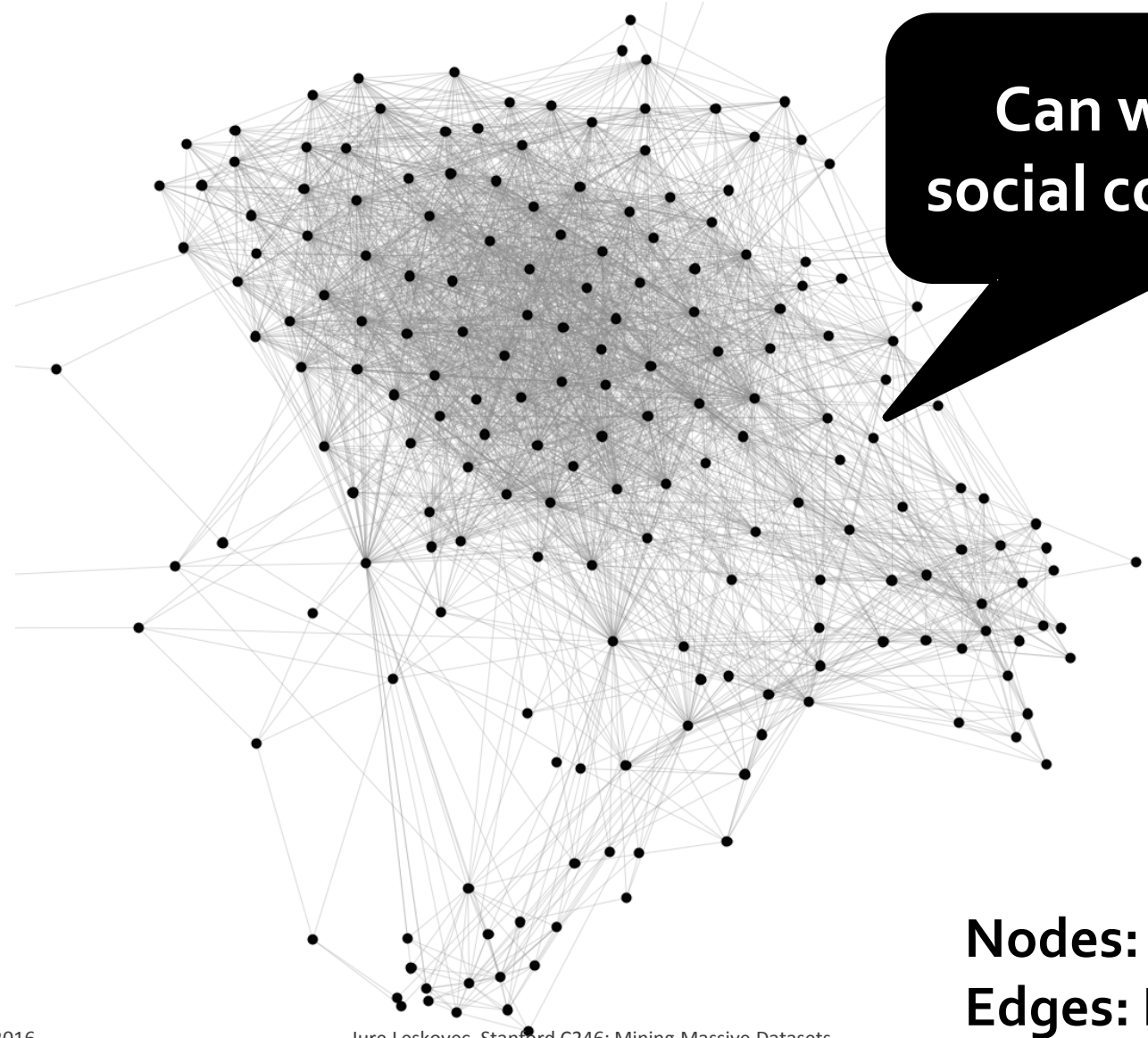
Community Detection in Graphs: Finding overlaps

CS246: Mining Massive Datasets

Jure Leskovec, Stanford University

http://cs246.stanford.edu

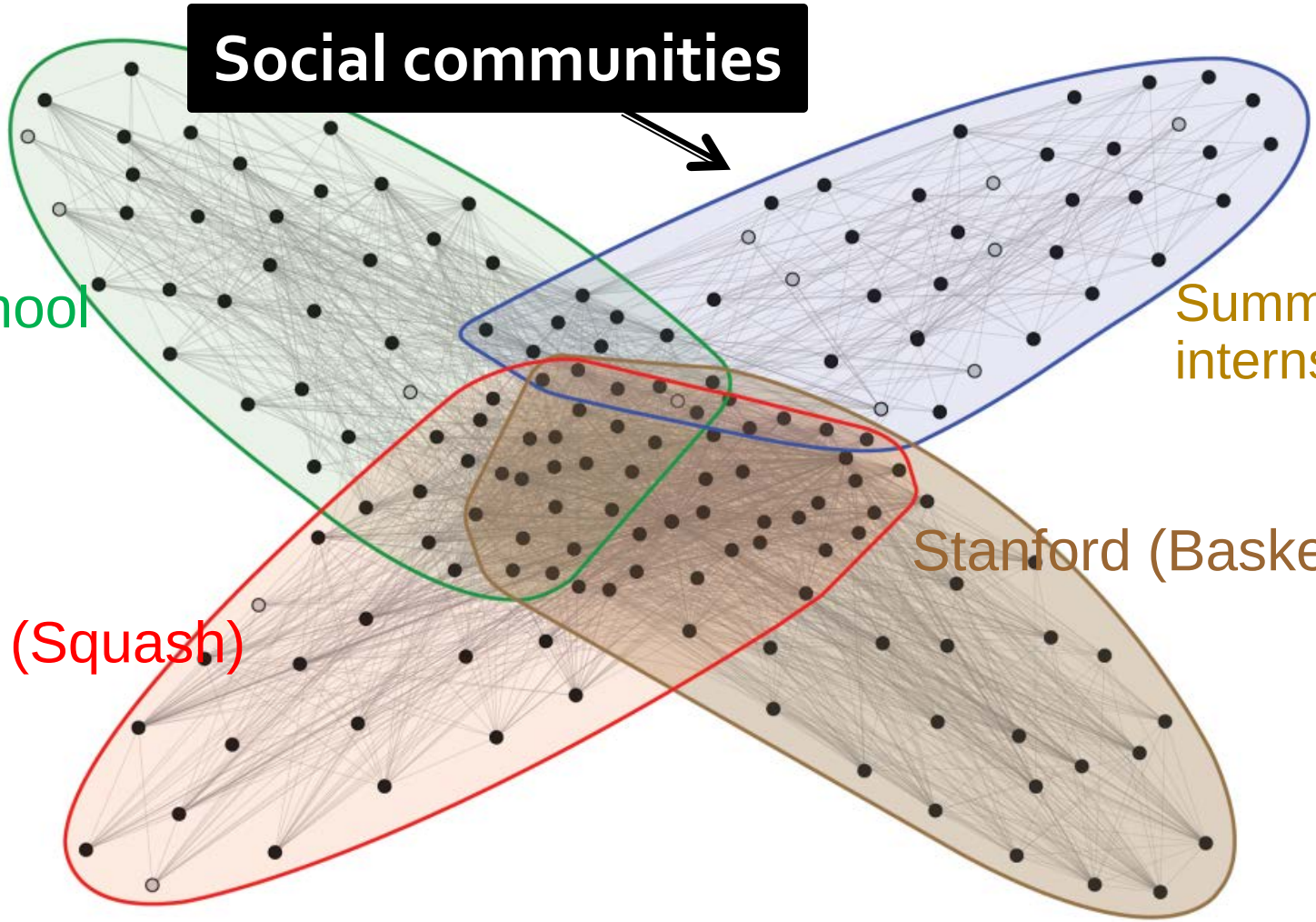
Facebook Network



Can we identify social communities?

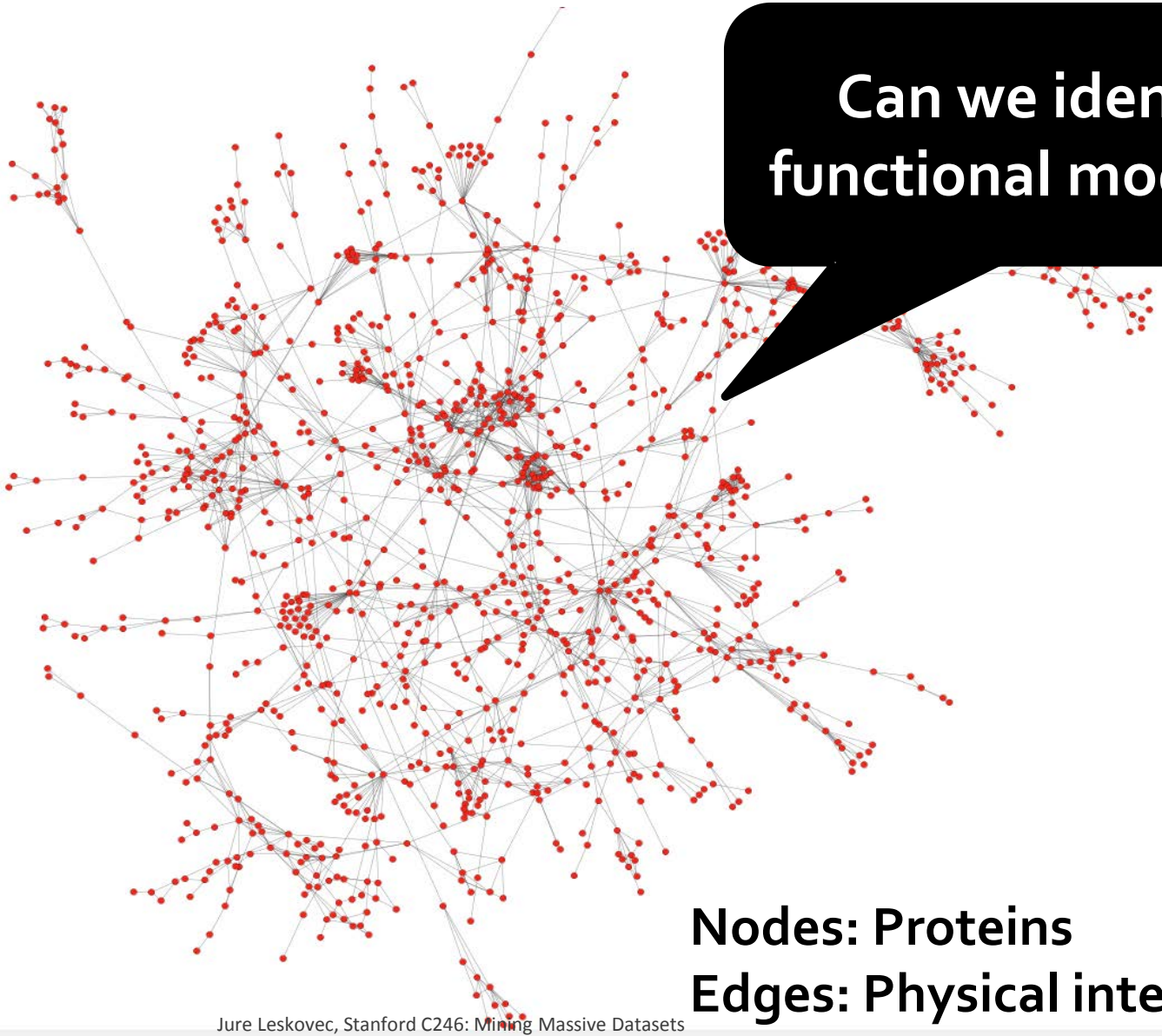
Nodes: Facebook Users
Edges: Friendships

