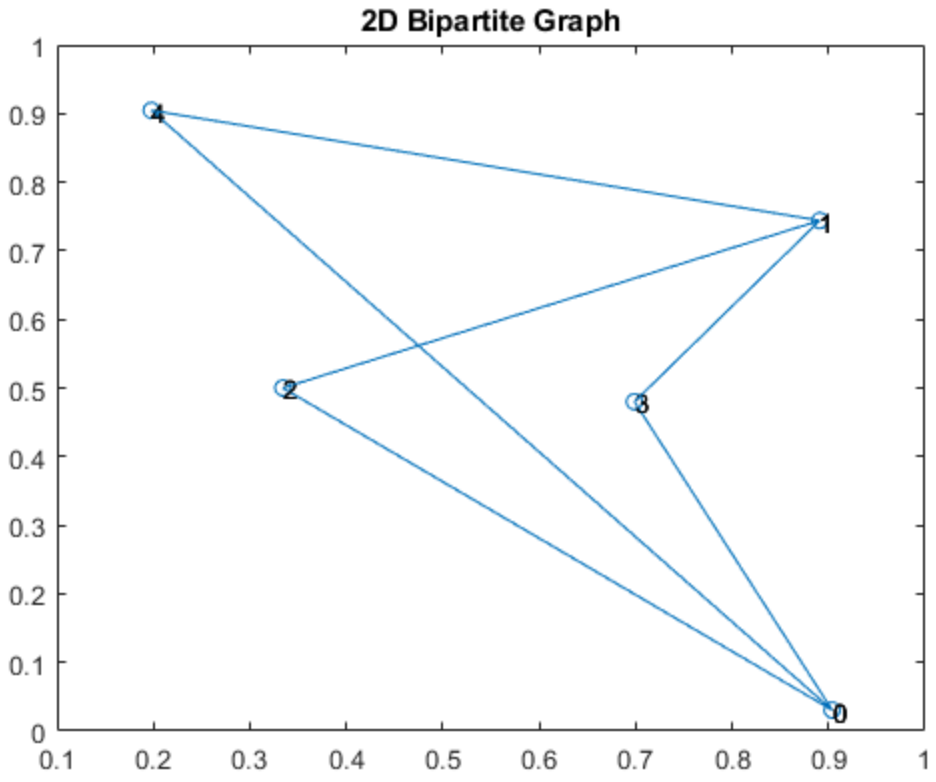
```
clc
clear all
close all
```

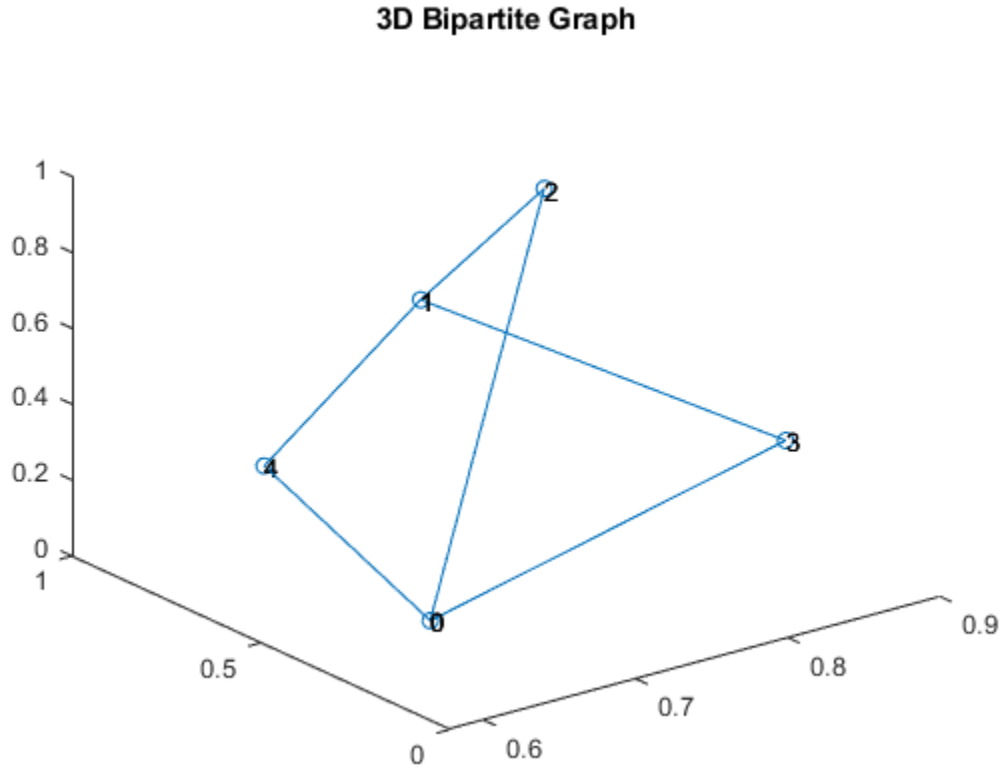
Question 1

```
bipartite = [[0 0 1 1 1];[0 0 1 1 1];[1 1 0 0 0 ];[1 1 0 0 0];[1 1 0 0
 0]];% Creating bipartite graph

ran2 = rand(length(bipartite),2); %computing 2D random points
ran2_3D = rand(length(bipartite),3); %computing 3D random points

plot2DGraph(bipartite,ran2); %plotting 2D bipartite graph
title('2D Bipartite Graph');
plot3Dv2(bipartite,ran2_3D); %plotting 2D bipartite graph
title('3D Bipartite Graph');
[rank,degree] = degreeCentrality(bipartite); %Computing degree
 centrality of each node and ranking them
```


