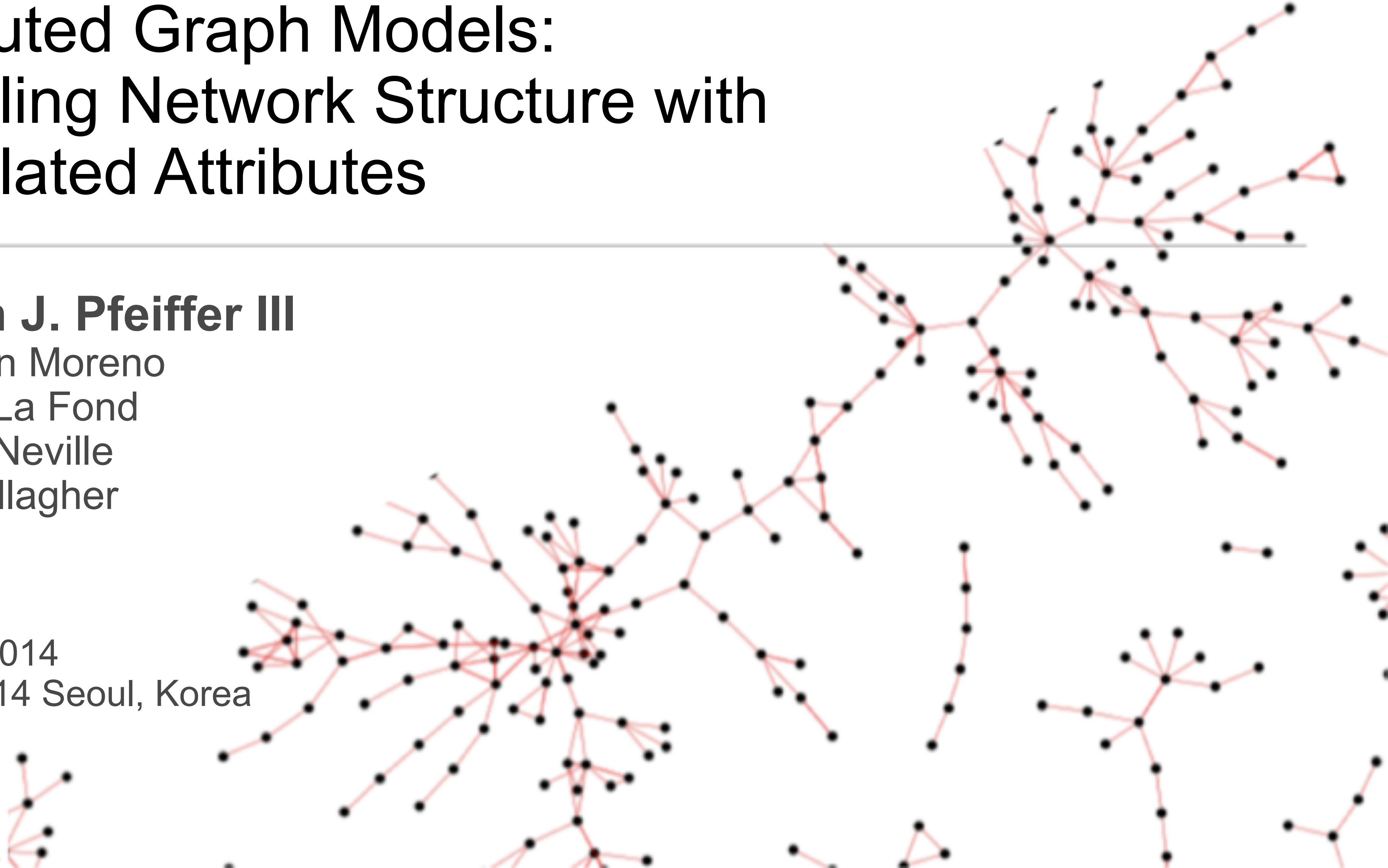
Attributed Graph Models: Modeling Network Structure with Correlated Attributes

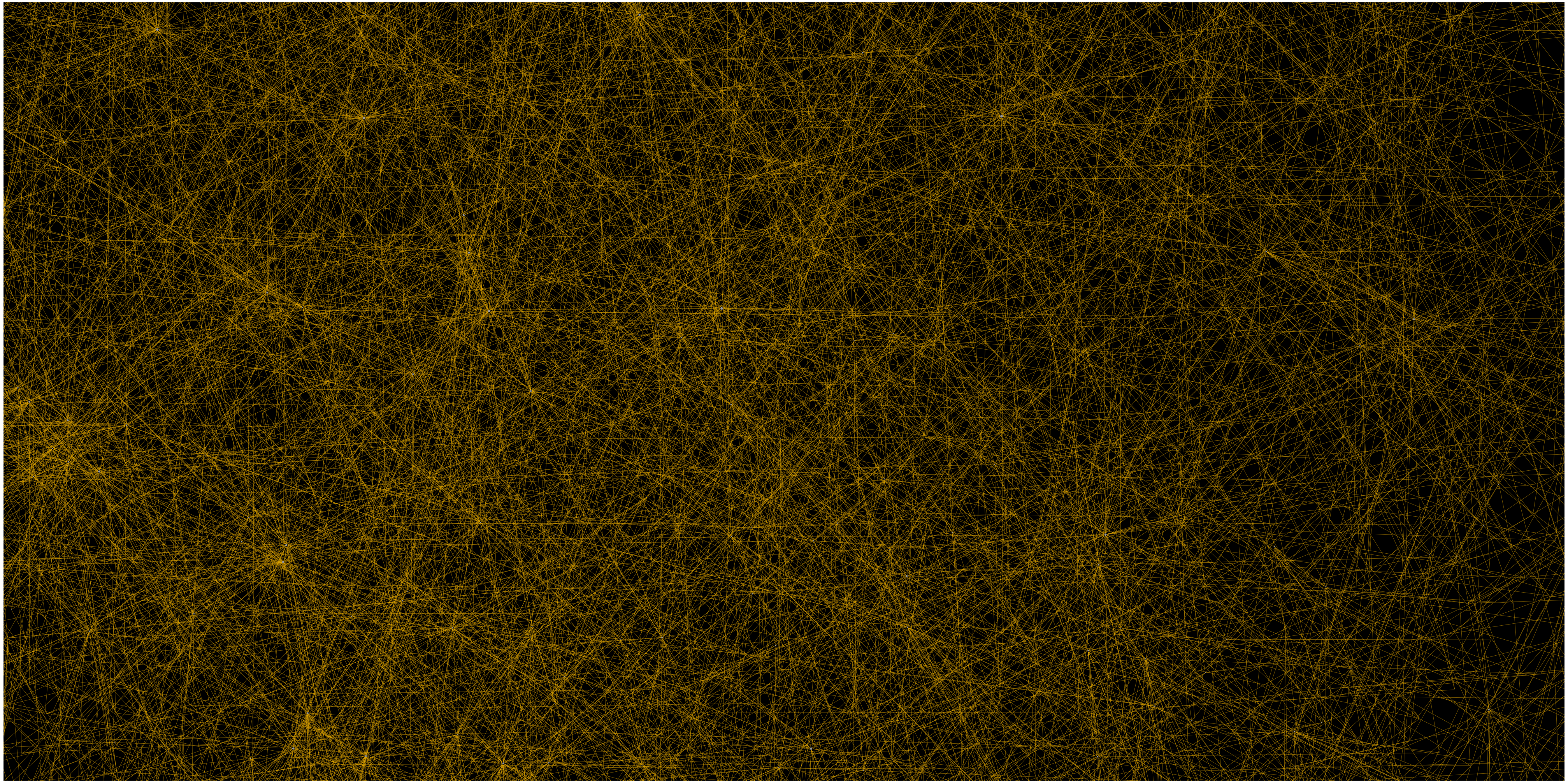
Joseph J. Pfeiffer III
Sebastian Moreno
Timothy La Fond
Jennifer Neville
Brian Gallagher

April 11, 2014
WWW 2014 Seoul, Korea



PURDUE
UNIVERSITY

Let’s look at a network...



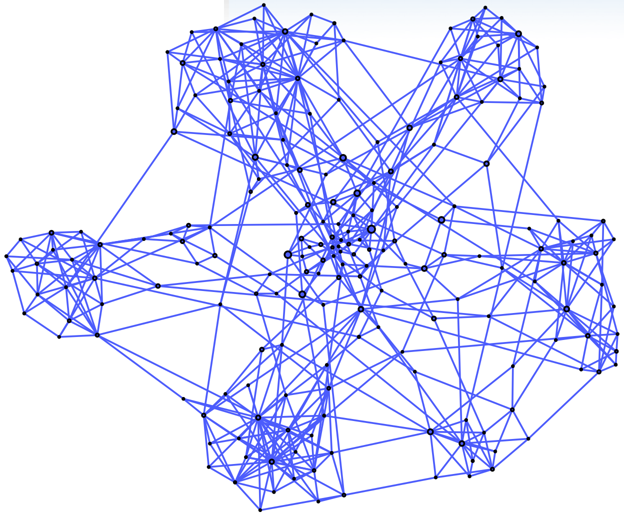
Scalable Generative Graph Models

Scalable Generative Graph Models

Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$

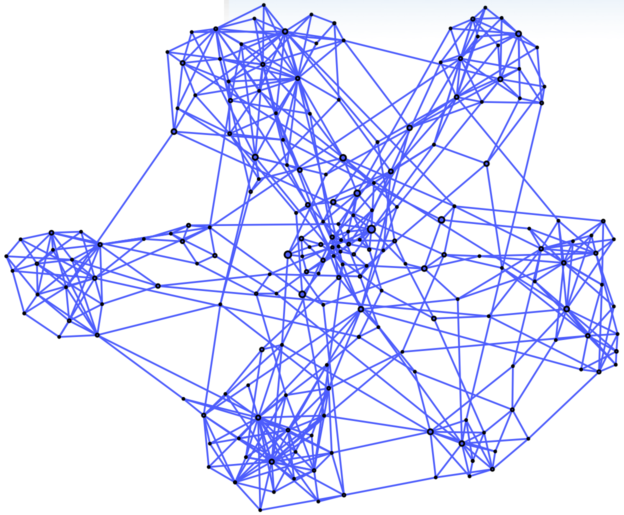
Scalable Generative Graph Models

Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$



Scalable Generative Graph Models

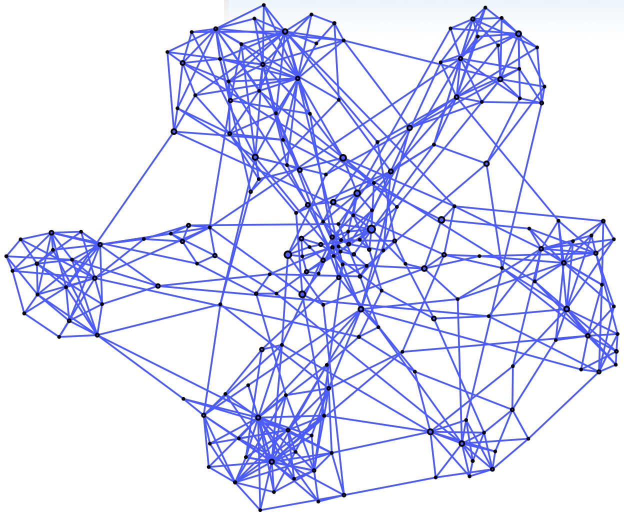
Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$

