Community Detection in Networks

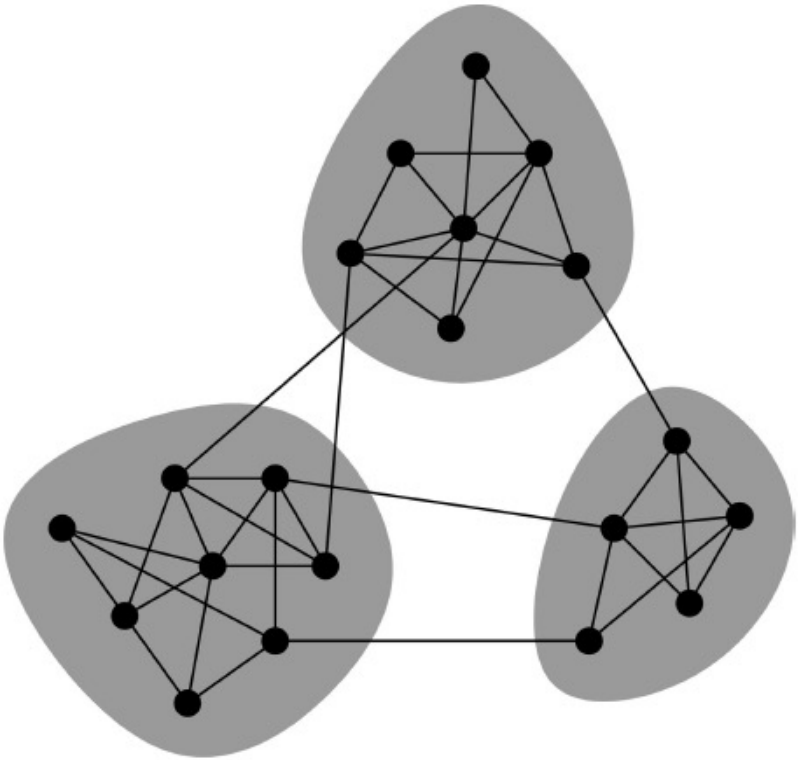
Agenda

Today's Topics:

- Understanding communities
- Measuring community structure
- Louvain modularity
- NOCD

Networks & Communities

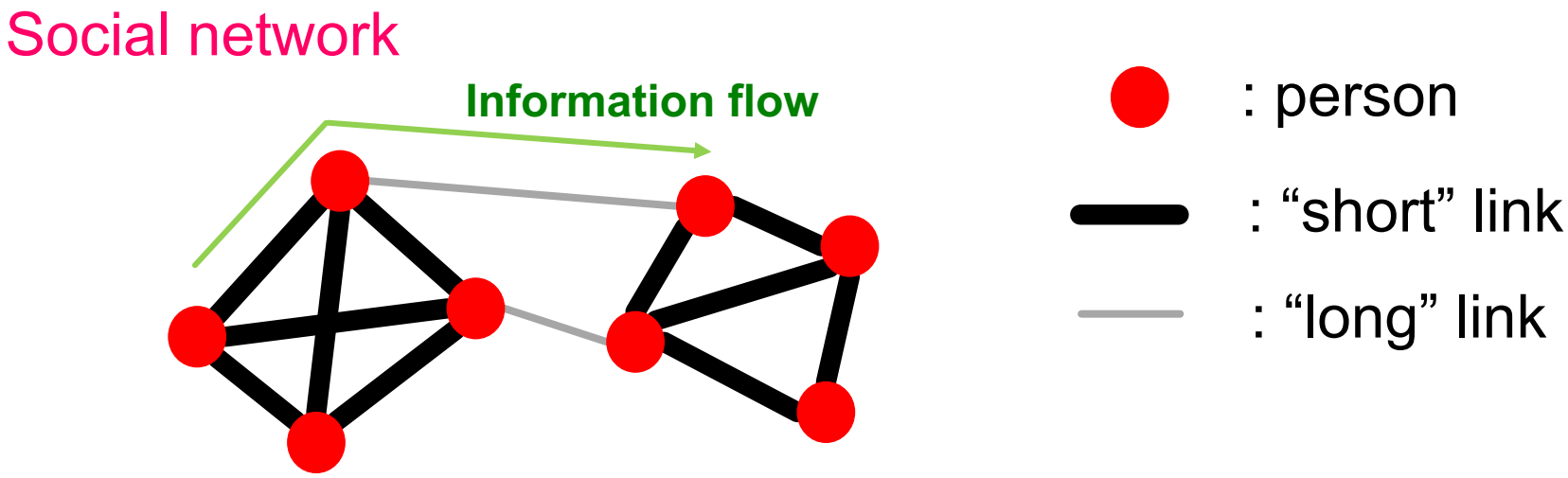
- We often think of networks "looking" like this:



- What led to such a conceptual picture?

Networks: Flow of Information

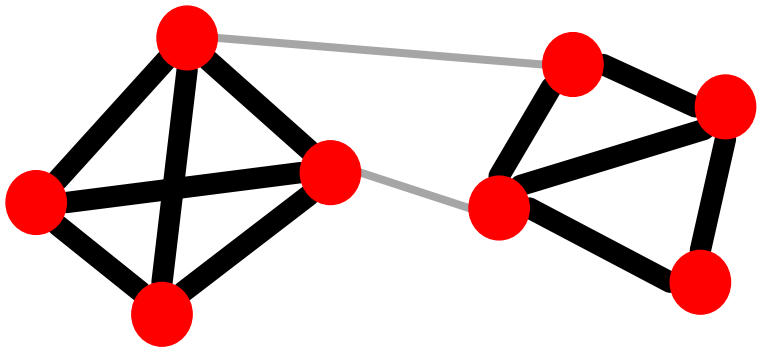
- **How does information flow through the network?**
  - People are "embedded" in a social network.
  - There are different links ("short" vs. "long") in the network, through which information flows.



Flow of Job Information

- **How do people find out about new jobs?**
  - Mark Granovetter, part of his PhD in 1960s
  - People find the information **through personal contacts**
- **But:** Contacts were often **acquaintances** rather than close friends
  - **This is surprising:** One would expect your friends to help you out more than casual acquaintances
- **Why is it that acquaintances are most helpful?**

Personal contact network



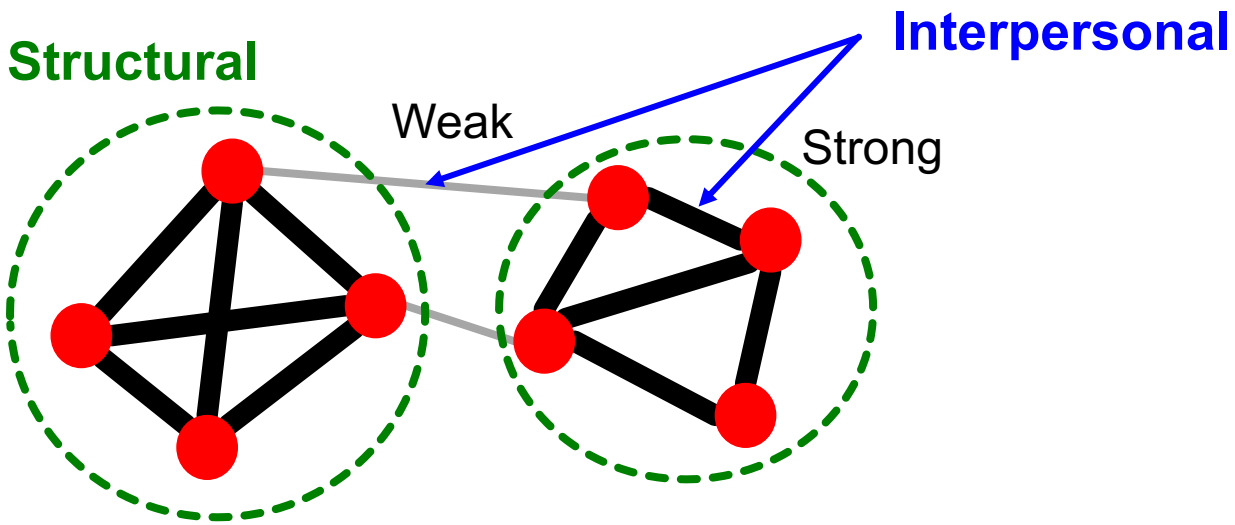
: "close-friend" link

: "acquaintance" link

[Granovetter '73]

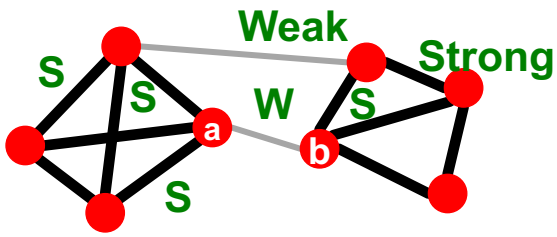
Granovetter's Answer

- **Two perspectives on friendships:**
  - **Structural:** Friendships span different parts of the network
  - **Interpersonal:** Friendship between two people is either **strong** or **weak**



Granovetter's Explanation

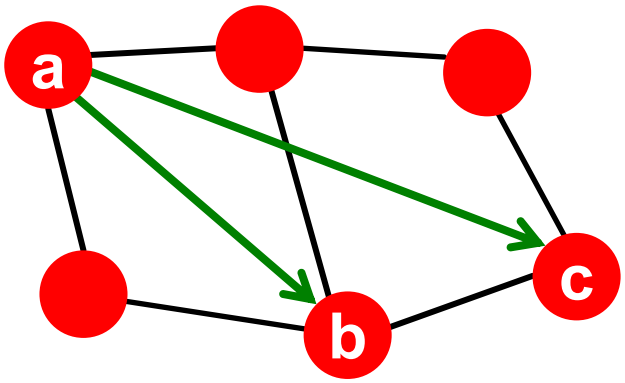
- **Granovetter makes a connection between the social and structural role of an edge**
- **First point: Structure**
  - Structurally embedded (tightly-connected) edges are also socially **strong**
  - Long-range edges spanning different parts of the network are socially **weak**
- **Second point: Information**
  - Long-range edges allow you to gather information from different parts of the network and get a job
  - Structurally embedded edges are heavily redundant in terms of information access



Triadic Closure

- How community (tightly-connected cluster of nodes) forms?

Which edge is more likely, a-b or a-c?



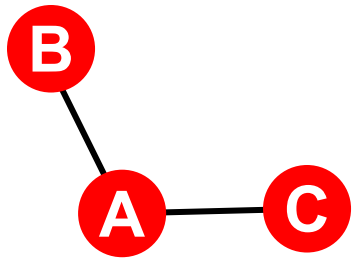
If two people in a network have a friend in common, then there is an increased likelihood they will become friends themselves.

Reasons for Triadic Closure

- **Triadic closure = High clustering coefficient**

Reasons for triadic closure:

- If B and C have a friend A in common, then:
  - B is **more likely to meet** C
    - (since they both spend time with A)
  - B and C **trust** each other
    - (since they have a friend in common)
  - A has **incentive** to bring B and C together
    - (since it is hard for A to maintain two disjoint relationships)



- **Empirical study by Bearman and Moody:**
  - Teenage girls with low clustering coefficient are more likely to contemplate suicide

Edge Strength in Real Data

- **For many years Granovetter's theory was not tested**
- But today we have large who-talks-to-whom graphs:
  - Email, Messenger, Cell phones, Facebook
- **Onnela et al. 2007:**
  - Cell-phone network of 20% of EU country's population
  - **Edge weight:** # phone calls

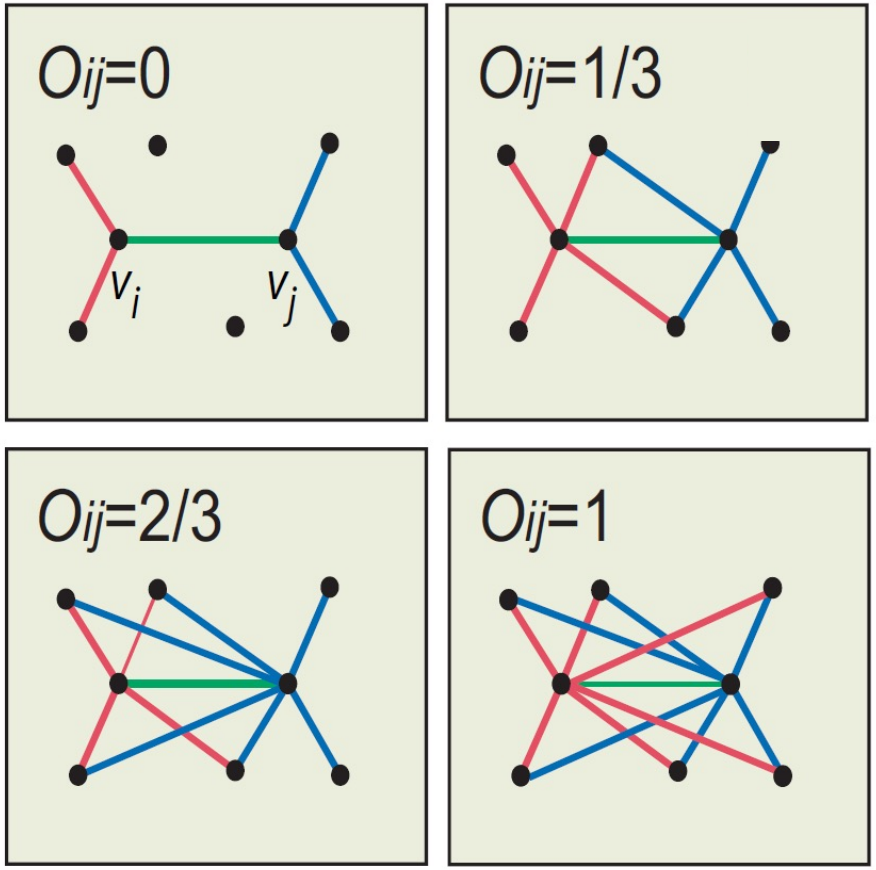
Edge Overlap

- **Edge overlap:**

$$O_{ij} = \frac{|(N(i) \cap N(j)) - \{i, j\}|}{|(N(i) \cup N(j)) - \{i, j\}|}$$

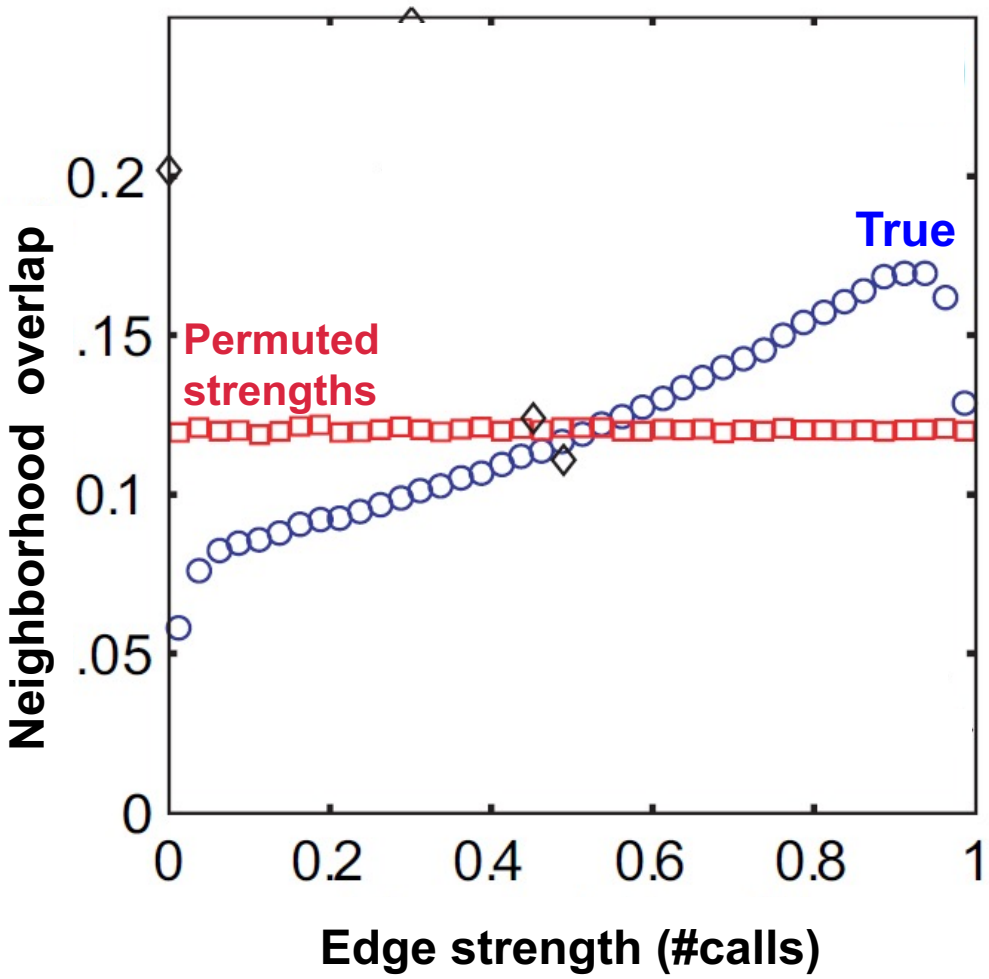
  - $N(i)$ … the set of neighbors of node i

