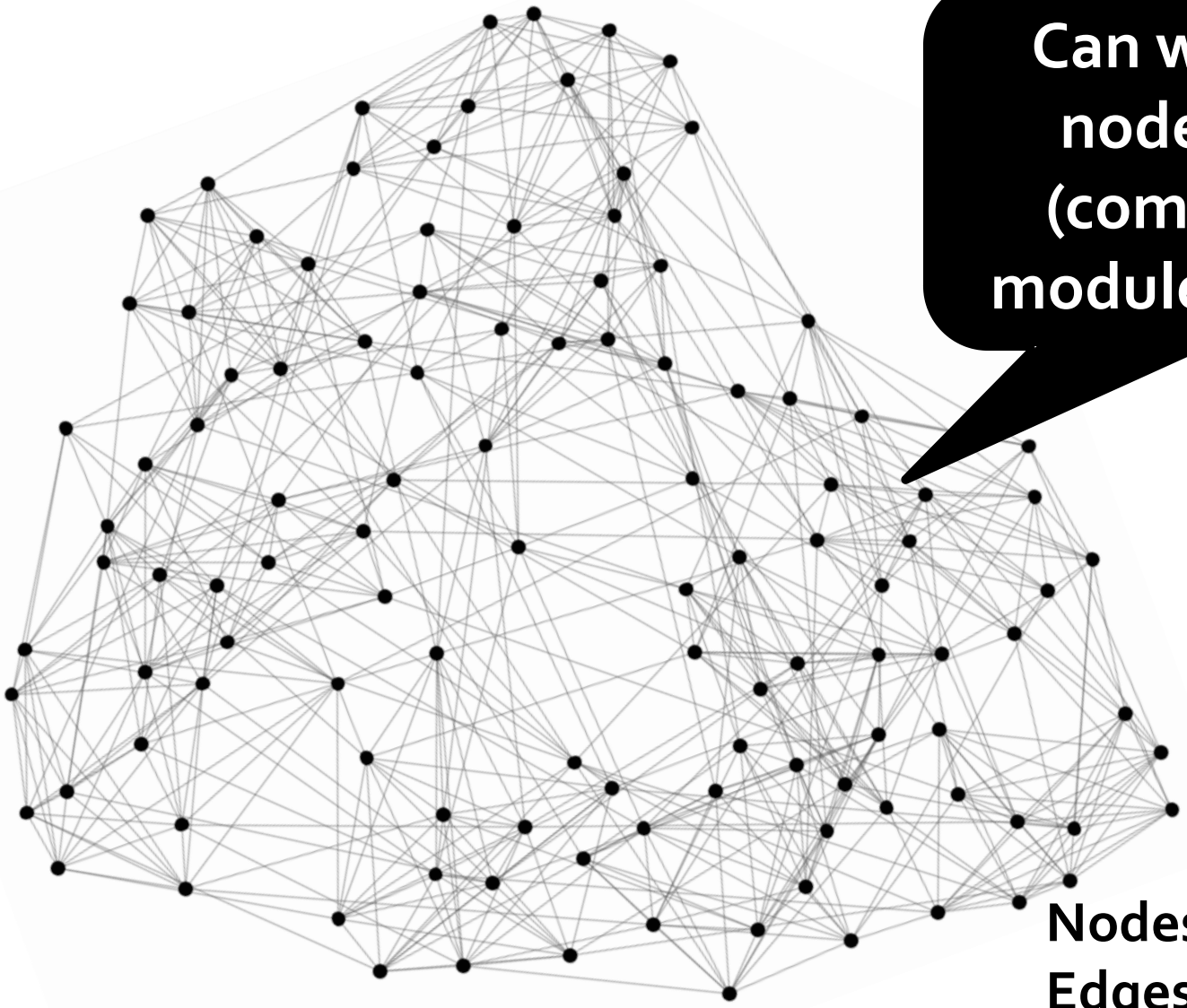
Community Detection in Graphs: Finding overlaps