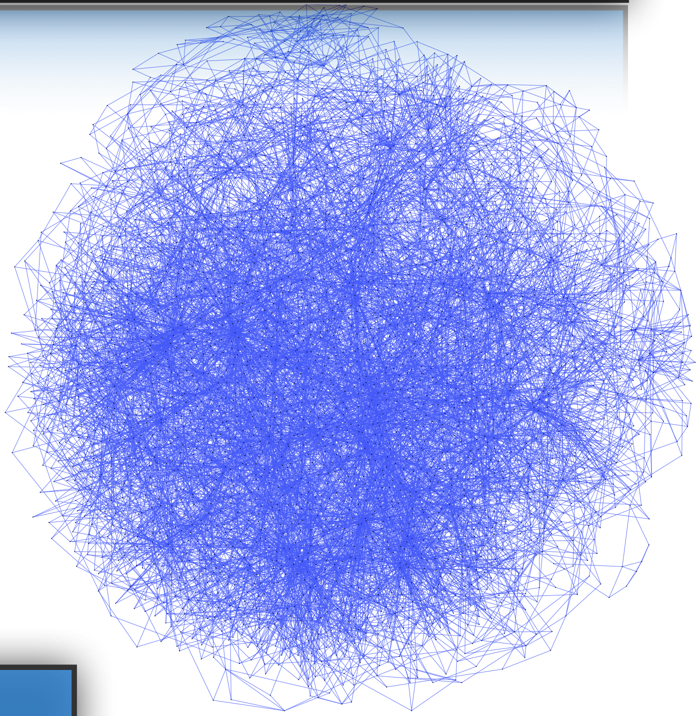


Evaluate on
Future Structure

Scalable Generative Graph Models

Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$

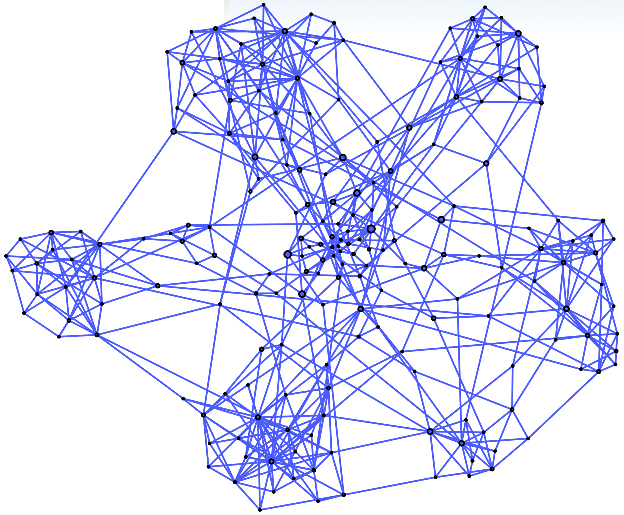




Evaluate on
Future Structure

Scalable Generative Graph Models

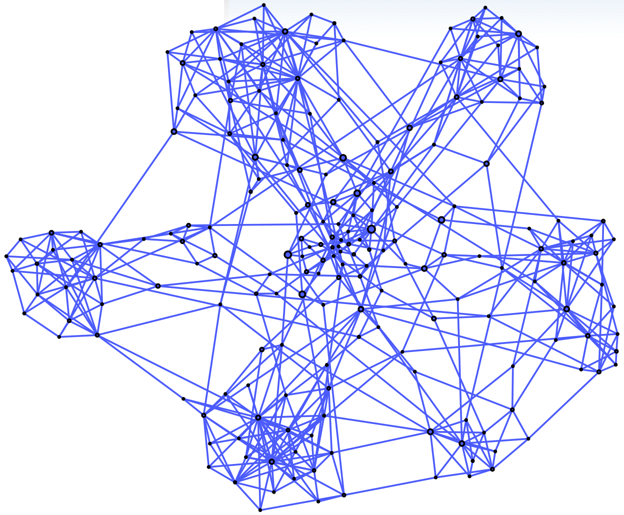
Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$



