Network Analysis

1

Flow of Lecture by Headings

- Network Analysis: What is a Network?
- Mathematical Abstraction and Definition of Networks
- Network Types and Structure
  - Undirected, Directed, Weighted, and Special Edges
- Network Representation (Adjacency Matrix and List)
- Link-centric and Temporal Network Types
  - Unipartite, Bipartite, n-Partite, Time-Varying Networks
- Node Degree and Its Variants
  - Degree, Weighted Degree, Indegree, Outdegree
- Degree Distribution and Charts
  - Cumulative and **Complementary Cumulative Distributions**
- Power Law and Scale-Free Networks

2

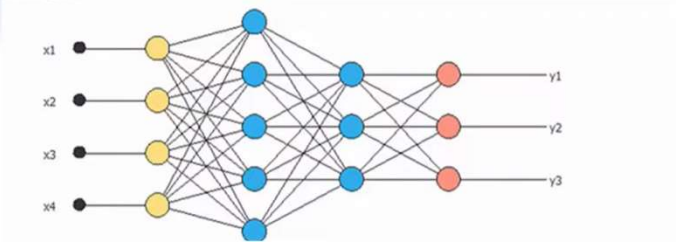
Flow of Lecture by Headings

- Graph Preliminaries
  - Adjacency, Incidence, Walks, Paths, Cycles
- Network Metrics
  - Distance, Diameter, Average Path Length, Density, Connected Components
- Centrality Measures
  - Degree Centrality
  - Closeness Centrality
  - Betweenness Centrality
  - Eigenvector Centrality
- MST and PMFG
- Network Modularity: Definition and Formula
- Community Detection and Modularity Maximization
  - Fast Greedy Algorithm
  - Louvain Method

3

What is a Network?

It is not (only) a computer network.



4

How many of you feel that you are part of a network?

- In fact, you are part of multiple networks!
- Do you feel any inertia/pull/attraction, being part of the networks?

5

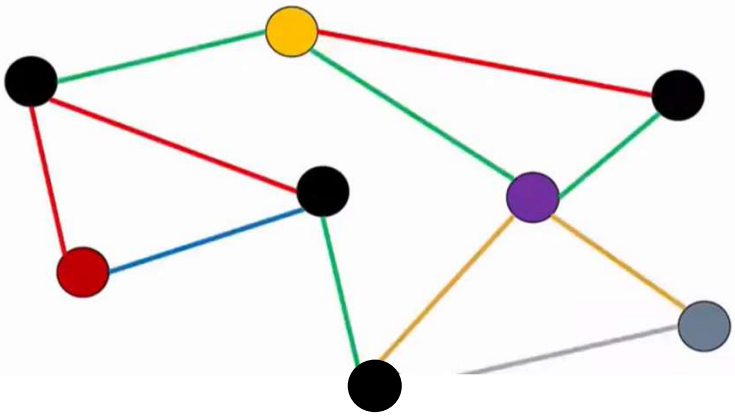
What is a Network?

A network is a simple (mathematical) abstraction of a complex system.

Networks are a general language for describing complex systems.

6

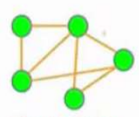
Network = Individuals + connections



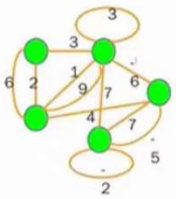
7

Network: Definition

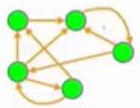
- Network (aka, *graph*): An ordered pair G(V, E), where V is a set of nodes/vertices, and E is a set of edges/links.
- Depending on the nature of application, the above definition may be revised or augmented, as follows:
  - ✓ Undirected (also called asymmetric, or irreversible), directed (also called symmetric, or reversible) edges, or hyperedges, etc.
  - ✓ Nodes and links with attributes/features like weights, timestamps, textual features, etc.
- Self-loops: Edges with same node as end nodes.
- parallel edges: More than one edge joining a pair of nodes.



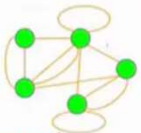
Simple graph



Weighted graph



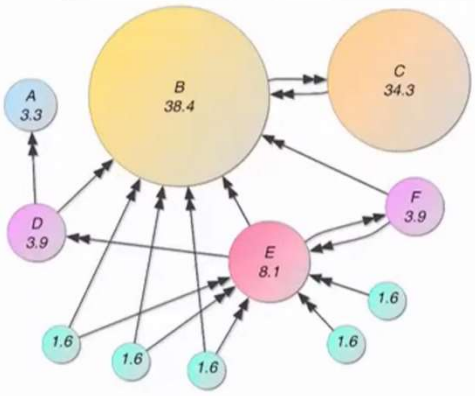
Directed graph



A graph with loops and parallel edges

8

Web Search Optimization

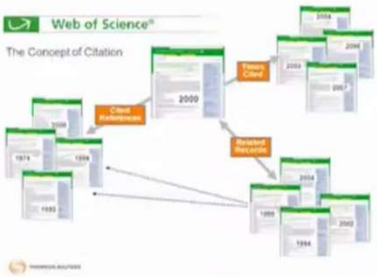


Toy Example of Google's PageRank

- Ranking the webpages based on hyperlinks
- Ordering of displayed pages based on ranks
- Displaying webpages based on user search profile

9

SNA Applications: Scientific Research & Academic Collaboration

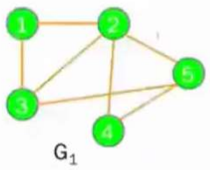


- Scientific authors cite (refer to) the works of other authors
- Various networks of scholarly articles may be formed exploiting this relationship
  - Paper-paper citation network
  - Paper-paper co-citation network, etc.
- Scientific authors collaborate with one another
- Various scholar networks may be formed using these relationships
  - Author collaboration network
  - Author citation network
  - Author co-citation network, etc.

10

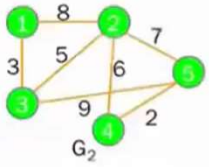
Network Representation: Adjacency Matrix/List

An adjacency matrix $A = (a_{ij})$ for a graph $G(V, E)$ is a square matrix of dimension $|V| \times |V|$ such that each element a_{ij} of A indicates the existence of an edge between the node v_i and node v_j (also the weight of the corresponding edge in case of a weighted graph) in G.



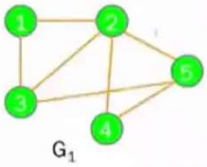
G_1

$$A(G_1) = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

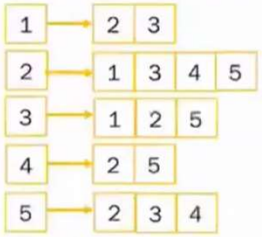


G_2

$$A(G_2) = \begin{bmatrix} 0 & 8 & 3 & 0 & 0 \\ 8 & 0 & 5 & 6 & 7 \\ 3 & 5 & 0 & 0 & 9 \\ 0 & 6 & 0 & 0 & 2 \\ 0 & 7 & 9 & 2 & 0 \end{bmatrix}$$

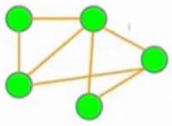


G_1



11

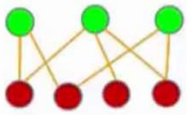
Network Types: **Link-centric View** Unipartite Network



- Consists of a vertex set *V* and an edge set *E*. There is no restriction on the formation of edges between nodes of the network
- Example: An organizational LAN, where nodes are the devices, and edges are the local area links.
- Used to model the situation when links can join any pair of nodes of the network

12

Network Types: **Link-centric View** Bipartite Network

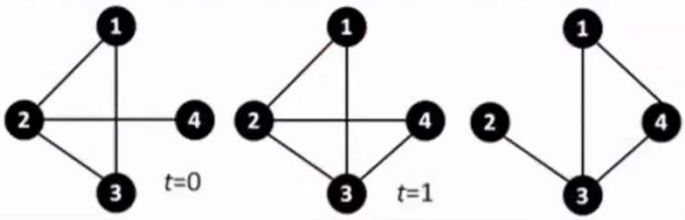


- Consists of a vertex set *V* that is divided into two sets *V1* and *V2* that are disjoint and independent. Each edge of the network connects a vertex in *V1* to another vertex in *V2*
- Example: An e-commerce user-product network. One part consists of the users, the other part consists of the products, the links are based on the basis of who bought what.
- Generalization of Bipartite network is n-partite networks, where the vertex set is partitioned into n number of part, an edges join a node from one part with a node from another part.

13

Network Types: **Temporal View:** Time-varying Network

- Consists of a set of nodes *V* and a set of edges *E* where each edge $e_{i,j} \in E$ is represented by a three- tuple $e_{i,j} = \{v_i, v_j, t_{ij}\}$. Here, v_i and v_j are two end-points, and t_{ij} indicates the persistence duration of the edge e_{ij}
- Example: Person-to-person communication network over a span of time. The visible components are snapshots of the network at different time instances.



14

Observed common properties

Small world property

Scale-free structure

Clustering and community structure

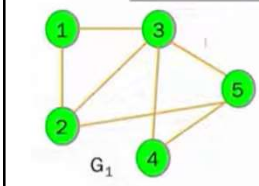
Robustness to random node failure

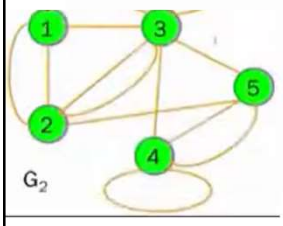
Cascading effect

Vulnerability to cascading failures

15

Degree of a Node

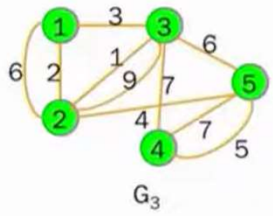




- For an undirected, unweighted network, degree of a node *v* is defined as the number of nodes in the network to which there is an edge from the node *v*.
- In other words, for an undirected, unweighted network, degree of a node *v* is the number of edges of the network that are incident on the node *v*.
- Putting differently, for an undirected, unweighted network, degree of a node *v* is the number of neighbours of the node *v*.
- In graph G_1, degrees of the nodes 1 through 5 are 2, 3, 4, 2, 3, respectively.
- In graph G_2, degrees of the nodes 1 through 5 are 3, 5, 7, 5, 4, respectively.
- Note: A self-loop is counted twice in evaluating degree of a node.

16

Weighted Degree of a Node

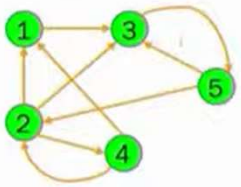


G_3

- For an undirected, weighted network, the weighted degree of a node is defined as the sum of weights of the edges incidents on that node
- For the weighted undirected graph G_3, the weighted degrees of the nodes are as follows:
  - Weighted degree of node 1 is 11
  - Weighted degree of node 2 is 22
  - Weighted degree of node 3 is 26
  - Weighted degree of node 4 is 16
  - Weighted degree of node 5 is 22

17

Indegree and Outdegree of a Node



G_4

- In a directed network, the indegree of a node is defined as the number of incoming edges to the node
- In a directed network, the outdegree of a node is defined as the number of outgoing edges from the node
- For the directed graph G_4, the indegrees and outdegrees of the nodes are as follows:
  - Indegrees of the nodes 1 through 5 are 2, 2, 3, 1, 1
  - Outdegrees of the nodes 1 through 5 are 1, 3, 1, 2, 2

18

Sum of the Degrees…

- For an unweighted, undirected network, **the sum of the degrees of the nodes in a graph is twice the number of edges in the graph**

19

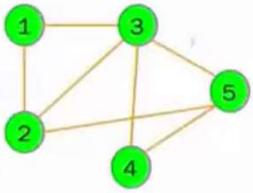
Degree Distribution

- Degree distribution of a network is the (probability) distribution of the degrees of nodes over the whole network.
- Let a network has $N = |V|$ nodes.
- Let P_k denotes the probability that a randomly chosen node has degree k.
- Then, $P_k = \frac{N_k}{N}$, where N_k refers to the number of nodes of degree k in the network.
- The distribution (k, P_k) represents the degree distribution of the concerned graph,
- The mean degree, denoted $\langle k \rangle$, is given by $\langle k \rangle = \sum_k k.P_k$.