Facebook Network



Nodes: Facebook Users
Edges: Friendships

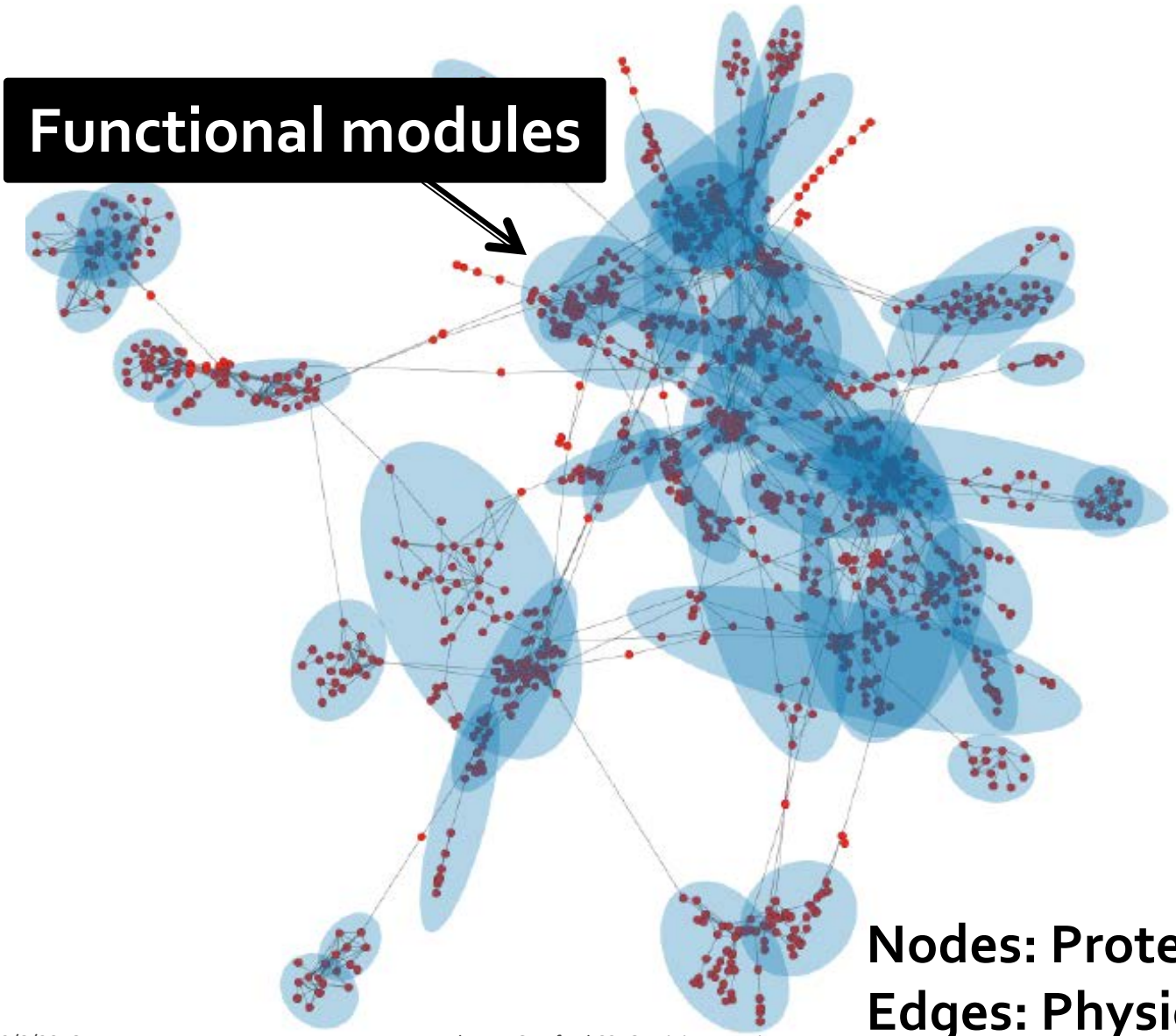
Protein-Protein Interactions



Can we identify functional modules?

Nodes: Proteins
Edges: Physical interactions

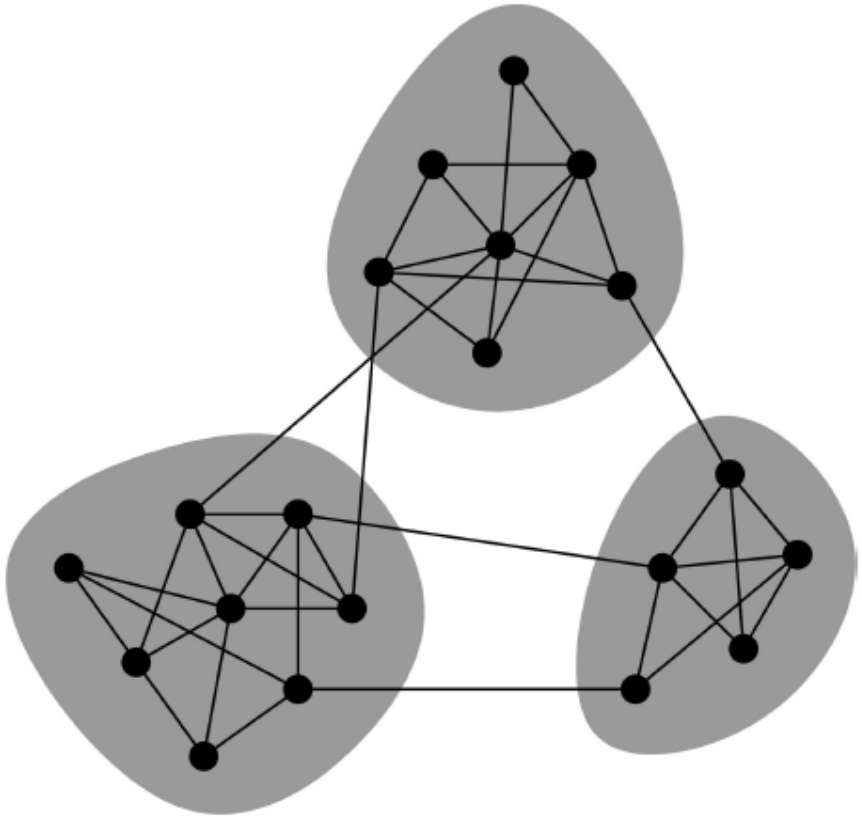
Protein-Protein Interactions



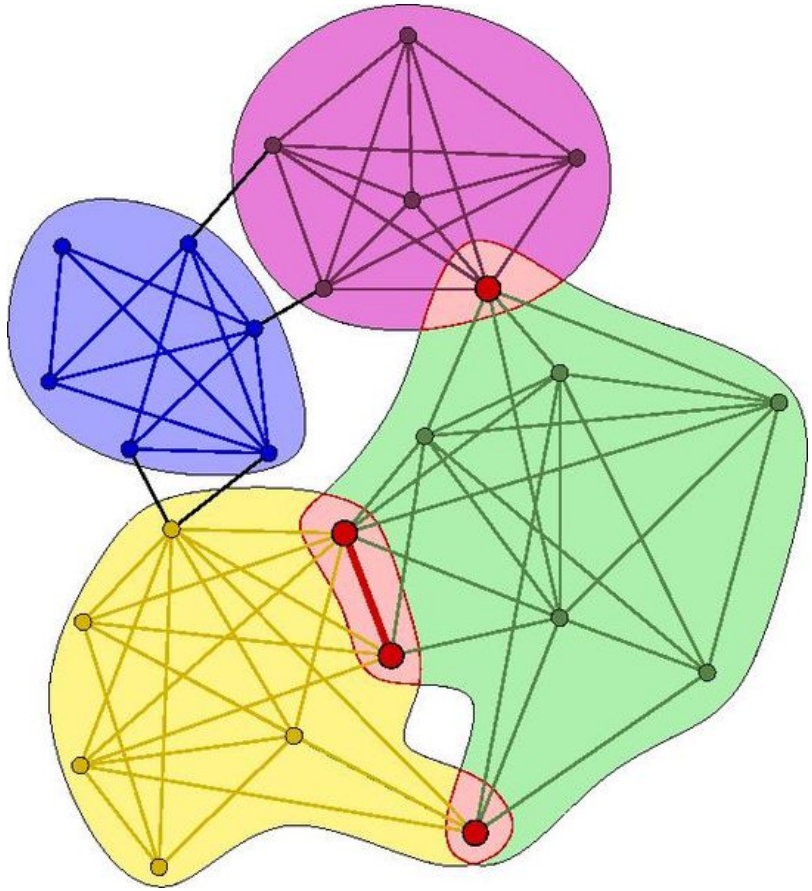
Nodes: Proteins
Edges: Physical interactions

