Community Structure in Networks

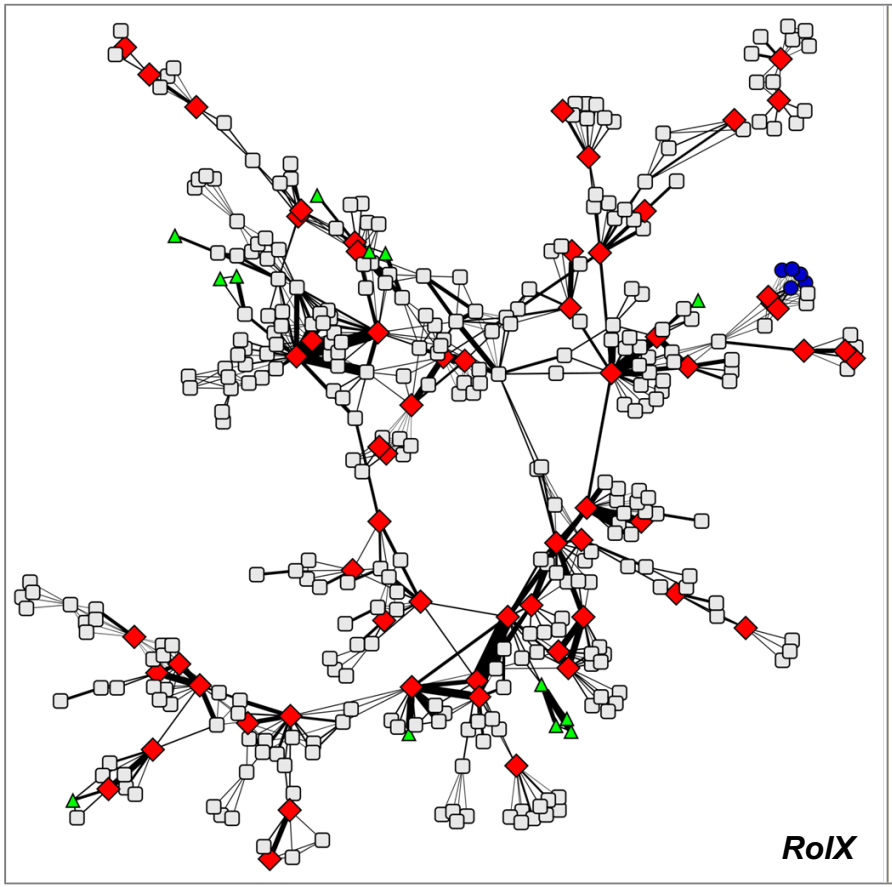
CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

http://cs224w.stanford.edu

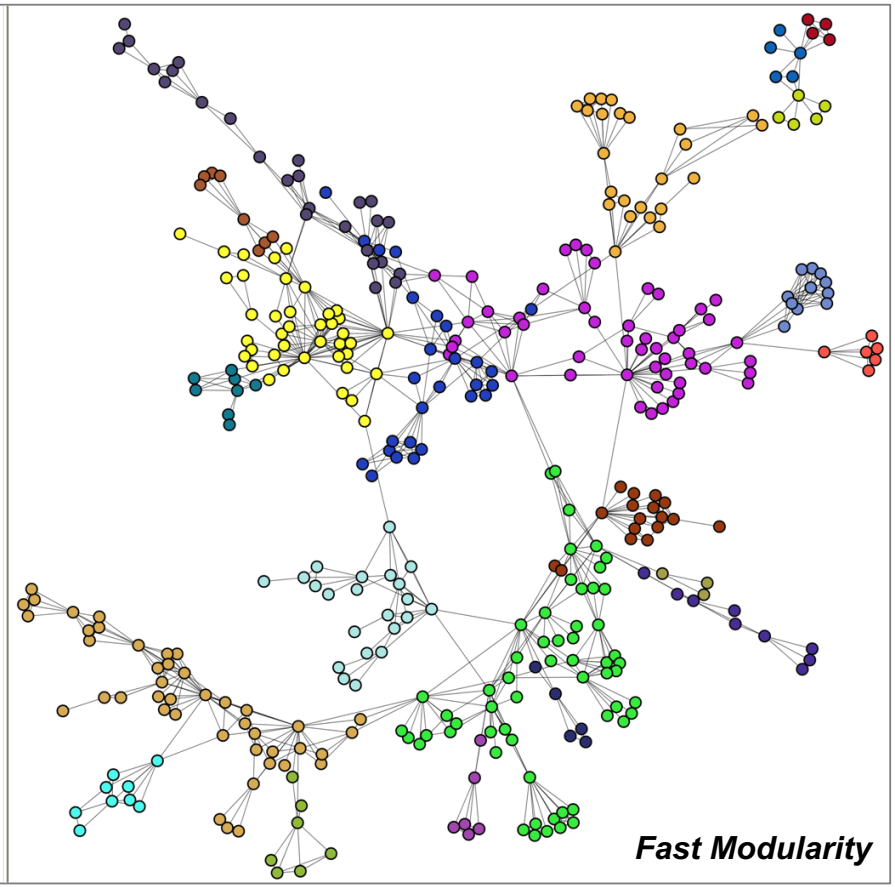
Roles and Communities: Example

Last Lecture: Roles



Henderson, *et al.,* KDD 2012

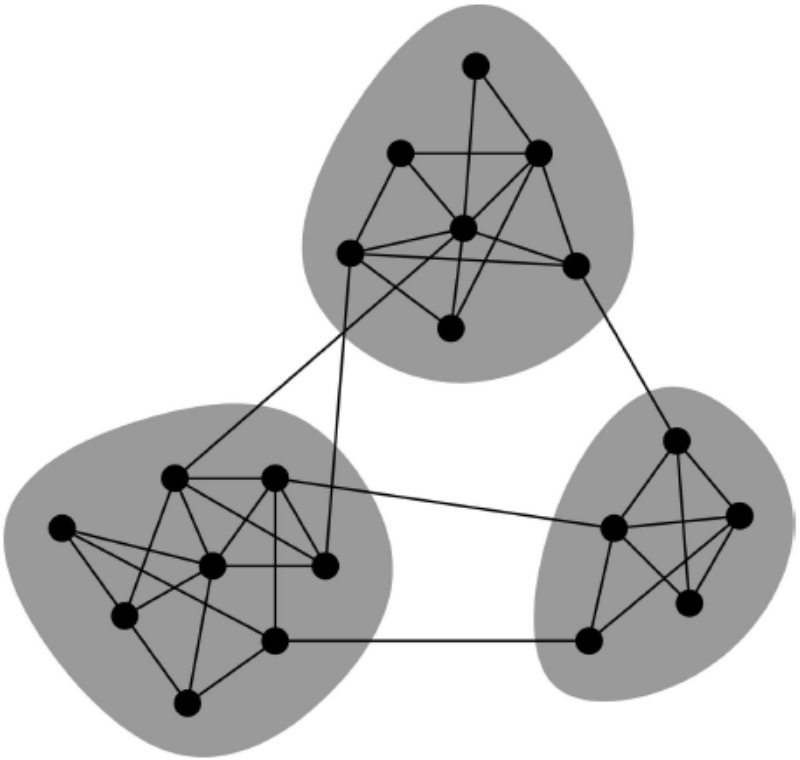
This Lecture: Communities



Clauset, *et al.,* Phys. Rev. E 2004

Networks & Communities

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



- What led to such a conceptual picture?

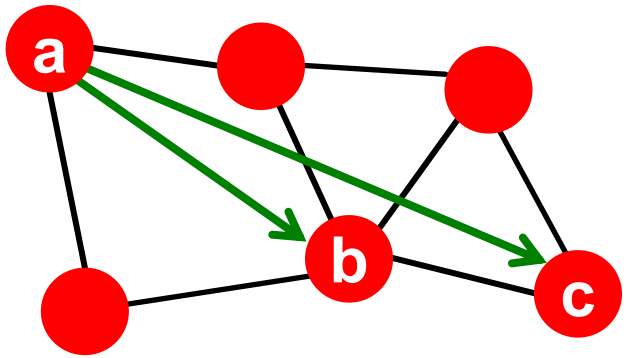
Networks: Flow of Information

- **How does information flow through the network?**
  - **What structurally distinct roles do nodes play?**
  - **What roles do different links ("short" vs. "long") play?**
- **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?**

[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**
- **Structural role: Triadic Closure**

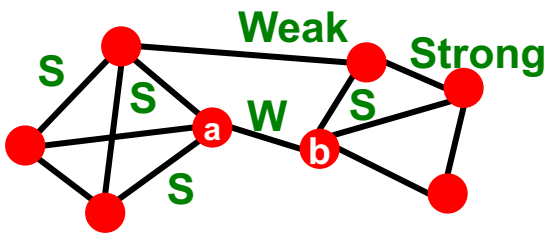


Which edge is more likely, a-b or a-c?

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

Granovetter's Explanation

- **Granovetter makes a connection between the social and structural role of an edge**
- **First point: Structure**
  - Structurally embedded 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

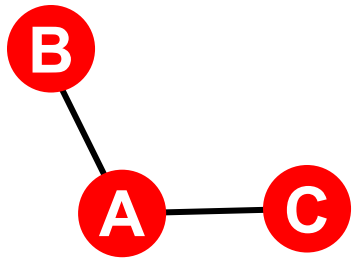


Reasons for Triadic Closure

- **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

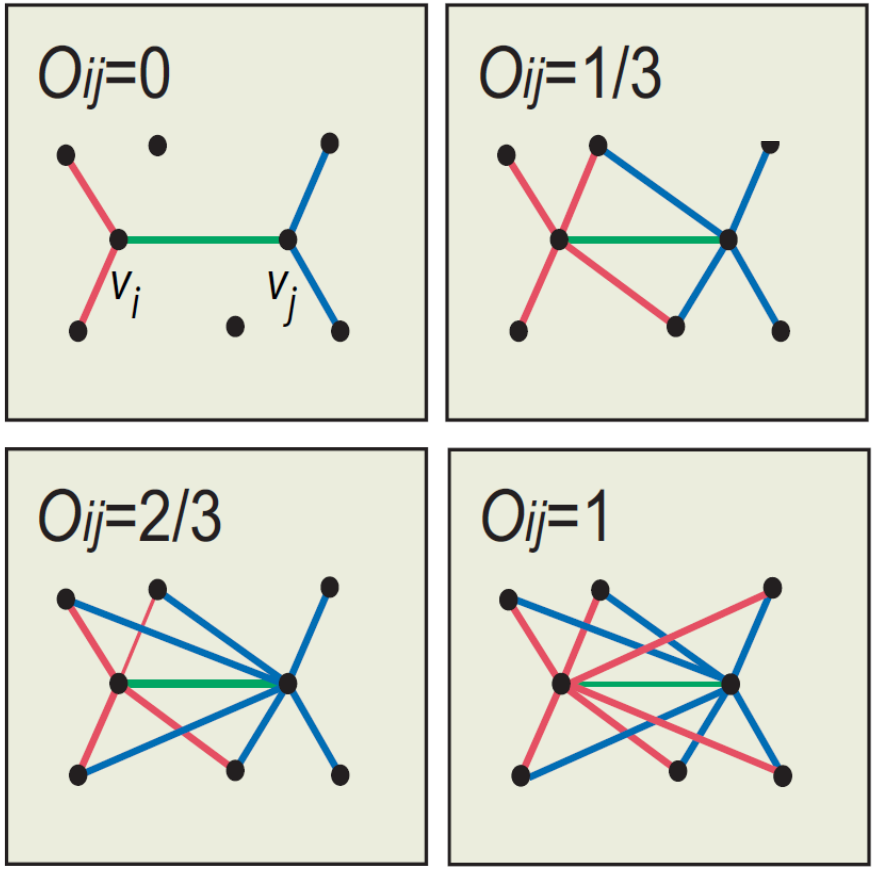
Edge Overlap

- **Edge overlap:**

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

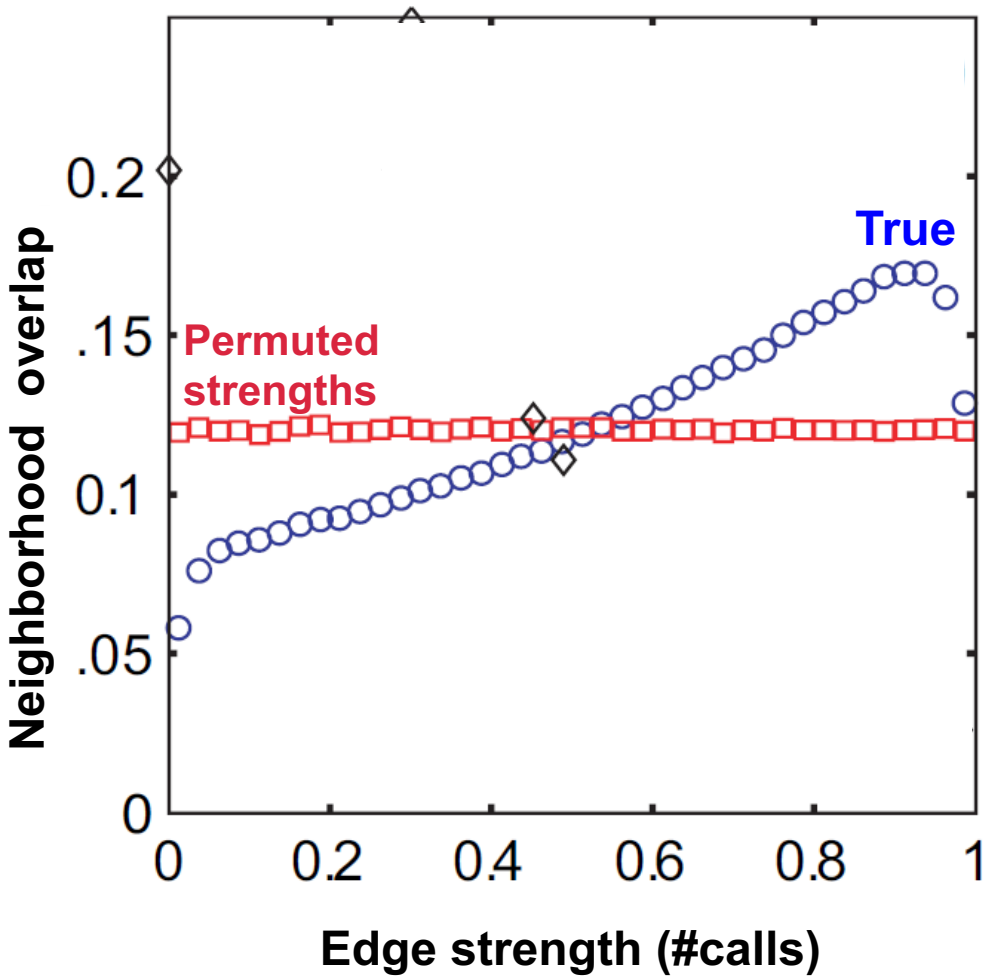
  - $N(i)$ … the set of neighbors of node i