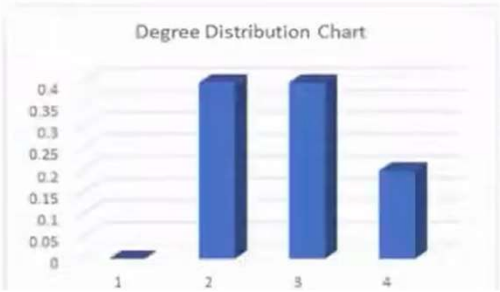
20

Degree Distribution: Example

- For the graph G_1, we have the following:
  $N = 5$, and $N_1 = 0, N_2 = 2, N_3 = 2, N_4 = 1$.
- The above implies, $P_1 = 0, P_2 = 0.4, P_3 = 0.4, P_4 = 0.2,$



G_1



21

Cumulative Degree Distribution

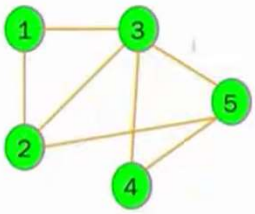
- Cumulative degree distribution (CDD) is given by the fraction of nodes with degree smaller than k.
- In other words, it is the distribution(k, C_k), where $C_k = \frac{\sum_{k' \leq k} N_{k'}}{N}$
- Complementary cumulative degree distribution (CCDD) is given by the fraction of nodes with degree greater than or equal to k.
- In other words, it is the distribution(k, CC_k), where $CC_k = 1 - C_k$

22

Degree Distribution: Example

- For the graph G_1, we have the following:

$N = 5$, and $N_1 = 0, N_2 = 2, N_3 = 2, N_4 = 1.$

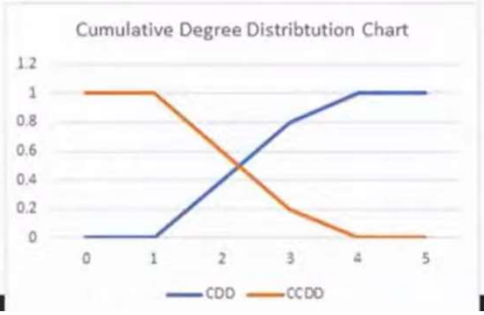


G_1

- The above implies, $P_1 = 0, P_2 = 0.4, P_3 = 0.4, P_4 = 0.2,$

$C_1 = 0, C_2 = 0.4, C_3 = 0.8, C_4 = 1.0,$

and $CC_1 = 1.0, CC_2 = 0.6, CC_3 = 0.2\ CC_4 = 0.0.$



23

Power Law

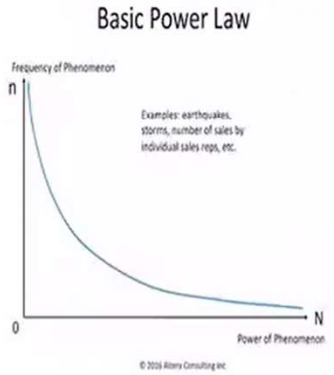
A **power law** is a functional relationship between two quantities: one quantity varies as a power of another.

- **Scale invariant:**
  - One attribute of power laws is their scale invariance.
  - Given a relation $f(x) = ax^{-k}$, scaling the argument x by a constant factor c causes only a proportionate scaling of the function itself.

$$f(cx) = a(cx)^{-k} = c^{-k} f(x) \propto f(x)$$

24

Scale-free Property: Power Law



https://exploitingchange.com/2016/09/14/another-powerful-idea-the-power-law/

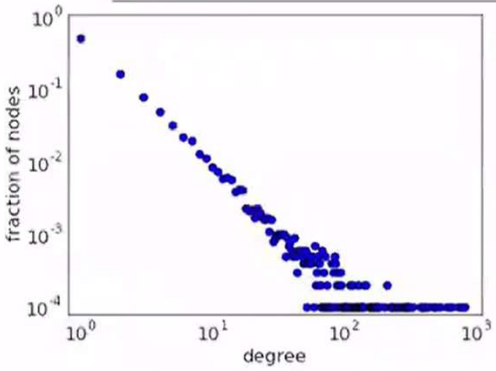
- Power Law: a relative change in the value of one variable leads to the proportional change in the value of other variable
- Independent of the initial values of both the variable
- Mathematically,

$$y \propto x^{-b}, b \in \mathbb{R}$$

- Functions that follows power-law are scale-invariant
- Pareto Principle (or the 80/20 rule) in Economics: 80% of the outcomes are results of 20% of the causes
- Power law principle is also coined as Pareto Distribution

25

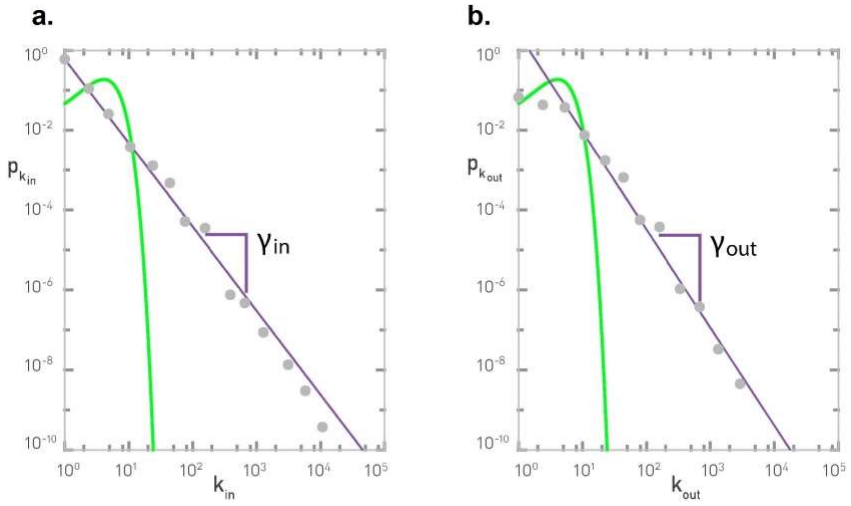
Scale-free Property: Real Networks



Degree distribution of a scale-free network
https://mathinsight.org/scale_free_network

- Networks whose degree distribution follows power-law are known as scale-free networks
- Most real-world networks are found to be scale-free

26

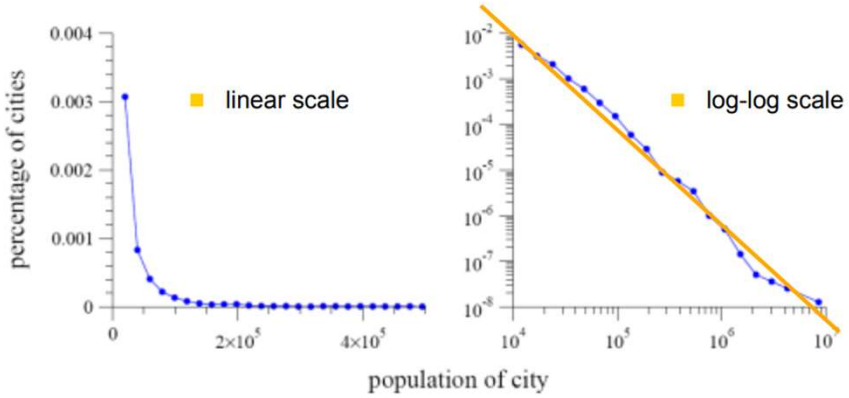


The Degree Distribution of the WWW

The incoming (a) and outgoing (b) degree distribution of the WWW sample mapped in the 1999 study of Albert *et al*. [1]. The degree distribution is shown on double logarithmic axis (log-log plot), in which a power law follows a straight line. The symbols correspond to the empirical data and the line corresponds to the power-law fit, with degree exponents γ_{in}= 2.1 and γ_{out} = 2.45. We also show as a green line the degree distribution predicted by a Poisson function with the average degree $\langle k_{in} \rangle = \langle k_{out} \rangle$ = 4.60 of the WWW sample.

27

Power-law distribution

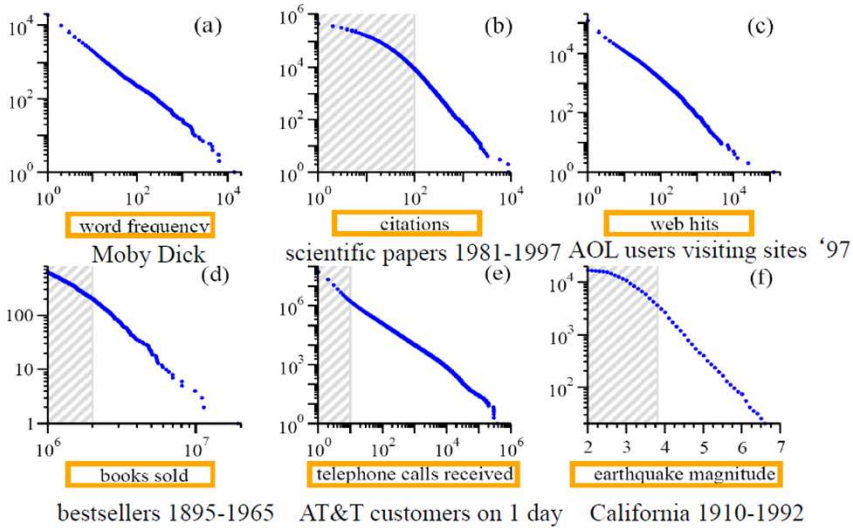


- high skew (asymmetry)
- straight line on a log-log plot

28

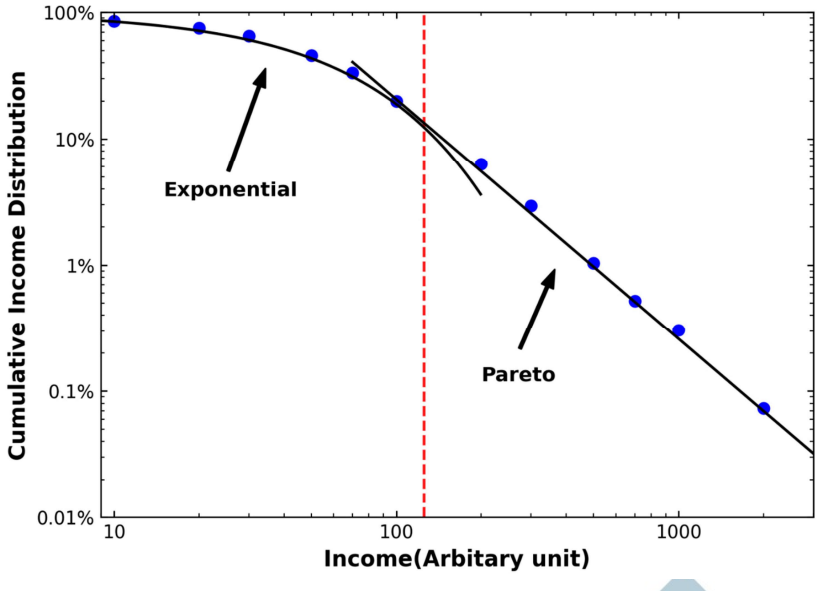
Power laws are seemingly everywhere

note: these are cumulative distributions, more about this in a bit...



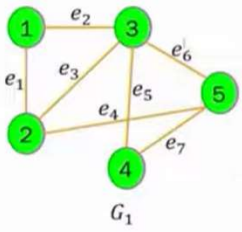
Source:MEJ Newman, ' Power laws, Pareto distributions and Zipf' s law' , *Contemporary Physics* **46**, 323–351 (2005)

29



30

Some Graph Preliminaries…

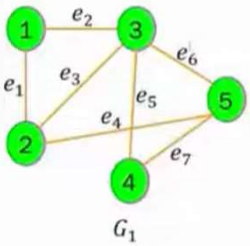


G_1

- In an undirected network,
  - Two nodes are called adjacent if they are linked by an edge.
  - Two edges are called incident if they share a common end-node.
- In graph G_1, the nodes **1 and 2** are adjacent, **1 and 3** are adjacent, and so on.
- In graph G_1, the edges $\boldsymbol{e_1}$ **and** $\boldsymbol{e_2}$ are incident, $\boldsymbol{e_1}$ **and** $\boldsymbol{e_3}$ are incident, and so on
- A walk in a network is an alternating sequence of nodes and edges, where every consecutive node pair is adjacent, and every consecutive edge pair is incident.
- A walk may pass through a node or an edge more than once. Length of a walk is the number of edges in the sequence.
- In graph G_1, the sequence **{3, e_3, 2, e_4, 5, e_6, 3, e_5, 4, e_7, 5, e_4, 2}** is a walk of length 6.
- For a simple graph, the edges from the above sequence may be omitted.

31

Some Graph Preliminaries…

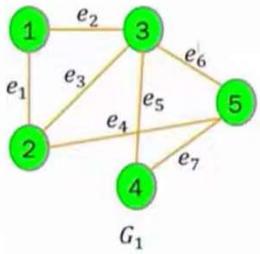


G_1

- A walk in a network is called
  - a closed walk if the last node in the sequence is same as the first node; else it is called an open walk.
  - a trail if the sequence has no repeated edge.
  - a path if the sequence has neither a repeated edge nor a repeated node. In other words, a path is an open trail having no repeated nodes.
  - a cycle if the sequence has all the edges distinct, and all the nodes, except the first and the last nodes, are also distinct. In other words, a cycle is a closed path with the only repetition of the first and the last nodes in the sequence.