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu

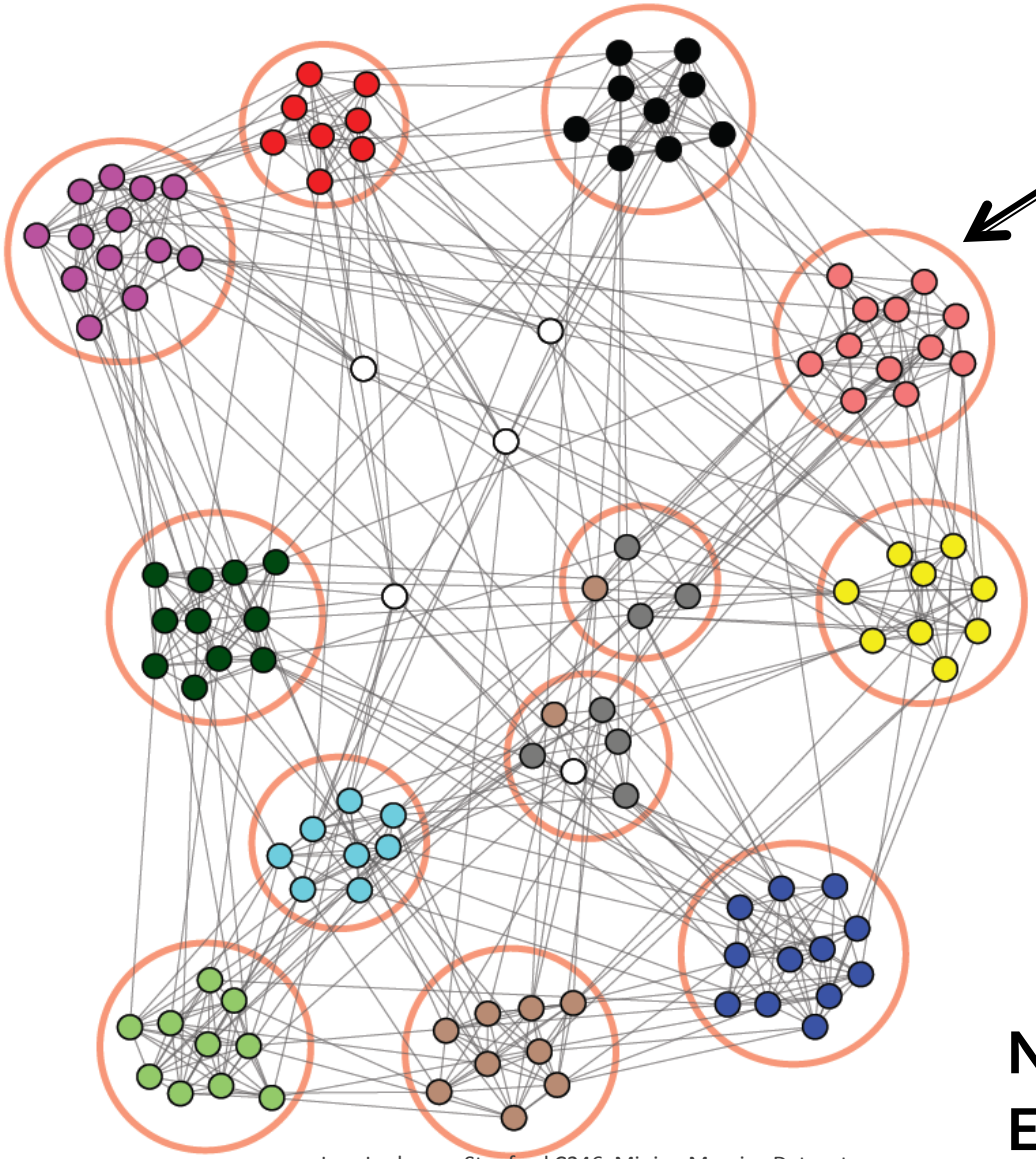
Identifying Communities



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

Nodes: Football Teams
Edges: Games played

NCAA Football Network

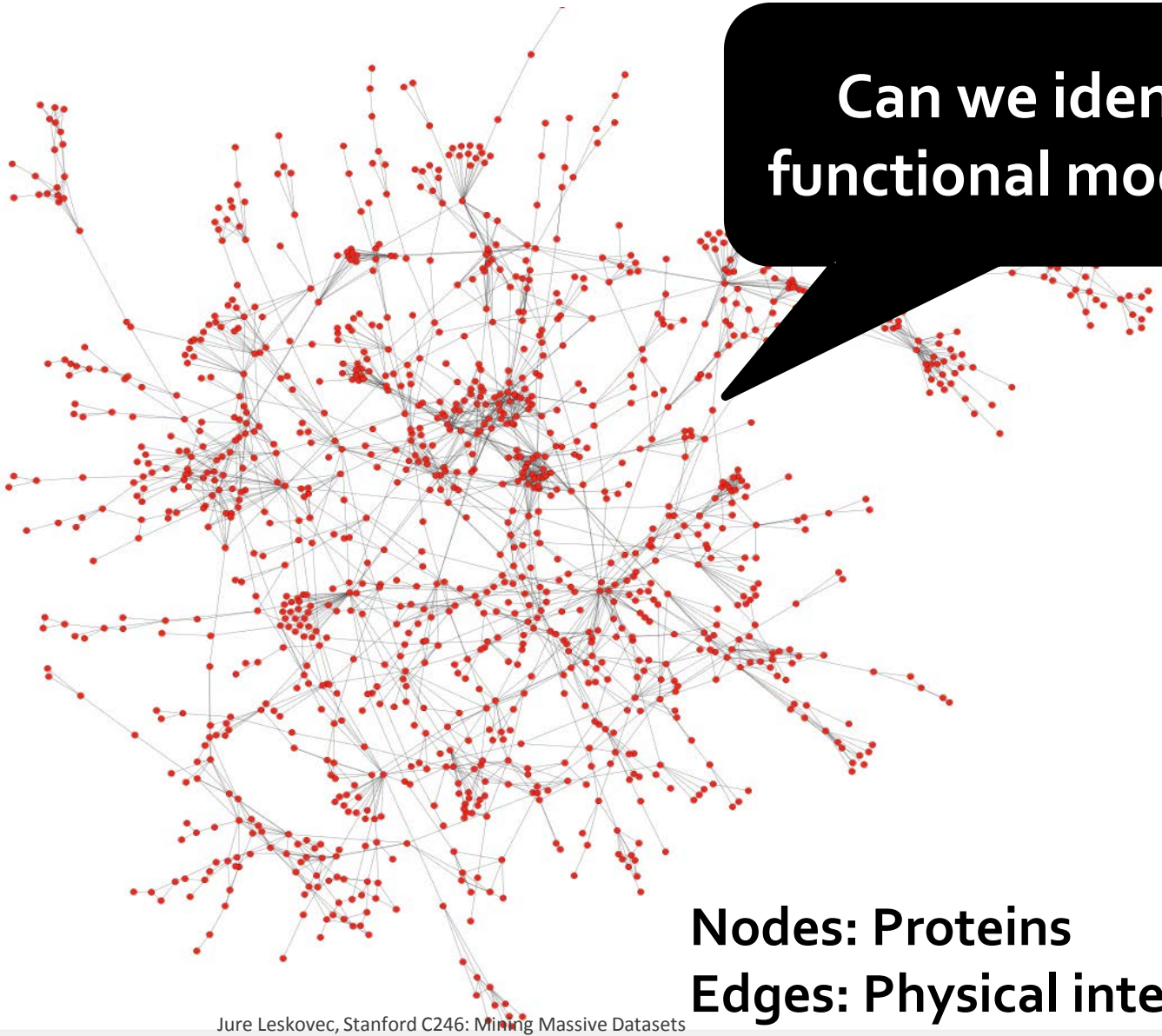


NCAA conferences

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

Nodes: Football Teams
Edges: Games played

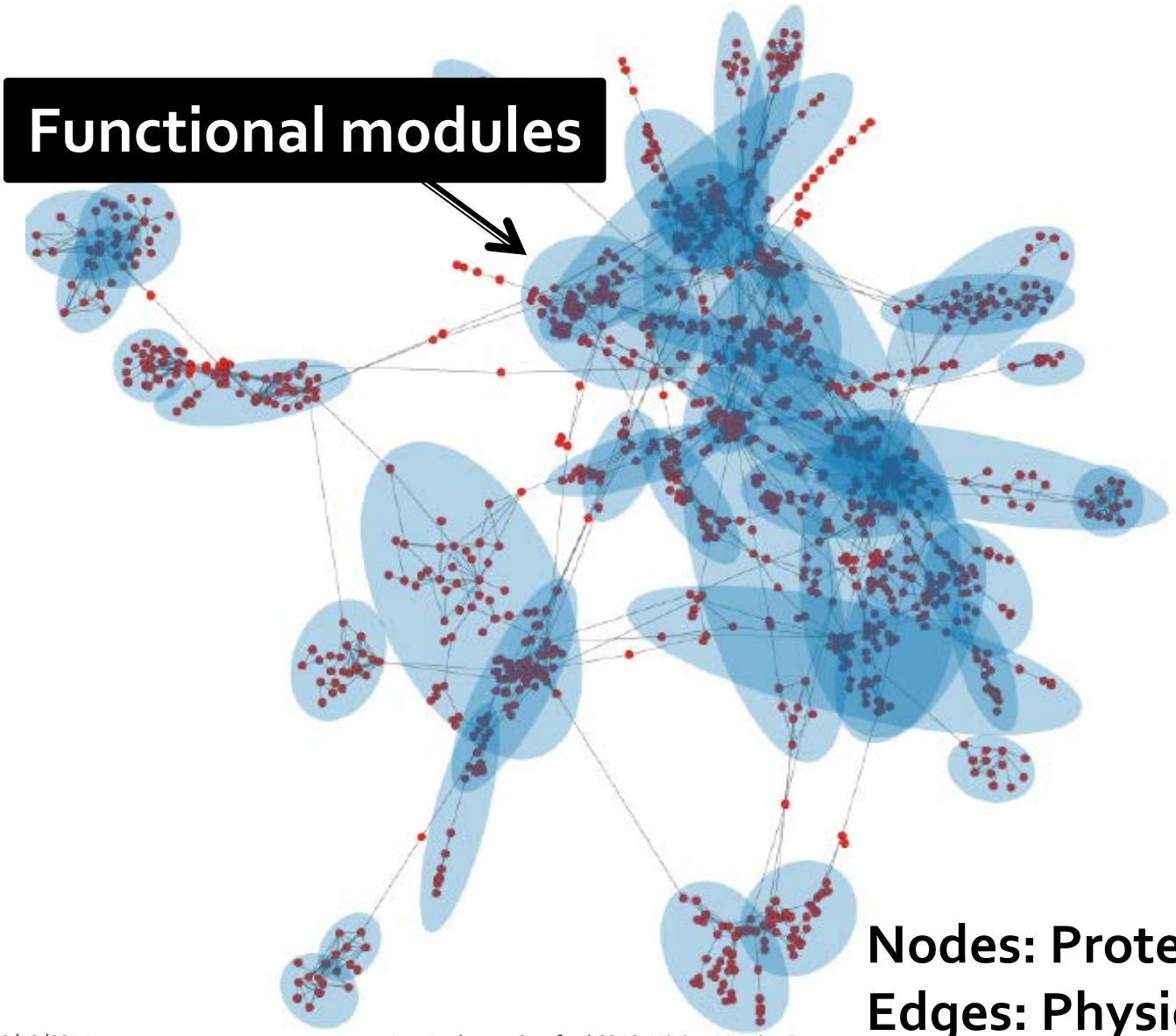
Protein-Protein Interactions



Can we identify functional modules?

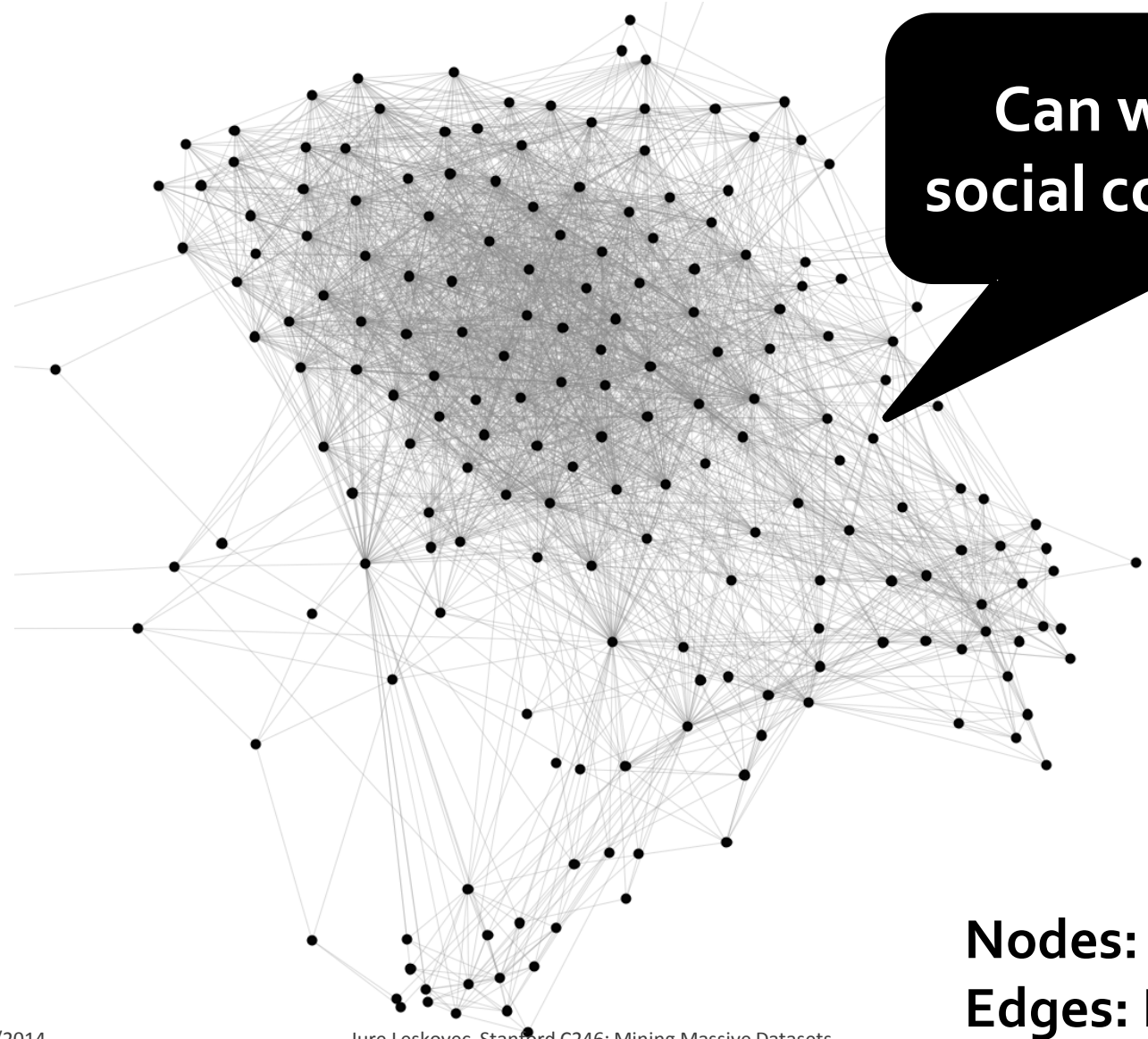
Nodes: Proteins
Edges: Physical interactions

Protein-Protein Interactions



Nodes: Proteins
Edges: Physical interactions

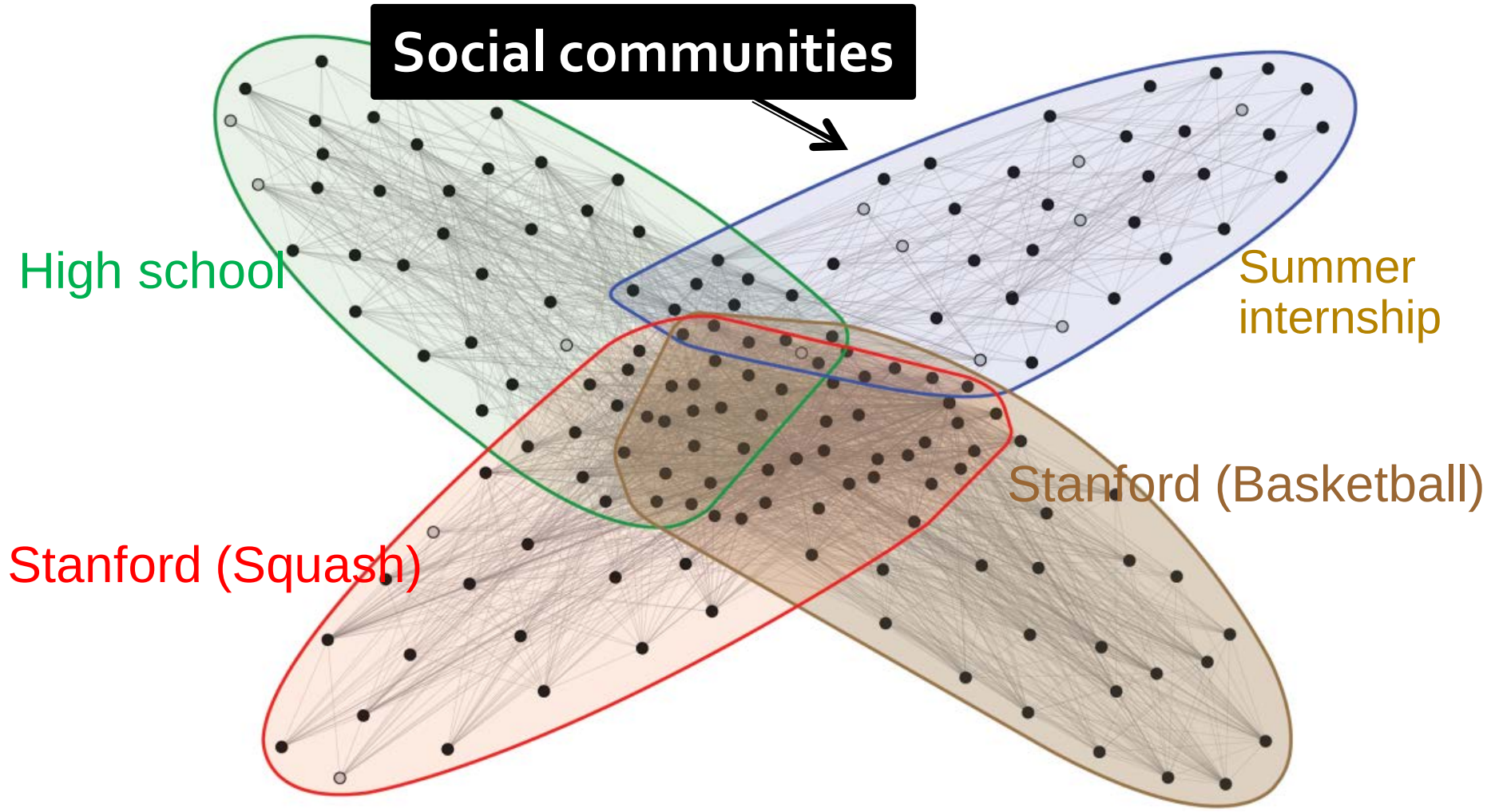
Facebook Network



Can we identify social communities?