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

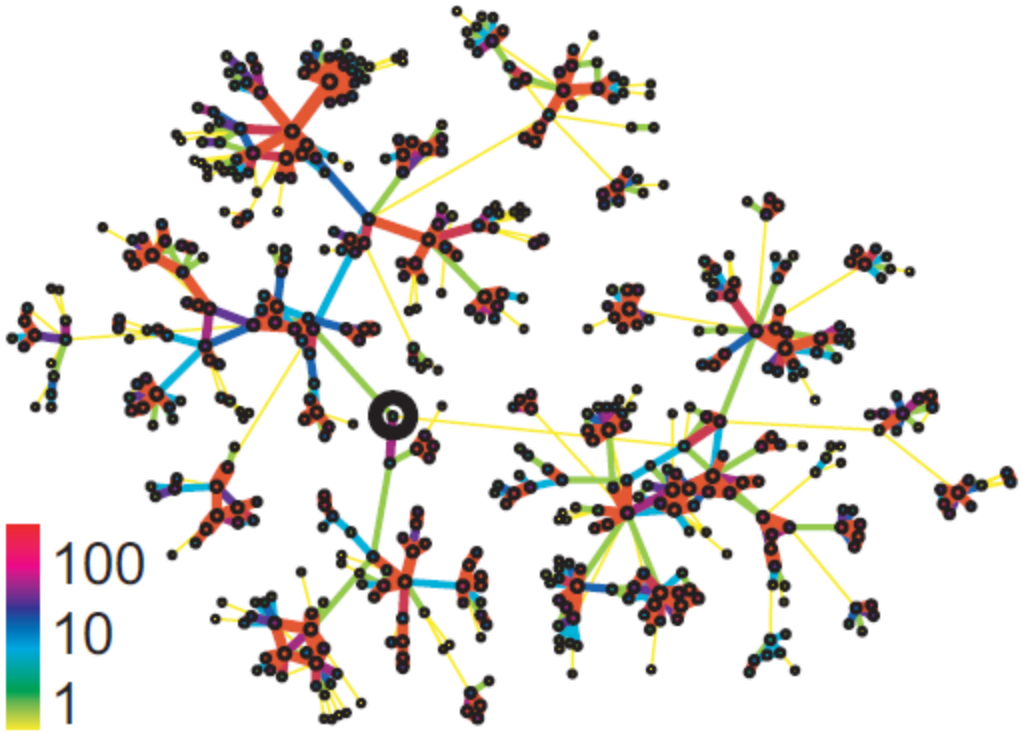


Phones: Edge Overlap vs. Strength

- **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

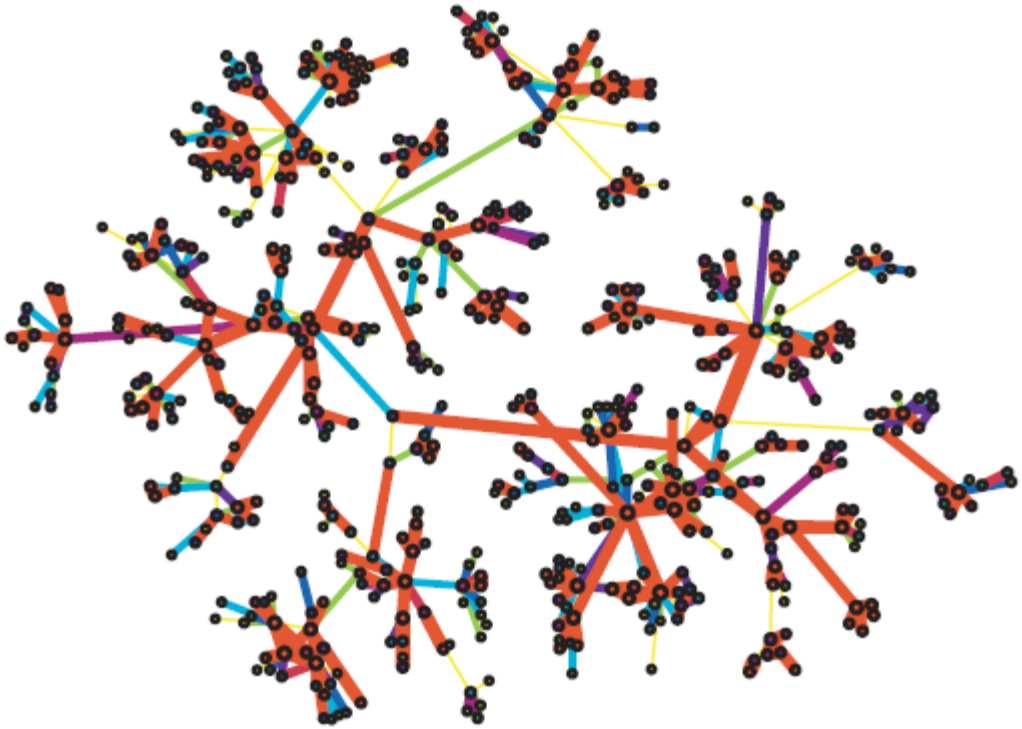


Real Network, Real Edge Strengths



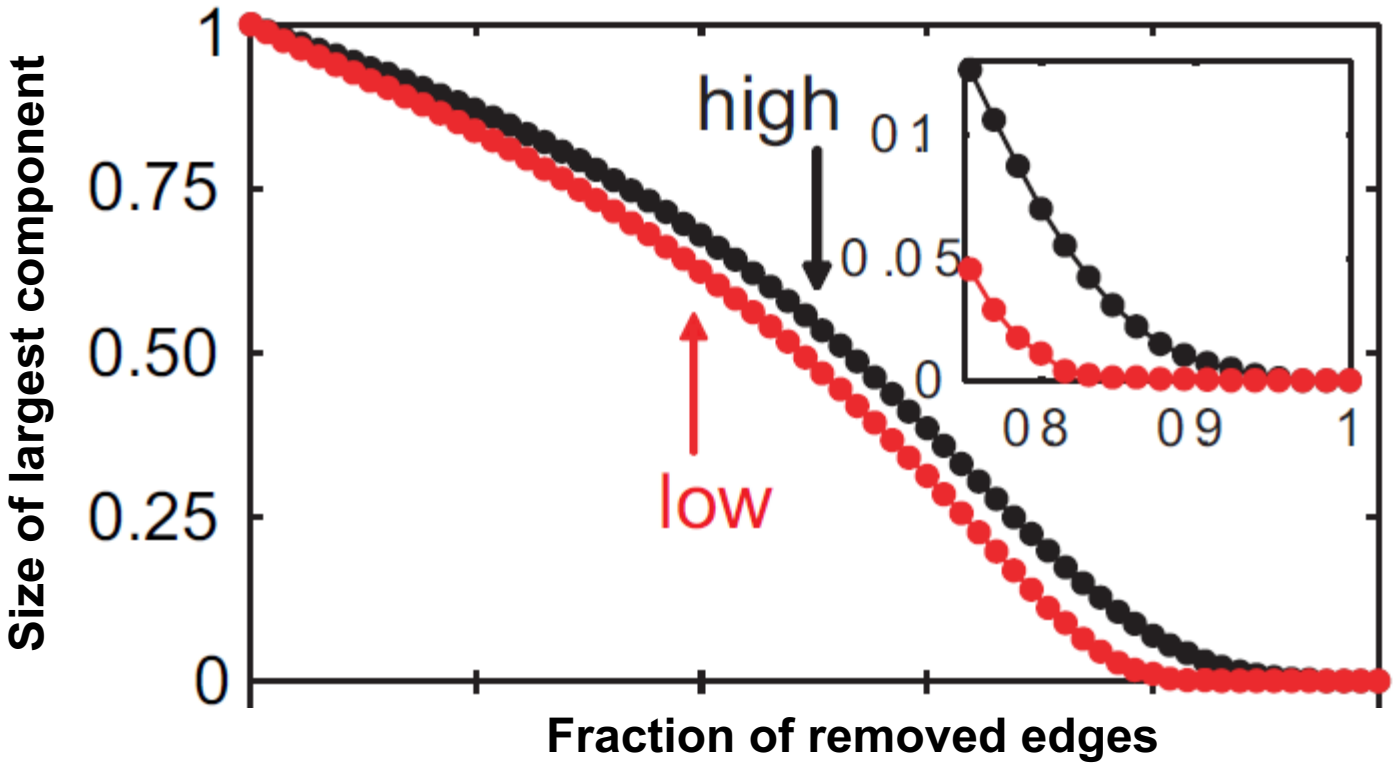
- **Real edge strengths in mobile call graph**
  - Strong ties are more embedded (have higher overlap)

Real Net, Permuted Tie Strengths

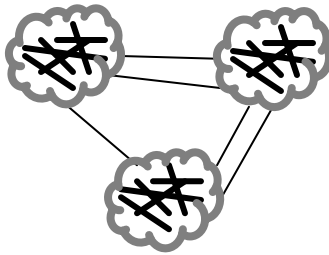


- **Same network, same set of edge strengths** but now **strengths are randomly shuffled**

Edge Removal by Strength



Low disconnects the network sooner

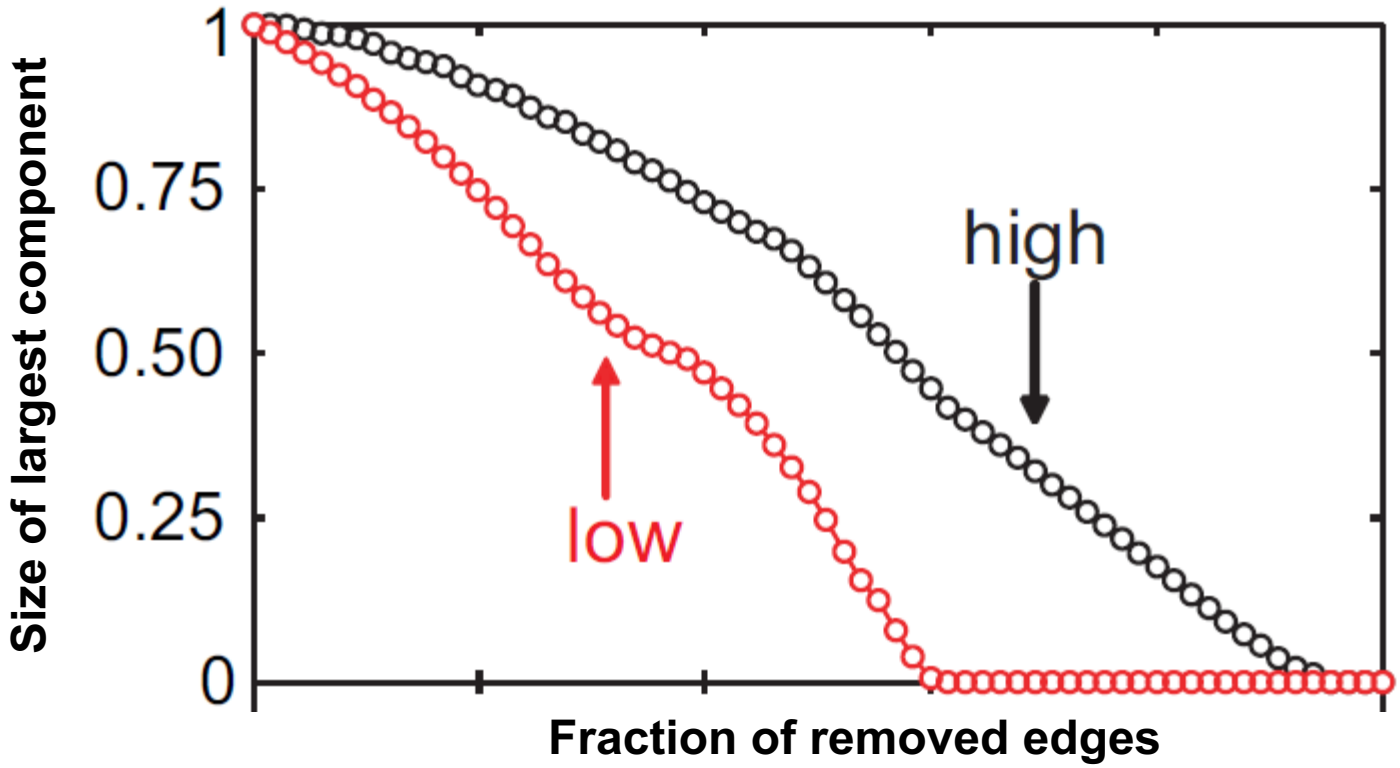


Conceptual picture of network structure

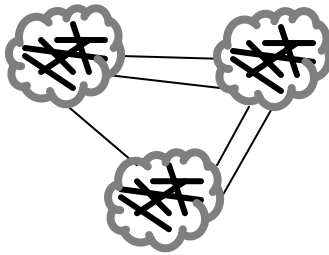
Removing edges based on **strength (#calls)**

