Complex Networks

A Networking and Signal Processing Perspective

The Solution Manual

for

End-of-Chapter Exercise Problems

Complex Networks

A Networking and Signal Processing Perspective

B. S. Manoj

Indian Institute of Space Science and Technology,
Thiruvananthapuram, Kerala, India

Abhishek Chakraborty

Indian Institute of Space Science and Technology,
Thiruvananthapuram, Kerala, India

Rahul Singh

Iowa State University, Ames, Iowa, USA

Contents

Guidelines for the Instructors

This solution manual provides answer keys to most of the end-of-chapter exercises in the book "Complex Networks: A Networking and Signal Processing Perspective". Instructors may request a copy of the solution manual from the publisher. Computer-based exercises, marked with the 💻 symbol, require a computer to execute programs to solve and obtain results by the readers. Challenge exercises are problems, marked with the symbol ✯, for which no solution exists in the current literature, and they are good research problems that can challenge an aspiring researcher. The solution manual provides answers or hints for answers for all exercise problems other than computer-based and challenge exercises.

In order to support the solution manual as well as the textbook, a set of code fractions for selected algorithms/programming exercises are available for download through the Authors' support website https://complexnetworksbook.github.io/. Additional programming samples, exercises, and support materials are available online through the same authors' support website. The support websites may also provide additional questions/exercises for instructors that could not be included in the main text. Similarly, code fractions will be continued to be added to the support website. Instructors may use the code fractions for introducing programming exercises, real-world course projects, as well as specific data analysis exercises that may be required for their courses.

We have made sufficient care to make the solution manual error free and accurate. We also have attempted to ensure the answers are sufficiently elaborate that instructors may find it satisfactory. We welcome feedback on the contents of this solution manual so that any undetected mistakes, that may have crept in without our knowledge, could be removed.

B. S. Manoj (bsmanoj@ieee.org)
Abhishek Chakraborty (abhishek2003slg@ieee.org)
Rahul Singh (rahul.s.in@ieee.org)

Chapter 2

Graph Theory Preliminaries

1. Find the adjacency matrix of the graph corresponding to the Königsberg bridge problem.

   Answer:
   Using the graph shown in Figure 2.1 in the textbook, the adjacency matrix for the Königsberg bridges is as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} \\ - & 0 & 1 & 1 \\ 0 & - & 1 & 1 \\ 1 & 1 & - & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

2. Find the weight matrix, adjacency matrix, and Laplacian matrix for the graphs shown in Figure 2.1.

(a) $\mathcal{G}_1$. **(b)** $\mathcal{G}_2$.

Figure 2.1: Example graphs (Question 2).

   Answer:
   Weight matrix of Figure 2.1 (a) is as follows:

$$\mathbf{W} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} \\ - & 0.7 & 0.5 & 0 \\ 0.7 & - & 0.9 & 0.2 \\ 0.5 & 0.9 & - & 1 \\ 0 & 0.2 & 1 & - \end{bmatrix}.$$

Adjacency matrix of Figure 2.1 (a) is as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} \\ - & 1 & 1 & 0 \\ 1 & - & 1 & 1 \\ 1 & 1 & - & 1 \\ 0 & 1 & 1 & - \end{bmatrix}.$$

Laplacian matrix of Figure 2.1 (a) is as follows:

$$\mathbf{L} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} \\ 1.2 & -0.7 & -0.5 & 0 \\ -0.7 & 1.8 & -0.9 & -0.2 \\ -0.5 & -0.9 & 2.4 & -1 \\ 0 & -0.2 & -1 & 1.2 \end{bmatrix}.$$

Weight matrix of Figure 2.1 (b) is as follows:

$$\mathbf{W} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} \\ - & 0 & 0.5 & 0 \\ 0.7 & - & 0.9 & 0.2 \\ 0.5 & 0.9 & - & 1 \\ 0 & 0 & 0 & - \end{bmatrix}.$$

Adjacency matrix of Figure 2.1 (b) is as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} \\ - & 0 & 1 & 0 \\ 1 & - & 1 & 1 \\ 1 & 1 & - & 1 \\ 0 & 0 & 0 & - \end{bmatrix}.$$

Laplacian matrix of Figure 2.1 (b) is as follows:

$$\mathbf{L} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{v_3} & \mathbf{v_4} \\ 0.5 & 0 & -0.5 & 0 \\ -0.7 & 1.8 & -0.9 & -0.2 \\ -0.5 & -0.9 & 2.4 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

3. Draw the graphs whose weight matrices are

$$\mathbf{W}_1 = \begin{bmatrix} 0 & 2 & 3 & 0 \\ 2 & 0 & 5 & 0 \\ 3 & 5 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{W}_2 = \begin{bmatrix} 0 & 3 & -1 & 0 \\ 3 & 1 & 0 & 2 \\ -1 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \end{bmatrix} \quad \mathbf{W}_3 = \begin{bmatrix} 0 & 0 & 4 & 0 \\ 3 & 0 & 0 & 2 \\ 1 & 8 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix}.$$

Also comment on graph types.