- **Note: Overlap $= 0$** when an edge is a **"local bridge"**

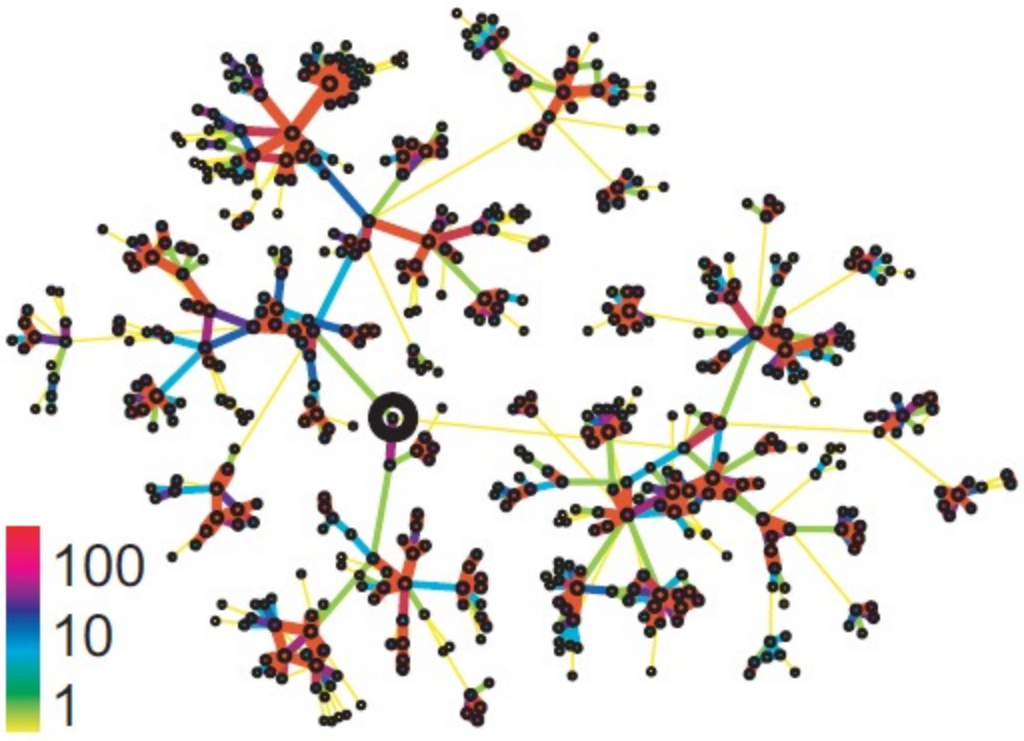


Phones: Edge Overlap vs. Strength

- **Cell-phone network**
- **Observation:**
  - **Highly used links have high overlap!**
- **Legend:**
  - **True:** The data
  - **Permuted strengths:** Keep the network structure but randomly reassign edge strengths

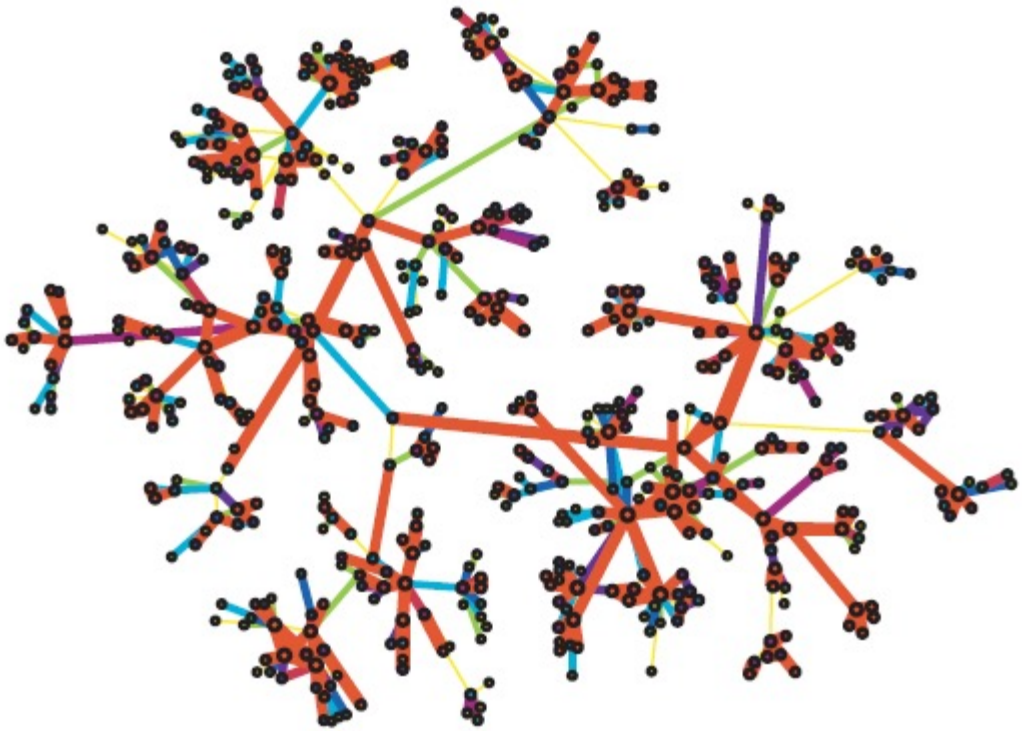


Real Network, Real Edge Strengths



- **Real edge strengths in mobile call graph**
  - Strong ties are more embedded (have higher overlap)

Real Net, Permuted Tie Strengths

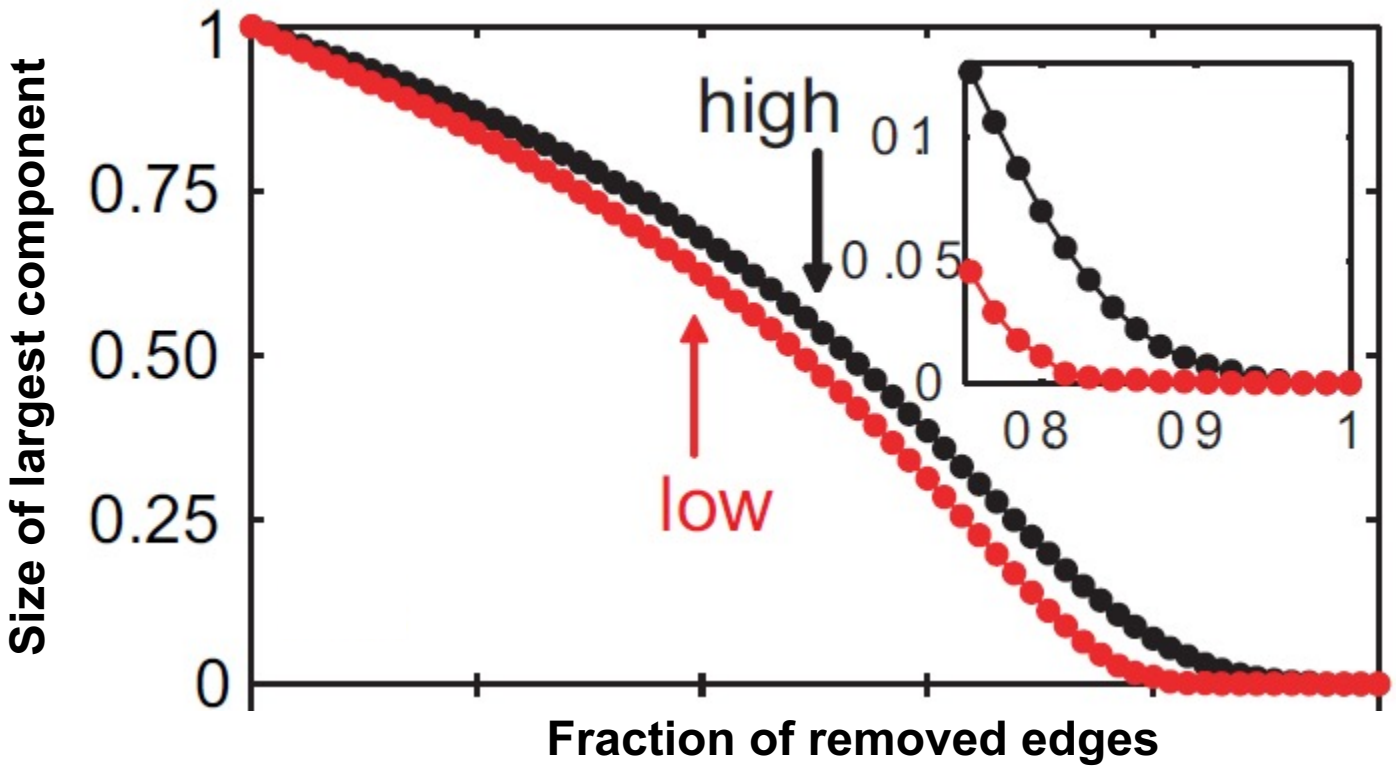


- **Same network, same set of edge strengths** but now **strengths are randomly shuffled**

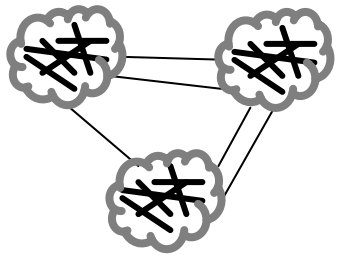
Edge Removal by Strength

Removing edges based on **strength (#calls)**

- Low to high
- High to low



Low disconnects the network sooner

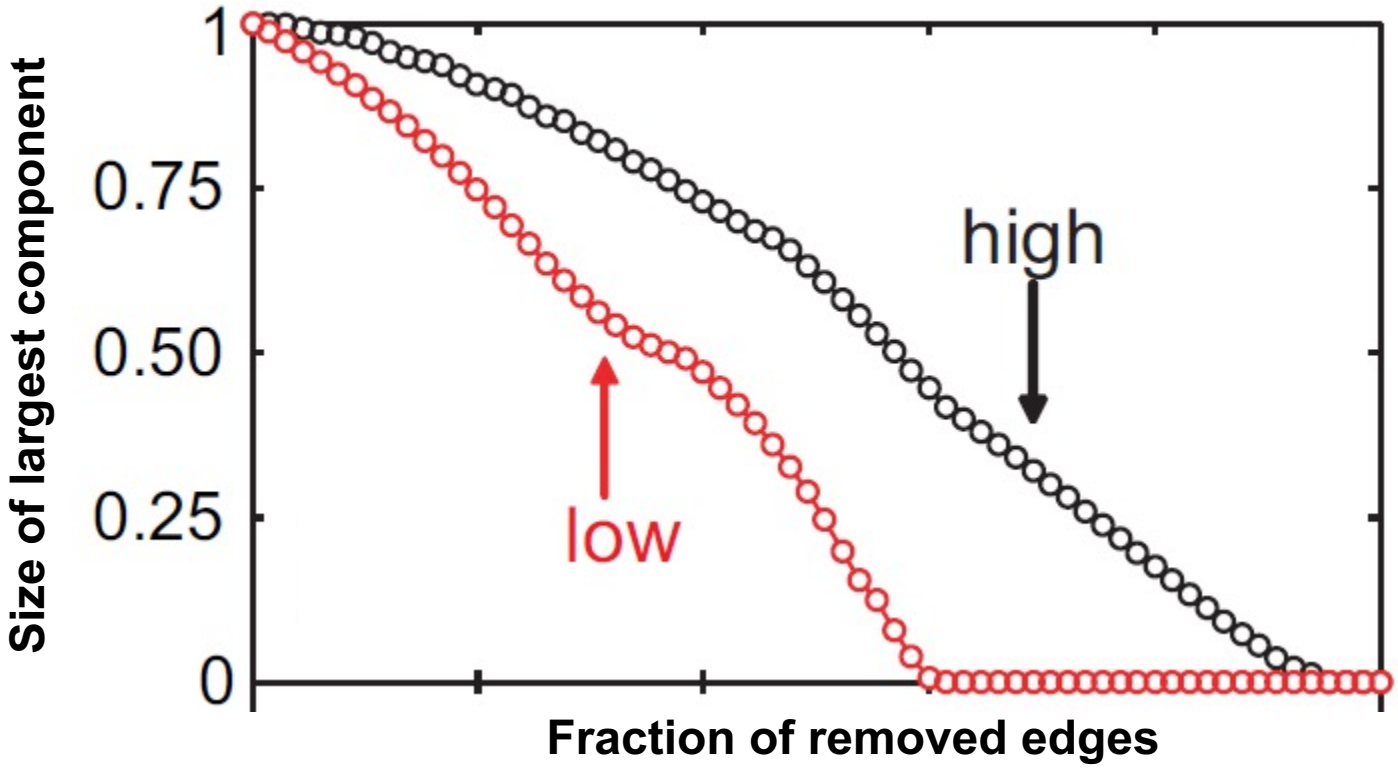


Conceptual picture of network structure

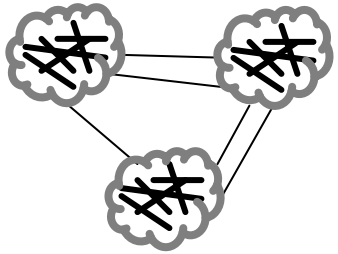
Edge Removal by Overlap

Removing edges based on **edge overlap**

- Low to high
- High to low



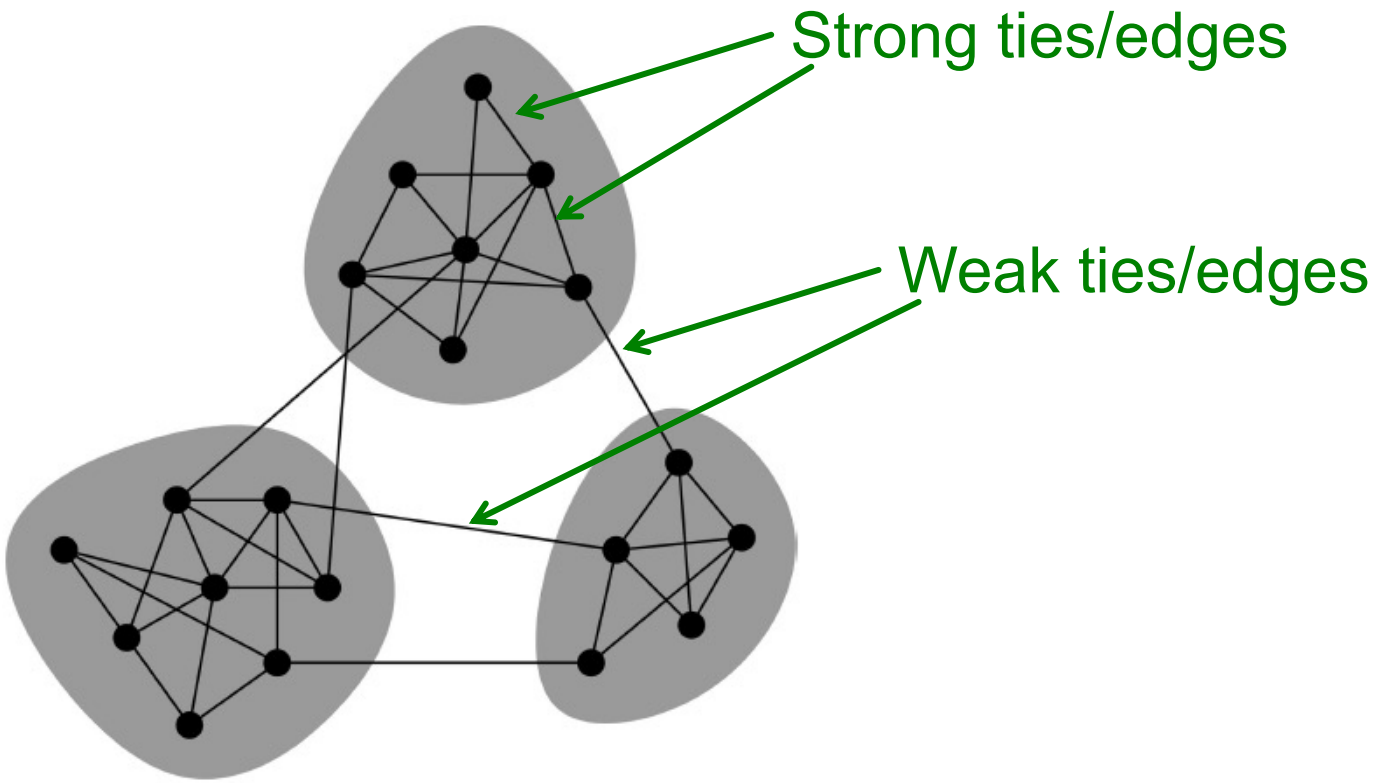
Low disconnects the network sooner



Conceptual picture of network structure

Conceptual Picture of Networks

- **Granovetter's theory leads to the following conceptual picture of networks**



Discussion

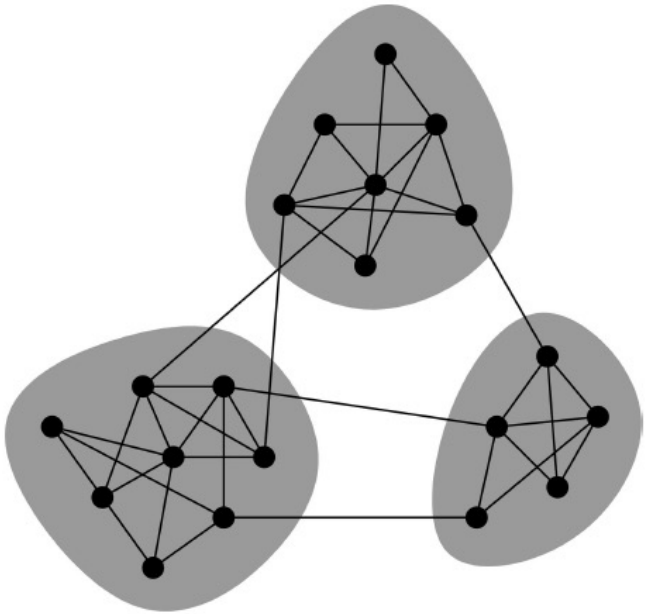
Questions?

