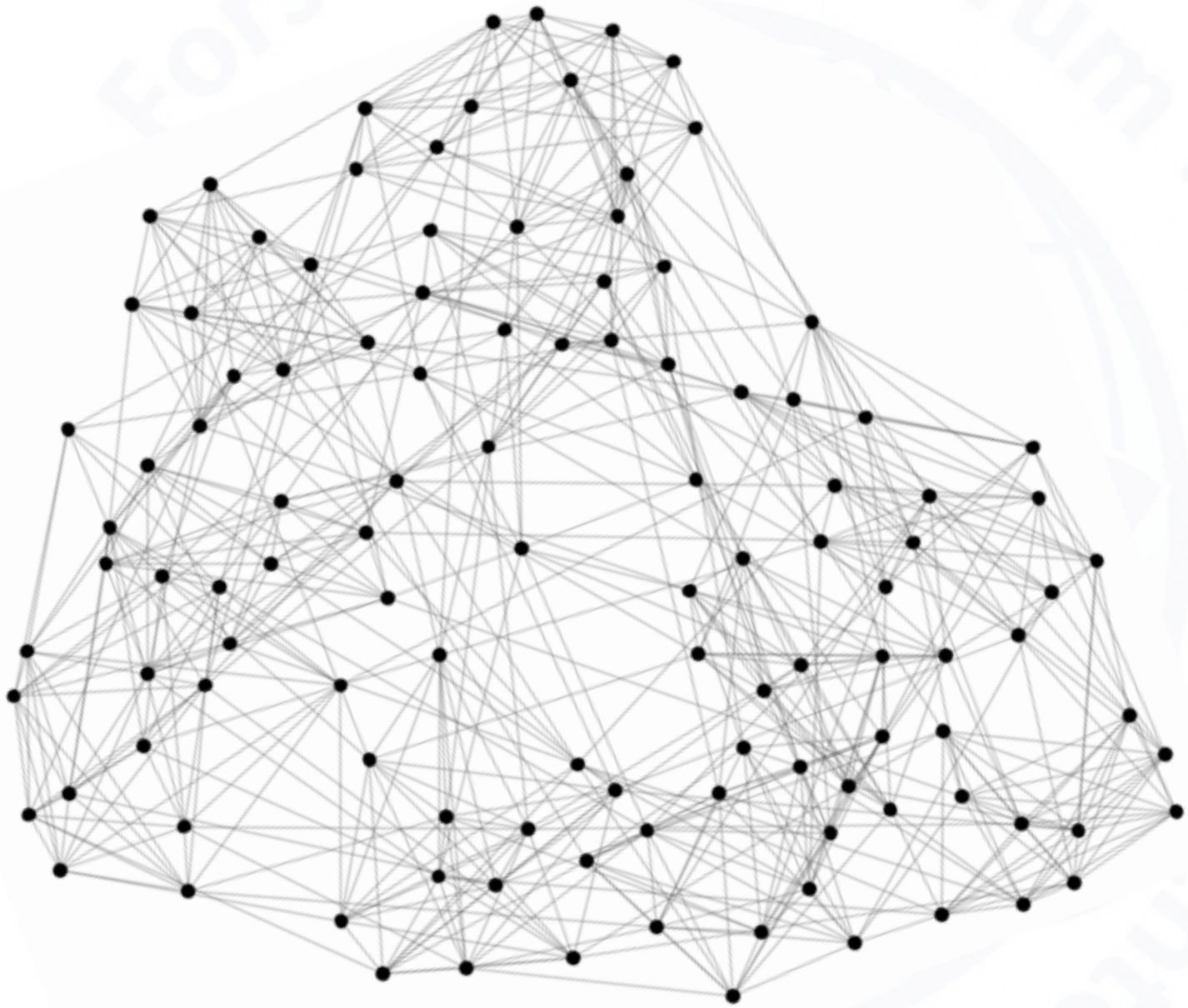
Graphs-2

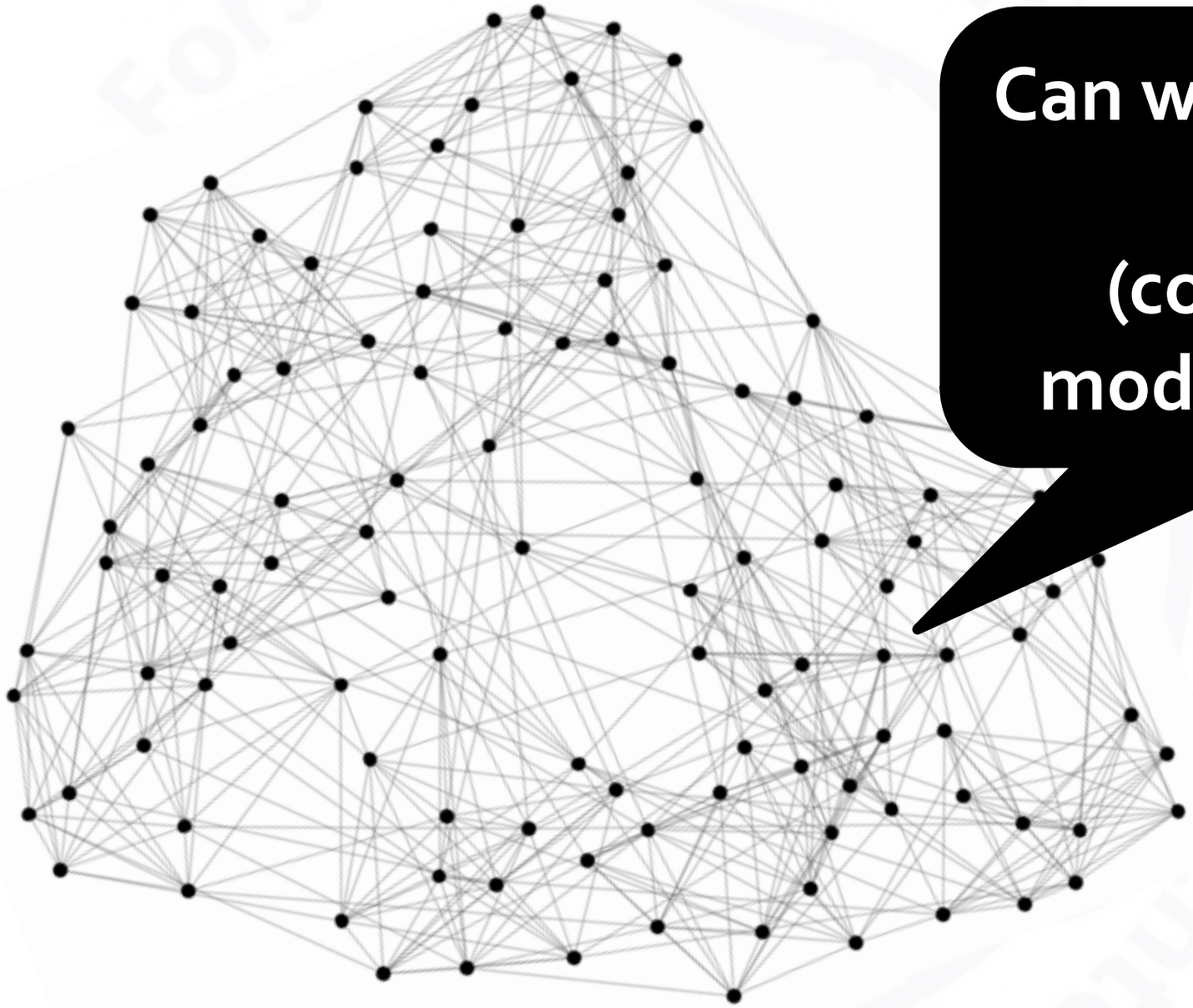
AGM, MLE on Graphs, Counting triangles

Identifying Communities



Nodes: Football Teams
Edges: Games played

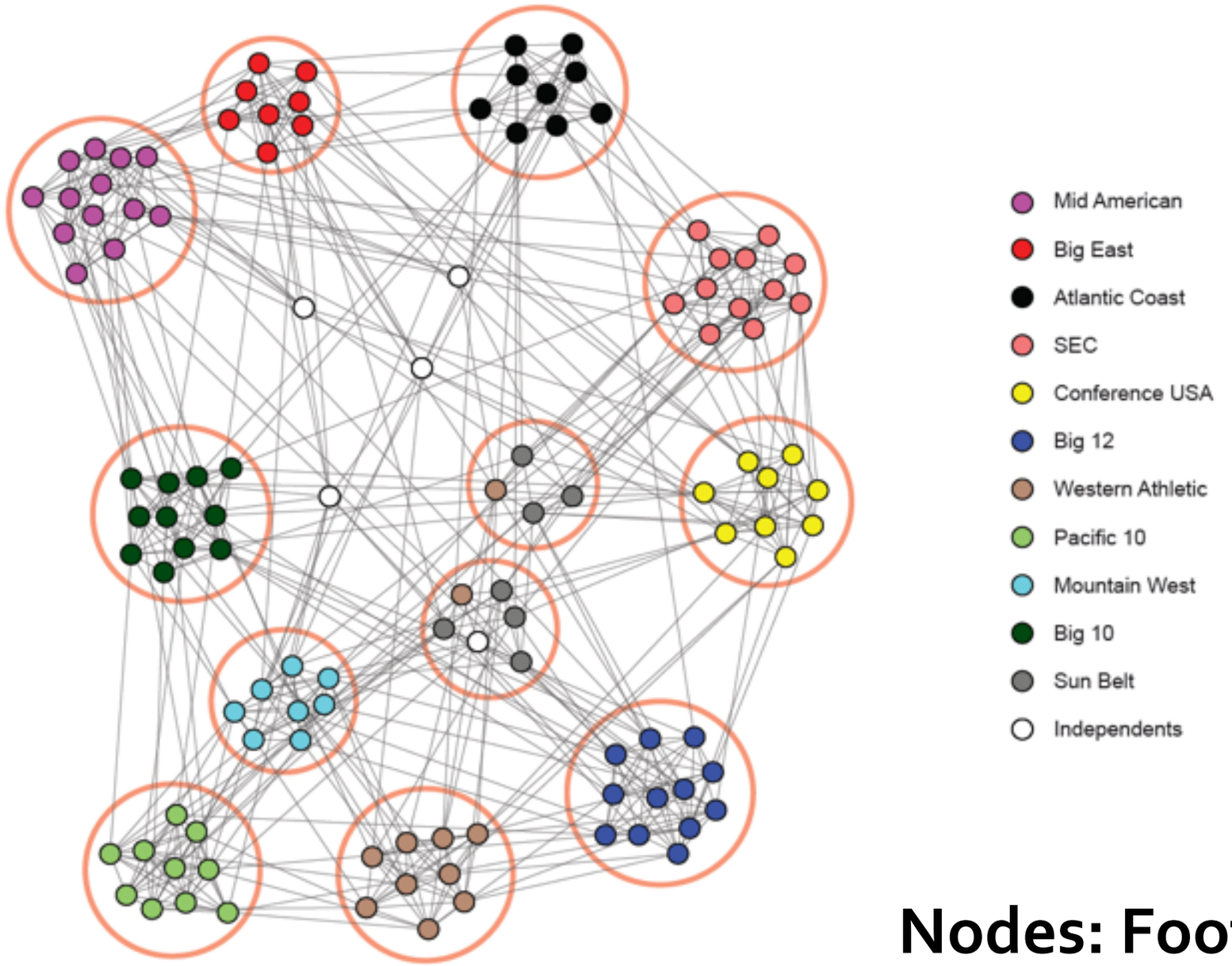
Identifying Communities



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

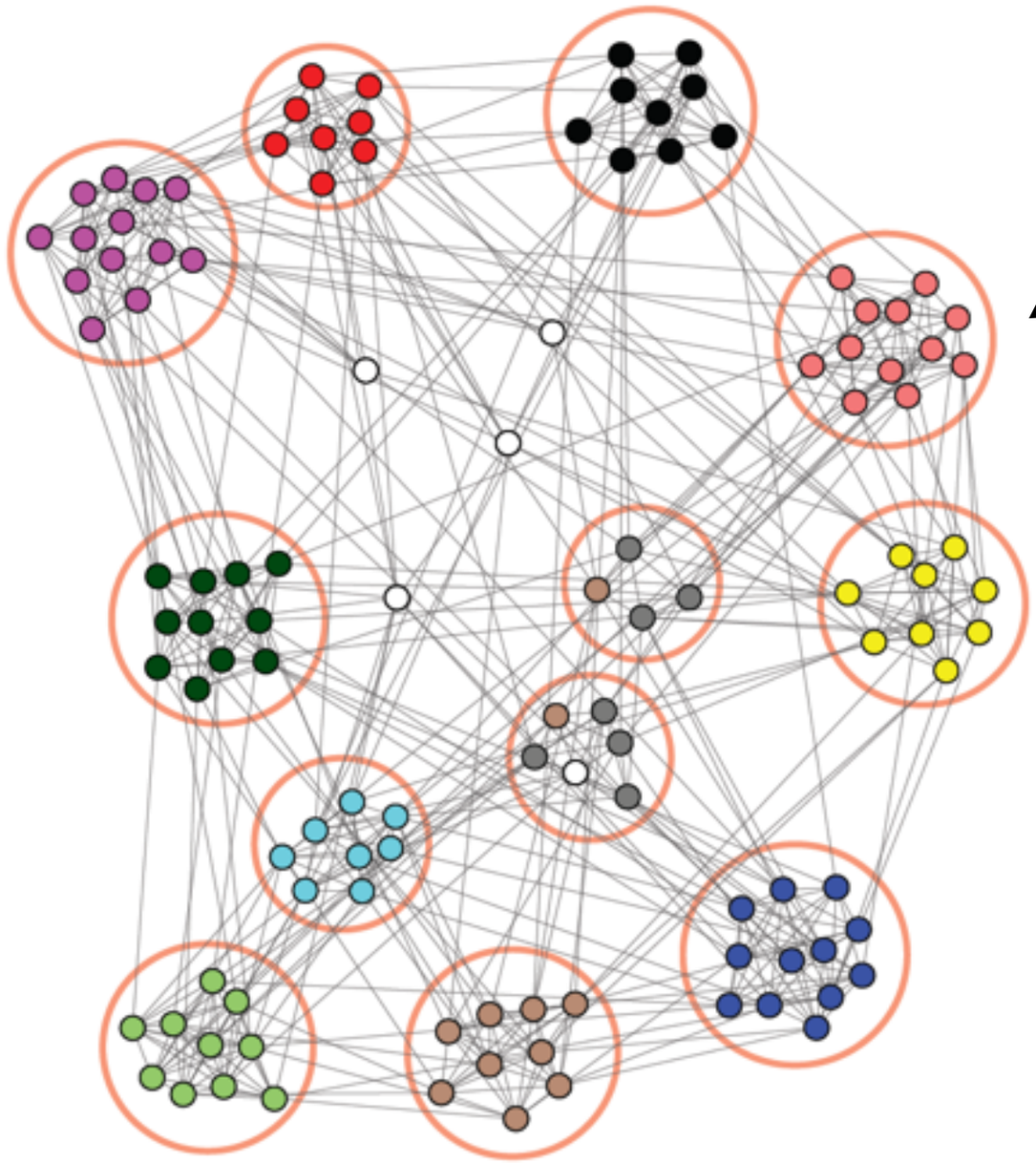
Nodes: Football Teams
Edges: Games played