Overlapping Communities

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

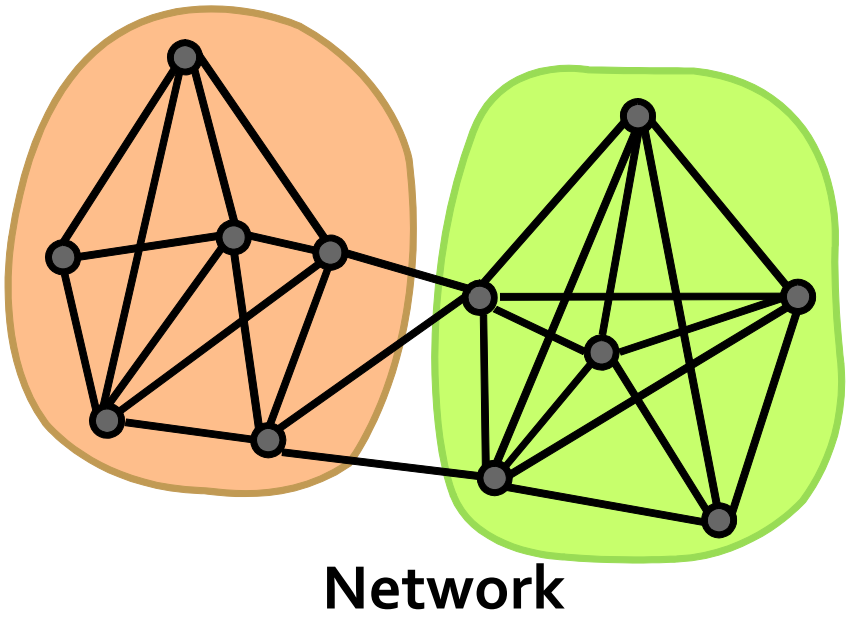


Previous lecture

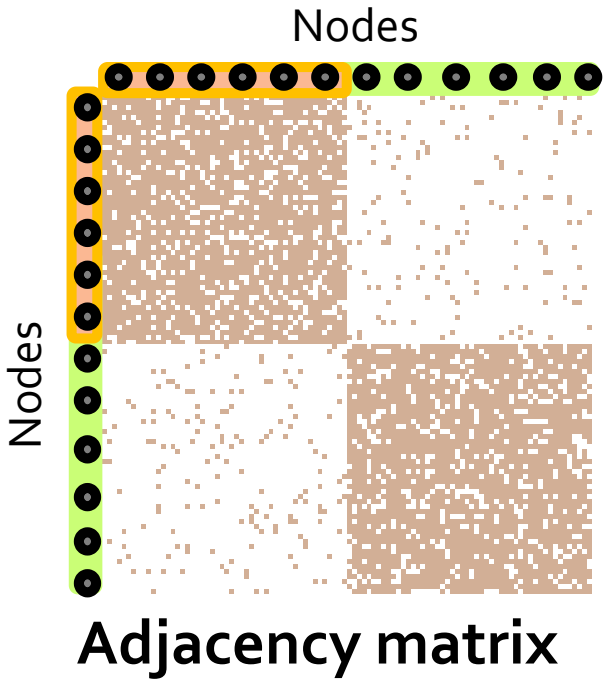


Today

Non-overlapping Communities



Network

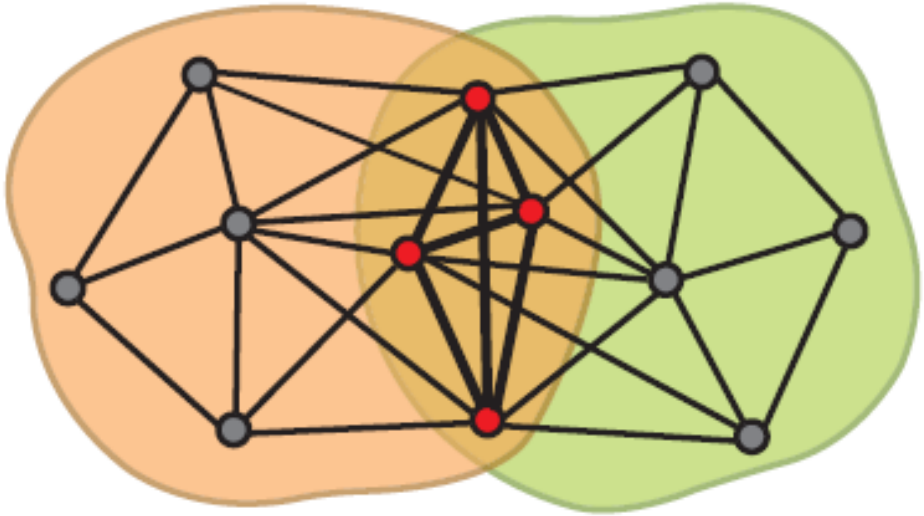


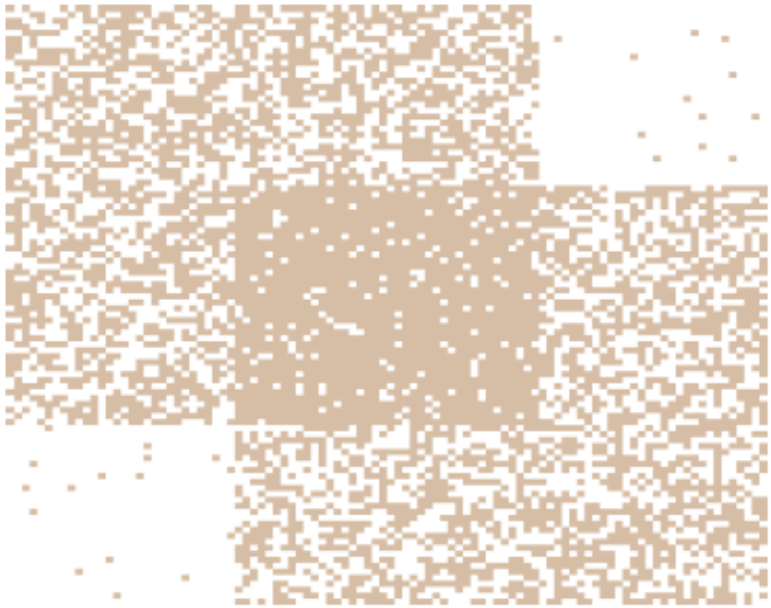
Adjacency matrix

Finding good "cuts"

Communities as Tiles!

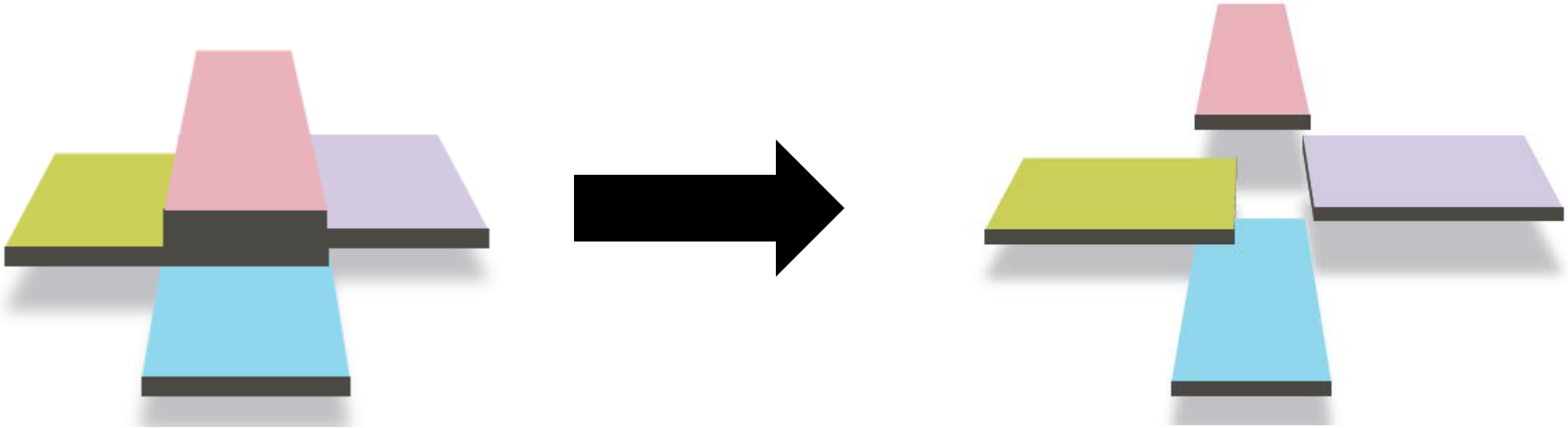
What if communities overlap?





Communities as "tiles"

Recap so far…

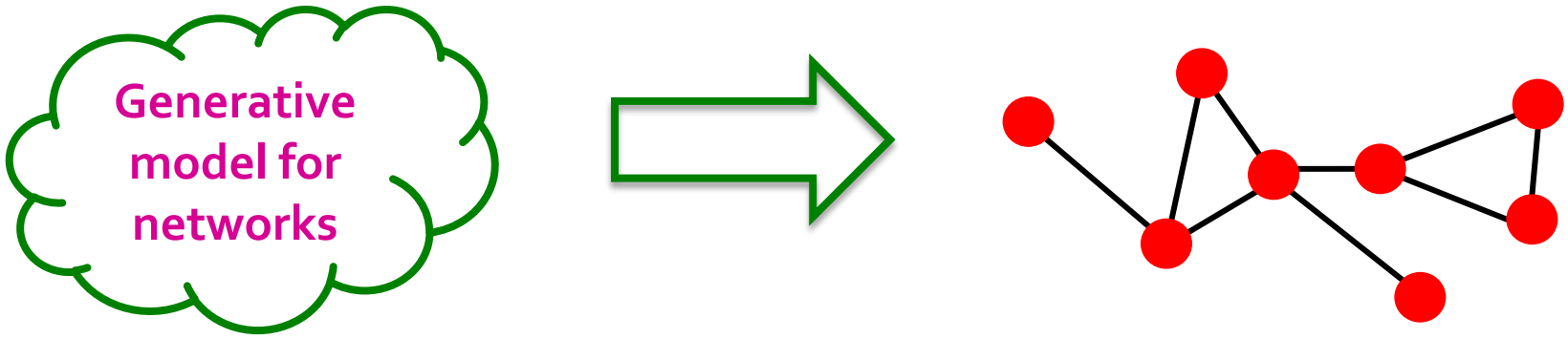


Communities in a network

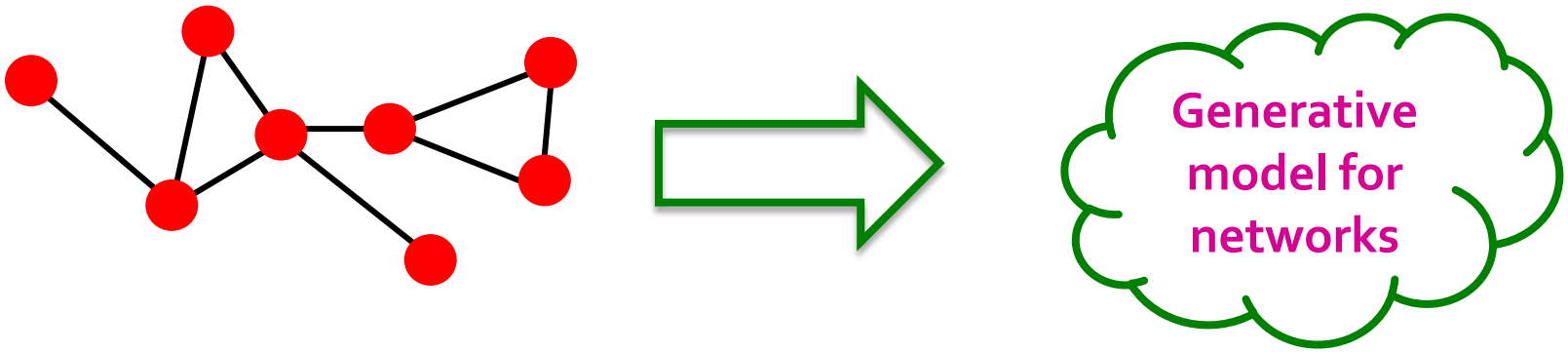
Decomposed network

Today's lecture: Plan of attack

(1) Given a model, we generate the network:

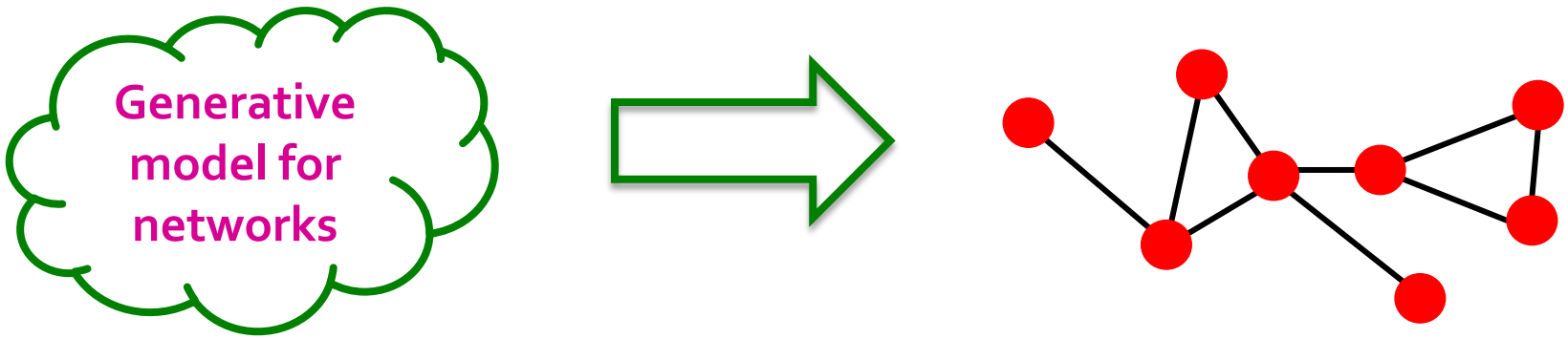


(2) Given a network, find the "best" model



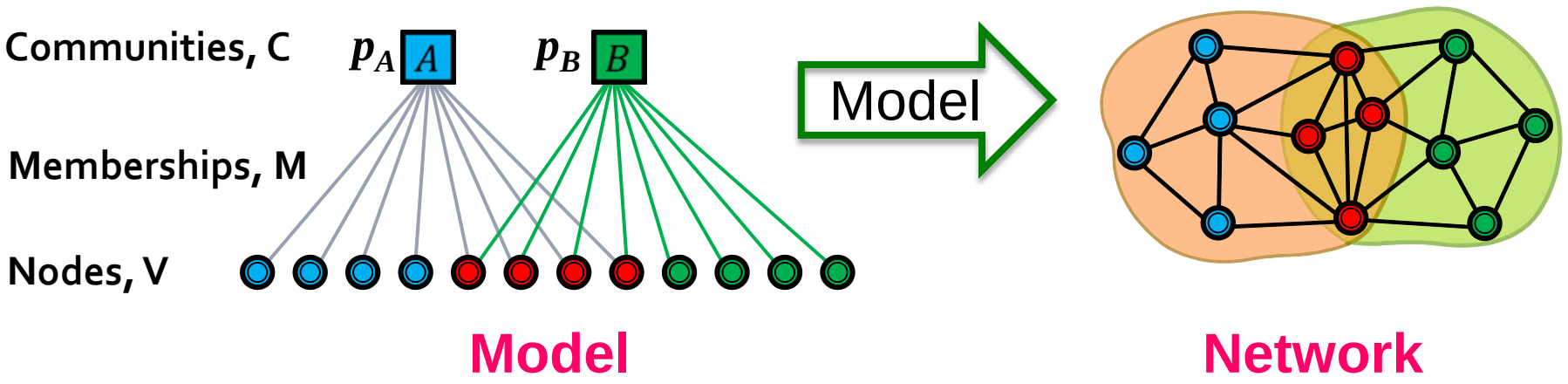
Model of networks

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



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

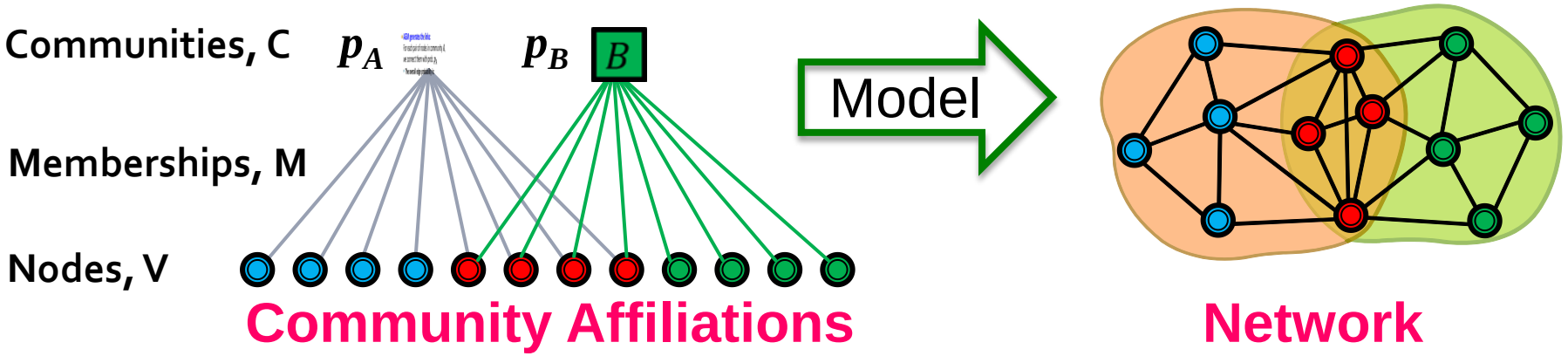
Community-Affiliation Graph



- **Generative model B(V, C, M, $\{p_c\}$) for graphs:**
  - Nodes **V**, Communities **C**, Memberships **M**
  - Each community **A** has a single probability $\mathbf{p_A}$

(Later we fit the model to networks to detect communities, that is, for each node find communities it belongs to)

AGM: Generative Process



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

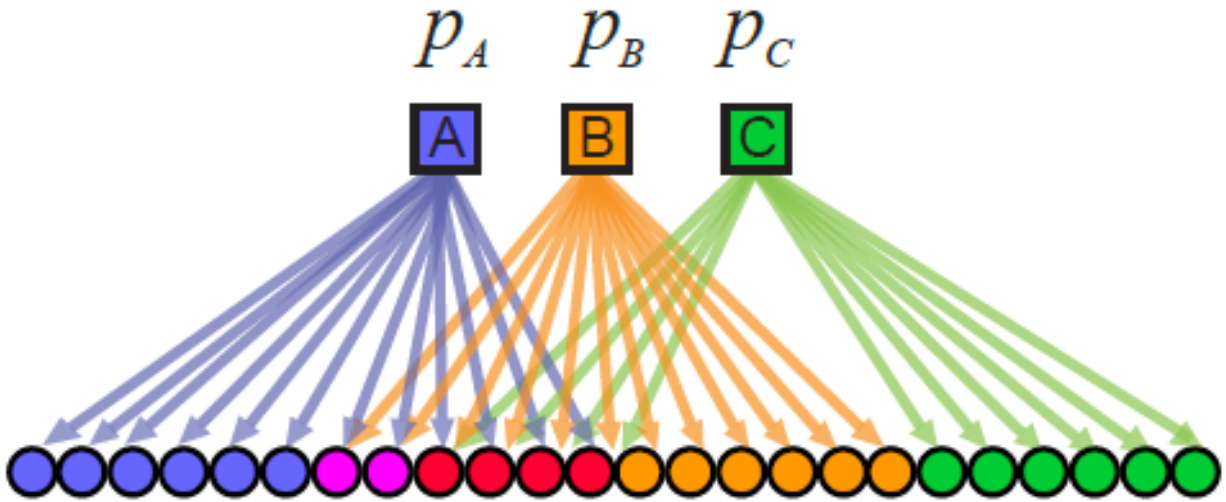
$$P(u,v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

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

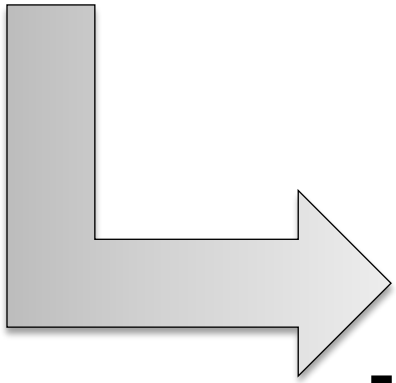
$\boldsymbol{M_u}$ … set of communities node $\boldsymbol{u}$ belongs to