32

Some Graph Preliminaries…



G_1

- The distance between nodes v_i and v_j in a graph is defined as the length of the **shortest path** between the nodes v_i and v_j.
- In graph G_1, the distance between 1 and 4 is 2, the same between 1 and 5 is also 2.
- The diameter of a network is defined as the maximum distance between any pair of nodes in the network.
- The diameter of the graph G_1 is 2.
- For a graph G with n nodes, the average path length l_G is defined as the average number of steps along the shortest paths for all possible pairs of nodes in the network.

$$l_G = \frac{\sum_{i \neq j} d_{ij}}{n(n-1)}$$, where d_{ij} is distance between nodes v_i and v_j

33

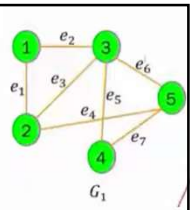
Some Graph Preliminaries…

- The density of a graph $G(V, E)$, denoted $\rho(G)$, is defined as the ratio of the number of edges in the graph to the total number of possible edges in the network. Mathematically,

$$\rho(G) = \frac{2 \times |E|}{|V| \times (|V| - 1)}$$

34

Local Clustering Coefficient

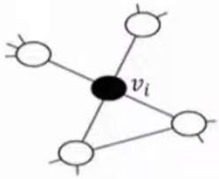


G_1

The local clustering coefficient C_i for a vertex v_i in a network $G(V, E)$ is given by the proportion of edges between the vertices within its neighborhood divided by the number of links that could possibly exist between them.

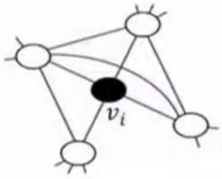
$$C_i = \frac{2 \times |\{e_{jk} | \, v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i . (k_i - 1)}$$

Where N_i is the neighbourhood of the vertex v_i, and $k_i = |N_i|$.



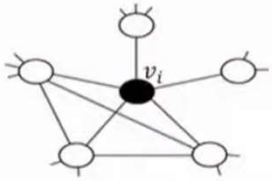
$k_i = 4$

$C_i = \frac{1}{6}$



$k_i = 4$

$C_i = \frac{2}{3}$

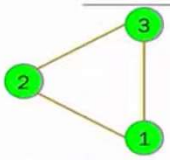


$k_i = 5$

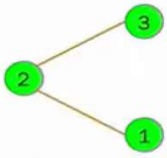
$C_i = \frac{3}{10}$

35

Global Clustering Coefficient



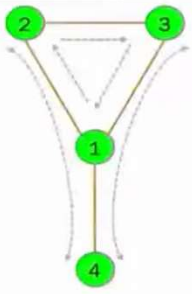
Closed Triplet



Open Triplet

❑The global clustering coefficient C of a network G is defined as

$$C = \frac{Total\ number\ of\ closed\ triplets\ in\ G}{Total\ number\ of\ triplets\ (open\ \&\ closed)\ in\ G}$$



G_2

36

Flow of Lecture by Headings

- Network Analysis: What is a Network?
- Mathematical Abstraction and Definition of Networks
- Network Types and Structure
  - Undirected, Directed, Weighted, and Special Edges
- Network Representation (Adjacency Matrix and List)
- Link-centric and Temporal Network Types
  - Unipartite, Bipartite, n-Partite, Time-Varying Networks
- Node Degree and Its Variants
  - Degree, Weighted Degree, Indegree, Outdegree
- Degree Distribution and Charts
  - Cumulative and **Complementary Cumulative Distributions**
- Power Law and Scale-Free Networks

37

Flow of Lecture by Headings

- Graph Preliminaries
  - Adjacency, Incidence, Walks, Paths, Cycles
- Network Metrics
  - Distance, Diameter, Average Path Length, Density, Connected Components
- Centrality Measures
  - Degree Centrality
  - Closeness Centrality
  - Betweenness Centrality
  - Eigenvector Centrality
- Network Modularity: Definition and Formula
- Community Detection and Modularity Maximization
  - Fast Greedy Algorithm
  - Louvain Method
- MST and PMFG

38

Analogy

Imagine fake news as a rumor:

- **Closeness**: Who can tell everyone fastest.
- **Betweenness**: Who connects different friend groups.
- **Eigenvector**: Who's friends with the most *popular* people.

39

Connected Components

- In a typical social network, there are loose links that connect the tightly-knit clusters
- In an undirected network G, two nodes v_i and v_j are said to be connected if there exists a path between v_i and v_j .
- An entire network is said to be connected if any pair of nodes in the network is connected.
- Connected subnetworks of a network, if exist, are called components of the network.
- In real-world networks, there often exist one giant component (consuming major chunk of nodes) and many smaller components.
- In a network, connectedness shows resilience to link breakdowns.

40

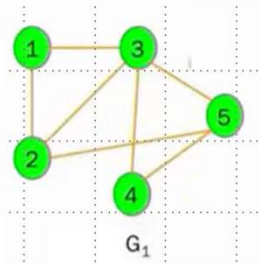
Degree Centrality

- The degree centrality $C_d(v)$ of a node v in a network $G(V, E)$ is defined as:

$$C_d(v) = \frac{\deg(v)}{\max_{u \in V} \deg(u)}$$

- Particularly useful for marketing scenarios, wherein the detected influential user can promote a product/service across her followers

- Degree centrality of the nodes 1 through 5 in network G_1 are $^2/_4$, $^3/_4$, $^4/_4$, $^2/_4$, and $^3/_4$, respectively; i.e., 0.5, 0.75, 1.0, 0.5, and 0.75, respectively. So, node 3 is most central according to degree centrality measure.



41

Closeness Centrality

- A means for detecting nodes that can spread information very efficiently through a graph
- The measure is useful in
  - Examining/restricting the spread of fake news/misinformation in social media
  - Examining/restricting the spread of a disease in epidemic modelling
  - Controlling/restricting the flow of vital information and resources within an organization (a terrorist network, for example)

Closeness Centrality measures how close a node is to all other nodes in a graph. It is calculated as the reciprocal of the sum of the shortest path distances from the node to all other nodes in the graph.

The formula is:

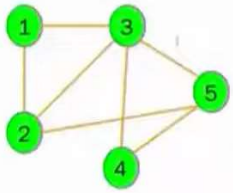
$$C(v) = \frac{|V| - 1}{\sum_{u \in V} d(u, v)}$$

Where:

- $C(v)$ is the closeness centrality of node v.
- $|V|$ is the total number of nodes in the graph.
- $d(u, v)$ is the shortest path distance between nodes u and v.

42

Closeness Centrality



G_1

- In graph G_1, the closeness centrality for the nodes are as follows

$$C(1) = \frac{5-1}{1+1+2+2} = \frac{4}{6} = 0.67$$

$$C(2) = \frac{5-1}{1+1+2+1} = \frac{4}{5} = 0.80$$

$$C(3) = \frac{5-1}{1+1+1+1} = \frac{4}{4} = 1.0$$

$$C(4) = \frac{5-1}{2+2+1+1} = \frac{4}{6} = 0.67$$

$$C(1) = \frac{5-1}{2+1+1+1} = \frac{4}{5} = 0.80$$

- Clearly, node 3 is most central according to closeness centrality measure

43

Betweenness Centrality

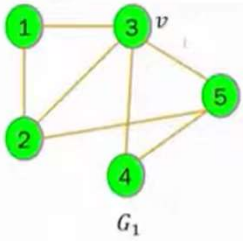
- A measure to compute how central a node is in between paths of the network
- A measure to compute how many (shortest) paths of the network pass through the node
- Useful in identifying
  - the articulation points, i.e., the points in a network which, if removed, may disconnect the network
  - The super spreaders in analyzing disease spreading in epidemiology
  - the suspected spies in security networks
- The betweenness centrality $C_B(v)$ of a node v in a network $G(V, E)$ is defined as

$$C_B(v) = \sum_{x,y \in V \setminus \{v\}} \frac{\sigma_{xy}(v)}{\sigma_{xy}}$$

where σ_{xy} denotes the number of shortest paths between nodes x and y in the network, $\sigma_{xy}(v)$ denotes the same passing though v. If $x = y$, then $\sigma_{xy} = 1$.

44

Betweenness Centrality



G_1

- To find the betweenness centrality of node $v = 3$ in graph G_1
- The following matrix is of the form $\sigma_{xy}(v)|\sigma_{xy}$

$\sigma_{xy}(v)\|\sigma_{xy}$	1	2	3	4	5
1	0|1	0|1	–	**1|1**	**1|2**
2	0|1	0|1	–	**1|2**	0|1
3	–	–	–	–	–
4	**1|1**	**1|2**	–	0|1	0|1
5	**1|2**	0|1	–	0|1	0|1

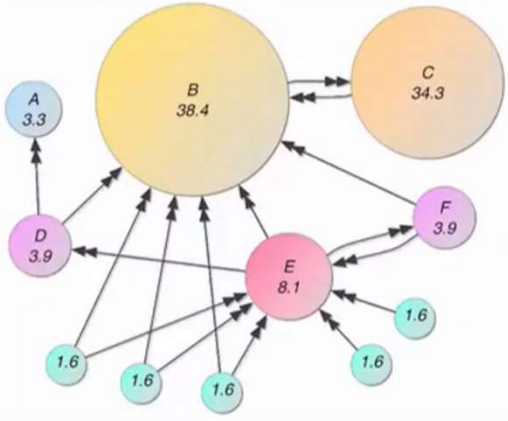
- Thus the betweenness centrality of node $3 = \frac{1}{1}+\frac{1}{2}+\frac{1}{2}+\frac{1}{1}+\frac{1}{2}+\frac{1}{2} = 4$

A node with **high betweenness**

- Lies on many shortest paths → it **controls the flow of information**.
- Acts as a **bridge** or **broker** between communities — can control or delay information.

45

Eigenvector Centrality



- Measures a node's importance by taking into consideration the preference of its neighbors
- Uses a recursive approach
- A node has a higher eigenvector centrality, if it is directly connected to other nodes having high eigenvector centrality
- Generally applied on directed networks

46

Eigenvector Centrality

- The eigenvector centrality x_v of a node v in a network $G(V, E)$ is given by

$$x_v = \frac{1}{\lambda_1} \sum_{t \in N(v)} x_t = \frac{1}{\lambda_1} \sum_{t \in V} (a_{vt} \times x_t)$$

where λ_1 is the largest eigen value of the matrix $A = (a_{ij})$, the adjacency matrix of the network G

- The largest eigen value λ_1 is obtained by solving the equation

$$A.X = \lambda_1.X$$

- X above is a column vector, whose v^{th} entry is x_v , the eigen vector centrality of the node v

47

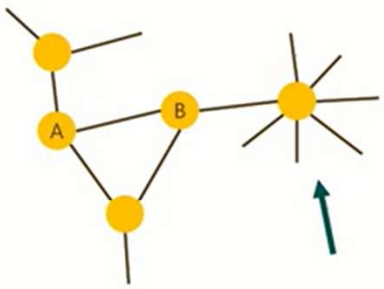
Ideas Behind Eigenvector Centrality

The importance of a node is defined by:

(1) How many connections it has (i.e., its degree)
(2) How important the connections are

Quantity & Quality of Connections

A node with few connections could have a very high eigenvector centrality if those few connections are very important (i.e., have high eigenvector centrality).

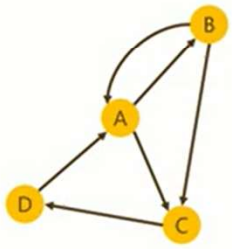


degree(A) = degree(B) = 3

48

Calculate Eigenvector Centrality Score

Calculate in a recursive way until the mutual influence between nodes reaches a stable state (convergence).



Round	A	B	C	D
0	1	1	1	1
1	2	1	2	1
2	2	2	3	2
3	4	2	4	3
4	5	4	6	4
...	...	...	...	...

- A = B + D = 1 + 1 = 2
- B = A = 1
- C = A + B = 1 + 1 = 2
- D = C = 1

A node receives influence from its neighbors through incoming edges.

In each round, update the score of each node by the sum of scores of all its in-neighbors.

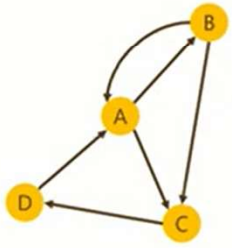
49

Calculate Eigenvector Centrality Score

Calculate in a recursive way until the mutual influence between nodes reaches a stable state (convergence).

L2 Normalization:

$$s_A = \frac{s_A}{\sqrt{s_A^2 + s_B^2 + s_C^2 + s_D^2}}$$



Round	A	B	C	D
0	1	1	1	1

A node receives influence from its neighbors through incoming edges.

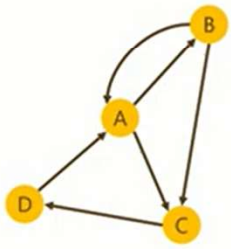
50

Calculate Eigenvector Centrality Score

L2 Normalization:

$$s_A = \frac{s_A}{\sqrt{s_A^2 + s_B^2 + s_C^2 + s_D^2}}$$

Calculate in a recursive way until the mutual influence between nodes reaches a stable state (convergence).



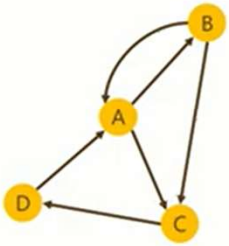
Round	A	B	C	D
0	1	1	1	1
1	0.632	0.316	0.632	0.316
2	0.436	0.436	0.655	0.436
3	0.596	0.298	0.596	0.447
...	...	...	...	...
19	0.543	0.370	0.623	0.425
20	0.543	0.370	0.623	0.425

A node receives influence from its neighbors through incoming edges.