- Low to high
- High to low

Link Removal by Overlap



Low disconnects the network sooner



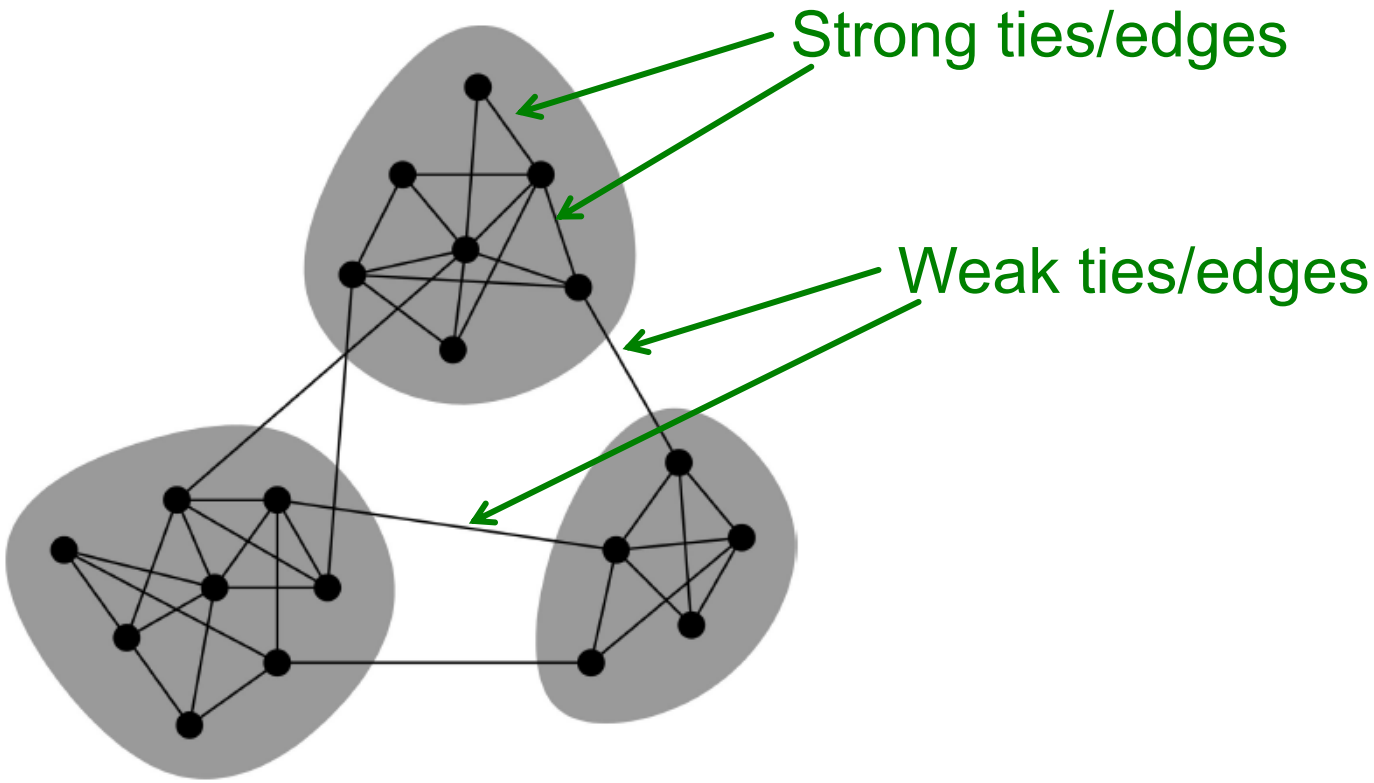
Conceptual picture of network structure

Removing edges based on **edge overlap**

- Low to high
- High to low

Conceptual Picture of Networks

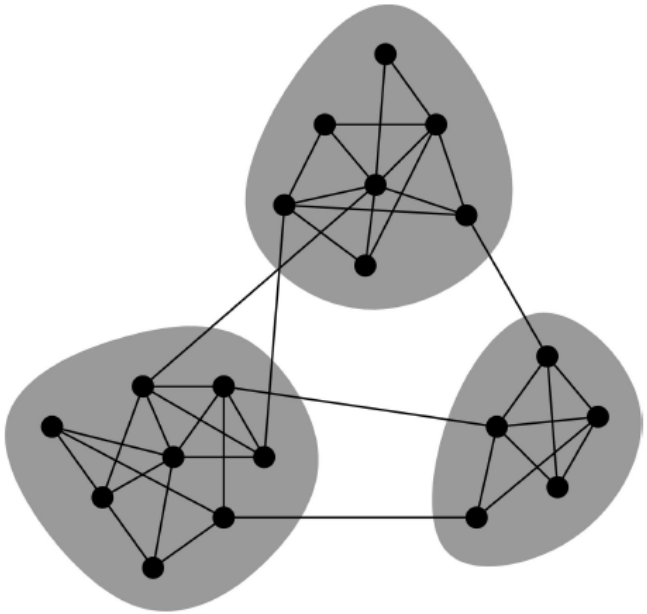
- **Granovetter's theory leads to the following conceptual picture of networks**