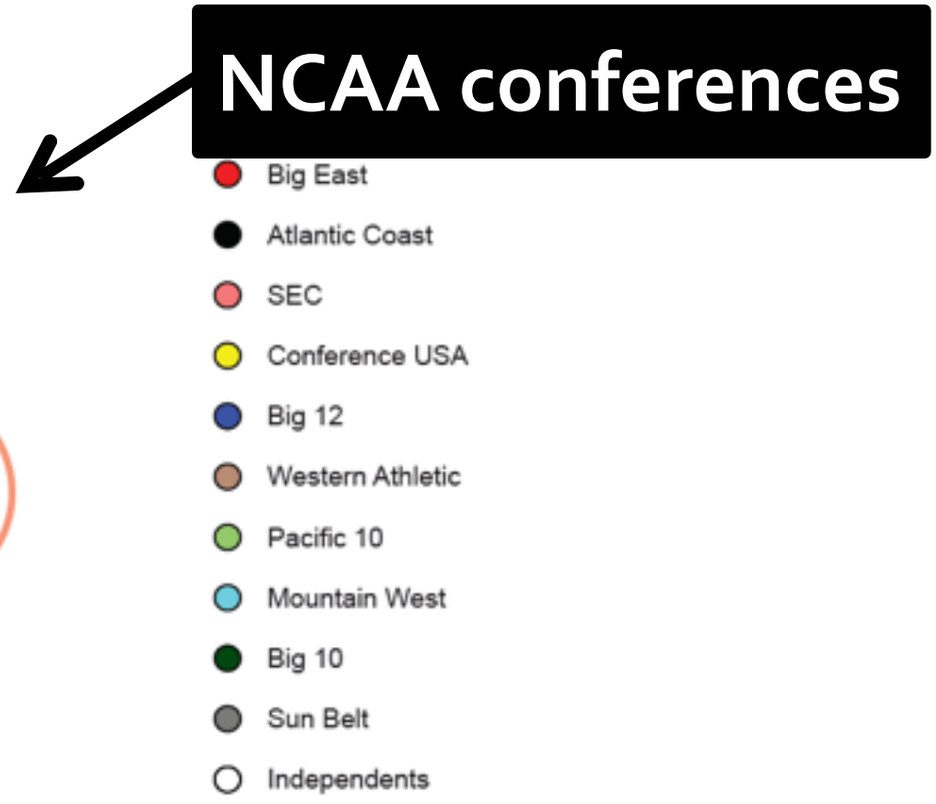
NCAA Football Network



Nodes: Football Teams
Edges: Games played

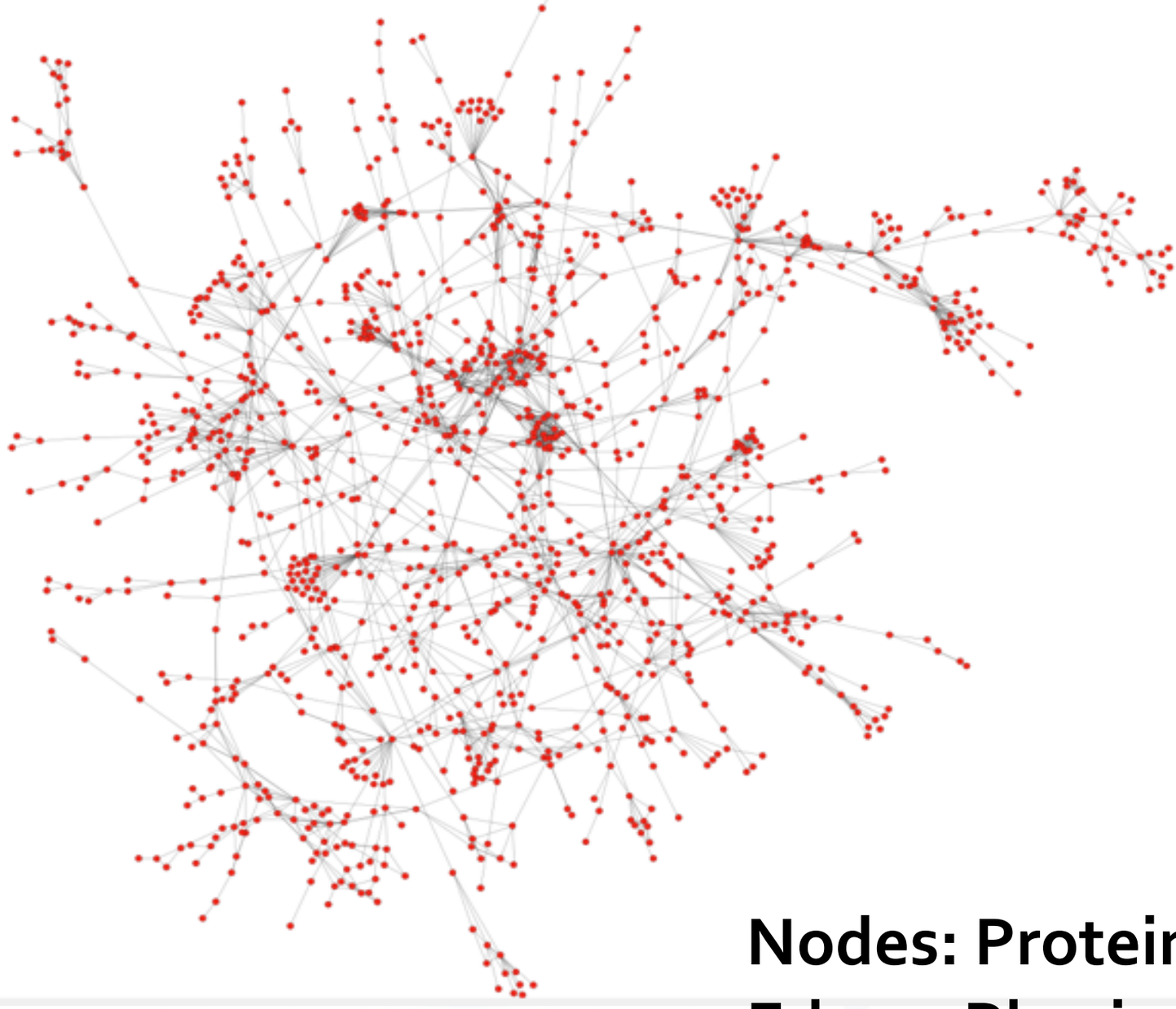
NCAA Football Network





Nodes: Football Teams
Edges: Games played

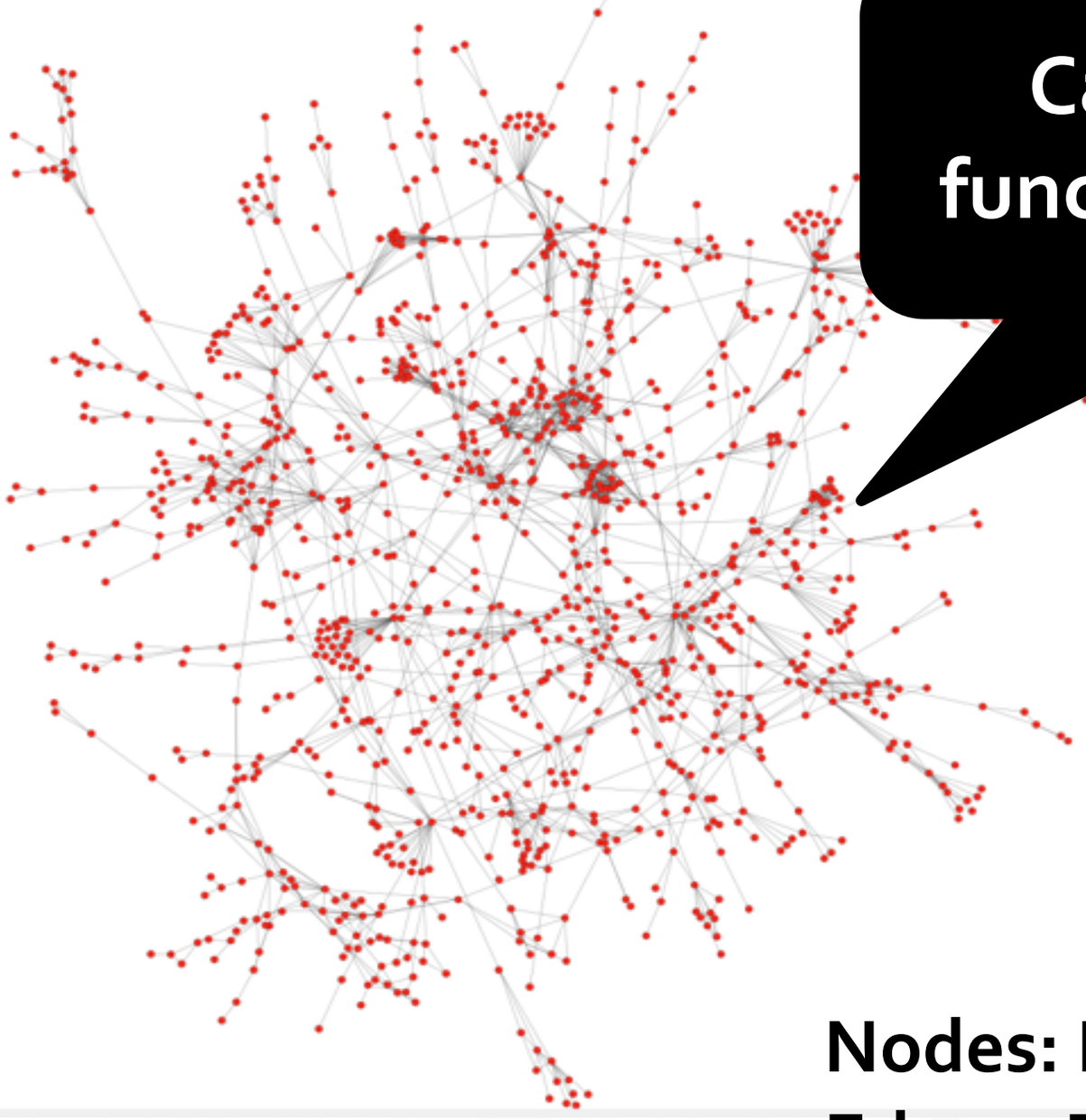
Protein-Protein Interactions



Nodes: Proteins

Edges: Physical interactions

Protein-Protein Interactions

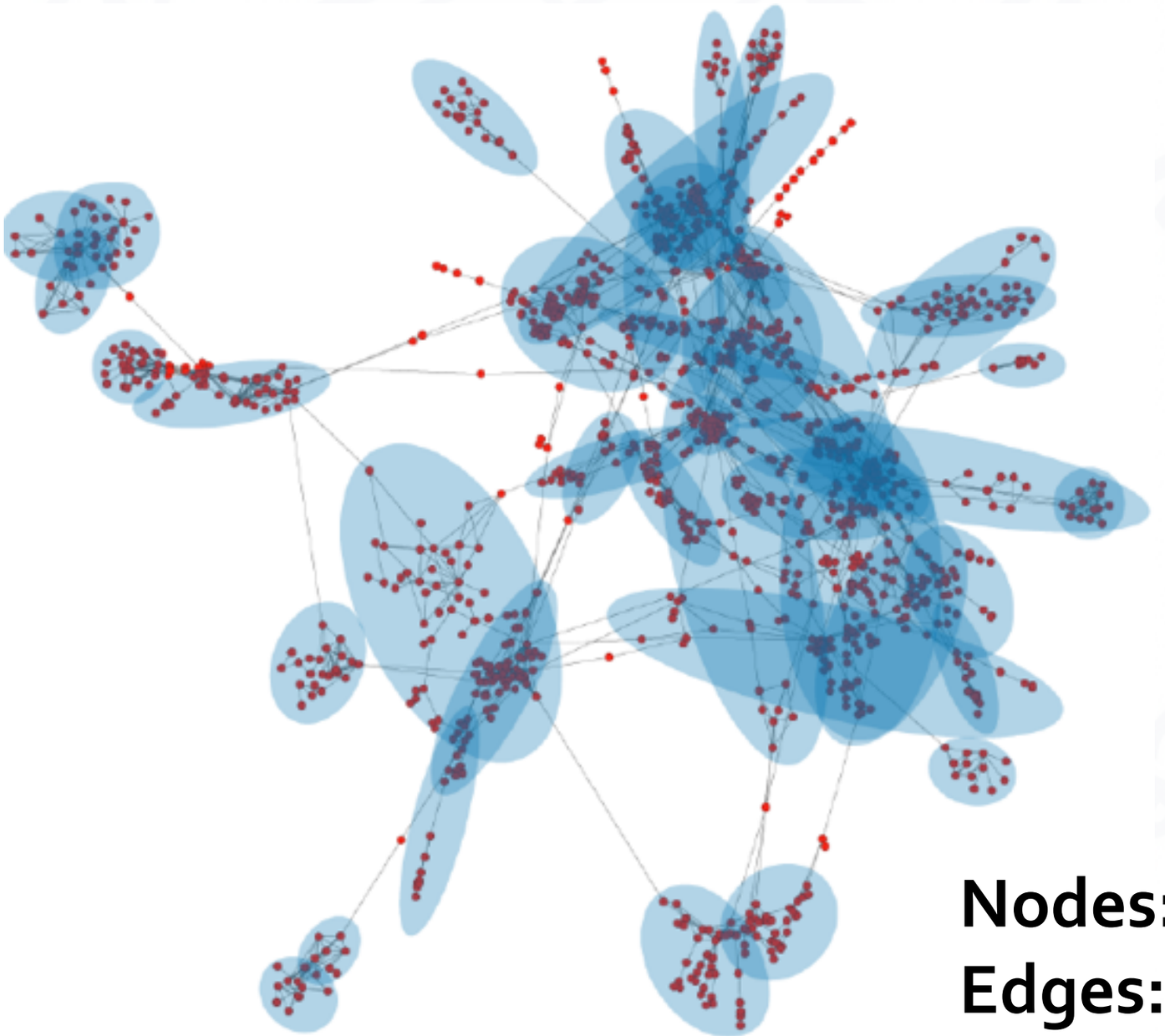


Can we identify functional modules?