Answer:
W1 is weighted undirected graph.

W2 is weighted undirected graph with self-loop and negative edge weights.

W3 is Weighted directed graph.

4. Temperature sensors are deployed in a geographical area, as shown in Figure 2.2 of this solution manual, where the circles represent sensors. Each small square box is equivalent to 100 square miles. Construct a graph by connecting two sensors i and j with weights according to the formula

$$w_{ij} = \begin{cases} \exp\left(-\frac{[\text{dist}(i,\ j)]^2}{2}\right) & \text{if dist}(i,\ j) < 22 \text{ miles} \\ 0 & \text{otherwise,} \end{cases}$$

where dist$(i,\ j)$ is the physical distance between two sensors.

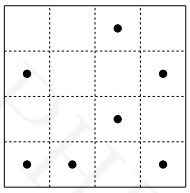


Figure 2.2: Sensor deployment in a geographical area (Question 4).

Answer:
Each sensor node of Figure 2.2 has given an identity as shown in Figure 2.3.

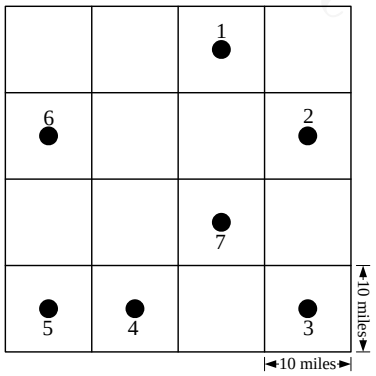


Figure 2.3: Assignment of sensor identity.

To construct a weighted graph where sensors are vertices and their connectivities are mimicked as edges, we need to measure the Euclidean distances among nodes. Note that as the weight of an edge is 0 if the distance between two sensor nodes are more than or equal to 22 miles, we

show distance estimations only between those sensor nodes where creating positive weighted links are possible.

Distance between sensor nodes 1 and 2: 14.14 miles.

Distance between sensor nodes 1 and 7: 20 miles.

Distance between sensor nodes 2 and 3: 20 miles.

Distance between sensor nodes 2 and 7: 14.14 miles.

Distance between sensor nodes 3 and 4: 20 miles.

Distance between sensor nodes 3 and 7: 14.14 miles.

Distance between sensor nodes 4 and 5: 10 miles.

Distance between sensor nodes 4 and 7: 14.14 miles.

Distance between sensor nodes 5 and 6: 20 miles.

In order to assign weight to each edge, the values are as follows:

Weight assigned for the edge length of 10 miles: $\exp(-50)$ (say x)

Weight assigned for the edge length of 14.14 miles: $\exp(-100)$ (say y)

Weight assigned for the edge length of 20 miles: $\exp(-200)$ (say z)

Therefore, the resultant graph is shown in Figure 2.4. Note that edge weights are mentioned as x, y, and z in the figure.

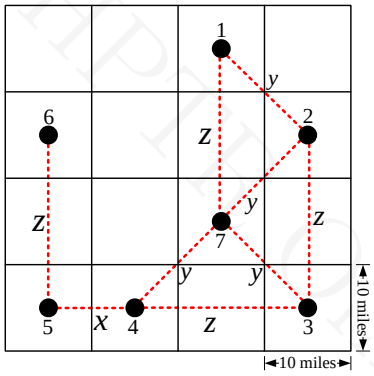


Figure 2.4: The constructed weighted graph.

5. Draw the graphs whose Laplacian matrices are (assuming that no self loops exist)

$$\mathbf{L}_1 = \begin{bmatrix} 5 & -1 & -4 & 0 \\ -1 & 8 & -5 & -2 \\ -4 & -5 & 8 & 1 \\ 0 & -2 & 1 & 1 \end{bmatrix} \qquad \mathbf{L}_2 = \begin{bmatrix} 5 & -2 & -3 & 0 \\ -1 & 8 & -3 & 0 \\ 1 & -5 & 8 & -4 \\ 0 & -2 & 1 & 1 \end{bmatrix}.$$

What if "no self loops" assumption is not taken into account? Whether the graphs can be uniquely determined in this case?