Stanford CS224W: Network Communities

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Network Communities

- Granovetter's theory suggests that networks are composed of **tightly connected sets of nodes**

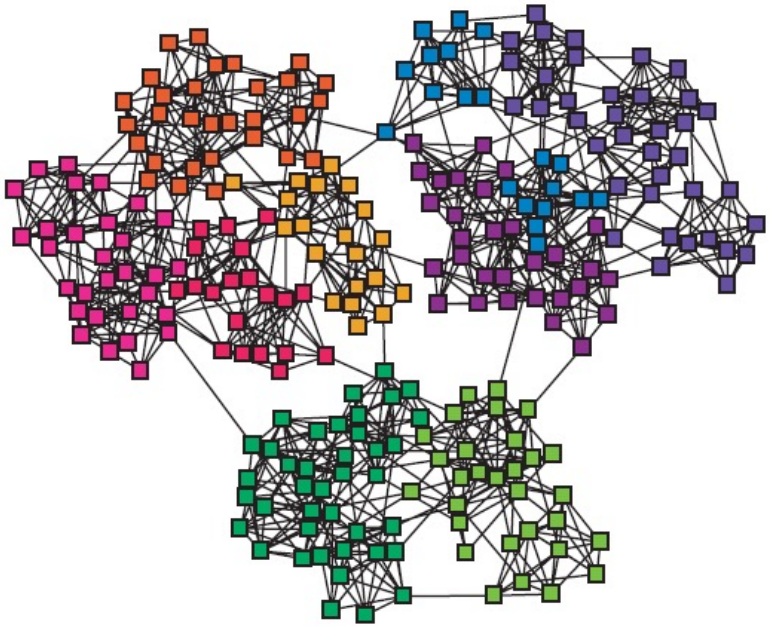


- **Network communities:**
  - Sets of nodes with **lots** of **internal** connections and **few external** ones (to the rest of the network).

Communities, clusters, groups, modules

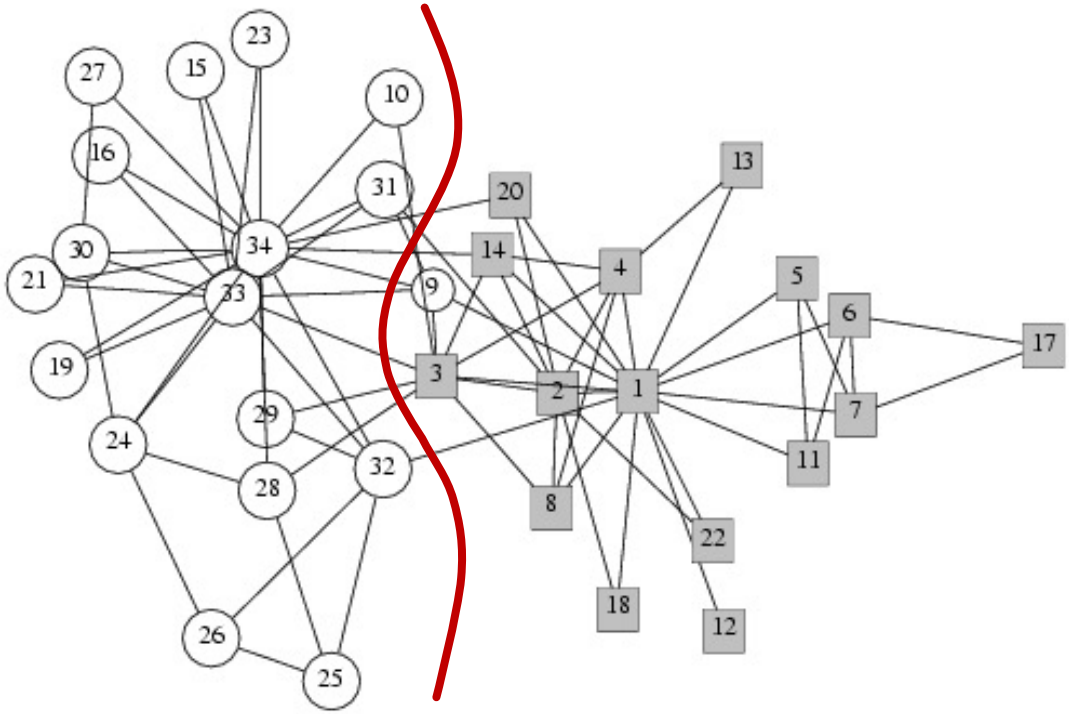
Finding Network Communities

- **How do we automatically find such densely connected groups of nodes?**
- Ideally such automatically detected clusters would then correspond to real groups
- **For example:**



Communities, clusters, groups, modules

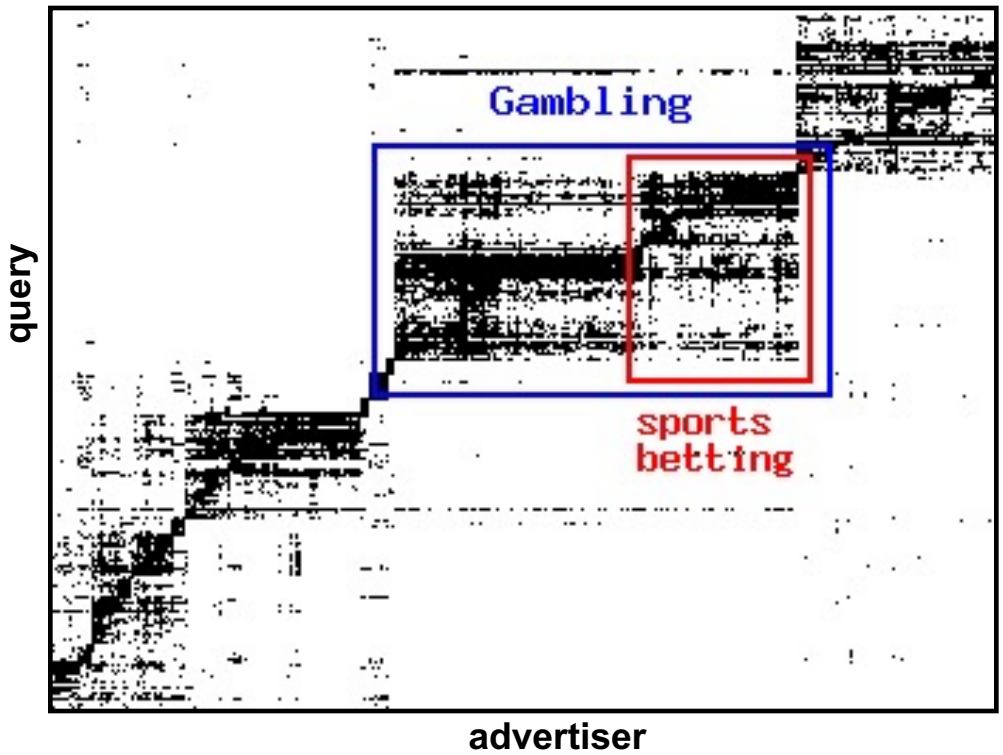
Social Network Data



- **Zachary's Karate club network:**
  - Observed social ties & rivalries in a university karate club
  - During the study, conflicts led the group to split
  - Split could be explained by a minimum cut in the network

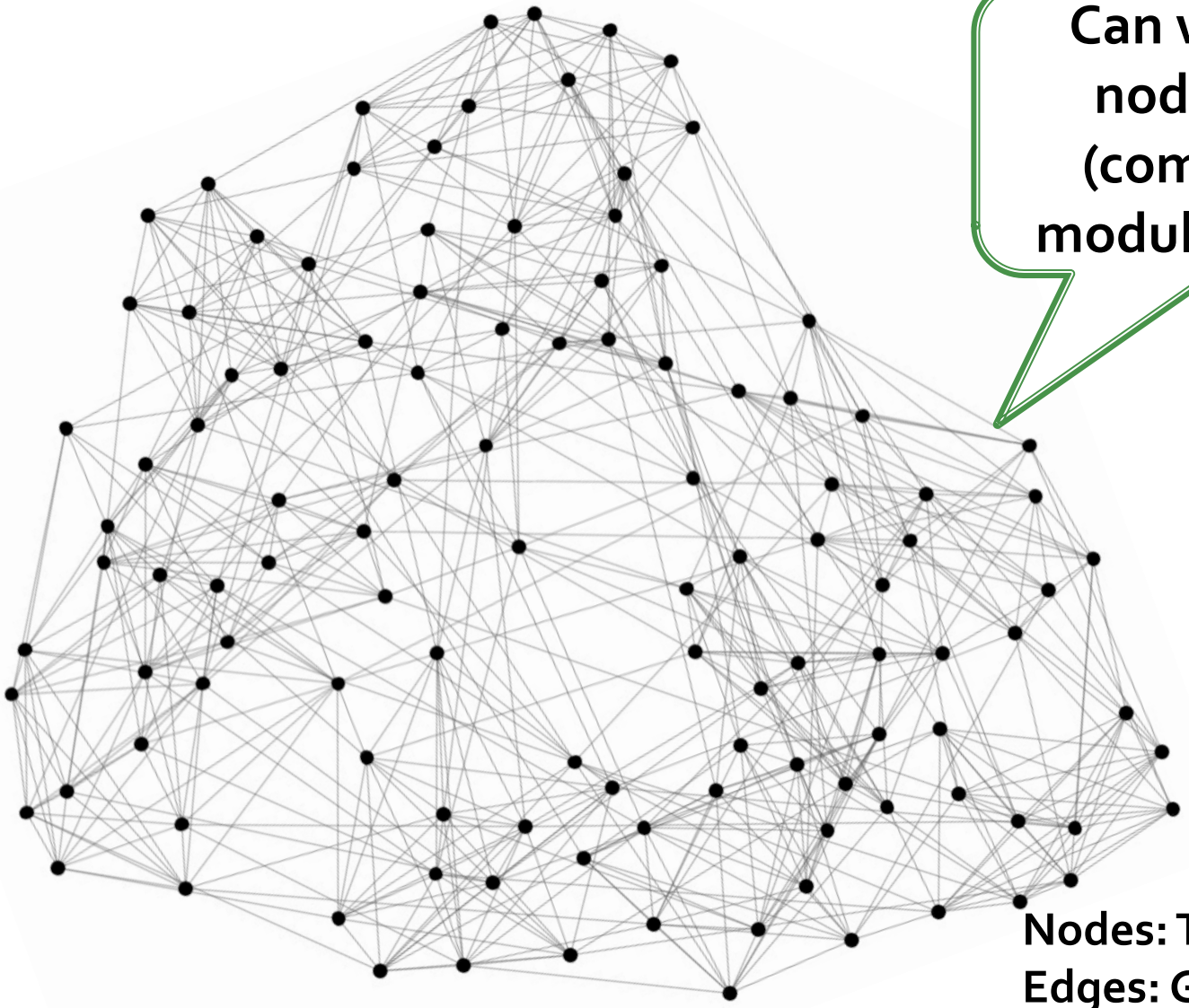
Micro-Markets in Sponsored Search

Find micro-markets by partitioning the "query-to-advertiser" graph in web search:



Nodes: Advertisers and queries/keywords; Edges: Advertiser advertising on a keyword.

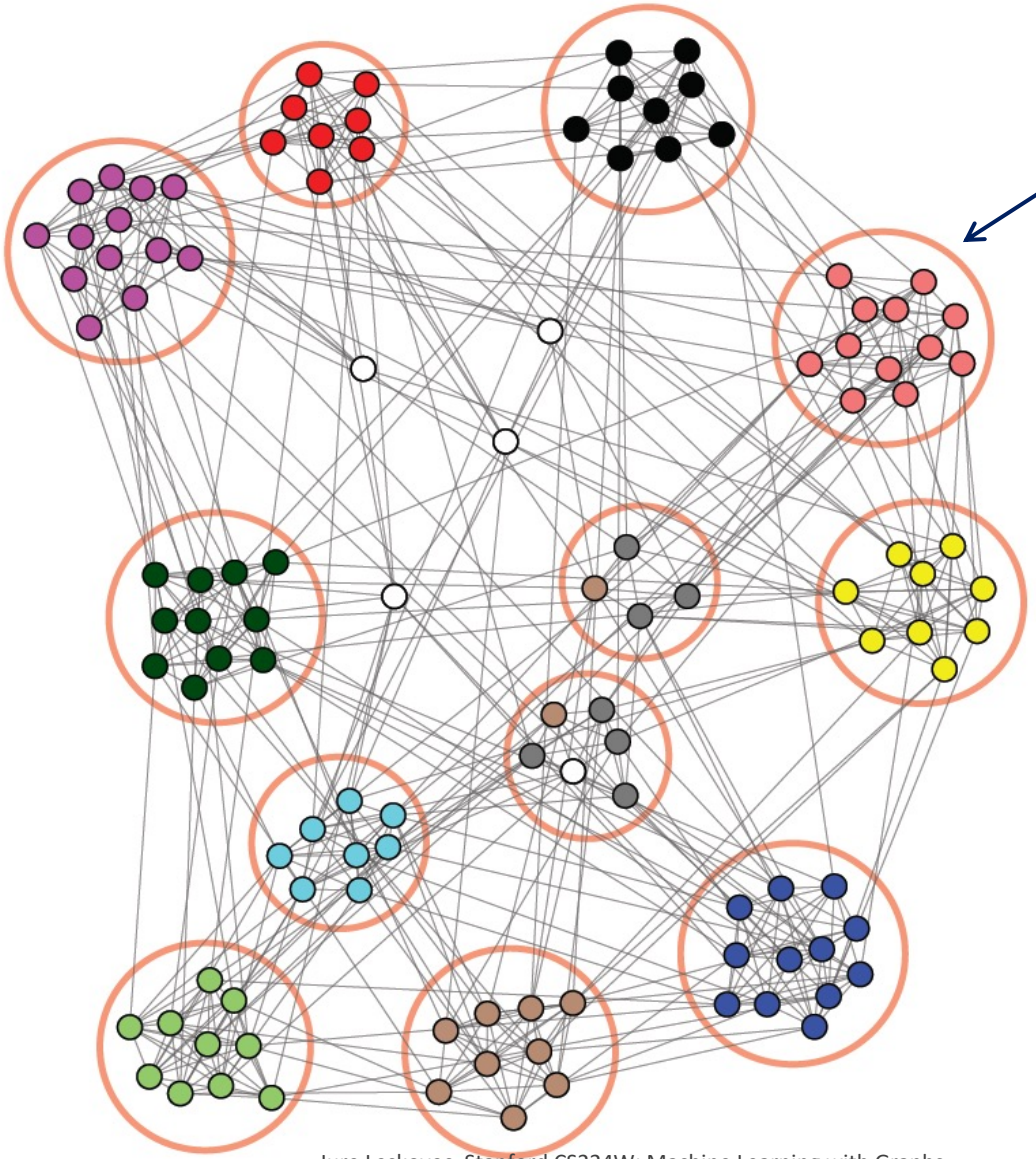
NCAA Football Network



Can we identify node groups? (communities, modules, clusters)

Nodes: Teams
Edges: Games played

NCAA Football Network



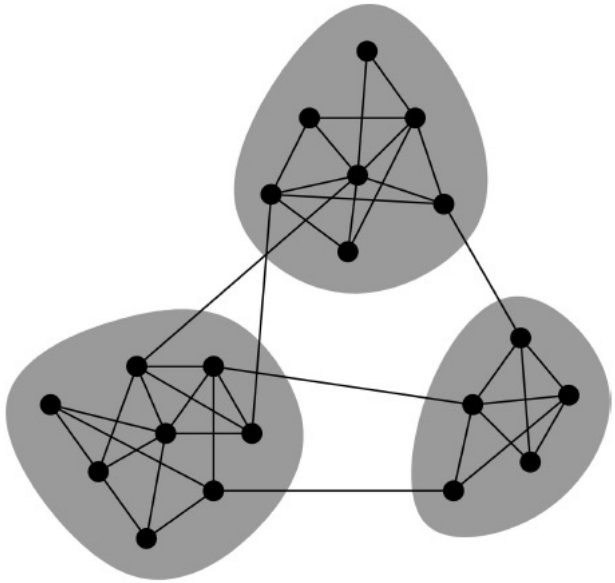
NCAA conferences

- Mid American
- Big East
- Atlantic Coast
- SEC
- Conference USA
- Big 12
- Western Athletic
- Pacific 10
- Mountain West
- Big 10
- Sun Belt
- Independents

Nodes: Teams
Edges: Games played

Network Communities

- **Communities:** Sets of **tightly connected nodes**
- Define: **Modularity Q**
  - A measure of how well a network is partitioned into communities
  - Given a **partitioning** of the network into groups disjoint $s \in S$:

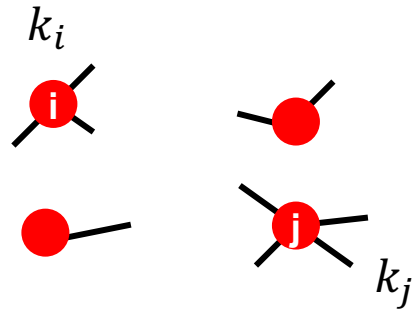


$$Q \propto \sum_{s \in S} [\ (\#\ \text{edges within group } s) - (\text{expected } \#\ \text{edges within group } s)\ ]$$

Need a null model

Null Model: Configuration Model

- **Given real G on n nodes and m edges, construct rewired network G'**
  - **Same degree distribution** but uniformly random connections



  - Consider G' as a multigraph (multiple edges exist between nodes)
  - **The expected number of edges between nodes i and j** of degrees k_i and k_j equals: $k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}$
    - There are $2m$ **directed** edges (counting i->j and j->i) in total.
    - For each of k_i out-going edges from node i, the chance of it landing to node j is $k_j/2m$, hence $k_i k_j/2m$.