Evaluate on
Future Structure

Scalable Generative Graph Models

Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$

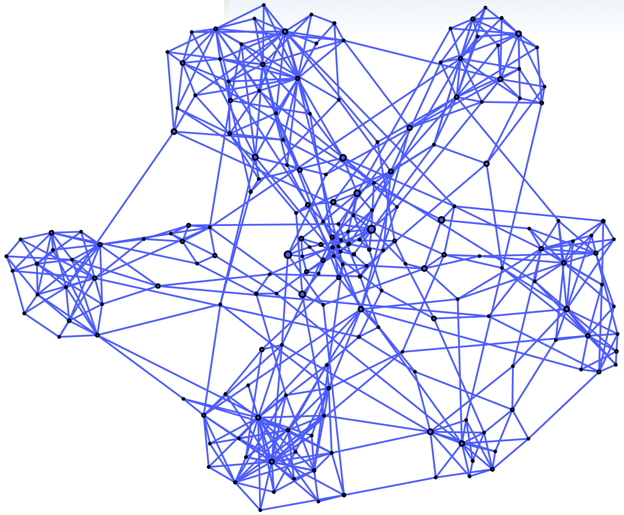


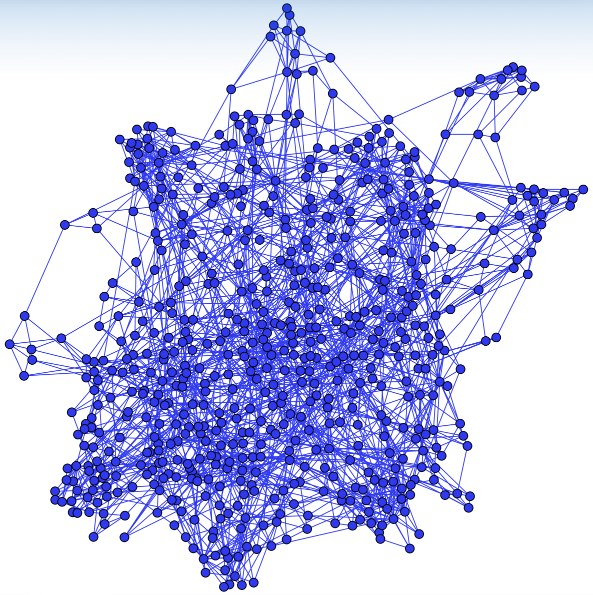
Evaluate on
Future Structure

Assess Anomalies

Scalable Generative Graph Models

Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$



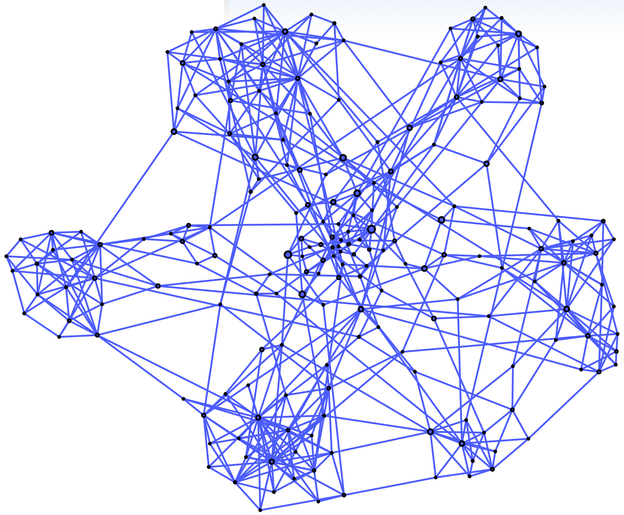


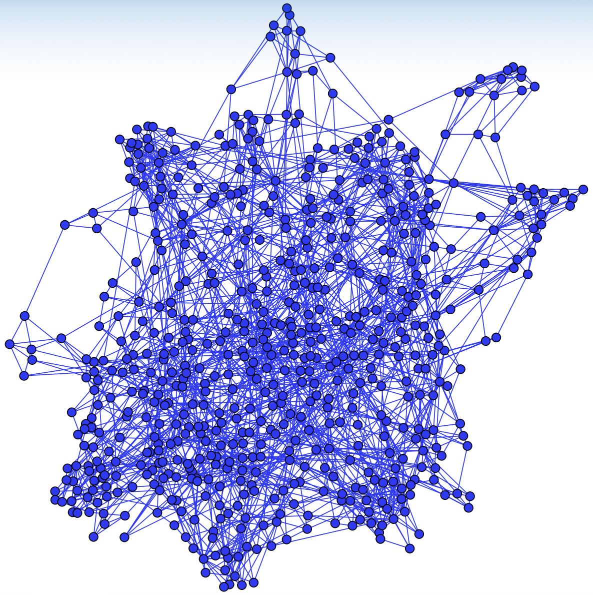
Evaluate on Future Structure

Assess Anomalies

Scalable Generative Graph Models

Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$

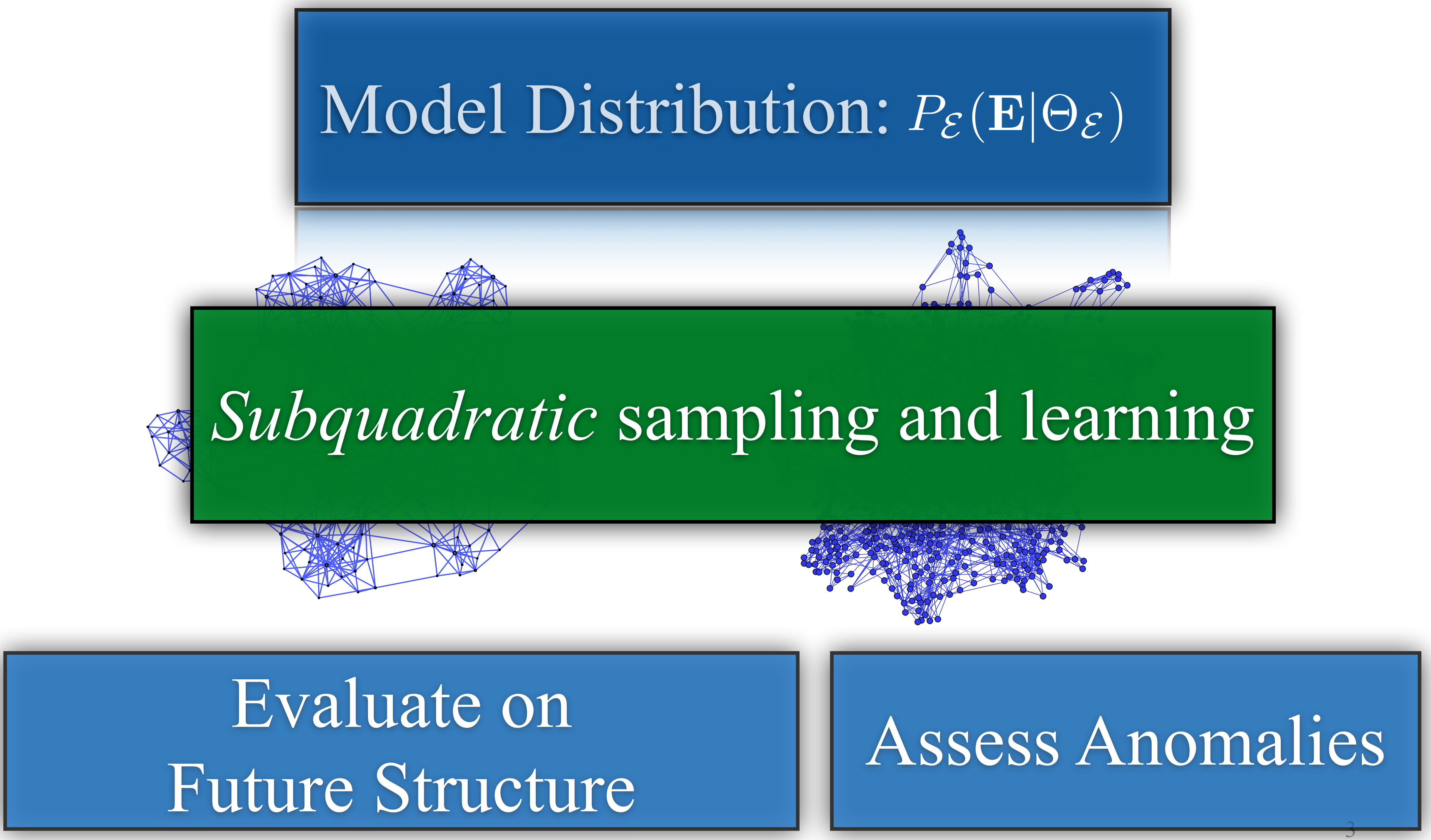




Evaluate on Future Structure

Assess Anomalies

Scalable Generative Graph Models



Model Distribution: $P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})$

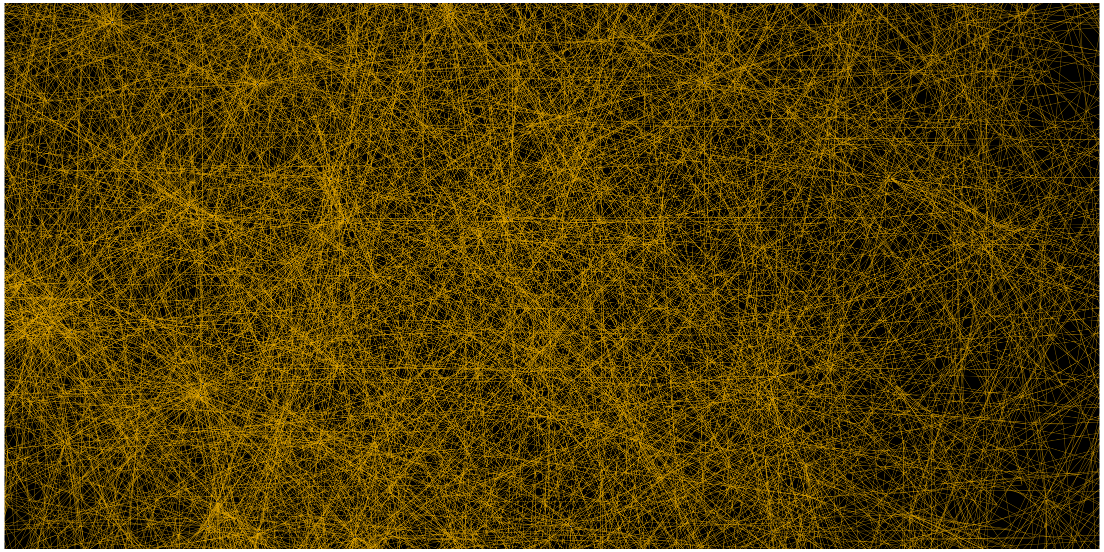
Subquadratic sampling and learning

Evaluate on Future Structure

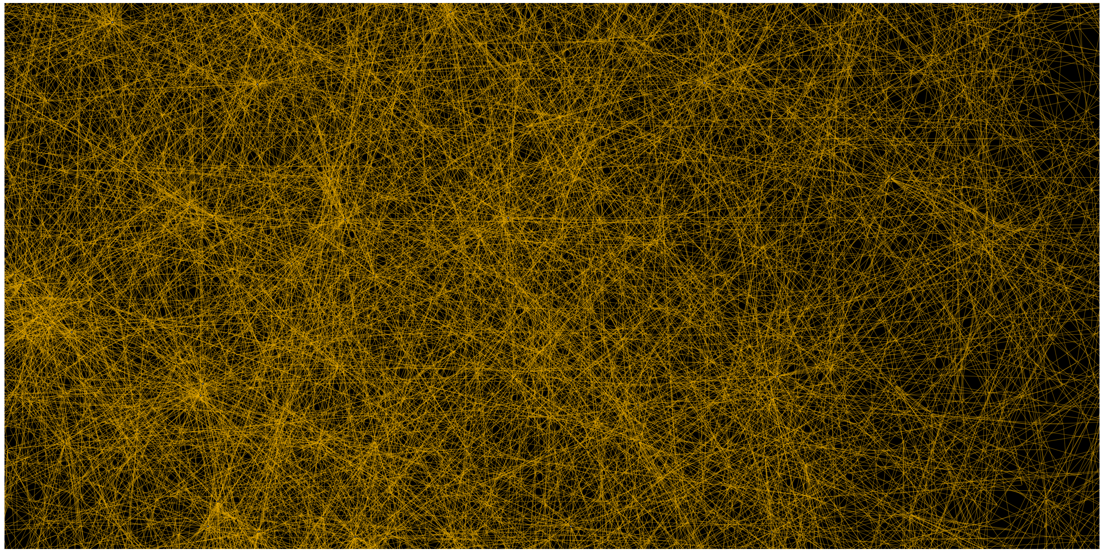
Assess Anomalies

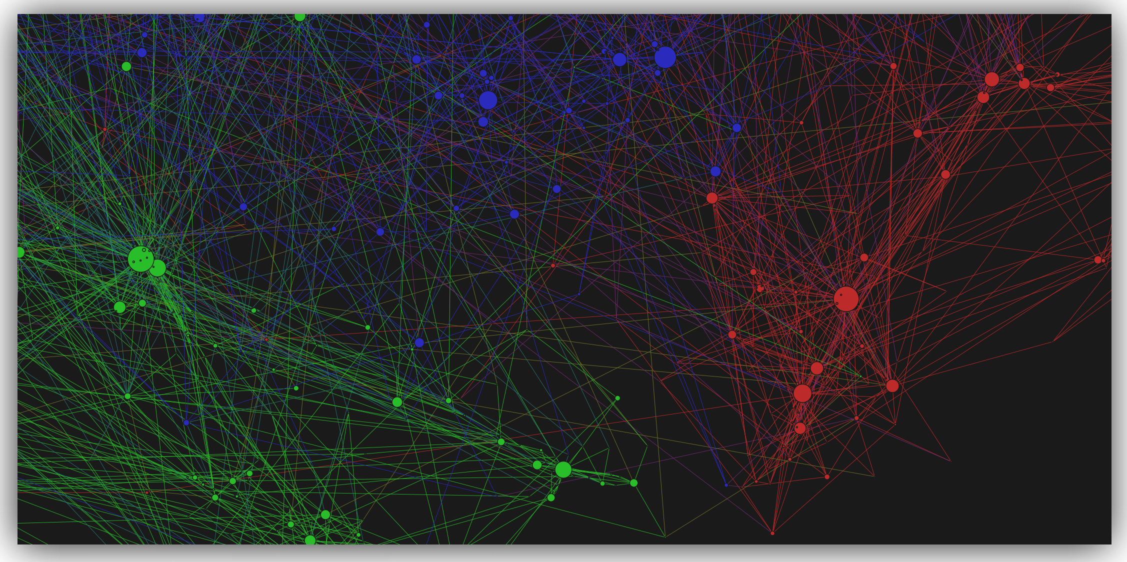
What about attributes?

What about attributes?

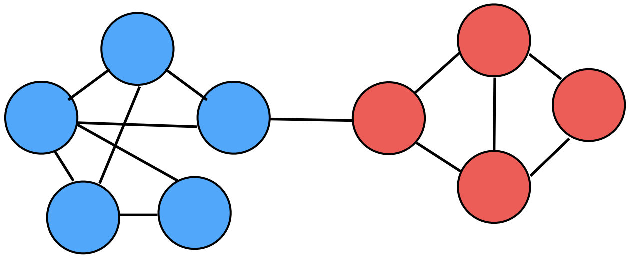


What about attributes?

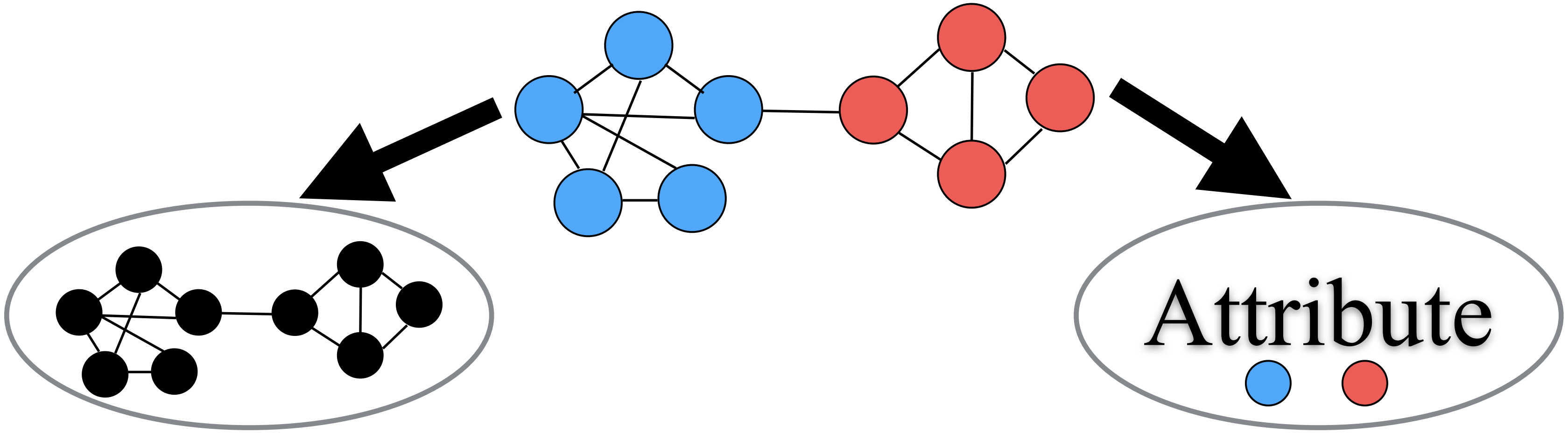




What about attributes?

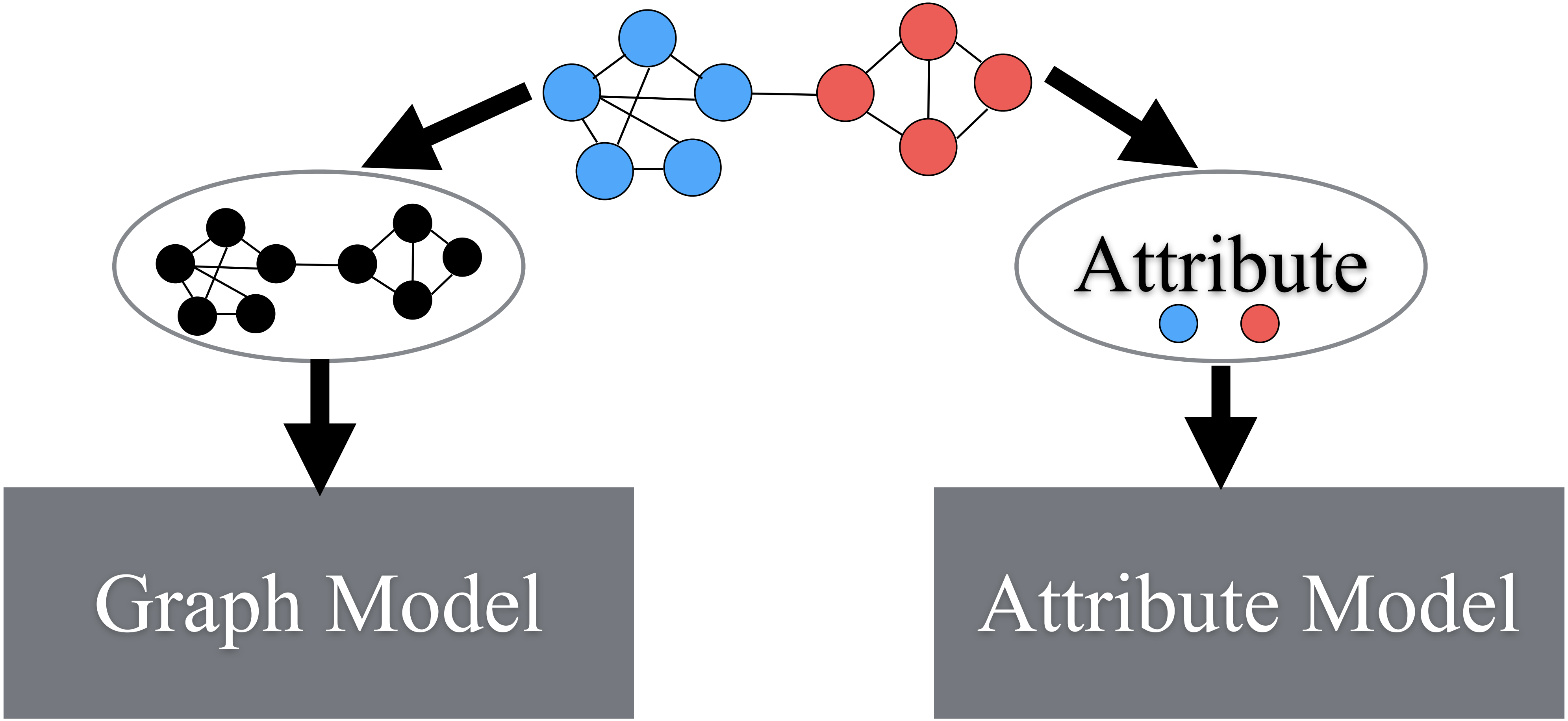


What about attributes?