Answer:

$$\mathbf{W}_1 = \begin{bmatrix} 0 & 1 & 4 & 0 \\ 1 & 0 & 5 & 2 \\ 4 & 5 & 0 & -1 \\ 0 & 2 & 1 & 0 \end{bmatrix} \qquad \mathbf{W}_2 = \begin{bmatrix} 0 & 2 & 3 & 0 \\ 1 & 0 & 3 & 0 \\ -1 & 5 & 0 & 4 \\ 0 & 2 & -1 & 0 \end{bmatrix}.$$

If "self loops" assumption is taken into account, we need to have the edge weight of the self loop for obtaining a unique answer. Without the edge weight of the self loop, we will not be able to obtain a unique answer. The answer below assumes a unit weight for self loops for each node.

$$\mathbf{W}_1 = \begin{bmatrix} 0 & 1 & 4 & 0 \\ 1 & 0 & 5 & 2 \\ 4 & 5 & 0 & -1 \\ 0 & 2 & 1 & 0 \end{bmatrix} \qquad \mathbf{W}_2 = \begin{bmatrix} 0 & 2 & 3 & 0 \\ 1 & 0 & 3 & 0 \\ -1 & 5 & 0 & 4 \\ 0 & 2 & -1 & 0 \end{bmatrix}.$$

6. What can be said about the weight matrix and the Laplacian matrix of a disconnected graph with three components?

Answer:
Weight matrix of graph with three components will be block diagonal with three blocks. Laplacian matrix of the graph with three components will also be block diagonal with three blocks.

7. Consider the graph $\mathcal{G}$ shown in Figure 2.47 (reproduced in this document as Figure 2.5). Answer the following.
   (a) Is $\mathcal{G}$ planar? If yes, draw a planar embedding of $\mathcal{G}$.
   (b) Identify all the faces in the planar embedding of $\mathcal{G}$ and verify Euler's formula.
   (c) Draw the dual graph $\mathcal{G}^*$ of $\mathcal{G}$.
   (d) Again find the dual graph $(\mathcal{G}^*)^*$ of $\mathcal{G}^*$. Whether $(\mathcal{G}^*)^* \equiv \mathcal{G}$?

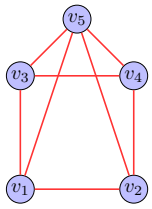


Figure 2.5: Graph $\mathcal{G}$ (Question 7).

Answer:
(a) Yes, $\mathcal{G}$ is planar. The planar embedding is shown in Figure 2.6.

(b) All faces of the planar embedding of $\mathcal{G}$ is shown in Figure 2.6 as $\{f_1,\ f_2,\ f_3,\ f_4,\ f_5\}$.

In this embedding, $N = 5$ (total number of nodes), $E = 8$ (total number of links), and $F = 5$ (total number of faces). Therefore, $N - E + F = 5 - 8 + 5 = 2$, which satisfies the Euler's formula.

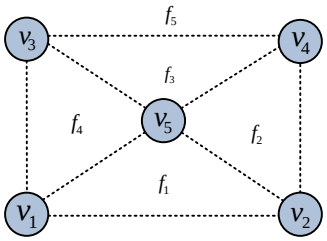


Figure 2.6: Planar embedding of $\mathcal{G}$.

(c) The dual graph $\mathcal{G}^*$ of $\mathcal{G}$ and its planar embedding are shown in Figure 2.7.

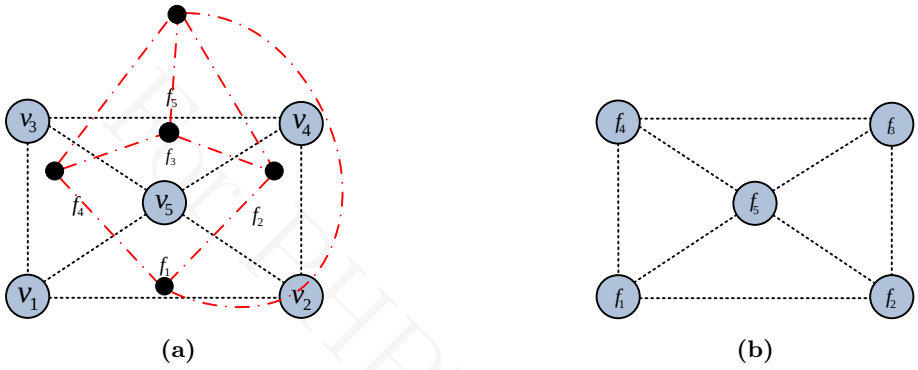


Figure 2.7: (a) $\mathcal{G}^*$ of $\mathcal{G}$ and (b) its planar embedding.