Nodes: Proteins

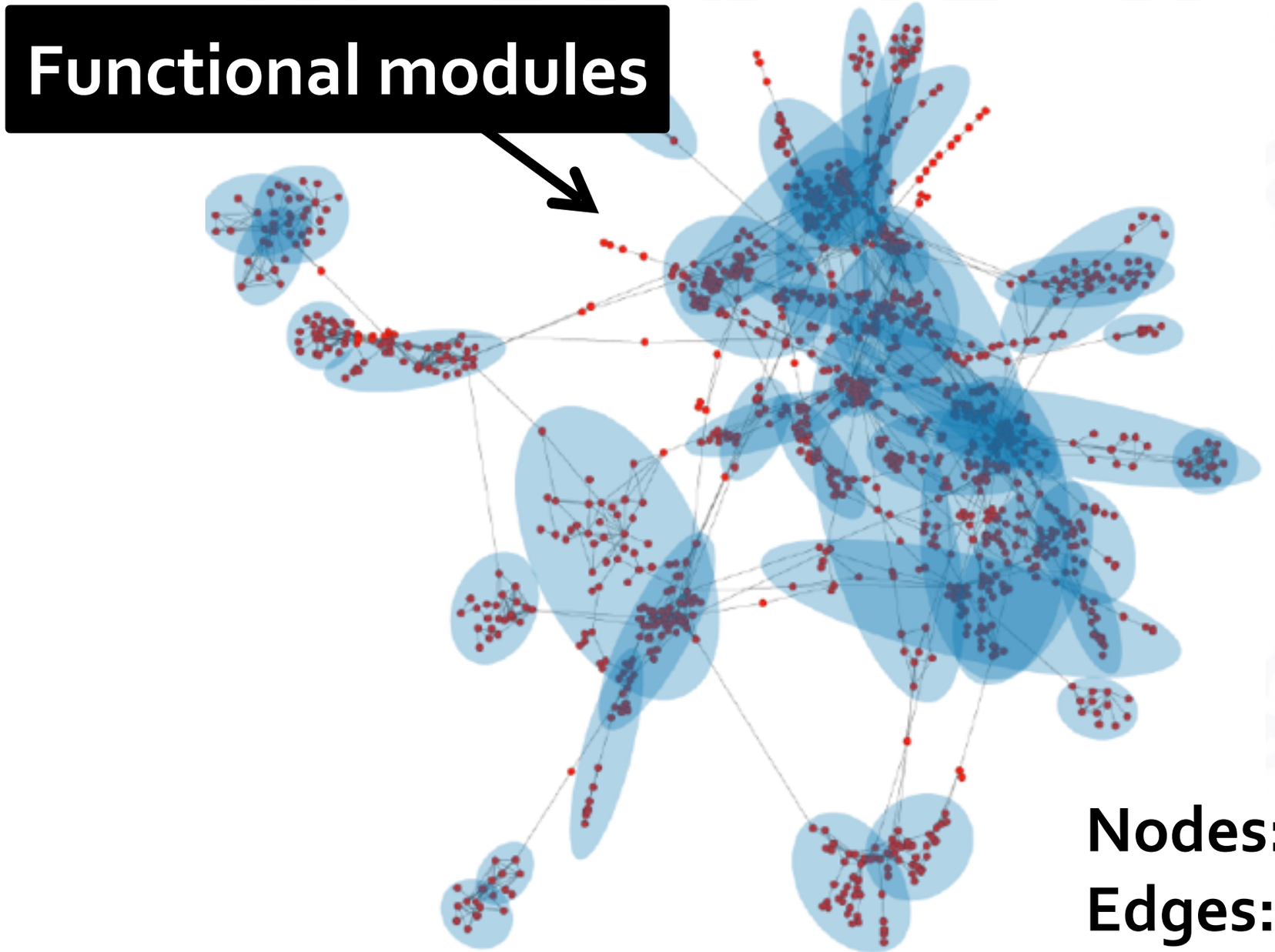
Edges: Physical interactions

Protein-Protein Interactions



Nodes: Proteins
Edges: Physical interactions

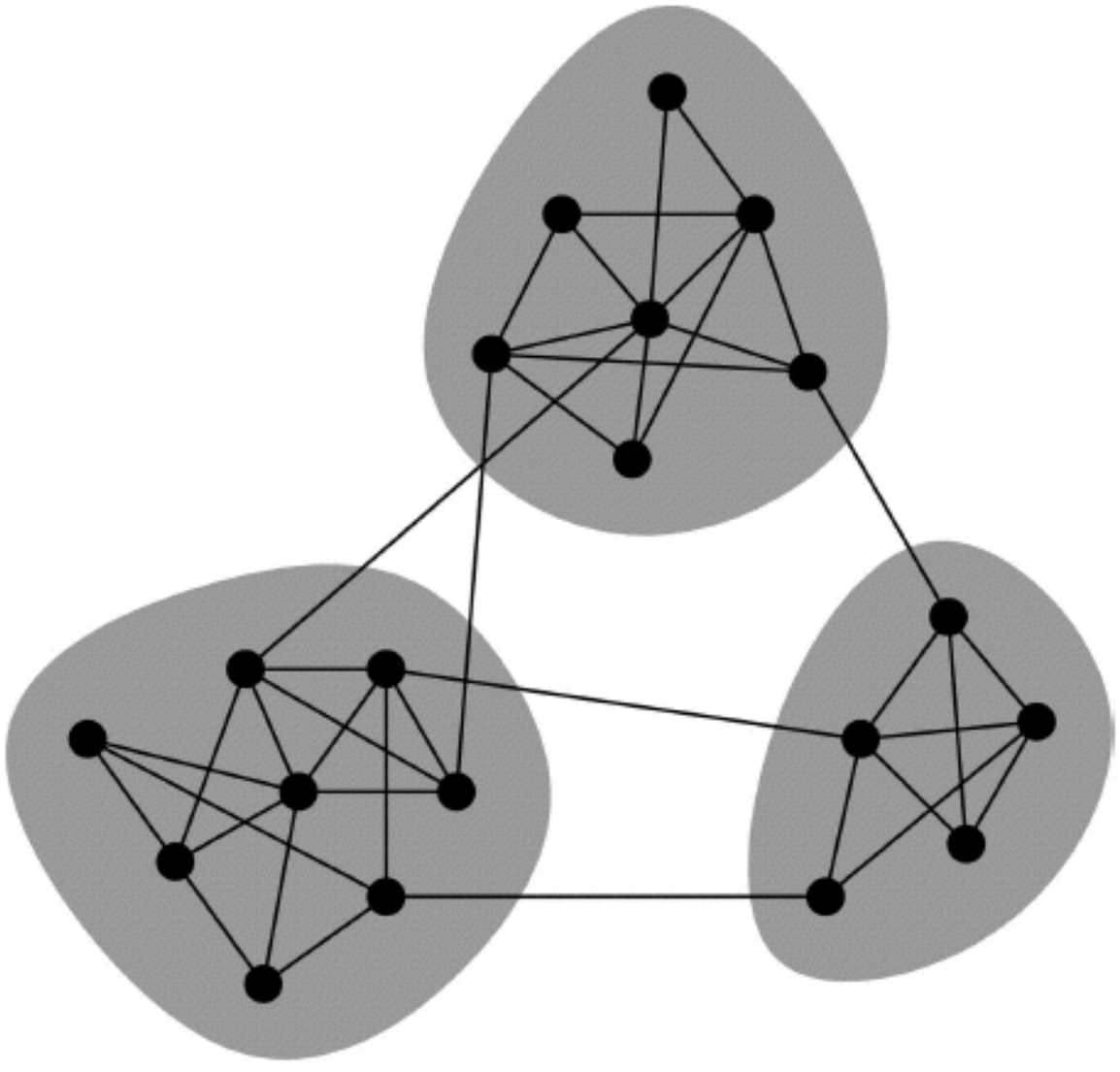
Protein-Protein Interactions

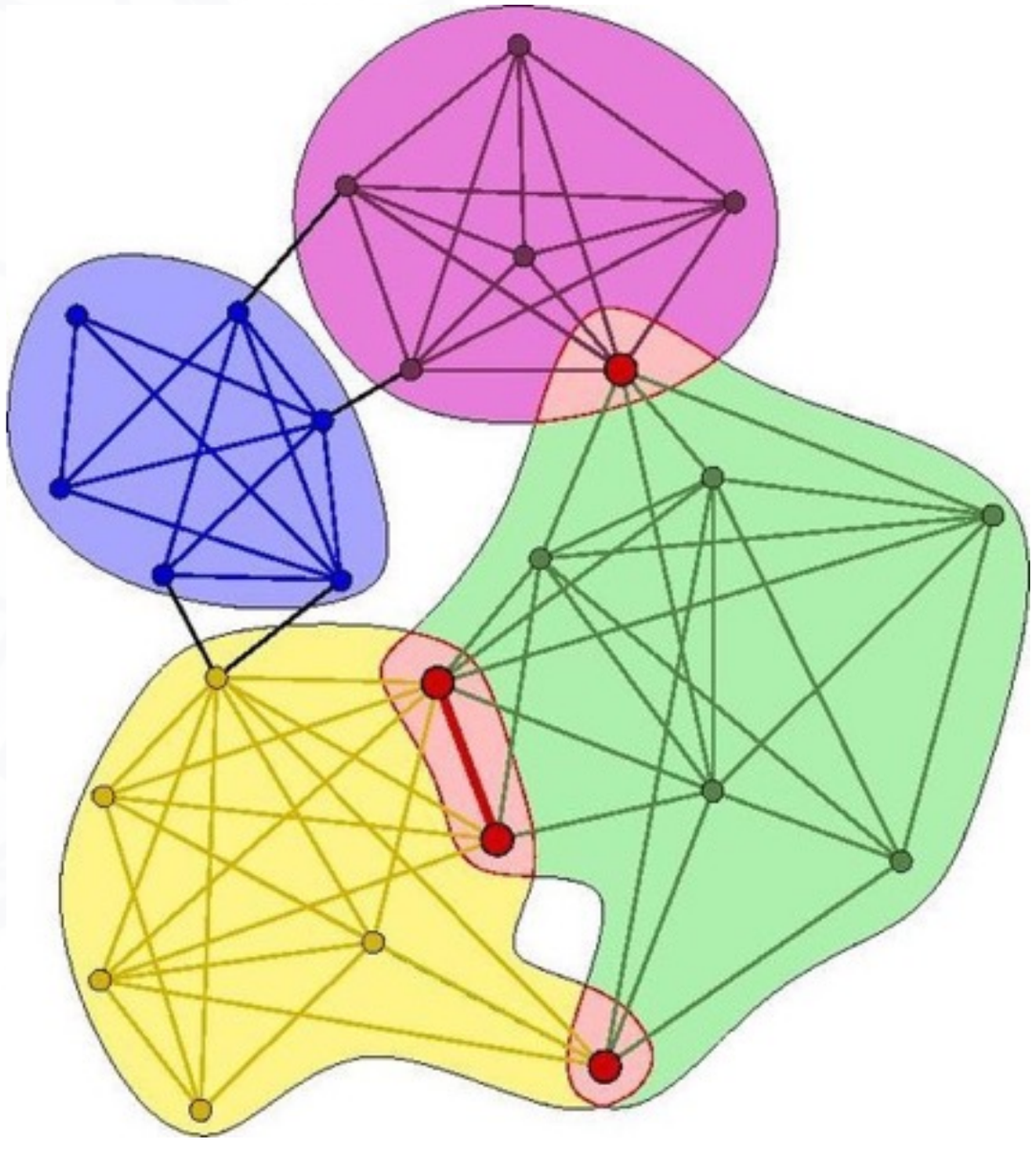


Nodes: Proteins
Edges: Physical interactions

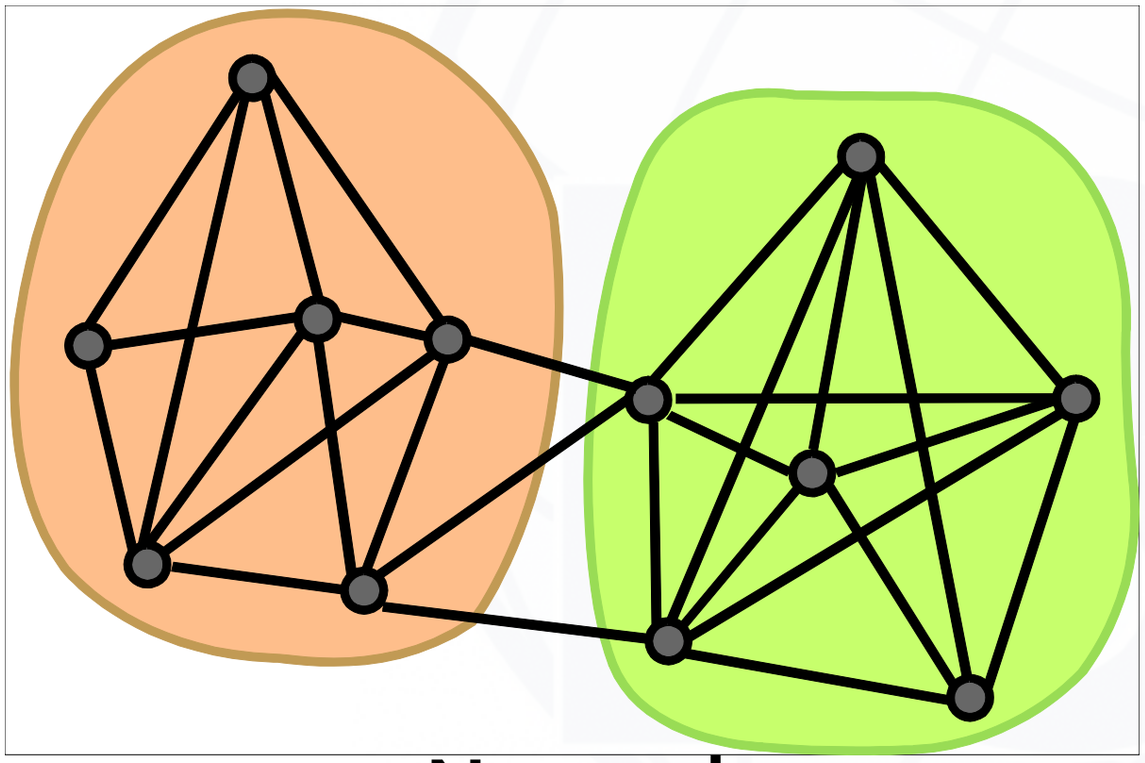
Overlapping Communities

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

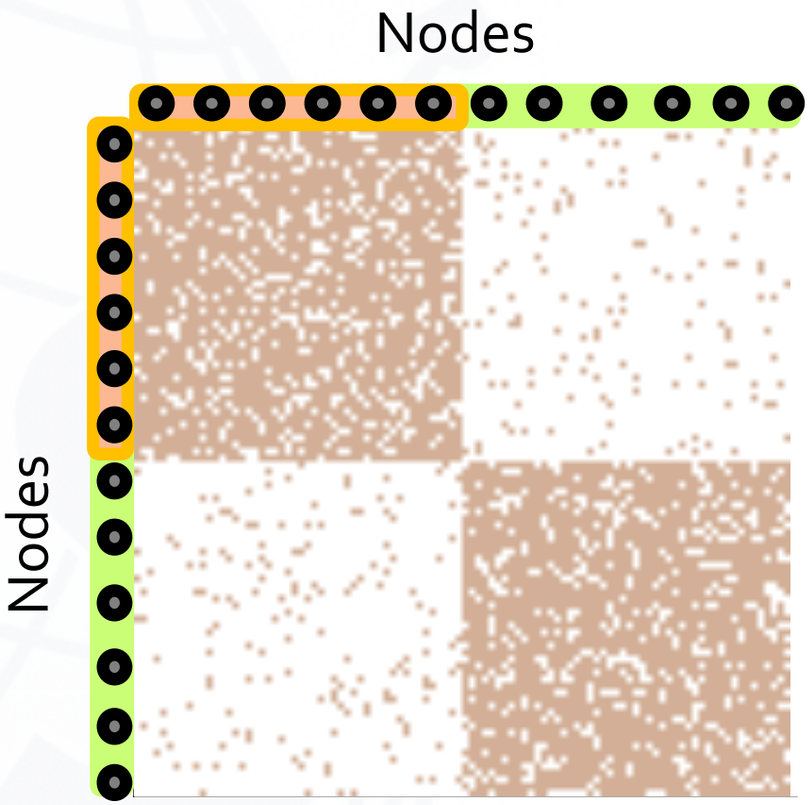




Non-overlapping Communities



Network



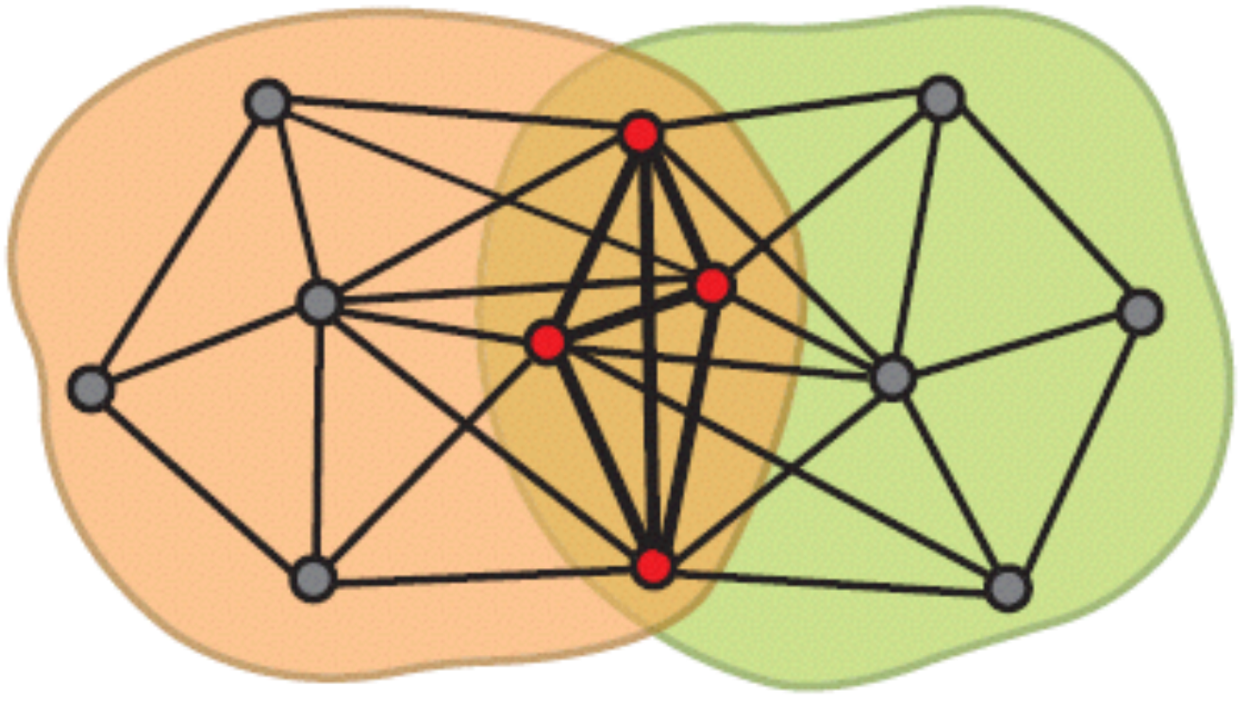
Adjacency matrix

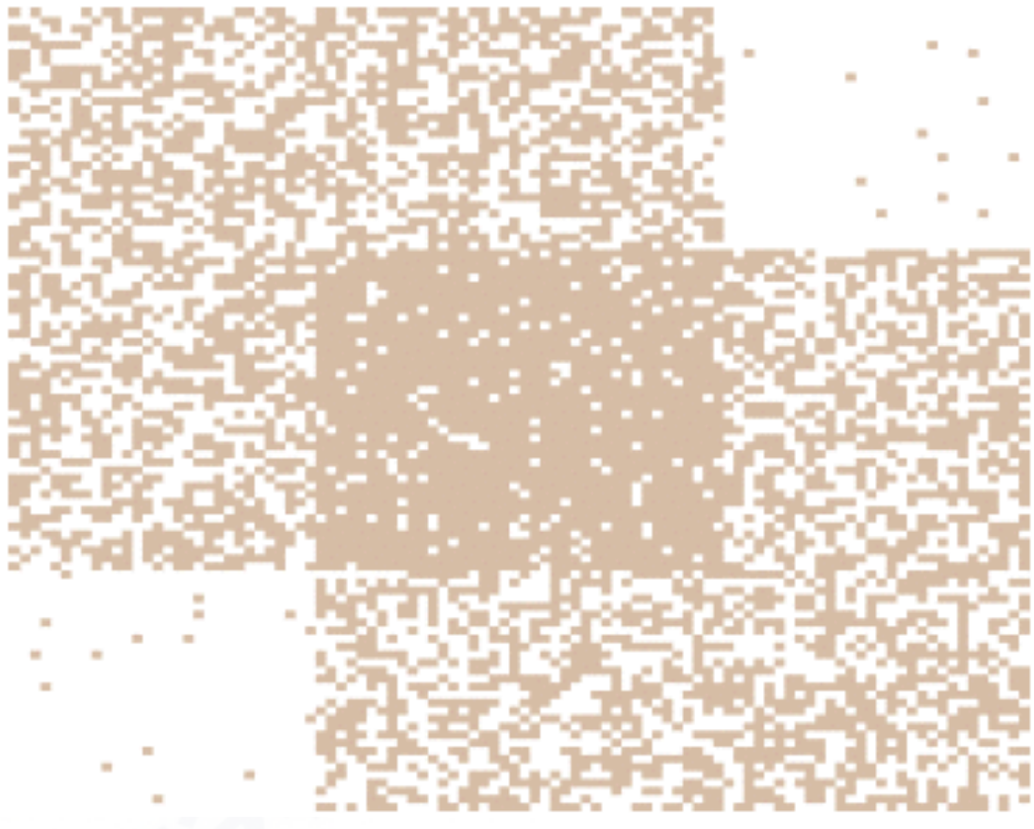
Communities as Tiles!

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

Communities as Tiles!

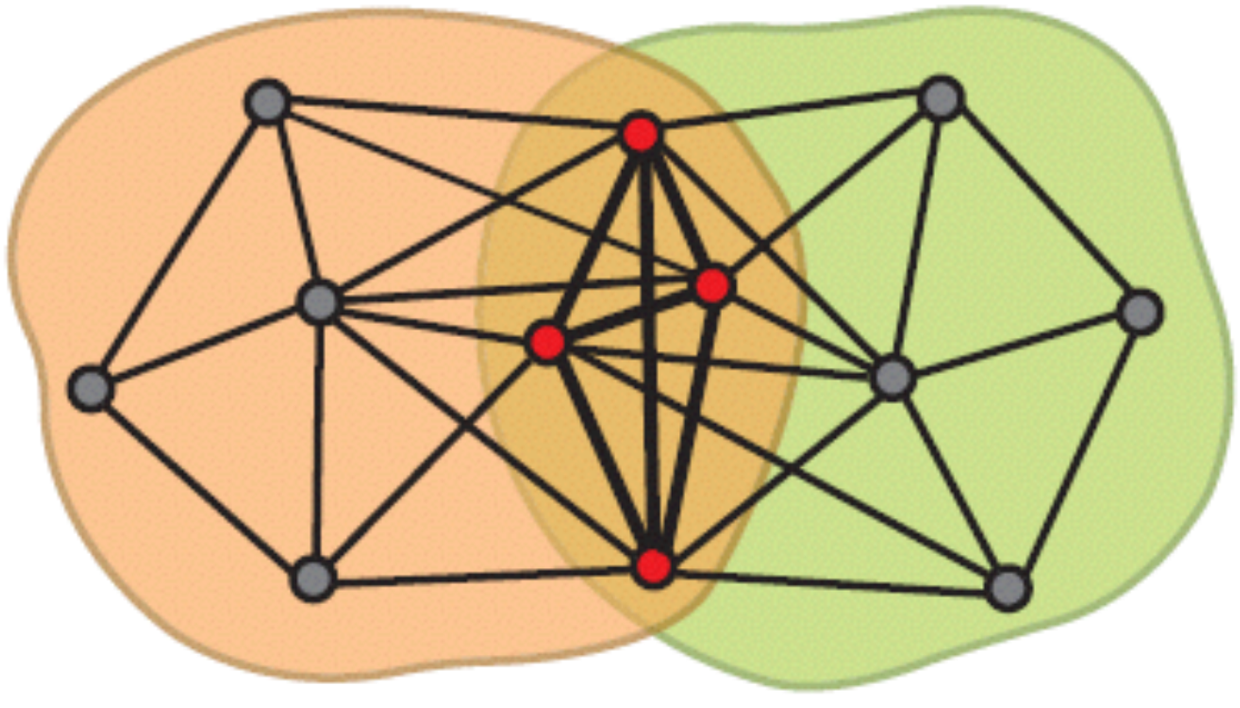
- **What is the structure of community overlaps:**
  Edge density in the overlaps is higher!

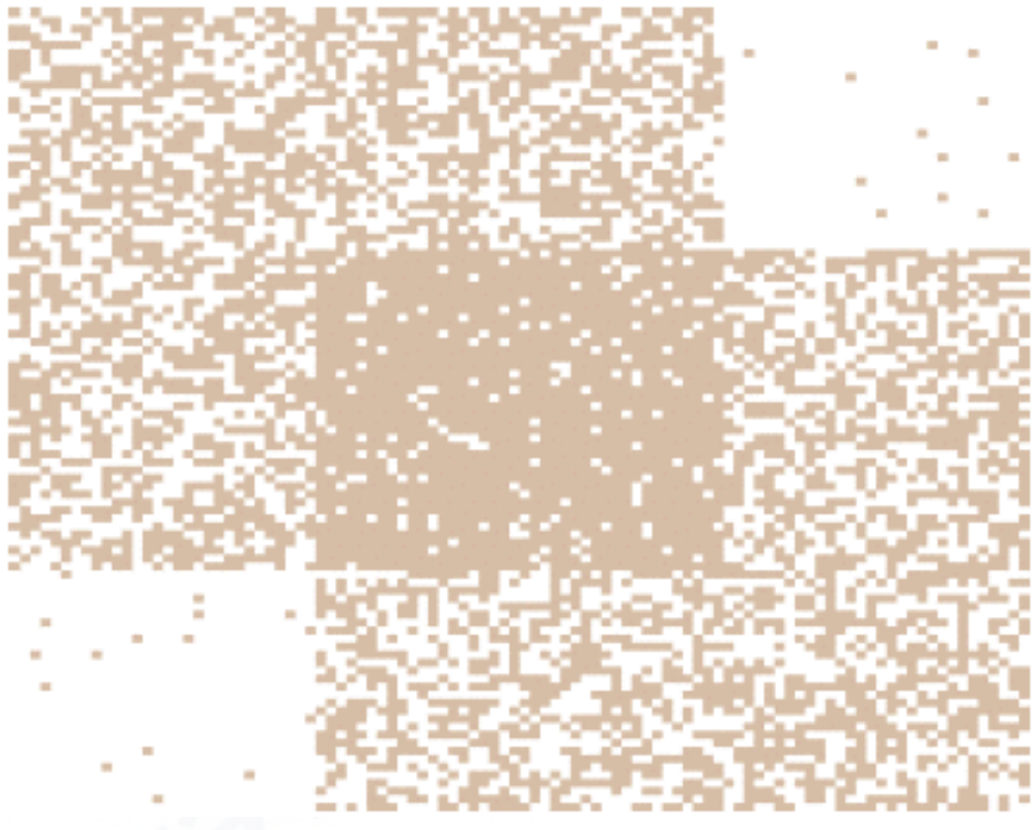




Communities as Tiles!

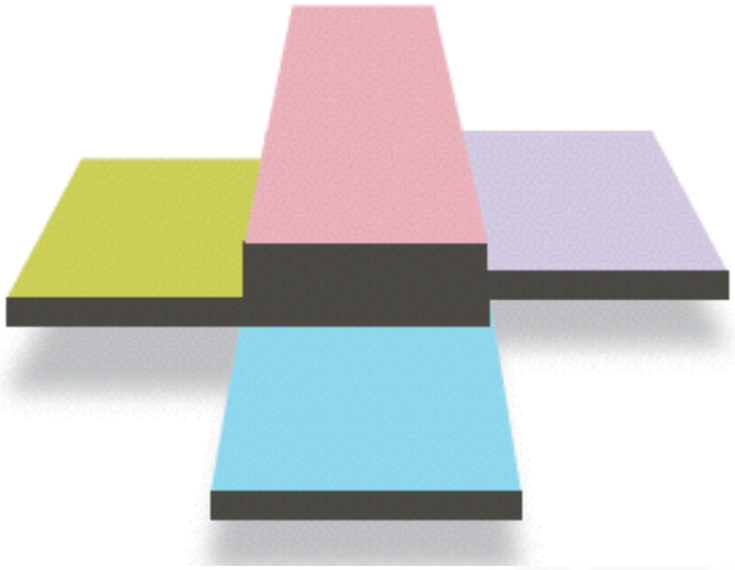
- **What is the structure of community overlaps:**
  Edge density in the overlaps is higher!





Communities as "tiles"

Recap so far…



Communities in a network

Recap so far…

Communities in a network

This is what we want!

Plan of attack

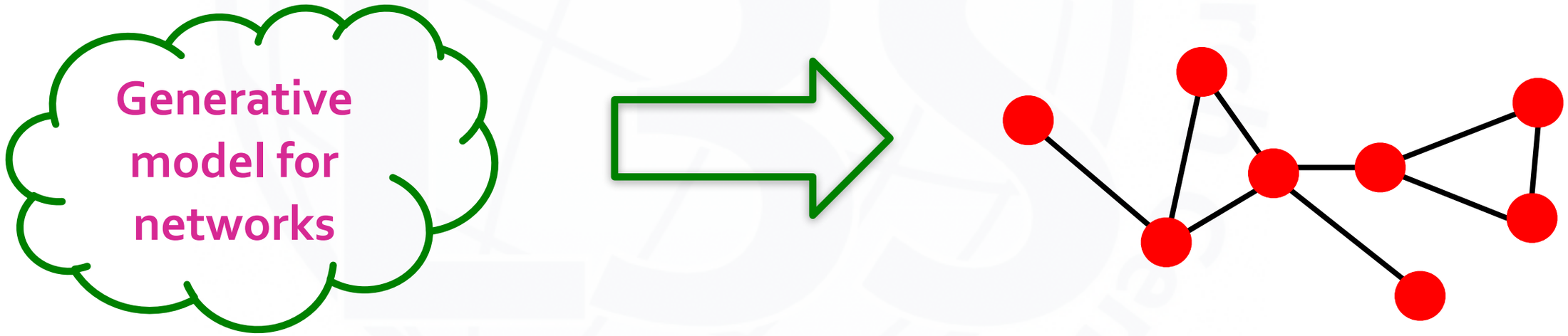
Plan of attack

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

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

Model of networks

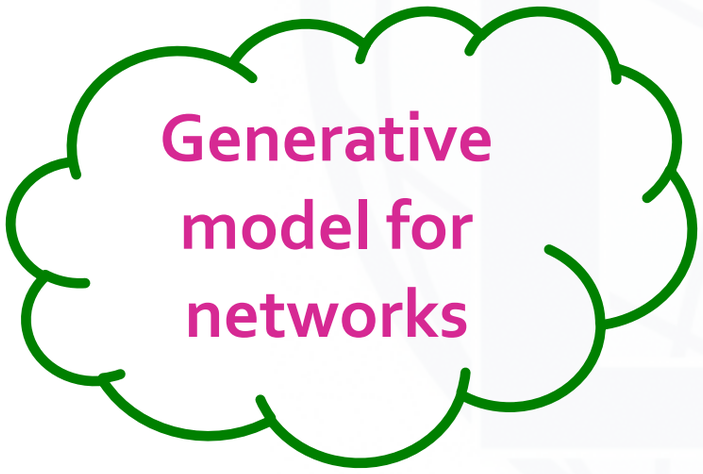
- **Goal: Define a model that can generate networks**
  - The model will have a set of "parameters" that we will later want to estimate (and detect communities)

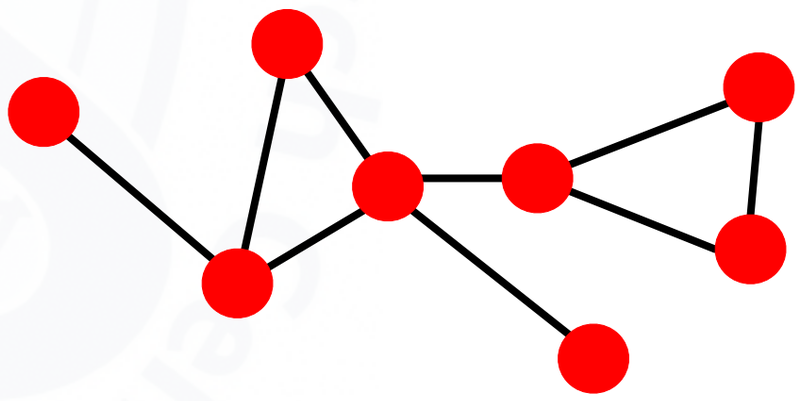


- **Q: Given a set of nodes, how do communities "generate" edges of the network?**

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)

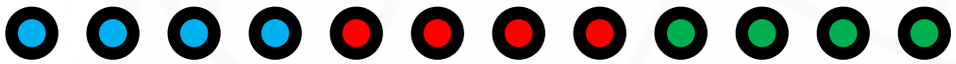




- **Q: Given a set of nodes, how do communities "generate" edges of the network?**