Nodes: Facebook Users
Edges: Friendships

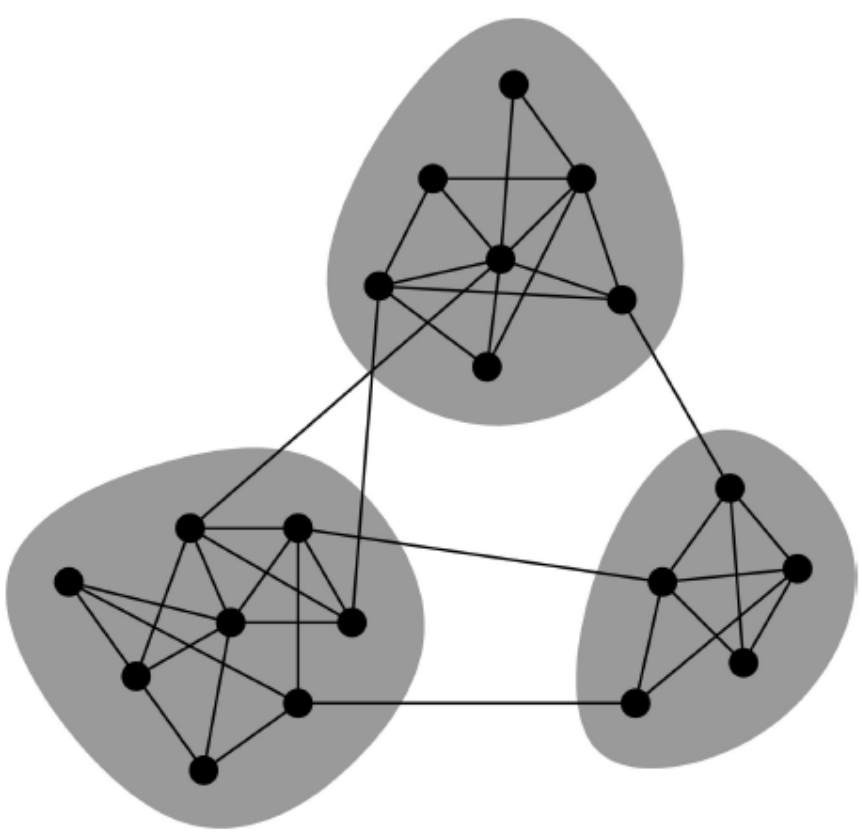
Facebook Network

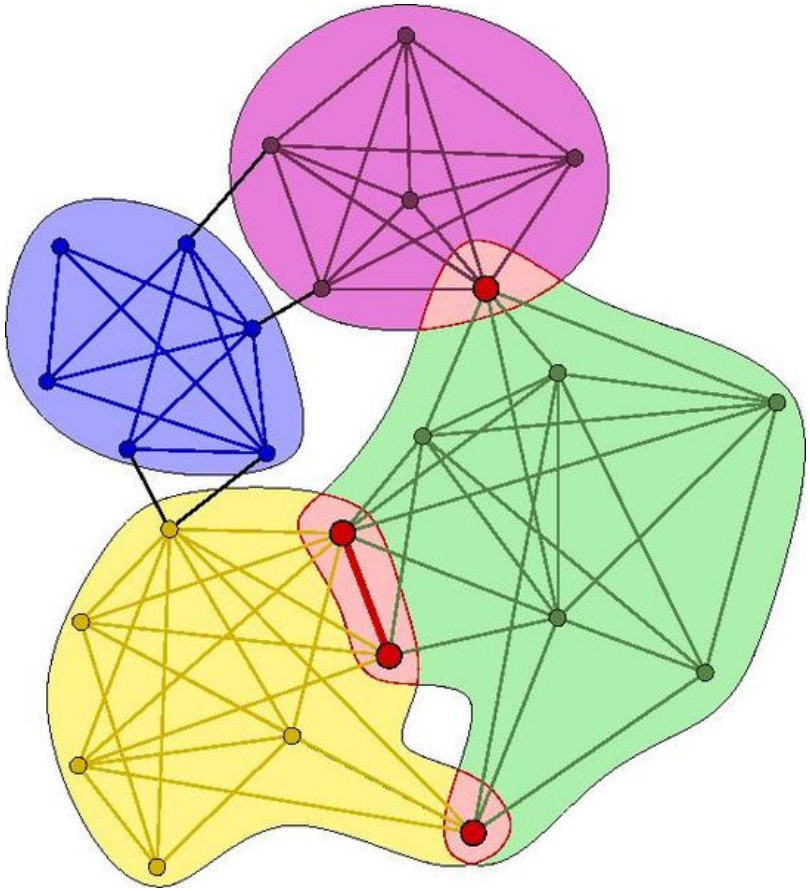


Nodes: Facebook Users
Edges: Friendships

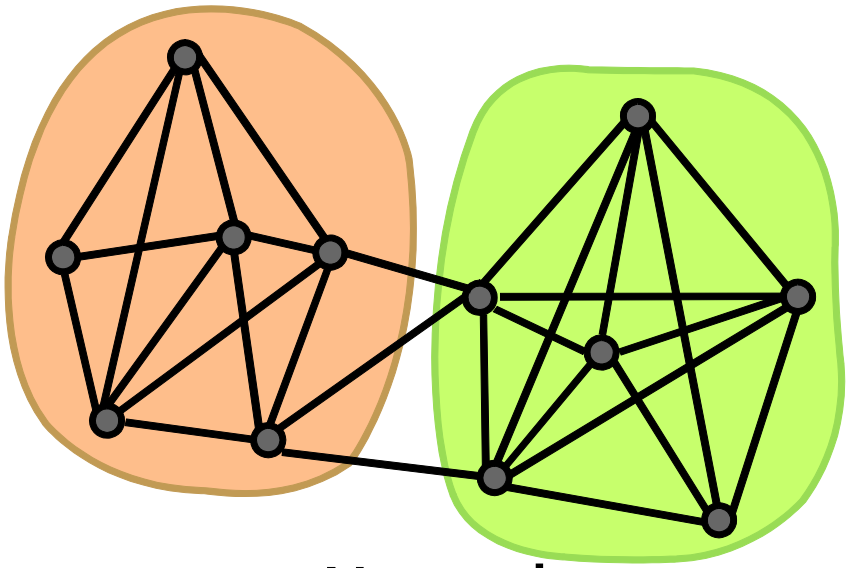
Overlapping Communities

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

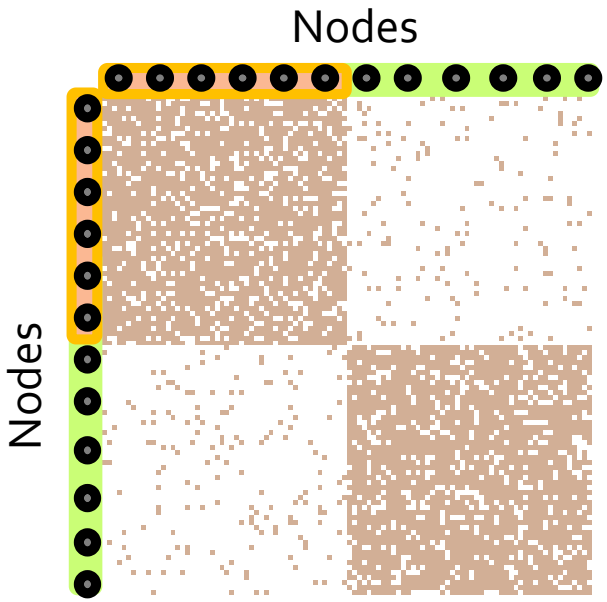




Non-overlapping Communities



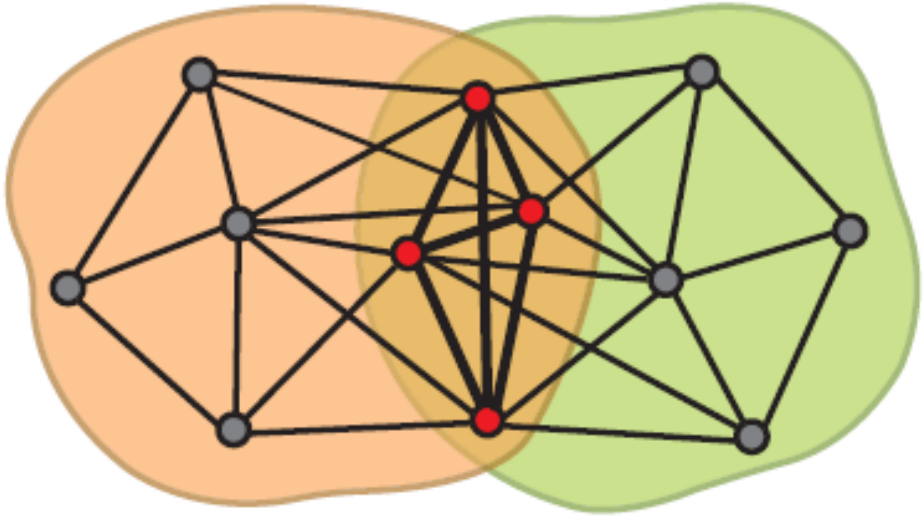
Network

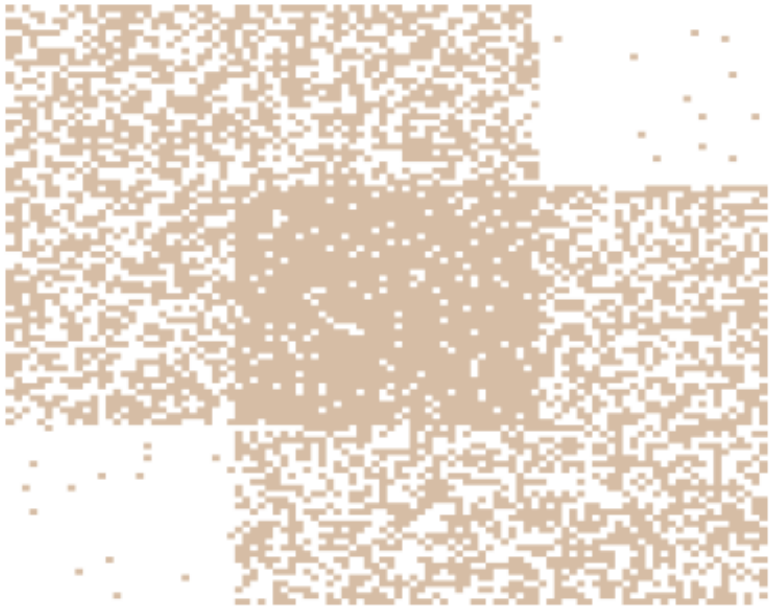


Adjacency matrix

Communities as Tiles!

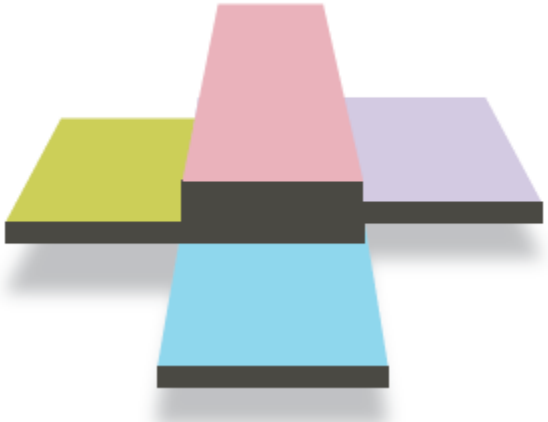
- **What is the structure of community overlaps:**
  Edge density in the overlaps is higher!