Attribute

What about attributes?

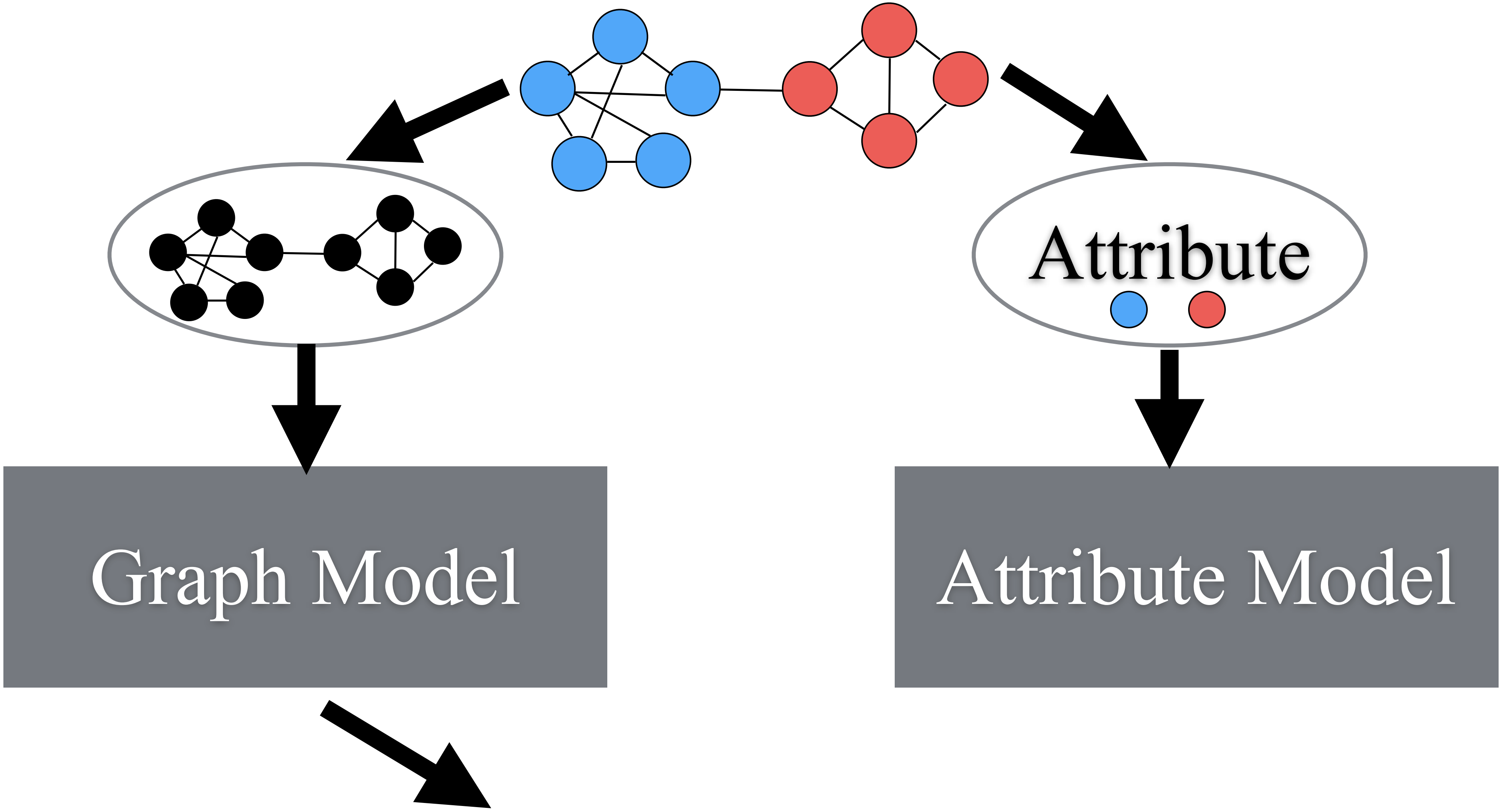


Attribute

Graph Model

Attribute Model

What about attributes?

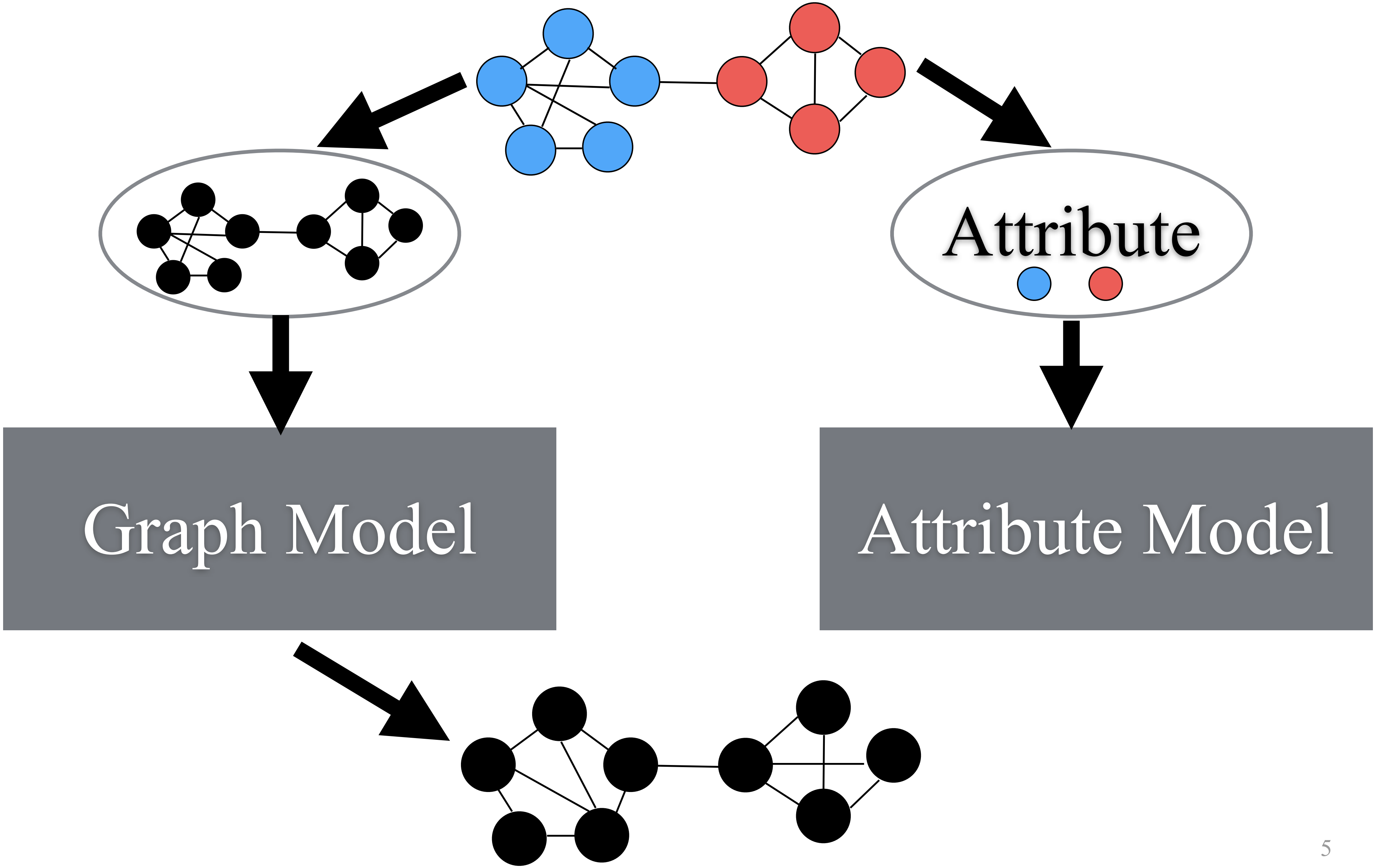


Attribute

Graph Model

Attribute Model

What about attributes?

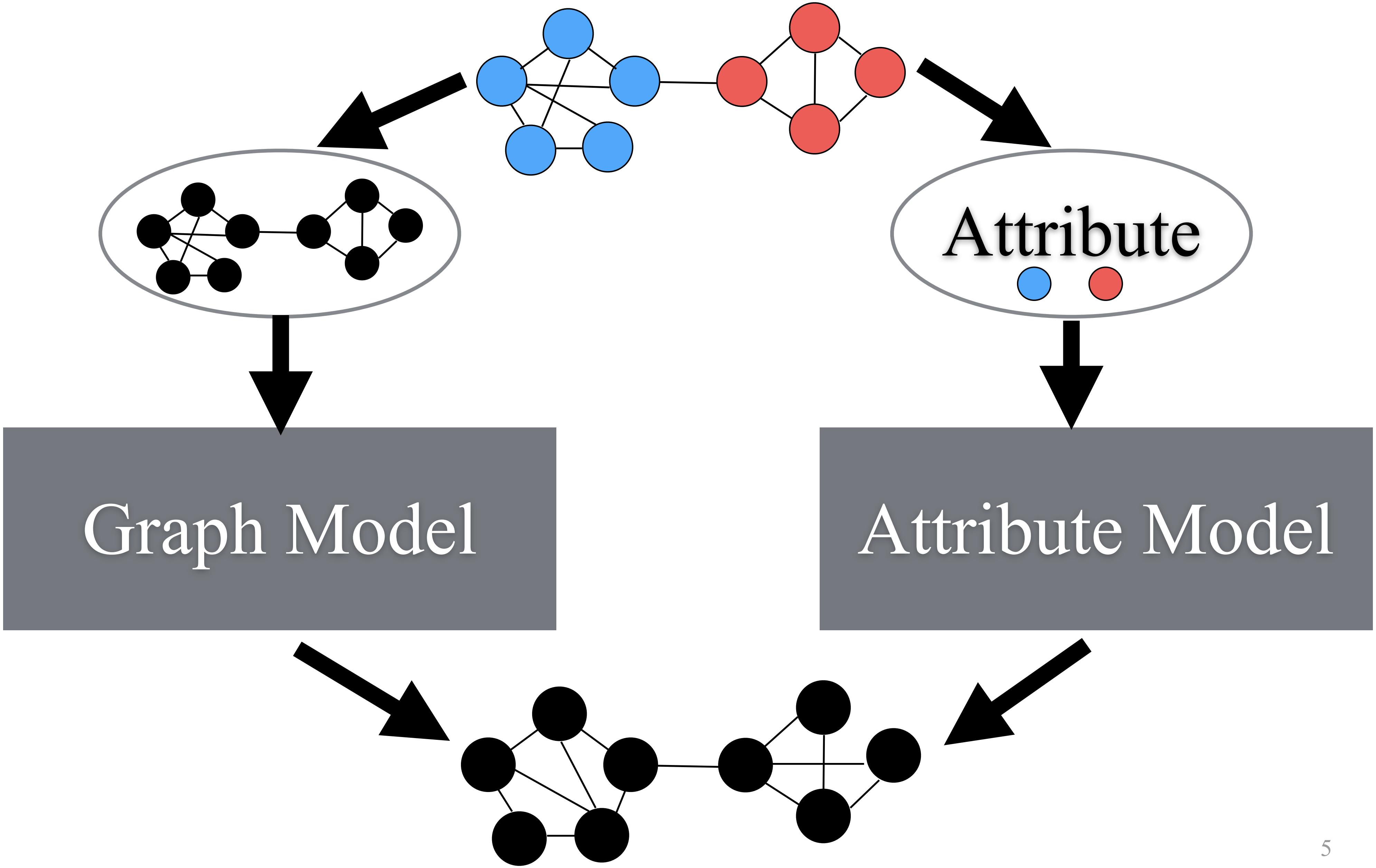


Attribute

Graph Model

Attribute Model

What about attributes?