51

The Secret of Convergence

1 **The calculation process is essentially matrix multiplication.**



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad s^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Adjacency Matrix (Incoming Edges)

Initial Score Vector

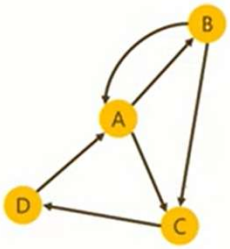
Round	Score Vector
1	$s^{(1)} = As^{(0)}$
2	$s^{(2)} = As^{(1)} = A^2 s^{(0)}$
...	...
k	$s^{(k)} = A^k s^{(0)}$

$$s^{(1)} = As^{(0)} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0\times1+1\times1+0\times1+1\times1 \\ 1\times1+0\times1+0\times1+0\times1 \\ 1\times1+1\times1+0\times1+0\times1 \\ 0\times1+0\times1+1\times1+0\times1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} \xrightarrow{L2\ Norm} \begin{bmatrix} 0.632 \\ 0.316 \\ 0.632 \\ 0.316 \end{bmatrix}$$

52

The Secret of Convergence

2 Given A is a square matrix, λ is a constant, x is a non-zero vector. If $Ax = \lambda x$ is true, then λ is called the **eigenvalue** of A, and x is the **eigenvector** that corresponds to λ.



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

Adjacency Matrix
(Incoming Edges)

$$s^{(20)} = \begin{bmatrix} 0.543 \\ 0.370 \\ 0.623 \\ 0.425 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} 0.543 \\ 0.370 \\ 0.623 \\ 0.425 \end{bmatrix} \quad \lambda_1 = 1.466$$

$$x_2 = \begin{bmatrix} 0.019 - 0.335i \\ -0.395 + 0.093i \\ -0.153 + 0.520i \\ 0.656 - 0.000i \end{bmatrix} \quad \lambda_2 = -0.233 + 0.793i$$

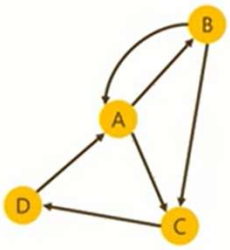
$$x_3 = \begin{bmatrix} 0.019 + 0.335i \\ -0.395 - 0.093i \\ -0.153 - 0.520i \\ 0.656 + 0.000i \end{bmatrix} \quad \lambda_3 = -0.233 - 0.793i$$

$$x_4 = \begin{bmatrix} -0.707 \\ 0.707 \\ 0.000 \\ 0.000 \end{bmatrix} \quad \lambda_4 = -1.000$$

53

The Secret of Convergence

2 Given A is a square matrix, λ is a constant, x is a non-zero vector. If $Ax = \lambda x$ is true, then λ is called the **eigenvalue** of A, and x is the **eigenvector** that corresponds to λ.



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

Adjacency Matrix
(Incoming Edges)

$$s^{(20)} = \begin{bmatrix} 0.543 \\ 0.370 \\ 0.623 \\ 0.425 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} 0.543 \\ 0.370 \\ 0.623 \\ 0.425 \end{bmatrix} \quad \lambda_1 = 1.466$$

Dominant Eigenvalue

$$x_2 = \begin{bmatrix} 0.019 - 0.335i \\ -0.395 + 0.093i \\ -0.153 + 0.520i \\ 0.656 - 0.000i \end{bmatrix} \quad \lambda_2 = -0.233 + 0.793i$$

$$x_3 = \begin{bmatrix} 0.019 + 0.335i \\ -0.395 - 0.093i \\ -0.153 - 0.520i \\ 0.656 + 0.000i \end{bmatrix} \quad \lambda_3 = -0.233 - 0.793i$$

$$x_4 = \begin{bmatrix} -0.707 \\ 0.707 \\ 0.000 \\ 0.000 \end{bmatrix} \quad \lambda_4 = -1.000$$

54

The Secret of Convergence

3 **Perron–Frobenius theorem:** Given matrix A has eigenvalues $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$ and the corresponding eigenvectors $x_1, x_2, x_3, \ldots, x_n$. When $k \to \infty$:

$$s^{(k)} = A^k s^{(0)} \to x_1$$

Power Iteration

Calculate eigenvector centrality for a graph represented by adjacency matrix A:

- 1st iteration: $s^{(1)} = \frac{As^{(0)}}{\|As^{(0)}\|_2}$
- 2nd iteration: $s^{(2)} = \frac{As^{(1)}}{\|As^{(1)}\|_2}$
- ...
- k-th iteration: $s^{(k)} = \frac{As^{(k-1)}}{\|As^{(k-1)}\|_2}$

55

Demo Dataset: Host-Parasite Network

Many diseases in human arise from animals. Identifying high-risk animal hosts of parasites would help to predict and prevent new pandemics.

Node

Schema: **@host**

- Number of nodes: 217

Edge

Schema: **@shareParasites**

- Number of edges: 15436
- Property:
  - weight

Dataset downloaded from https://parasites.nunn-lab.org/ and modified.

56

Flow of Lecture by Headings

- Graph Preliminaries
  - Adjacency, Incidence, Walks, Paths, Cycles
- Network Metrics
  - Distance, Diameter, Average Path Length, Density, Connected Components
- Centrality Measures
  - Degree Centrality
  - Closeness Centrality
  - Betweenness Centrality
  - Eigenvector Centrality
- MST and PMFG
- Network Modularity: Definition and Formula
- Community Detection and Modularity Maximization
  - Fast Greedy Algorithm
  - Louvain Method

57

Network Modularity

Modularity is a measure used to evaluate the quality of a network partition into **communities** or **clusters**. It quantifies the strength of division of a network into modules by comparing the actual connections in the network to the expected connections in a random network with the same degree distribution.

The modularity Q is defined as:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

Where:

- m: Total number of edges in the network.
- A_{ij}: Adjacency matrix entry (1 if there is an edge between nodes i and j, 0 otherwise).
- k_i, k_j: Degrees of nodes i and j, respectively.
- $\frac{k_i k_j}{2m}$: Expected number of edges between i and j in a random network.
- $\delta(c_i, c_j)$: Kronecker delta, which is 1 if nodes i and j belong to the same community and 0 otherwise.

58

High Modularity Example

Social Network with Clear Communities

•**Example:** Facebook or LinkedIn network divided into friend groups — e.g., college friends, family, and coworkers.

•**Explanation:**
Within each group, people are highly interconnected (many mutual friends).
Between groups, connections are sparse (you rarely connect your family to your work circle).

•**Result:** Strongly defined communities → **High modularity (Q ≈ 0.4–0.7)**

Only a few links connect these clusters.

Low Modularity Example

Airline Transportation Network
•**Example:** Global flight network connecting major airports (e.g., Delhi, Dubai, London, New York).

•**Explanation:**
Airports are highly interconnected globally — there are many inter-community connections (international flights), so it's hard to divide the network into isolated clusters.

•**Result:**
Weakly defined communities → **Low modularity (Q ≈ 0.05–0.2)**

Almost every node connects to multiple clusters.

59

Feature	High Modularity	Low Modularity
Community separation	Strong	Weak
Intra-group links	Many	Moderate
Inter-group links	Few	Many
Typical Q value	0.4–0.7	0–0.2
Real-life example	Friend groups, ecosystems	Airline network, global web

Domain	High Modularity	Low Modularity
Academic collaboration	Research fields (e.g., physicists vs. economists)	Interdisciplinary conferences (lots of overlap)
Corporate email network	Teams working on separate projects	Company-wide open communication culture
Online communities	Subreddits with focused topics	Twitter/X trending discussions where everyone connects globally

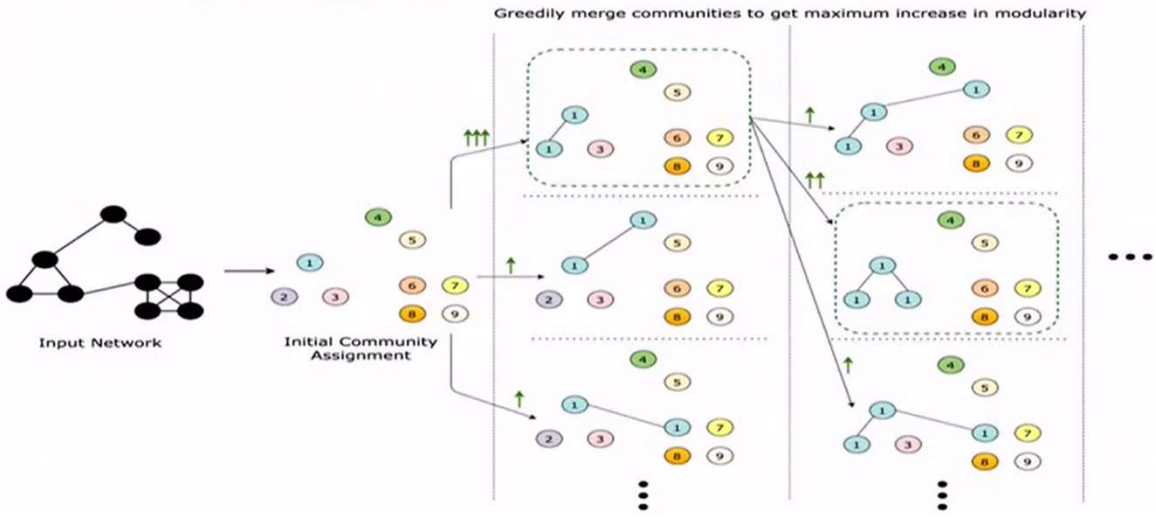
60

Community Detection: Modularity Maximization

- Modularity can be positive, negative, and zero
  - Positive modularity shows presence of strong community structure
- Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.
- Different community assignments can lead to different values of modularity
- an assignment that maximizes the modularity of the overall network often finds the communities in the network
  - Fast Greedy Algorithm [Clauset et al. (2004)]
  - Louvain Method

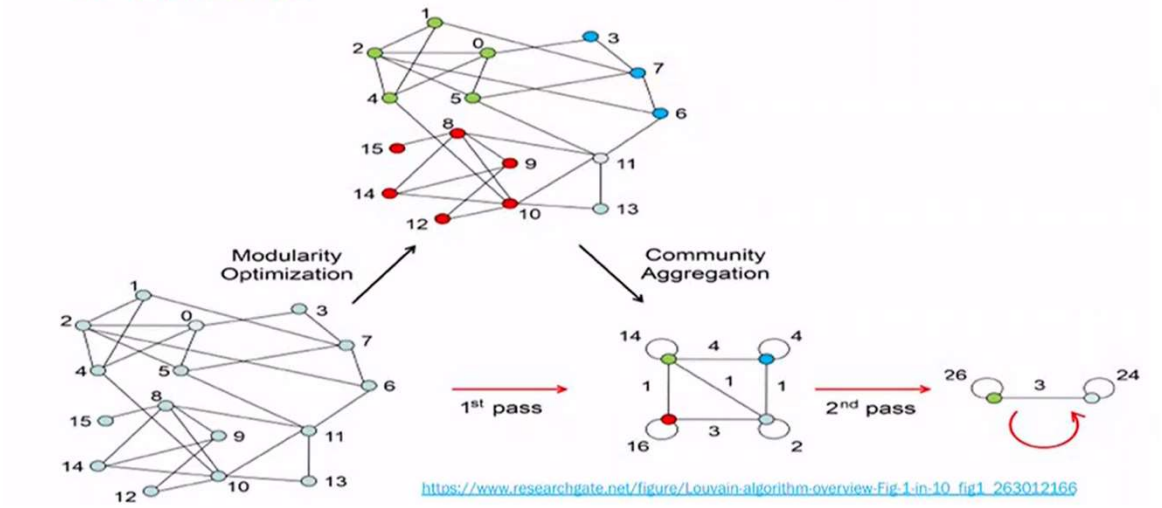
61

Community Detection: Fast Greedy Algorithm



62

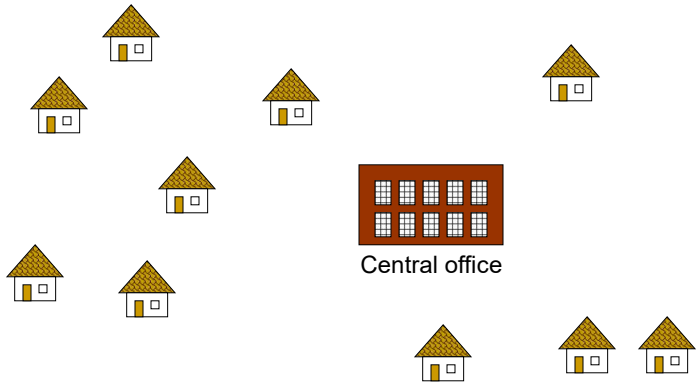
Community Detection: Louvain Method



63

64

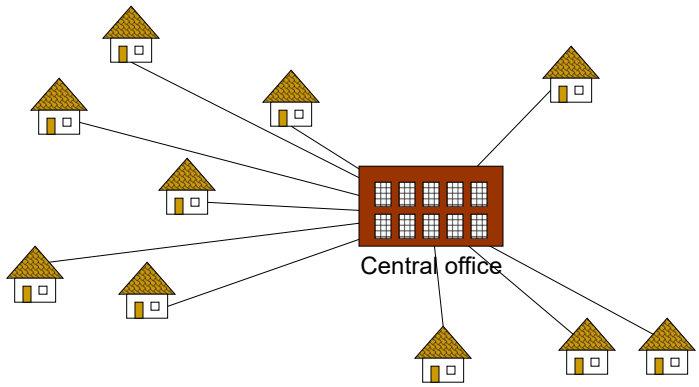
Problem: Laying Telephone Wire



65

65

Wiring: Naive Approach

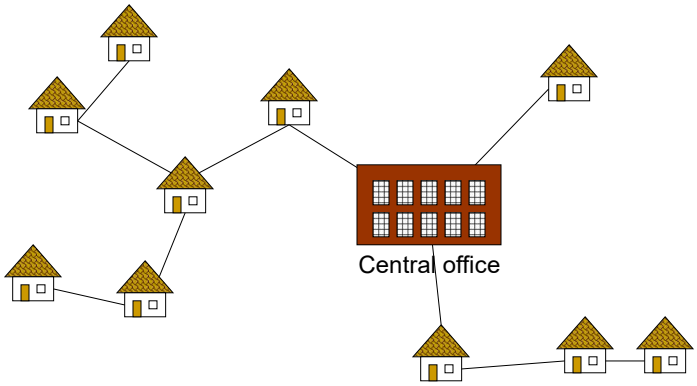


Expensive!