(d) Follow the similar steps, as shown in (c), to draw $(\mathcal{G}^*)^*$ of $\mathcal{G}^*$.

Yes, $(\mathcal{G}^*)^* \equiv \mathcal{G}$.

8. Draw 5-node connected planar graph with chromatic numbers of (a) 1, (b) 2, (c) 3, (d) 4, and (e) 5. .

Answer:

(a) A connected graph with more than one node cannot have chromatic number 1. Only a single node graph can have a chromatic number equal to 1.

(b) For chromatic number equal to 2, refer to Figures 2.8(a)-(b).

(c) For chromatic number equal to 3, refer to Figures 2.9(a)-(b).

(d) For chromatic number equal to 4, refer to Figures 2.10(a)-(b).

(e) For chromatic number equal to 5, refer to Figures 2.11(a)-(b).

9. Identify all the maximal and maximum cliques in graph $\mathcal{G}$ shown in Figure 2.48 in the text book (reproduced in this document as Figure 2.12). What is the clique number of the graph?

Answer:

Maximum clique size of the graph is 4. Therefore, clique number of the graph is 4. Max-

Figure 2.8: 5-node planar graph with Chromatic number of 2.

Figure 2.9: 5-node planar graph with Chromatic number of 3.

Figure 2.10: 5-node planar graph with Chromatic number of 4.

Figure 2.11: 5-node planar graph with Chromatic number of 5.

imum clique is (v_2, v_4, v_6, v_7). This graph has six maximal 3-cliques:(v_1, v_3, v_5), (v_1, v_2, v_5), (v_2, v_4, v_6), (v_2, v_6, v_7), (v_4, v_6, v_7), and (v_2, v_4, v_7).

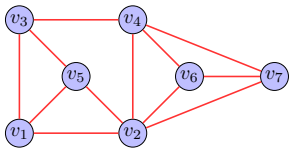


Figure 2.12: Graph $\mathcal{G}$ (Question 9).

10. What will happen if an edge from a tree is removed? What if an edge is added arbitrarily to a vertex pair of the tree?

    Answer:
    A tree follows three important properties: (i) existence of a unique path between every pair of vertices, (ii) minimal connectivity, and (iii) maximal acyclicity.

    When an existing edge is removed from the tree, the property (ii) is violated and the connectivity of the network is affected. That is, the tree will get partitioned into two separate trees.

    When a new edge is added to the tree, then the property (iii) is violated. That is, addition of an edge results in formation of a cycle in the tree.

11. Prove that a tree is a bipartite graph.

    Answer:
    One can prove this by mathematical induction. Before proceeding with the proof, we know that every tree T with N vertices contains atleast one vertex with degree 1. Let this 1-degree vertex be v. It is also true that $T - v$ can also be found to be a tree.

    Now proceed with mathematical induction for the cases N=1 and 2. For N=1 and 2, we have only two possibilities of trees, T_1 and T_2. Now T_1 has only one node and T_2 has only two nodes. Both these trees are bi-partite from the definition of bipartite trees.

    Now consider the tree with $N-1$ nodes that is bipartite where atleast one node exist with degree 1 as well as there exist a bipartite partition $A - B$.

    Now consider the tree with N nodes where vertex a is the vertex with degree 1, b is the only neighbor for vertex a, and $T - a$ is a tree with $N-1$ nodes where it has a bipartite partition $A - B$ exists where one set of vertices are in set A and another in set B. Now, if a is in A, then A, BUb is a bipartition of T. However, if a is in B, then B, AUb is a bipartition of T. Thus the proof.

12. Find the number of spanning trees in the graph shown in Figure 2.5 in this document.

    Answer:
    For a complete graph with N nodes, the number of spanning trees can be estimated as $N^{(N-2)}$. However, here we consider not a complete graph. The following algorithm can be used for estimating the number of possible spanning trees:

    Figure 2.13 illustrates Algorithm 2.1 for the graph in Figure 2.5 of this solution manual.