Network Communities

Network Communities

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

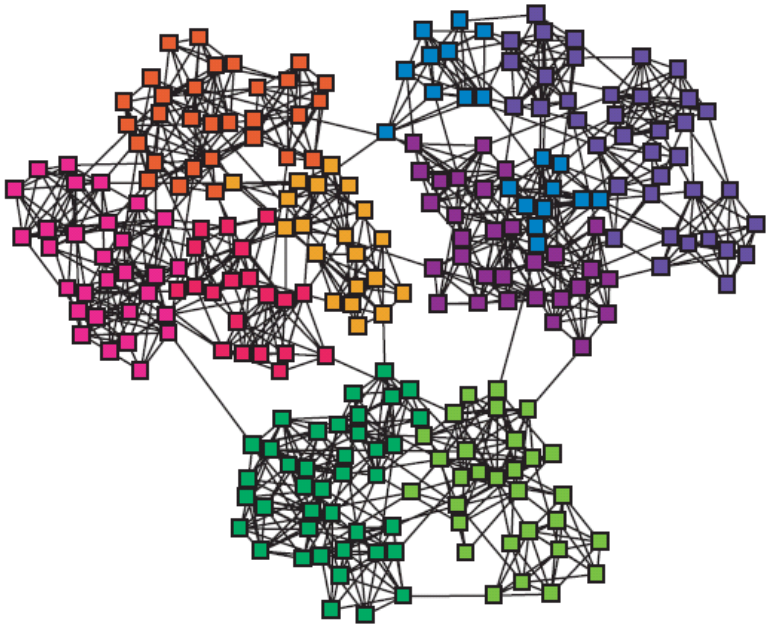


Communities, clusters, groups, modules

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

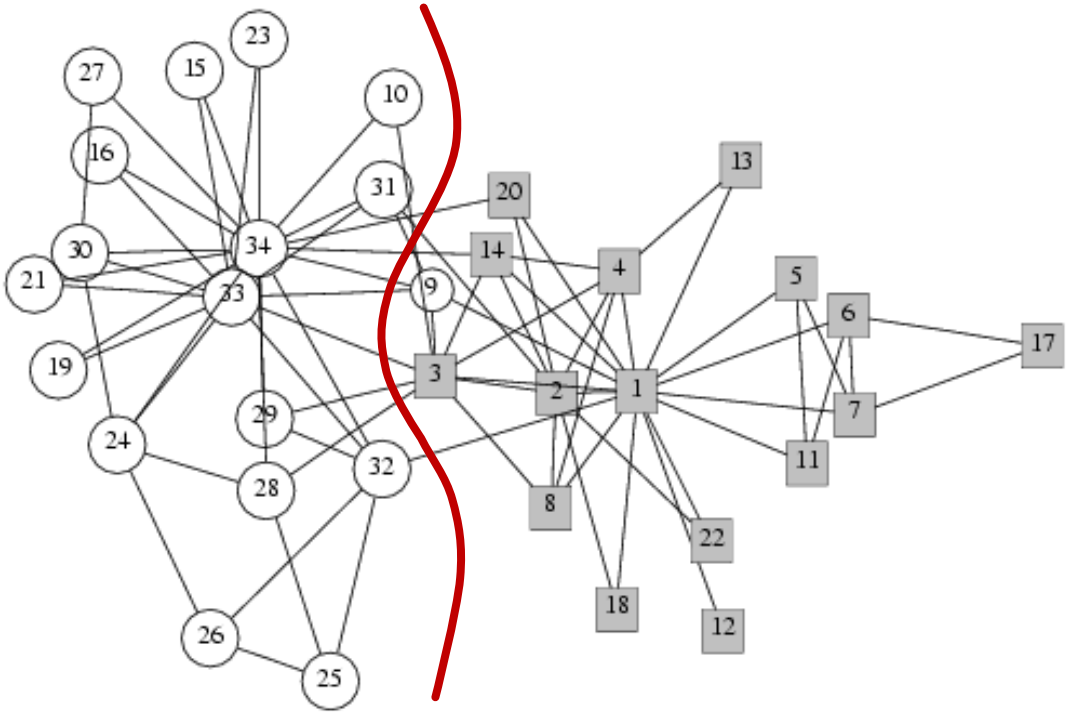
Finding Network Communities

- **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

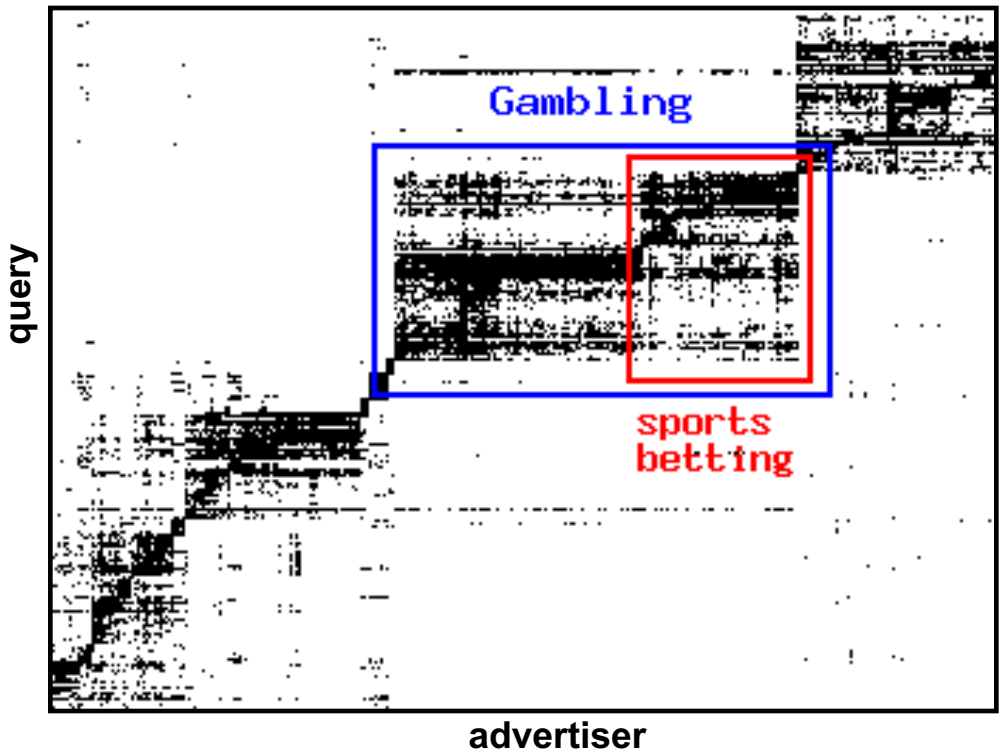
Social Network Data



- **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

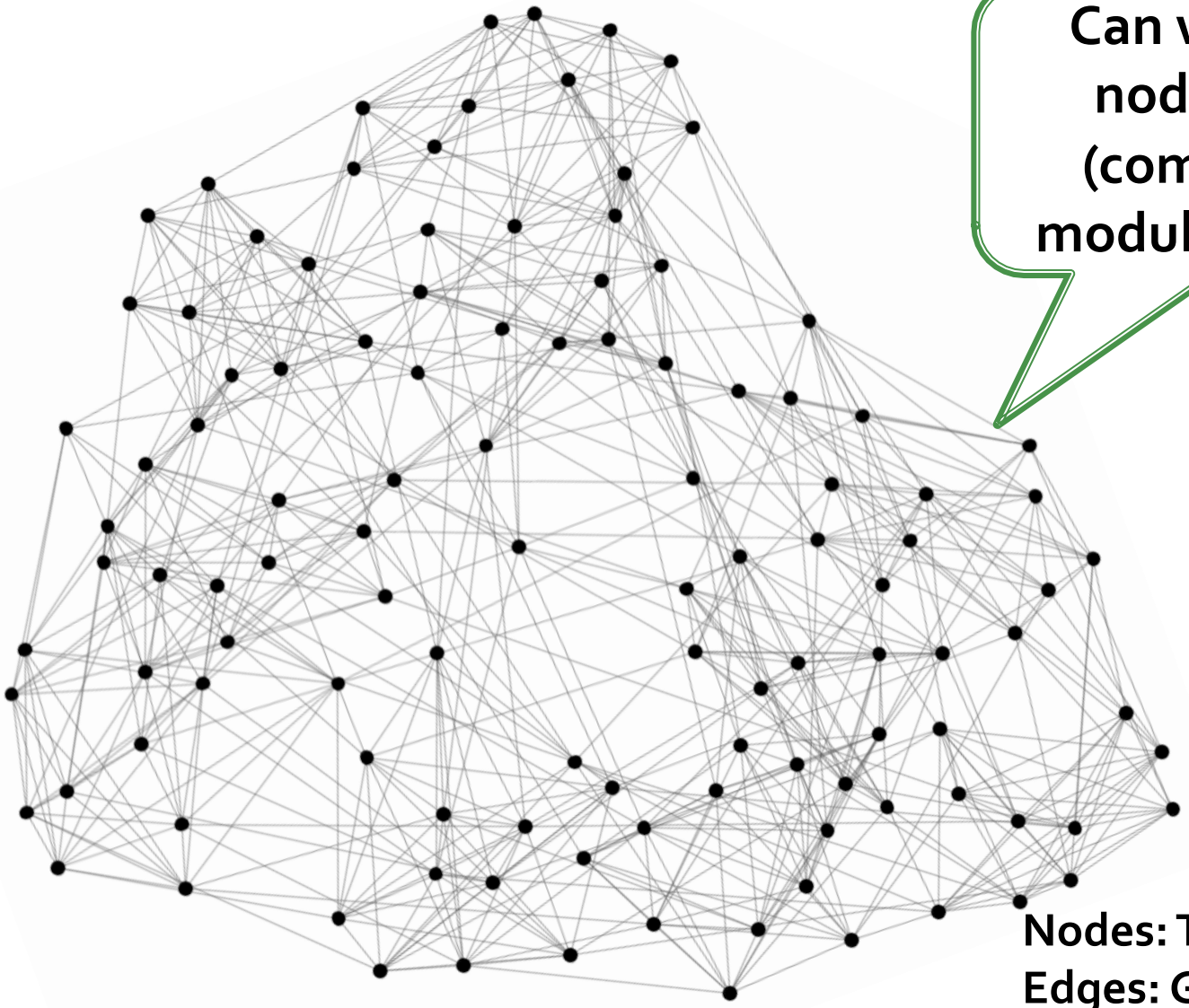
Micro-Markets in Sponsored Search

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



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

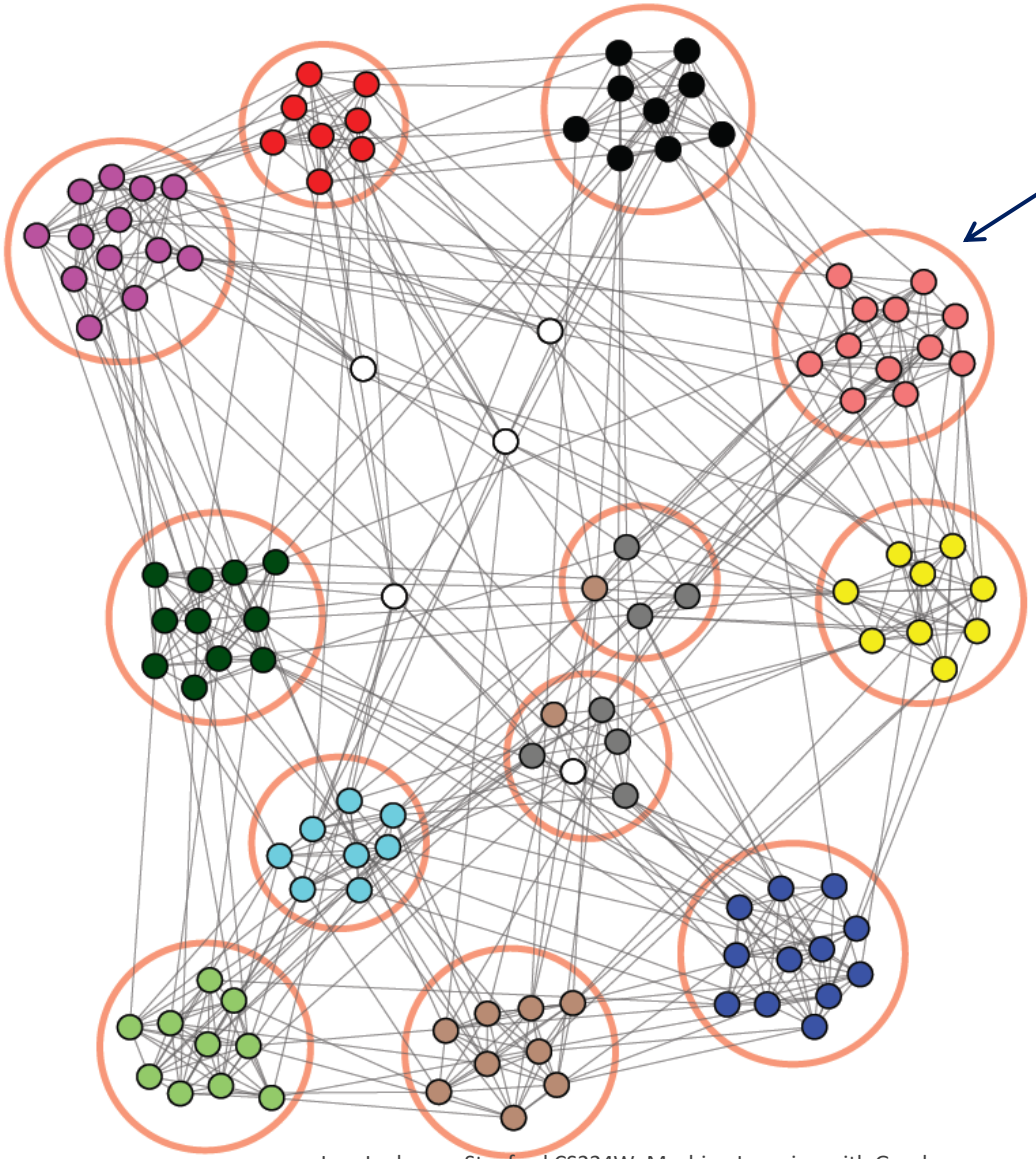
NCAA Football Network



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

Nodes: Teams
Edges: Games played

NCAA Football Network



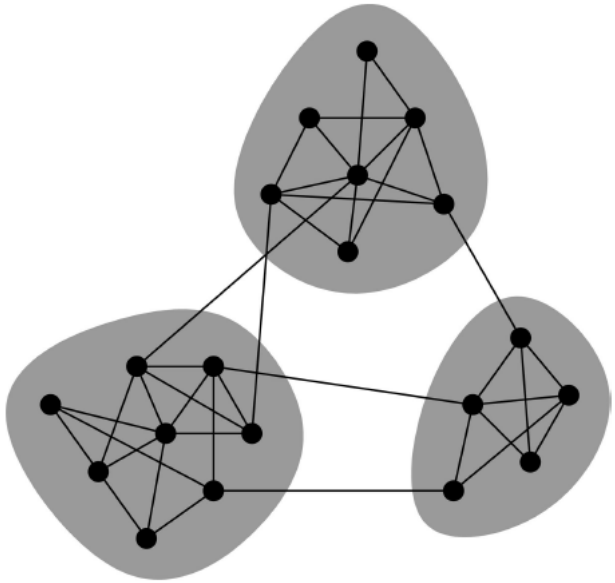
NCAA conferences

- 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 $s \in S$:

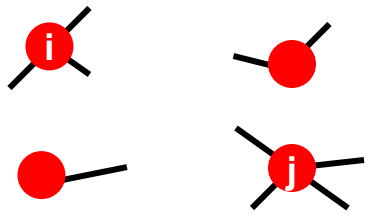


$$Q \propto \sum_{s \in S} [\text{(\# edges within group } s) - \text{(expected \# edges within group } s)]$$

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

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
- **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**

RECAP: Modularity

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

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 $\delta(c_i, c_j) = 1 \; if \; c_i = c_j \; else \; 0$

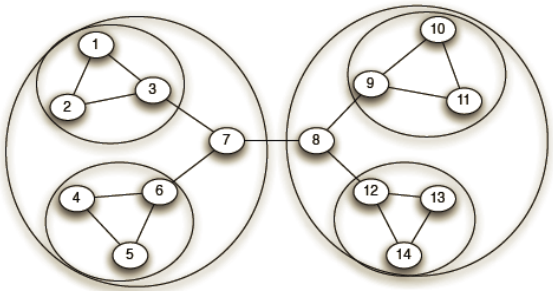
Idea: We can identify communities by maximizing modularity

Louvain Algorithm

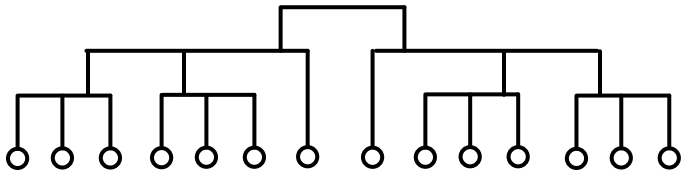
Louvain Algorithm

- **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:

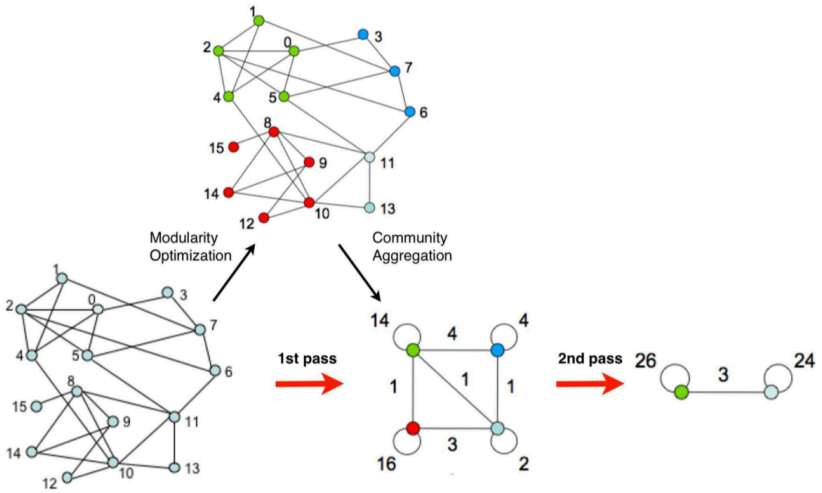


"Fast unfolding of communities in large networks" Blondel et al. (2008)

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
  - **Goto Phase 1**

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



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 we move node i to community C?

$$\Delta Q(i \to C) = \left[\frac{\sum_{in}+k_{i,in}}{2m} - \left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

- where:
  - Σ_{in}… sum of link weights between nodes in C
  - Σ_{tot}… sum of all link weights of nodes in C
  - $k_{i,in}$… sum of link weights between node i and C
  - k_i… sum of all link weights (i.e., degree) of node i

Σ_{in}:

Σ_{tot}:

- Also need to derive $\Delta Q(D \to i)$ of taking node i out of community D.
- And then: $\Delta Q = \Delta Q(i \to C) + \Delta Q(D \to i)$

Louvain: Modularity Gain

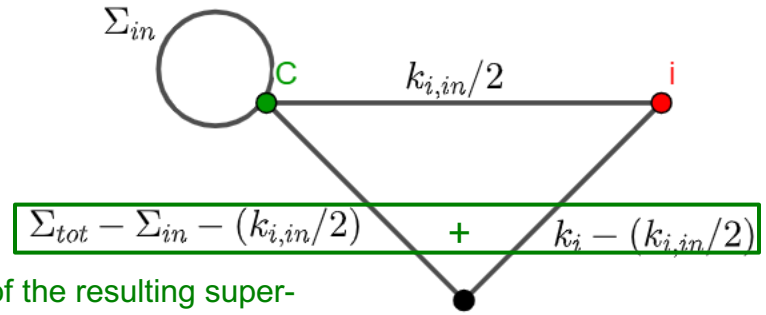
More in detail:

Modularity contribution **after merging** node *i*

Modularity contribution **before merging** node *i*

$$\Delta Q(i \to C) = \left[\frac{\Sigma_{in}+k_{i,in}}{2m} - \left(\frac{\Sigma_{tot}+k_i}{2m}\right)^2\right] - \left[\underbrace{\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2}_{\text{Modularity of C}} - \underbrace{\left(\frac{k_i}{2m}\right)^2}_{\text{Modularity of } i}\right]$$

Self-edge weight



Edge weight of the resulting super-node from merging C and *i*

rest of the graph (modeled as a single node)

By applying the Modularity definition:

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

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 Algorithm

Algorithm 1: Sequential Louvain Algorithm

Input: $G=(V,E)$: graph representation.
Output: C: community sets at each level;
Q: modularity at each level.
Var: $\hat{c}$: vertex u's best candidate community set.