66

66

Wiring: Better Approach

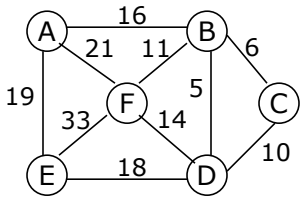


Minimize the total length of wire connecting the customers

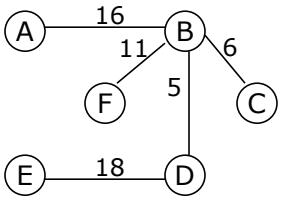
67

Minimum-cost spanning trees

- Suppose you have a connected undirected graph with a weight (or cost) associated with each edge
- The cost of a spanning tree would be the sum of the costs of its edges
- A minimum-cost spanning tree is a spanning tree that has the lowest cost



A connected, undirected graph



A minimum-cost spanning tree

68

Minimum Spanning Tree (MST)

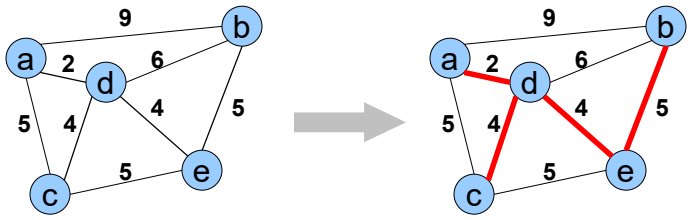
A **minimum spanning tree** is a subgraph of an undirected weighted graph G, such that

- it is a tree (i.e., it is acyclic)
- it covers all the vertices V
  - contains $|V| - 1$ edges
- the total cost associated with tree edges is the minimum among all possible spanning trees
- not necessarily unique

69

69

How Can We Generate a MST?



70

70

Prim(-Jarnik)'s Algorithm

- Similar to Dijkstra's algorithm (for a connected graph)
- We pick an arbitrary vertex ***s*** and we grow the MST as a cloud of vertices, starting from ***s***
- We store with each vertex ***v*** a label ***d***(***v***) = the smallest weight of an edge connecting ***v*** to a vertex in the cloud

- At each step:
  - We add to the cloud the vertex u outside the cloud with the smallest distance label
  - We update the labels of the vertices adjacent to u

71

71

Prim's algorithm

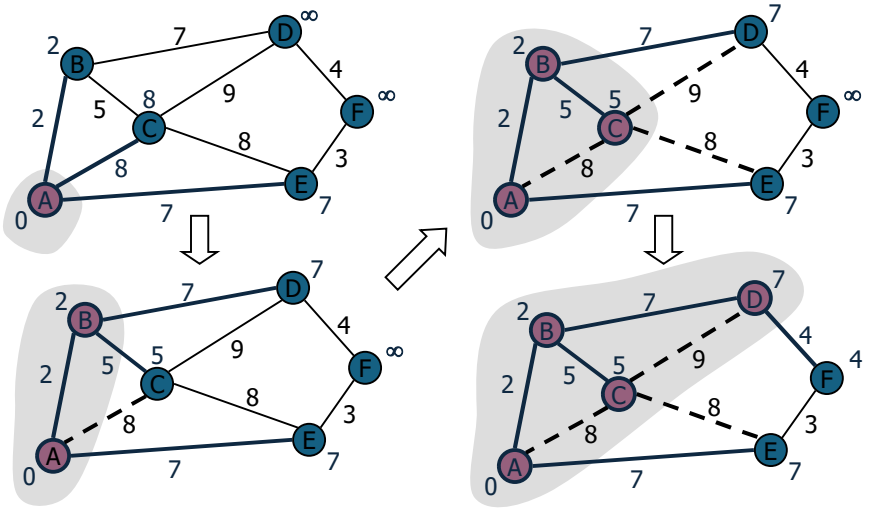
```
T = a spanning tree containing a single node s;
E = set of edges adjacent to s;
while T does not contain all the nodes {
    remove an edge (v, w) of lowest cost from E
    if w is already in T then discard edge (v, w)
    else {
        add edge (v, w) and node w to T
        add to E the edges adjacent to w
    }
}
```

- An edge of lowest cost can be found with a priority queue
- Testing for a cycle is automatic
  - Hence, Prim's algorithm is far simpler to implement than Kruskal's algorithm (presented below)

72

72

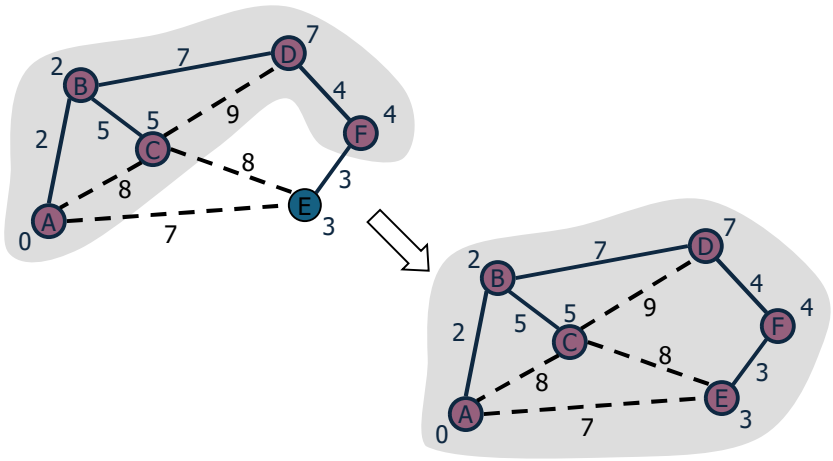
Example



73

73

Example (contd.)



74

74

Prim's Algorithm Invariant

- At each step, we add the edge (***u,v***) s.t. the weight of (***u,v***) is **minimum** among all edges where ***u*** is in the tree and ***v*** is not in the tree
- Each step maintains a minimum spanning tree of the vertices that have been included thus far
- When all vertices have been included, we have a MST for the graph!

75

75

Correctness of Prim's

- This algorithm adds *n-1* edges without creating a cycle, so clearly it creates a spanning tree of any connected graph (*you should be able to prove this*).

But is this a *minimum* spanning tree?

Suppose it wasn't.

- There must be point at which it fails, and in particular there must a single edge whose insertion first prevented the spanning tree from being a minimum spanning tree.

76

76

Kruskal's algorithm

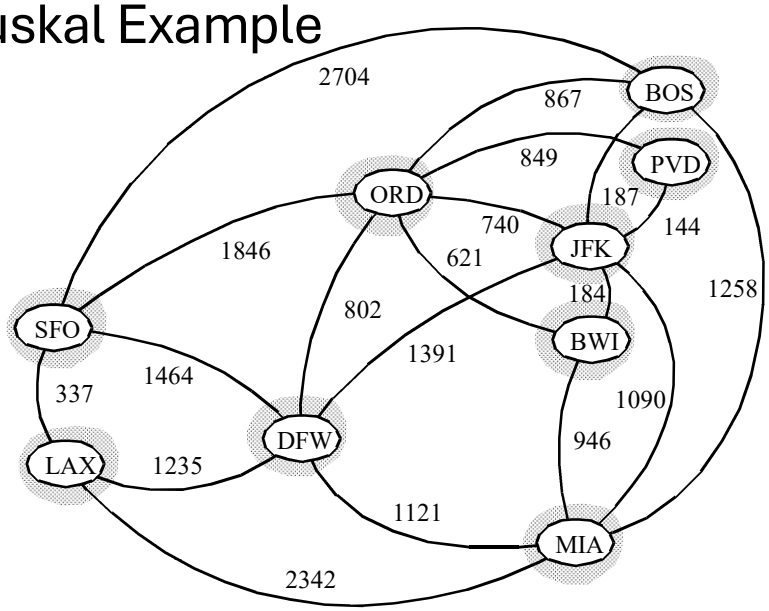
```
T = empty spanning tree;
E = set of edges;
N = number of nodes in graph;
while T has fewer than N - 1 edges {
    remove an edge (v, w) of lowest cost from E
    if adding (v, w) to T would create a cycle
        then discard (v, w)
        else add (v, w) to T
}
```

- Finding an edge of lowest cost can be done just by sorting the edges
- **Testing for a cycle**: Efficient testing for a cycle requires a additional algorithm (UNION-FIND) which we don't cover in this course. The main idea: If both nodes v, w are in the same component of T, then adding (v, w) to T would result in a cycle.