Community-Affiliation Graph

Nodes, V

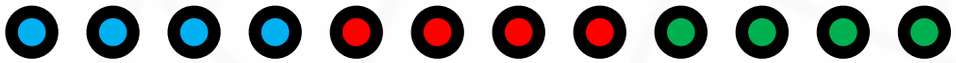


Model

Community-Affiliation Graph

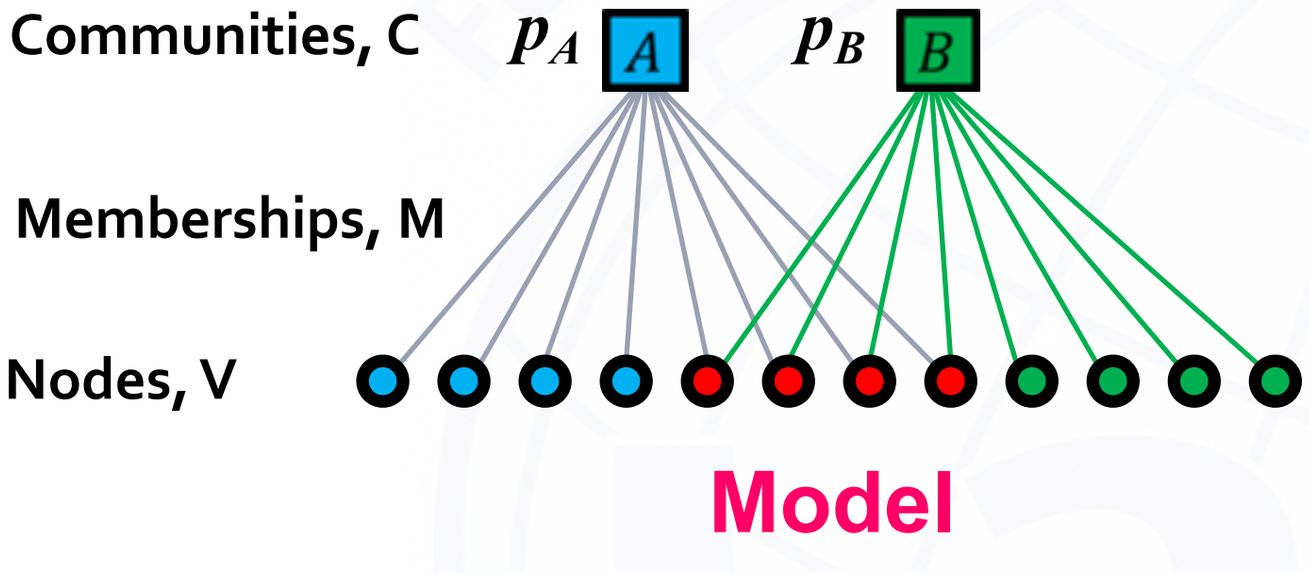
Communities, C p_A A p_B B

Nodes, V



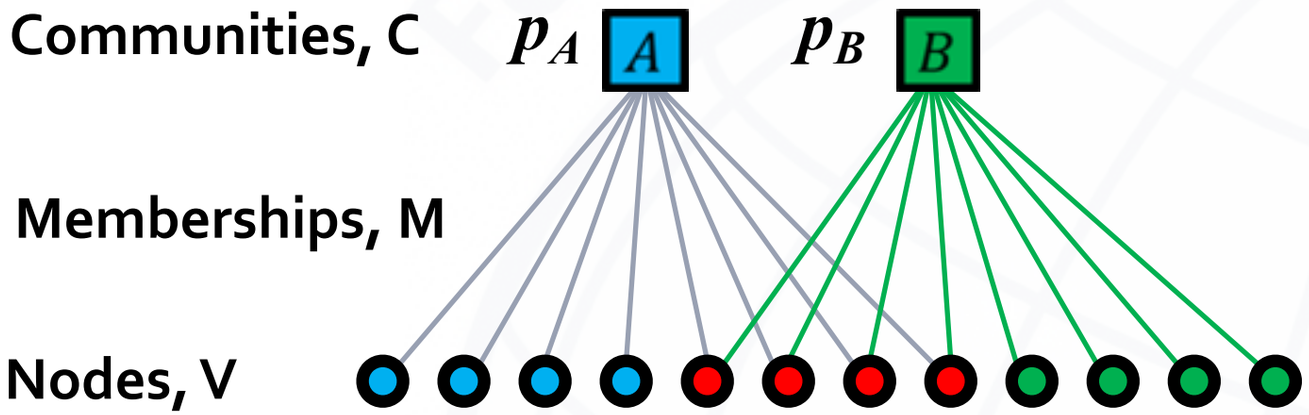
Model

Community-Affiliation Graph



Model

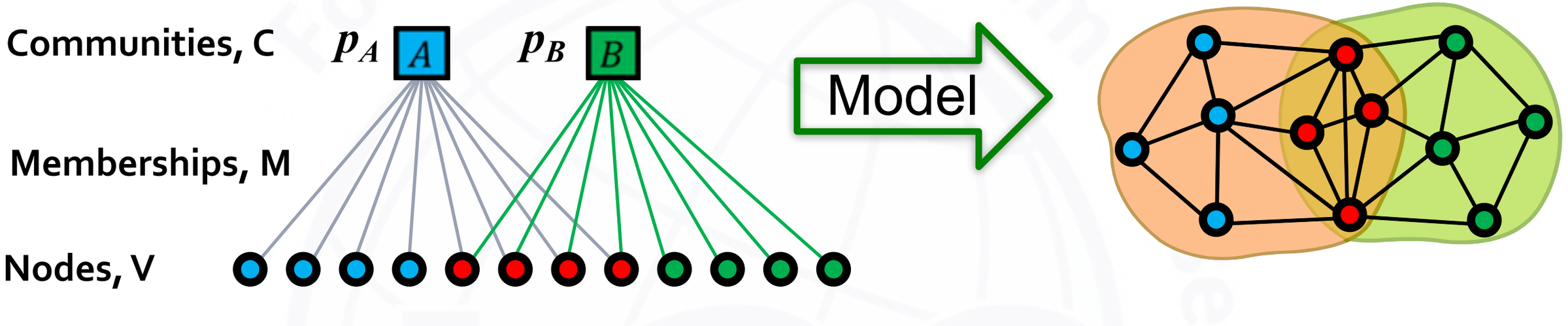
Community-Affiliation Graph



Model

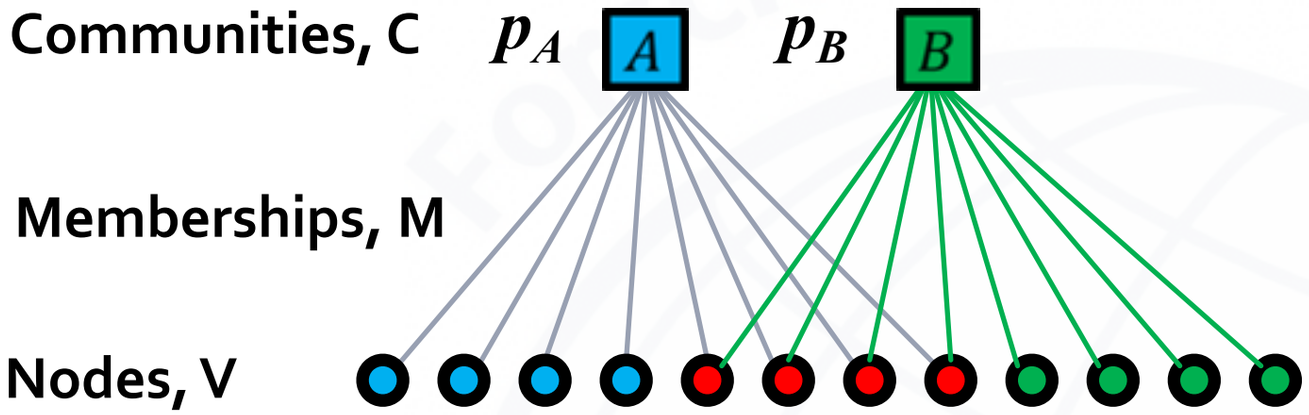
- **Generative model B($V, C, M, \{p_c\}$) for graphs:**
  - Nodes **V**, Communities **C**, Memberships **M**
  - Each community **c** has a single probability $\mathbf{p_c}$
  - Later we fit the model to networks to detect communities

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



Community Affiliations

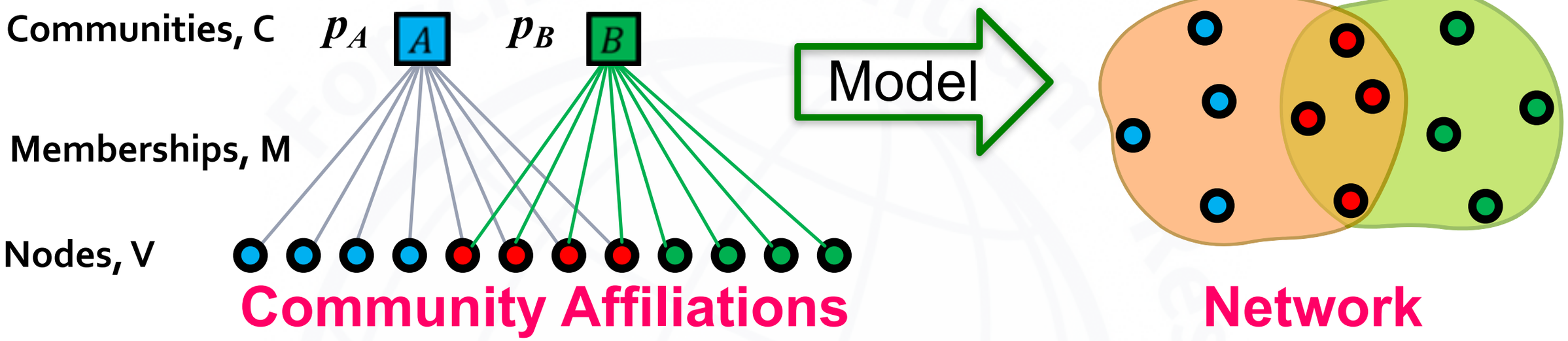
Network

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

M_u … set of communities node u belongs to

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

AGM: Generative Process



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

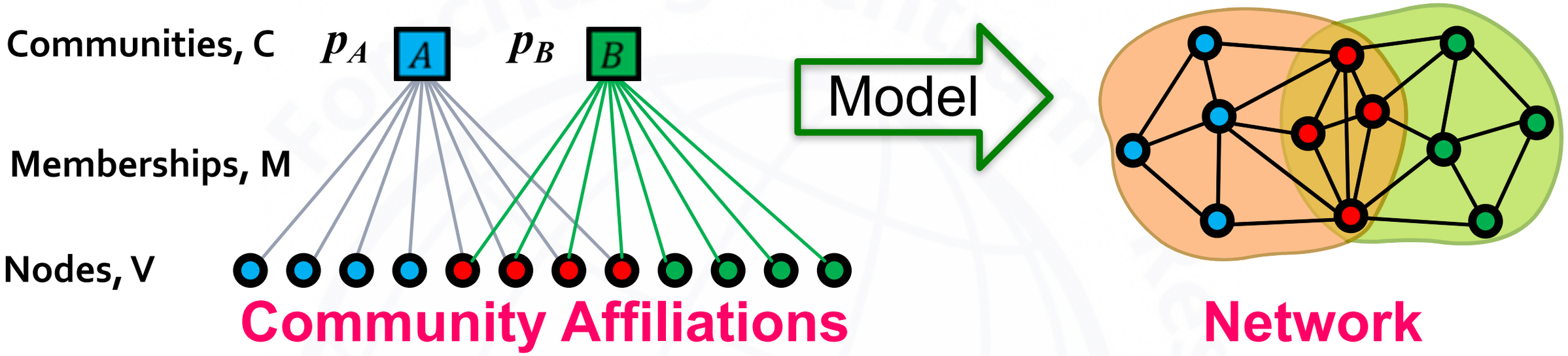
$$P(u,v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

M_u … set of communities node u belongs to

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

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

AGM: Generative Process



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

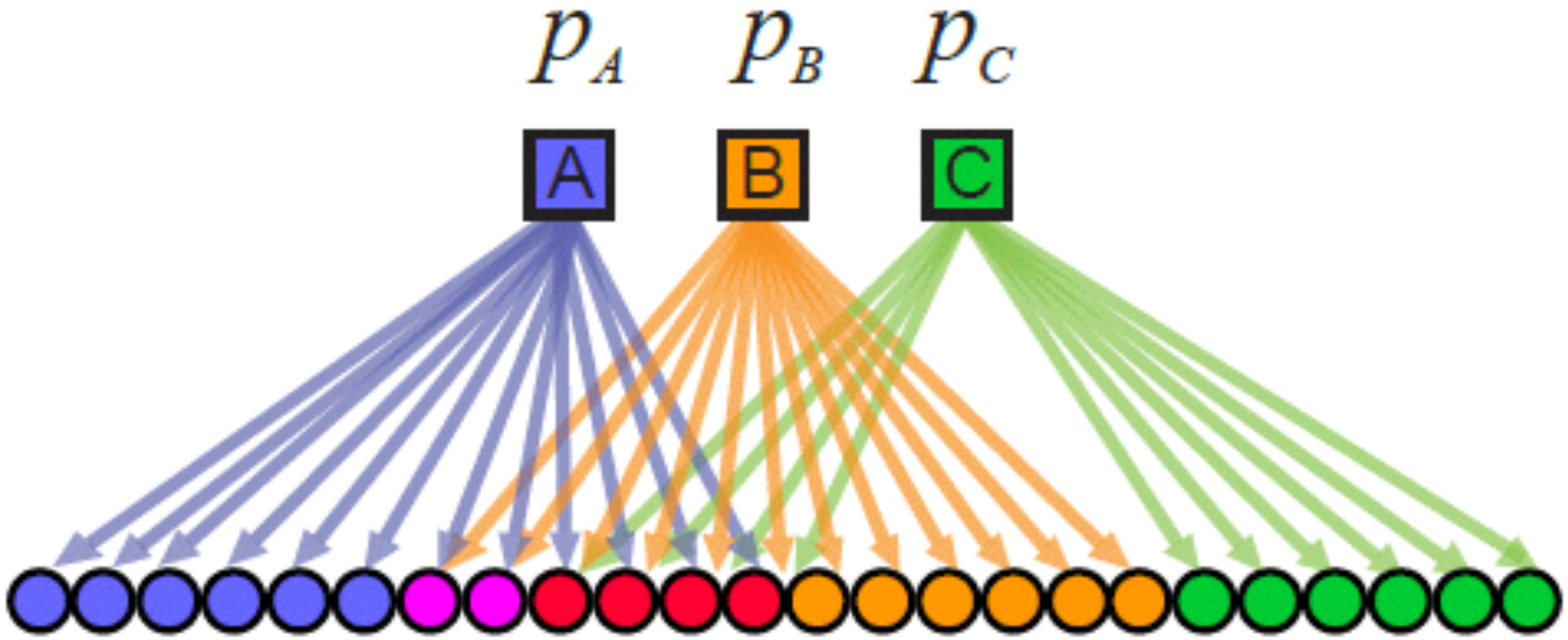
$$P(u,v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

M_u … set of communities node u belongs to

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

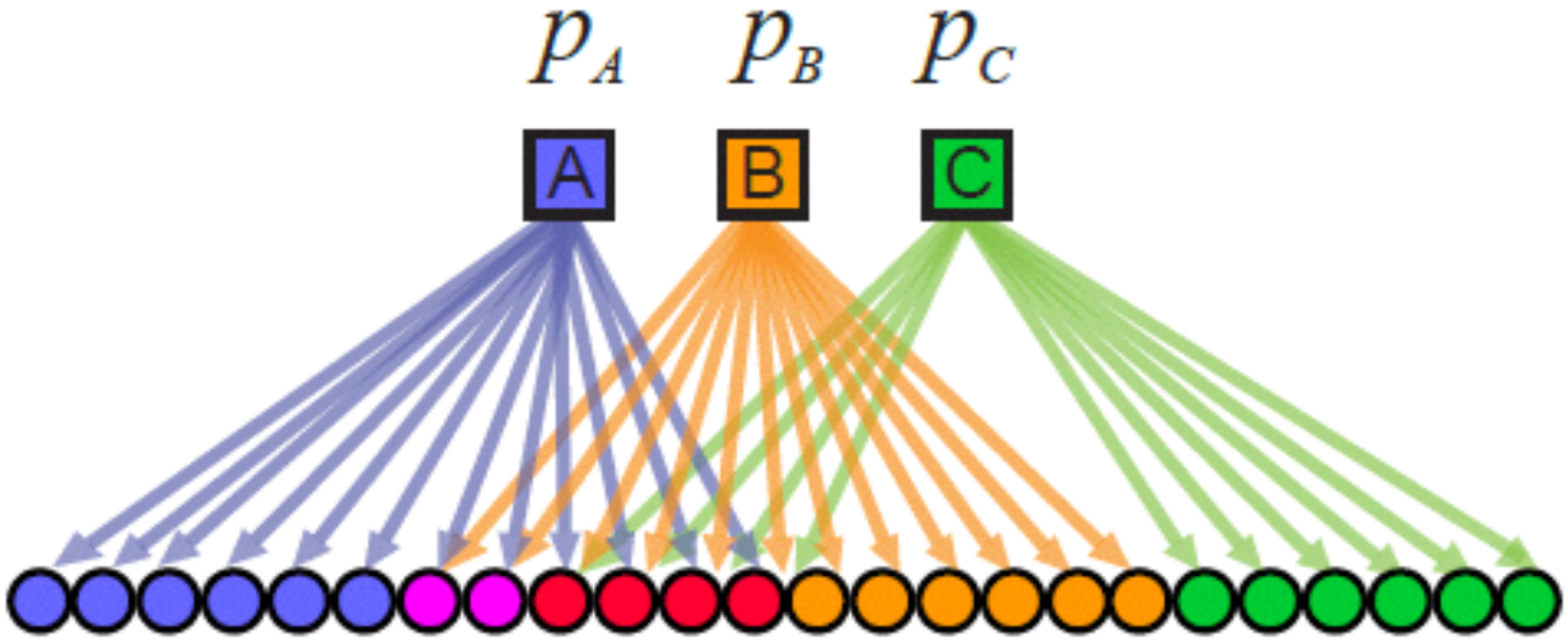
Think of this as an "OR" function: If at least 1 community says "YES" we create an edge

Recap: AGM networks



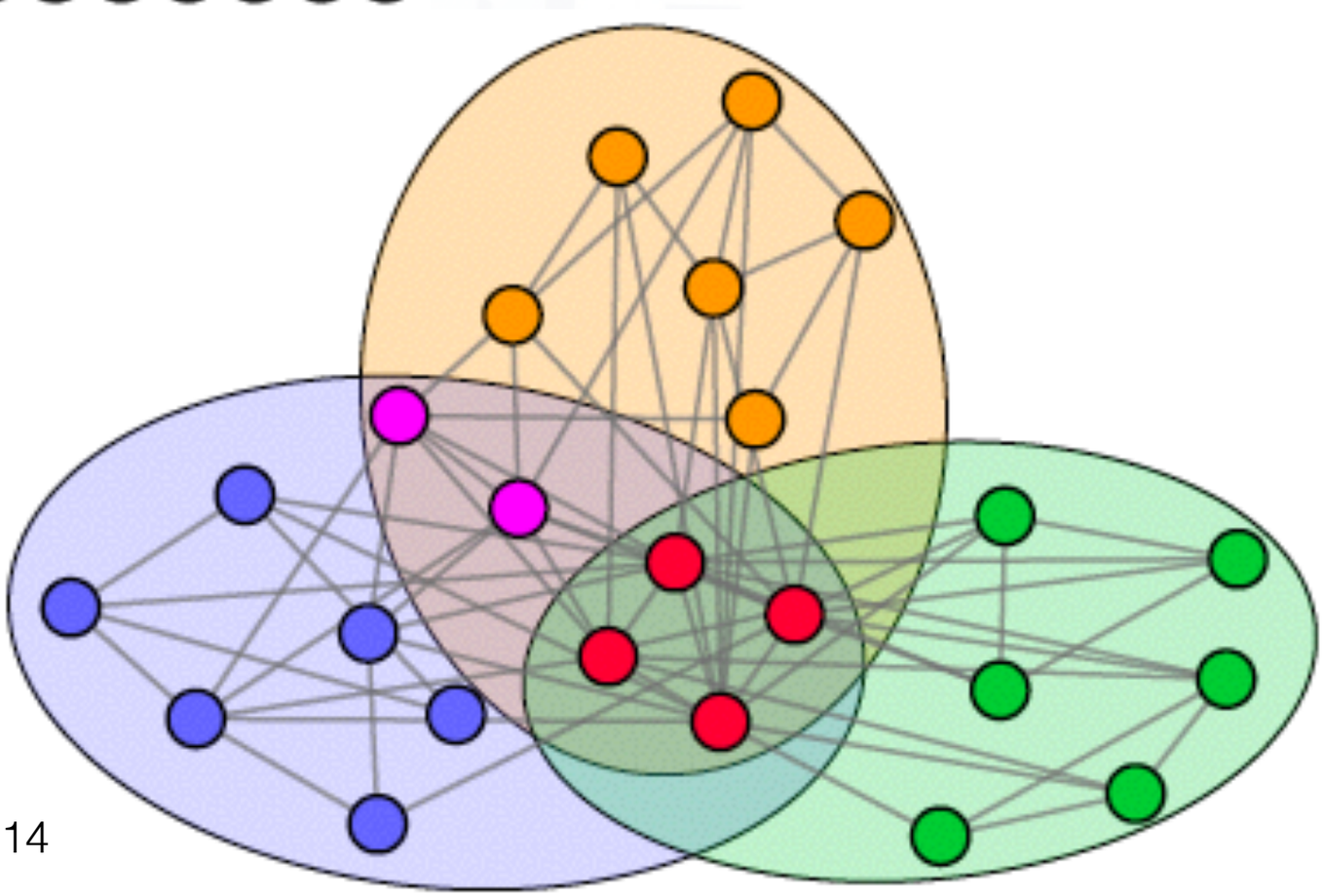
Model

Recap: AGM networks



Model

Network

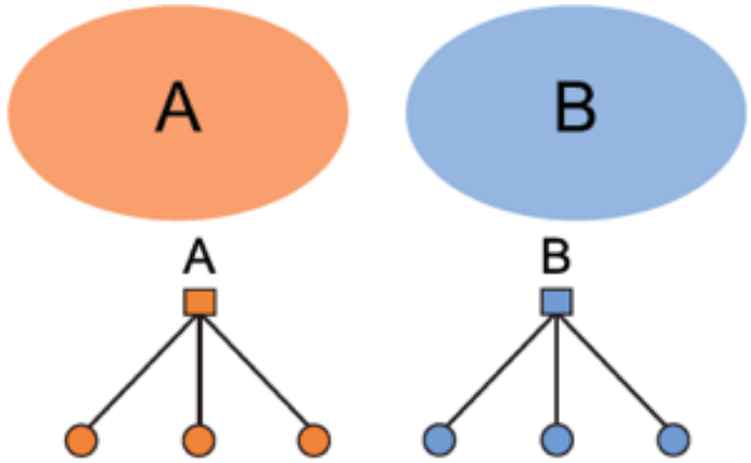


AGM: Flexibility

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

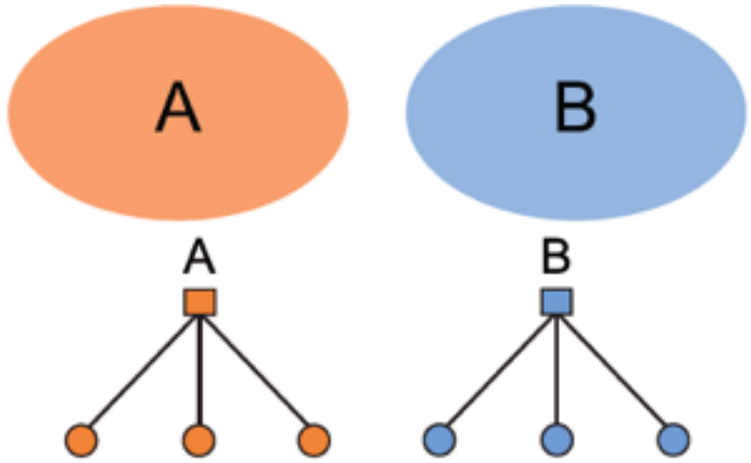
AGM: Flexibility

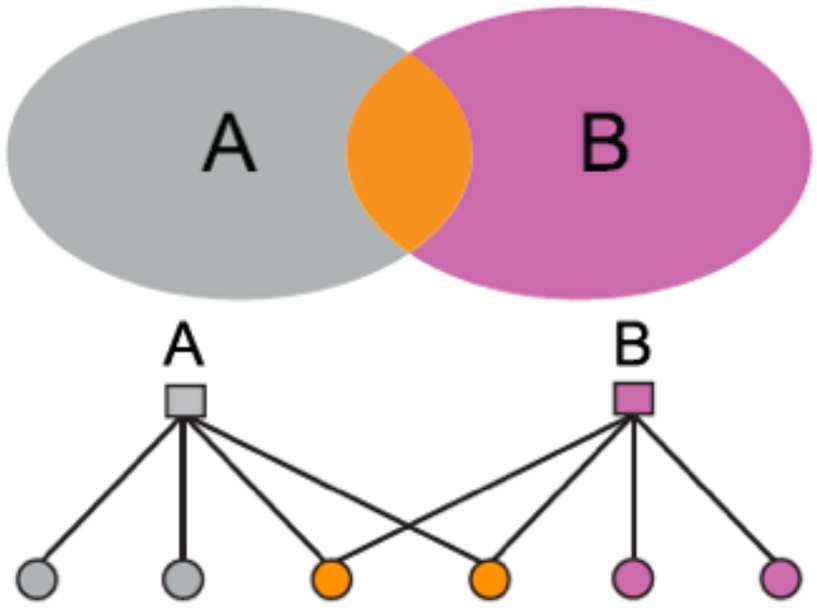
- **AGM can express a variety of community structures:**
  Non-overlapping, Overlapping, Nested