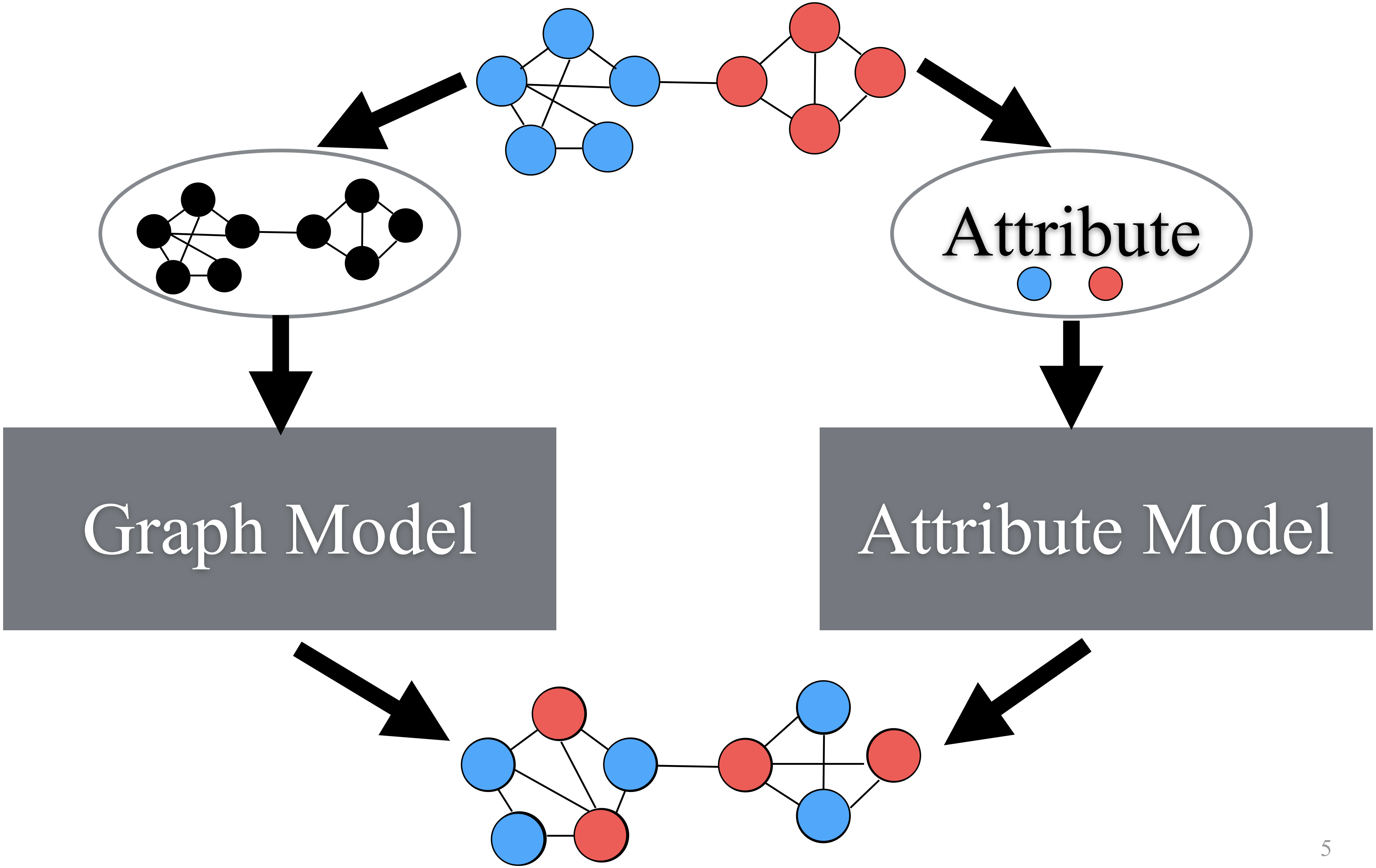


Attribute

Graph Model

Attribute Model

What about attributes?

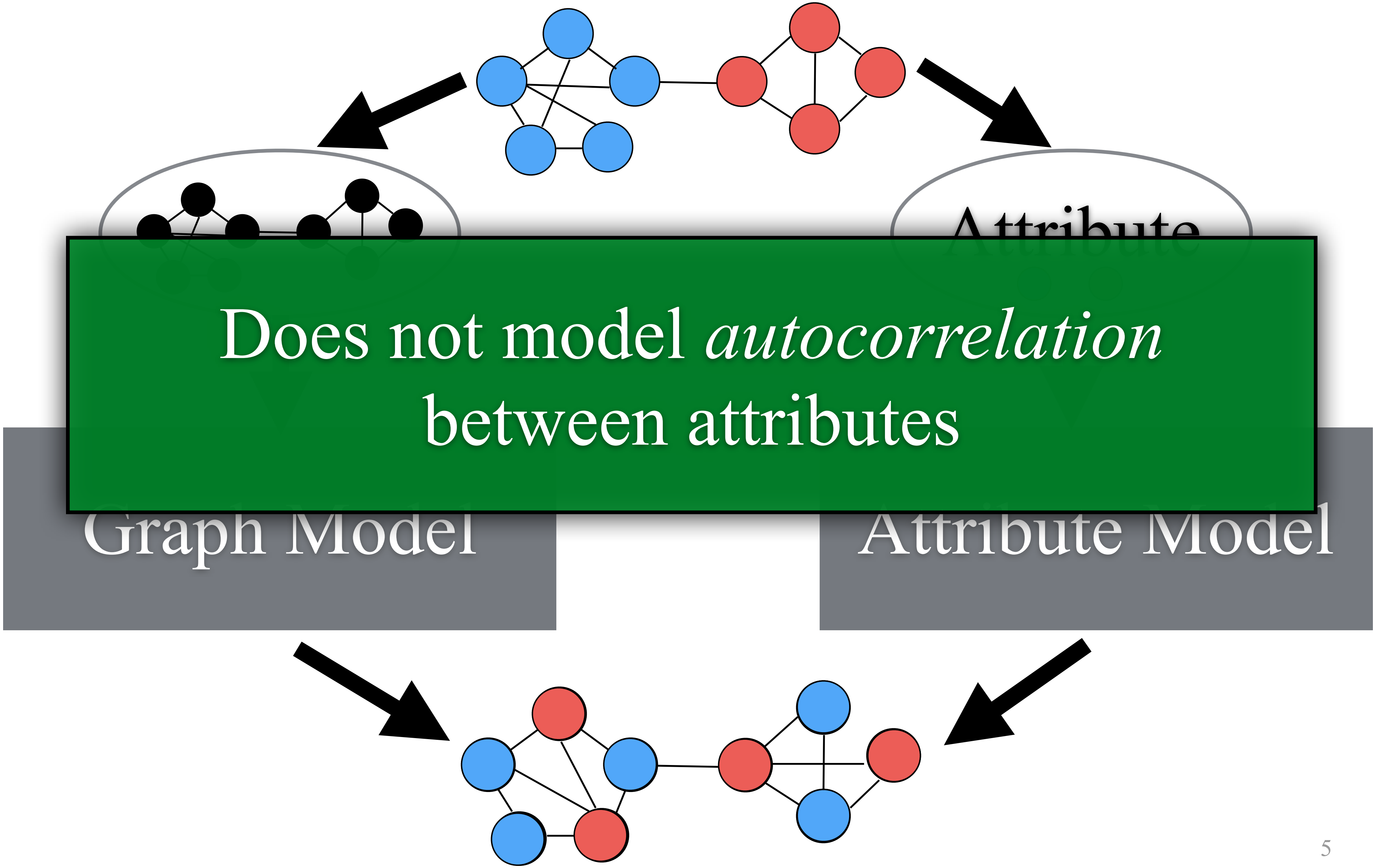


Attribute

Graph Model

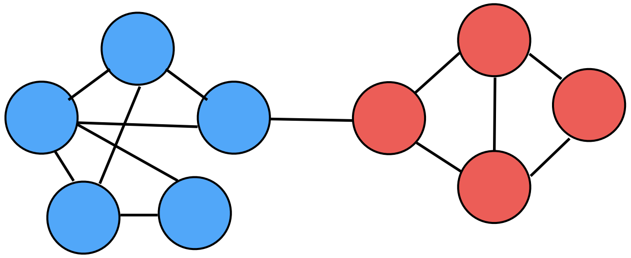
Attribute Model

What about attributes?

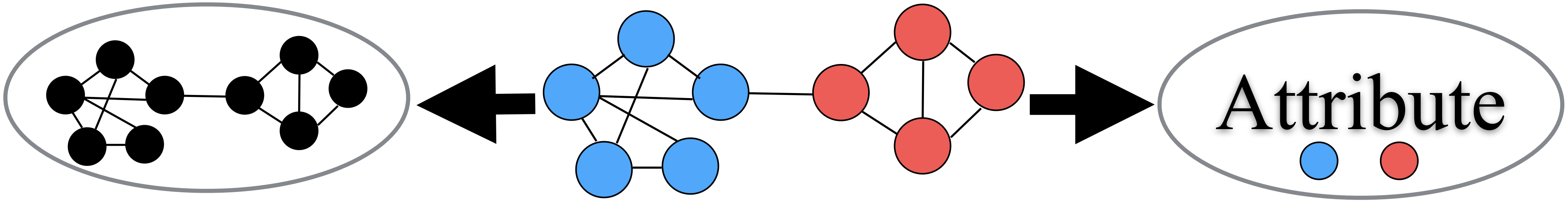


Does not model *autocorrelation* between attributes

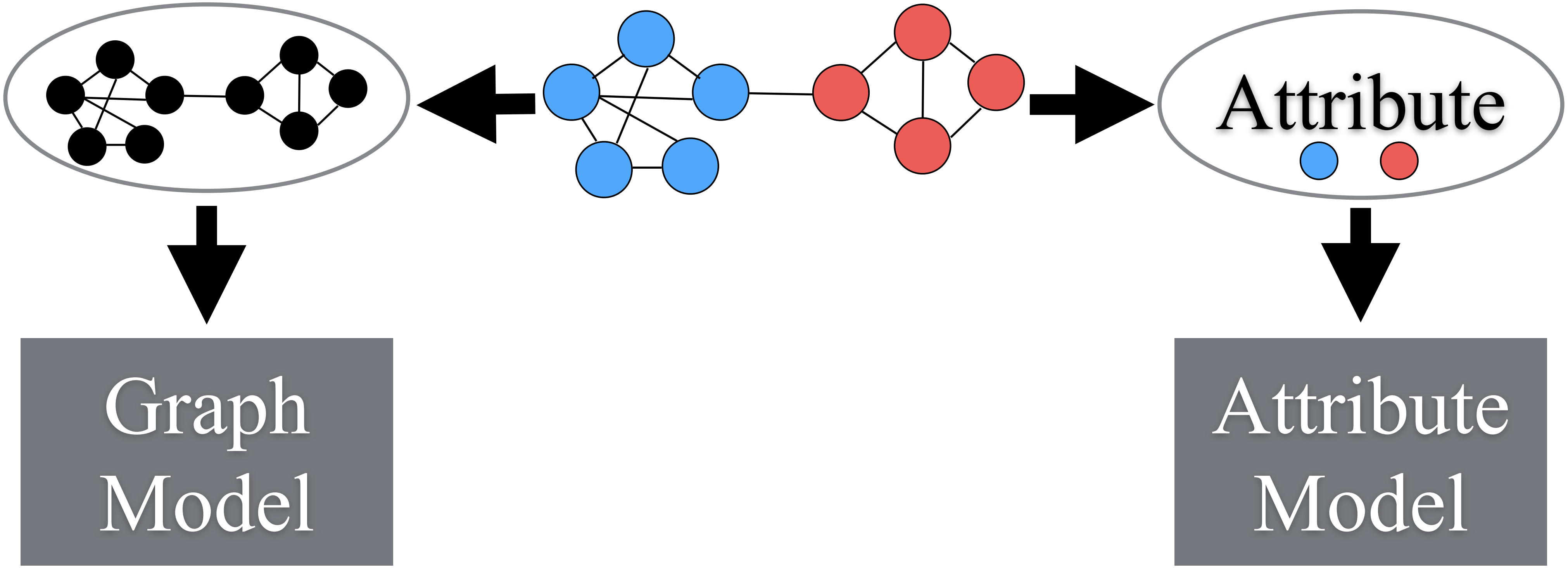
Attributed Graph Models (AGM)