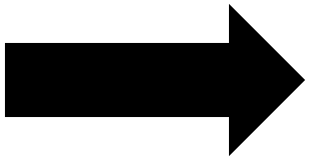


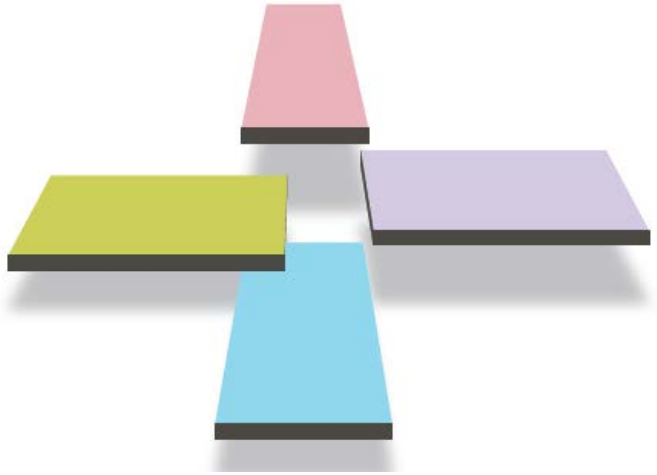
Communities as "tiles"

Recap so far…



Communities in a network

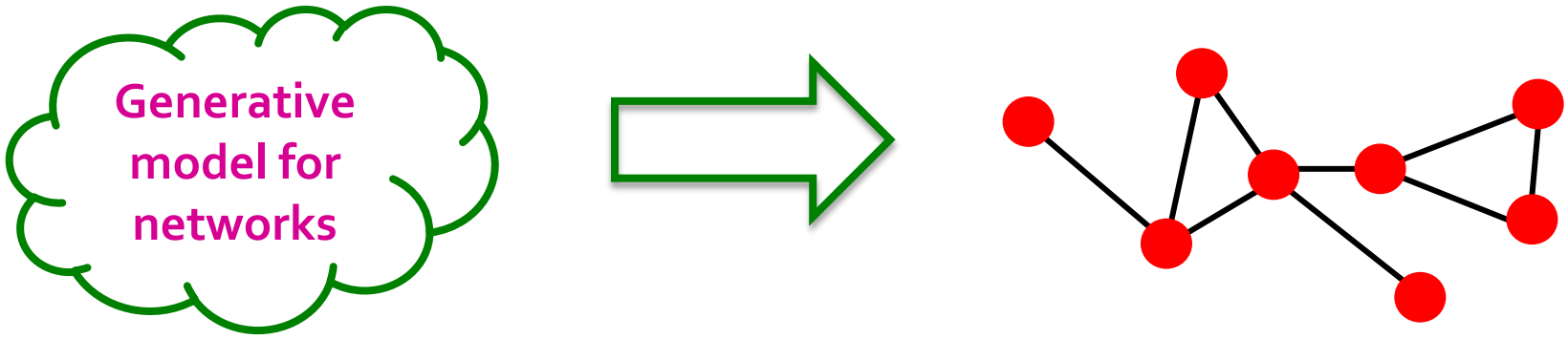




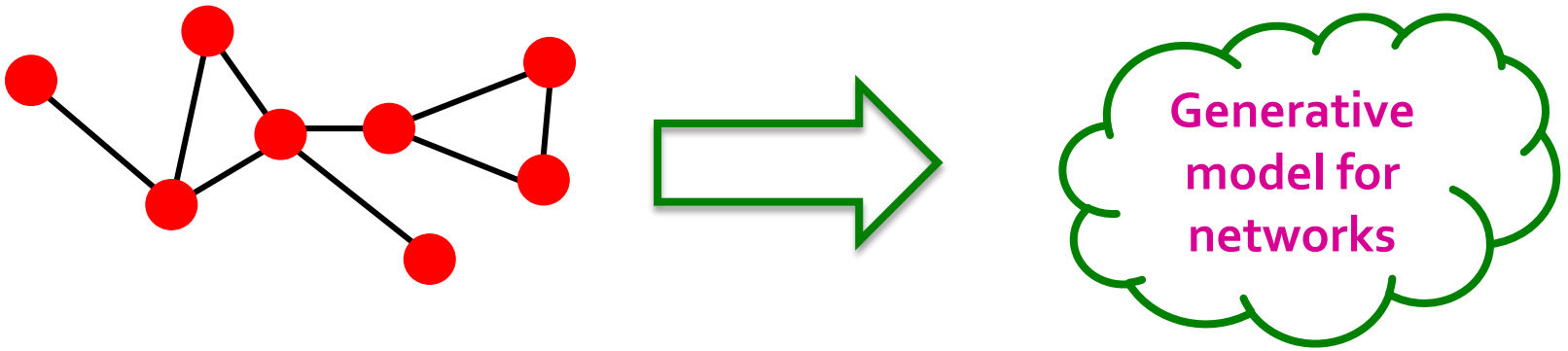
This is what we want!

Plan of attack

- **1) Given a model, we generate the network:**

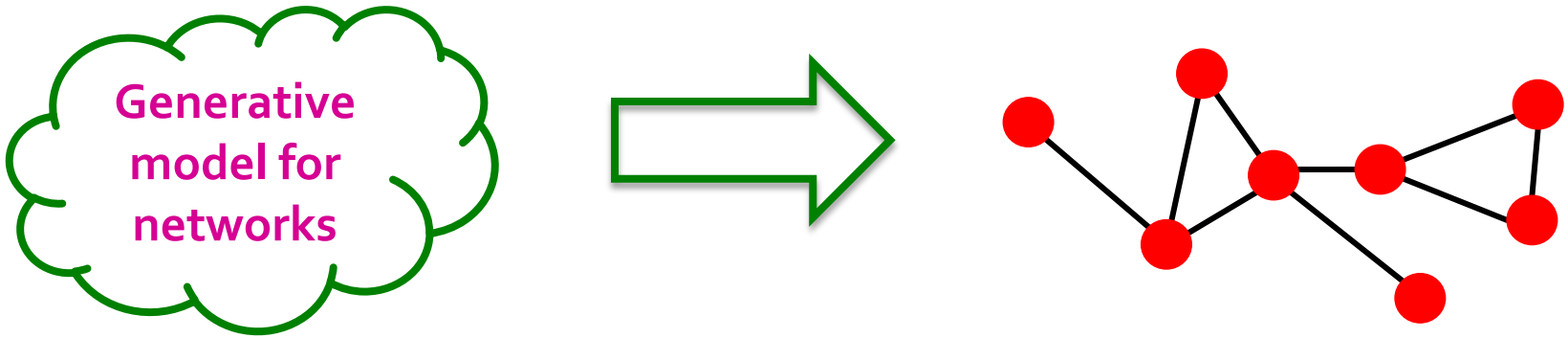


- **2) Given a network, find the "best" model**



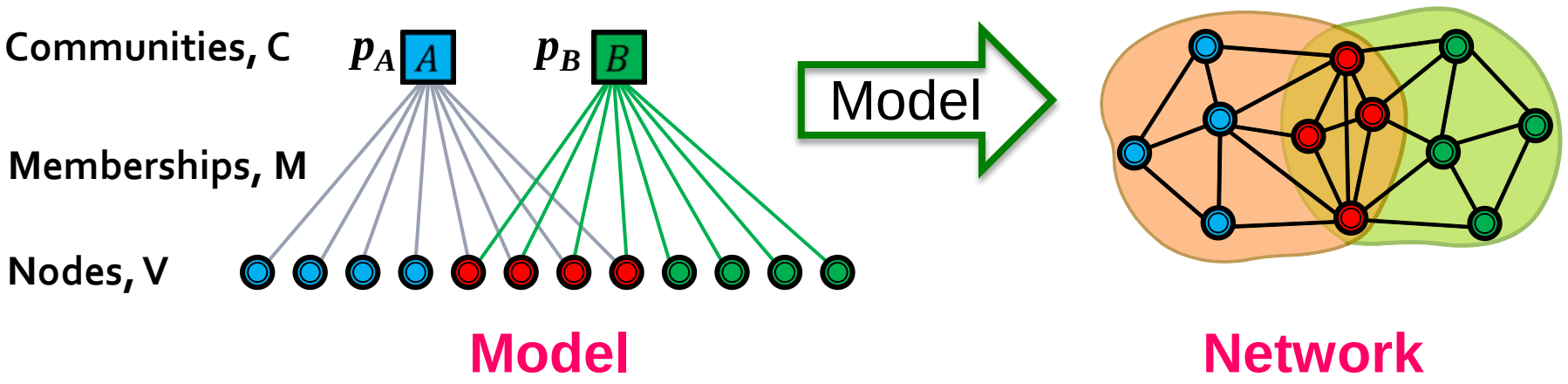
Model of networks

- **Goal: Define a model that can generate networks**
  - The model will have a set of "parameters" that we will later want to estimate (and detect communities)



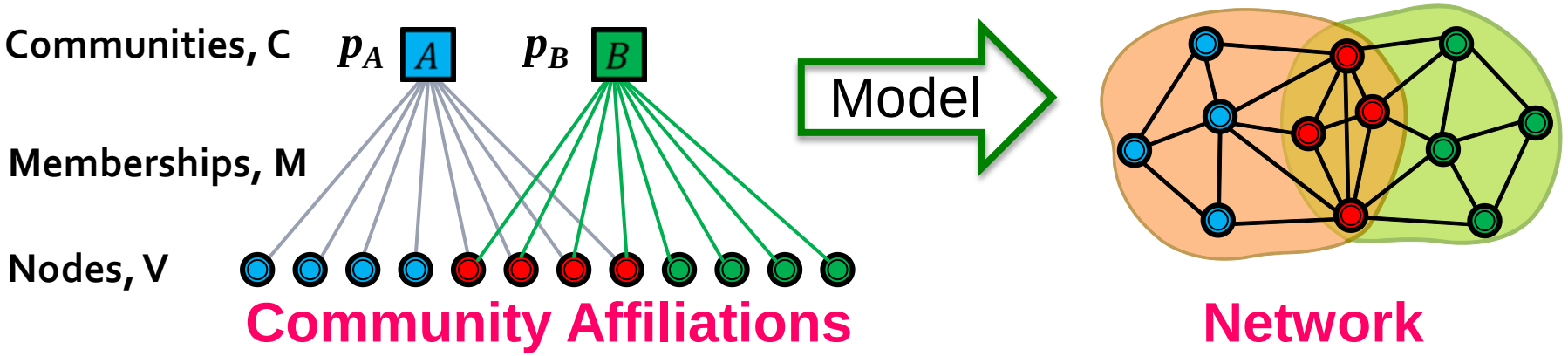
- **Q: Given a set of nodes, how do communities "generate" edges of the network?**

Community-Affiliation Graph



- **Generative model B(V, C, M, $\{p_c\}$) for graphs:**
  - Nodes **V**, Communities **C**, Memberships **M**
  - Each community **c** has a single probability $\mathbf{p_c}$
  - Later we fit the model to networks to detect communities

AGM: Generative Process



- **AGM generates the links: For each**
  - For each pair of nodes in community $\boldsymbol{A}$, we connect them with prob. $\boldsymbol{p_A}$
  - **The overall edge probability is:**