1 **Loop** ***outer***
2 $C \leftarrow \{\{u\}\}, \forall u \in V$;
3 $\Sigma_{in}^{c} \leftarrow \sum w_{u,v},\ e(u,v) \in E,\ u \in c$ and $v \in c$;
4 $\Sigma_{tot}^{c} \leftarrow \sum w_{u,v},\ e(u,v) \in E,\ u \in c$ or $v \in c$;
5 *// Phase 1.*
6 **Loop** ***inner***
7 **for** $u \in V$ *and* $u \in c$ **do**
8 *// Find the best community for vertex u.*
9 $\hat{c} \leftarrow \underset{\forall c',\ \exists e(u,v) \in E,\ v \in c'}{\text{argmax}} \Delta Q_{u \to c'}$; Modularity gain
10 **if** $\Delta Q_{u \to \hat{c}} > 0$ **then**
11 *// Update* Σ_{tot} *and* Σ_{in}.
12 $\Sigma_{tot}^{\hat{c}} \leftarrow \Sigma_{tot}^{\hat{c}} + w(u)$; $\Sigma_{in}^{\hat{c}} \leftarrow \Sigma_{in}^{\hat{c}} + w_{u \to \hat{c}}$;
13 $\Sigma_{tot}^{c} \leftarrow \Sigma_{tot}^{c} - w(u)$; $\Sigma_{in}^{c} \leftarrow \Sigma_{in}^{c} - w_{u \to c}$;
14 *// Update the community information.*
15 $\hat{c} \leftarrow \hat{c} \cup \{u\}$; $c \leftarrow c - \{u\}$;
16 **if** *No vertex moves to a new community* **then**
17 **exit** ***inner*** **Loop**;

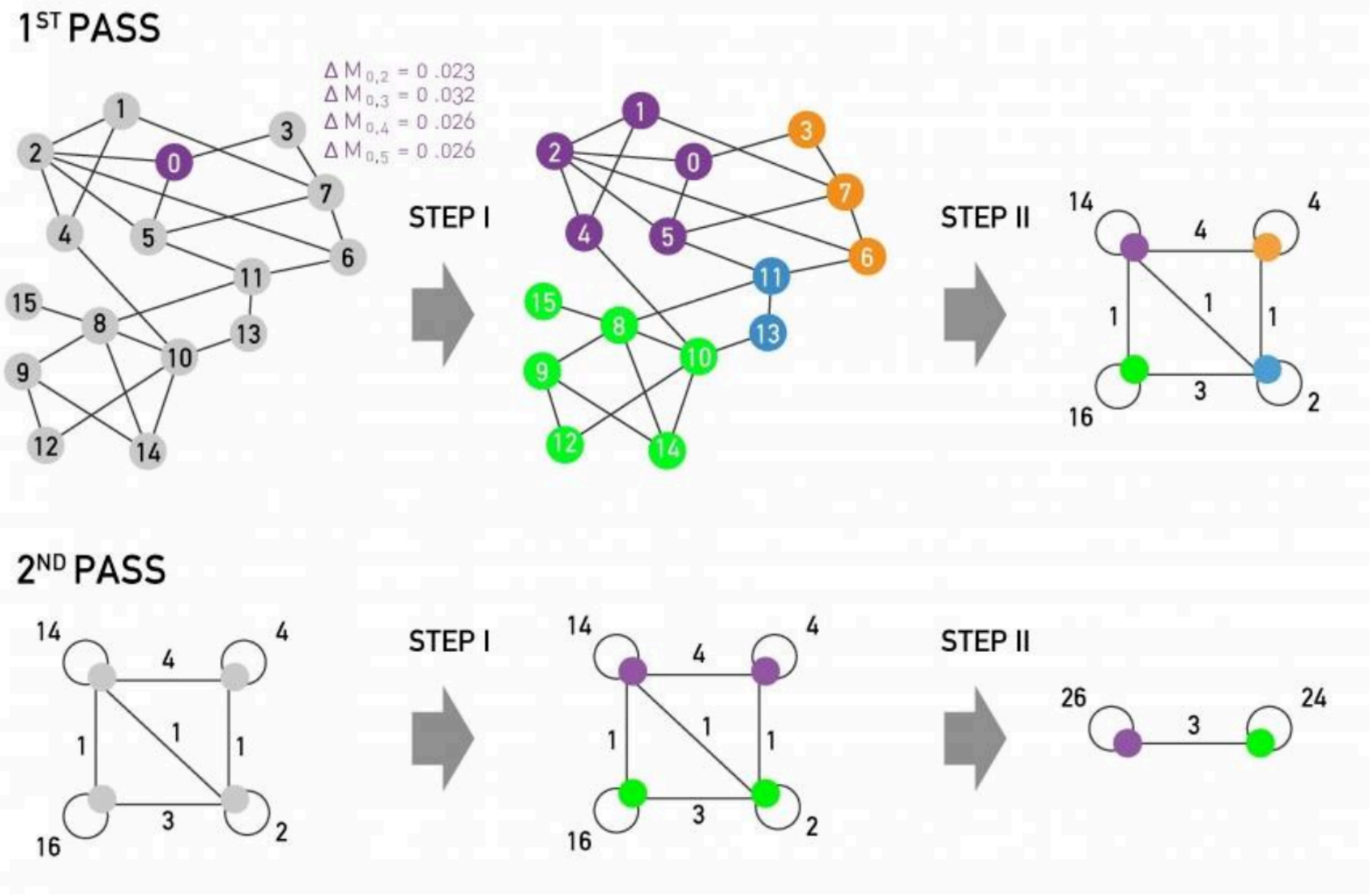
Halting criterion for 1st Phase

18 *// Calculate community set and modularity.*
19 $Q \leftarrow 0$;
20 **for** $c \in C$ **do**
21 $Q \leftarrow Q + \frac{\Sigma_{in}^{c}}{2m} - (\frac{\Sigma_{tot}^{c}}{2m})^2$;
22 $C' \leftarrow \{c\}, \forall c \in C$; print C' and Q ;
23 *// Phase 2: Rebuild Graph.*
24 $V' \leftarrow C'$; Communities contracted into super-nodes
25 $E' \leftarrow \{e(c,c')\},\ \exists e(u,v) \in E,\ u \in c,\ v \in c'$;
26 $w_{c,c'} \leftarrow \sum w_{u,v},\ \forall e(u,v) \in E,\ u \in c,\ v \in c'$;
27 **if** *No community changes* **then**
28 **exit** ***outer*** **Loop**;
29 $V \leftarrow V'$; $E \leftarrow E'$;

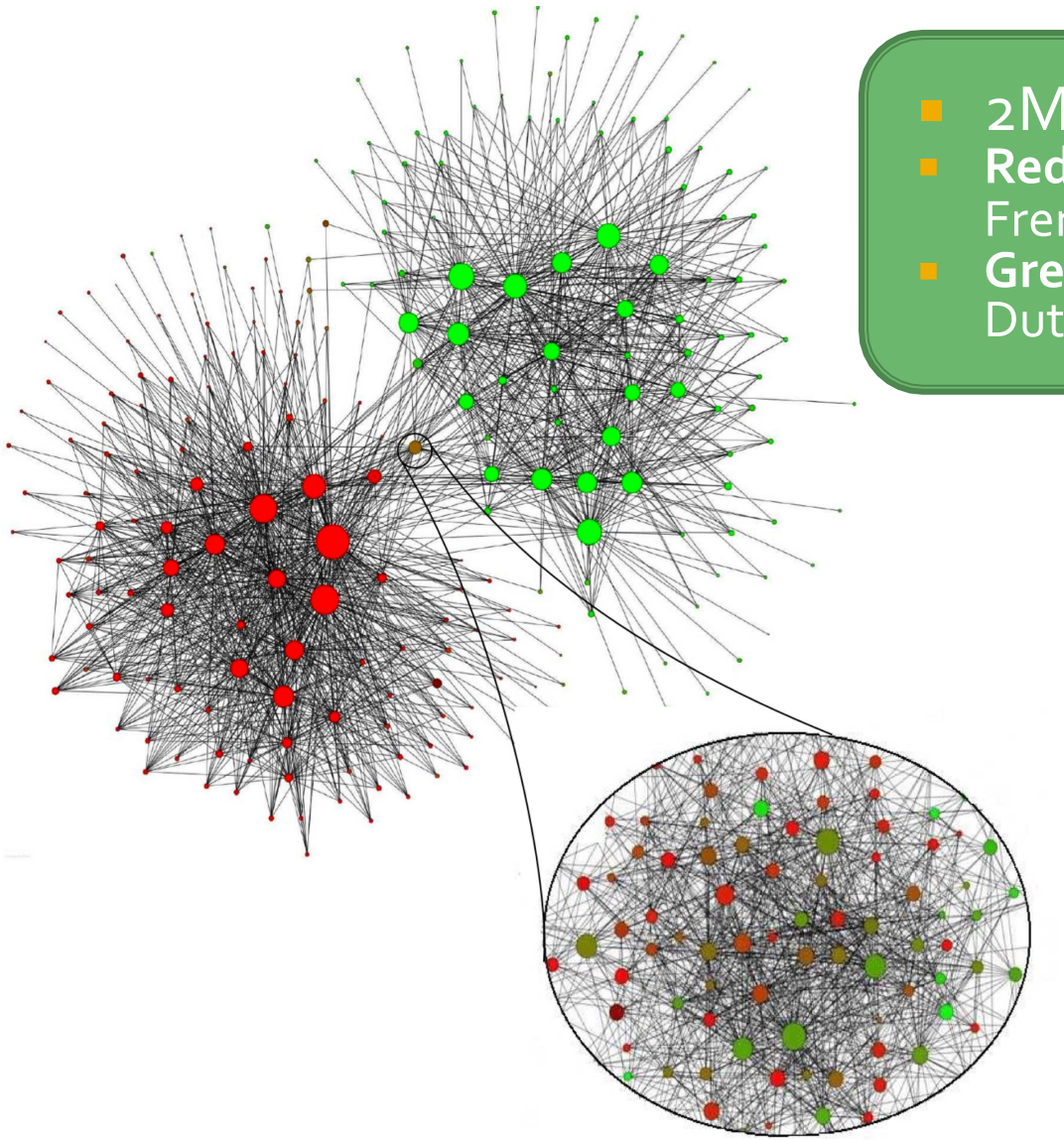
Halting criterion for 2nd Phase

the weights of the edges between the new super-nodes are given by the **sum of the weights of the edges** between vertices in the corresponding two communities

Louvain Algorithm



Belgian Mobile phone network



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

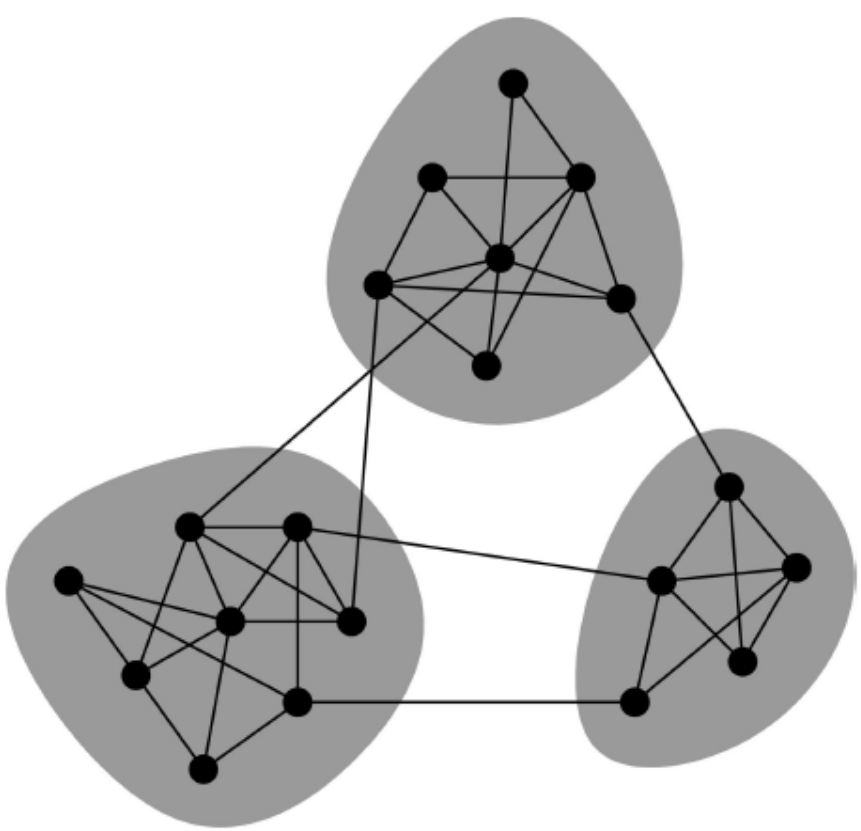
Summary: Modularity

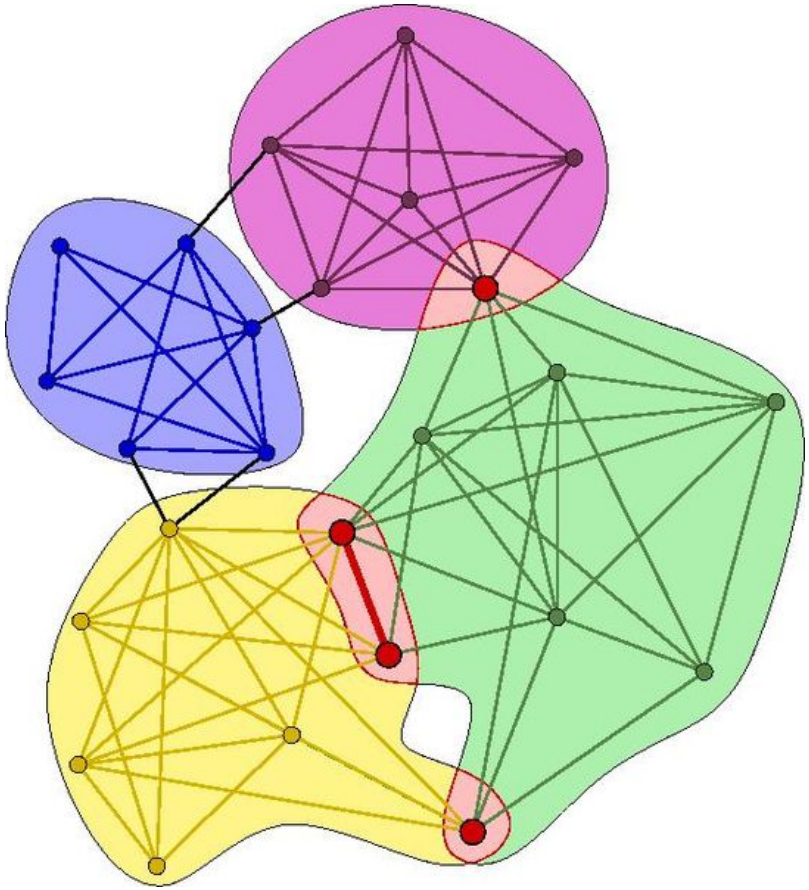
- **Modularity:**
  - Overall quality of the partitioning of a graph into communities
  - Used to determine the number of communities
- **Louvain modularity maximization:**
  - Greedy strategy
  - Great performance, scales to large networks