AGM: Flexibility

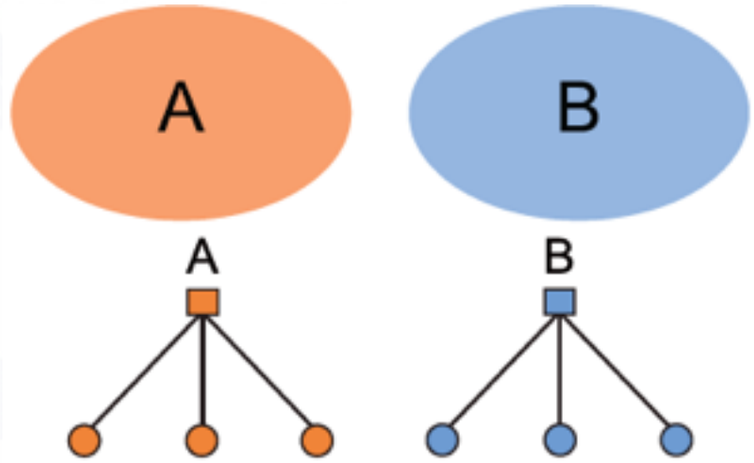
- **AGM can express a variety of community structures:**
  Non-overlapping, Overlapping, Nested

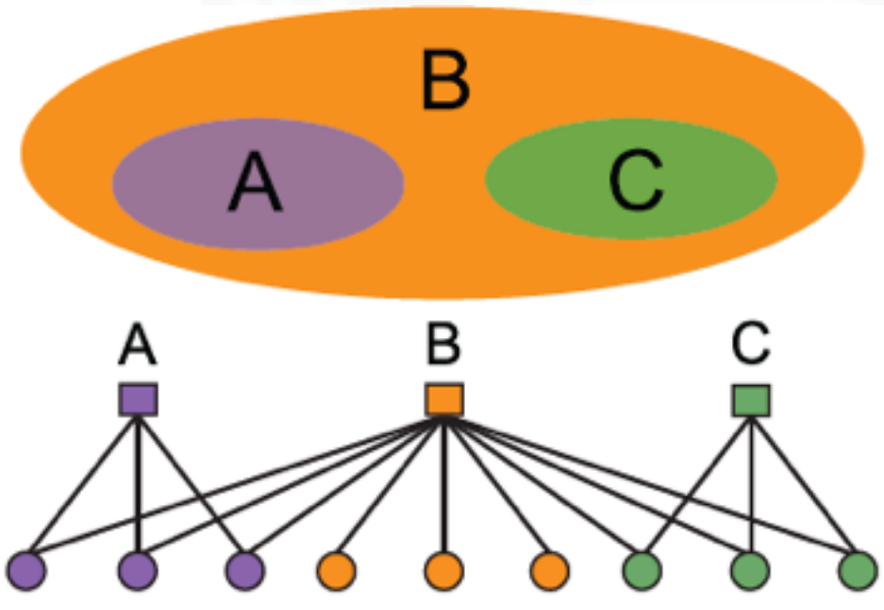


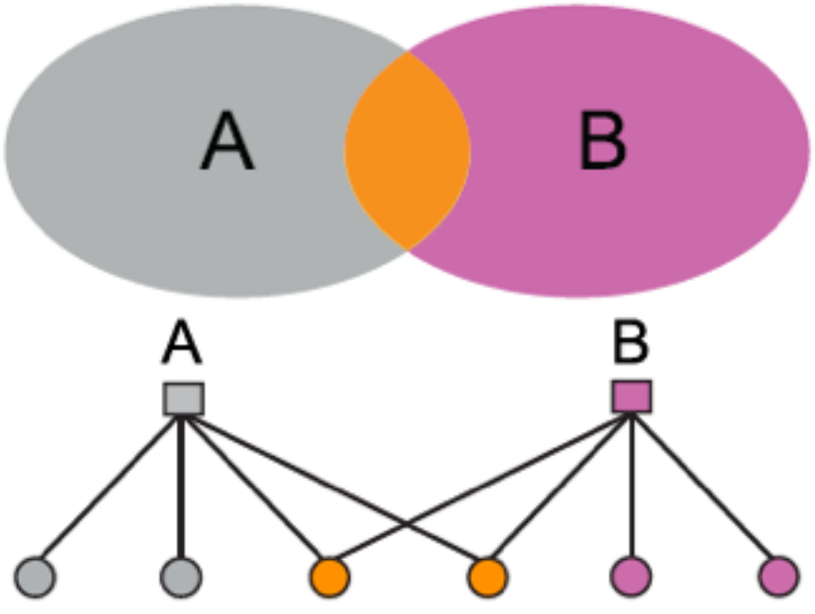


AGM: Flexibility

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



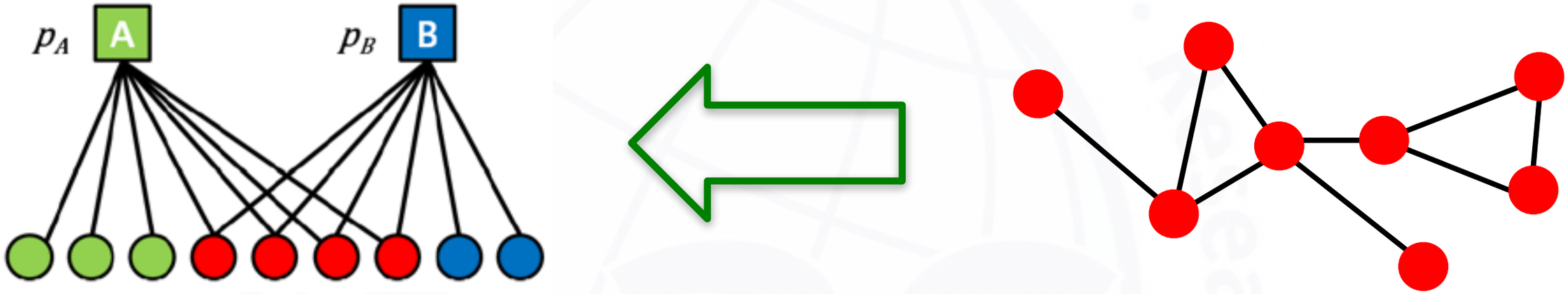




Detecting Communities

- **Detecting communities with AGM:**

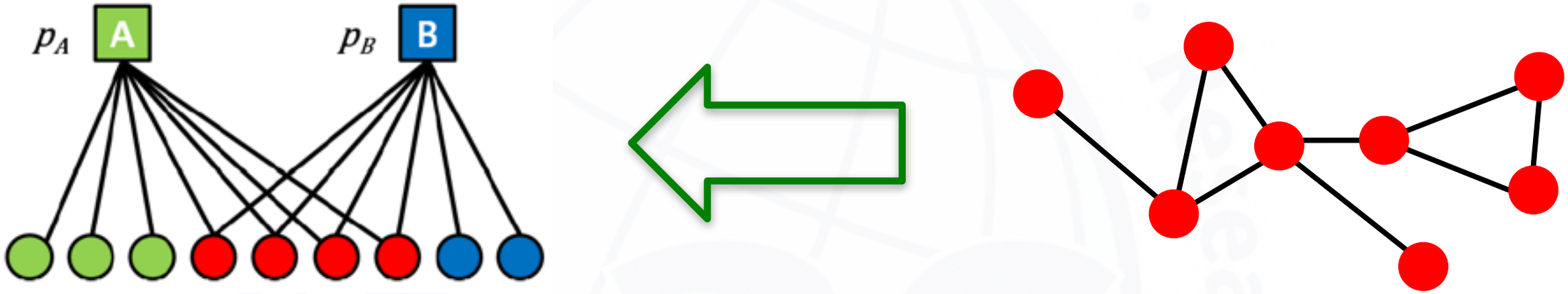
Detecting Communities



- **Detecting communities with AGM:**

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

Detecting Communities



- **Detecting communities with AGM:**

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

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

Likelihood of a Graph

- Given a set of parameters, the likelihood of the graph is computed by

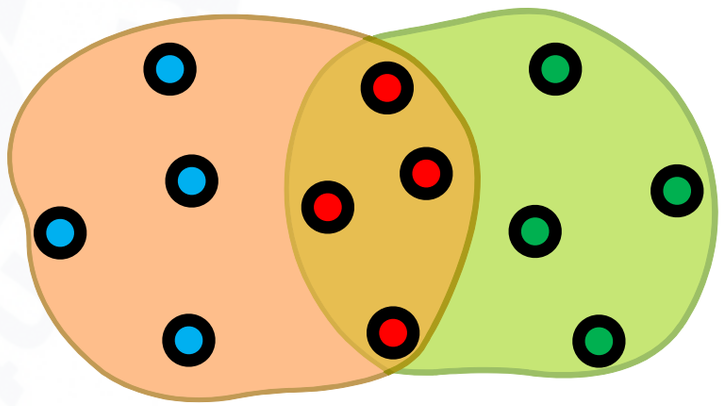
Estimate parameters $\hat{\theta}$ by using Maximum Likelihood estimation

Likelihood of a Graph

- Given a set of parameters, the likelihood of the graph is computed by

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

Nodes, V

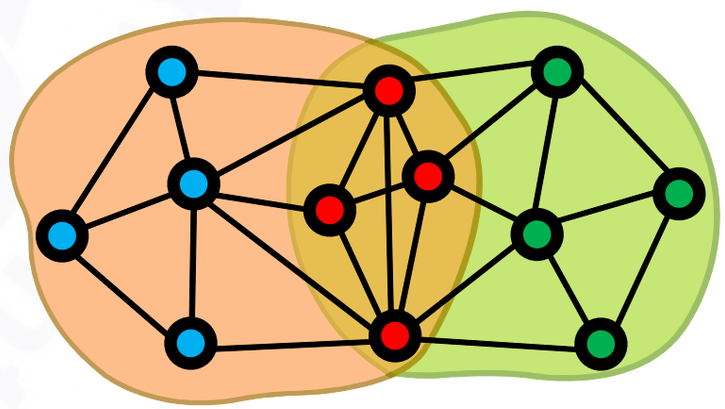


Estimate parameters $\hat{\theta}$ by using Maximum Likelihood estimation

Likelihood of a Graph

- Given a set of parameters, the likelihood of the graph is computed by

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



Estimate parameters $\hat{\theta}$ by using Maximum Likelihood estimation

Likelihood of a Graph

- Given a set of parameters, the likelihood of the graph is computed by

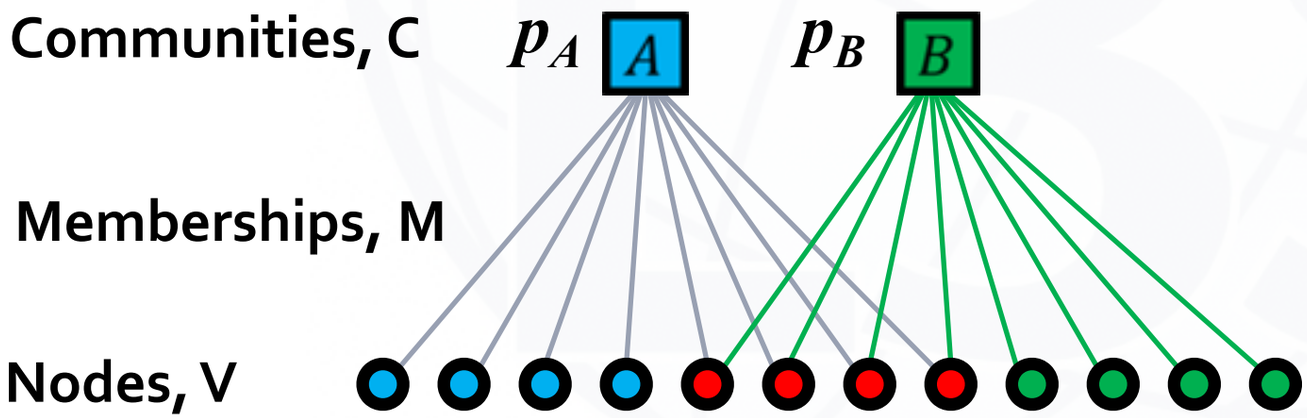
$$P(G \mid \Theta) = \prod_{(u,v)\in E} P(u,v) \prod_{(u,v)\notin E} (1 - P(u,v))$$

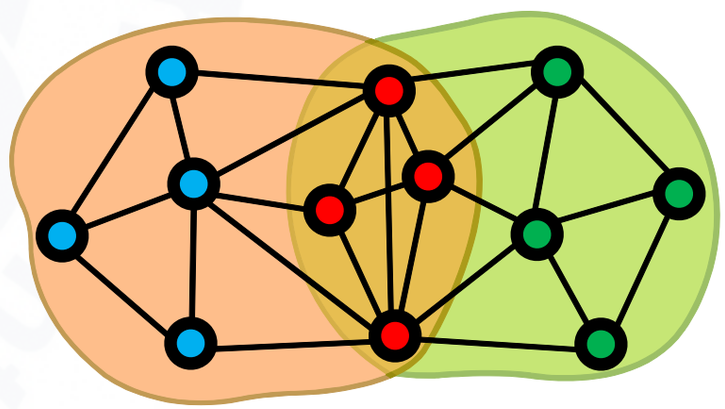
Estimate parameters $\hat{\theta}$ by using Maximum Likelihood estimation

Likelihood of a Graph

- Given a set of parameters, the likelihood of the graph is computed by

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





Estimate parameters $\hat{\theta}$ by using Maximum Likelihood estimation

Parameter Estimation : MLE

Given observed data D

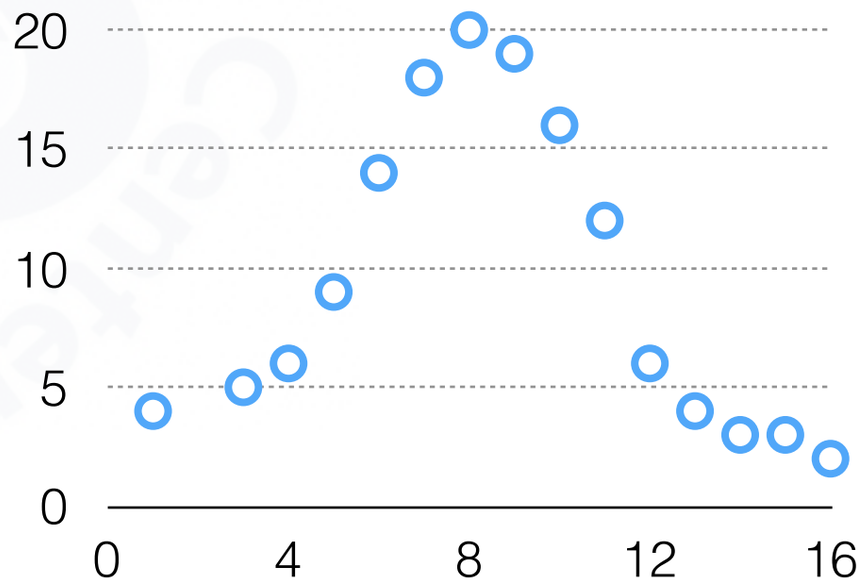
Assume that you have a probability density func. p(x) with parameters θ which generates it

parameter estimation problem

What are the value of the parameters which best explain my data ?

Say you feel that D is generated from a Normal distribution

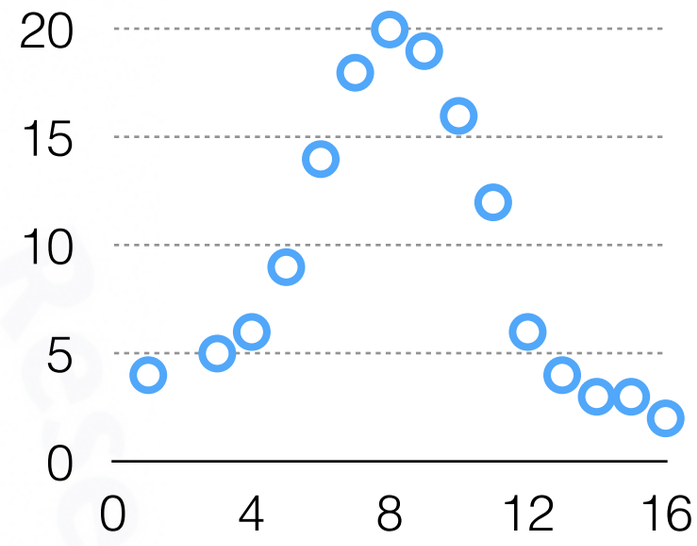
$$\theta = N(\mu, \sigma)$$



Maximum Likelihood Estimator (MLE)

Say you feel that D is generated from a Normal distribution

$$\theta = N(\mu, \sigma)$$



The **likelihood** of observing this independent and identically distributed sample is

$$\mathcal{L}(\theta \mid x = (x_1, x_2, \ldots, x_n)) = p(x \mid \theta)$$

parameters to be estimated

observed data

MLE chooses $\hat{\theta}$ maximizes this likelihood of generating the observed data

Maximum Likelihood Estimator (MLE)

- Desired probability distribution is the one that makes the observed data "most likely," which means that one must seek the value of the parameter vector $\hat{\theta}$ that maximizes the likelihood function $\mathcal{L}(\hat{\theta}|x)$

- For computational convenience, the MLE estimate is obtained by maximizing the log-likelihood function $ln(\,\mathcal{L}(\hat{\theta}|x)\,)$

$$ln(\,\mathcal{L}(\hat{\theta}|x)\,) = \sum_{x_i} ln\; p(x_i\,|\,\theta)$$

- If log-likelihood is differentiable find $\hat{\theta}$ partial derivatives

Maximum Likelihood Estimator (MLE)

- Desired probability distribution is the one that makes the observed data "most likely," which means that one must seek the value of the parameter vector $\hat{\theta}$ that maximizes the likelihood function $\mathcal{L}(\hat{\theta}|x)$

$$p(x = (x_1, x_2, \ldots, x_n) \mid \theta) = p(x_1|\theta) \, . \, p(x_2|\theta) \ldots p(x_n|\theta)$$

- For computational convenience, the MLE estimate is obtained by maximizing the log-likelihood function $ln(\, \mathcal{L}(\hat{\theta}|x)\,)$

$$ln(\, \mathcal{L}(\hat{\theta}|x)\,) = \sum_{x_i} ln \; p(x_i \,|\, \theta)$$

- If log-likelihood is differentiable find $\hat{\theta}$ partial derivatives

Maximum Likelihood Estimator (MLE)

$$\mathcal{L}(\theta \mid x = (x_1, x_2, \ldots, x_n)) = \prod_{x_i} p(x_i \mid \theta) \longrightarrow ln(\, \mathcal{L}(\hat{\theta}|x)\,) = \sum_{x_i} ln\; p(x_i \,|\, \theta)$$

likelihood

log likelihood

- Find maxima by using partial derivatives i.e. $\dfrac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \theta_i} = 0$

Maximum Likelihood Estimator (MLE)