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

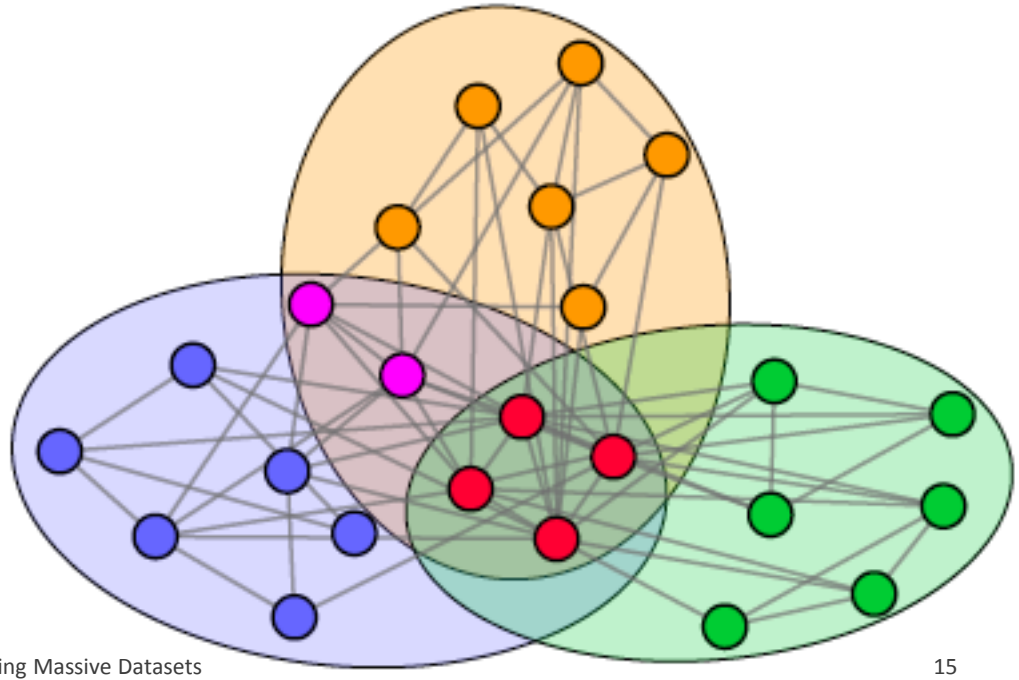
Recap: AGM networks



Model

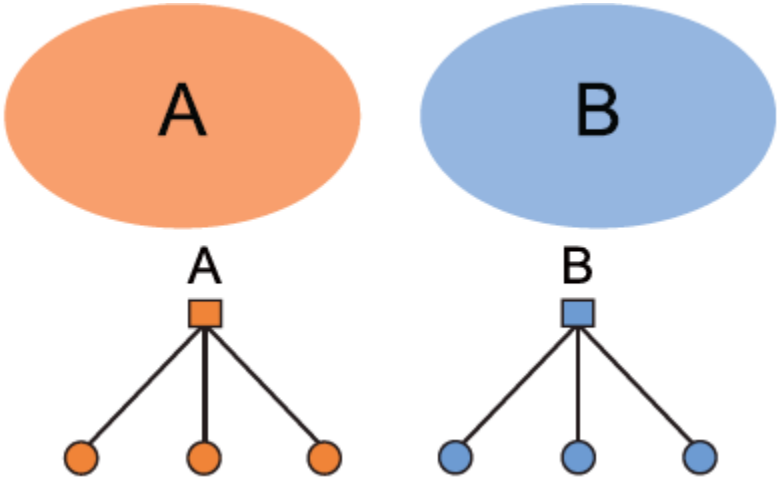


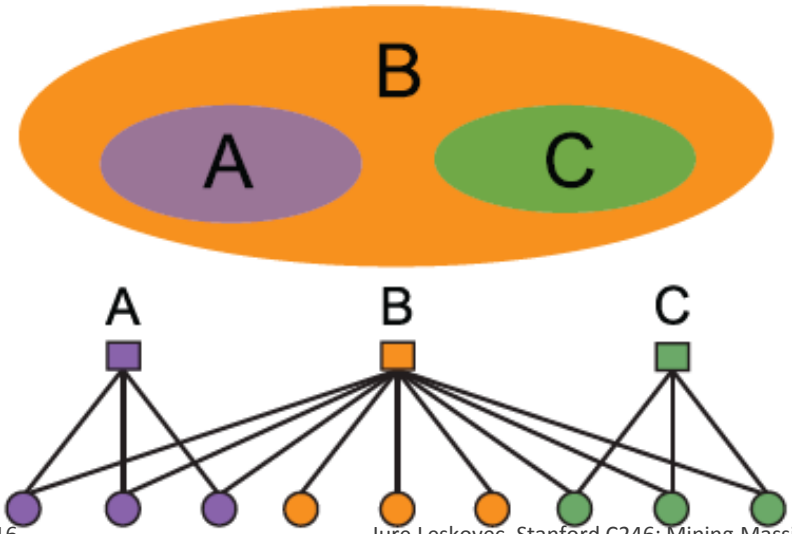
Network

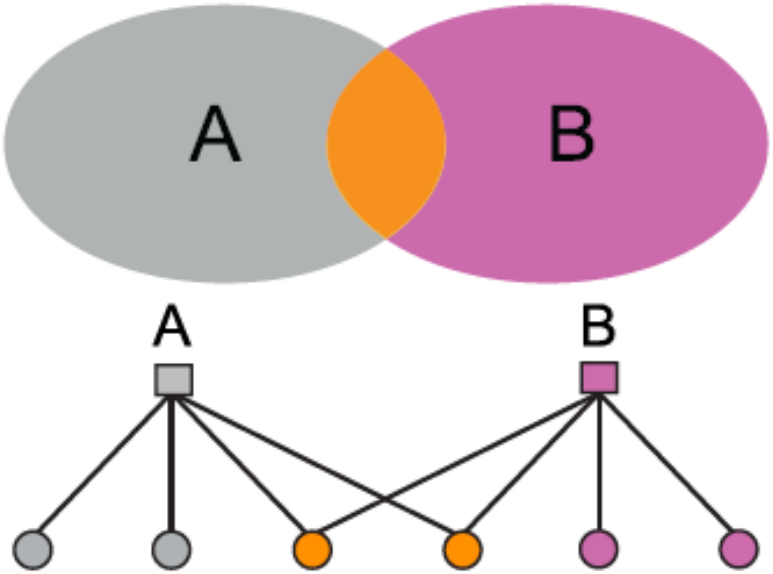


AGM is Expressive

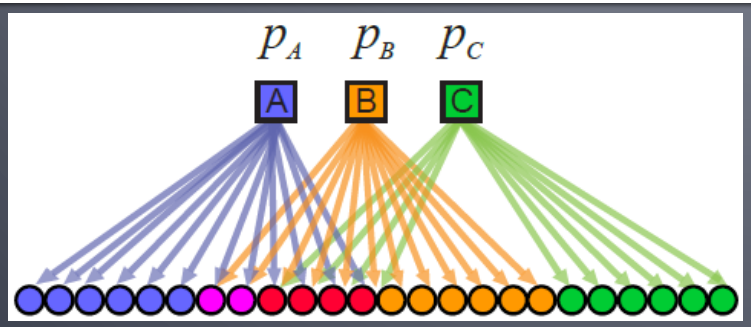
- **AGM can express a variety of community structures:**
  Non-overlapping,
  Overlapping, Nested

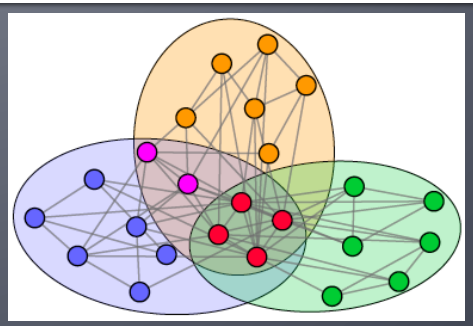






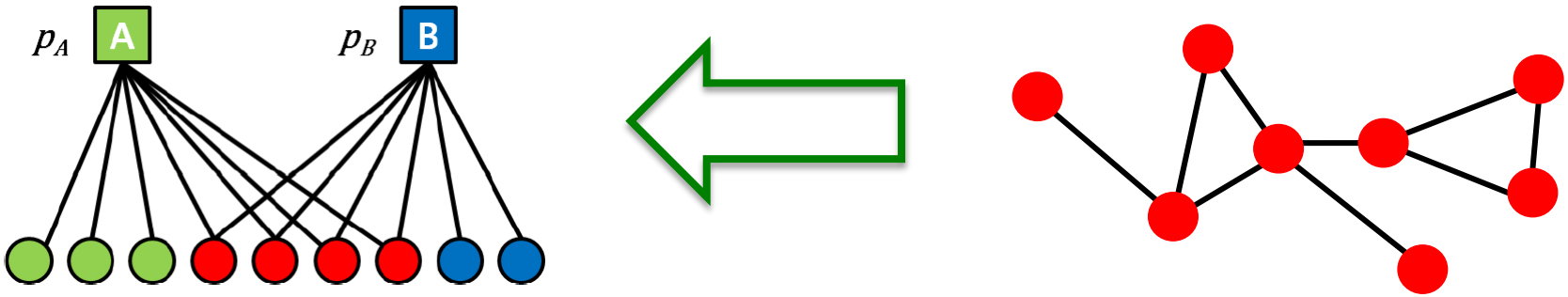
How do we detect communities with AGM?





Detecting Communities

Detecting communities with AGM:



Given a Graph $G(V, E)$, find the AGM

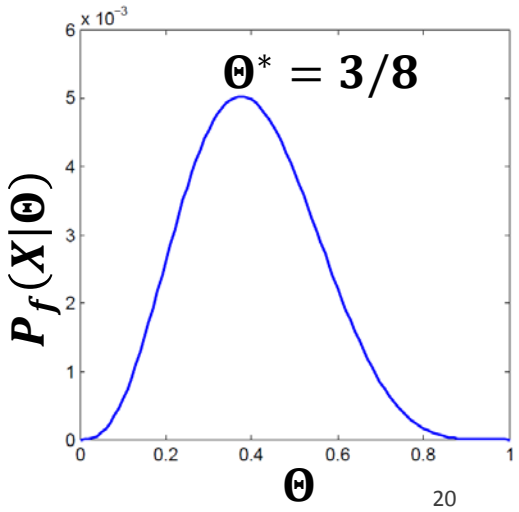
1) Affiliation graph ***M***
2) Number of communities **C**
3) Parameters $\boldsymbol{p_c}$

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 $\boldsymbol{X}$
  - **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 out 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. $\boldsymbol{\Theta}$, else return 0
  - **What is $\boldsymbol{P_f(X|\Theta)}$?** Assuming coin flips are independent
    - So, $\boldsymbol{P_f(X|\Theta) = P_f(1|\Theta) * P_f(0|\Theta) * P_f(0|\Theta) \ldots * P_f(1|\Theta)}$
      - What is $\boldsymbol{P_f(1|\Theta)}$? **Simple,** $\boldsymbol{P_f(1|\Theta) = \Theta}$
    - **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 is $\boldsymbol{\arg\max_\Theta P_f(X|\Theta)}$?** $\boldsymbol{\Theta = 3/8}$
    - Our data was generated by a coin with bias 3/8



MLE for Graphs

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

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

0	0.10	0.10	0.04
0.10	0	0.02	0.06
0.10	0.02	0	0.06
0.04	0.06	0.06	0

Flip biased coins

G

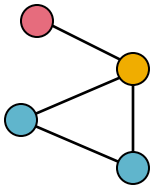
0	1	0	0
1	0	1	1
0	1	0	1
0	1	1	0

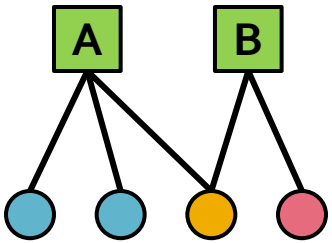
- **The likelihood of AGM generating graph G:**

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

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

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





Θ=B(V, C, M, $\{p_c\}$)

0	0.9	0.9	0
0.9	0	0.9	0
0.9	0.9	0	0.9
0	0	0.9	0

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

G

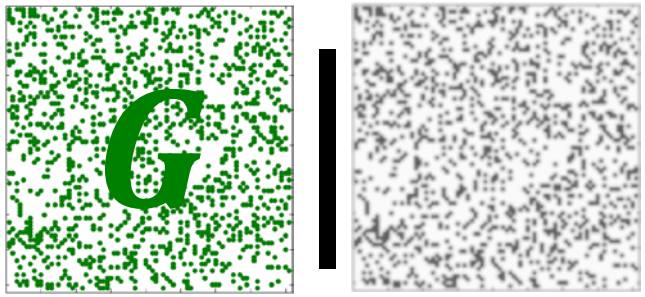
$P(G|\Theta)$

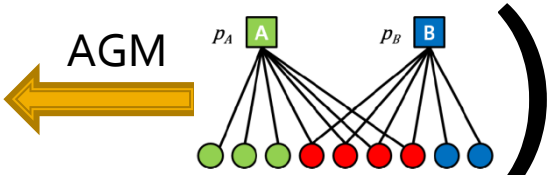
$$P(G \mid \Theta) = \prod_{(u,v)\in E} P(u,v) \prod_{(u,v)\notin E} (1 - P(u,v))$$

MLE for Graphs

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

$$\arg\max_{\Theta} P(\; G \; | \; \; \text{AGM} \; \; p_A \; A \quad p_B \; B \;)$$





$$P(G|\Theta) = \prod_{u,v} P(u,v)^{G_{uv}} \big(1 - P(u,v)\big)^{1-G_{uv}}$$

- Often we take the logarithm of the likelihood, and call it **log-likelihood**: $l(\Theta) = \log P(G|\Theta)$

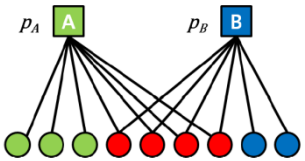
$$l(G|\Theta) = \sum_{(u,v)\in E} \log(P(u,v)) + \sum_{(u,v)\notin E} \log(1 - P(u,v))$$

MLE for AGM

- **Our goal is to find $B(V, C, M, \{p_C\})$ such that:**

$$\arg\max_{B(V,C,M,\{p_C\})} \sum_{u,v \in E} log P(u,v) + \sum_{u,v \notin E} log(1 - P(u,v))$$

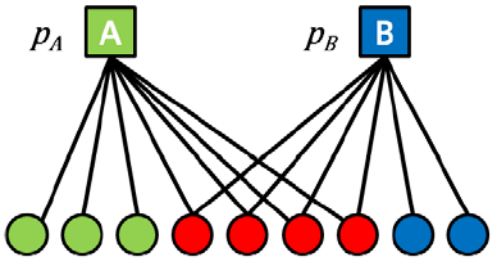
- **Problem:** Finding **B** means finding the bipartite affiliation network

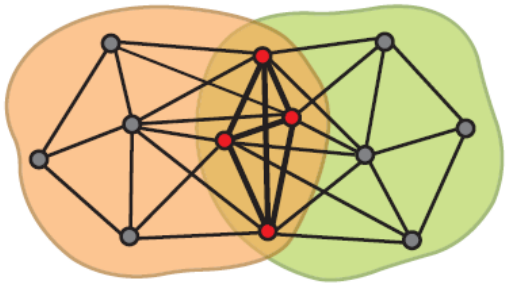


  - **There is no nice way to do this**
  - **Fitting $B(V, C, M, \{p_C\})$ is too hard, let's change the model (so it is easier to fit)!**

MLE for AGM

- **If $B(V, C, M)$ is given, finding $\{p_C\}$ is easy:**
  - Just write down the log-likelihood $l(G|\Theta)$ as a function of $\{p_C\}$ by computing each $P(u, v)$
  - **Find $\{p_C\}$ that maximizes the log-likelihood**