77

77

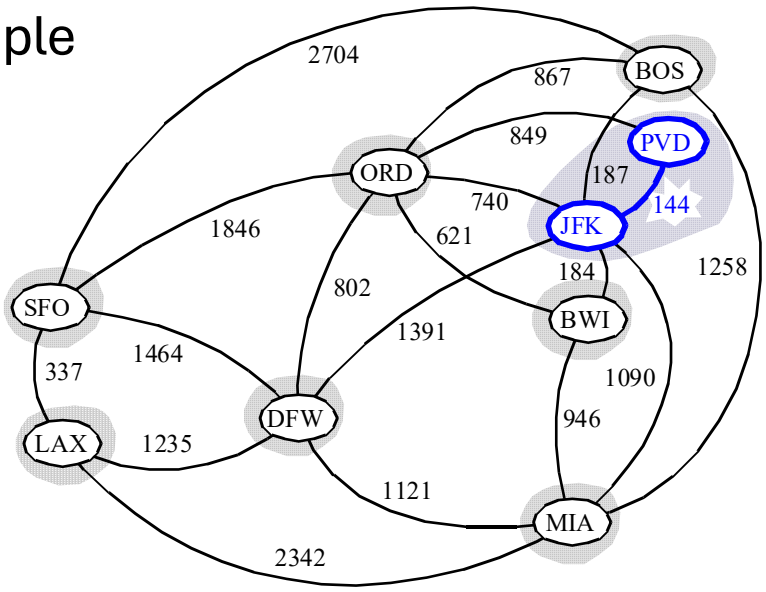
Kruskal Example



78

78

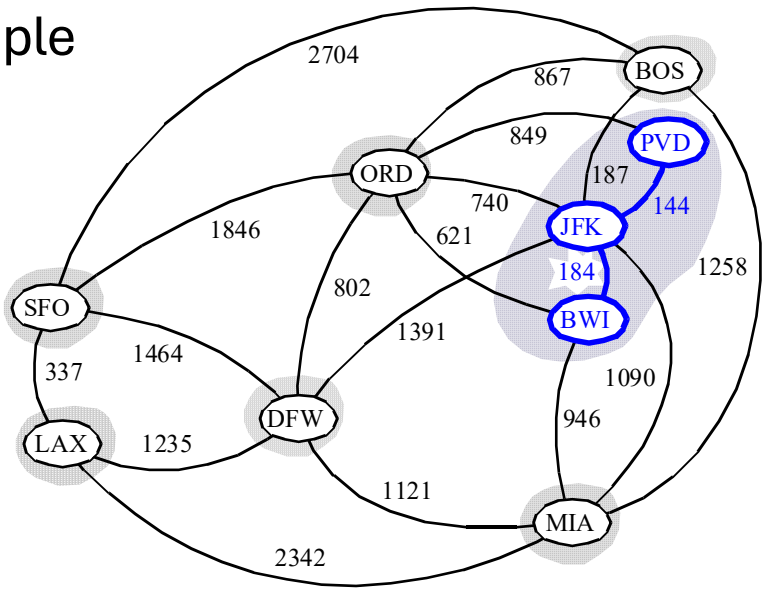
Example



79

79

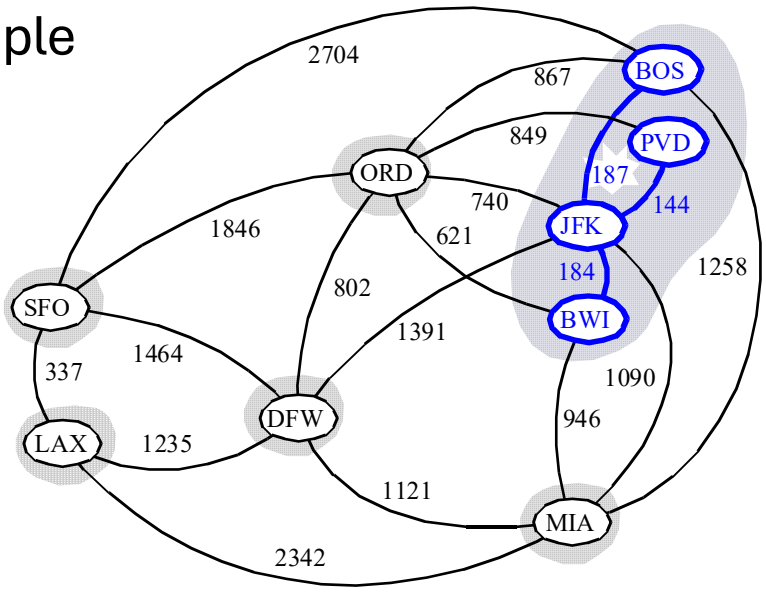
Example



80

80

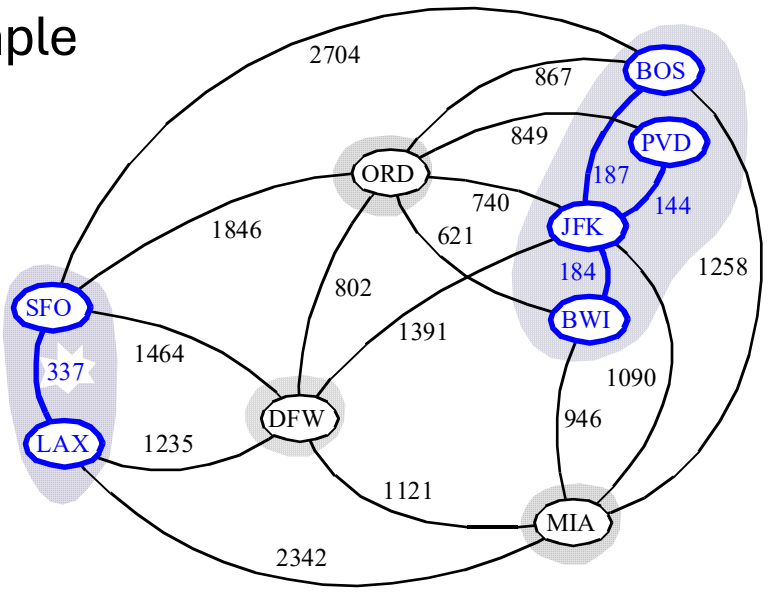
Example



81

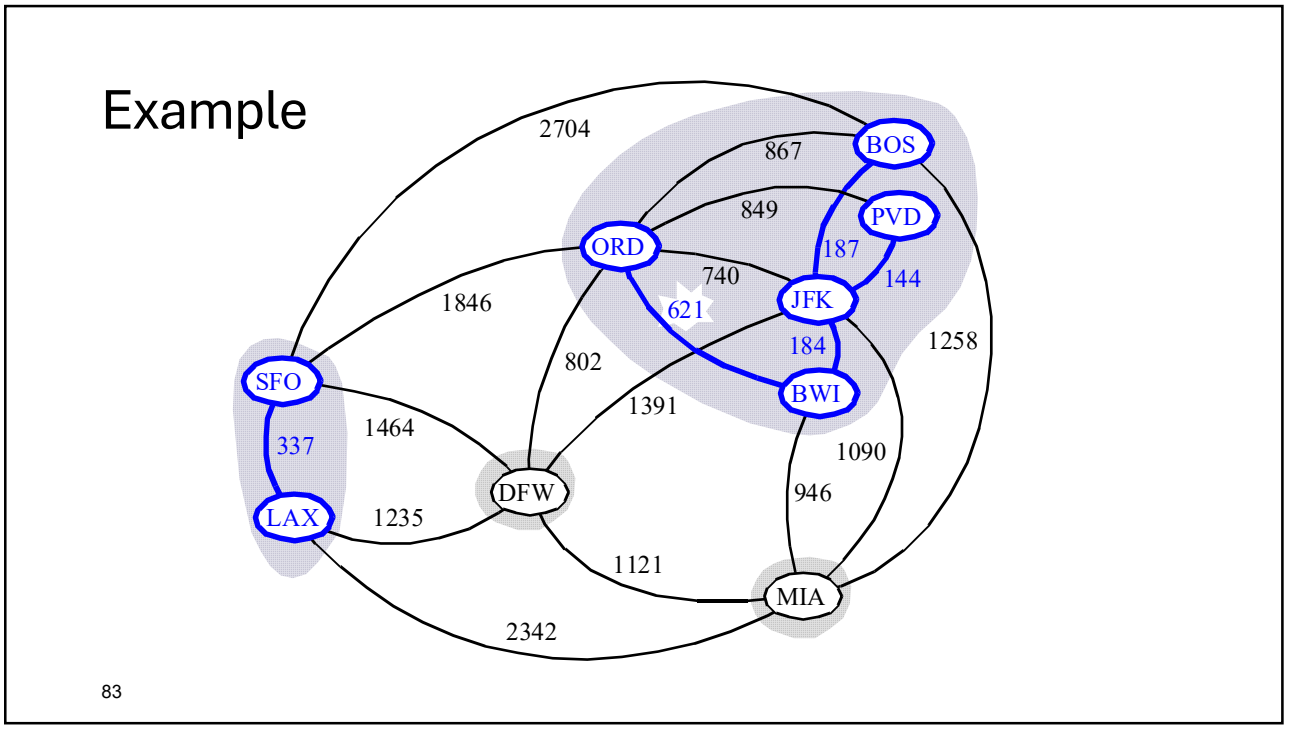
81

Example

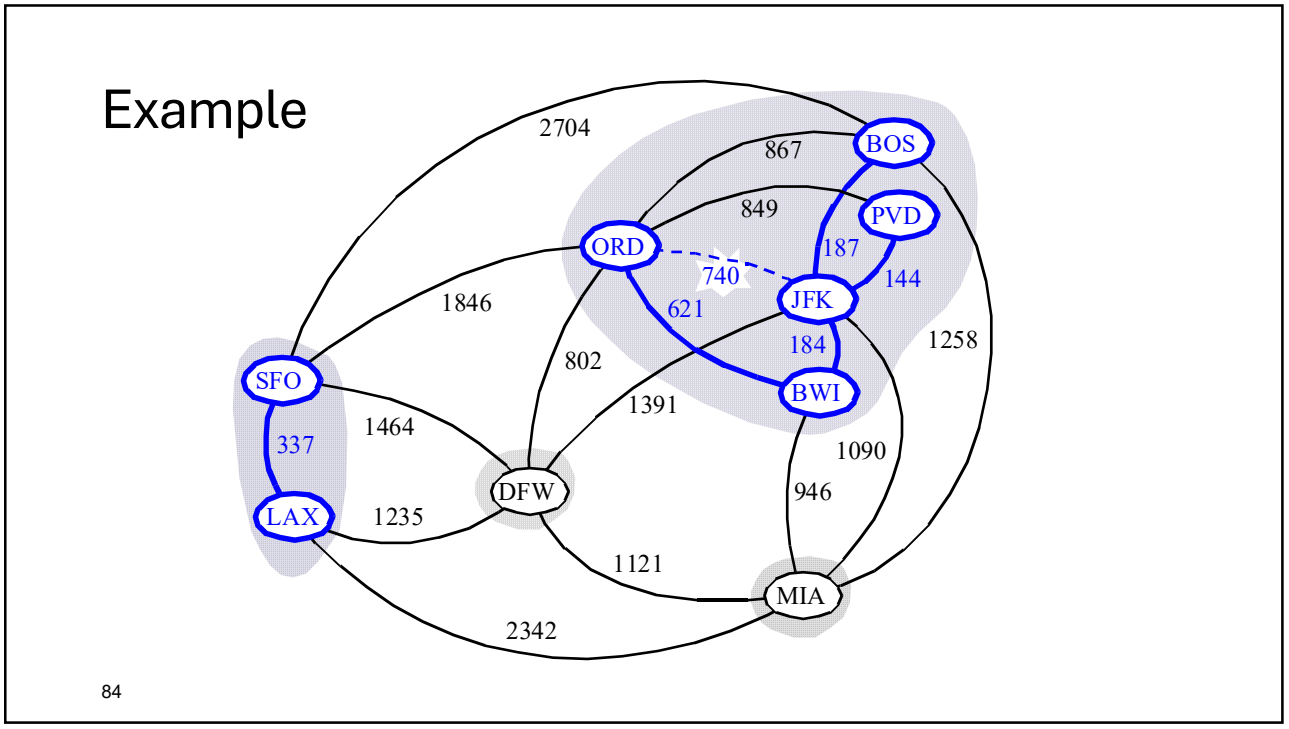


82

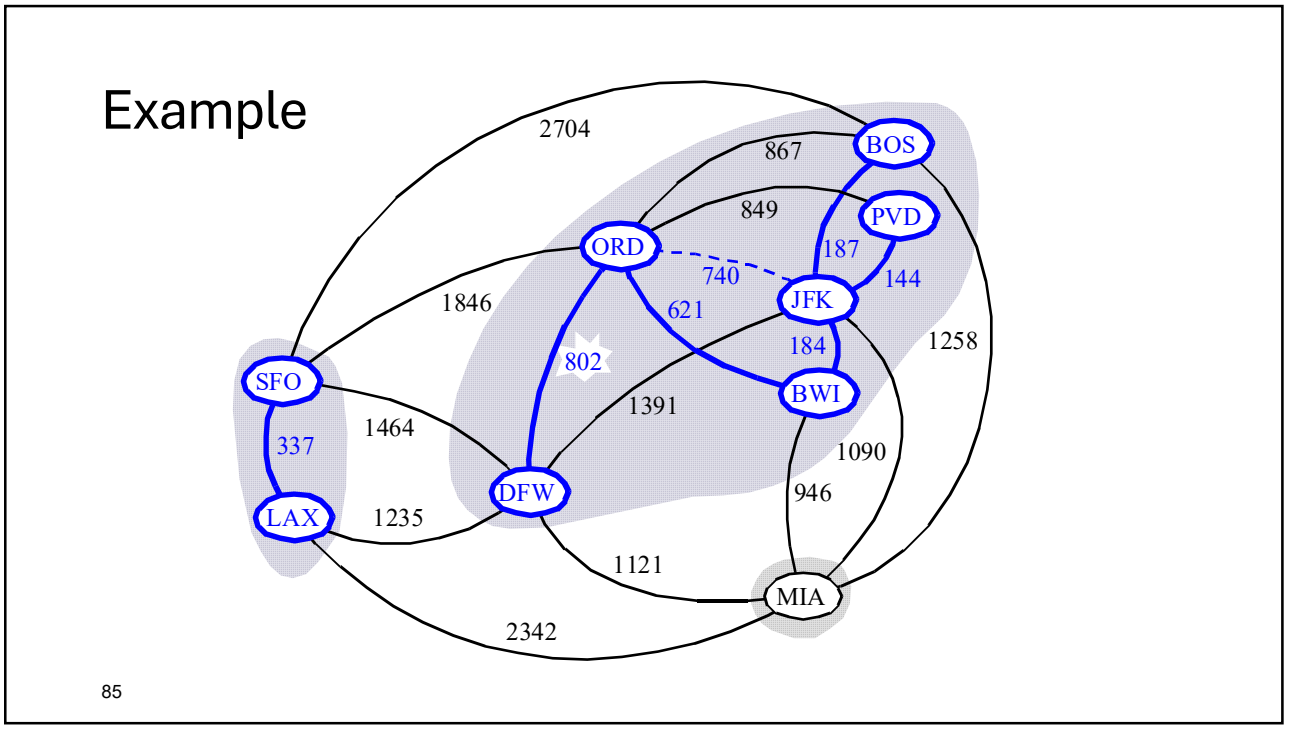
82


Example
2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
184
1258
802
SFO
BWI
1391
1464
337
1090
946
DFW
LAX
1235
1121
MIA
2342
83