Algorithm 2.1 Algorithm for estimating the number of Spanning Trees in a graph

Require: Graph G
1: Let matrix A contains the adjacency matrix of the Graph G
2: Let N be the number of nodes in G.
3: Step-I: A_1 = Matrix obtained by replacing the diagonal elements in the matrix A with the degree of the corresponding node.
4: Step-II: A_2 = Matrix obtained by replacing all elements with 1 in A_1 with -1.
5: **return** The co-factor for any element of the matrix A_2.

	$\mathbf{v_1}$	$\mathbf{v_2}$	$\mathbf{v_3}$	$\mathbf{v_4}$	$\mathbf{v_5}$
$\mathbf{v_1}$	–	1	1	0	1
$\mathbf{v_2}$	1	–	0	1	1
$\mathbf{v_3}$	1	0	–	1	1
$\mathbf{v_4}$	0	1	1	–	1
$\mathbf{v_5}$	1	1	1	1	–

(a) Step-0: Adjacency matrix

	$\mathbf{v_1}$	$\mathbf{v_2}$	$\mathbf{v_3}$	$\mathbf{v_4}$	$\mathbf{v_5}$
$\mathbf{v_1}$	3	1	1	0	1
$\mathbf{v_2}$	1	3	0	1	1
$\mathbf{v_3}$	1	0	3	1	1
$\mathbf{v_4}$	0	1	1	3	1
$\mathbf{v_5}$	1	1	1	1	4

(b) Step 1

	$\mathbf{v_1}$	$\mathbf{v_2}$	$\mathbf{v_3}$	$\mathbf{v_4}$	$\mathbf{v_5}$
$\mathbf{v_1}$	3	−1	−1	0	−1
$\mathbf{v_2}$	−1	3	0	−1	−1
$\mathbf{v_3}$	−1	0	3	−1	−1
$\mathbf{v_4}$	0	−1	−1	3	−1
$\mathbf{v_5}$	−1	−1	−1	−1	4

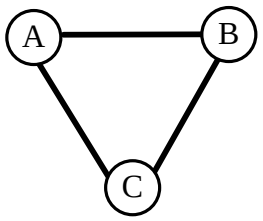
(c) Step 2

Figure 2.13: Illustration of the operation of Algorithm 2.1 (see Figure 2.5 of this solution manual).

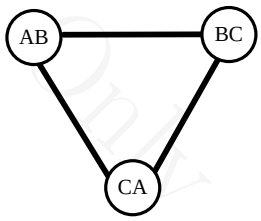
From the figure, it can be found that the co-factor of any element of the matrix is 45. Therefore, 45 spanning trees are possible with the graph shown in Figure 2.5 of this solution manual.

13. Draw a graph whose line graph is isomorphic to the graph.

Answer:
Refer to Figure 2.14 for details of the answer.



(a) A graph G.



(b) Line graph $\acute{G}$ of graph G.

Figure 2.14: A graph and its line graph which is isomorphic.

14. Draw two graphs with 5 nodes that cannot be line graphs.

Answer:
If a 4-node star graph is an induced subgraph of a graph G, then G cannot be a line graph. For example, Figure 2.42 of the textbook shows an example graph which is not the line graph of any graph. That is, the 4-node star graph of Figure 2.42 consisting of nodes V_2, V_3, V_4, and

V_5 is an induced subgraph. Another example is the graph that is obtained after removing the edge between vertices v_1 and v_5 of Figure 2.42.

15. Check whether the given graph shown in Figure 2.15 of this solution manual has cycles.

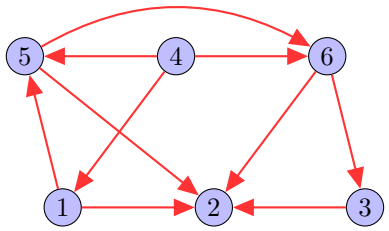


Figure 2.15: Graph $\mathcal{G}$ (Question 15).

Answer:
No. The graph does not have cycles.

C 16. Take the adjacency matrix of a directed graph $\mathbf{A}$ shown below. Write a MATLAB program to graphically visualize the graph. On the graph you visualized, (i) check if you can visually find any cycles; (ii) implement Acyclicity test algorithm using MATLAB program and conduct an Acyclicity test; (iii) create your own larger adjacency matrices and conduct an Acyclicity test.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Answer:
The following command(s) in MATLAB version 2016a can be used to visualize the graph:

view(biograph(DG))

To check if a graph is Directed Acyclic Graph the following command can be used:

graphisdag(DG),

where DG is and N-by-N sparse matrix that represents a directed graph. Nonzero entries in matrix DG indicate the presence of an edge.