Attributed Graph Models (AGM)



Attributed Graph Models (AGM)

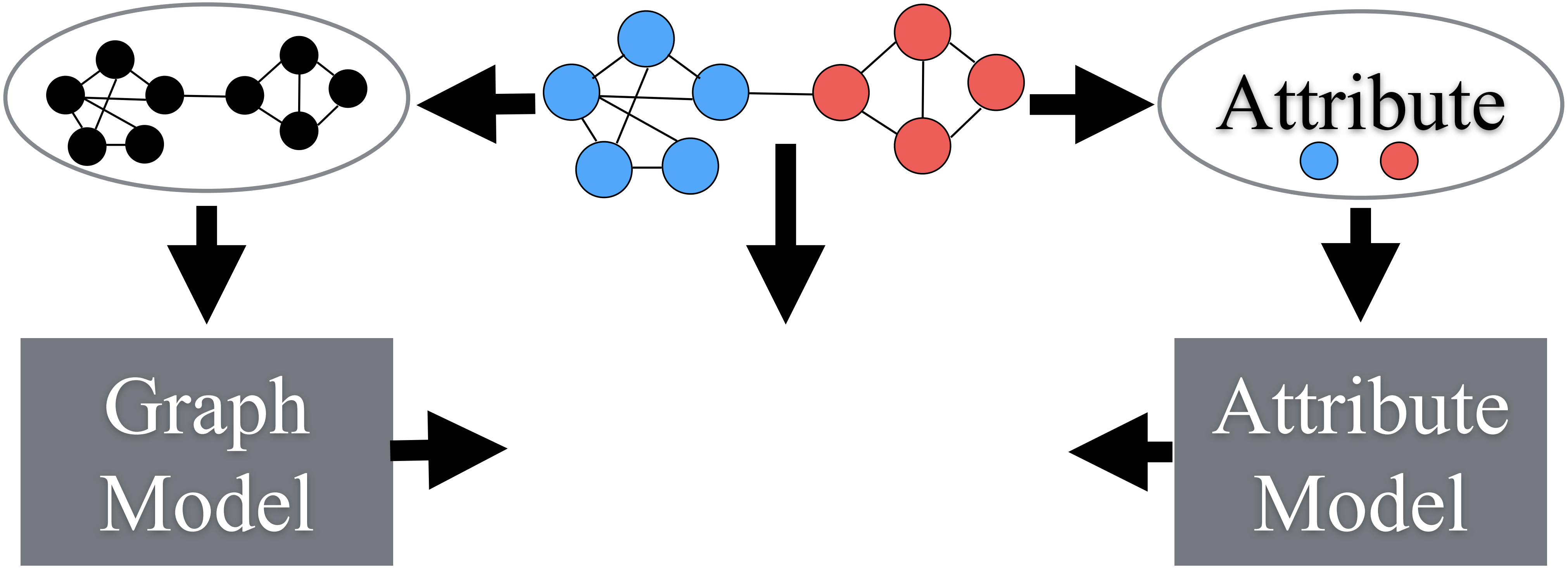


Attribute

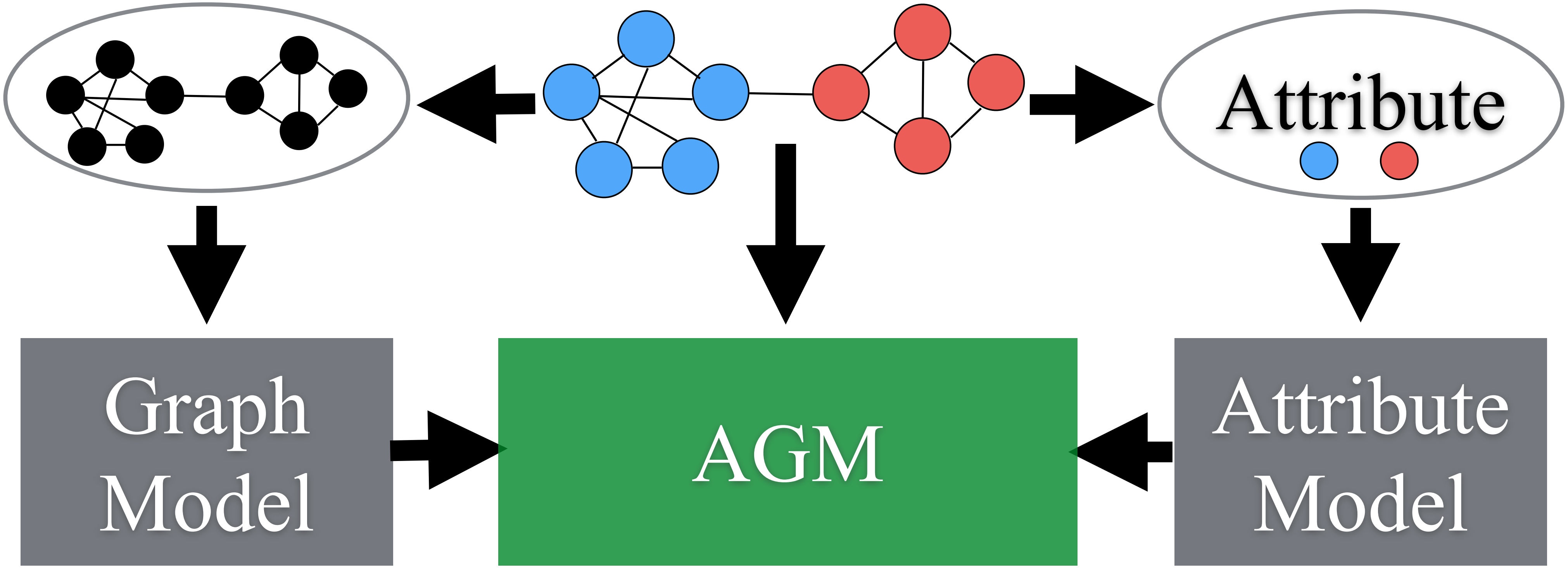
Graph Model

Attribute Model

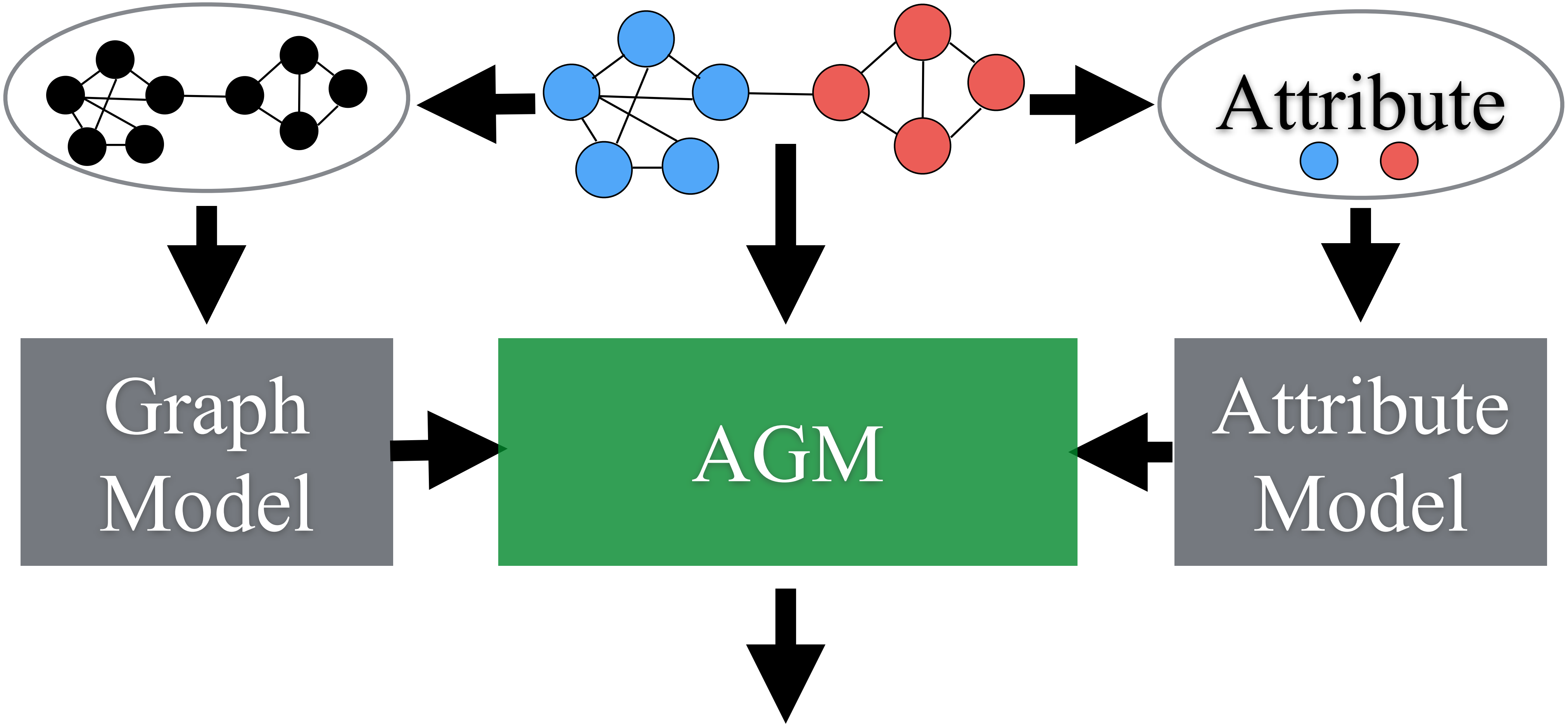
Attributed Graph Models (AGM)



Attributed Graph Models (AGM)



Attributed Graph Models (AGM)