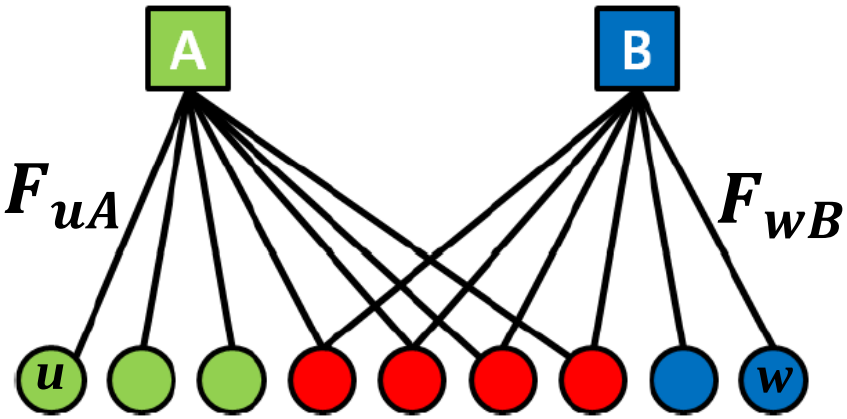




$$l(G \mid \Theta) = \sum_{(u,v)\in E} \log(P(u,v)) + \sum_{(u,v)\notin E} \log(1 - P(u,v))$$

From AGM to BigCLAM

- **Relaxation: Memberships have strengths**

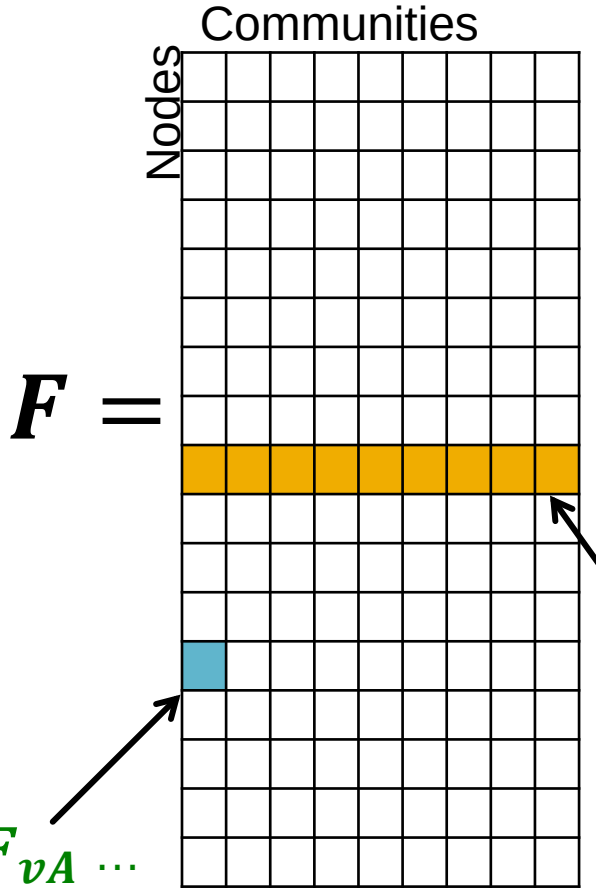


  - F_{uA} (≥ 0): The membership strength of node u to community A (if $F_{uA} = 0$ then no membership)
  - **Each community A links nodes independently:**

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

Factor Matrix F

- **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**
    - **Notice:** If one of the nodes doesn't belong to the community A ($F_{uA} = 0$) then $P_A(u,v) = 0$
- Prob. that **at least one** common community C links the two 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:

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

- **Example F matrix:**

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: $P(u,w) = 0.88$

$P(v,w) = 0$

BigCLAM: How to find F

- **Task:** Given a network $G(V, E)$, estimate F
  - Find F that maximizes the log-likelihood $l(F)$:

$$\arg\max_{F} \sum_{u,v \in E} \boldsymbol{log P(u,v)} + \sum_{u,v \notin E} \boldsymbol{log(1 - P(u,v))}$$

  - where: $\boldsymbol{P(u,v) = 1 - \exp\left(-F_u \cdot F_v^T\right)}$

- **Goal:** Find F that maximizes $l(F)$:

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

BigCLAM: V1.0

$$l(F) = \sum_{u,v \in E} logP(u,v) + \sum_{u,v \notin E} log(1 - P(u,v))$$

- Can rewrite $\boldsymbol{l(F) = \frac{1}{2}\sum_{u \in V} l(F_u)}$ where

$$l(F_u) = \sum_{v \in \mathcal{N}(u)} \log(P(u,v)) + \sum_{v \notin \mathcal{N}(u)} log(1 - P(u,v))$$

Summing over all the edges is equivalent to summing over all the nodes and then over the neighbors $\mathcal{N}$ of each node. ½ is since we count every edge twice.

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

$\boldsymbol{\mathcal{N}(u)}$.. Set out-outgoing neighbors

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:**

$\mathcal{N}(u)$.. Set out outgoing neighbors

  - **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!

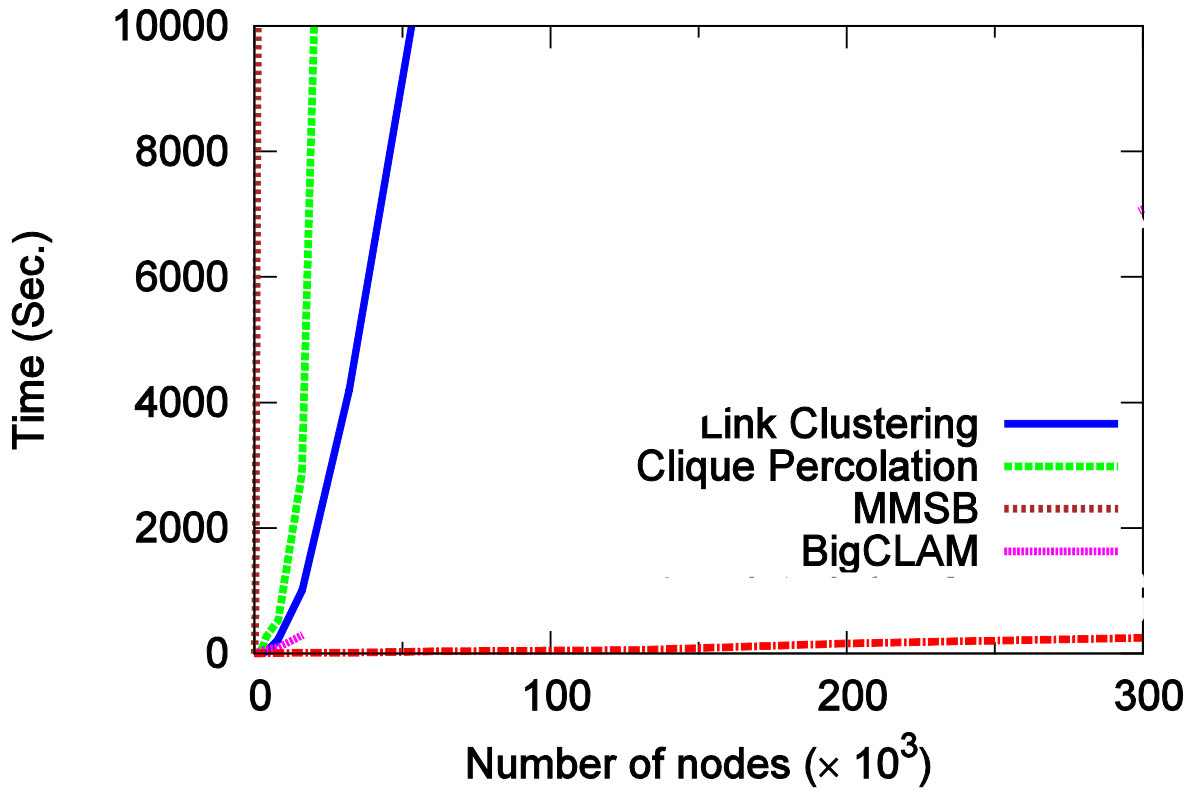
BigCLAM: V2.0

- **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 cache $\sum_v \boldsymbol{F_v}$
    - Note $\sum_v \boldsymbol{F_v}$ changes during each gradient descent step. But we cache it and update it only every so often (say every N steps).
  - So, computing $\sum_{v \notin \mathcal{N}(u)} \boldsymbol{F_v}$ now takes **linear time** in the degree $|\mathcal{N}(\boldsymbol{u})|$ of node $\boldsymbol{u}$
    - 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

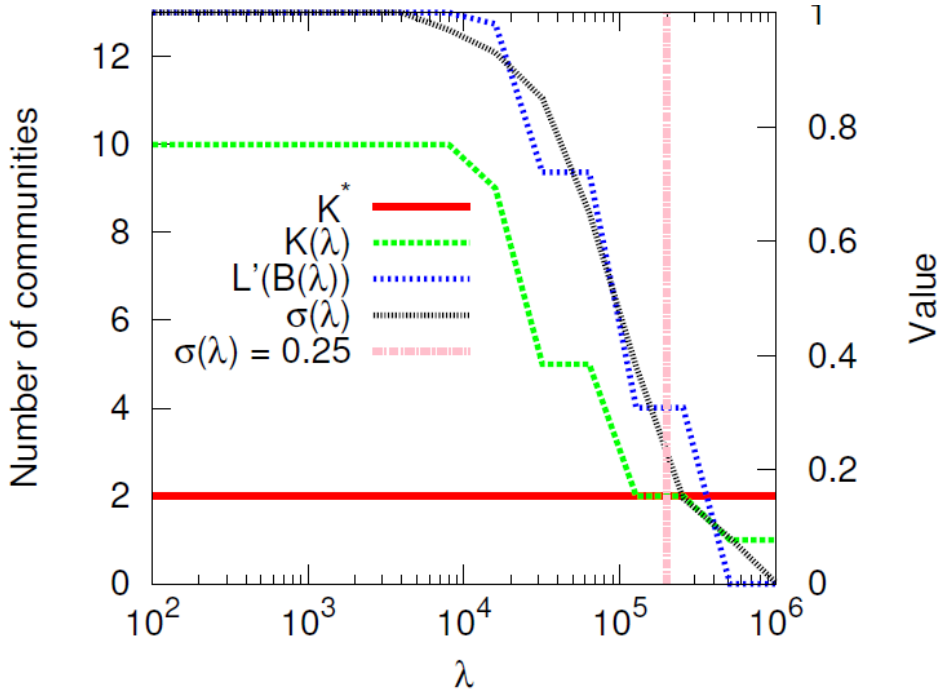


- **BigCLAM takes 5 minutes for 300k node nets**
  - **Other methods take 10 days**
- **Can process networks with 100M edges!**

Number of Communities

How to determine the number of communities?

- Use regularization to tune λ
- Solve: $\{\hat{p}_c(\lambda)\} = \underset{\{p_c\}}{\operatorname{argmax}} P(G|B_0, \{p_c\}) - \lambda \sum_c |p_c|$



Search for minimum number of communities (largest λ) where the log-likelihood is still high:
$L'(B(\lambda)) = \log P(G|B, \{p_c\})$
$K(\lambda)$ = #communities detected by AGM at value λ

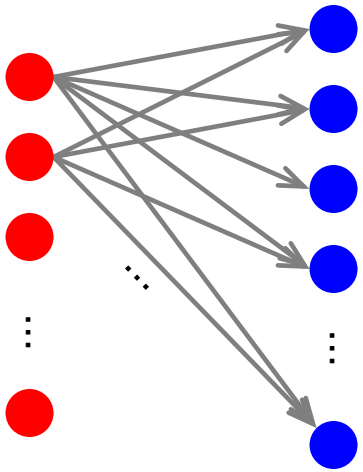
Details in http://cs.stanford.edu/people/jure/pubs/agmfit-icdm12.pdf

Analysis of Large Graphs: Trawling

[Kumar et al. '99]

Trawling

- **Searching for small communities in the Web graph**
- **What is the signature of a community / discussion in a Web graph?**



Dense 2-layer graph

Use this to define "topics":
What the same people on the left talk about on the right
Remember HITS!