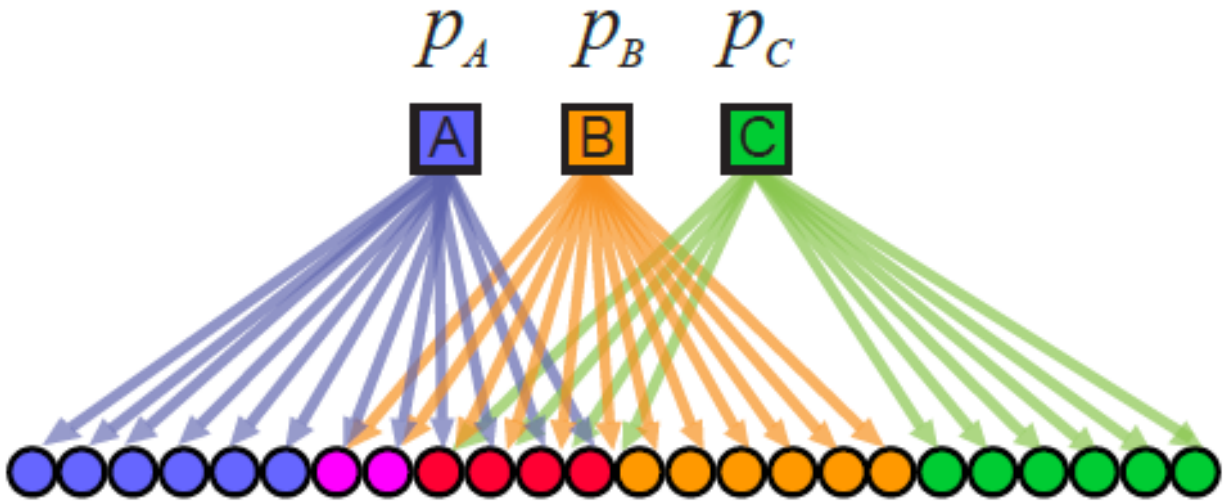
$$P(u,v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

If $\boldsymbol{u}, \boldsymbol{v}$ share no communities: $\boldsymbol{P(u,v) = \varepsilon}$

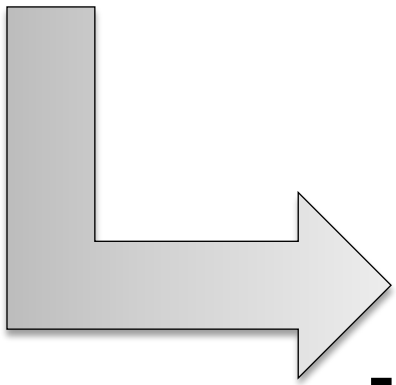
$\boldsymbol{M_u}$ … set of communities node $\boldsymbol{u}$ belongs to

Think of this as an "OR" function: If at least 1 community says "YES" we create an edge

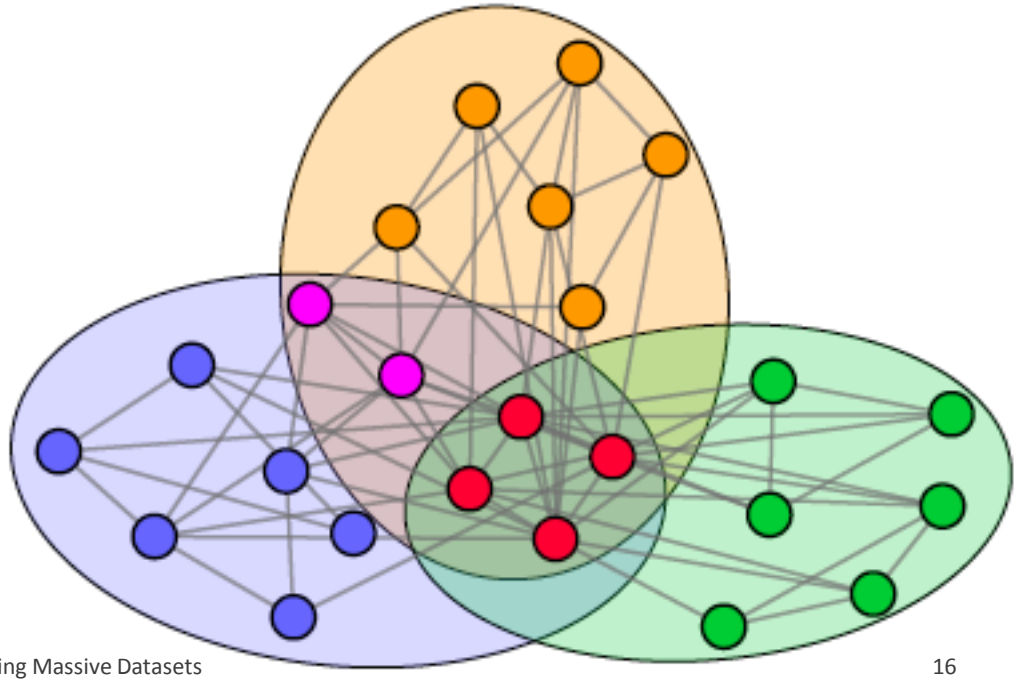
Recap: AGM networks



Model

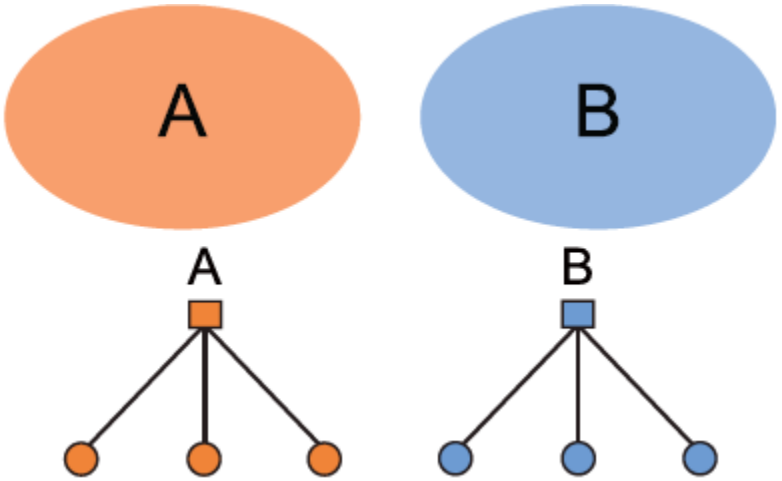


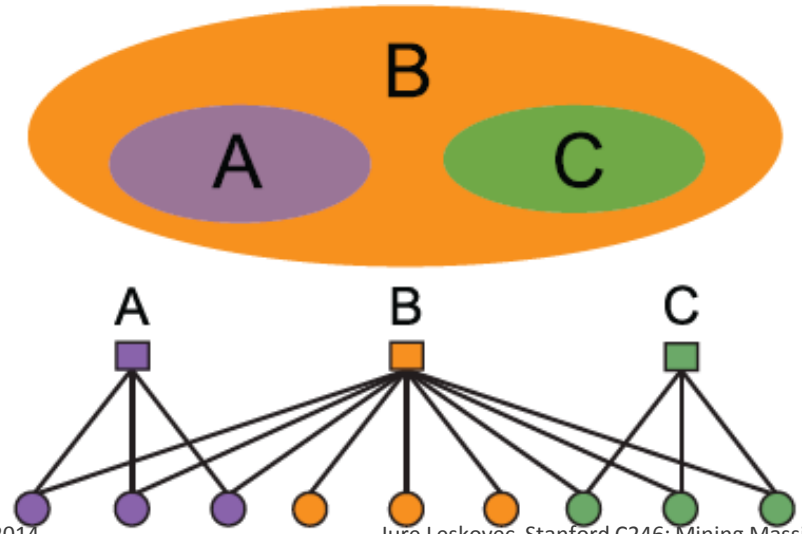
Network

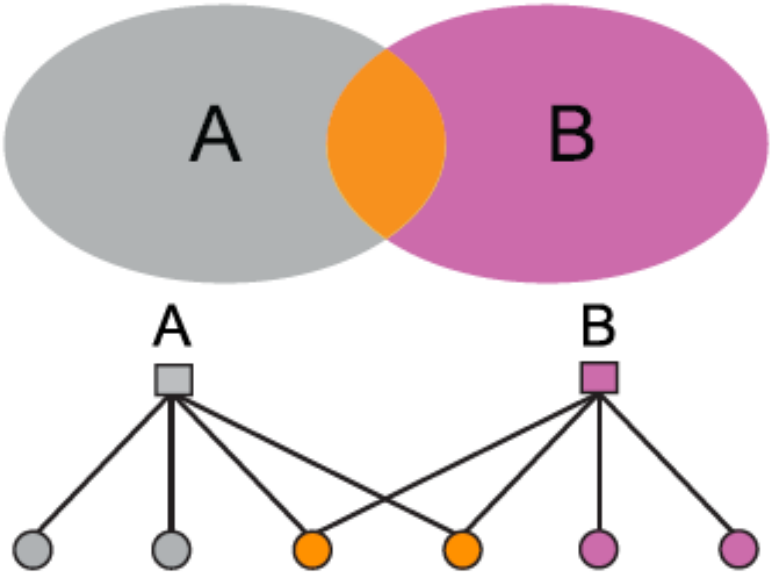


AGM: Flexibility

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

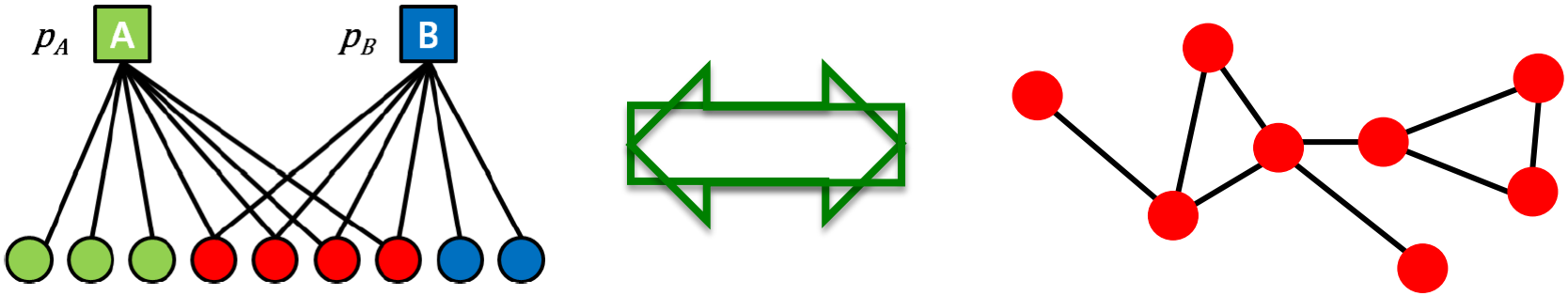






Detecting Communities

- **Detecting communities with AGM:**



Given a Graph, find the Model

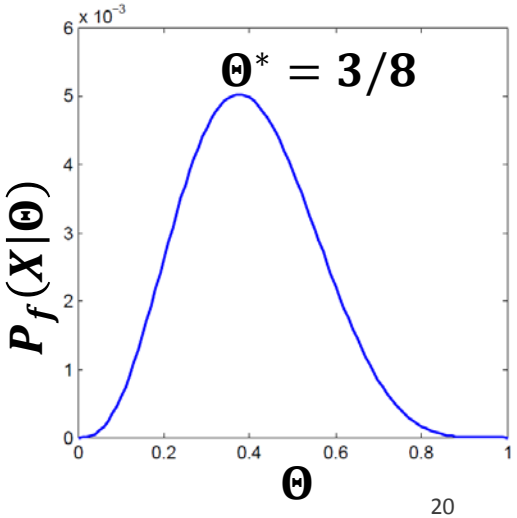
1) Affiliation graph ***M***
2) Number of communities **C**
3) Parameters $\boldsymbol{p_c}$