Null Model: Configuration Model

- **The expected number of edges between nodes i and j** of degrees k_i and k_j equals: $k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}$
  - The expected number of edges in (multigraph) **G'**:
    - $= \frac{1}{2}\sum_{i \in N}\sum_{j \in N}\frac{k_i k_j}{2m} = \frac{1}{2}\cdot\frac{1}{2m}\sum_{i \in N} k_i\left(\sum_{j \in N} k_j\right) =$
    - $= \frac{1}{4m} 2m \cdot 2m = m$

Note: $\sum_{u \in N} k_u = 2m$

- **Under null model, both the degree distribution and the total number of edges are preserved.**

Notice: This model applies to both weighted and unweighted networks. For weighted networks we use the weighted degree (sum of the edge weights).

Modularity

- **Modularity of partitioning S of graph G:**
  - **Q $\propto \sum_{s \in S}$ [(# edges within group s) – (expected # edges within group s)]**
  - $$Q(G,S) = \underbrace{\frac{1}{2m}} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left(A_{ij} - \frac{k_i k_j}{2m} \right)$$

    Normalizing const.: $-1 \leq Q \leq 1$

    A_{ij} = 1 if i→j, 0 otherwise (if G is weighted then A_{ij} is the edge weight)

- **Modularity values take range [–1,1]**
  - It is positive if the number of edges within groups exceeds the expected number
  - **Q** greater than **0.3-0.7** means **significant community structure**
  - **Notice Modularity applies to weighted and unweighted networks.**

RECAP: Modularity

For each group s

$$Q(G,S) = \frac{1}{2m}\sum_{s\in S}\sum_{i\in s}\sum_{j\in s}\left(A_{ij} - \frac{k_i k_j}{2m}\right)$$

Equivalently modularity can be written as:

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

- A_{ij} represents the edge weight between nodes i and j;
- k_i and k_j are the sum of the weights of the edges attached to nodes i and j, respectively;
- $2m$ is the sum of all of the edge weights in the graph;
- c_i and c_j are the communities of the nodes; and
- δ is an indicator function $\delta(c_i, c_j) = 1 \; if \; c_i = c_j \; else \; 0$

Idea: We can identify communities by maximizing modularity

Discussion

Questions?

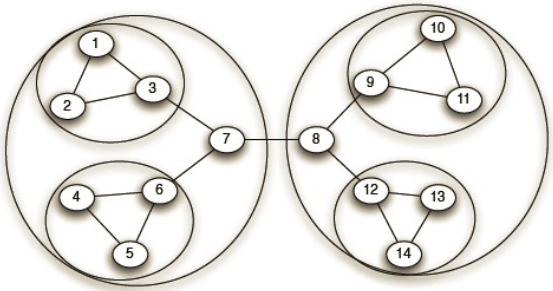
Stanford CS224W: Louvain Algorithm

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

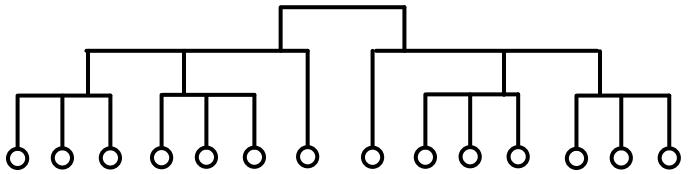
Louvain Algorithm

- **Greedy algorithm** for community detection
  - $O(n \log n)$ run time
- Supports weighted graphs
- Provides hierarchical communities
- Widely utilized to **study large networks** because:
  - Fast
  - Rapid convergence
  - High modularity output (i.e., "better communities")

Network and communities:



Dendrogram:

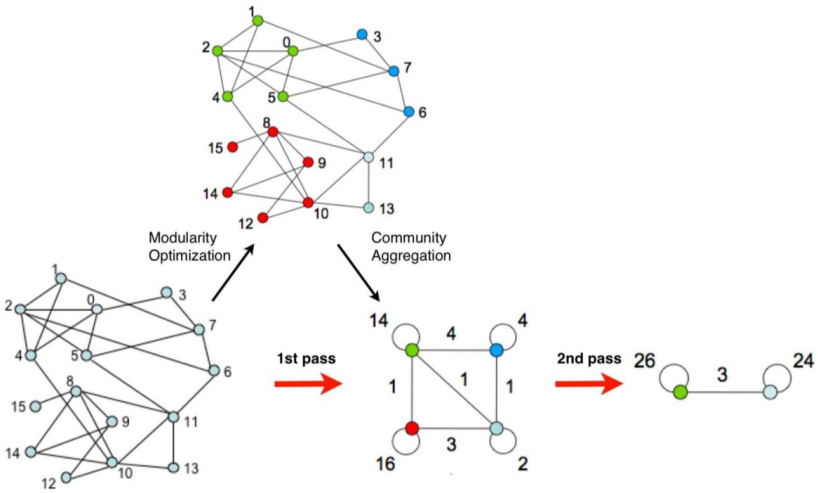


"Fast unfolding of communities in large networks" Blondel et al. (2008)

Louvain Algorithm: At High Level

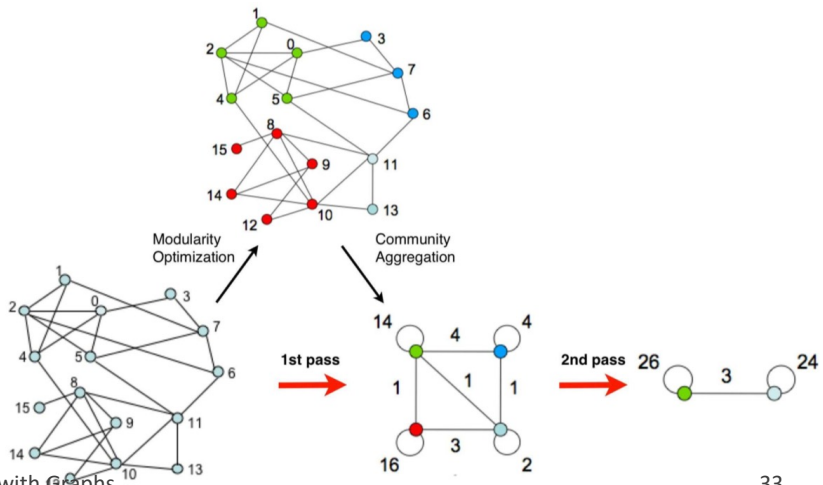
- Louvain algorithm **greedily maximizes** modularity
- **Each pass is made of 2 phases:**
  - **Phase 1:** Modularity is optimized by allowing only local changes to node-communities memberships
  - **Phase 2:** The identified communities are aggregated into super-nodes to build a new network
  - **Go to Phase 1**

The passes are repeated **iteratively** until no increase of modularity is possible.



Louvain Algorithm: At High Level

- Louvain algorithm considers graphs as **weighted**
  - The original graph can be unweighted (i.e., edge weights are all 1)
  - As the communities get identified and aggregated into super-nodes, weighted graphs are created (weights count the number of edges in the original graph)
  - Weighted version of Modularity is applied



Louvain: 1st phase (Partitioning)

- Put each node in a graph into a distinct community (one node per community)
- For each node i, the algorithm performs two calculations:
  - Compute the modularity delta (ΔQ) when putting node i into the community of some neighbor j
  - Move i to a community of node j that yields the largest gain in ΔQ
- **Phase 1 runs until no movement yields a gain**

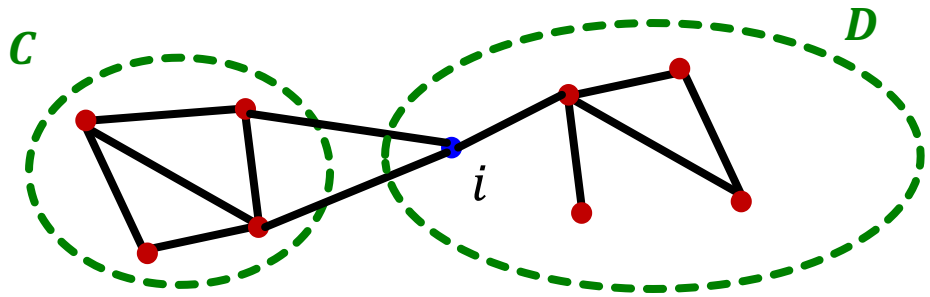
This first phase stops when a local maxima of the modularity is attained, i.e., when no individual node move can improve the modularity.
Note that the output of the algorithm depends on the order in which the nodes are considered.
Research indicates that the ordering of the nodes does not have a significant influence on the overall modularity that is obtained.

Louvain: Modularity Gain

What is ΔQ if node i moves from community D to C?

$$\Delta Q(D \to i \to C) = \Delta Q(D \to i) + \Delta Q(i \to C)$$

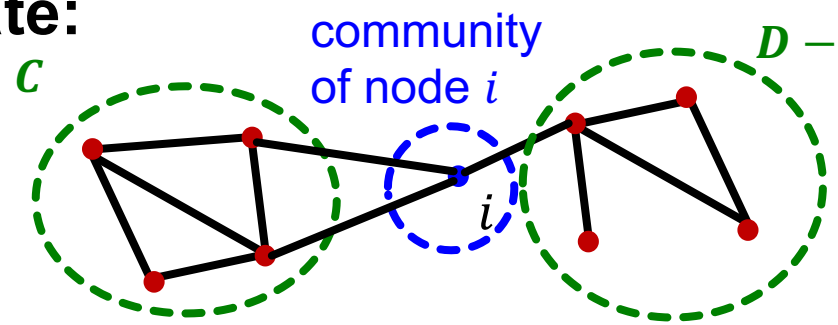
Before:



Removing i from D

$$\Delta Q(D \to i)$$

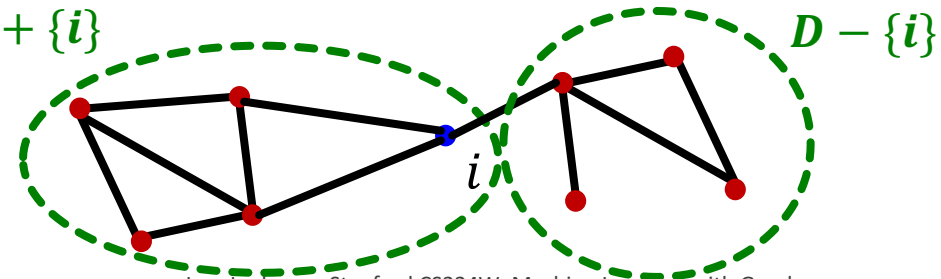
Intermediate:



Merging i into C

$$\Delta Q(i \to C)$$

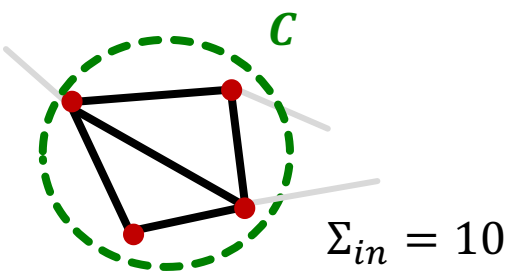
After:



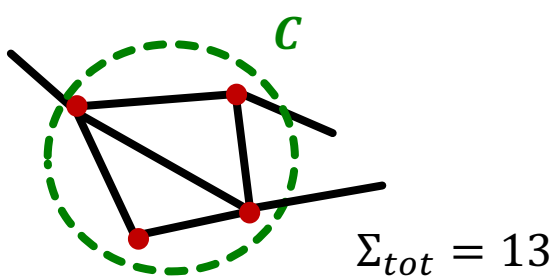
Deriving $\Delta Q(i \to C)$

- **Let's derive $\boldsymbol{\Delta Q(i \to C)}$**
- First, we derive modularity **within $\boldsymbol{C}$**, i.e., $Q(C)$.
- **Define:**
  - $\boldsymbol{\Sigma_{in}} \equiv \sum_{i,j \in C} A_{ij}$… sum of link weights between nodes in C
  - $\boldsymbol{\Sigma_{tot}} \equiv \sum_{i \in C} k_i$… sum of all link weights of nodes in C

$\boldsymbol{\Sigma_{in}}$:



$\boldsymbol{\Sigma_{tot}}$:



Deriving $\Delta Q(i \to C)$

- **Let's derive $\boldsymbol{\Delta Q(i \to C)}$**
- First, we derive modularity **within $\boldsymbol{C}$**, i.e., $Q(C)$.
- **Define:**
  - $\boldsymbol{\Sigma_{in}} \equiv \sum_{i,j \in C} A_{ij}$… sum of link weights between nodes in C
  - $\boldsymbol{\Sigma_{tot}} \equiv \sum_{i \in C} k_i$… sum of all link weights of nodes in C
- Then, we have

$$Q(C) \equiv \frac{1}{2m} \sum_{i,j \in C} \left[A_{ij} - \frac{k_i k_j}{2m} \right] = \frac{\Sigma_{i,j \in C} A_{ij}}{2m} - \frac{(\Sigma_{i \in C} k_i)(\Sigma_{j \in C} k_j)}{(2m)^2}$$

$$= \frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m} \right)^2$$

Links within the community

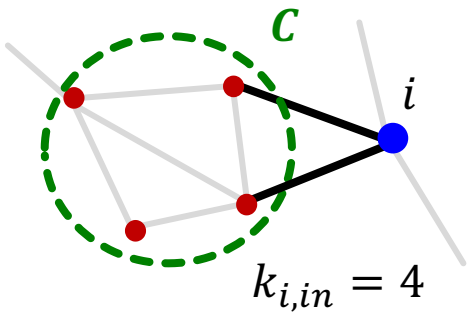
Total links

$Q(C)$ is large when most of the total links are within-community links

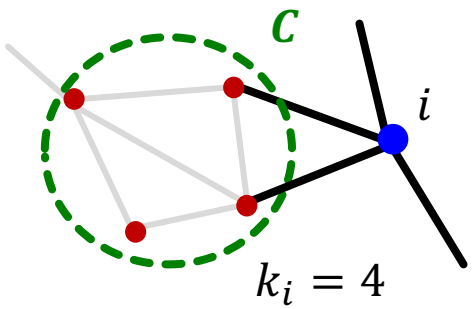
Deriving $\Delta Q(i \to C)$

- **Further define:**
  - $\boldsymbol{k_{i,in}} \equiv \sum_{j \in C} A_{ij} + \sum_{j \in C} A_{ji}$ … sum of link weights connecting node i and C
    - **(note that each edge gets counted twice, see formula)**
  - $\boldsymbol{k_i}$… sum of all link weights (i.e., degree) of node i

$\boldsymbol{k_{i,in}}$:



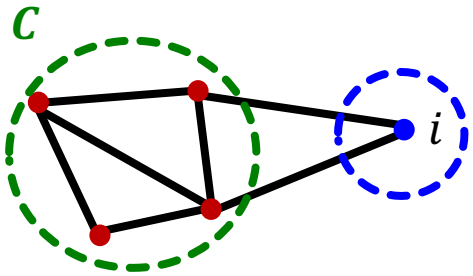
$\boldsymbol{k_i}$:



Deriving $\Delta Q(i \to C)$

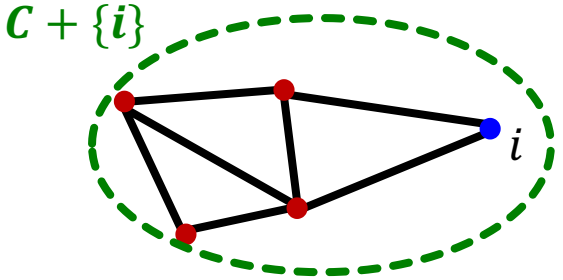
Q ∝ (# edges within group) – (expected # edges within group)

Before merging



Isolated community of node i

After merging

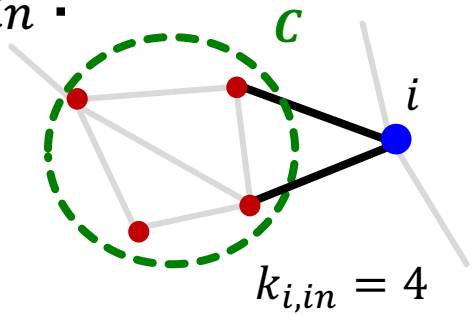


$$Q_{\text{before}} = Q(C) + Q(\{i\})$$
$$= \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2\right] + \left[0 - \left(\frac{k_i}{2m}\right)^2\right]$$

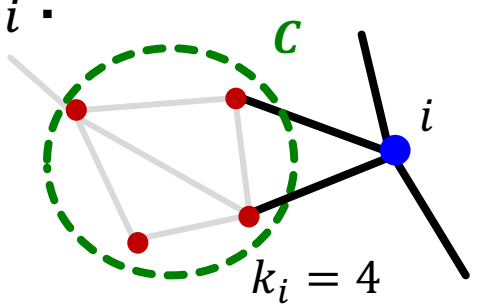
$$Q_{\text{after}} = Q(C + \{i\})$$
$$= \frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2$$

"Σ_{in}" of $C + \{i\}$ — "Σ_{tot}" of $C + \{i\}$

Recall: $k_{i,in}$:



k_i :



Louvain: Modularity Gain

- $\Delta Q(i \to C) = Q_{\text{after}} - Q_{\text{before}}$

$$= \left[\frac{\Sigma_{in}+k_{i,in}}{2m} - \left(\frac{\Sigma_{tot}+k_i}{2m}\right)^2\right]$$

$$- \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

- $\Delta Q(D \to i)$ can be derived similarly.

