Social-Network Graphs

Mining Social Networks

- Facebook, Google+, Twitter
- Email Networks, Collaboration Networks

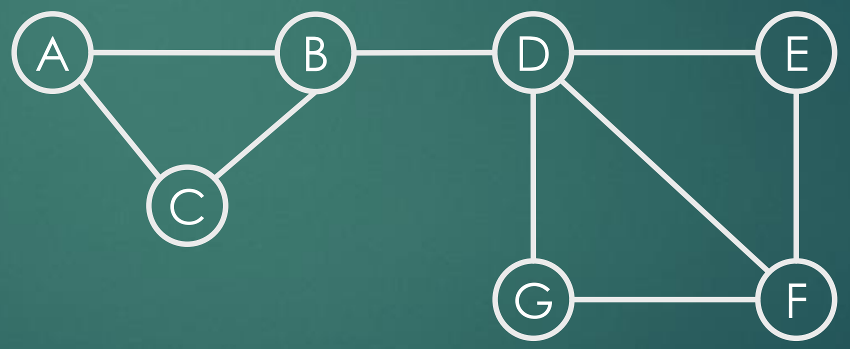
- Identify communities
  - Similar to clustering
  - Communities usually overlap
- Identify similarities amongst nodes of a graph
- Connectedness
- Transitive closure → reachability

What is a social network ?

- Collection of entities that participate in the network
  - People, telephones, email addresses
- At least one relationship between entities
  - Binary → facebook friends
  - Discrete → Google+ circles
  - Real → average time two phones talk to each other
  - Directed → Twitter
- Locality, non-randomness

Social Networks as Graphs

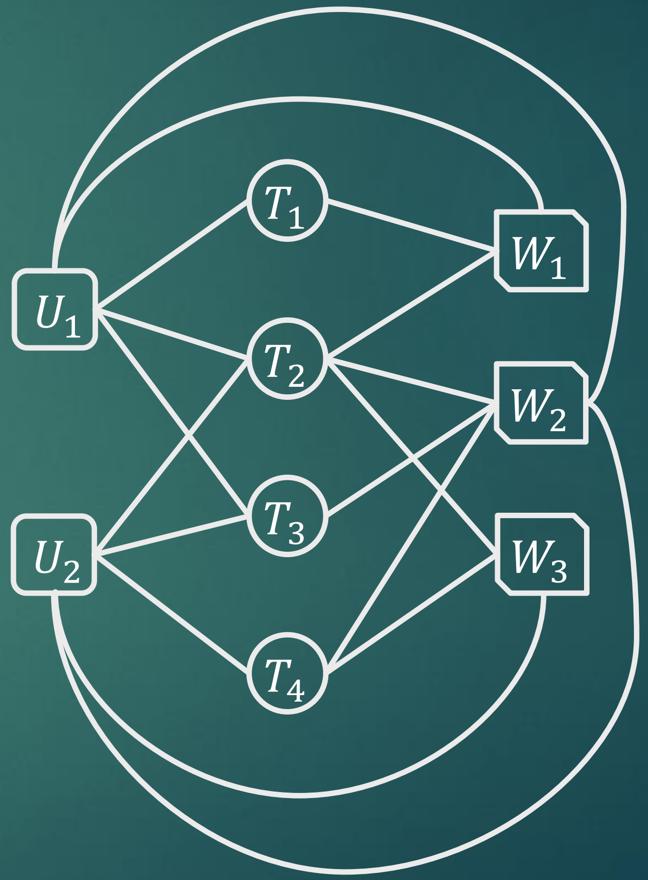
- Entities as nodes
- Edges represent the relationship
  - Weights
  - Directed



Collaborative filtering → pair of networks

Graphs with multiple node types

- Amazon → Users, products
- Research publications → Authors, Papers
- deli.cio.us → users, tags, webpages

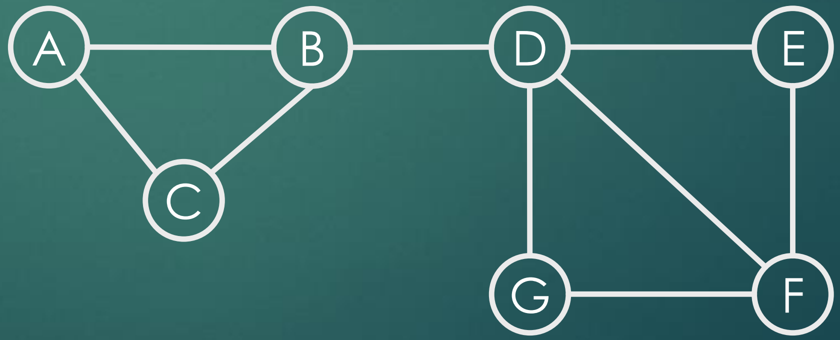


Form k-partite graphs

Form k-partite graphs

Clustering Social-Network Graphs

- Distance measures
- Consider a distance of 1 if edge exists, ∞ otherwise
- Violates triangle inequality
- Consider standard clustering
  - Agglomerative
  - Point-assignment