17. Prove that the existence of a back-edge, in the DFS output of a directed graph, is sufficient for detecting the presence of cycles.

Answer:
Assume a cyclic Graph G on which you execute DFS algorithm as shown in Figure 2.16 (a).

Assume p is the vertex on the cycle with lowest DFS number. Since p is on a cycle, there must exist another vertex q with a directed edge $q \to p$. Further, q is a descendant of p due

to p's low DFS number (as can be seen from Figure 2.16(b). The edge $q \rightarrow p$ cannot be a tree arc because DFS number of p is lower than DFS number of q. Further the edge $q \rightarrow p$ cannot be a forward arc as the edge $q \rightarrow p$ is not a tree arc. Since vertices q and p are on the same tree, it cannot be a cross arc either. The only possibility is that $q \rightarrow p$ is a back arc. Therefore, presence of a back arc indicates presence of a cycle in the graph.

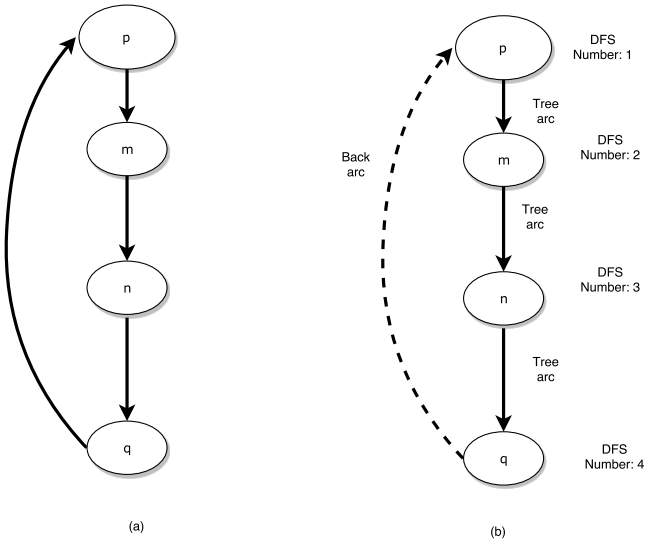


Figure 2.16: (a) Example Cycle Graph $\mathcal{G}$ for (Question 17).

18. Does Dijkstra's shortest path algorithm work effectively with negative values as edge weights? Justify your answer.

Answer:
No. Dijkstra's shortest path algorithm does not work when the edge weights are negative values. The reason for this is that the function used for finding minimal path cost (Min function) does not work correctly due to the addition of edges with negative edge weights. A simple example of a graph with negative edge weights can prove this fact.

19. Provide a proof of correctness for Dijkstra's shortest path algorithm.

Answer:
Dijkstra's shortest path algorithm is provided in Algorithm 2.4 and is reproduced in this document as Algorithm 2.2.

//For the algorithm, the proof can be attempted using mathematical induction.

//Our aim is to show that $Dist[x]$ is shortest distance for every vertex x //from source node i when the algorithm terminates. It can be proved by //mathematical induction the set $|T|$ using the following lemma: //

Lemma 2.1. *//For each $x \in T$, $Dist(x)$ is shortest.*

Algorithm 2.2 Dijkstra's Shortest Path Algorithm

Require:

$\mathcal{G} = (\mathcal{V},\ \mathcal{E})$ — A network graph with $\mathcal{V}$ nodes and $\mathcal{E}$ edges where each edge $(x,\ y)$ is weighed by the cost of the edge

Dist[] is an array at node i storing the distance to various nodes

1: **for** each node $j \in \mathcal{V}$ and $j \neq i$ **do**
2: Dist[j] = W[$i,\ j$]; *// Initialize the Dist array with edge weights of each neighbor node. Non-neighbor nodes will have Distance value ∞.*
3: **end for**
4: P = {i}
5: $T\ =\ \mathcal{V}\ -\ P$ *// P is a set of nodes selected at each iteration of the algorithm and is initialized with node i. T is a set of nodes yet to be selected for inclusion in P. It is initialized with all nodes except node i.*
6: **for** All nodes $j \in \mathcal{V}$ and $j \neq i$ **do**
7: Choose node $x \in T$ such that $Dist[x]$ minimum
8: $P \leftarrow P|x$ *// Newly selected node x is included in Set P.*
9: $T \leftarrow T - x$
10: **for** each node $r \in T$ **do**
11: **if** $Dist[x]\ +\ W[x, r] < Dist[r]$ **then**
12: $Dist[r]\ =\ W[x,\ r]\ +\ Dist[x]$ *// Cost is updated for all nodes in the Set T.*
13: $P[r]\ =\ x$ *// Updating the predecessor information.*
14: **end if**
15: **end for**
16: **end for**

//Initial case ($|T| = 1$): The unique situation where $|T| = 1$ is when $x = i$ //and $Dist[x] = Dist[i] = 0$ which is correct.