- **In summary, we can compute:**

$$\Delta Q(D \to i \to C) = \Delta Q(D \to i) + \Delta Q(i \to C)$$

Louvain 1st Phase: Summary

- **Iterate until no node moves to a new community**:
  - For each node $i \in V$ currently in community C, compute the best community C':
  - $C' = \text{argmax}_{C'} \Delta Q(C \to i \to C')$
  - If $\Delta Q(C \to i \to C') > 0$, then **update the community:**
    - $C \leftarrow C - \{i\}$
    - $C' \leftarrow C' + \{i\}$

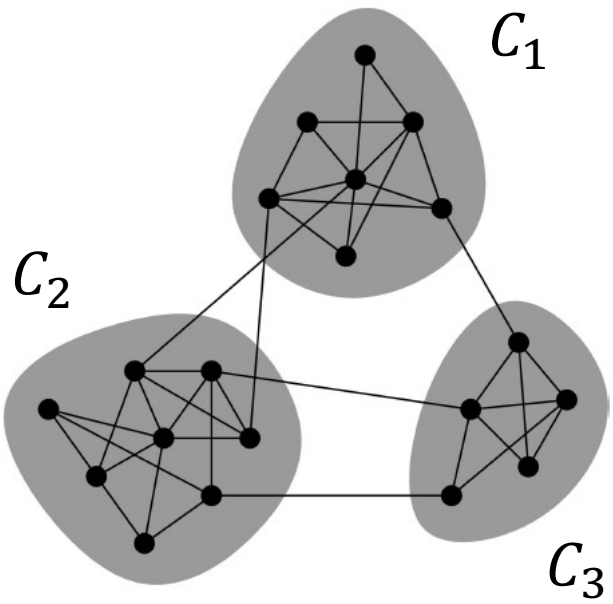
Louvain: 2nd phase (Restructuring)

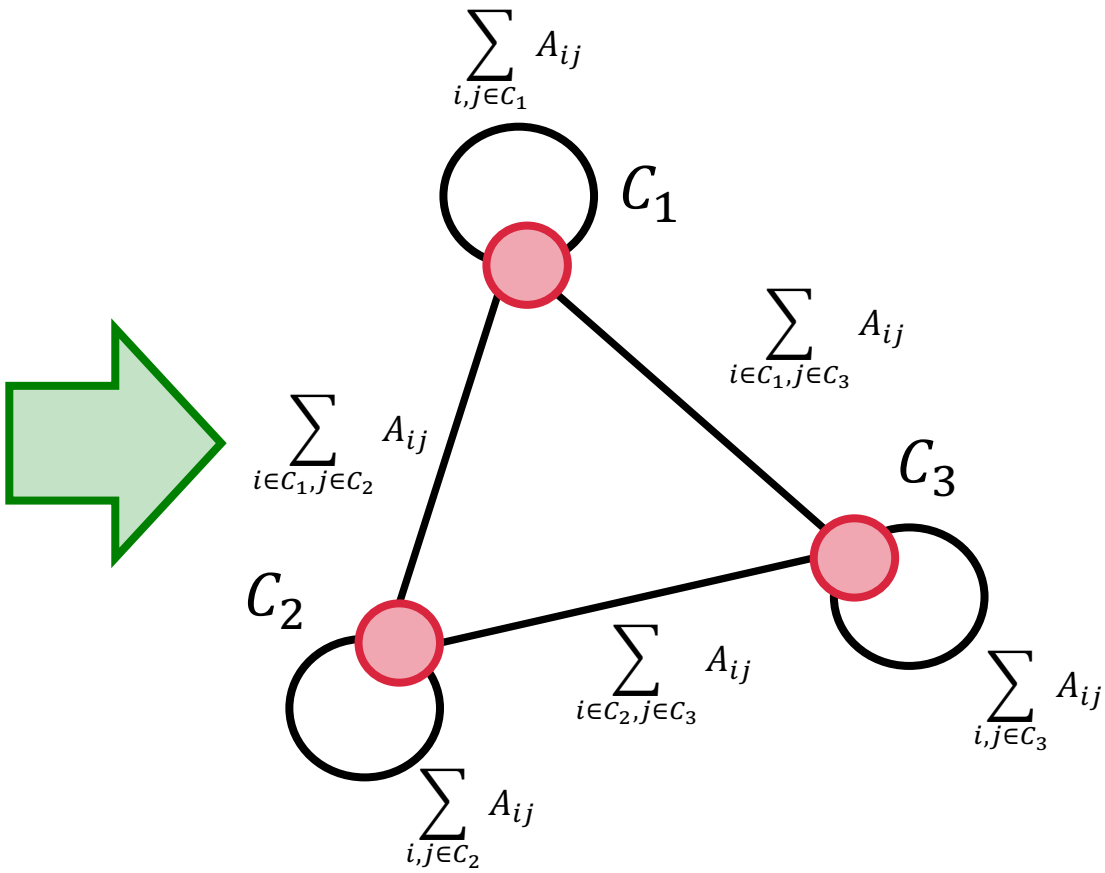
- The communities obtained in the first phase are contracted into **super-nodes**, and the network is created accordingly:
  - Super-nodes are connected if there is at least one edge between the nodes of the corresponding communities
  - The weight of the edge between the two super-nodes is the sum of the weights from all edges between their corresponding communities
- **Phase 1 is then run on the super-node network**

Louvain 2st Phase: Summary

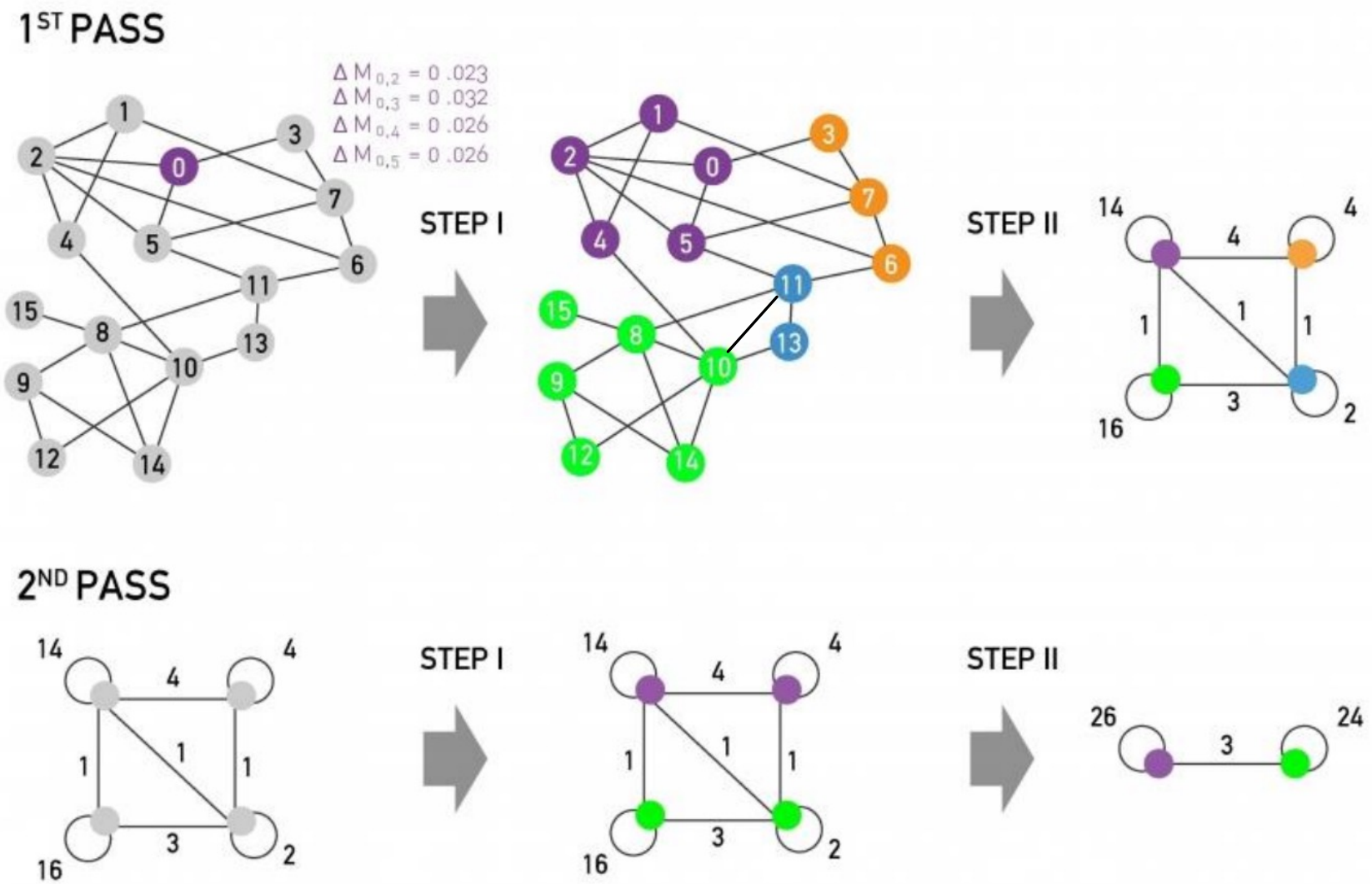
- Super nodes are constructed by merging nodes in the same community.

Community assignment obtained after 1st phase

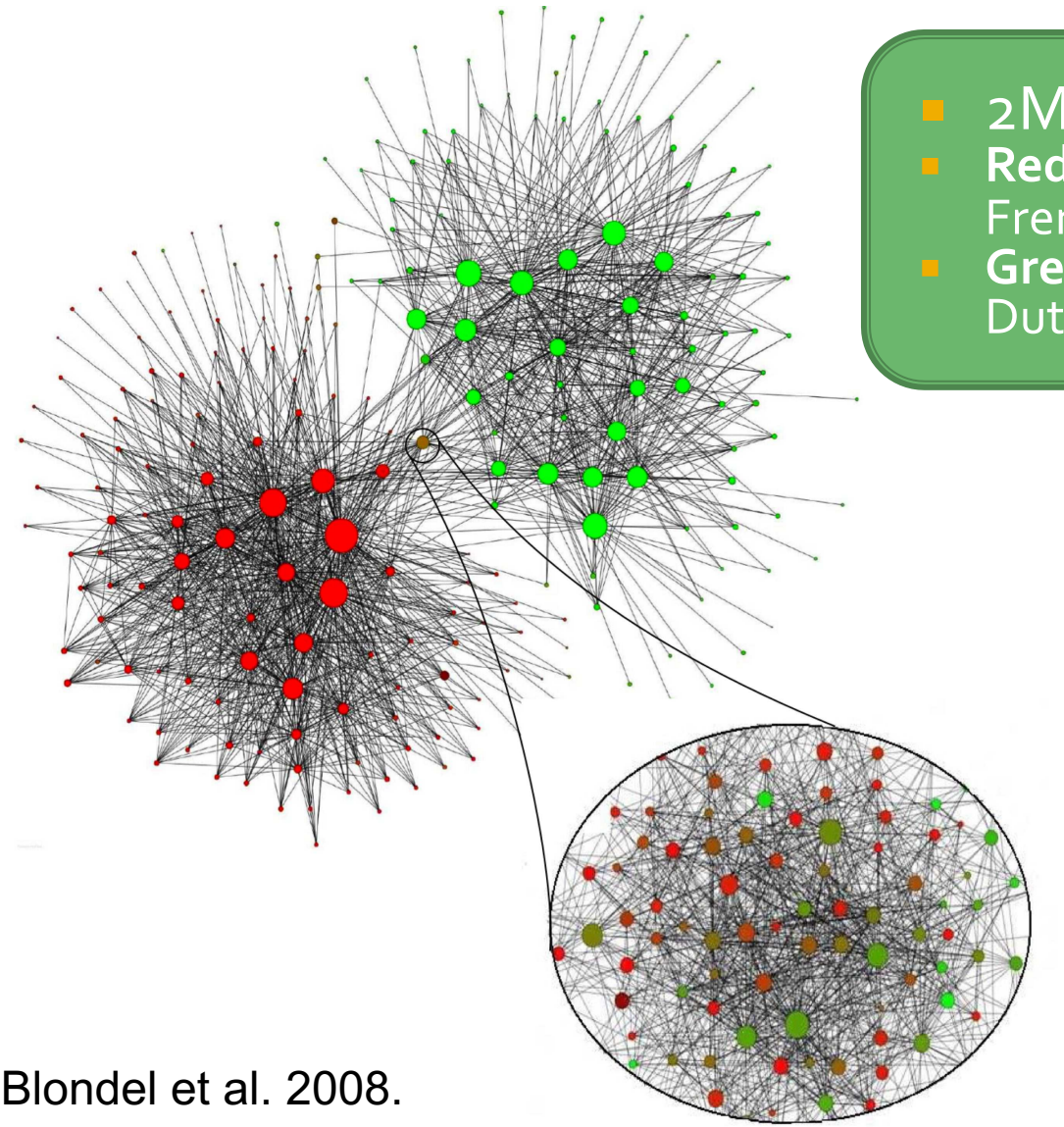




Louvain Algorithm



Belgian Mobile phone network



- 2M nodes
- **Red nodes**: French speakers
- **Green nodes**: Dutch speakers

Adopted from Blondel et al. 2008.

Summary: Modularity

- **Modularity:**
  - Overall quality of the partitioning of a graph into communities
  - Used to determine the number of communities
- **Louvain modularity maximization:**
  - Greedy strategy
  - Great performance, scales to large networks

Discussion

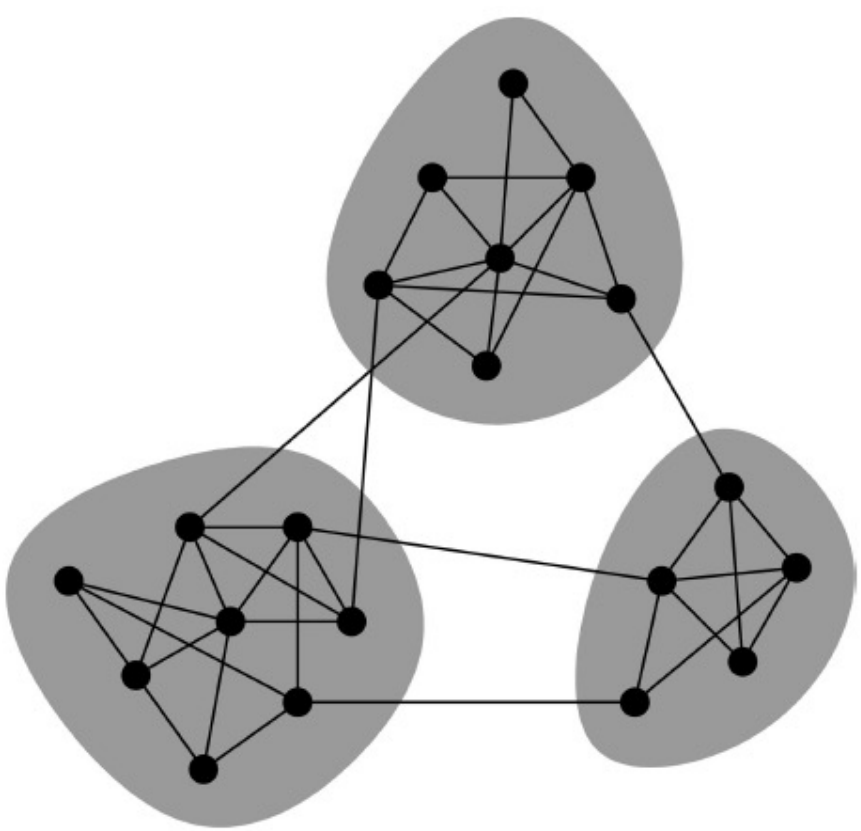
Questions?