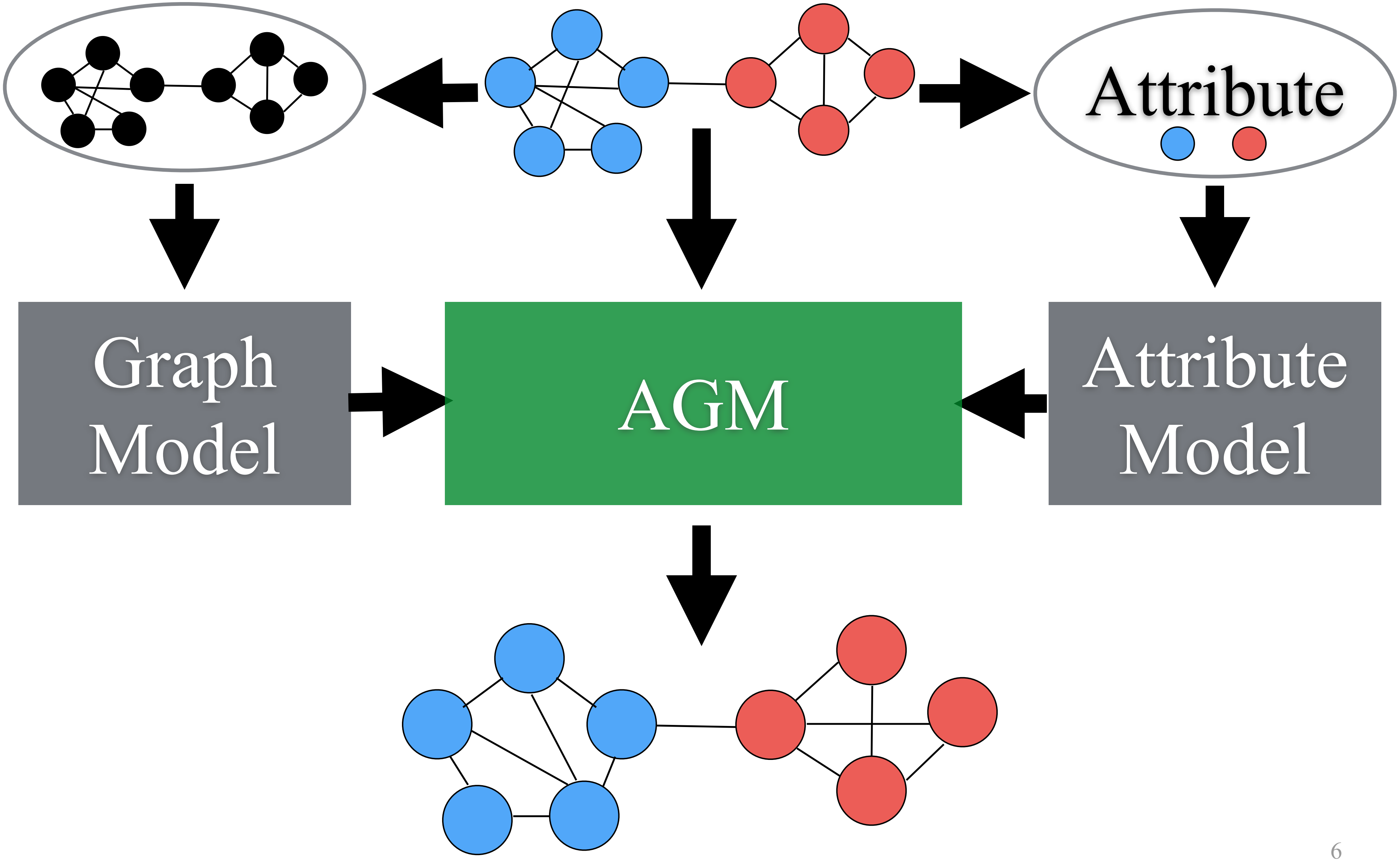
Attribute

Graph Model

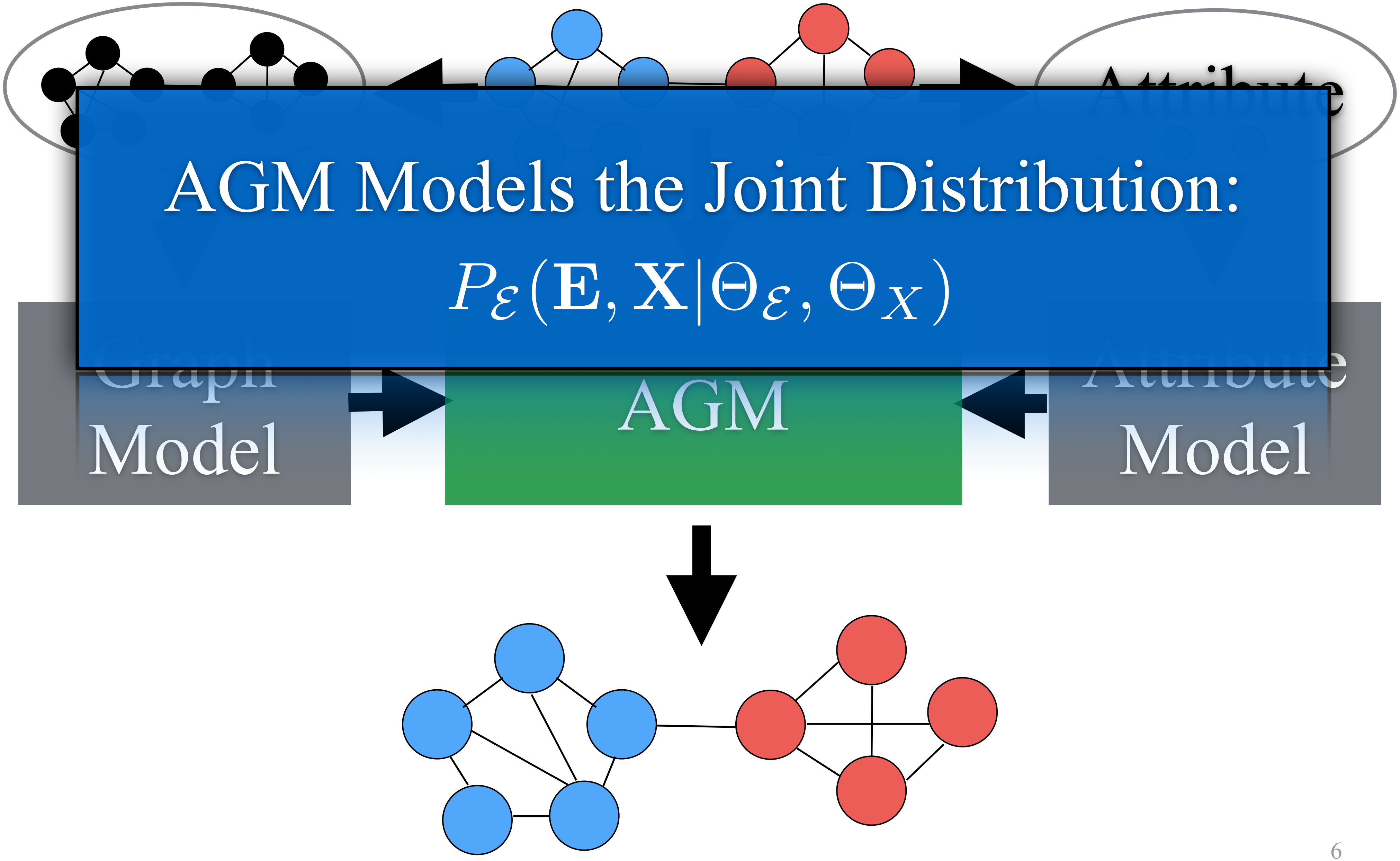
AGM

Attribute Model

Attributed Graph Models (AGM)



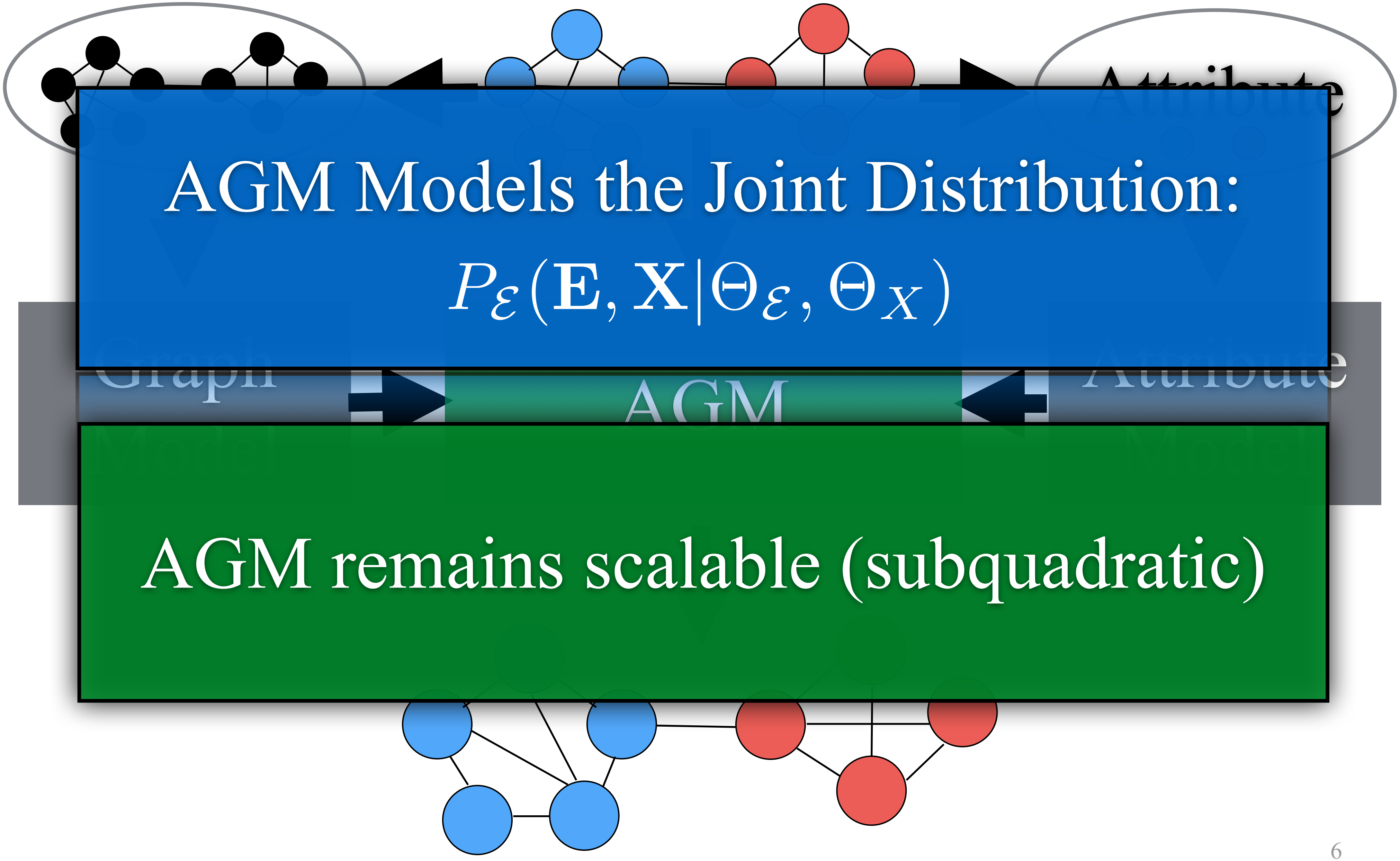
Attributed Graph Models (AGM)



AGM Models the Joint Distribution:

$$P_{\mathcal{E}}(\mathbf{E}, \mathbf{X} | \Theta_{\mathcal{E}}, \Theta_X)$$

Attributed Graph Models (AGM)



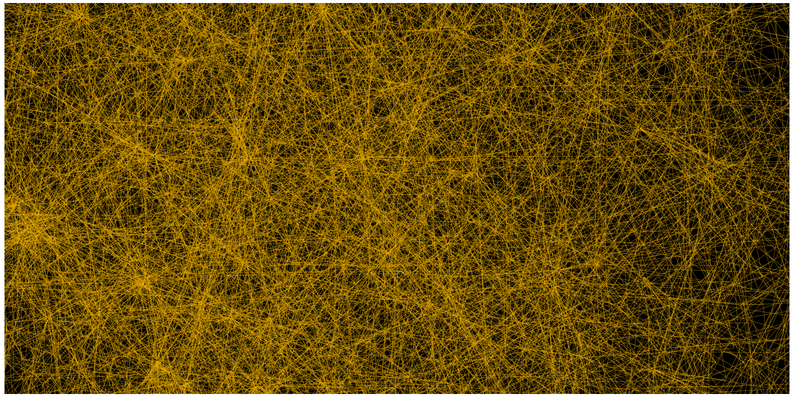
AGM Models the Joint Distribution:

$$P_{\mathcal{E}}(\mathbf{E}, \mathbf{X} | \Theta_{\mathcal{E}}, \Theta_X)$$

AGM remains scalable (subquadratic)

Outline:

- **Background**
- Scalable Graph Sampling
- Attributed Graph Models
  - Sampling
  - Theoretical Results
  - Learning From Data
- Experiments
- Conclusions /
  Future Directions