2D Bipartite Graph
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
4
1
2
3
0

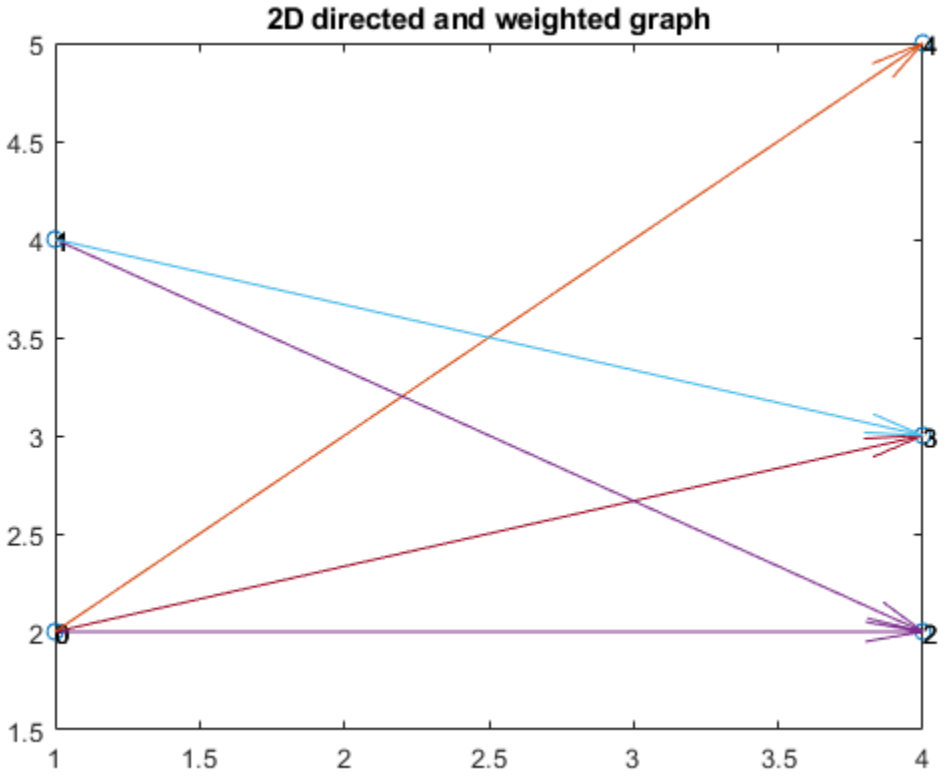

3D Bipartite Graph
1
0.8
0.6
0.4
0.2
0
1
0.5
0
0.6
0.7
0.8
0.9
2
1
3
4
0

Question 2

```
[SIE_rank, SIE_score]= SIE_ranking(bipartite); %Computing Structural
 Information entropy of each node and ranking them
```

Question 3

```
A = [[0 0 2 3 5]; [0 0 3 2 2]; [2 3 0 0 0]; [2 4 0 0 0]; [2 4 0 0
 0]]; % Creating directed and weighted bipartite graph
C = [[1 2]; [1 4]; [4 2]; [4 3]; [4 5]];
plot2DQuiverGraph(A,C) % Plotting graph
title('2D directed and weighted graph')
```