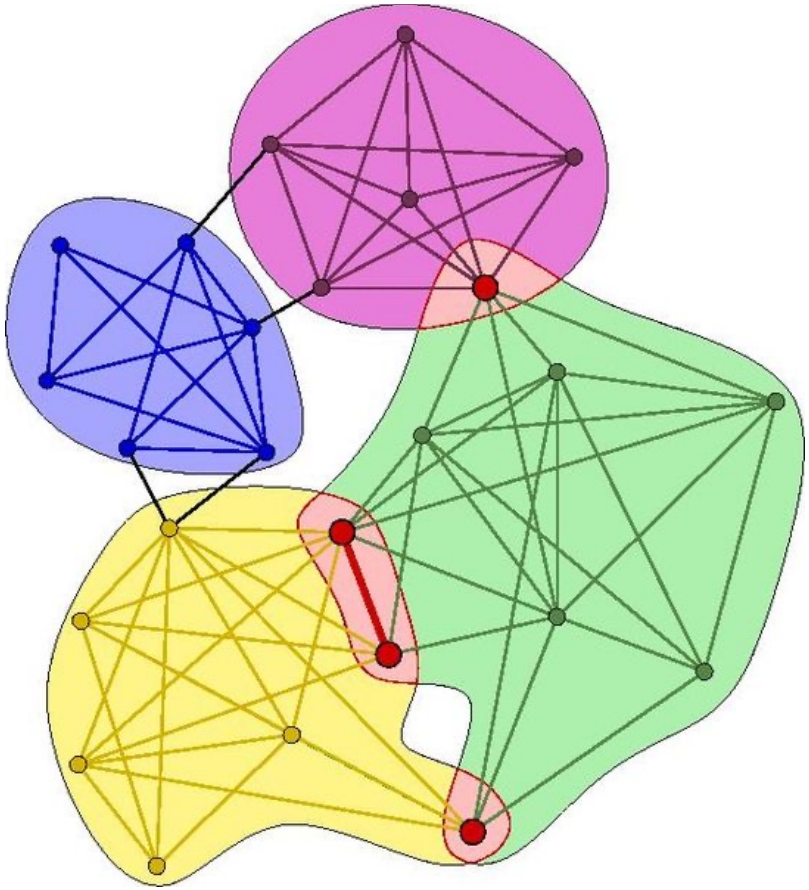
Stanford CS224W: Detecting Overlapping Communities: AGM & NOCD

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

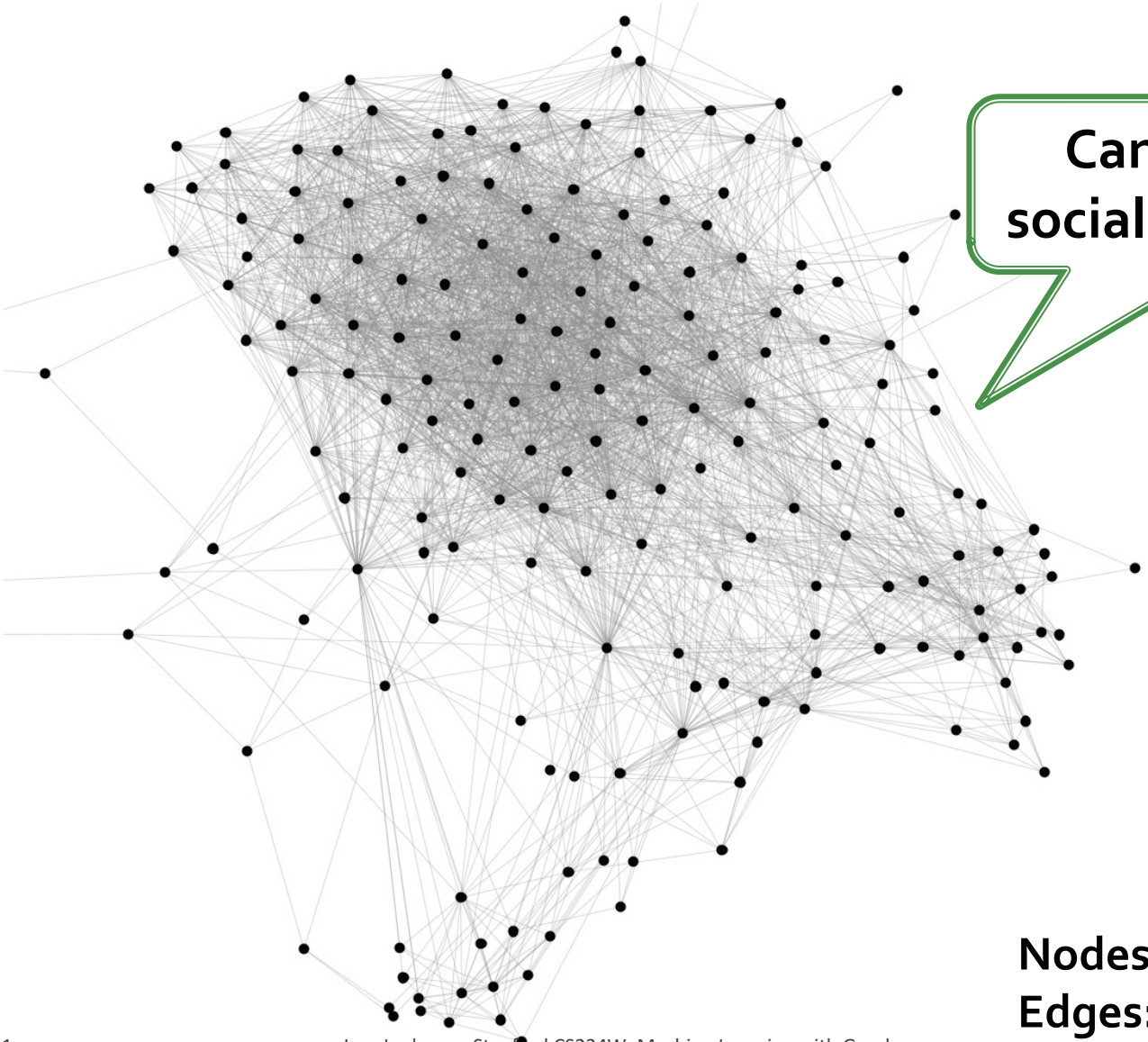
Overlapping Communities

- **Non-overlapping vs. overlapping communities**





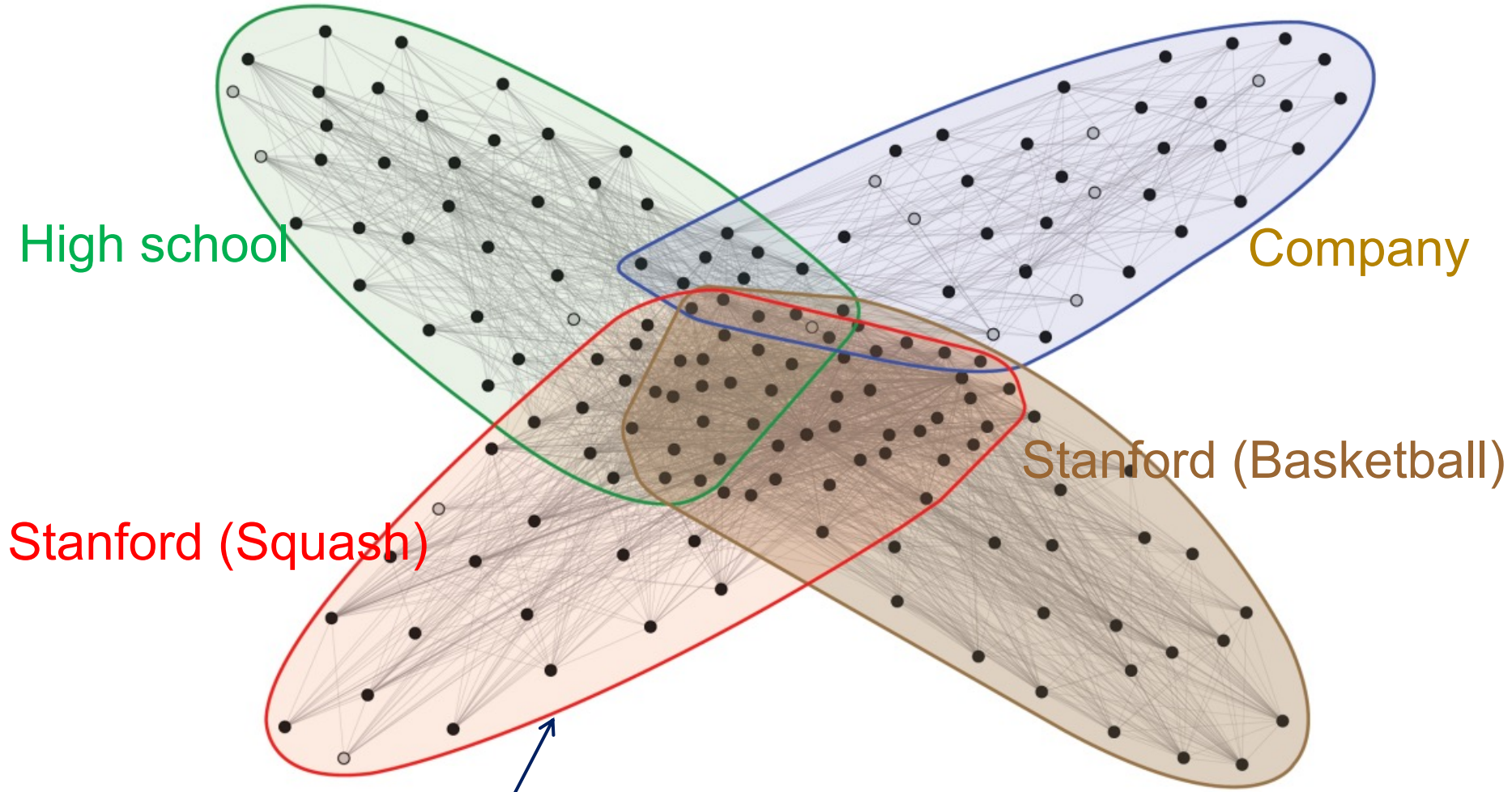
Facebook Ego-network



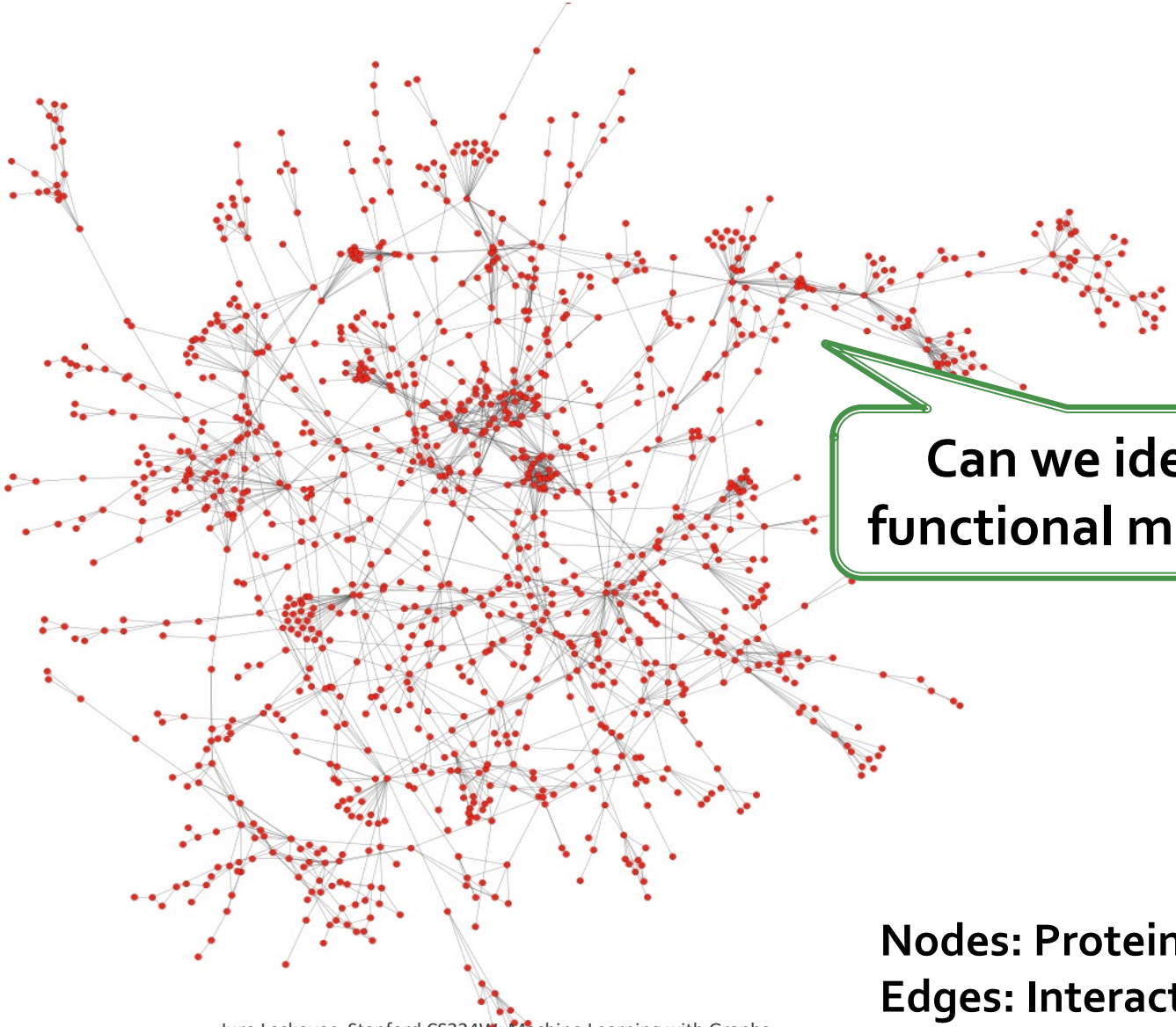
Can we identify social communities?

Nodes: Users
Edges: Friendships

Facebook Ego-network



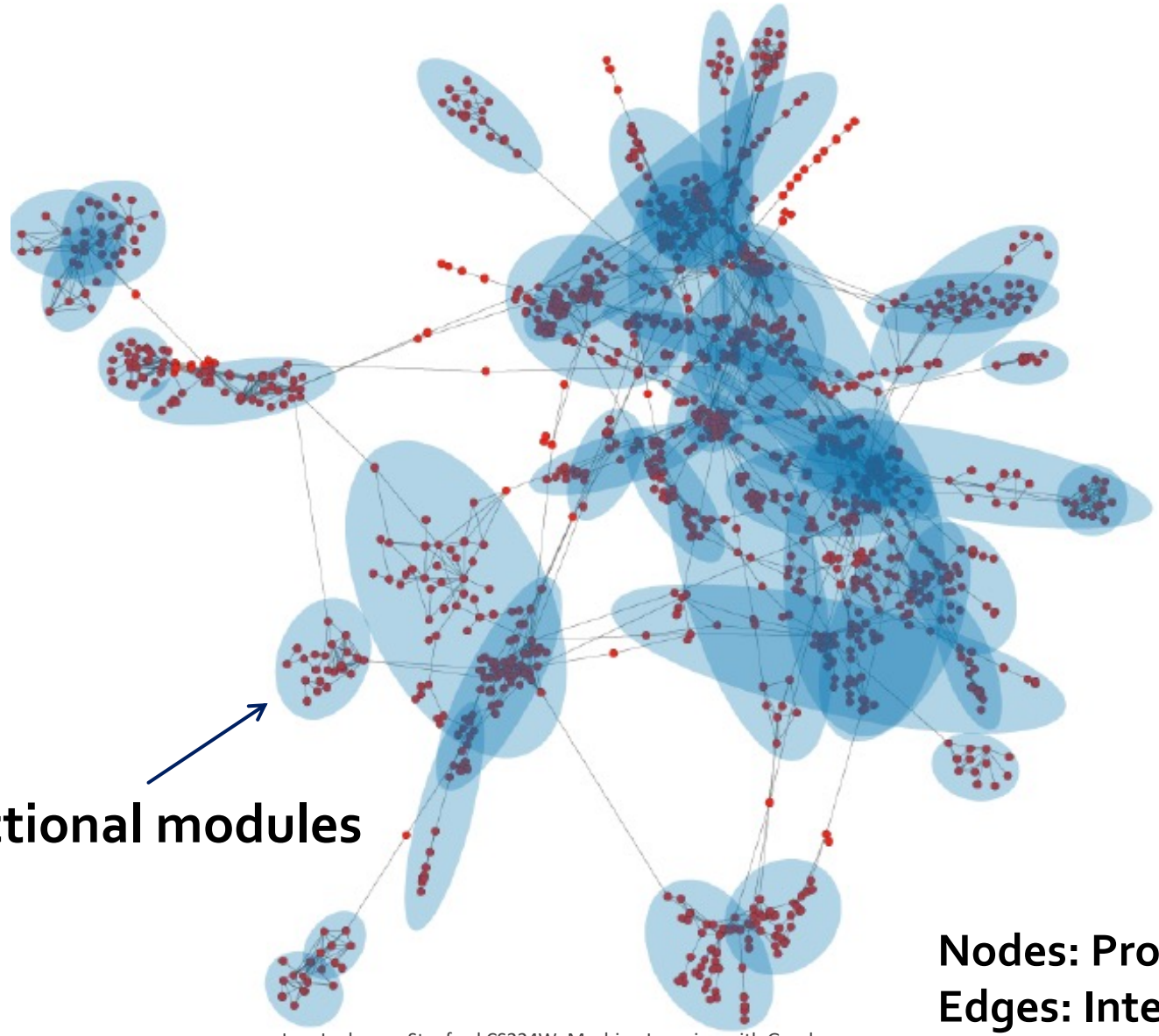
Protein-Protein Interactions



Can we identify functional modules?