Maximum Likelihood Estimation

- **Maximum Likelihood Principle (MLE):**
  - **Given:** Data $\boldsymbol{X}$
  - **Assumption:** Data is generated by some model $\boldsymbol{f(\Theta)}$
    - $\boldsymbol{f}$ **… model**
    - $\boldsymbol{\Theta}$ **… model parameters**
  - Want to estimate $\boldsymbol{P_f(X|\Theta)}$:
    - The probability that our model $\boldsymbol{f}$ (with parameters $\boldsymbol{\Theta}$) generated the data
  - **Now let's find the most likely model that could have generated the data:** $\arg\max_{\Theta} \boldsymbol{P_f(X|\Theta)}$

Example: MLE

- **Imagine we are given a set of coin flips**
- **Task:** Figure the bias of a coin!
  - **Data:** Sequence of coin flips: $\boldsymbol{X = [1, 0, 0, 0, 1, 0, 0, 1]}$
  - **Model:** $\boldsymbol{f(\Theta)}$ = return 1 with prob. Θ, else return 0
  - **What is $\boldsymbol{P_f(X|\Theta)}$?** Assuming coin flips are independent
    - So, $\boldsymbol{P_f(X|\Theta) = P_f(0|\Theta) * P_f(1|\Theta) * P_f(1|\Theta) \ldots * P_f(0|\Theta)}$
      - What is $\boldsymbol{P_f(1|\Theta)}$? **Simple,** $\boldsymbol{P_f(0|\Theta) = \Theta}$
    - **Then,** $\boldsymbol{P_f(X|\Theta) = \Theta^3(1-\Theta)^5}$
    - **For example:**
      - $\boldsymbol{P_f(X|\Theta = 0.5) = 0.003906}$
      - $\boldsymbol{P_f\left(X\middle|\Theta = \frac{3}{8}\right) = 0.005029}$
  - **What did we learn?** Our data was most likely generated by coin with bias $\boldsymbol{\Theta = 3/8}$



MLE for Graphs

- **How do we do MLE for graphs?**
  - **Model generates a probabilistic adjacency matrix**
  - We then flip all the entries of the probabilistic matrix to obtain the **binary adjacency matrix**

For every pair of nodes u, v AGM gives the prob. p_{uv} of them being linked

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

Flip biased coins

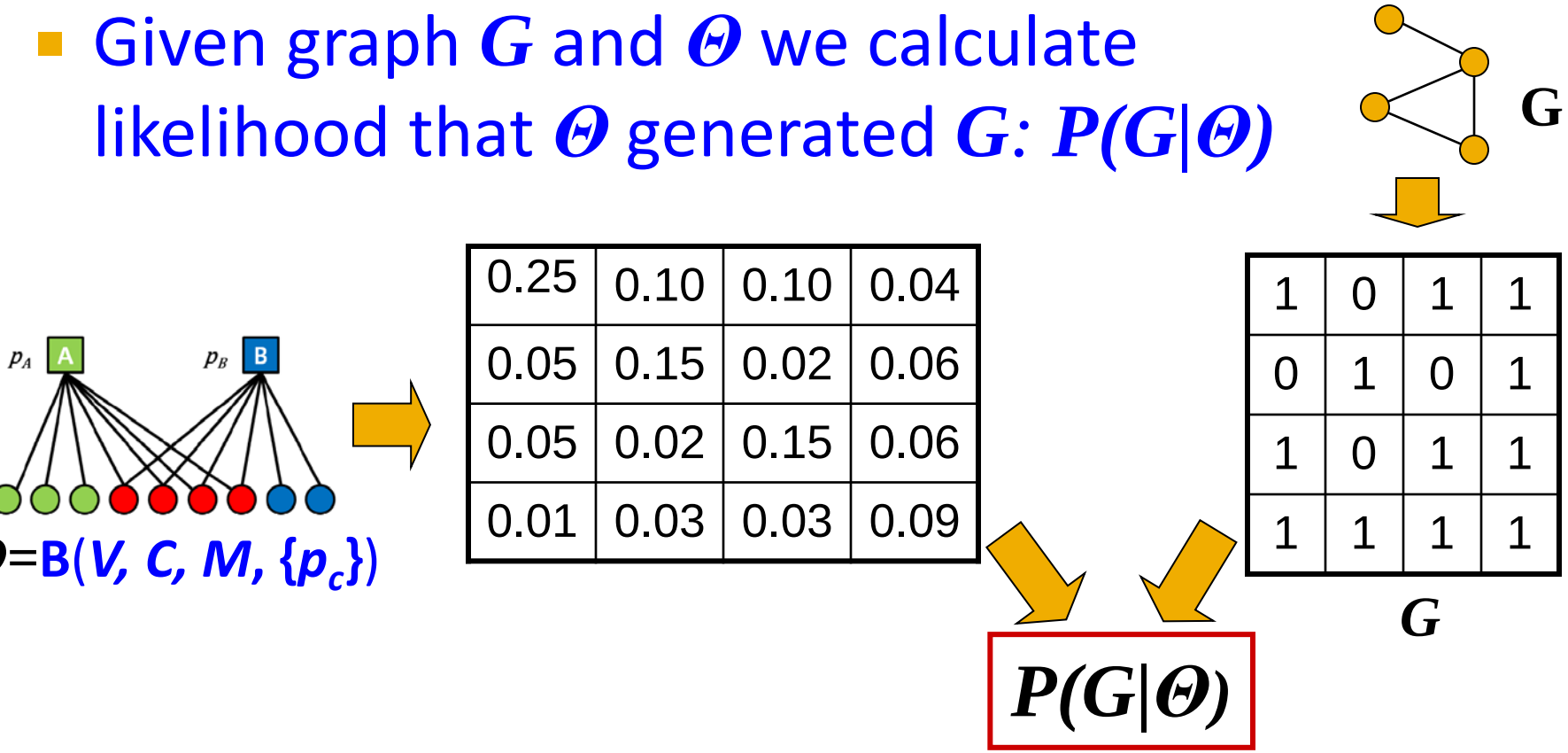
1	1	0	0
0	1	0	1
1	0	1	1
0	1	0	1

- **The likelihood of AGM generating graph G:**

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

Graphs: Likelihood $P(G|\Theta)$

- Given graph G and Θ we calculate likelihood that Θ generated G: $P(G|\Theta)$

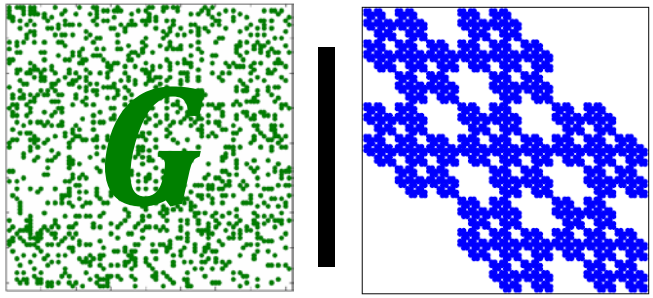


$$P(G \mid \Theta) = \prod_{(u,v)\in G} P(u,v) \prod_{(u,v)\notin G} (1 - P(u,v))$$

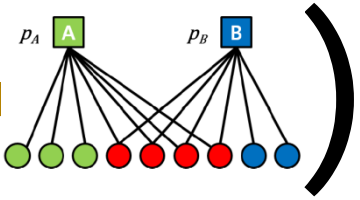
MLE for Graphs

- **Our goal:** Find $\Theta = B(V, C, M, \{p_C\})$ **such that:**

$$\arg\max_{\Theta} P(G \mid \text{AGM}(\Theta))$$



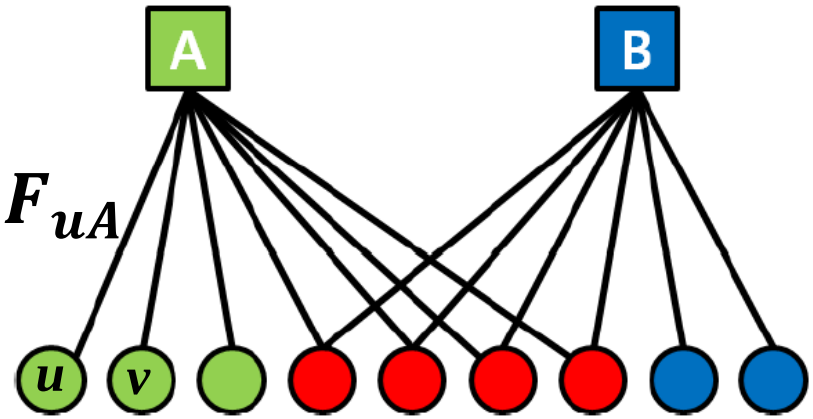
AGM



- **How do we find $B(V, C, M, \{p_C\})$ that maximizes the likelihood?**
  - No nice way, have to search over all affiliation graphs ☹ But, don't despair…

From AGM to BigCLAM

- **Relaxation: Memberships have strengths**

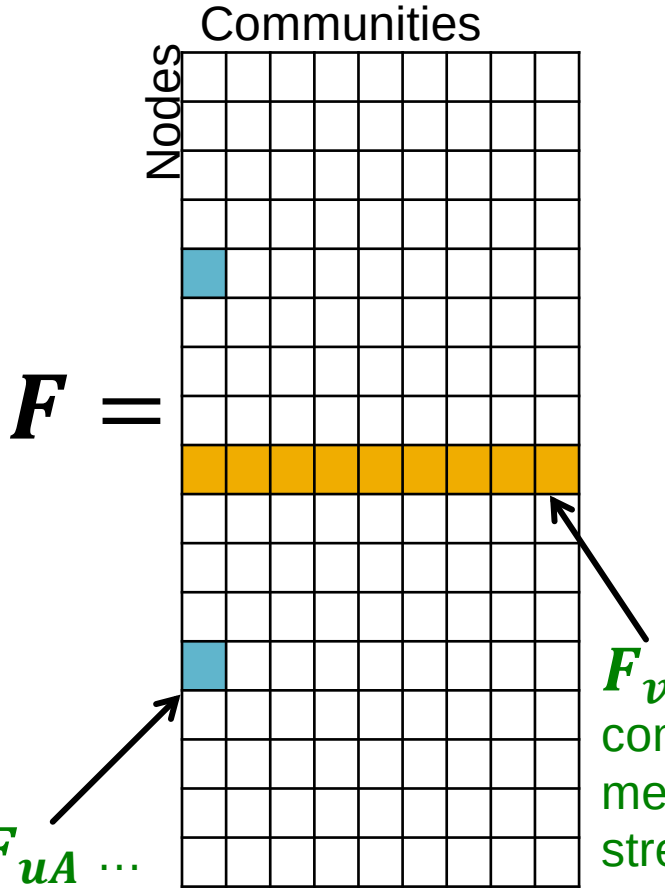


  - F_{uA}: The membership strength of node u to community A ($F_{uA} = 0$: no membership)
  - **Each community A links nodes independently:**

$$P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$$

From AGM to BigCLAM

- **Community membership strength matrix F**



- $P_A(u,v) = 1 - \exp(-F_{uA} \cdot F_{vA})$
  - **Probability of connection is proportional to the product of strengths**
- Prob. that **at least one** common community C links the nodes:
  - $P(u,v) = 1 - \prod_C (1 - P_C(u,v))$