//Inductive hypothesis: Let k be the last vertex added to T. //Let $R0 = R \bigcup u$. Our I.H. is: for each $x \in R0$, //$Dist(x)$ is shortest.

//Let us assume a contradiction that $Dist(x)$ is not the shortest where the //shortest distance is $Dist`(x)$ where $Dist(x) > Dist`(x)$. //

Lemma 2.2. *Shortest paths are composed of shortest sub-paths.*

The proof of this lemma is based on the idea that if there was a shorter sub-path for any segment of the path exists, then the shorter sub-path would have replaced the the path segment in order to make the entire path shorter. For example, consider a path between source node S and destination node D: $S- > a- > b- > c- > d- > e- > f- > g- > D$. Take one segment of the path: $c- > d- > e- > f$. If the path between $S - D$ is shortest, then the sub-path for the segment $c- > d- > e- > f$ shall also be shortest. Otherwise, including a shorter-subpath between c and f would have made the $S - D$ path shorter.

Proof follows from the above two Lemmas.

20. Compare the time complexities of Dijkstra's shortest path algorithm modified for All Pair Shortest Path finding and Floyd's all pair shortest path algorithm.

Answer:
Asymptotically, they both have the same complexity of $O(N^3)$ for dense graphs. In practice, Floyd's algorithm runs faster in finite sized graphs as its implementation can be done more efficient in dense graphs. However, for sparse graphs, repeated Dijkstra's algorithm can be implemented with fibonacci heaps resulting in a time complexity of $O(N^2 \log N)$ which is better than Floyd's algorithm.

C 21. Write a simple computer program to list the shortest paths between all node pairs.

Answer:
The following recursive pseudo code/algorithm can be translated to a computer program to achieve the path listing.

Algorithm 2.3 APSP_PathListing(x, y) Algorithm

Require: Path
1: Let PR[] metric contains the predecessor information generated by running Floyd's All Pair Shortest Path algorithm. p=x, q=y; x and y are source and destination nodes.
2: TempNode = PR[p, q]
3: **if** TempNode = 0 **then** Return
4: **end if**
5: APSP_PathListing(p, TempNode)
6: printf(TempNode);

Chapter 3

Introduction to Complex Networks

1. For the given network in Figure 3.1 of this solution manual, estimate the following: (i) degree centrality (DC), (ii) closeness centrality (CC), and (iii) betweenness centrality (BC).

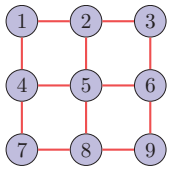


Figure 3.1: An unweighted grid network topology for Problem 1

Answer:
Refer to Table 3.1 for DC, CC, and BC values of all nodes in the network of Figure 3.1.

Table 3.1
DC, CC, and BC values of all nodes in Figure 3.1

Node	DC	CC	BC
1	2	0.0556	1.3333
2	3	0.0667	5.0000
3	2	0.0556	1.3333
4	3	0.0667	5.0000
5	4	0.0833	10.6667
6	3	0.0667	5.0000
7	2	0.0556	1.3333
8	3	0.0667	5.0000
9	2	0.0556	1.3333

2. Estimate (i) graph degree centrality (G_{DC}), (ii) graph closeness centrality (G_{CC}), and (iii) graph betweenness centrality (G_{BC}) for the network in Figure 3.2.

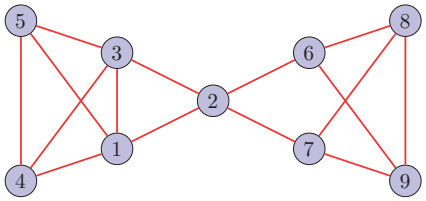


Figure 3.2: Network for Problem 2

Answer:
The values are as follows: $G_{DC} = 0.1071$, $G_{CC} = 0.0482$, and $G_{BC} = 0.2455$. Please refer to Section 3.2.6 for more details on graph centrality metrics.