Nodes: Proteins
Edges: Interactions

Protein-Protein Interactions



Functional modules

Nodes: Proteins
Edges: Interactions

Plan of Action

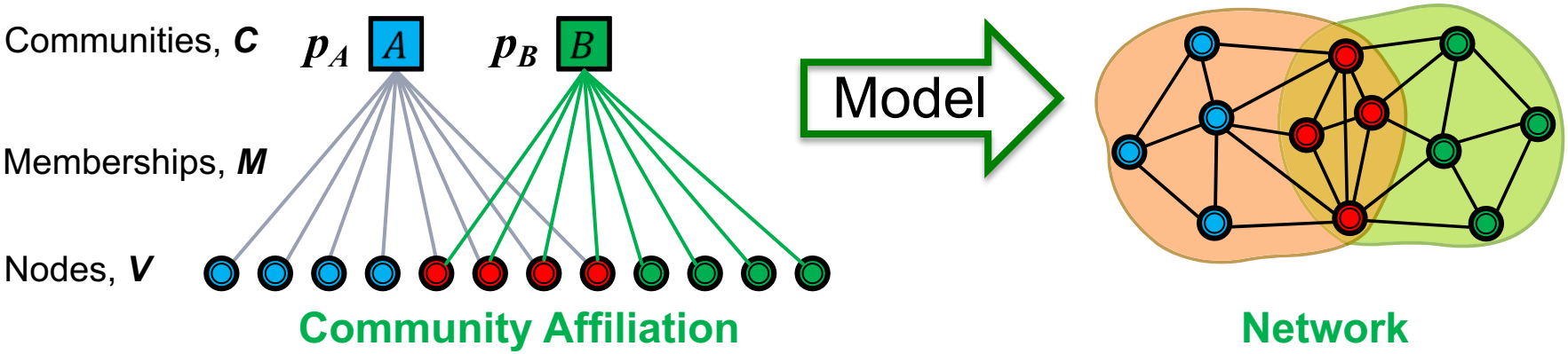
Step 1)

- Define a generative model for graphs that is based on node community affiliations
  - **Community Affiliation Graph Model (AGM)**

Step 2)

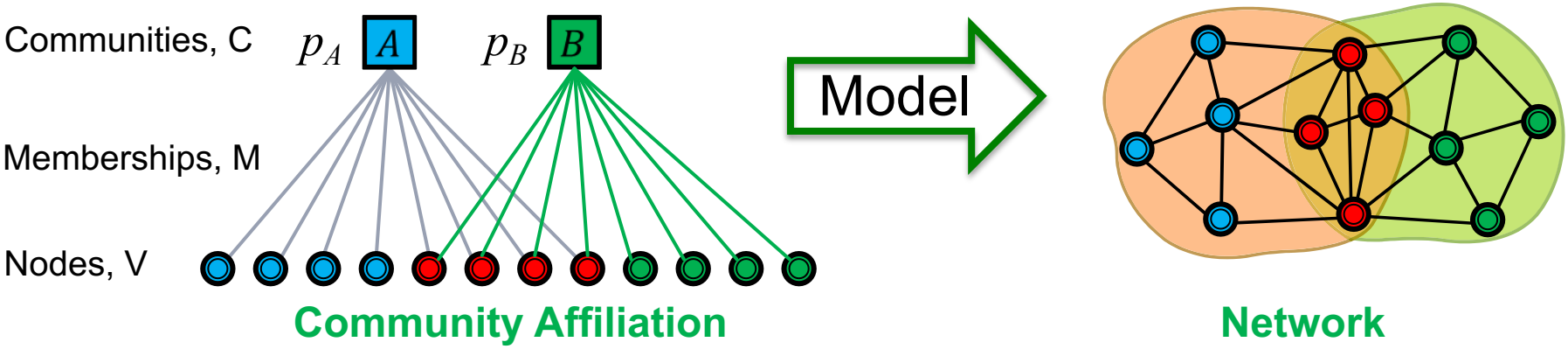
- Given graph G, make the assumption that G was generated by AGM
- Find the best AGM that could have generated G
- **And this way we discover communities**

Community-Affiliation Graph Model (AGM)



- **Generative model: How is a network generated from community affiliations?**
- **Model parameters:**
  - Nodes V, Communities C, Memberships M
  - Each community c has a single probability p_c

AGM: Generative Process

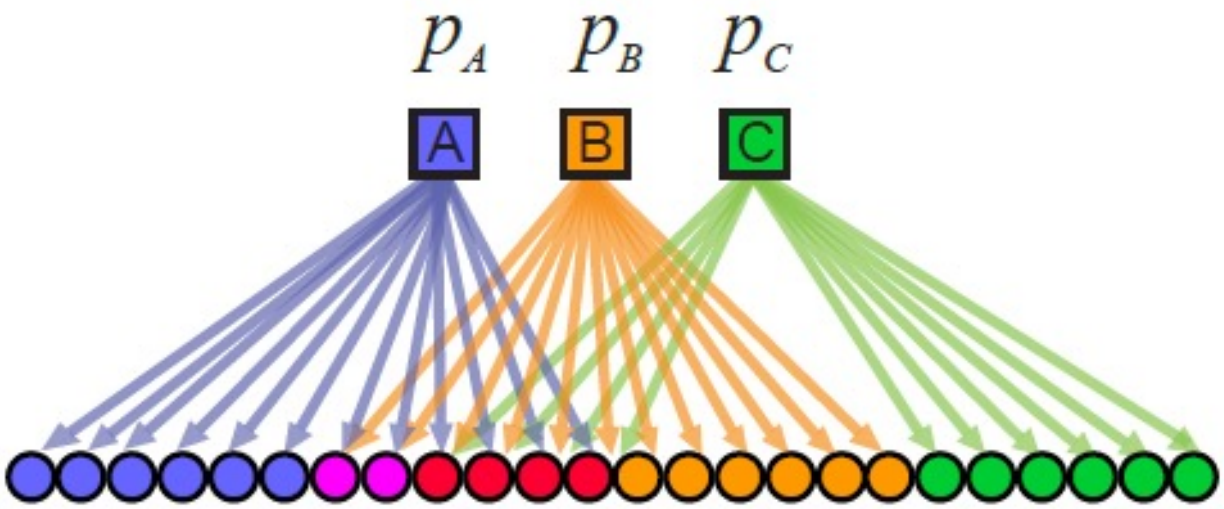


- **Given parameters (V, C, M, $\{p_c\}$)**
  - Nodes in community c connect to each other by flipping a coin with probability p_c
  - **Nodes that belong to multiple communities have multiple coin flips**
    - If they "miss" the first time, they get another chance through the next community

$$p(u,v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

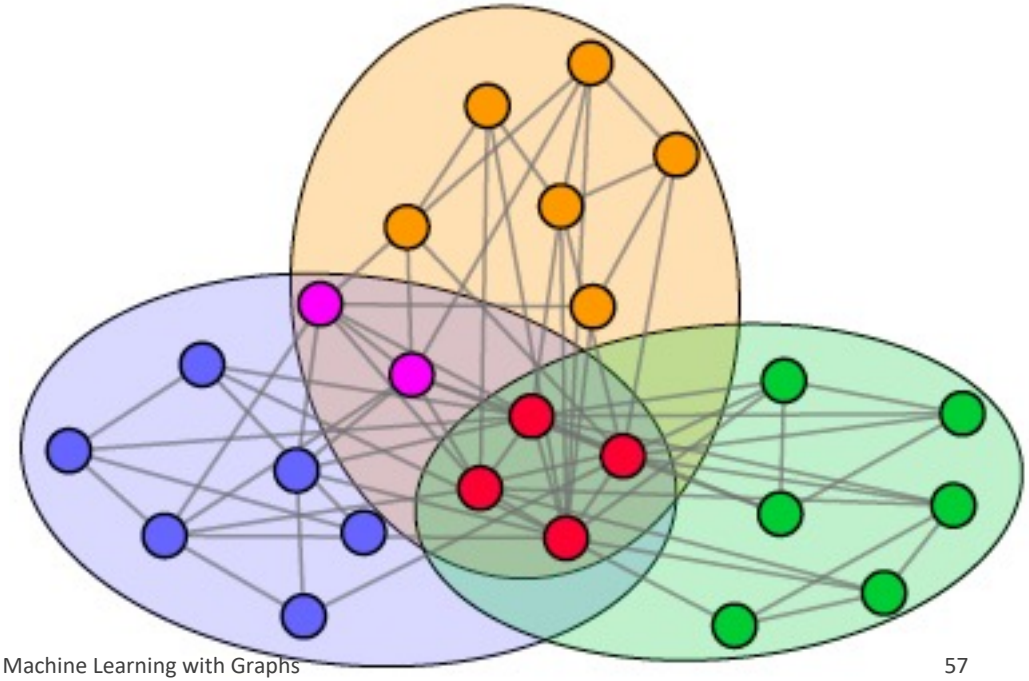
Note: If nodes u and v have no communities in common, then p(u,v)=0. We resolve this by having a background "epsilon" community that every node is a member of.

AGM: Dense Overlaps



Model

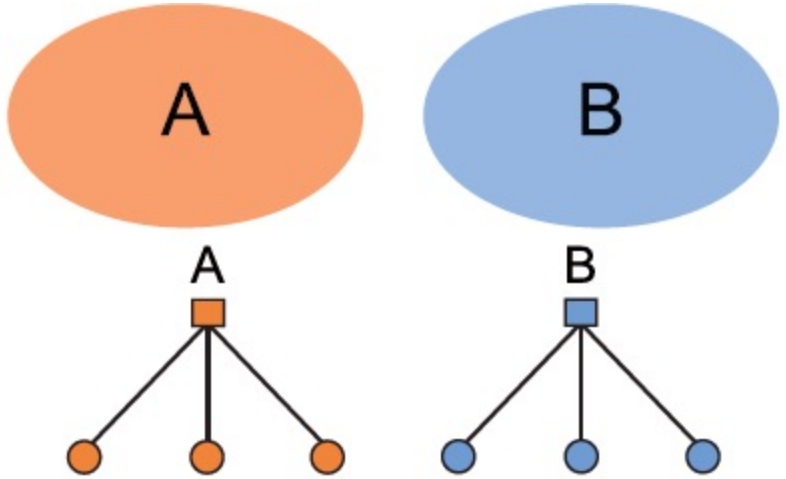
Network

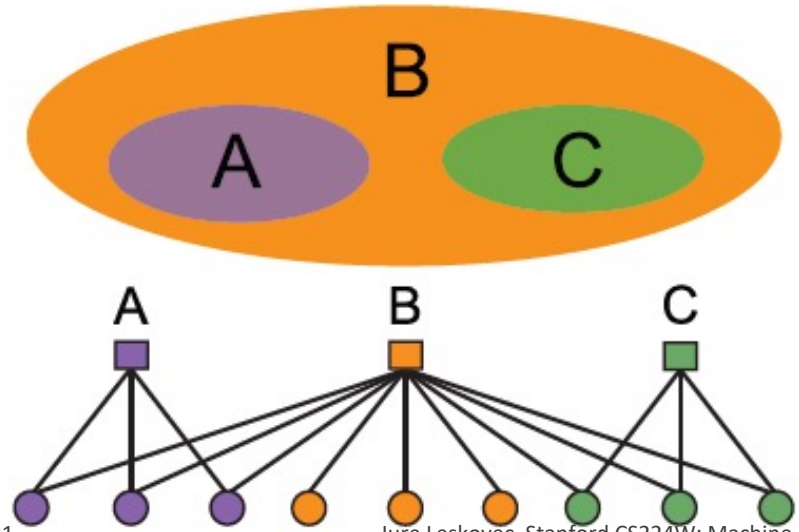


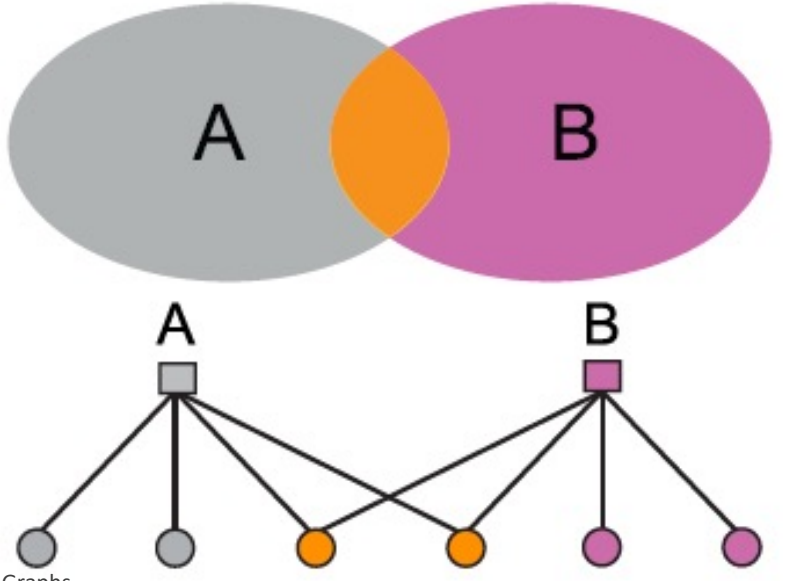
AGM: Flexibility

[icdm '12]

- **AGM can express a variety of community structures:**
  Non-overlapping,
  Overlapping, Nested

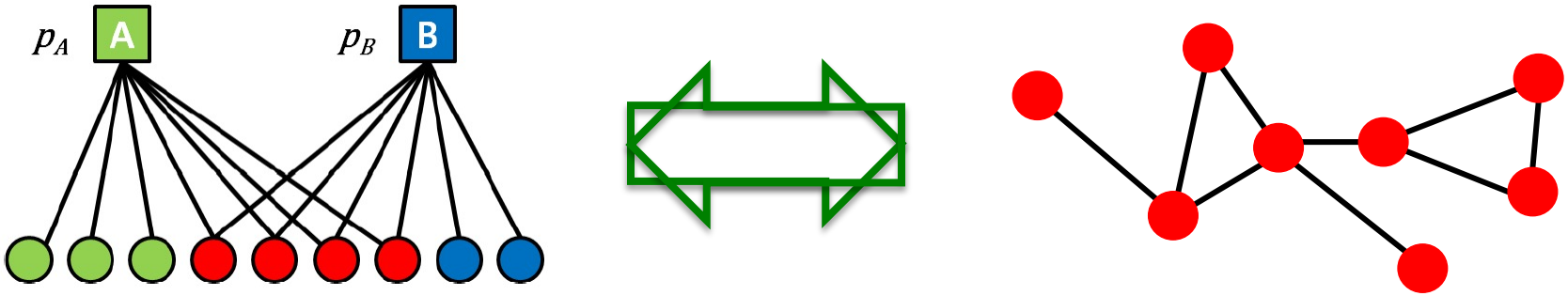






Detecting Communities

- **Detecting communities with AGM:**



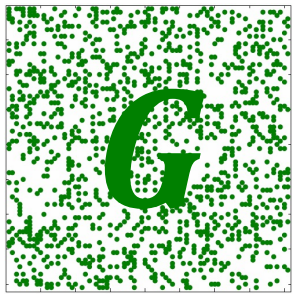
Given a Graph, find (fit) the model F

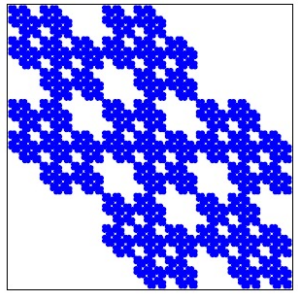
Graph Fitting

How to estimate model parameters F given a G?