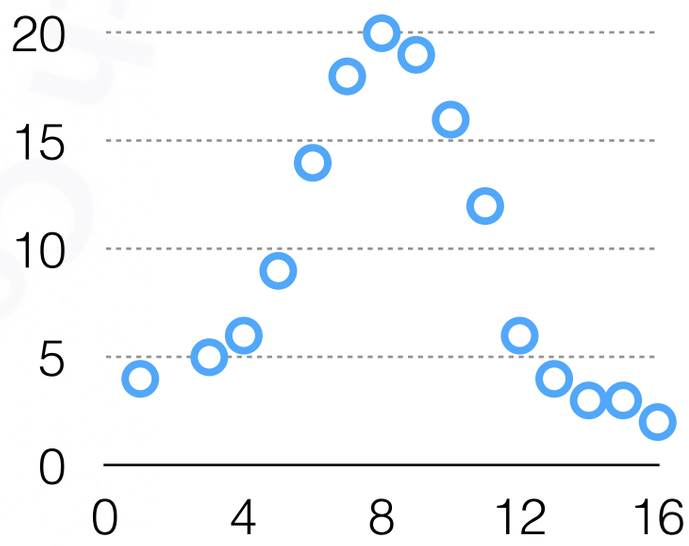
$$\mathcal{L}(\theta \mid x = (x_1, x_2, \ldots, x_n)) = \prod_{x_i} p(x_i \mid \theta) \longrightarrow ln(\,\mathcal{L}(\hat{\theta}|x)\,) = \sum_{x_i} ln\; p(x_i \,|\, \theta)$$

likelihood

log likelihood

- Find maxima by using partial derivatives i.e. $\dfrac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \theta_i} = 0$

- Example :



Maximum Likelihood Estimator (MLE)

$$\mathcal{L}(\theta \mid x = (x_1, x_2, \ldots, x_n)) = \prod_{x_i} p(x_i \mid \theta) \longrightarrow ln(\,\mathcal{L}(\hat{\theta}|x)\,) = \sum_{x_i} ln\; p(x_i \,|\, \theta)$$

likelihood

log likelihood

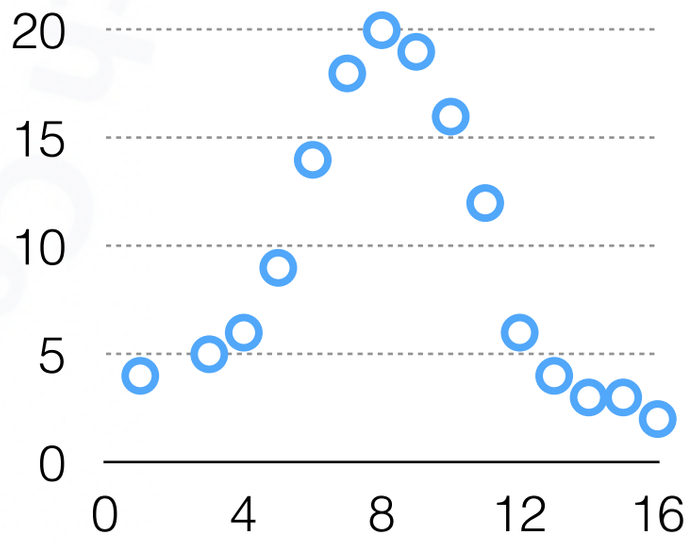
- Find maxima by using partial derivatives i.e. $\dfrac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \theta_i} = 0$

- Example :

$$p(x_i \mid (\mu, \sigma)) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

gaussian

generative model



Maximum Likelihood Estimator (MLE)

$$\mathcal{L}(\theta \mid x = (x_1, x_2, \ldots, x_n)) = \prod_{x_i} p(x_i \mid \theta) \longrightarrow ln(\, \mathcal{L}(\hat{\theta}|x)\,) = \sum_{x_i} ln\; p(x_i \,|\, \theta)$$

likelihood

log likelihood

- Find maxima by using partial derivatives i.e. $\dfrac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \theta_i} = 0$

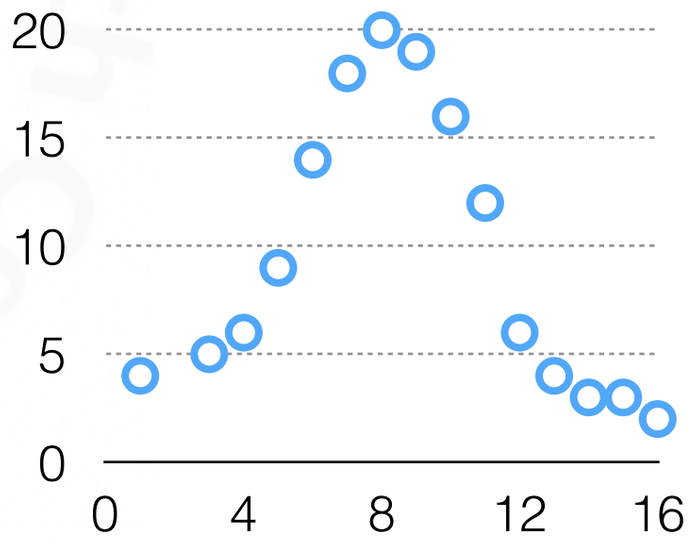
- Example :

$$p(x_i \mid (\mu, \sigma)) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

gaussian

generative model

$$\frac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \mu} = 0$$



Maximum Likelihood Estimator (MLE)

$$\mathcal{L}(\theta \mid x = (x_1, x_2, \ldots, x_n)) = \prod_{x_i} p(x_i \mid \theta) \longrightarrow ln(\,\mathcal{L}(\hat{\theta}|x)\,) = \sum_{x_i} ln\; p(x_i \,|\, \theta)$$

likelihood

log likelihood

- Find maxima by using partial derivatives i.e. $\dfrac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \theta_i} = 0$

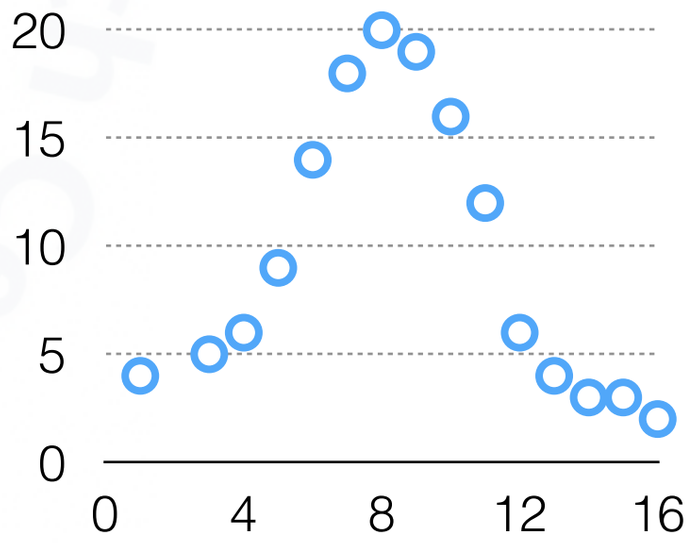
- Example :

$$p(x_i \mid (\mu, \sigma)) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

generative model

gaussian

$$\frac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \mu} = 0 \qquad \frac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \sigma} = 0$$



Maximum Likelihood Estimator (MLE)

$$\mathcal{L}(\theta \mid x = (x_1, x_2, \ldots, x_n)) = \prod_{x_i} p(x_i \mid \theta) \longrightarrow ln(\,\mathcal{L}(\hat{\theta}|x)\,) = \sum_{x_i} ln\; p(x_i \,|\, \theta)$$

likelihood

log likelihood

- Find maxima by using partial derivatives i.e. $\dfrac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \theta_i} = 0$

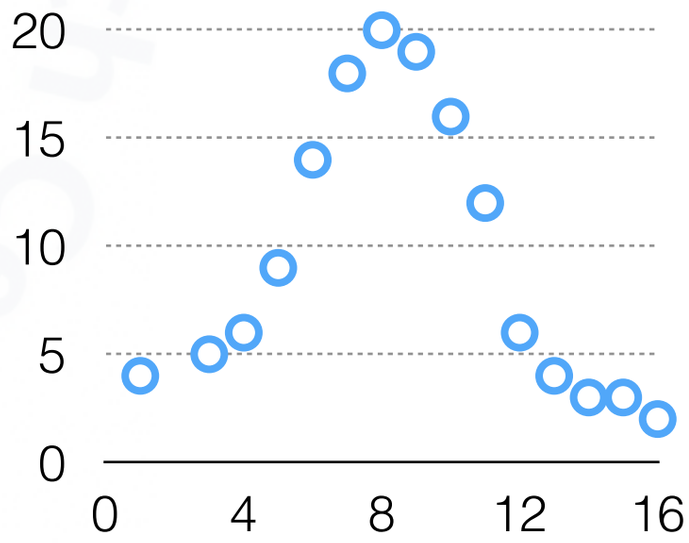
- Example :

$$p(x_i \mid (\mu, \sigma)) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

generative model

gaussian

$$\frac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \mu} = 0 \qquad \frac{\partial\, ln\, \mathcal{L}(\theta \mid x)}{\partial\, \sigma} = 0$$



- What are the estimated parameters of the gaussian ?

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 A**
  -

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

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 A**

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

A

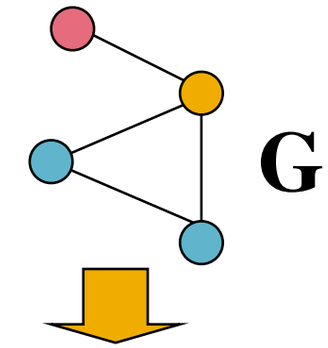
0	1	0	0
1	0	1	1
0	1	0	1
0	1	1	0

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

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

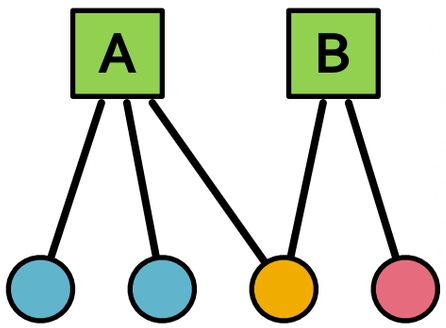
Graphs: Likelihood $P(G|\Theta)$

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



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

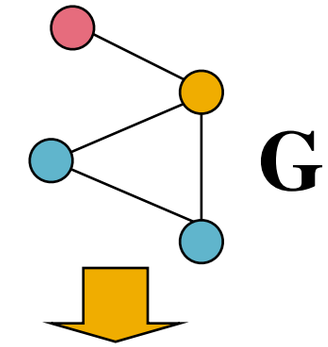
G

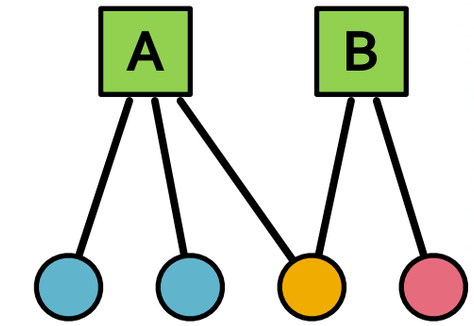


Θ=B(V, C, M, $\{p_c\}$)

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, $\{p_c\}$)

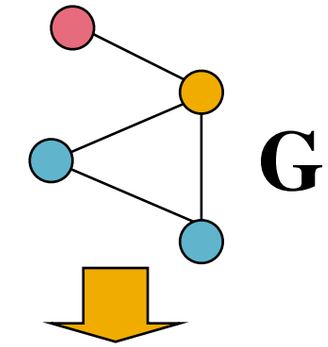
0	0.9	0.9	0
0.9	0	0.9	0
0.9	0.9	0	0.9
0	0	0.9	0

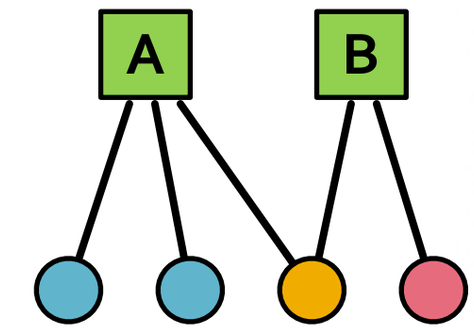
0	1	1	0
1	0	1	0
1	1	0	1
0	0	1	0

G

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

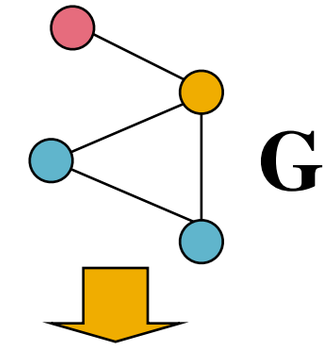
0	1	1	0
1	0	1	0
1	1	0	1
0	0	1	0

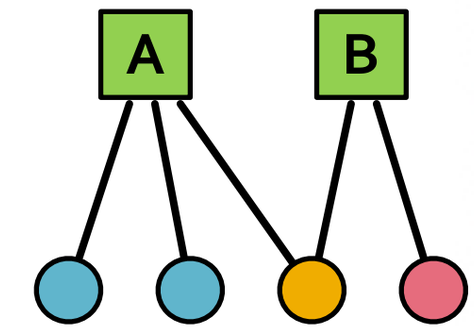
G

$P(G|\Theta)$

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

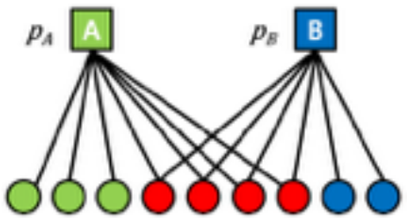
$P(G|\Theta)$

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

MLE for Graphs

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

$$\boldsymbol{G}$$

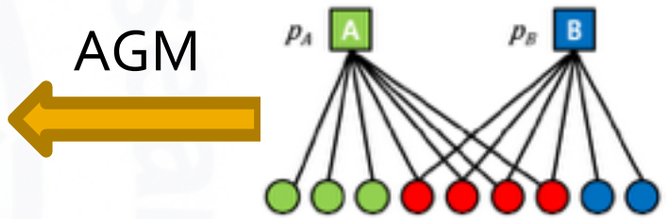


-
  - **How do we find $\boldsymbol{B(V, C, M, \{p_C\})}$ that maximizes the likelihood?**

MLE for Graphs

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

$$\boldsymbol{G}$$

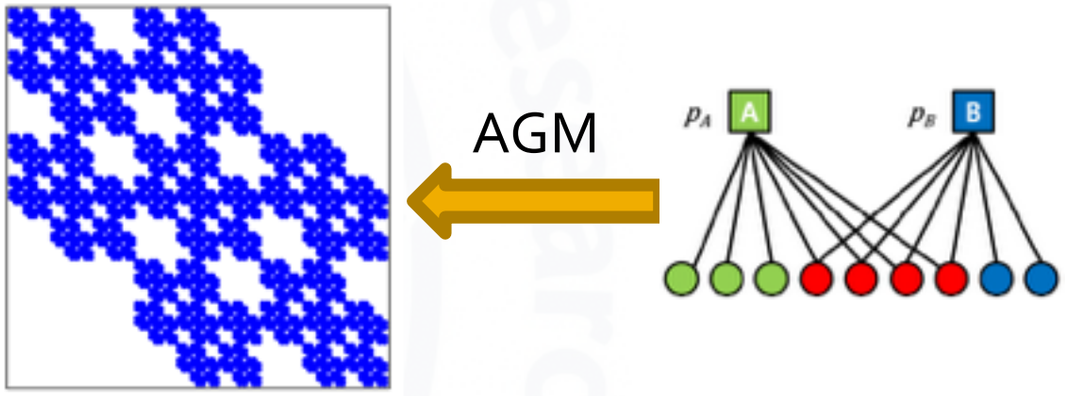


-
  - **How do we find $\boldsymbol{B(V, C, M, \{p_C\})}$ that maximizes the likelihood?**

MLE for Graphs

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

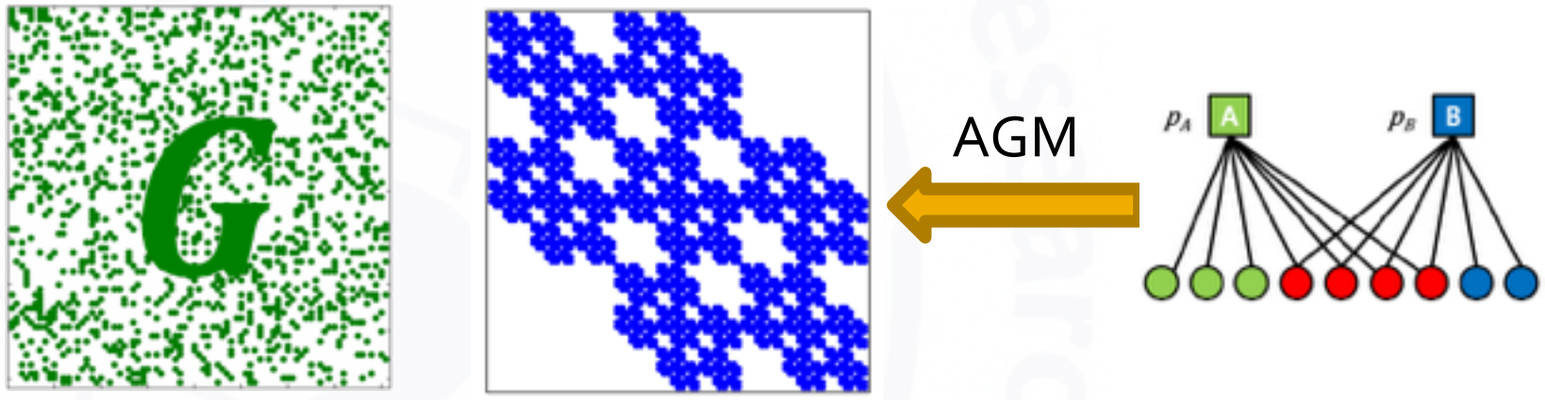
$\boldsymbol{G}$



- **How do we find $\boldsymbol{B(V, C, M, \{p_C\})}$ that maximizes the likelihood?**

MLE for Graphs

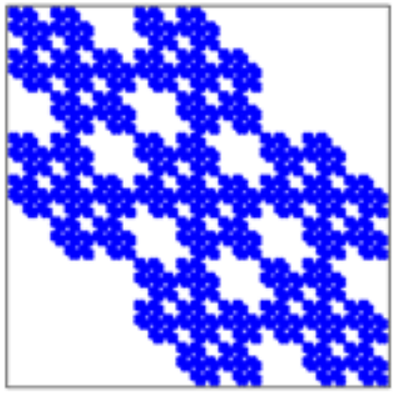
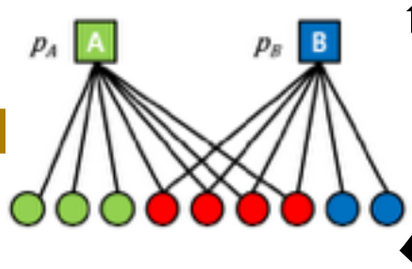
- **Our goal:** Find $\boldsymbol{\Theta = B(V, C, M, \{p_C\})}$ **such that:**



- **How do we find $\boldsymbol{B(V, C, M, \{p_C\})}$ that maximizes the likelihood?**

MLE for Graphs

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

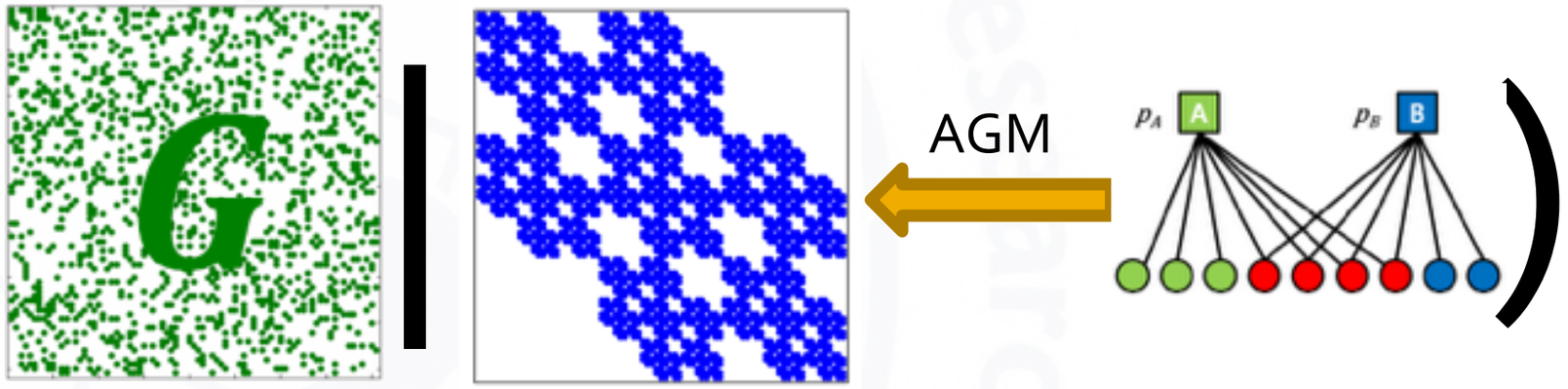
$$P($$  $$\mid$$   $$)$$

- **How do we find $\boldsymbol{B(V, C, M, \{p_C\})}$ that maximizes the likelihood?**

MLE for Graphs

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

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



- **How do we find $B(V, C, M, \{p_C\})$ that maximizes the likelihood?**

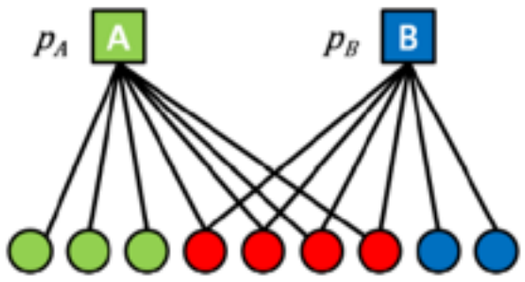
$$P(G \mid \Theta) = \prod_{(u,v)\in E} P(u,v) \prod_{(u,v)\notin E} (1 - P(u,v))$$