From AGM to BigCLAM

- Community A links nodes u, v independently:

$$P_A(u,v) = 1 - \exp(-F_{uA} \cdot F_{vA})$$

- Then prob. at least one common C links them:

$$P(u,v) = 1 - \prod_C (1 - P_C(u,v))$$
$$= 1 - \exp(-\sum_C F_{uC} \cdot F_{vC})$$
$$= 1 - \exp(-F_u \cdot F_v^T)$$

- **For example:**

F_u:	0	1.2	0	0.2
F_v:	0.5	0	0	0.8
F_w:	0	1.8	1	0

Node community membership strengths

Then: $F_u \cdot F_v^T = 0.16$

And: $P(u,v) = 1 - exp(-0.16) = 0.14$

But: $P(u,w) = 0.88$

$P(v,w) = 0$

From AGM to BigCLAM

- **Another way to think about this is:**
  - **Think of a weighted graph (but we don't see the weights)**
  - u, v interact with strength $X_{uv}^{A} \sim Pois(F_{uA} \cdot F_{vA})$
  - Total interaction strength is the sum: $X_{uv} = \sum_C X_{uv}^{C}$
  - Then: $X_{uv} \sim Pois(\sum_C F_{uC} \cdot F_{vC})$
  - **We observe an edge if there is non-zero interaction:**

$$P(X_{uv} > 0) = 1 - P(X_{uv} = 0) =$$
$$= \mathbf{1 - \exp(-F_u \cdot F_v^T)}$$

Poisson PMF:

$$P(X = x) = \frac{\lambda^k}{k!} e^{-\lambda}$$

BigCLAM

[wsdm '13]

- **Task: Given a network, estimate F**
  - **Find F that maximizes the log-likelihood**
    - Many times we take the logarithm of the likelihood and call it log-likelihood: $l(F) = \log P(G|F)$

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

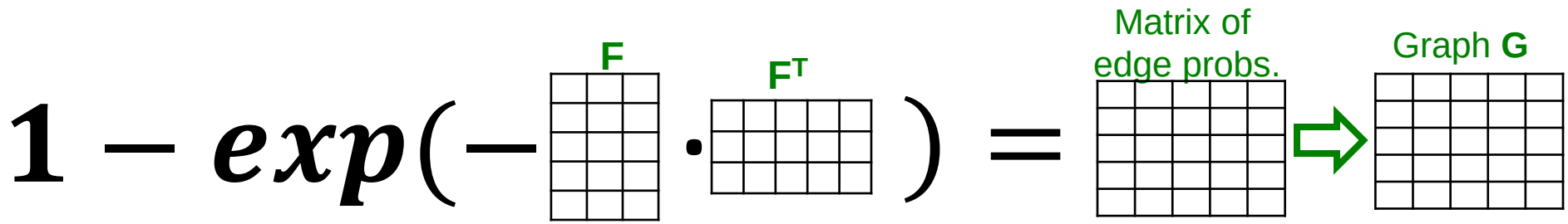
  - Objective function is biconvex
- **Non-negative matrix factorization:**
  - Update $\boldsymbol{F_{uC}}$ for node $\boldsymbol{u}$ while fixing the memberships of all other nodes

BigCLAM

- **Task: Given a network, estimate F**
  - **Find F that maximizes the log-likelihood**
    - Many times we take the logarithm of the likelihood and call it log-likelihood: $l(F) = \log P(G|F)$

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

- **Connection: Non-negative matrix factorization**
  - We want to "**factor**" the adjacency matrix **G**



How to find MLE

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

- **Non-negative matrix factorization:**
  - **Use a gradient based approach!**
  - But updating the whole $\boldsymbol{F}$ takes too long
    - **Computing the gradient takes quadratic time!**
    - Our network are far too big to allow for that:
      - Network with 1M nodes, can have $\mathbf{10^{12}}$ different edges
  - **Instead, update $\boldsymbol{F}$ in small "units":**
    - Update $\boldsymbol{F_{uC}}$ for node $\boldsymbol{u}$ while fixing the memberships of all other nodes

BigCLAM: V1.0

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

- **Compute gradient of a single row F_u of F:**

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

- **Coordinate gradient ascent:**
  - **Iterate over the rows of F:**
    - Compute gradient $\nabla l(F_u)$ of row u (while keeping others fixed)
    - Update the row F_u: $F_u \leftarrow F_u + \eta \nabla l(F_u)$
    - Project F_u back to a non-negative vector: If $F_{uC} < 0$: $F_{uC} = 0$
- **This is slow!** Computing $\nabla l(F_u)$ takes linear time!

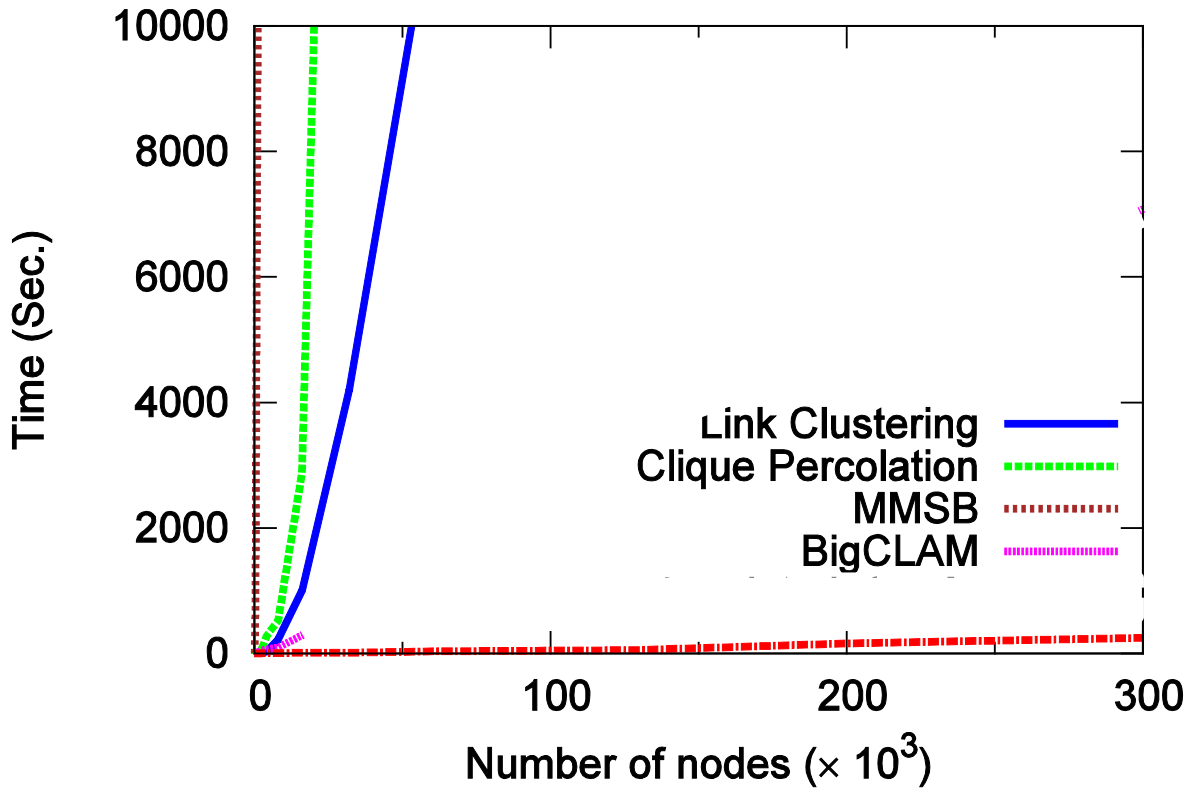
BigCLAM: V2.0

- **However, we notice:**

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

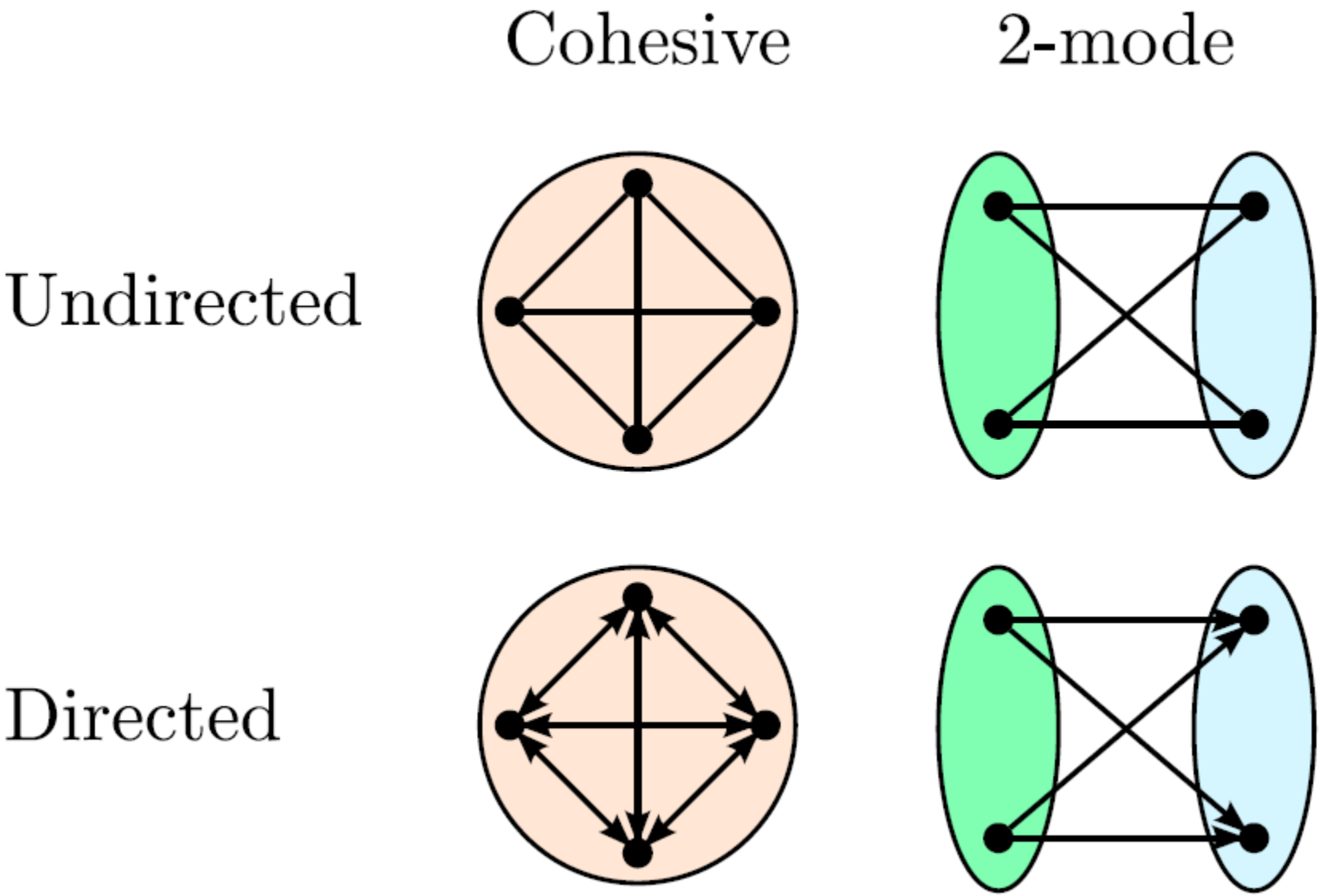
  - We can cache $\sum_v \boldsymbol{F_v}$
  - So, computing $\sum_{\boldsymbol{v \notin \mathcal{N}(u)}} \boldsymbol{F_v}$ now takes **linear time** in the degree $\boldsymbol{\mathcal{N}(u)}$ of $\boldsymbol{u}$
    - In networks degree of a node is much smaller to the total number of nodes in the network, so this is a significant speedup!