- **Maximum likelihood estimation**
- Given real graph G
- Find model/parameters F which

$$\arg\max_F P(G \mid \text{Model } F)$$

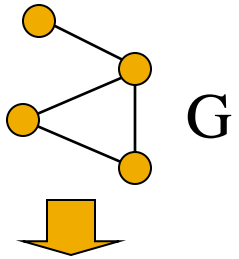




- **To solve this we need to:**
  - Efficiently calculate $\boldsymbol{P(G|F)}$
  - Then maximize over $\boldsymbol{F}$ (e.g., using gradient descent)

Graph Likelihood $P(G|F)$

- Given G and F we calculate likelihood that F generated G: $P(G|F)$



F →

0.25	0.10	0.10	0.04
0.05	0.15	0.02	0.06
0.05	0.02	0.15	0.06
0.01	0.03	0.03	0.09

1	0	1	1
0	1	0	1
1	0	1	1
1	1	1	1

G

$P(G|F)$

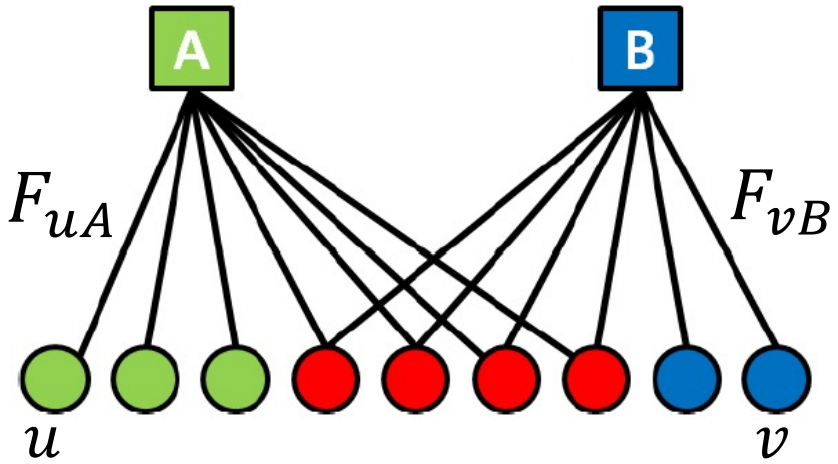
$$P(G|F) = \prod_{(u,v)\in G} P(u,v) \prod_{(u,v)\notin G} (1 - P(u,v))$$

Likelihood of edges in the graph Likelihood of edges not in the graph

[wsdm '13]

"Relaxing" AGM: Towards P(u,v)

- **"Relax" the AGM: Memberships have strengths**



- F_{uA}: The membership strength of node u to community $\boldsymbol{A}$ ($F_{uA} = \mathbf{0}$: no membership)

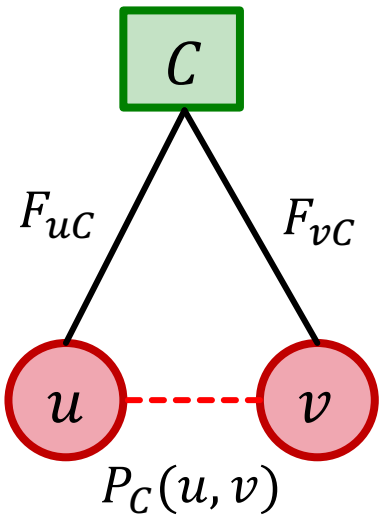
"Relaxing" AGM: Towards P(u,v)

- For community C, we model the probability of u and v being connected as

$$P_C(u,v) = 1 - exp(-F_{uC} \cdot F_{vC})$$

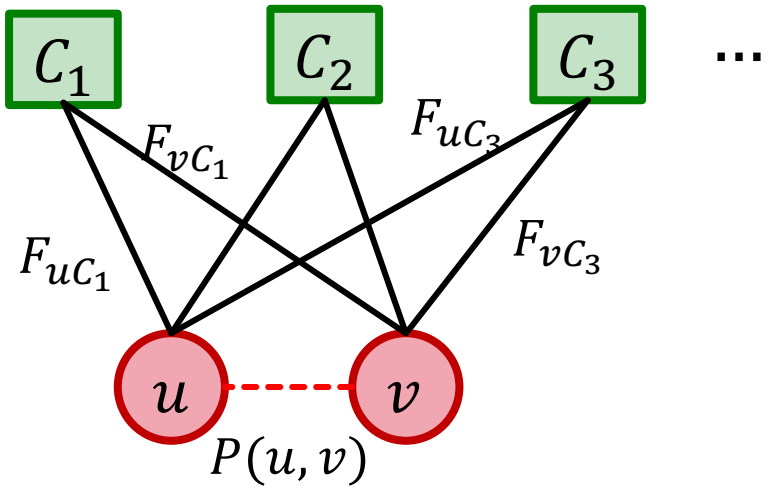
Non-negative membership strength

  - $P_C(u,v)$ satisfies $0 \leq P_C(u,v) \leq 1$ (**valid probability**) because $F_{uC} \cdot F_{vC} \geq 0$.
    - $P_C(u,v) = 0$ iff $F_{uC} \cdot F_{vC} = 0$ (i.e., $F_{uC} = 0$ or $F_{vC} = 0$)
      - Nodes u or v are not connected via C iff **at least one of them** has zero membership strength for C.
    - $P_C(u,v) \approx 1$ iff $F_{uC} \cdot F_{vC}$ is large.
      - Nodes u or v are connected via C iff **both** u and v have nigh membership strength for C.



"Relaxing" AGM: Towards P(u,v)

- Nodes u and v can connected via multiple communities $C_i \in \Gamma$ (**a set of all communities**).



- Probability that u and v are connected by at least one of the communities:
  - $P(u,v) = 1 - \prod_{C \in \Gamma}(1 - P_C(u,v))$

Probability that u and v are ***not*** connected by any communities

"Relaxing" AGM: Towards P(u,v)

- **Expanding $P(u, v)$:**

$$P(u,v) = 1 - \prod_{C \in \Gamma} \left(1 - P_C(u,v)\right)$$

$1 - \exp(-F_{uC} \cdot F_{vC})$

$$= 1 - \prod_{C \in \Gamma} \exp(-F_{uC} \cdot F_{vC})$$

$$= 1 - \exp\left(-\sum_{c \in \Gamma} F_{uC} \cdot F_{vC}\right)$$

$$= 1 - \exp(-\boldsymbol{F}_{\boldsymbol{u}}^{T} \boldsymbol{F}_{\boldsymbol{v}})$$

Dot product

$\boldsymbol{F}_{\boldsymbol{u}}$: A vector of $\{F_{uC}\}_{C \in \Gamma}$

$\boldsymbol{F}_{\boldsymbol{v}}$: A vector of $\{F_{vC}\}_{C \in \Gamma}$

[KDD' 19]

NOCD Model

- **Prob. of nodes u, v linking is proportional to the strength of shared memberships:**

$$\mathbf{P}(u, v) = \mathbf{1} - \mathbf{exp}(-\boldsymbol{F}_u^T \boldsymbol{F}_v)$$

- **Given a network $G(V, E)$, we maximize the likelihood (probability) of G under our model**

$$P(G|\boldsymbol{F}) = \prod_{(u,v)\in E} P(u, v) \prod_{(u,v)\notin E} (1 - P(u, v))$$

$$= \prod_{(u,v)\in E} \left(\mathbf{1} - \mathbf{exp}(-\boldsymbol{F}_u^T \boldsymbol{F}_v)\right) \prod_{(u,v)\notin E} \mathbf{exp}(-\boldsymbol{F}_u^T \boldsymbol{F}_v)$$

NOCD Model

- Likelihood involves a product of many small probabilities → Numerically unstable.
- We consider the **log** likelihood:

$$\log(P(G|\boldsymbol{F}))$$

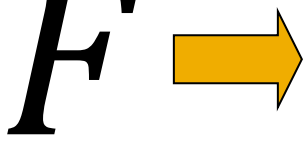
$$= \log\left(\prod_{(u,v)\in E}\left(1-\exp(-\boldsymbol{F}_u^T\boldsymbol{F}_v)\right)\prod_{(u,v)\notin E}\exp(-\boldsymbol{F}_u^T\boldsymbol{F}_v)\right)$$

$$= \sum_{(u,v)\in E}\log(1-\exp(-\boldsymbol{F}_u^T\boldsymbol{F}_v)) - \sum_{(u,v)\notin E}\boldsymbol{F}_u^T\boldsymbol{F}_v$$

$\equiv \ell(\boldsymbol{F})$: **Our objective**

NOCD Model

- Recall that

F 

0.25	0.10	0.10	0.04
0.05	0.15	0.02	0.06
0.05	0.02	0.15	0.06
0.01	0.03	0.03	0.09

- Given a graph $G = (A, X)$, how do we find F?
  - F… matrix of node-to-community membership

Key idea of **Neural Overlapping Community Detection (NOCD)**

Generate F using a GNN!
Train a GNN(A, X) to output F

[WSDM' 13]

Why use a GNN?

- **Benefits of GNN:**
  - GNN can **generalize** to other graphs! Otherwise, we need to optimize F for each new graph.
  - GNN takes good use of the **graph structure -- adjacency matrix A and node feature X**.

For example: X includes occupation, hobbies, education of users on FB graph. After trained on FB graph, the GNN may generalize to another social network from Instagram / Twitter.

NOCD Model

Other GNN variants also work

- **Consider a 2-layer GCN as an example**

$$\boldsymbol{F} = GCN(A, X) = \sigma\Big(\tilde{A}\,\underbrace{\sigma\big(\tilde{A}XW_1\big)}_{\text{The first layer of GCN}}\,W_2\Big)$$

where $\tilde{A} = D^{-1}A$.

The second layer of GCN

- **Optimizing $\ell(\boldsymbol{F})$** w.r.t GCN parameters W_1 and W_2

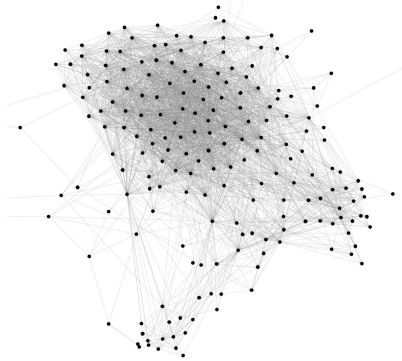
$$\ell(\boldsymbol{F}) = \sum_{(u,v)\in E} \log(1 - \exp(-\boldsymbol{F}_u^T\boldsymbol{F}_v)) - \sum_{(u,v)\notin E} \boldsymbol{F}_u^T\boldsymbol{F}_v$$

NOCD Model

- **Issues with $\ell(\boldsymbol{F})$**

$$\ell(\boldsymbol{F}) = \sum_{(u,v)\in E} \log(1 - \exp(-\boldsymbol{F}_u^T \boldsymbol{F}_v)) - \sum_{(u,v)\notin E} \boldsymbol{F}_u^T \boldsymbol{F}_v$$

Real-world graphs are often **extremely sparse**.



$$\frac{\text{\# existing edges}}{\text{\# possible edges}} = \frac{|E|}{n^2} \ll 1$$

FB ego-network
Ratio: 0.0054

PPI network
Ratio: 0.000736

NOCD Model

- **Issues of $\ell(\boldsymbol{F})$**

$$\ell(\boldsymbol{F}) = \sum_{(u,v)\in E} \log(1 - \exp(-\boldsymbol{F}_u^T \boldsymbol{F}_v)) - \sum_{(u,v)\notin E} \boldsymbol{F}_u^T \boldsymbol{F}_v$$

Real-world graphs are often **extremely sparse**.

The second term has **a much larger contribution**.

- **Solution: Take average of both terms**

$$\ell(\boldsymbol{F}) = \frac{1}{|E|} \sum_{(u,v)\in E} \log(1 - \exp(-\boldsymbol{F}_u^T \boldsymbol{F}_v)) - \frac{1}{n^2 - |E|} \sum_{(u,v)\notin E} \boldsymbol{F}_u^T \boldsymbol{F}_v$$

↑ # existing edges

↑ # non-existing edges

NOCD Model

- **After optimizing $\ell(\boldsymbol{F})$**

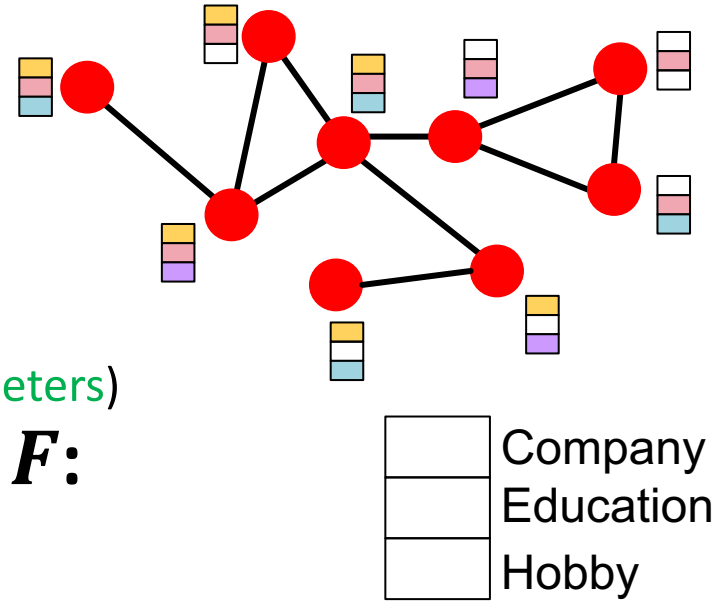
$$\ell(\boldsymbol{F}) = \frac{1}{|E|} \sum_{(u,v)\in E} \log(1 - \exp(-\boldsymbol{F}_u^T \boldsymbol{F}_v)) - \frac{1}{n^2 - |E|} \sum_{(u,v)\notin E} \boldsymbol{F}_u^T \boldsymbol{F}_v$$

- **Assign nodes to communities**
  - Set a threshold ρ:
    - a hyperparameter, NOCD picks $\rho = 0.5$
  - Assign node u to community C if $\boldsymbol{F}_{uC} > \rho$

The strength of node u belong to C

Synthetic Example

- **Given**
  - a synthetic social network
  - #nodes = 9
  - set #communities = 2 (hyperparameters)
- **NOCD uses a GNN to output F:**

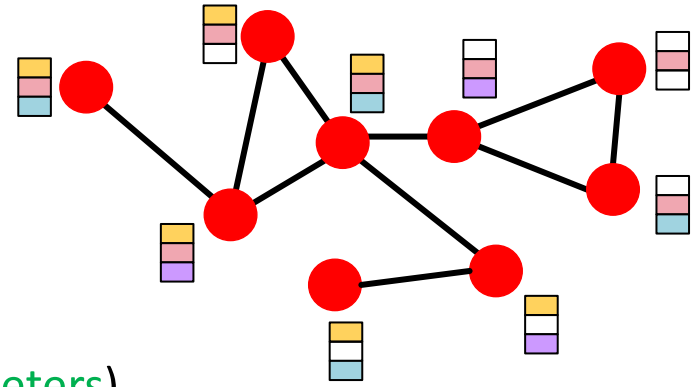


0.65	0.10
0.75	0.15
0.54	0.02
0.66	0.73
0.8	0.72
0.77	0.66
0.59	0.85
0.2	0.88
0.15	0.9

F is a #node by #community matrix.
An entry $F_{ij} \in [0,1]$ represents the strength of node i in community j

Synthetic Example

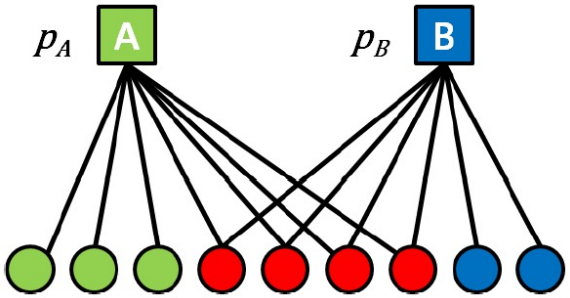
- **Given**
  - a synthetic social network
  - #nodes = 9
  - set #communities = 2 (hyperparameters)



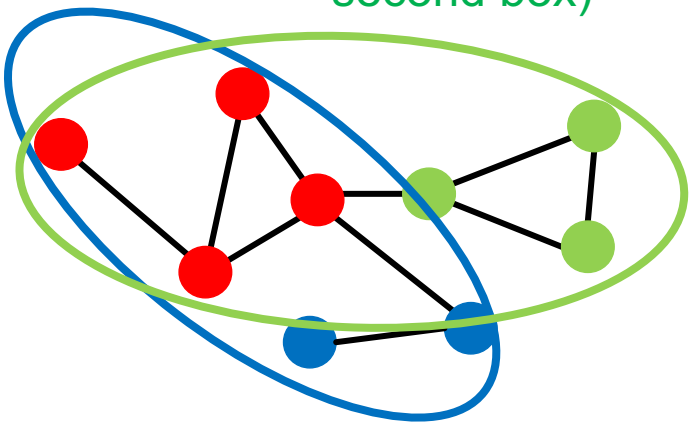
- **NOCD uses a GNN to output F:**

0.65	0.10
0.75	0.15
0.54	0.02
0.66	0.73
0.8	0.72
0.77	0.66
0.59	0.85
0.2	0.88
0.15	0.9

Set threshold $\rho = 0.5$, we can derive the affiliation graph.



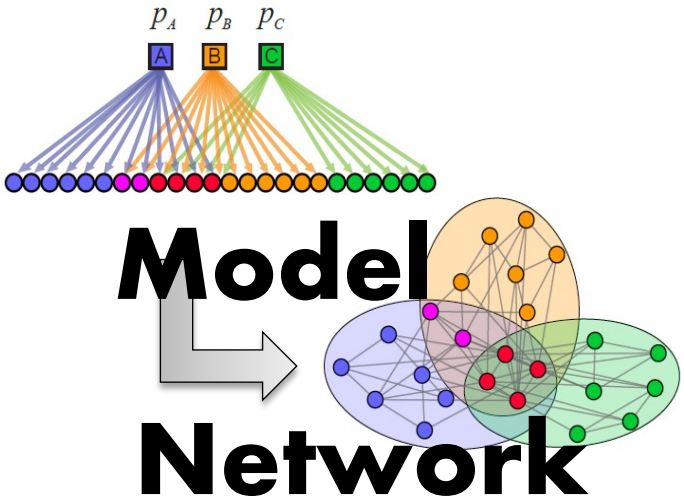
same education background (the second box)



same company (the first box)

NOCD: Summary

- **NOCD** defines the model to generate a network with **overlapping community structure**.



- Given a graph, **NOCD**'s parameters (**membership strength of each node**) can be estimated by maximizing the log-likelihood of generating the graph under the model.

Discussion

Questions?

Summary

Community structure:

- Community structure is natural to networks
- Tightly connected edges are also socially strong, and weakly connected edges are socially weak
- Modularity measures edges within communities
- Louvain modularity calculates disjoint communities, and provides hierarchical clustering
- NOCD calculates overlapping communities, using AGM & a simple 2-layer graph convolutional GNN

Links

Algorithm Links:

- Louvain - https://arxiv.org/abs/0803.0476
- BigCLAM - http://i.stanford.edu/~crucis/pubs/paper-nmfagm.pdf
- NOCD - https://arxiv.org/abs/1909.12201
  Poster - https://www.in.tum.de/fileadmin/w00bws/daml/nocd/nocd_poster_kdd19.pdf

What's next

Session Roadmap:

- **Today:** Community Structure
- **Next:** GNNs for Recommender Systems
- **Then:** Deep Generative Models for Graphs