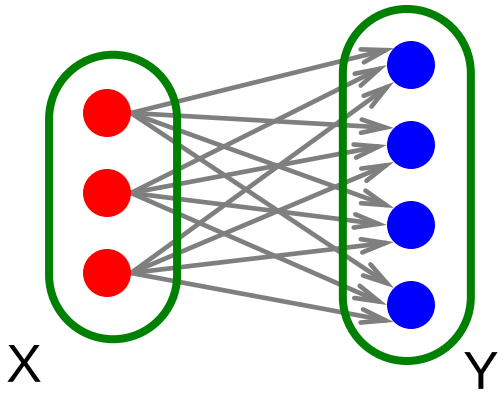
Intuition: Many people all talking about the same things

Searching for Small Communities

- **A more well-defined problem:**

  Enumerate complete bipartite subgraphs $K_{s,t}$

  - Where $K_{s,t}$: s nodes on the "left" where each links to the same t other nodes on the "right"



$K_{3,4}$

$|X| = s = 3$
$|Y| = t = 4$

Fully connected

[Agrawal-Srikant '99]

Frequent Itemset Enumeration

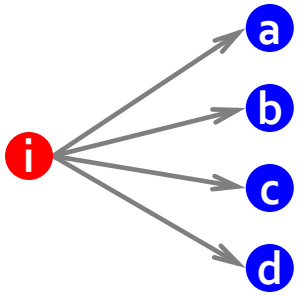
- **Market basket analysis.** Setting:
  - **Market:** Universe U of n items
  - **Baskets:** m subsets of U: $S_1, S_2, \ldots, S_m \subseteq U$ (S_i is a set of items one person bought)
  - **Support:** Frequency threshold f
- **Goal:**
  - **Find all subsets T s.t. $T \subseteq S_i$ of at least f sets S_i** (items in T were bought together at least f times)
- **What's the connection between the itemsets and complete bipartite graphs?**

[Kumar et al. '99]

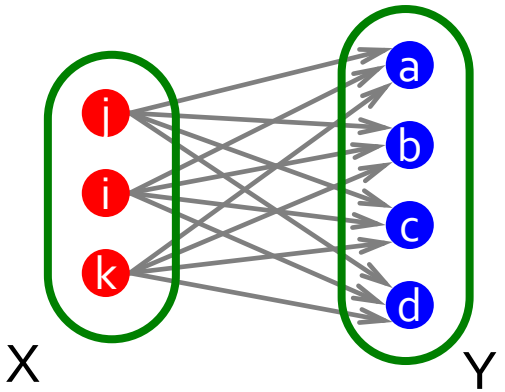
From Itemsets to Bipartite $K_{s,t}$

Frequent itemsets = complete bipartite graphs!

- **How?**
  - View each node i as a set S_i of nodes i points to
  - $K_{s,t}$ = a set Y of size t that occurs in s sets S_i
  - Looking for $K_{s,t}$ → set of frequency threshold to s and look at layer t – all frequent sets of size t



S_i={a,b,c,d}

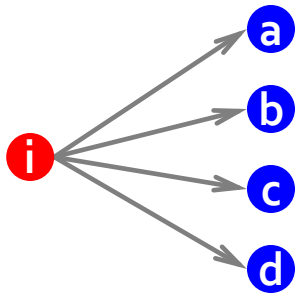


s … minimum support (|X|=s)
t … itemset size (|Y|=t)

[Kumar et al. '99]

From Itemsets to Bipartite $K_{s,t}$

View each node ***i*** as a set S_i of nodes ***i*** points to



S_i={a,b,c,d}

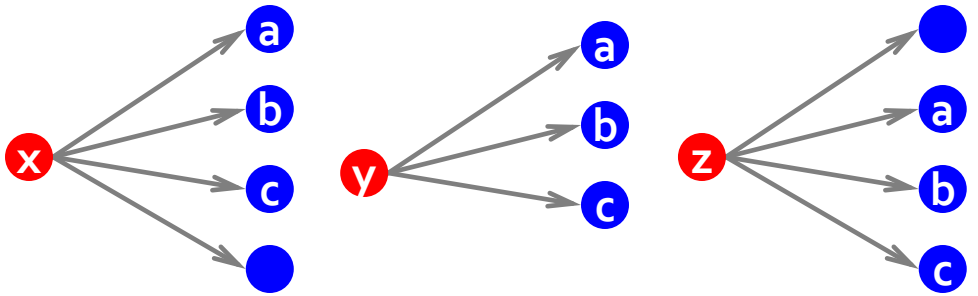
Find frequent itemsets:

s … minimum support

t … itemset size

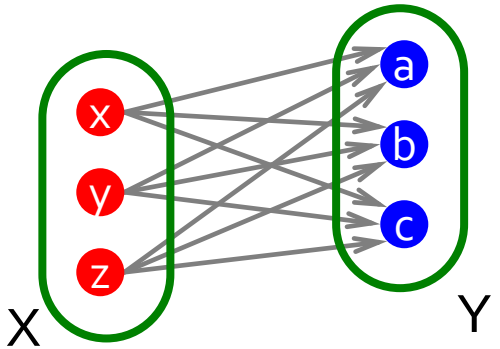
Say we find a **frequent itemset *Y={a,b,c}*** of supp ***s***

So, there are ***s*** nodes that link to all of **{a,b,c}**:

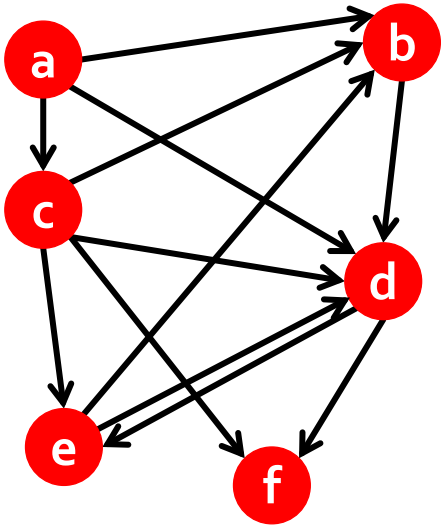


We found $K_{s,t}$!

$K_{s,t}$ = a set ***Y*** of size ***t*** that occurs in *s* sets S_i



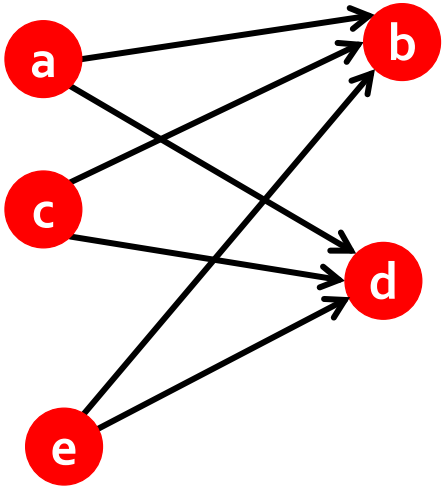
Example (1)

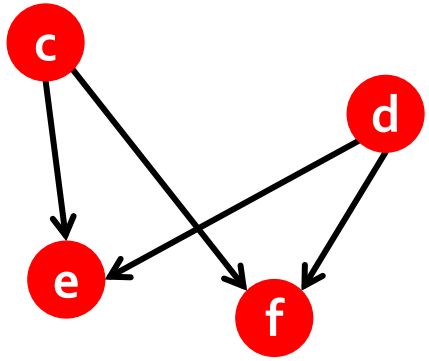


- **Support threshold s=2**
  - **{b,d}**: support 3
  - **{e,f}**: support 2
- **And we just found 2 bipartite subgraphs:**

Itemsets:

a = {b,c,d}
b = {d}
c = {b,d,e,f}
d = {e,f}
e = {b,d}
f = {}





Example (2)

- **Example of a community from a web graph**

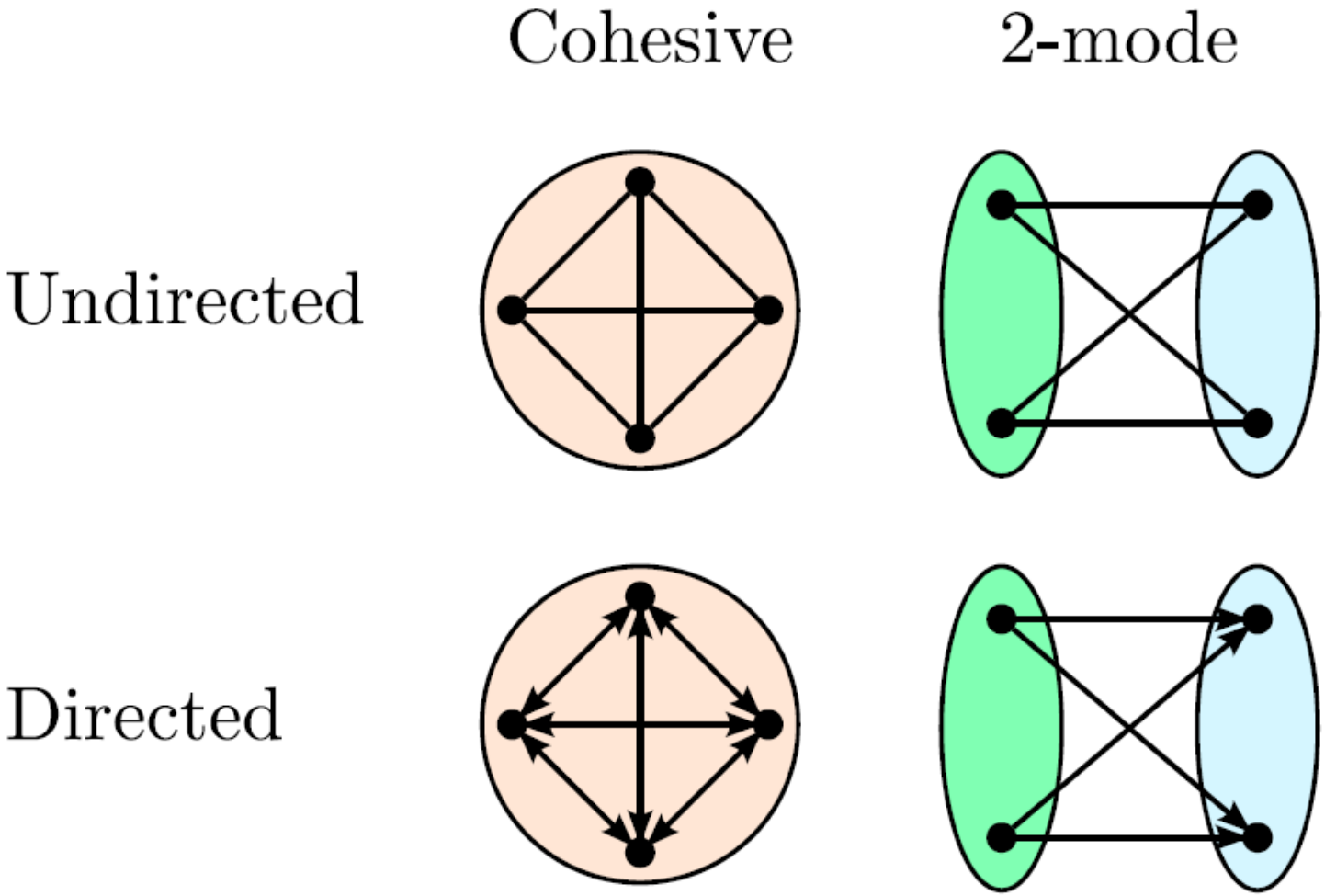
A community of Australian fire brigades

Nodes on the right	Nodes on the left
NSW Rural Fire Service Internet Site	New South Wales Fir...ial Australian Links
NSW Fire Brigades	Feuerwehrlinks Australien
Sutherland Rural Fire Service	FireNet Information Network
CFA: County Fire Authority	The Cherrybrook Rur...re Brigade Home Page
"The National Cente...ted Children's Ho...	New South Wales Fir...ial Australian Links
CRAFTI Internet Connexions-INFO	Fire Departments, F... Information Network
Welcome to Blackwoo... Fire Safety Serv...	The Australian Firefighter Page
The World Famous Guestbook Server	Kristiansand brannv...dens brannvesener...
Wilberforce County Fire Brigade	Australian Fire Services Links
NEW SOUTH WALES FIR...ES 377 STATION	The 911 F,P,M., Fir...mp; Canada A Section
Woronora Bushfire Brigade	Feuerwehrlinks Australien
Mongarlowe Bush Fire – Home Page	Sanctuary Point Rural Fire Brigade
Golden Square Fire Brigade	Fire Trails "l...ghters around the...
FIREBREAK Home Page	FireSafe – Fire and Safety Directory
Guises Creek Volunt...fficial Home Page...	Kristiansand Firede...departments of th...