Detecting Overlapping Communities: BigCLAM

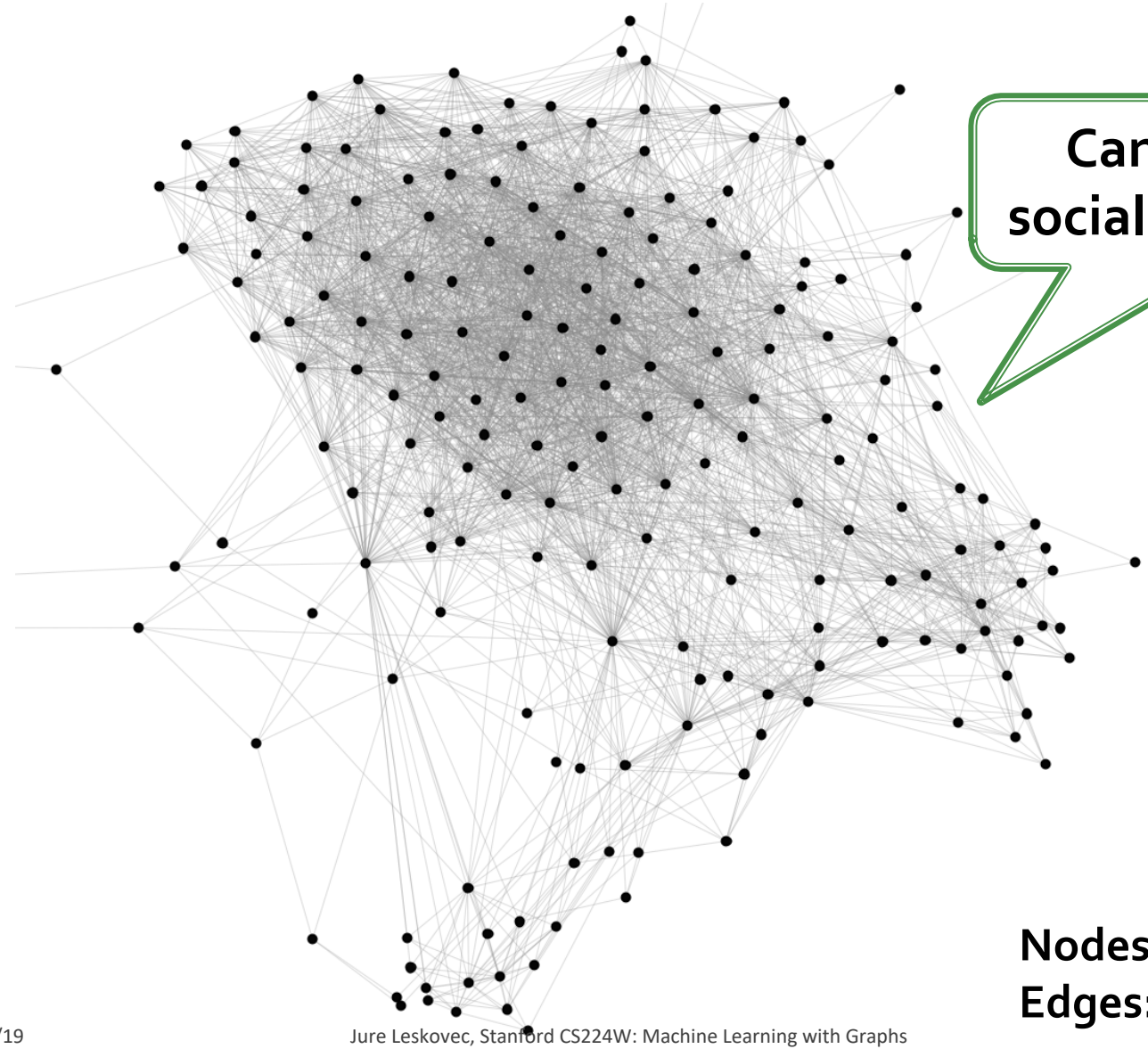
Overlapping Communities

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





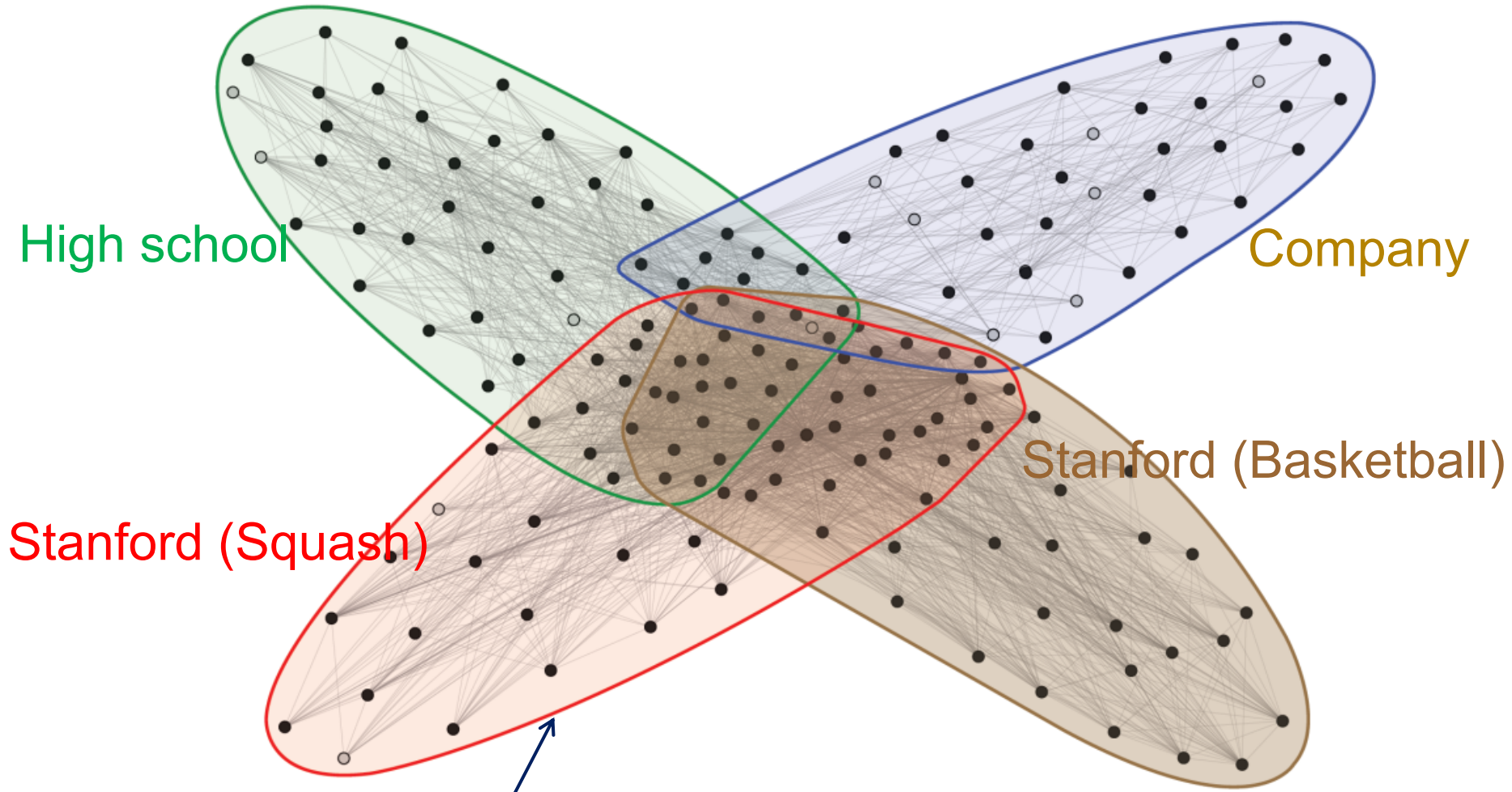
Facebook Ego-network



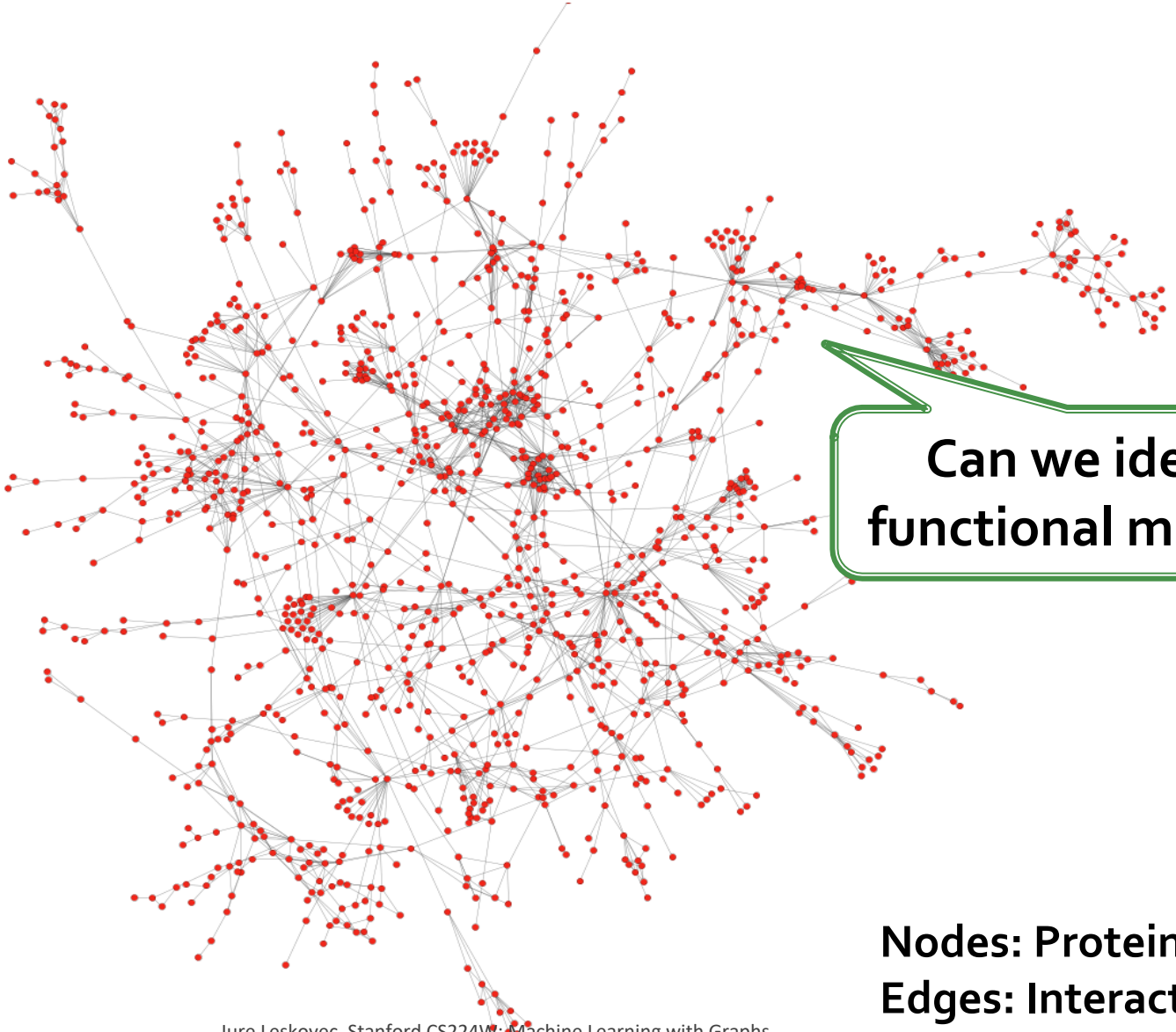
Can we identify social communities?