BigClam: Scalability



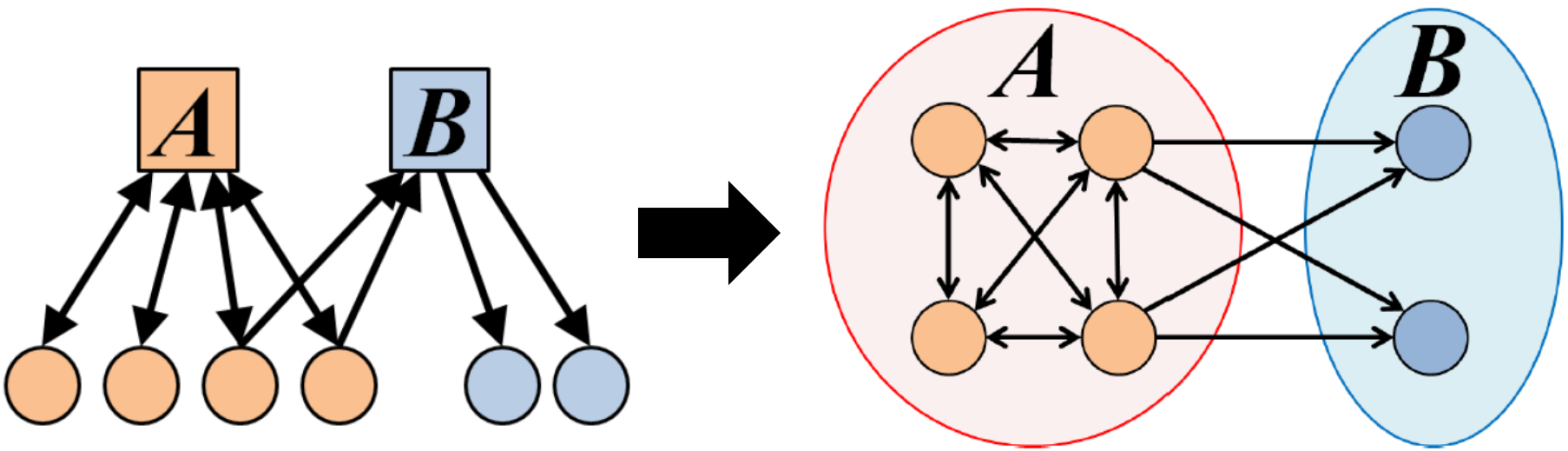
- **BigCLAM takes 5 minutes for 300k node nets**
  - **Other methods take 10 days**
- **Can process networks with 100M edges!**

Extension: Beyond Clusters

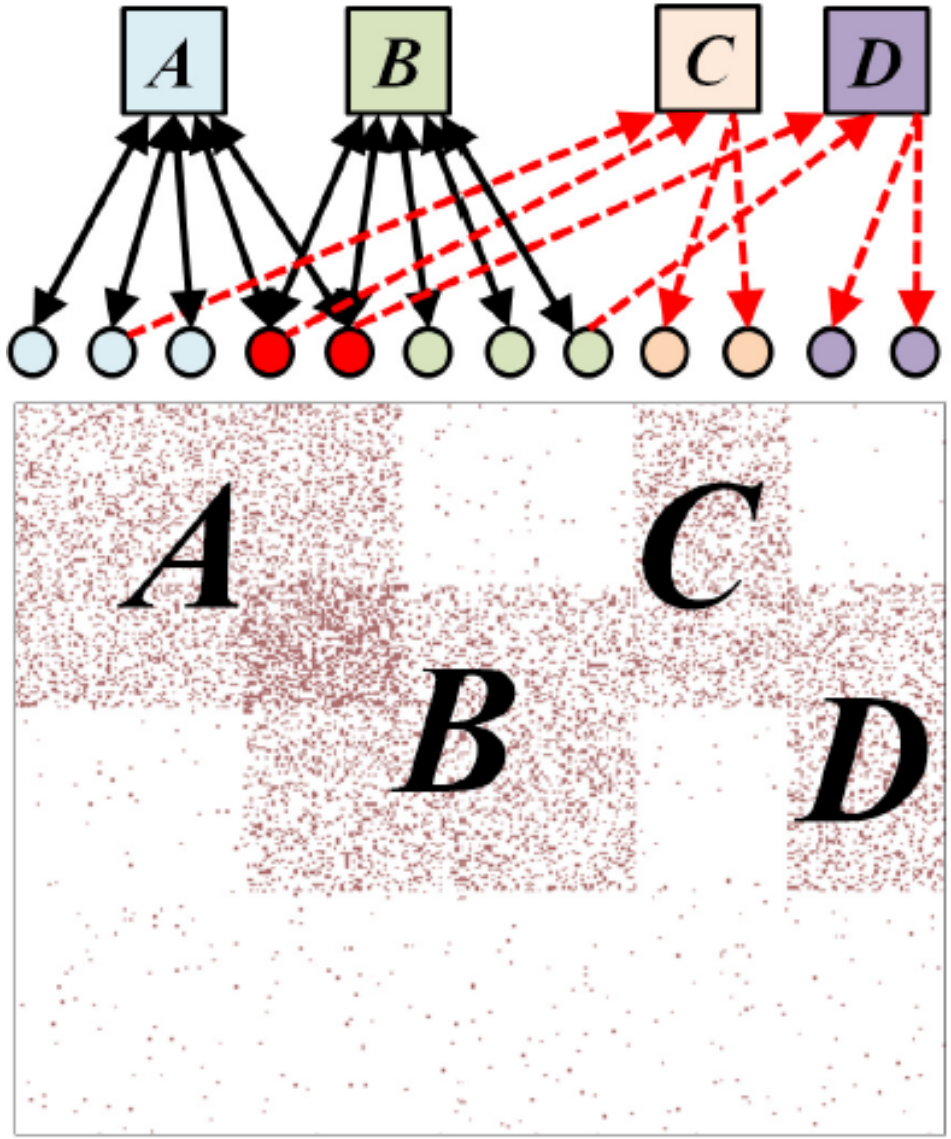


Extension: Directed AGM

- **Extension: Make community membership edges directed:**
  - **Outgoing membership:** Nodes "**sends**" edges
  - **Incoming membership:** Node "**receives**" edges

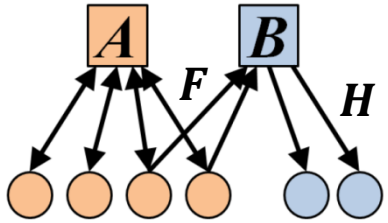


Example: Model and Network



Directed AGM

- Everything is almost the same except now we have 2 matrices: $\boldsymbol{F}$ and $\boldsymbol{H}$
  - $\boldsymbol{F}$… out-going community memberships
  - $\boldsymbol{H}$… in-coming community memberships

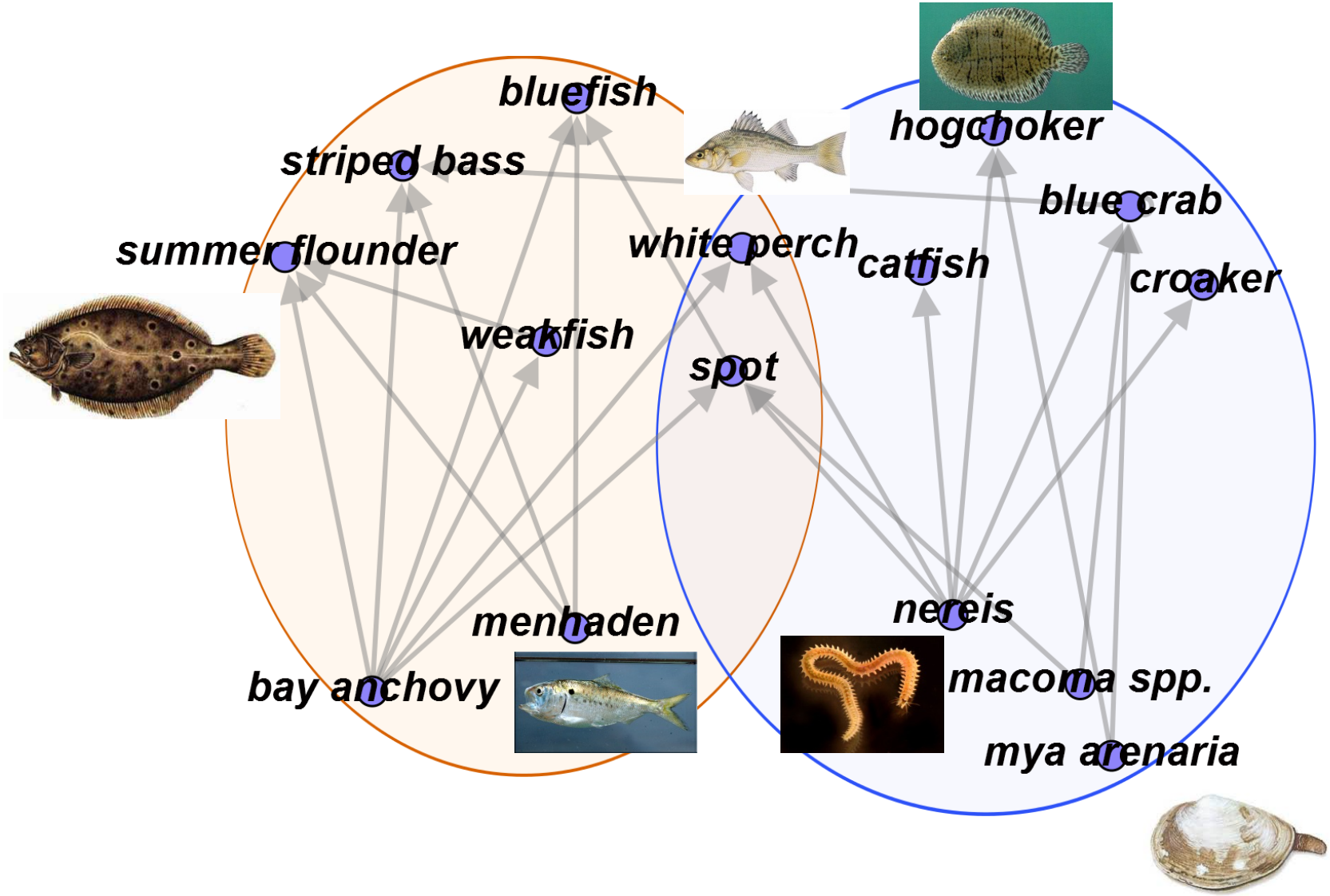


- **Edge prob.:** $\boldsymbol{P(u,v) = 1 - exp(-F_u H_v^T)}$
- **Network log-likelihood:**

$$l(F,H) = \sum_{(u,v)\in E} \log(1 - \exp(-F_u H_v^T)) - \sum_{(u,v)\notin E} F_u H_v^T$$

which we optimize the same way as before

Predator-prey Communities



More details at…

- Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach by J. Yang, J. Leskovec. *ACM International Conference on Web Search and Data Mining (WSDM)*, 2013.
- Detecting Cohesive and 2-mode Communities in Directed and Undirected Networks by J. Yang, J. McAuley, J. Leskovec. *ACM International Conference on Web Search and Data Mining (WSDM)*, 2014.
- Community Detection in Networks with Node Attributes by J. Yang, J. McAuley, J. Leskovec. *IEEE International Conference On Data Mining (ICDM)*, 2013.