[Kumar, Raghavan, Rajagopalan, Tomkins: Trawling the Web for emerging cyber-communities 1999]

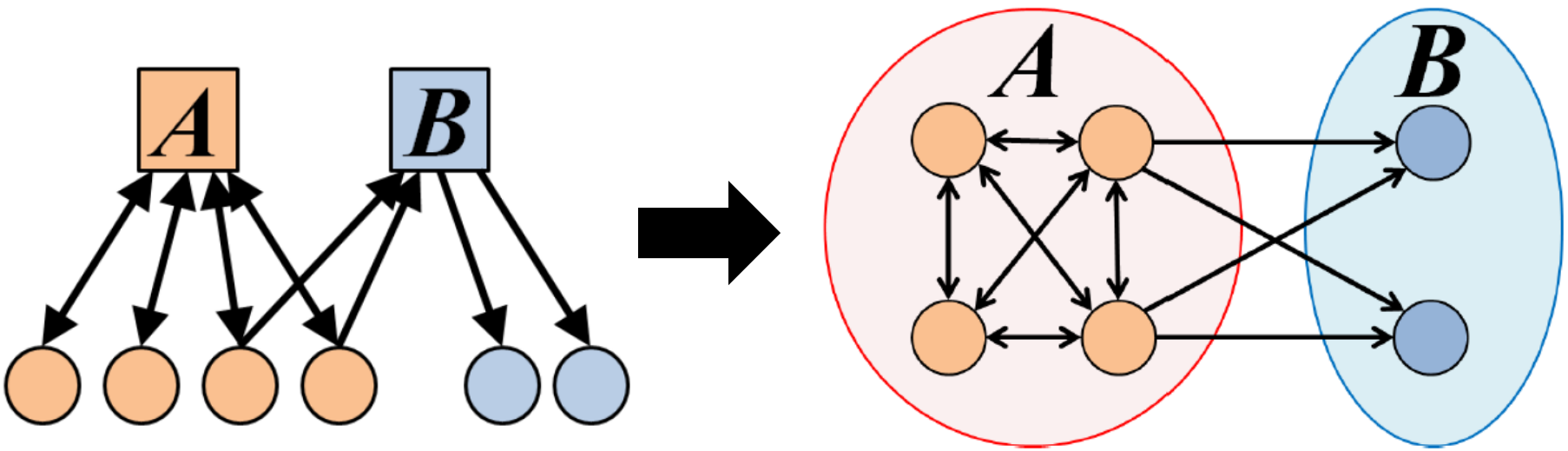
Extensions of AGM: Directed memberships

Extension: Beyond Clusters

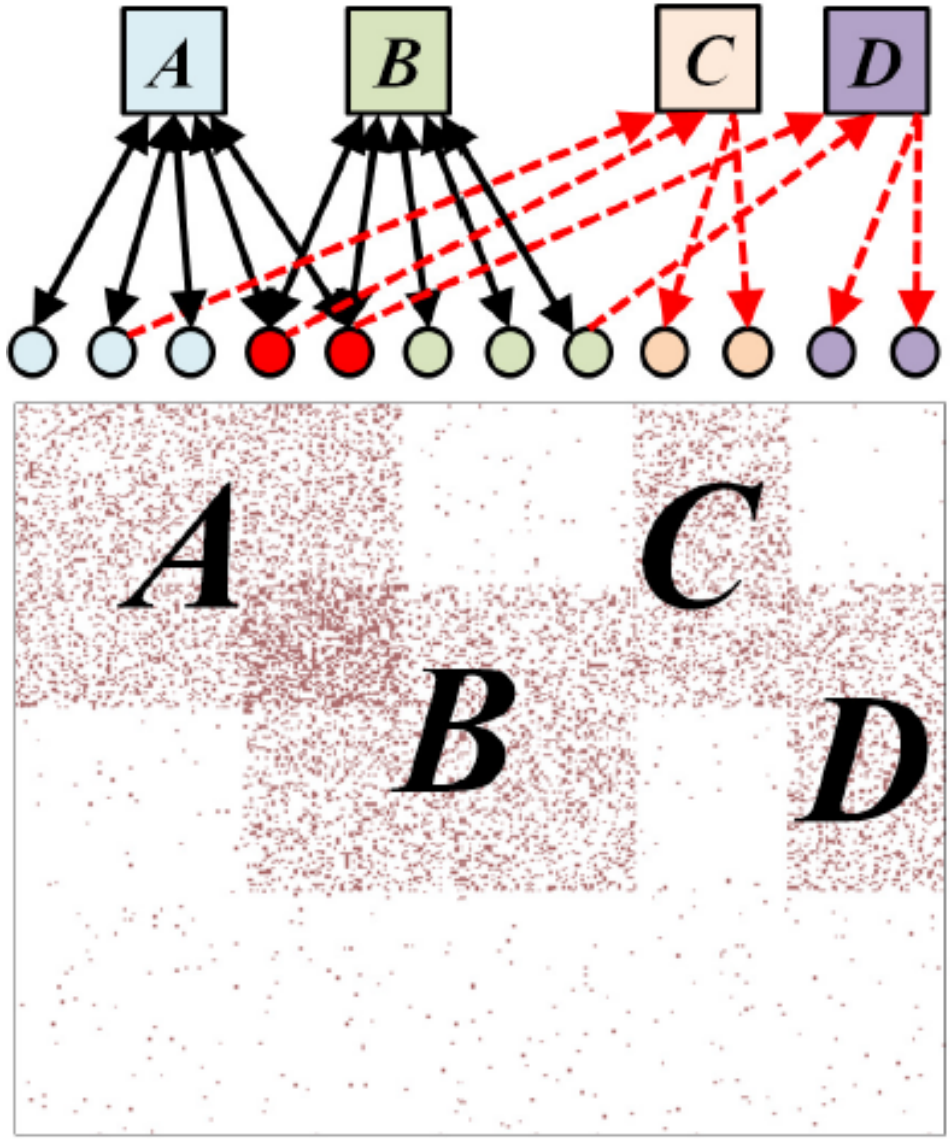


Extension: Directed AGM

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

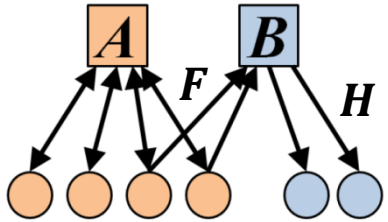


Example: Model and Network



Directed AGM

- **Everything is almost the same except now we have 2 matrices: F and H**
  - F… out-going community memberships
  - H… in-coming community memberships

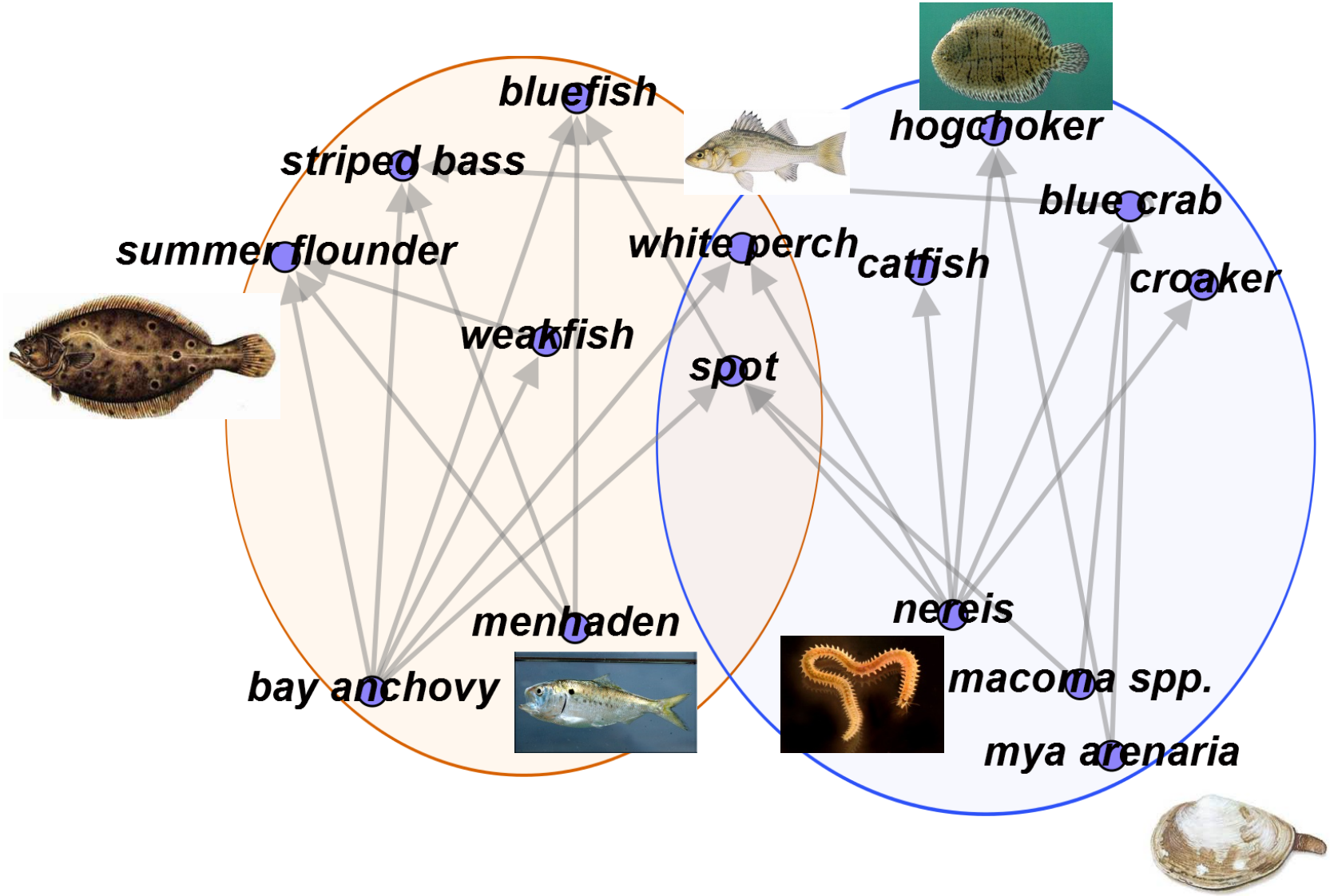


- **Edge prob.: $P(u,v) = 1 - exp(-F_u H_v^T)$**
- **Network log-likelihood:**

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

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.