MLE for AGM

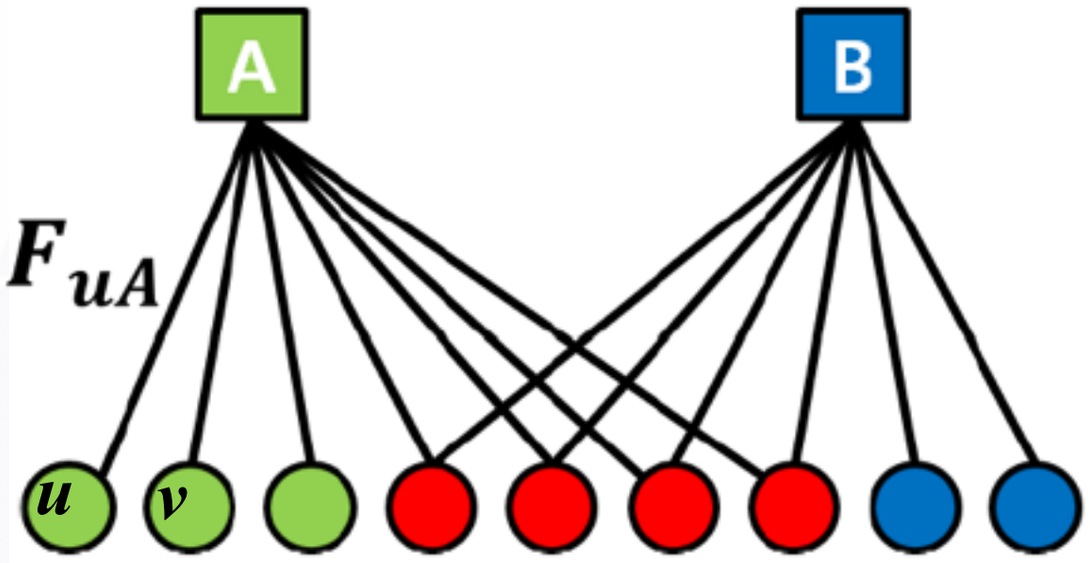
- **Our goal is to find $B(V, C, M, \{p_C\})$ such that:**

$$\underset{B(V,C,M,\{p_C\})}{\arg\max} \prod_{u,v \in E} P(u,v) \prod_{uv \notin E} (1 - P(u,v))$$

- **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)!**

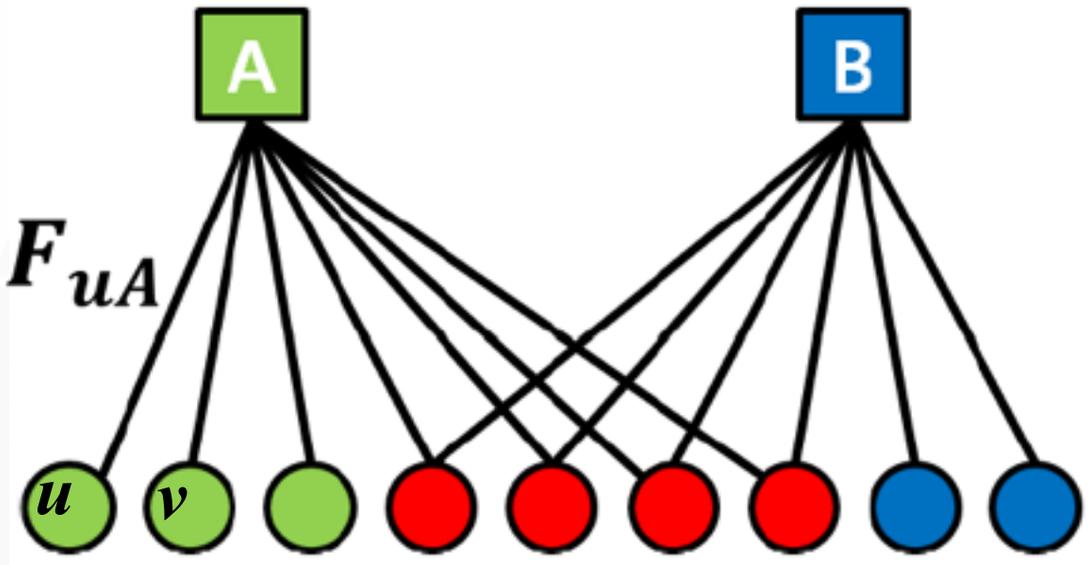


From AGM to BigCLAM



From AGM to BigCLAM

- **Relaxation: Memberships have strengths**

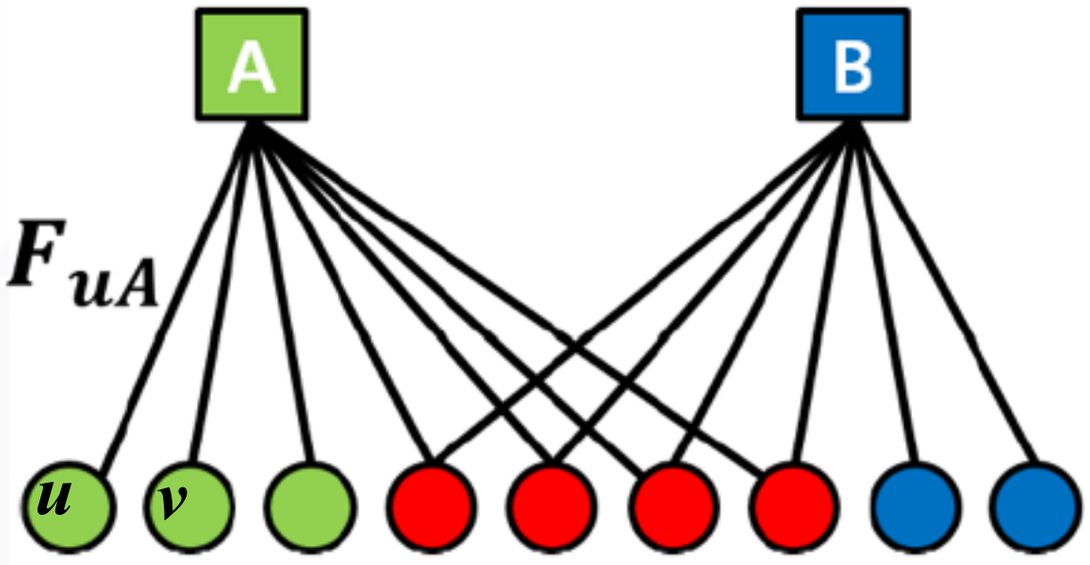


- F_{uA}: The membership strength of node u to community A ($F_{uA} = 0$: no membership)
- **Each community A links nodes independently:**

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

From AGM to BigCLAM

- **Relaxation: Memberships have strengths**

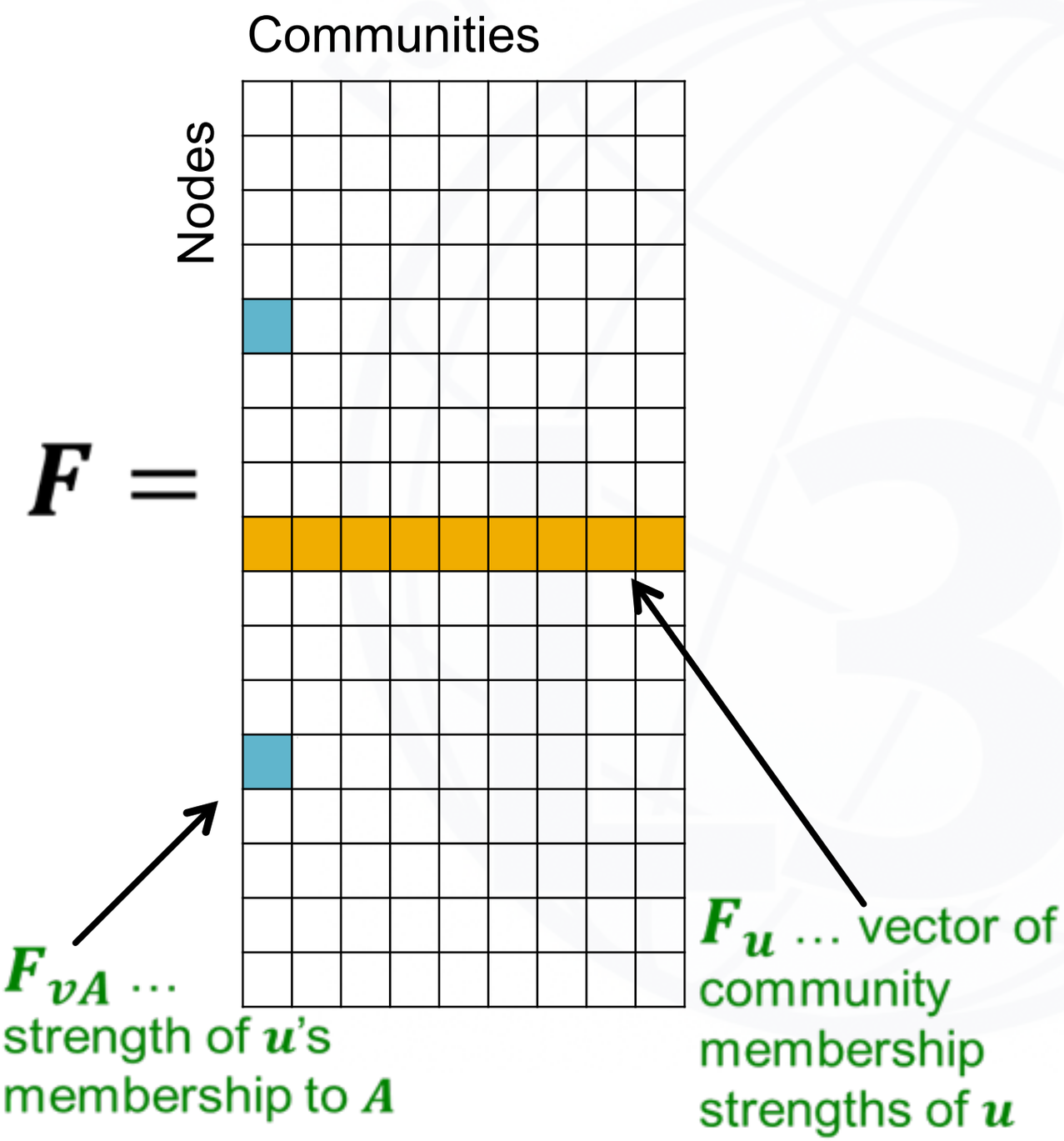


- F_{uA}: The membership strength of node u to community A ($F_{uA} = 0$: no membership)
- **Each community A links nodes independently:**

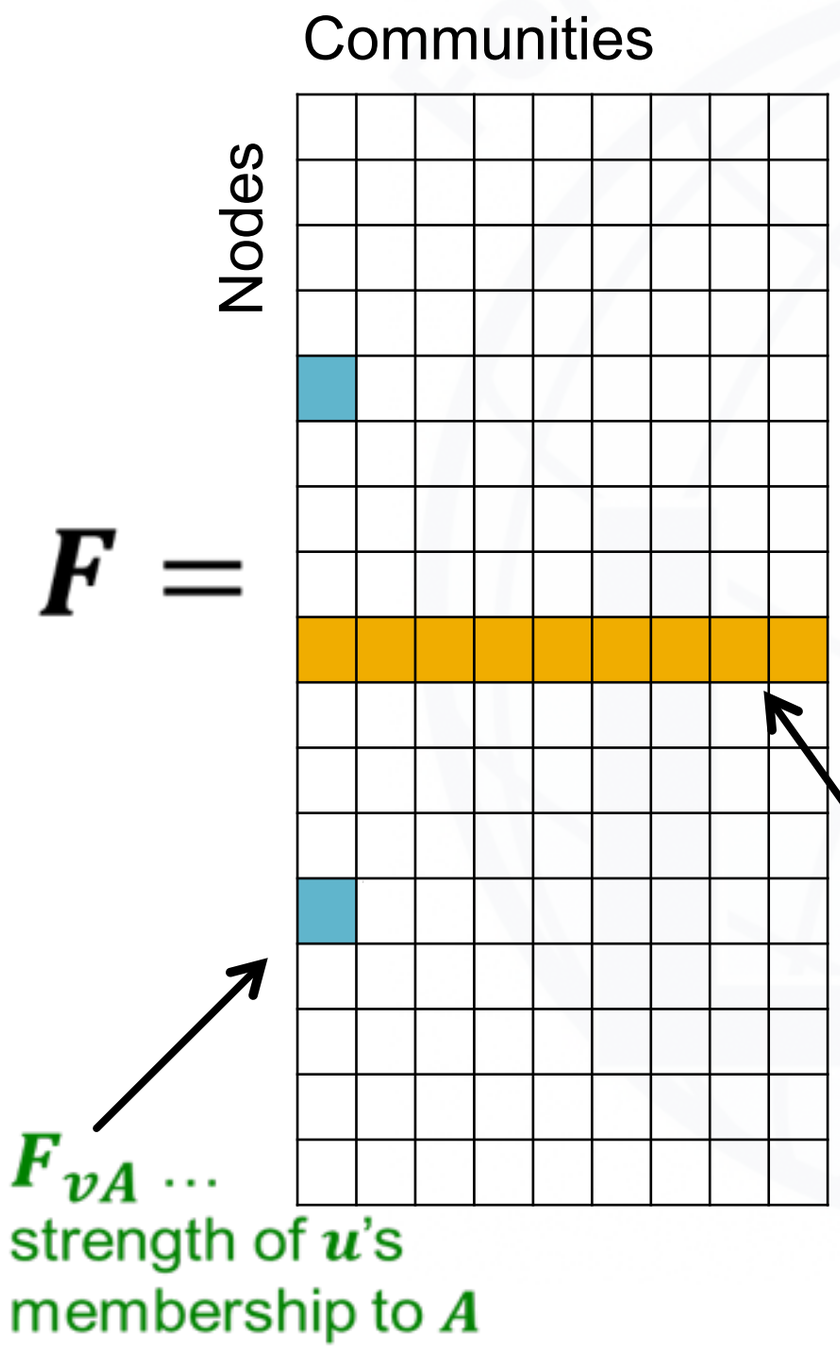
$$P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$$

Assume if a pair of nodes are strongly related to a community then they have a high probability of a connection

Community membership strength 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 node doesn't belong to the community ($F_{uC} = 0$) then $P(u, v) = 0$
- 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

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:**

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

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 likelihood:

$$arg\ max_F \prod_{(u,v)\in E} P(u,v) \prod_{(u,v)\notin E} (1 - P(u,v))$$

  - where: $P(u,v) = 1 - \exp(-F_u \cdot F_v^T)$
  - Many times we take the logarithm of the likelihood, and call it log-likelihood: $l(F) = \log P(G|F)$
- **Goal:** Find F that maximizes $l(F)$:

BigCLAM: How to find F

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

$$arg\ max_F \prod_{(u,v)\in E} P(u,v) \prod_{(u,v)\notin E} (1 - P(u,v))$$

  - where: $P(u,v) = 1 - \exp(-F_u \cdot F_v^T)$
  - Many times we take the logarithm of the likelihood, and call it log-likelihood: $l(F) = \log P(G|F)$
- **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

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

$\mathcal{N}(u)$.. 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:**
  - **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!

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

BigCLAM: V2.0

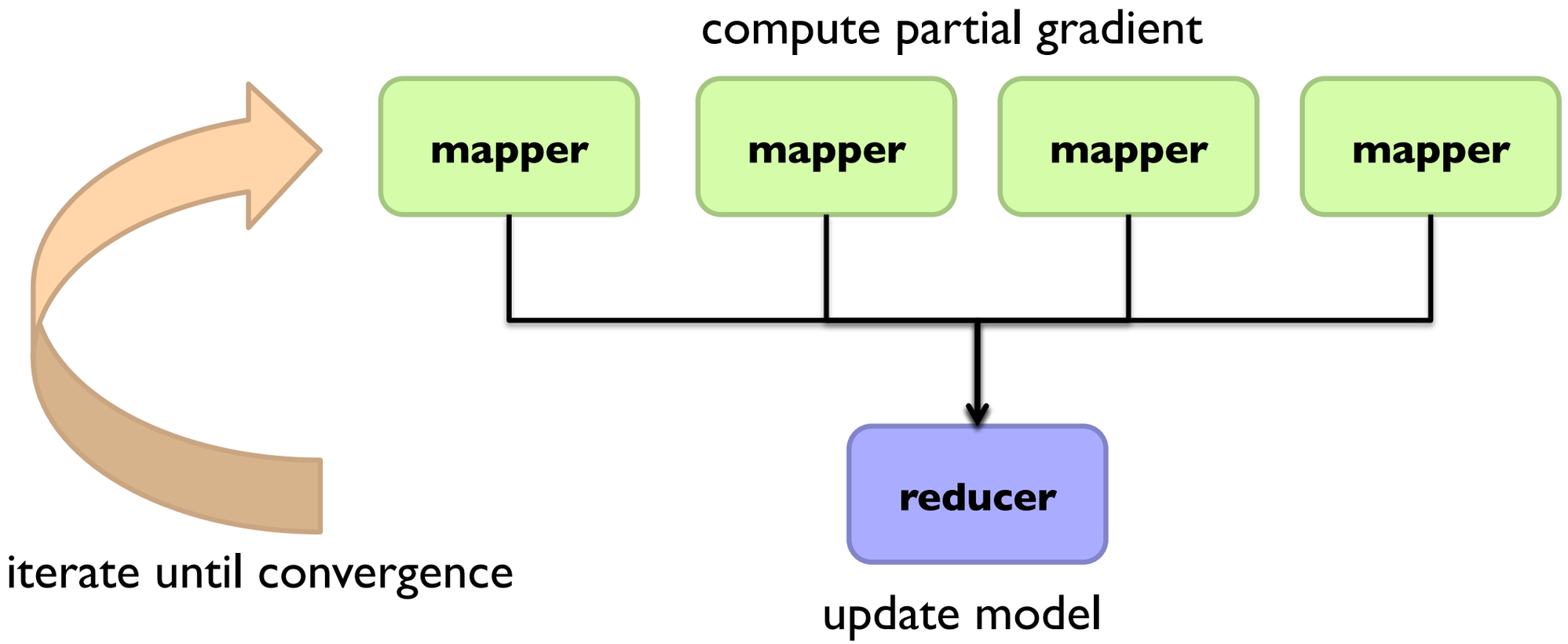
- **However, we notice:**

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

- - We cache $\sum_v \boldsymbol{F_v}$
  - So, computing $\sum_{v \notin \mathcal{N}(u)} \boldsymbol{F_v}$ now takes **linear time** in the degree $|\mathcal{N}(\boldsymbol{u})|$ 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!