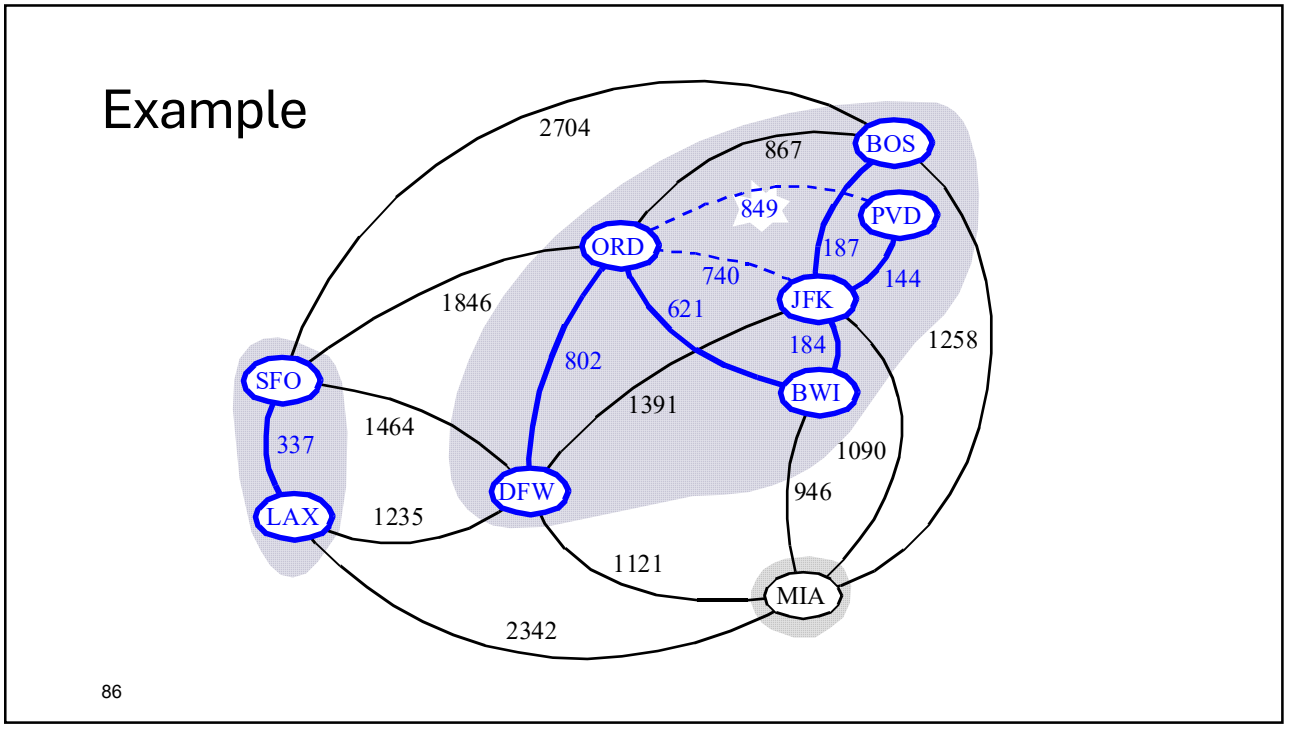
83


Example
2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
184
1258
802
SFO
BWI
1391
1464
337
1090
946
DFW
LAX
1235
1121
MIA
2342
84

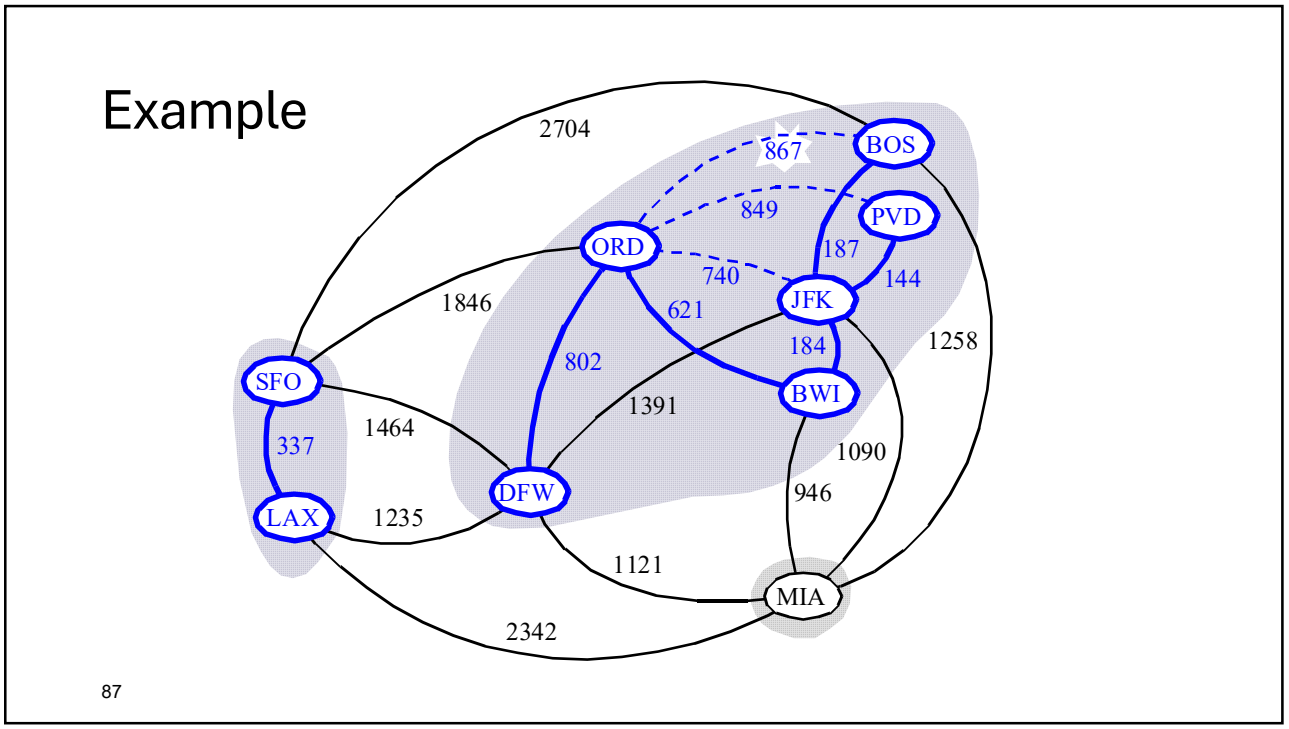
84


Example
2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
184
1258
802
SFO
BWI
1391
1464
337
1090
946
DFW
LAX
1235
1121
MIA
2342
85

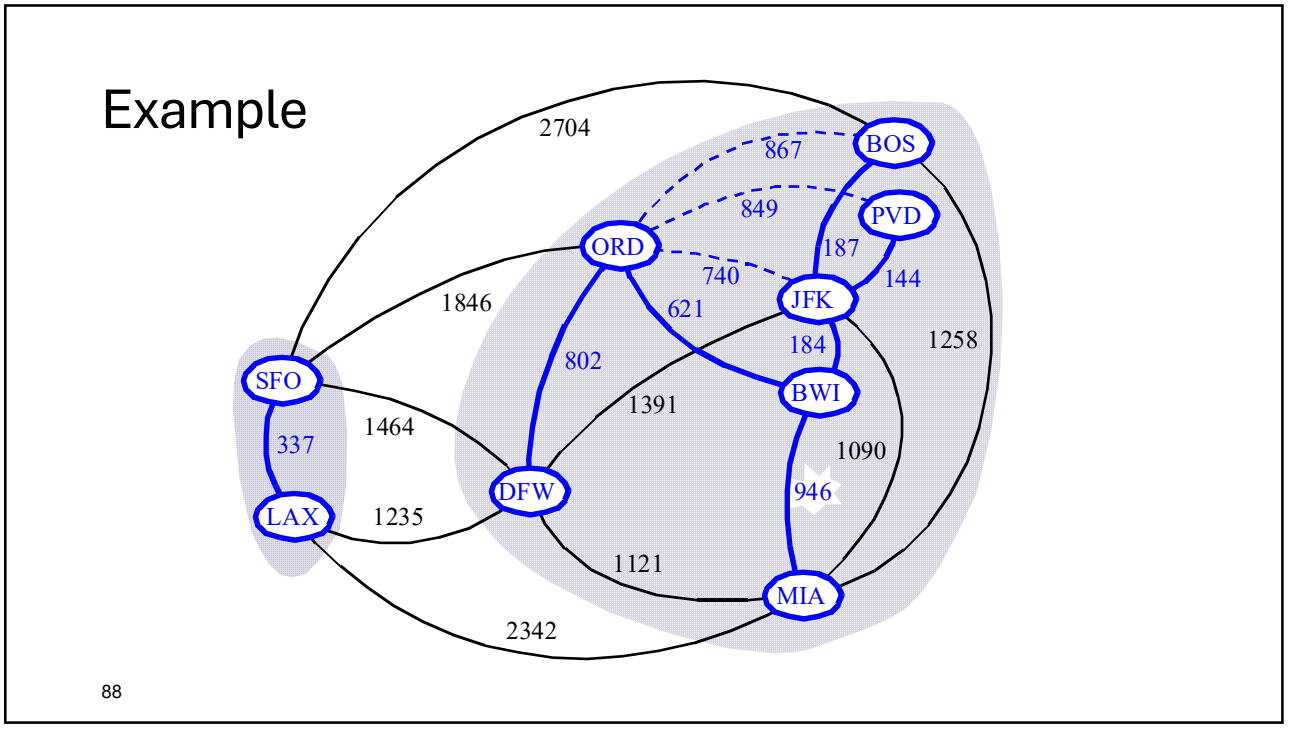
85


Example
2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
184
1258
802
SFO
BWI
1391
1464
337
1090
946
DFW
LAX
1235
1121
MIA
2342
86

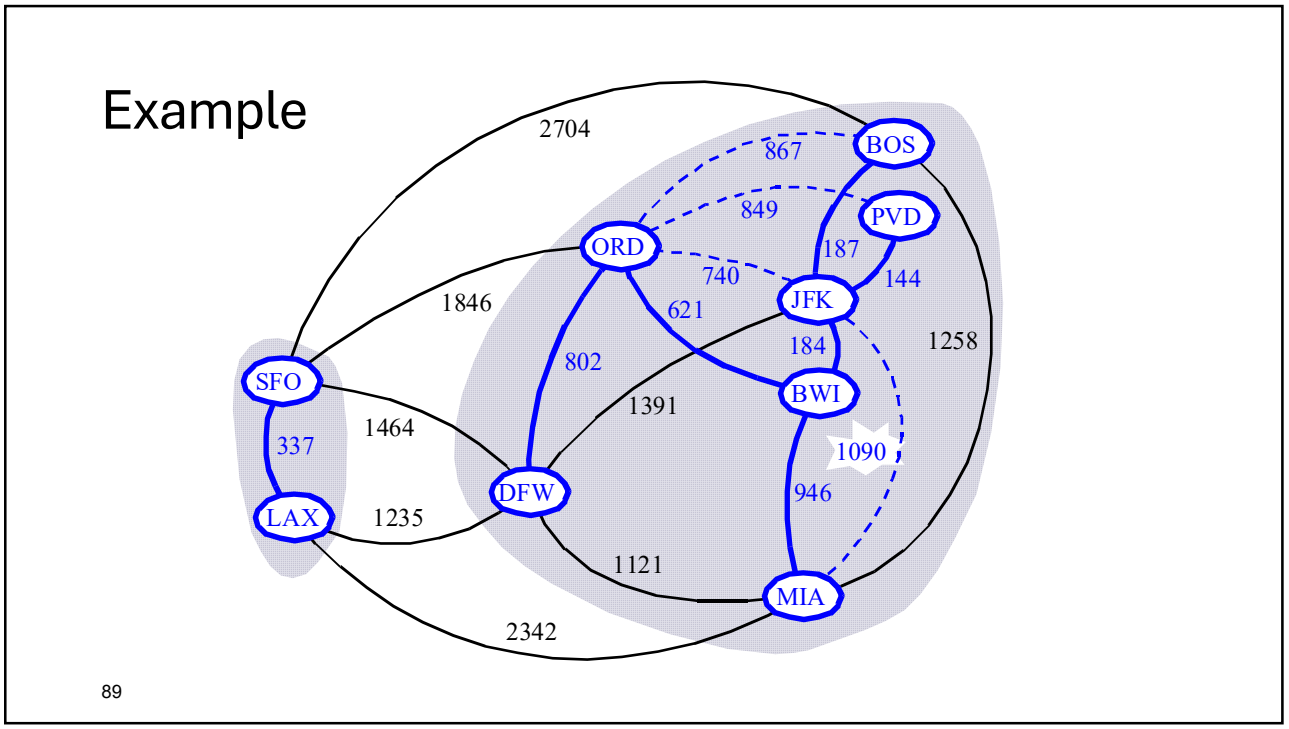
86


Example
2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
184
1258
802
SFO
BWI
1391
1464
337
1090
946
DFW
LAX
1235
1121
MIA
2342
87