C 3. Create a large ER network with $\mathcal{G}(10000,\ 0.1)$ and estimate the following: (i) graph degree centrality (G_{DC}), (ii) graph closeness centrality (G_{CC}), and (iii) graph betweenness centrality (G_{BC}).

Answer:
This problem is a computer-based exercise. Use any of the following network simulators: NS-2, NS-3, QualNet, OPNET, or write a Matlab program on your own to simulate the network. Some Matlab code fractions available for simulations can be found at https://complexnetworksbook.github.io.

4. Derive the closed-form expressions for (i) graph degree centrality (G_{DC}), (ii) graph closeness centrality (G_{CC}), and (iii) graph betweenness centrality (G_{BC}).

Answer:
In Equation 3.2.7 of the textbook, the maximum sum difference at the denominator (i.e., $Max \sum_{i=1}^{N}[GC_{metric}(x^{\star}) - GC_{metric}(x_i)]$) can be estimated as $(N-2)$ in an N node network if the maximum value of $GC_{metric}(x^{\star})$ is $(N-1)$. Therefore, the difference sum is $(N-1)(N-2)$ as there are $(N-1)$ components.

Therefore,

$$GC_{DC} = \frac{\sum_{i=1}^{N}[GC_{DC}(x^{\star}) - GC_{DC}(x_i)]}{(N-1)(N-2)} = \frac{\sum_{i=1}^{N}[GC_{DC}(x^{\star}) - GC_{DC}(x_i)]}{N^2 - 3N + 2}.$$

Conversely, the maximum possible closeness for a node only takes place when its closeness sum is 1. Rest of the nodes are at the unit distance from the center node and at most two from the other nodes in a network. Therefore, the closeness sum for each node is

$$\frac{N-1}{1+2(N-2)} = \frac{N-1}{2N-3}.$$

Hence, the difference becomes

$$1 - \frac{N-1}{2N-3} = \frac{N-2}{2N-3}.$$

Now, there are $(N-1)$ such nodes in the network, and the maximum possible difference becomes

$$\frac{N-1}{2N-3}(N-1) = \frac{N^2-3N+2}{2N-3}.$$

The graph closeness centrality can be estimated as

$$GC_{CC} = \frac{\sum_{i=1}^{N}[GC_{CC}(x^\star) - GC_{CC}(x_i)]}{(N^2-3N+2)/(2N-3)}.$$

Now, to estimate GC_{BC}, we take the normalized point betweenness centrality values as mentioned in Equation 3.2.6 (ignoring the term 2) in the textbook and for $(N-1)$ nodes the graph betweenness centrality becomes

$$GC_{BC} = \frac{\sum_{i=1}^{N}[GC_{BC}(x^\star) - GC_{BC}(x_i)]}{(N-1)(N-2)\times(N-1)} = \frac{\sum_{i=1}^{N}[GC_{BC}(x^\star) - GC_{BC}(x_i)]}{N^3-4N^2+5N-2}.$$

Please refer to [1] for more details.

5. Estimate (i) eigenvector centrality (EC) and (ii) graph Fourier transform centrality (GFT-C) for the network shown in Figure 3.1. Make your own observation on the most central node in the network. Consider necessary assumptions.

Answer:
Eigenvector centralities (EC) of the nodes 1 through 9 are
$0.0858, 0.1213, 0.0858, 0.1213, 0.1716, 0.1213, 0.0858, 0.1213, 0.0858$.

Graph Fourier transform centralities (GFT-C) of the nodes 1 through 9 are
$0.0815, 0.1396, 0.0815, 0.1396, 0.1159, 0.1396, 0.0815, 0.1396, 0.0815$.

Refer to Section 9.7.2 for more details on GFT-C.

C 6. For the network shown in Figures 3.3(a) and 3.3(b), estimate the degree correlations. Make your observation on the degree correlations for the two networks.

Answer:
This problem is a computer-based exercise. Use any of the following network simulators: NS-2, NS-3, QualNet, OPNET, or write a Matlab program on your own to simulate the network. Some Matlab code fractions available for simulations can be found at https://complexnetworksbook.github.io.

7. Estimate the network resistance distance for the following: (a) nodes 12 and 20 as well as nodes 11 and 9 in Figure 3.3(a) of this solution manual and (b) nodes 1 (club instructor) and 34 (club administrator) in Zachary's karate club network depicted in Figure 1.1 of the textbook.

Answer:
(a) The network resistance distance between nodes 12 and 20 is 6, and nodes 11 and 9 is 1.5 when Figure 3.3(a) of this solution manual is concerned.