Nodes: Users
Edges: Friendships

Facebook Ego-network



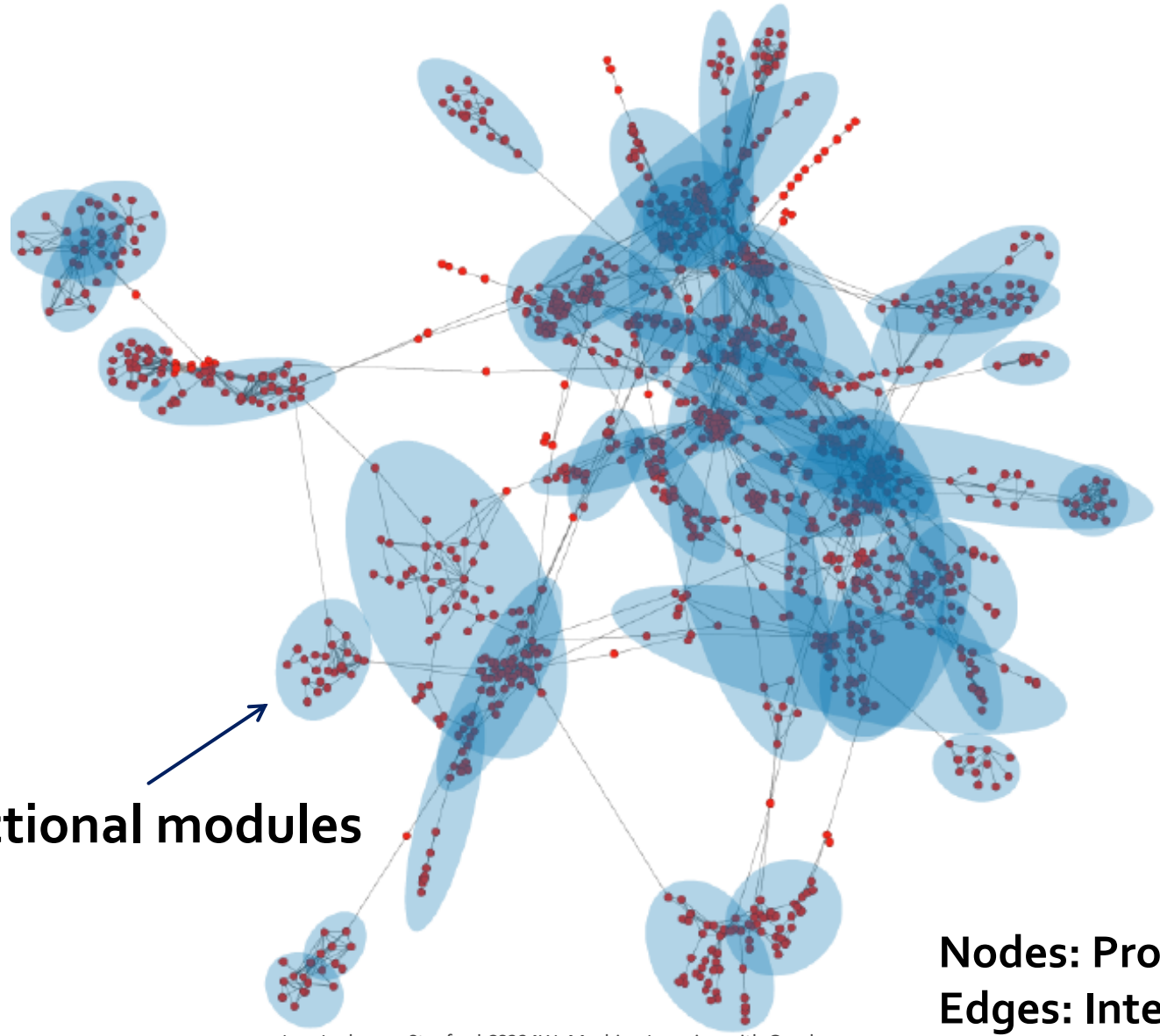
Protein-Protein Interactions



Can we identify functional modules?

Nodes: Proteins
Edges: Interactions

Protein-Protein Interactions

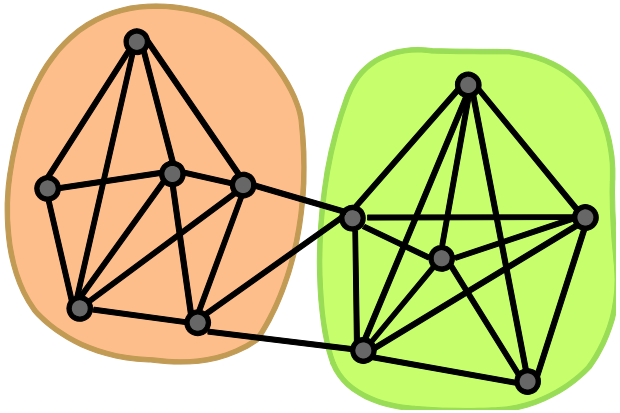


Functional modules

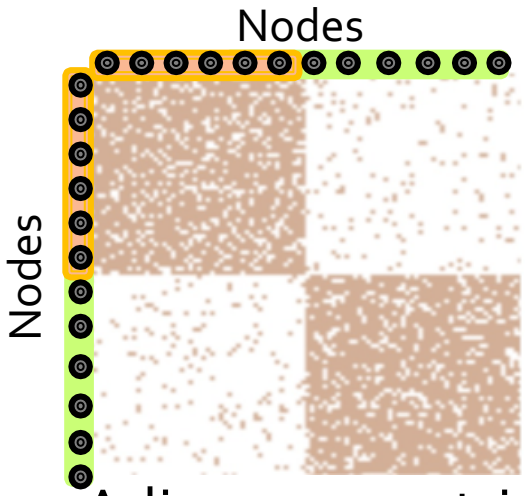
Nodes: Proteins
Edges: Interactions

Communities

Non-overlapping

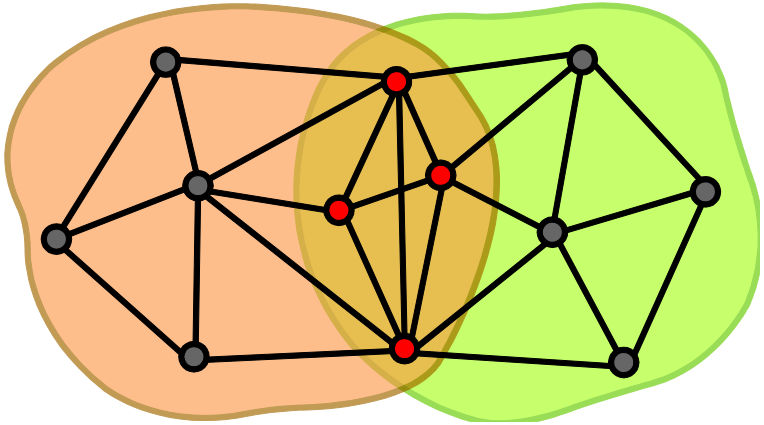


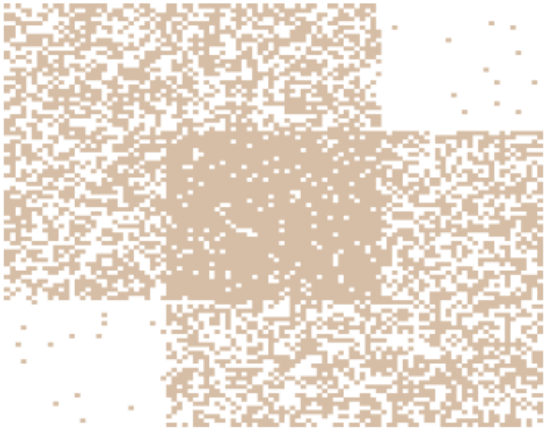
Network



Adjacency matrix

Overlapping





Plan of Action

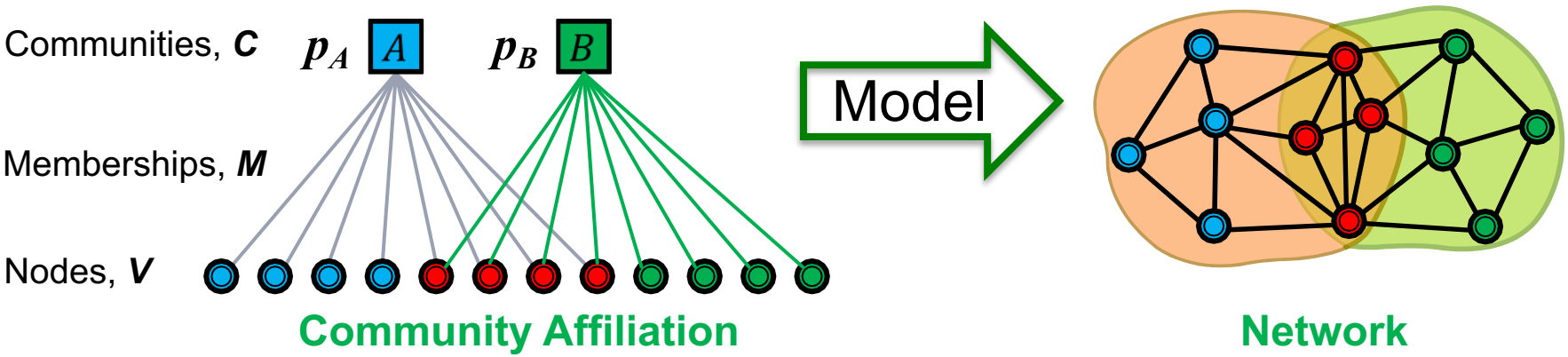
Step 1)

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

Step 2)

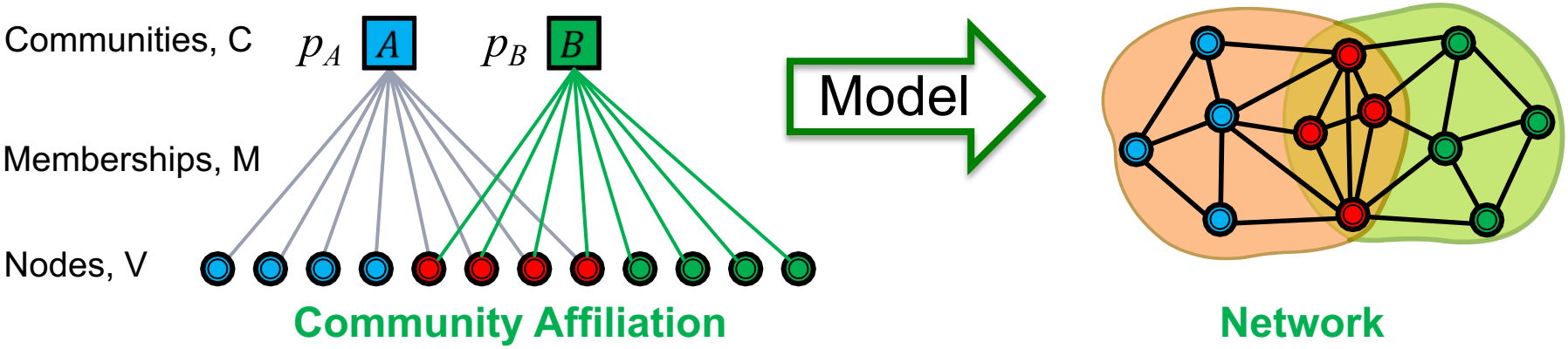
- Given graph G, make the assumption that G was generated by AGM
- Find the best AGM that could have generated G
- **And this way we discover communities**

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

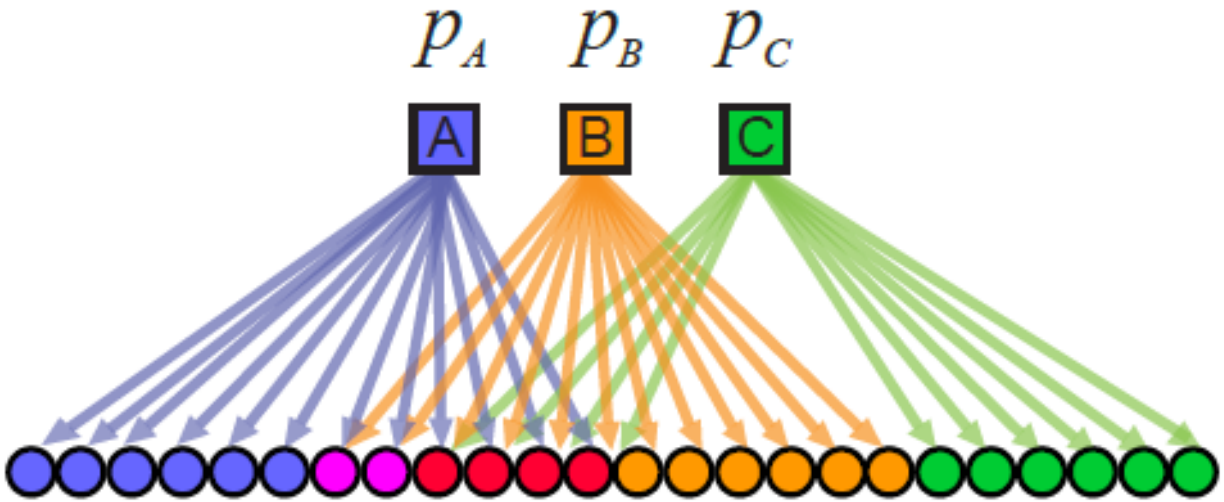


- **Given parameters (V, C, M, $\{p_c\}$)**
  - Nodes in community c connect to each other by flipping a coin with probability p_c
  - **Nodes that belong to multiple communities have multiple coin flips**
    - If they "miss" the first time, they get another chance through the next community

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

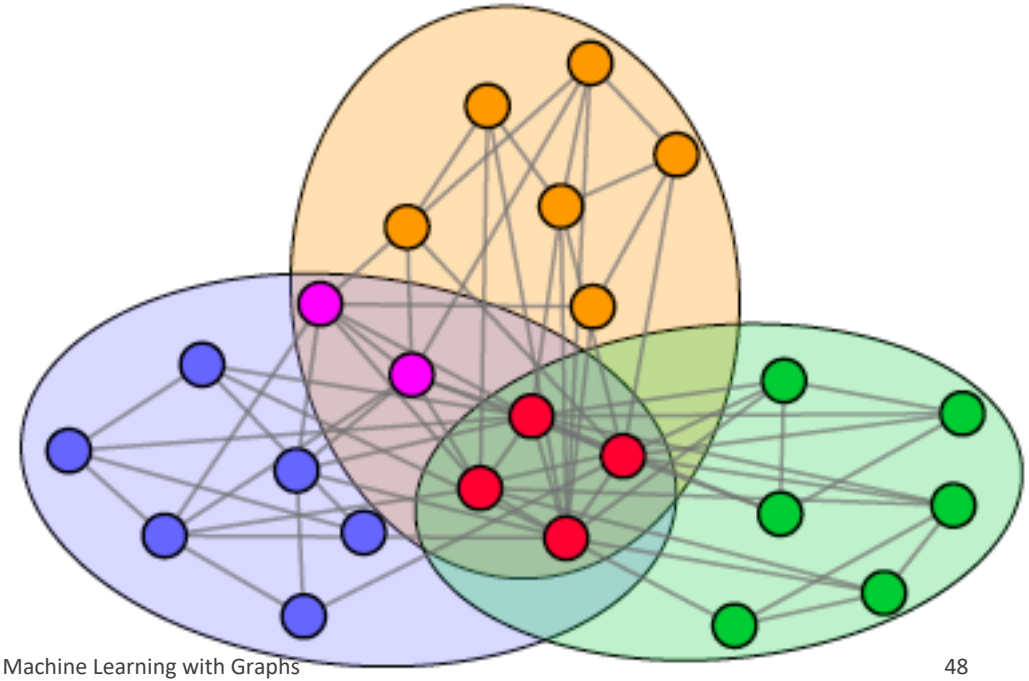
Note: If nodes u and v have no communities in common, then p(u,v)=0. We resolve this by having a background "epsilon" community that every node is a member of.