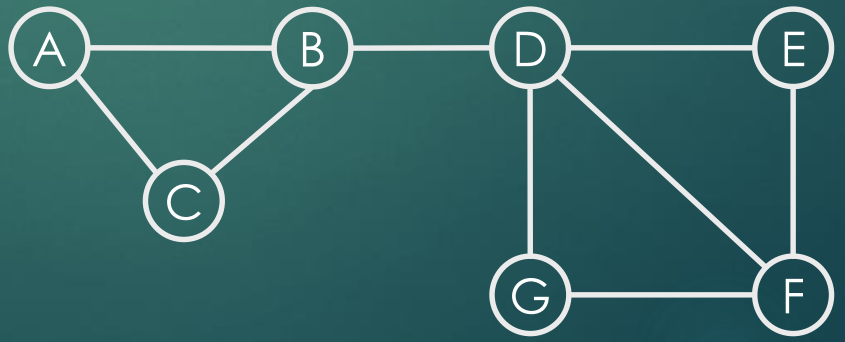


Betweenness

The betweenness of an $edge(a, b)$ is the number of pairs of nodes x and y such that the $edge(a, b)$ lies on the shortest path between x and y

A high *betweenness* suggests that the $edge(a, b)$ runs between two different groups, i.e., a and b do not belong to the same group.

Node-betweenness
Edge-betweenness



Project idea

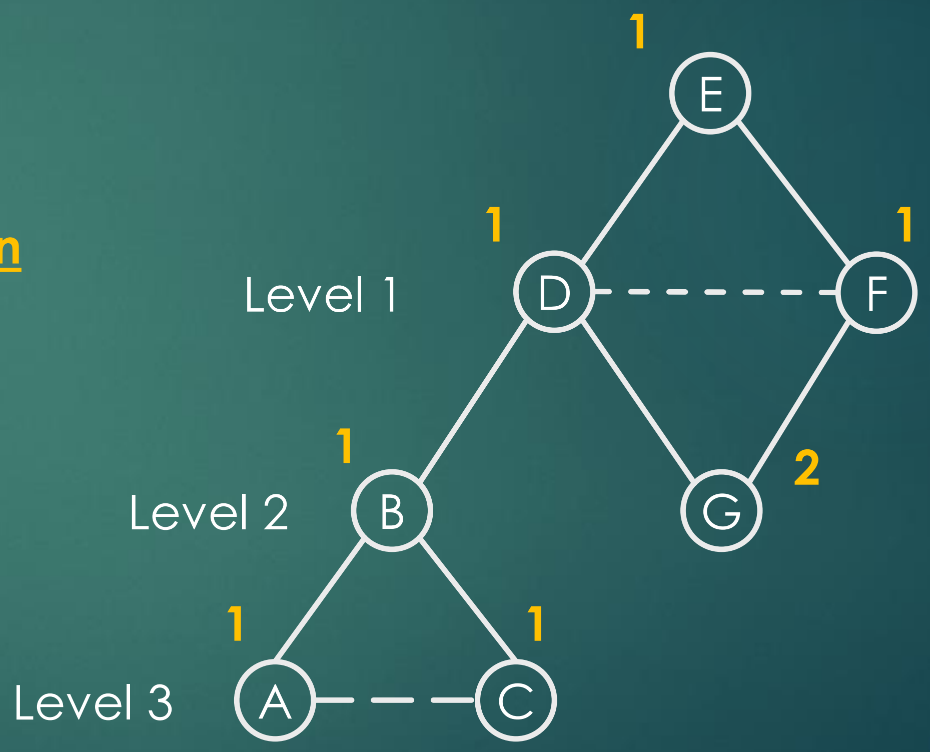
Girvan-Newman Algorithm

- Hierarchical detection of communities using edge-betweenness
- visits each node X once and computes the number of shortest paths from X to all other reachable nodes
  1. BFS starting at X
  2. label each node by the number of shortest paths that reach it from the root (X)
  3. calculate for each edge e the sum over all nodes Y of the fraction of shortest paths from the root X to Y that go through e