Batch Gradient Descent in Mapreduce

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \underbrace{\frac{1}{n} \underbrace{\sum_{i=0}^{n} \nabla \ell(f(\mathrm{x}_i; \theta^{(t)}), y_i)}_{\text{mappers}}}_{\text{single reducer}}$$



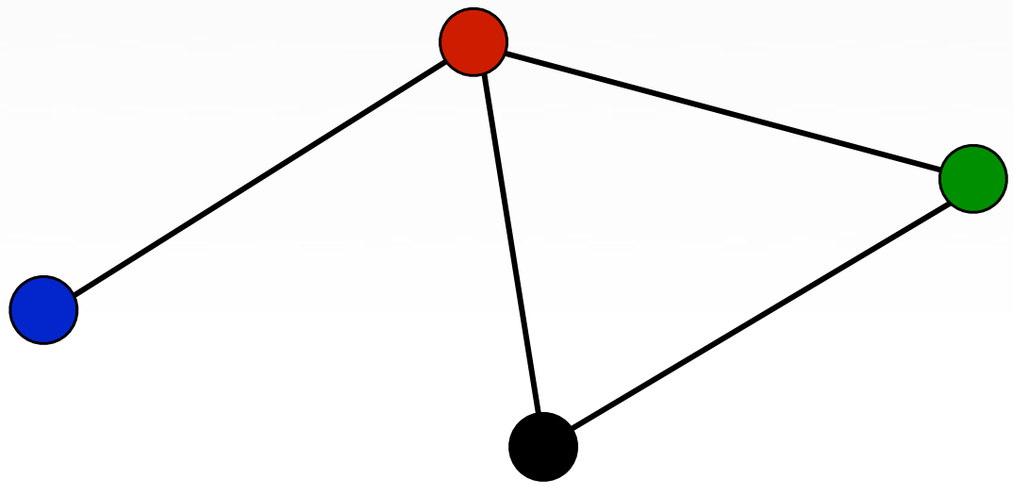
Counting Triangles

- Counting triangles in a graph is an important operation for determining the degree of connectedness in a neighbourhood

- Clustering Coefficient : fraction of a node's neighbors who are neighbors themselves

- cc(v) $= \frac{|\{(u,w) \in E | u \in \Gamma(v) \wedge w \in \Gamma(v)\}|}{\binom{d_v}{2}} = \frac{\#\Delta s \text{ incident on } v}{\binom{d_v}{2}}$

Clustering Co-efficient



cc ( ) = N/A

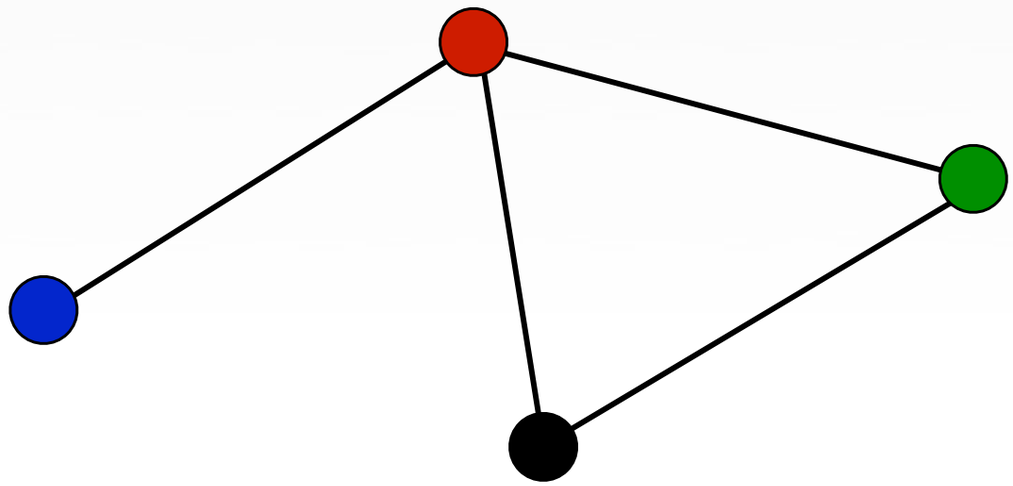
cc ( ) = 1/3

cc ( ) = 1

cc ( ) = 1

$$\frac{\#\Delta s \text{ incident on } v}{\binom{d_v}{2}}$$

Clustering Co-efficient



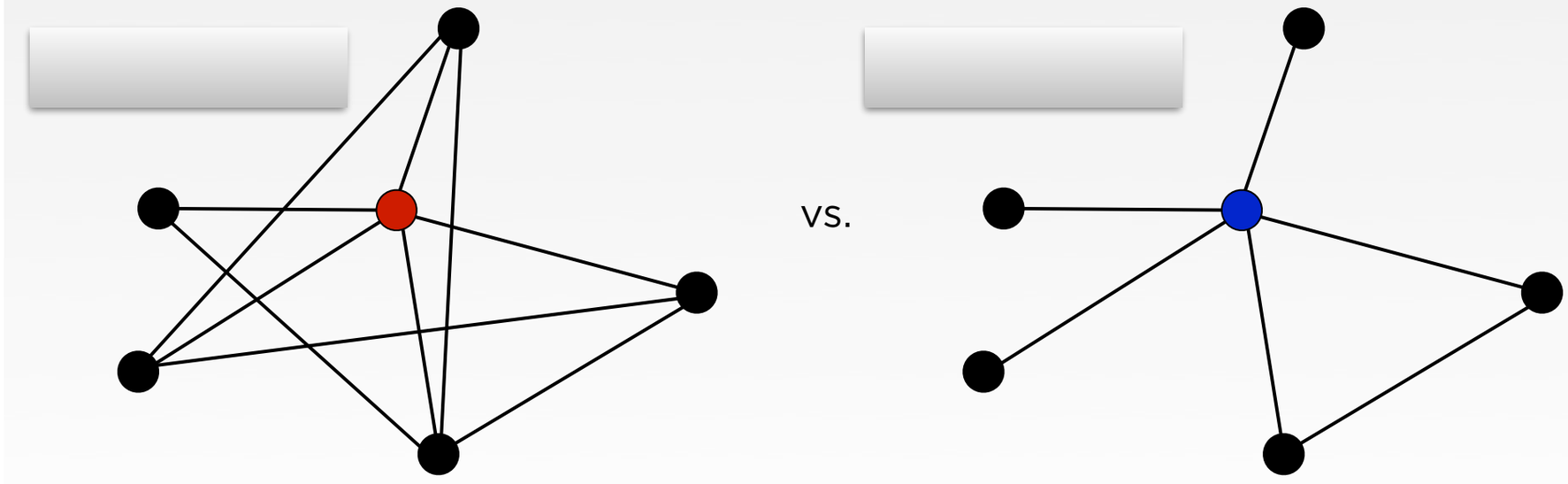
cc (●) = N/A

cc (●) = 1/3

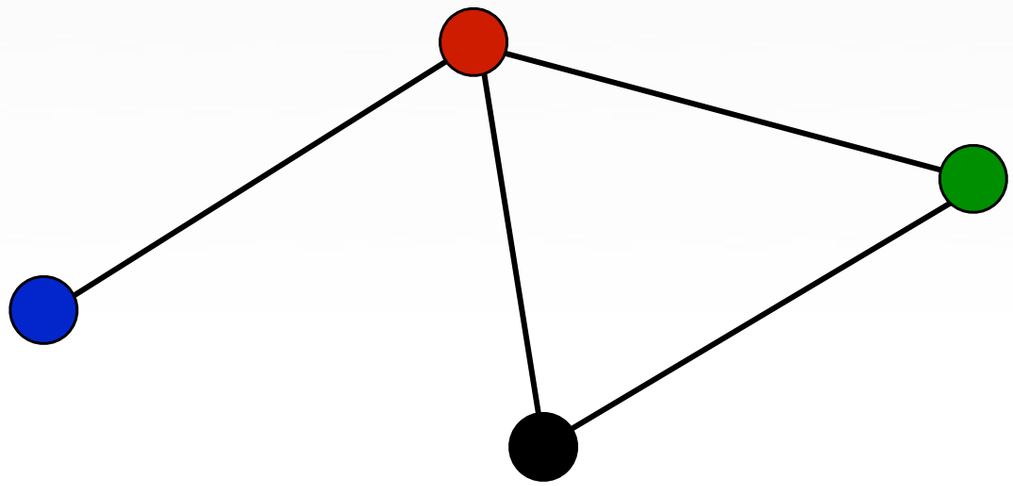
cc (●) = 1

cc (●) = 1

$$\frac{\#\Delta s \text{ incident on } v}{\binom{d_v}{2}}$$



Clustering Co-efficient



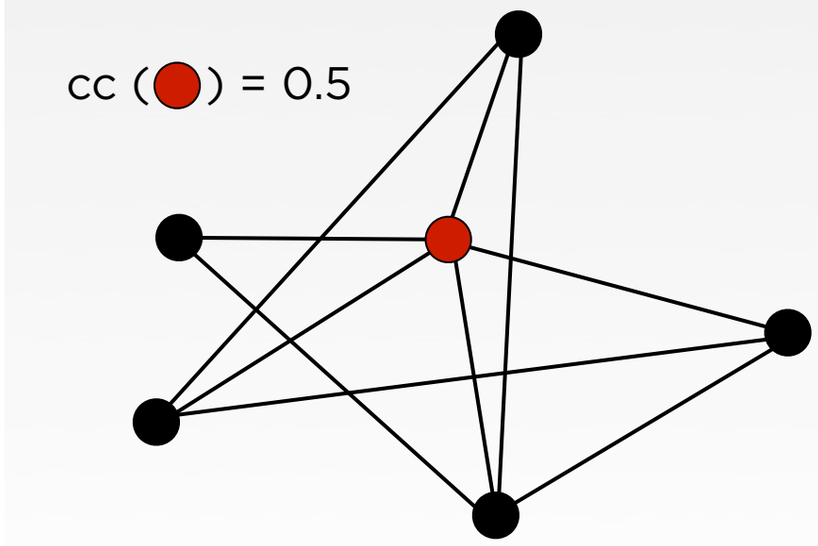
cc ( ) = N/A

cc ( ) = 1/3

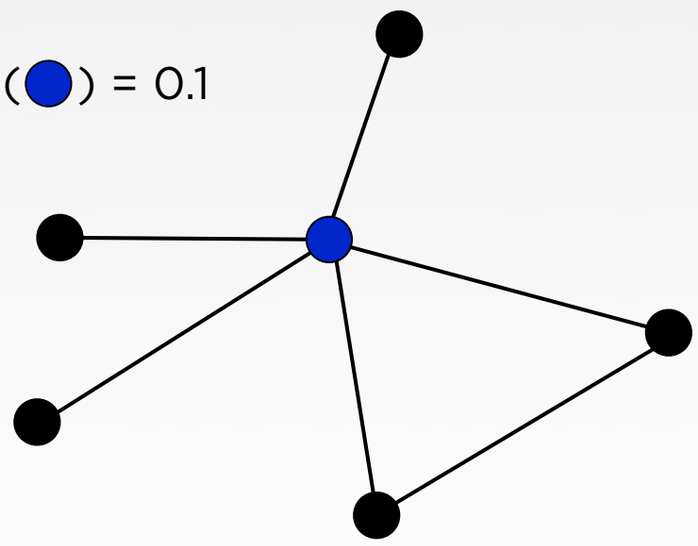
cc ( ) = 1

cc ( ) = 1

$$\frac{\#\Delta s \text{ incident on } v}{\binom{d_v}{2}}$$

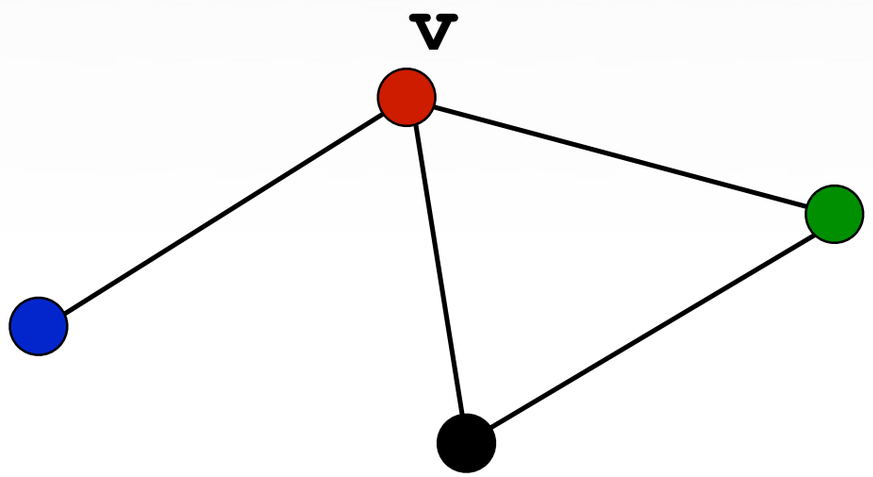


vs.



Naiive Method

```
foreach v in V
    foreach u,w in Adjacency(v)
        if (u,w) in E
            Triangles[v]++
```



What is its complexity ?

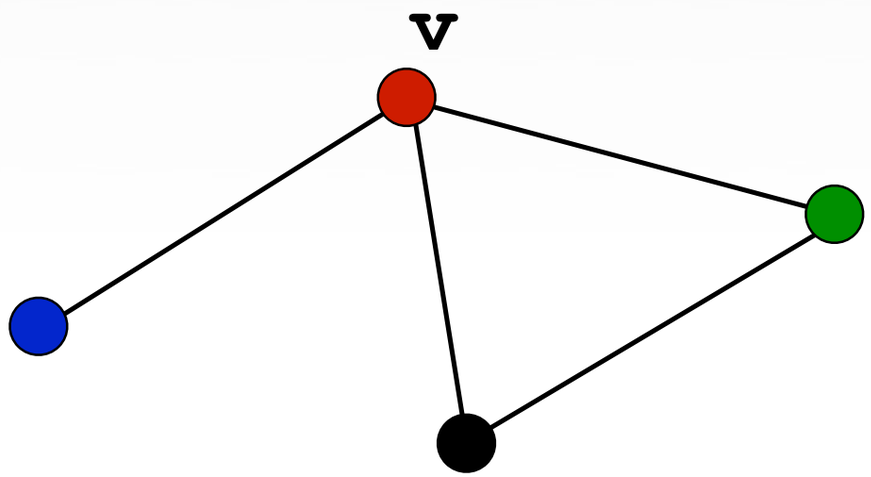
Possible graph inputs: (a) adjacency list , (b) edge list

Preprocessing : Create a hashset of (u,w) pairs for constant lookup

Preprocessing : Create adjacency list **O(m)**

Naiive Method - Complexity

```
foreach v in V
    foreach u,w in Adjacency(v)
        if (u,w) in E
            Triangles[v]++
```



Running time $\sum_{v \in V} d_v^2$

For sparse graphs : quadratic even if a few nodes have large out degree

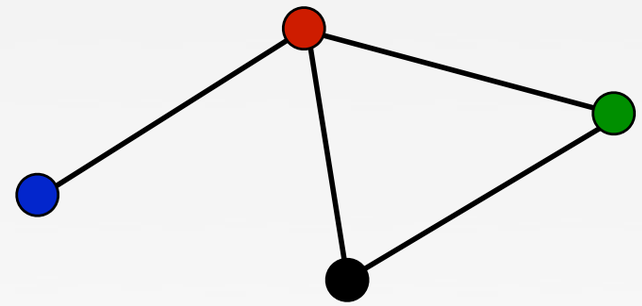
Parallel Method

2-stage Mapreduce

- Map 1: For each v send $(v, \Gamma(v))$ to single machine.
- Reduce 1: Input: $\langle v; \Gamma(v) \rangle$

  Output: all 2 paths $\langle (v_1, v_2); u \rangle$ where $v_1, v_2 \in \Gamma(u)$

  (●,●);● (●,●);● (●,●);●



- Map 2: Send $\langle (v_1, v_2); u \rangle$ and $\langle (v_1, v_2); \$ \rangle$ for $(v_1, v_2) \in E$ to same machine.
- Reduce 2: input: $\langle (v, w); u_1, u_2, \ldots, u_k, \$? \rangle$

  Output: if $\$$ part of the input, then: $u_i = u_i + 1/3$

(●,●);●,$ ⟶ ● +1/3 ● +1/3 ● +1/3

(●,●);● ⟶

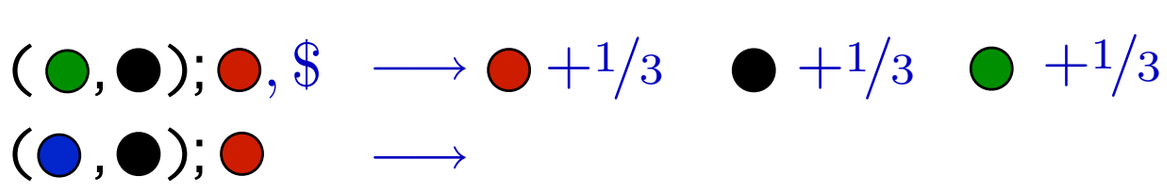
check edges in parallel

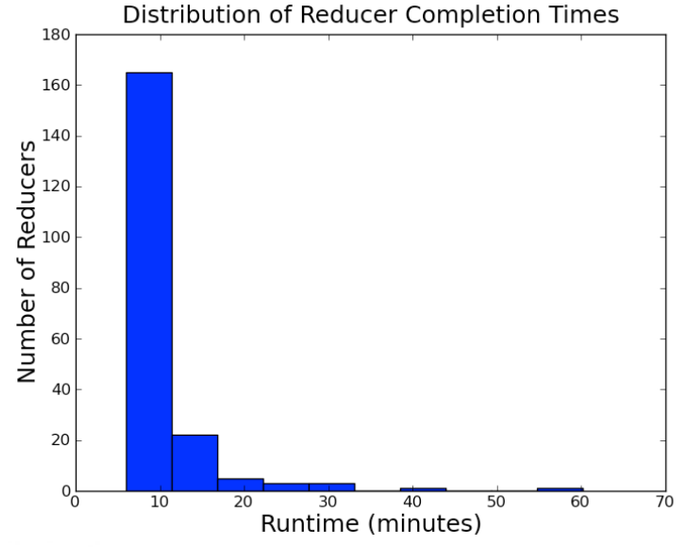
Parallel Method

2-stage Mapreduce

- Map 2: Send $\langle (v_1, v_2); u \rangle$ and $\langle (v_1, v_2); \$ \rangle$ for $(v_1, v_2) \in E$ to same machine.
- Reduce 2: input: $\langle (v, w); u_1, u_2, \ldots, u_k, \$? \rangle$

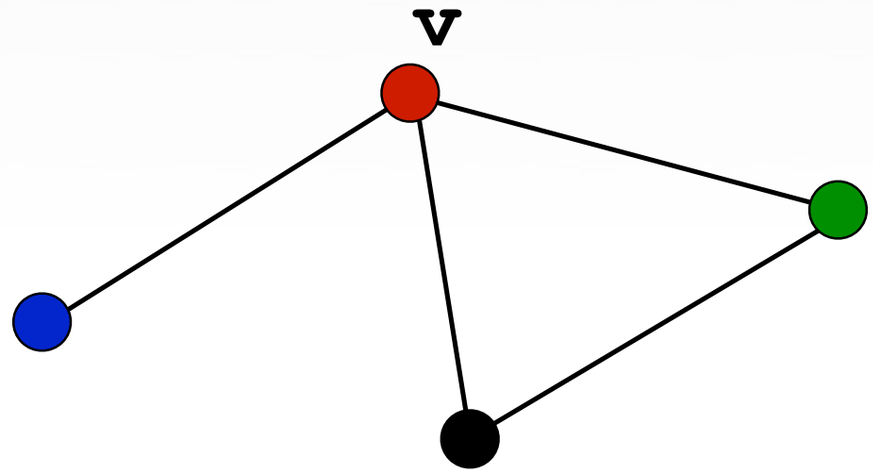
  Output: if $\$$ part of the input, then: $u_i = u_i + 1/3$





- Each triangle counted thrice
- How do we ensure that we count a triangle only once ?
- How do we handle skew ?

Heavy Hitters

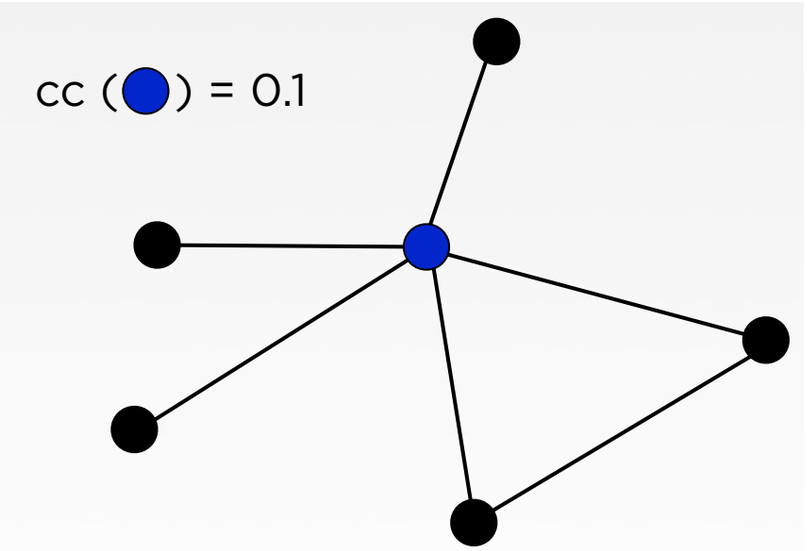


- Heavy hitter nodes: Nodes with out-degree > $m^{1/2}$
- Heavy hitter triangles are triangles with all vertices as heavy hitters
  - How many heavy hitter nodes can there be ?
- Partition nodes into heavy hitters and non heavy hitters
- Treat each of these partitions separately

Algorithm

- For Heavy hitter (HH) nodes:
  - Consider all possible combinations of HH nodes and verify if they form triangles
    - Complexity: $O(m^{3/2})$
- For non HH nodes:
  - only consider neighbours with higher degrees or numerical order
  - Complexity: $O(m^{3/2})$
    - Each edge considered only once $O(m)$ and max out-degree is $O(m^{1/2})$

Algorithm



- For Heavy hitter (HH) nodes:
  - Consider all possible combinations of HH nodes and verify if they form triangles
    - Complexity: $O(m^{3/2})$

```
foreach v in V
  foreach u,w in Adjacency(v)
    if deg(u) > deg(v) && deg(w) > deg(v)
      if (u,w) in E
        Triangles[v]++
```

- For non HH nodes:
  - only consider neighbours with higher degrees or numerical order
  - Complexity: $O(m^{3/2})$
    - Each edge considered only once $O(m)$ and max out-degree is $O(m^{1/2})$

Map reduce Modification

parallelized

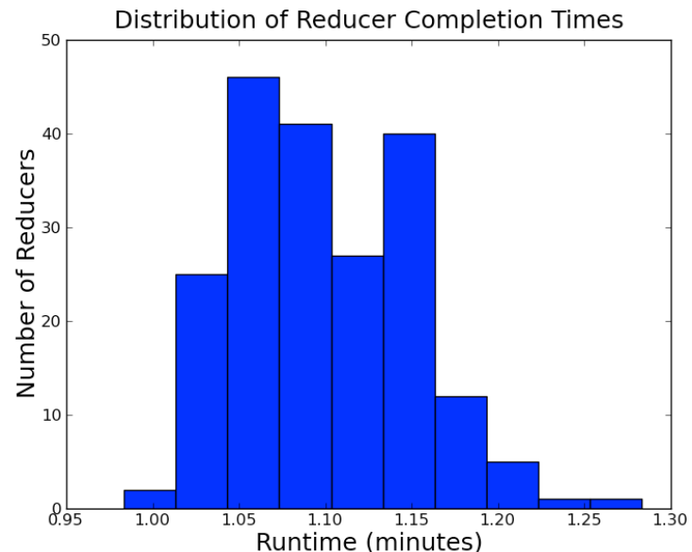
- Assume an ordering of the nodes : $u_1 < u_2 < \ldots u_n$
- Minor modification in the 2nd MR step:

  - Map 2: Send $\langle (v_1, v_2); u \rangle$ and $\langle (v_1, v_2); \$ \rangle$ for $(v_1, v_2) \in E$ to same machine.

    - **Map:** emit <(v_1 , v_2); u> only if u < v_1 and u < v_2

  - Reduce 2: input: $\langle (v, w); u_1, u_2, \ldots, u_k, \$? \rangle$

    - Better parallelisation



Summary

- Community detection
  - Overlapping communities have high edge density in the overlapping region
  - AGM to model overlapping communities and BIGCLAM as a relaxed version
  - Efficiently performing MLE using caching and MR
- Counting triangles
  - Heavy hitters and partitioning for skew and runtime improvement
  - Exploiting order for better MR performance

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.