Question 4

```
[IFE_rank, IFE_score] = InteractionEntropy(A); %Computing Interaction
 Frequency entropy of each node and ranking them

Total = IFE_score + SIE_score; % adding both Structural and
 interaction entropies
nodes = (0:length(A)-1).'; % List of nodes
pair = [nodes Total.']; % Pairing nodes with their respective Total =
 IFE_score + SIE_score
node_rank = sortRank(pair); %Ranking them
```

```
function [node_rank_pair,deg] = degreeCentrality(A)
    deg = []; % returns degree of each node
    Rank = []; % returns rank of each node
    for i = 1:length(A)
        dc = 0; %for each node in A, initialixe its degree centrality
        for j = 1:length(A)
            if A(i,j) == 1
                dc = dc + 1; % if a node is connected to another node
            end% increment its degree centrality by 1
        end
        deg = [deg dc];
    end
    nodes = (0:length(A)-1).'; % List of nodes
    pair = [nodes deg.']; %pair of nodes with their respective degree
 centrality
    node_rank_pair = sortRank(pair); %ranking nodes acc to their dc
end



function [node_rank_pair, Ii_rank] = SIE_ranking(A)
    [rank, degree] = degreeCentrality(A); %finding degree of each node
    Ii_rank = [];
    pair = [];
    for i = 1:length(A)
        Ii = 0;
        sumDegNeigh = 0; %num of neighnours
        for j = 1:length(A)
            if A(i,j) == 1 %if neighbour exists, num of neighbours
 increment by degree of that neighbour
                sumDegNeigh = sumDegNeigh + degree(j);
            end
        end
        Ii = Ii + degree(i)/(sumDegNeigh + degree(i))*log(degree(i)/
(sumDegNeigh + degree(i))); %calculating SIE
        Ii_rank = [Ii_rank -Ii]; %Storing SIE of respective node
        pair = [pair; i-1 Ii_rank(i)];%pair of nodes with their
 respective SIE
    end
    node_rank_pair = sortRank(pair);%ranking nodes acc to their SIE
end



function [node_rank_pair, IFE_Rank] = InteractionEntropy(A)
    IFE_Rank = [];
    pair = [];
    for i = 1:length(A)
        sum = 0;
        sumNW = 0; %To store weight between ith and jth neighbours
        for j = 1:length(A)
```

```
            sumNW = sumNW + A(i,j); % sum of weights between ith and
 jth node in ith row
        end
        for k = 1:length(A)
            if A(i,k) ~= 0 %if kth node in ith row is having an edge
                Cij = A(i,k); %Take that kth node and store in Cij
                sum = sum + (Cij/sumNW)*log(Cij/sumNW); %Calculate
 Interaction entropy for ith node
            end
        end
        IFE_Rank = [IFE_Rank -sum]; %store in IFE of ith node in
 IFE_rank
        pair = [pair; i-1, IFE_Rank(i)]; % Create pair of ith node and
 its IFE
    end
    node_rank_pair = sortRank(pair); %sort the pair to rank
end




function node_rank = sortRank(pair)
    for i = 1:length(pair)-1
        for j = i+1:length(pair)%Use bubble sort to sort according to
 metric score
            if pair(i,2) < pair(j,2)% If ith metric is less than jth
 metric swap using temp
                                    % Sort in descending order
                temp1 = pair(j,1);
                pair(j,1) = pair(i,1);
                pair(i,1) = temp1;

                temp2 = pair(j,2);
                pair(j,2) = pair(i,2);
                pair(i,2) = temp2;
            end
        end
    end
    %rank = pair(:,2); %store sorted IFE values in to rank
    Rank = [1]; %First rank will be 1

    for k=2:length(pair)
        currentRank = Rank(k-1);%Current rank is extracted from Rank
        if pair(k-1,2) == pair(k,2)% if next node has same metric
 score
            Rank = [Rank; currentRank];% Give same rank to next node
        elseif pair(k-1,2) > pair(k,2) % if metric of current node is
 less than that of
            Rank = [Rank; currentRank+1];%prev node, increment the
 rank by 1 and store it
        end
    end
    node_rank = [pair(:,1) Rank]; % make sorted node - rank pair
end
```

```
function plot2DQuiverGraph(A,ran)

figure;
plot(ran(:,1),ran(:,2),'O');
edge = [];
for k = 1: length(A)
    row = A(k,:);
    for i = 1:length(row)
        if A(k,i) ~= 0 % if edge exists between kth and ith row
            x = [ran(k,1) ran(i,1)]; %Storing x and y coordinates
            y = [ran(k,2) ran(i,2)];
            k1 = int2str(k-1);
            edge = [edge; ran(k,:) ran(i,:)]; %Storing edge between k
 and i node
            text(ran(k,1),ran(k,2),k1);
            dp=[edge(k,3), edge(k,4)]-[edge(k,1), edge(k,2)];
            hold on;
            quiver(edge(k,1),edge(k,2),dp(1),dp(2),0);%Using quiver to
 draw directed lines
        end
    end
end
end
```

Published with MATLAB® R2018a