GN Algorithm

- Lets start with node E

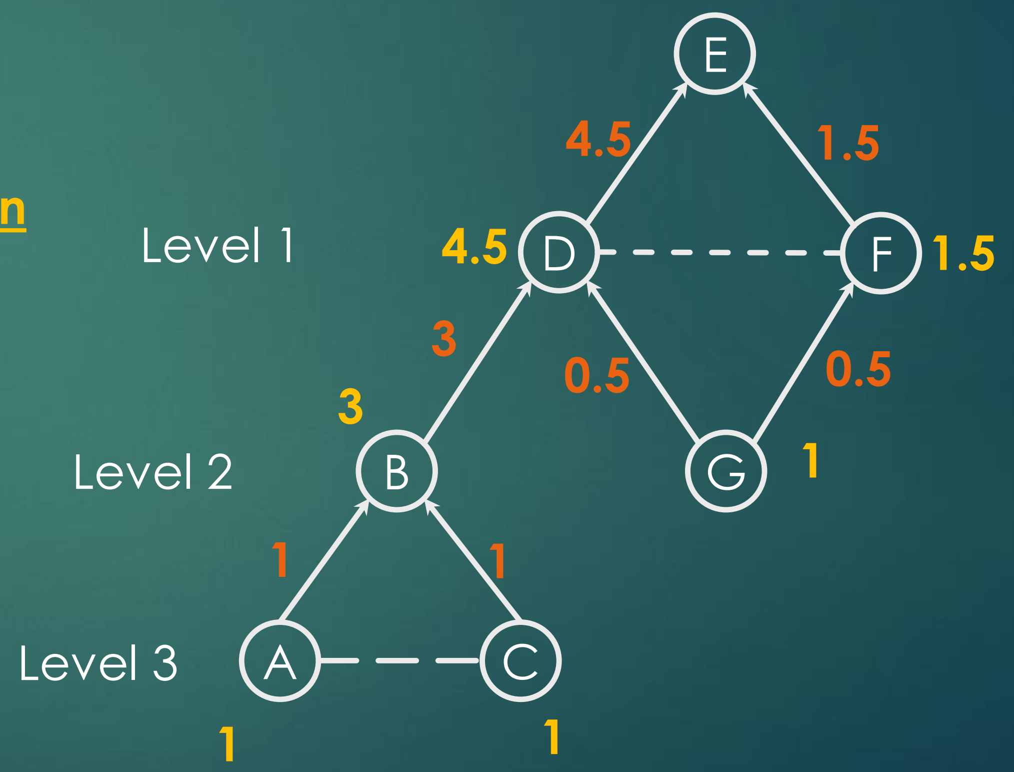
1. Build the BFS traversal of the graph
2. label each node by the number of shortest paths that reach it from the root E **top-down**
3. calculate for each edge e the sum over all nodes Y of the fraction of shortest paths from the root X to Y that go through e **bottom-up**



GN Algorithm

- Lets start with node E

1. Build the BFS traversal of the graph
2. label each node by the number of shortest paths that reach it from the root E **top-down**
3. calculate for each edge e the sum over all nodes Y of the fraction of shortest paths from the root X to Y that go through e **bottom-up**



GN Algorithm

- Repeat the calculation for every node as the root and sum the contributions (edge factors)
- Divide by 2 to get the true betweenness
  - Every shortest path will be discovered twice