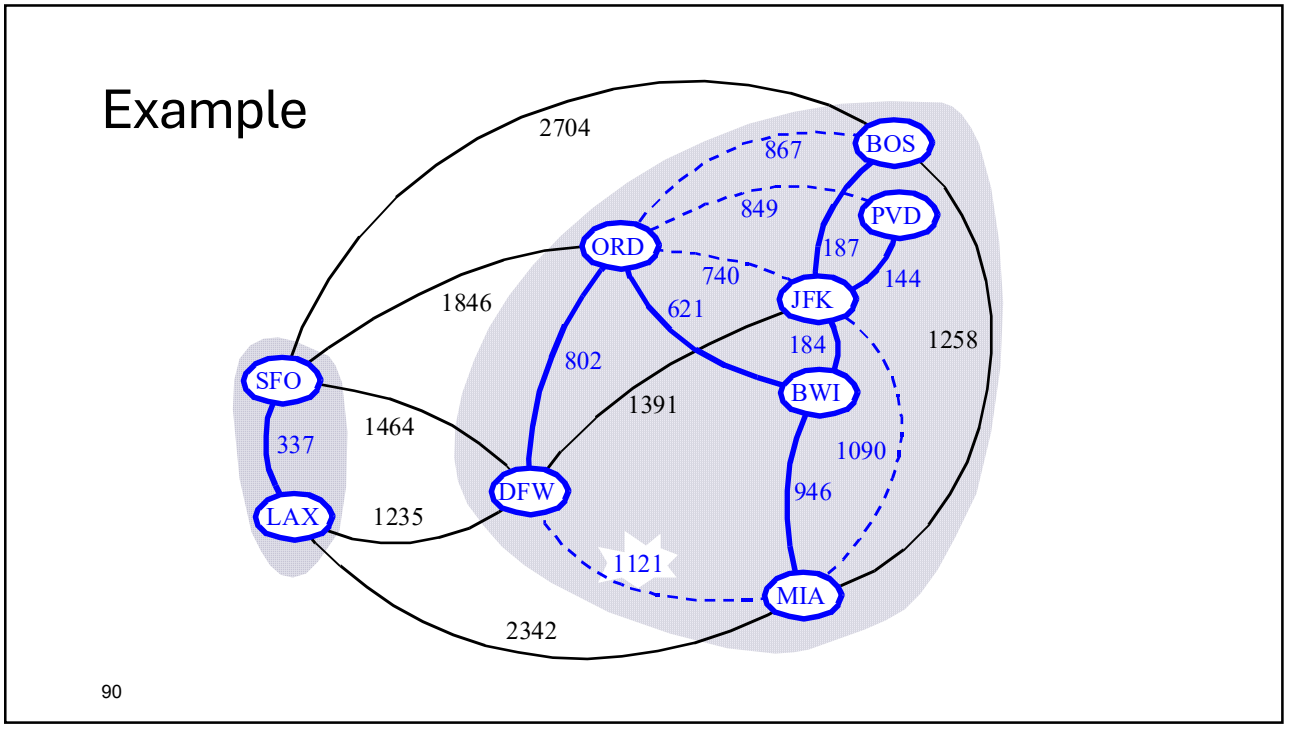
87


Example
2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
184
1258
802
SFO
BWI
1391
1464
337
1090
946
DFW
LAX
1235
1121
MIA
2342
88

88

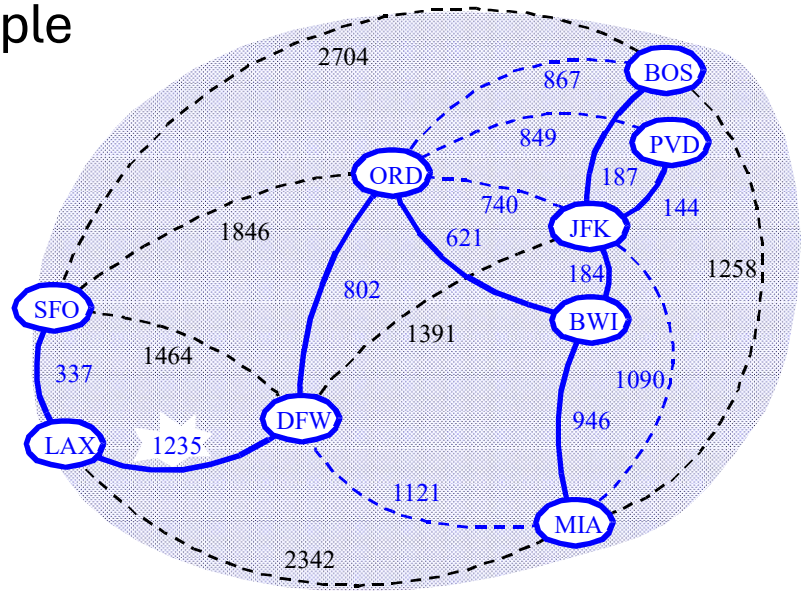

Example
2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
184
1258
802
SFO
BWI
1391
1464
337
1090
946
DFW
LAX
1235
1121
MIA
2342
89

89


Example
2704
867
BOS
849
PVD
ORD
187
740
144
JFK
1846
621
184
1258
802
SFO
BWI
1391
1464
337
1090
946
DFW
LAX
1235
1121
MIA
2342
90

90

Example



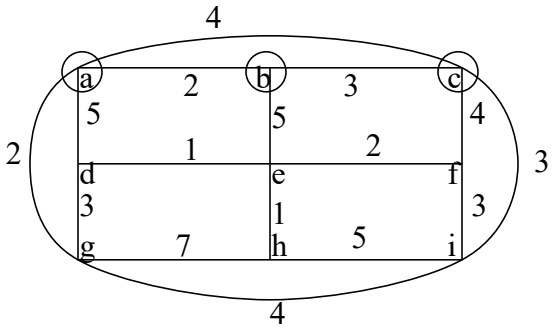
91

91

Feature	**Prim's Algorithm**	**Kruskal's Algorithm**
Approach	Grows a single connected tree	Merges multiple small trees (forests)
Starting Point	Starts with any vertex	Starts with all edges sorted
Edge Selection	Chooses minimum edge **connected to the existing tree**	Chooses **globally smallest** edge that doesn't form a cycle
Cycle Checking	Automatically avoided (edges always connect tree to new vertex)	Must explicitly check for cycles (using Disjoint Set / Union-Find)
Data Structure Used	Priority Queue or Min Heap	Disjoint Set Union (DSU) or Union-Find
Graph Type Suitability	Works well with **dense graphs**	Works well with **sparse graphs**
Time Complexity	(O(E + V log V)) with Min Heap	(O(E log E)) due to sorting edges
Edge Consideration	Only considers edges connected to the current tree	Considers all edges globally
Example of Execution	Similar to Dijkstra's logic (expanding vertex set)	Similar to sorting and merging clusters

92

MST with Prim's and Kruskal algorithm



93

93

Planar Maximally Filtered Graph (PMFG)

A Planar Maximally Filtered Graph (PMFG) is a graph created from a **correlation-based network** to filter out the most relevant information, like connections between stocks. It is built using a **greedy** algorithm that adds the highest-weight edges while ensuring the **graph remains planar** (**can be drawn without edge crossings**). The resulting PMFG preserves the **hierarchical structure of a minimum spanning tree (MST)** but contains more links, providing a richer and more robust structure for analysis.

A **Planar Maximally Filtered Graph (PMFG)** is a **network filtering method** used to extract the most significant relationships from a fully connected weighted network (typically derived from correlations).
It was first proposed by **Tumminello, Aste, Di Matteo, and Mantegna (PNAS, 2005)** as an extension of the **Minimum Spanning Tree (MST)** to retain **more information** while ensuring the **planarity constraint**.

PMFG plays a crucial role in **complex systems**, **econophysics**, **financial networks**, and **biological systems** for discovering meaningful structures hidden in large correlation matrices.

94

Concept

When we compute correlations (e.g., between financial assets, stocks, or time series), we obtain a **fully connected weighted graph** with $N(N-1)/2$ edges.

This graph is too dense to interpret. Filtering methods aim to:

- Keep only the **most important connections** (high correlations).
- Preserve **topological interpretability**.
- Reduce **redundancy and noise**.

Common filtering approaches:

- **Thresholding** (remove weak correlations) → may break connectivity.
- **Minimum Spanning Tree (MST)** → keeps connectivity, but too sparse.
- **Planar Maximally Filtered Graph (PMFG)** → balances both.

The PMFG is a **planar graph**, meaning it can be drawn on a 2D plane **without edge crossings**.
It is **maximally planar**, i.e., **no more edges can be added** without violating planarity.

For a graph with N —nodes:

$$E_{PMFG} = 3(N-2)$$

which is much larger than MST's $N-1$edges.

95

The PMFG is built **greedily** from a correlation (or distance) matrix:

1.Compute the Correlation Matrix

$$\rho_{ij} = corr(X_i, X_j)$$

Often converted to a **distance measure**:

$$d_{ij} = \sqrt{2(1-\rho_{ij})}$$

2. Sort all edges

1. Sort pairs (i, j)in **descending order of correlation** (or ascending order of distance).

3. Initialize an empty graph with all nodes but no edges.

4. Iteratively add edges

1. Add the next strongest edge **only if** the graph remains **planar**.
2. Planarity can be checked via algorithms like:
   1. **Kuratowski's theorem**
   2. or computational planarity check (Hopcroft–Tarjan algorithm).

Stop when the graph reaches **3(N - 2)** edges — maximal planarity

Mathematical Properties

1. Planarity

A planar graph satisfies:

$$E \leq 3N - 6$$

For PMFG, equality holds (maximal planar).

2. Connectivity

- PMFG is **always connected** if constructed from connected correlation data.
- MST is a **subset** of the PMFG:

3. Hierarchical Preservation

- PMFG **preserves the hierarchy** derived from MST.
- All edges of MST appear in PMFG.
- The additional edges create **loops** and **cliques** (especially 3- and 4-cliques).

4. Clique Structure

- PMFG can contain up to $N-3$**four-cliques** (complete subgraphs of 4 nodes).
- These cliques reveal **tightly connected clusters** — e.g., stocks within the same economic sector.

96

Relation Between PMFG and MST

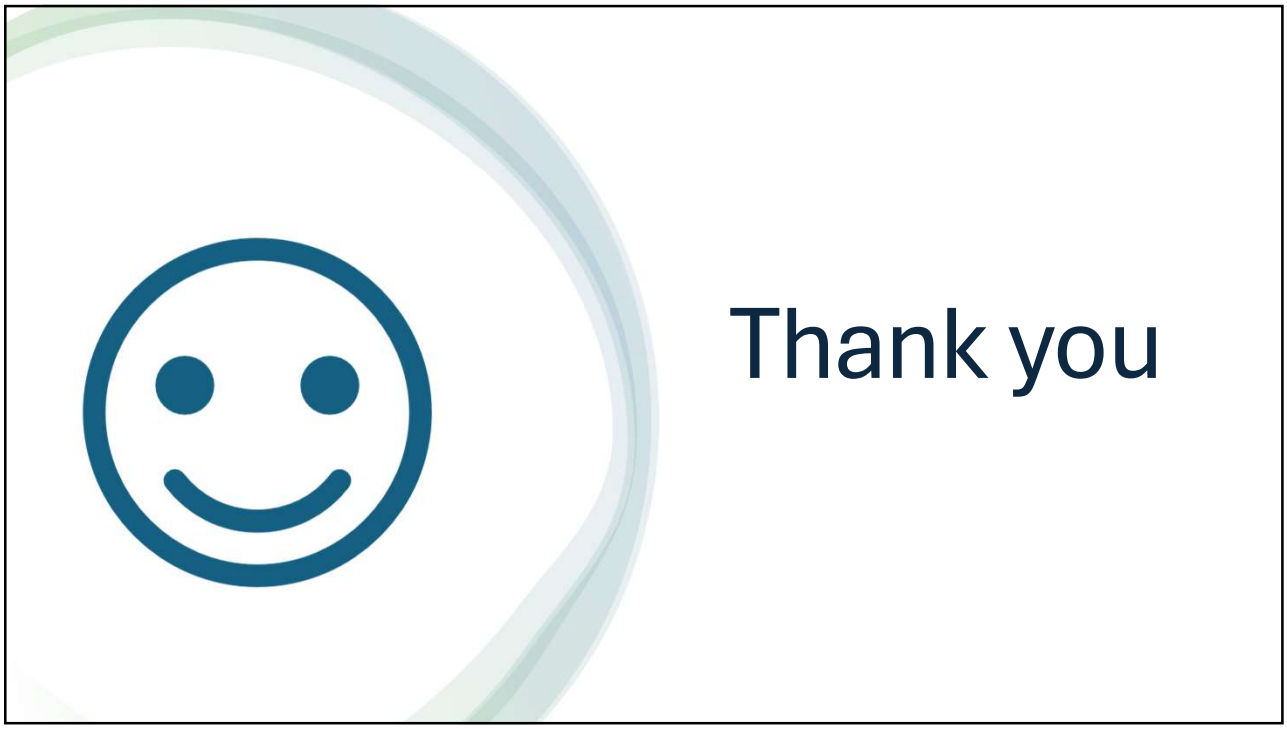
Feature	MST	PMFG
Edges	(N - 1)	(3(N - 2))
Loops	None	Allowed
Planarity	Yes	Yes
Hierarchical Information	Preserved	Preserved
Robustness to Noise	Low	Higher
Clustering Capability	Weak	Strong
Structural Richness	Sparse	Richer

97

Example

	A	B	C	D	E
A	1	**0.9**	**0.8**	0.4	0.3
B	0.9	1	**0.75**	0.5	0.4
C	0.8	0.75	1	**0.6**	0.55
D	0.4	0.5	0.6	1	**0.7**
E	0.3	0.4	0.55	0.7	1

98



99