AGM: Dense Overlaps



Model

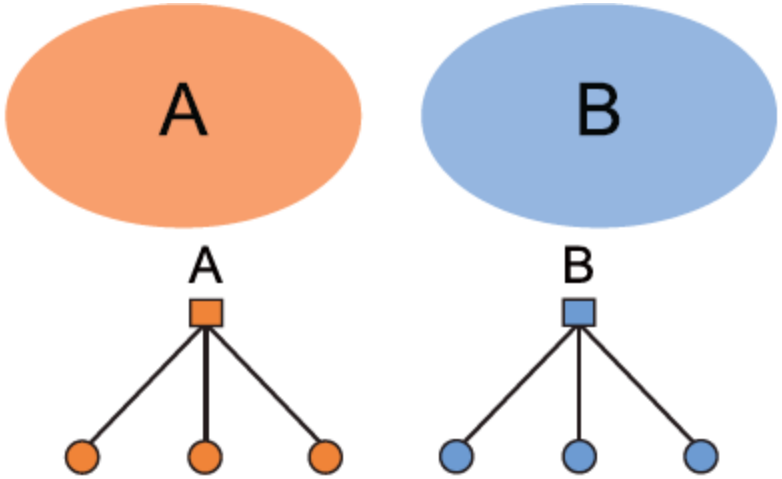
Network

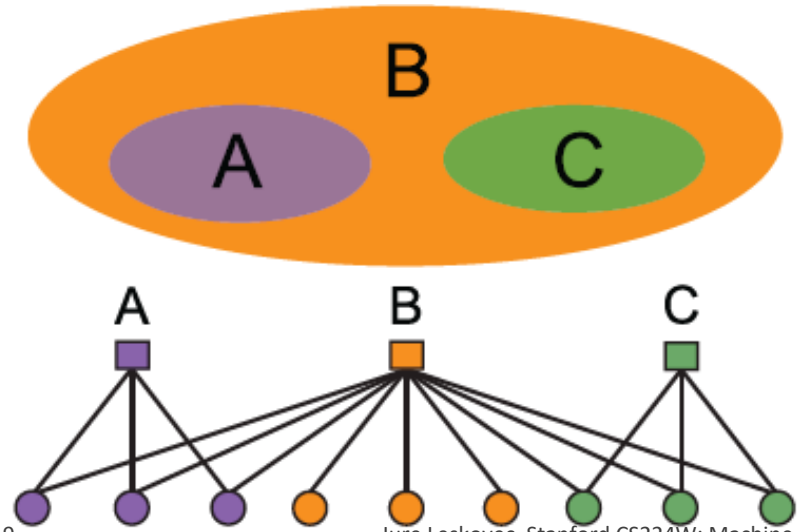


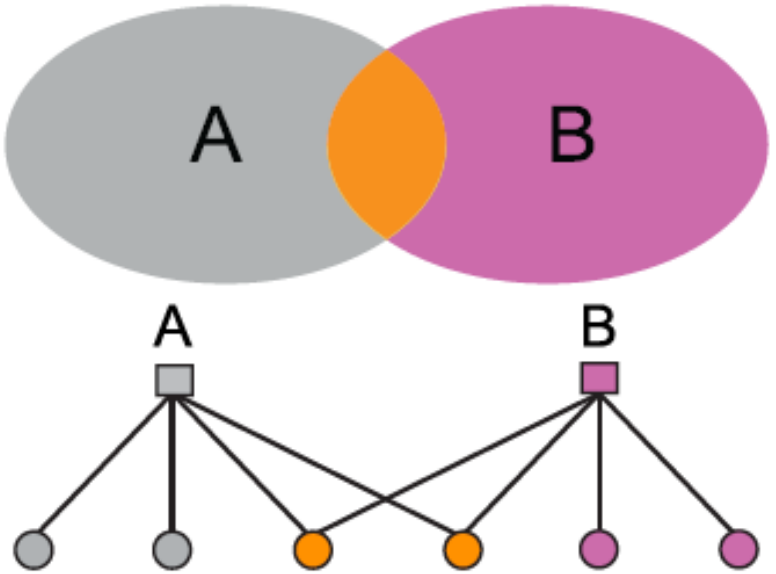
AGM: Flexibility

[icdm '12]

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

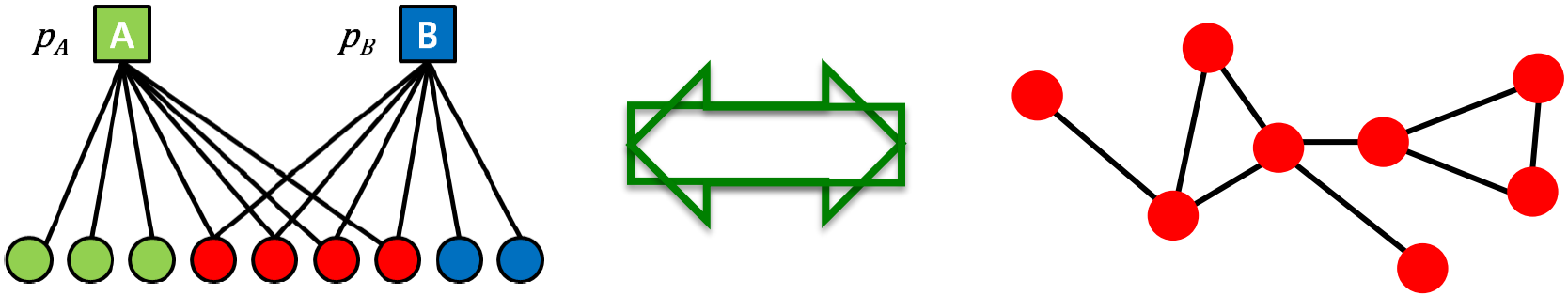






Detecting Communities

- **Detecting communities with AGM:**



Given a Graph, find the model F

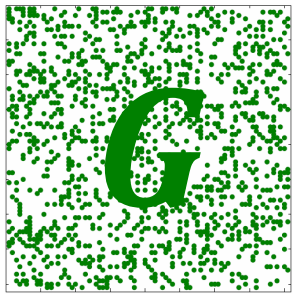
1) Affiliation graph ***M***
2) Number of communities **C**
3) Parameters $\boldsymbol{p_c}$

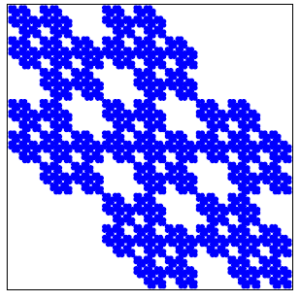
Graph Fitting

How to estimate model parameters F given a G?

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

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

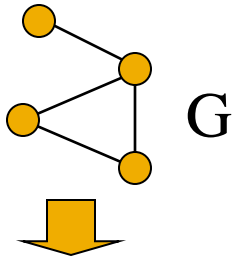




- **To solve this we need to:**
  - Efficiently calculate $\boldsymbol{P(G|F)}$
  - Then maximize over $\boldsymbol{F}$ (e.g., using gradient descent)

Graph Likelihood $P(G|F)$

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



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

$P(u,v)$: Edge prob. of edge (u,v)

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

G

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

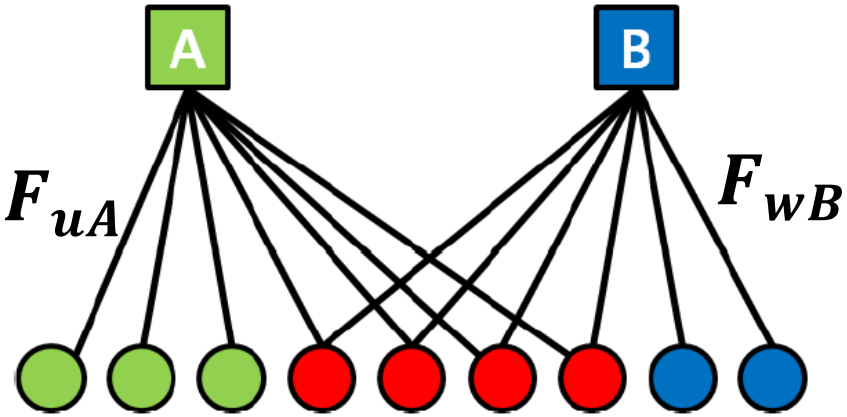
$$P(G|F) = \prod_{(u,v)\in G} P(u,v) \prod_{(u,v)\notin G} (1 - P(u,v))$$

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 A ($F_{uA} = 0$: no membership)
- F_u: A row vector of community memberships of node u

BigCLAM Model

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

$$P(u, v) = 1 - \exp(-F_u \cdot F_v^T)$$

- **Given a network $G(V, E)$, we maximize $l(F)$**

$$l(F) = \sum_{(u,v)\in E} \log(1 - \exp(-\underbrace{F_u F_v^T}_{\text{Dot product}})) - \sum_{(u,v)\notin E} F_u F_v^T$$

This is log-likelihood of network G – total probability of all edges occurring and all non-edges not occurring.

- **Optimization:**
  - Start with random F
  - Update F_{uC} for node u while fixing the memberships of all other nodes
  - Updating takes linear time in the degree of u

BigCLAM Model

- **Gradient ascent:**

$$\nabla l(F_u) = \sum_{v \in \mathcal{N}(u)} F_v \frac{\exp(-F_u F_v^T)}{1 - \exp(-F_u F_v^T)} - \sum_{v \notin \mathcal{N}(u)} F_v$$

  - Perform gradient ascent, where we make small changes to F that lead to increase in log-likelihood
- Pure gradient ascent is slow! **However:**

$$\sum_{v \notin \mathcal{N}(u)} F_v = (\sum_{v} F_v - F_u - \sum_{v \in \mathcal{N}(u)} F_v)$$

  - By caching $\boldsymbol{F_v}$ the gradient step takes **linear time** in the degree of $\boldsymbol{u}$

Information About the Course Project

Announcement: Course Project

- **Project is a substantial part of the class**
  - Students put significant effort, and great results have been obtained in the past
- **Types of projects:**
  - **Empirical analysis** of network data to develop a model of behavior
  - **Algorithms and models** to make predictions on a network dataset
  - **Scalable algorithms** for massive graphs
    - Fast algorithms for big graphs. Can be integrated into SNAP.
  - **Theoretical project** that considers a model/algorithm and derives a rigorous result about it
- **Other points:**
  - The project should contain some mathematical analysis, and some experimentation on real or synthetic data
  - The result of the project will typically be an 8 page paper, describing the approach, the results, and related work.
  - **Come to us if you need help with a project idea!**

Announcement: Project Proposal

Project proposal: 3-4 pages, teams of up to 3 students

- **Project proposal has 3 parts:**
  - **(0) Quick 200 word abstract**
  - **(1) Related work / Reaction paper (1-2 pages):**
    - **Read 3 papers related to the project/class**
    - Do reading beyond what was covered in class
    - Think beyond what you read. Don't take other's work for granted!
    - **1-2 pages:** Summary (~1 page), Critique (~1 page)
  - **(2) Proposal (1-2 pages):**
    - Clearly define the problem you are solving.
    - **How does it relate to what you read for the Reaction paper?**
    - What data will you use? (**make sure you already have it!**)
    - Which algorithm/model will you use/develop? **Be specific!**
    - **How will you measure success (evaluate/test your method)?**

See http://cs224w.stanford.edu/info.html for detailed instructions and examples of previous proposals

Announcement: Project Proposal

- **Logistics:**
  - **1) Register your group on this Google Form:** https://forms.gle/kZ1zWP1Y3tnRzzJM6
  - **2) Submit** PDF on GradeScope
  - **Due on Thu Oct 17 at 11:59pm PT!**

- **If you need help/ideas/advice, Michele's Office Hours are exclusively focused on projects**