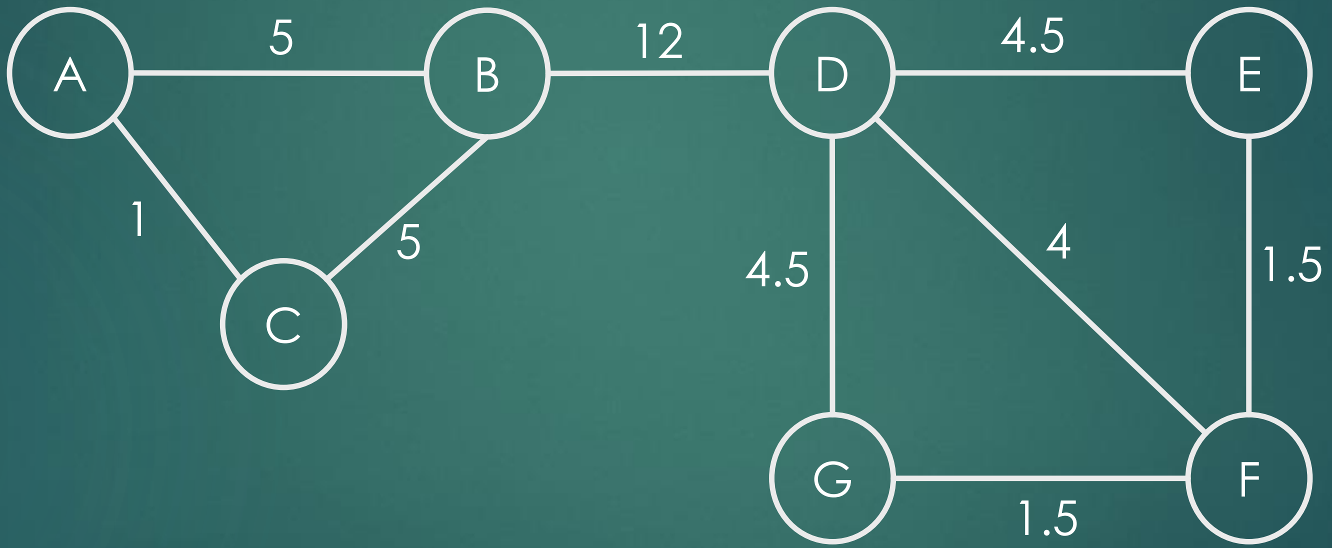
Using betweenness to find communities

- Behaves something like a distance measure
- Add edges in order of increasing betweenness

- Alternatively, think in terms of removing edges with highest betweenness

- As we progressively remove edges we are left with a larger number of smaller communities

Example



Girvan-Newman Algorithm

Complexity

- visits each node X once and computes the number of shortest paths from X to all other reachable nodes - n total nodes
  1. BFS starting at X $\mathcal{O}(e)$
  2. label each node by the number of shortest paths that reach it from the root (X) $\mathcal{O}(e)$
  3. calculate for each edge e the sum over all nodes Y of the fraction of shortest paths from the root X to Y that go through e $\mathcal{O}(e)$

$$\mathcal{O}(ne)$$

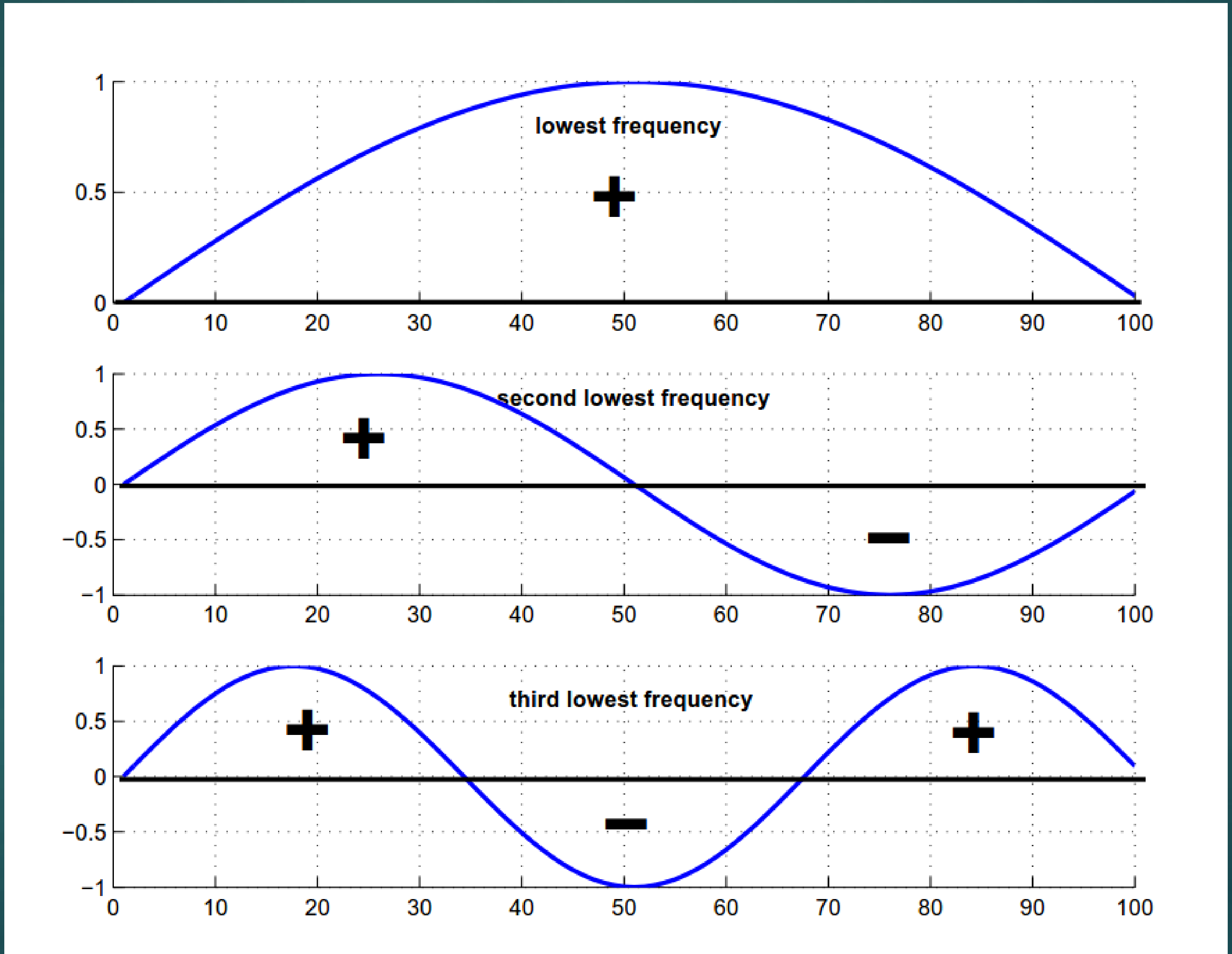
Graph Partitioning

- Using spectral methods
- Problem of finding the minimizing cut
  - Minimize the number of edges removed (potentially weighted)
  - constrain the selection of the cut so that the two sets are approximately equal in size

- Normalized cuts
  - Define the volume of a set of nodes S, denoted $V(S)$, to be the number of edges with at least one end in S
  - Suppose we partition the graph into two disjoint sets S and T
  - Let $C(S,T)$ be the number of edges that connect a node in S to a node in T
  - The *normalized cut* value for S and T is

$$\frac{C(S,T)}{V(S)} + \frac{C(S,T)}{V(T)}$$

Motivation



lowest frequency
+
second lowest frequency
+
−
third lowest frequency
+
−
+

Graph Laplacian

- Adjacency matrix - A
- Degree Matrix - D → diagonal
- Laplacian matrix - $L = D - A$
- L is a $n \times n$ symmetric positive semi-definite
  - This means the eigenvalues of L are real and its eigenvectors are real and orthogonal
- The eigenvalues of L are nonnegative

$$0 \leq \lambda_1 \leq \lambda_2 \leq \ \ldots \leq \lambda_n$$

Graph Laplacian - eigenvalues

$\lambda_1(L(\mathcal{G})) = 0$ and the corresponding eigenvector $w_1 = [1,1,\ldots,1]$

$$L\mathbf{1} = 0\mathbf{1}$$

The number of connected components of $\mathcal{G}$ is equal to the number of λ_i that are equal to 0

Definition: $\lambda_2(L(\mathcal{G}))$ is the algebraic connectivity of $\mathcal{G}$

- The magnitude of λ_2 measures connectivity
- In particular, $\lambda_2 \neq 0$ if and only if $\mathcal{G}$ is connected

Spectral Bisection

Compute eigenvector w_2 corresponding to λ_2

if $w_2(n) < 0$

assign n to first half

else

assign n to second half

Computing eigenvectors

- Iterative method to compute eigenvectors
  - Inverse power method, Lanczos
  - Requires matrix vector multiplication
  - Given x, compute Lx

- Requires partitioning!

Computing λ_2, w_2

- The second-smallest eigenvalue of L is the minimum of $\boldsymbol{x}^T L \boldsymbol{x}$, s.t.
  - $\sum_{i=1}^{n} x_i^2 = 1$
  - $\boldsymbol{x}$ is orthogonal to $\mathbf{1}$

- $\boldsymbol{x^T L x} = \boldsymbol{x^T D x} - \boldsymbol{x^T A x}$
- $\boldsymbol{x^T D x} = \sum d_i x_i^2$
- $\boldsymbol{x^T A x} = -2 \sum x_i x_j$
- $\boldsymbol{x^T L x} = \sum_{e_{ij} \in E} \left(x_i - x_j\right)^2$

Constrained minimization problem

Computing λ_2, w_2 using Lanczos

- Given any $n \times n$ symmetric matrix A (such as $L(\mathcal{G})$), Lanczos computes a tridiagonal $k \times k$ approximation T by doing k matrix-vector products, $k \ll n$

- Calculate the eigenvalues/eigenvectors of the much smaller and simpler matrix T → tridiagonal QR iteration

Direct discovery of communities

- Although partitioning the graph using betweenness is effective, it has some drawbacks
  - Not possible to place an individual in two different communities
  - Everyone is assigned a community
- Alternatively, discover communities by looking for subsets of the nodes that have a relatively large number of edges among them
  - Finding cliques → NP Complete
  - Easier to find complete bipartite subgraphs
  - Counting tringles