Background: Scalable Generative Graph Models

- Erdos-Renyi
  [Erdos & Renyi, 1960]
- Chung Lu (FCL)
  [Chung & Lu, 2002]
- Kronecker Product (KPGM)
  [Leskovec et al., 2010]
- Transitive Chung Lu (TCL)
  [Pfeiffer et al., 2012]
- BTER
  [Kolda et al., 2012]

Background: Scalable Generative Graph Models

- Erdos-Renyi
  [Erdos & Renyi, 1960]
- Chung Lu (FCL)
  [Chung & Lu, 2002]
- Kronecker Product (KPGM)
  [Leskovec et al., 2010]
- Transitive Chung Lu (TCL)
  [Pfeiffer et al., 2012]
- BTER
  [Kolda et al., 2012]

Scalable Sampling

Background: Scalable Generative Graph Models

- Erdos-Renyi
  [Erdos & Renyi, 1960]
- Chung Lu (FCL)
  [Chung & Lu, 2002]
- Kronecker Product (KPGM)
  [Leskovec et al., 2010]
- Transitive Chung Lu (TCL)
  [Pfeiffer et al., 2012]
- BTER
  [Kolda et al., 2012]

Scalable Sampling

$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}}) < O(N_v^2)$$

Variable	Definition
N_v	Number Vertices
N_e	Number Edges
$\tau_{\mathcal{E}}$	Construction Cost
$\kappa_{\mathcal{E}}$	Sample Cost

Background: Scalable Generative Graph Models

- Erdos-Renyi
  [Erdos & Renyi, 1960]
- **Chung Lu (FCL)**
  [Chung & Lu, 2002]
- Kronecker Product (KPGM)
  [Leskovec et al., 2010]
- Transitive Chung Lu (TCL)
  [Pfeiffer et al., 2012]
- BTER
  [Kolda et al., 2012]

Scalable Sampling

$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}}) < O(N_v^2)$$

Variable	Definition
N_v	Number Vertices
N_e	Number Edges
$\tau_{\mathcal{E}}$	Construction Cost
$\kappa_{\mathcal{E}}$	Sample Cost

Chung Lu Graph Model

Chung Lu Graph Model

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}

Chung Lu Graph Model

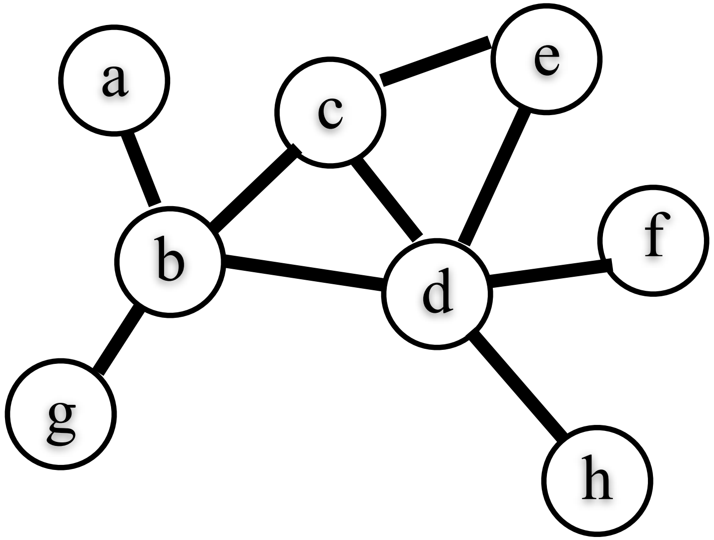
- Produces Networks with
  Given Expected Degree Distribution

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}

Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}

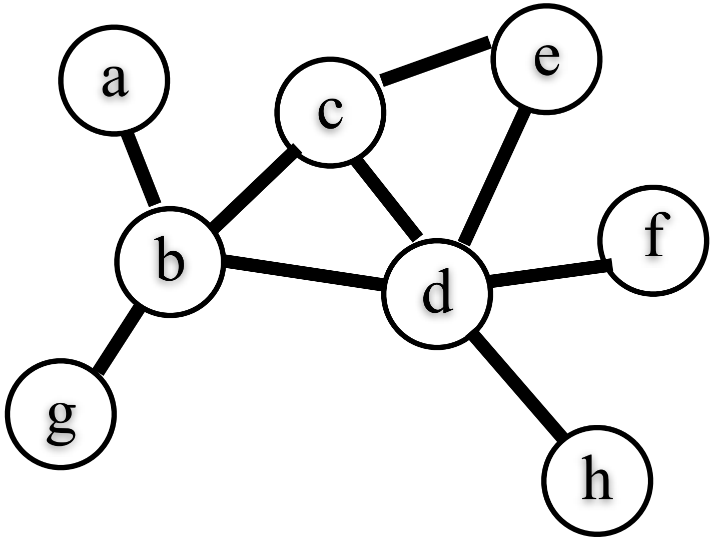


Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



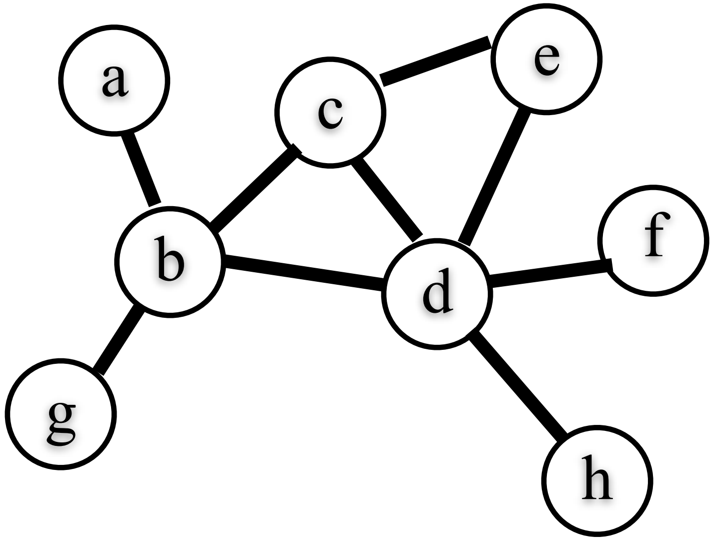
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



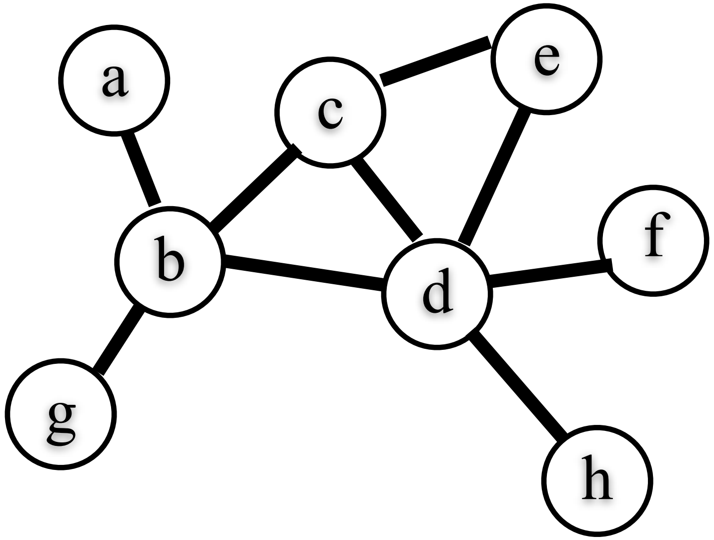
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}

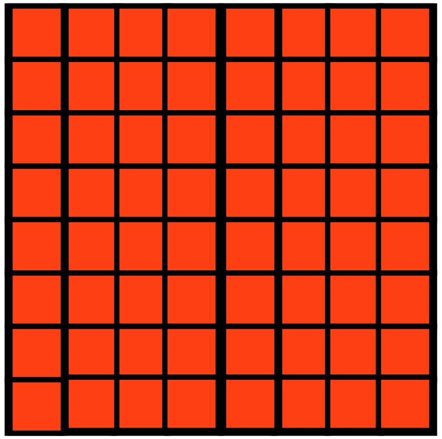


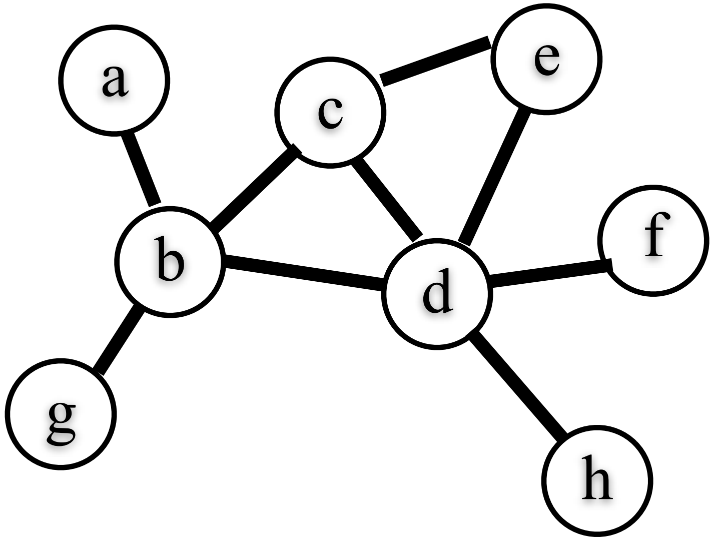
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$





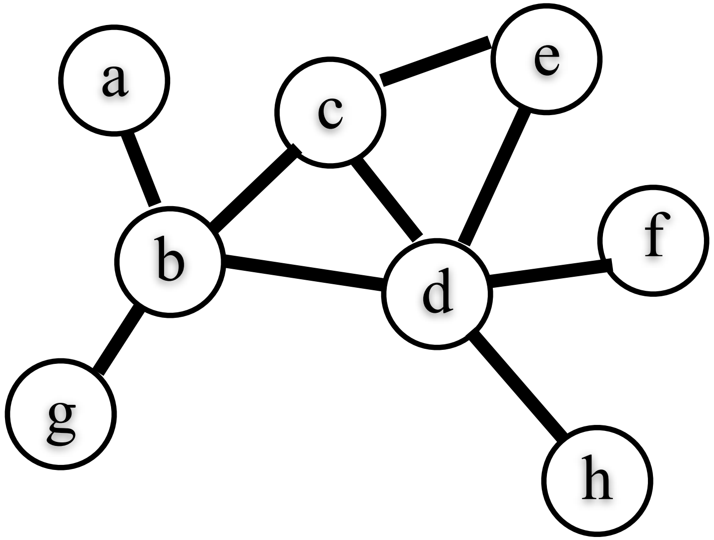
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



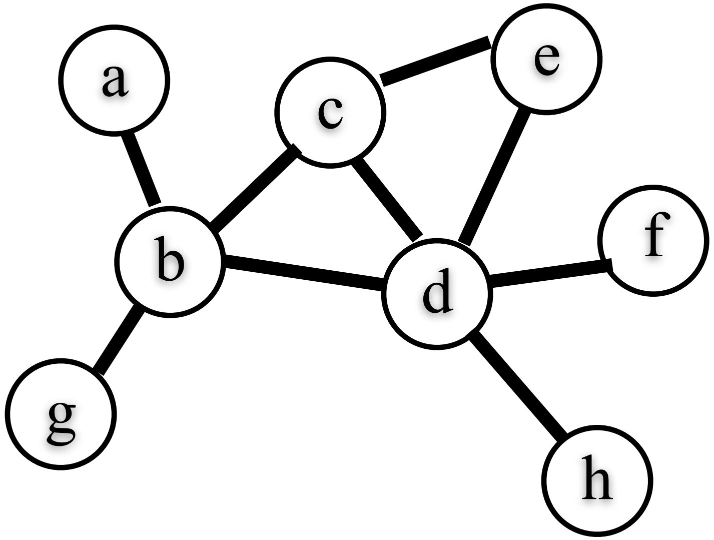
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



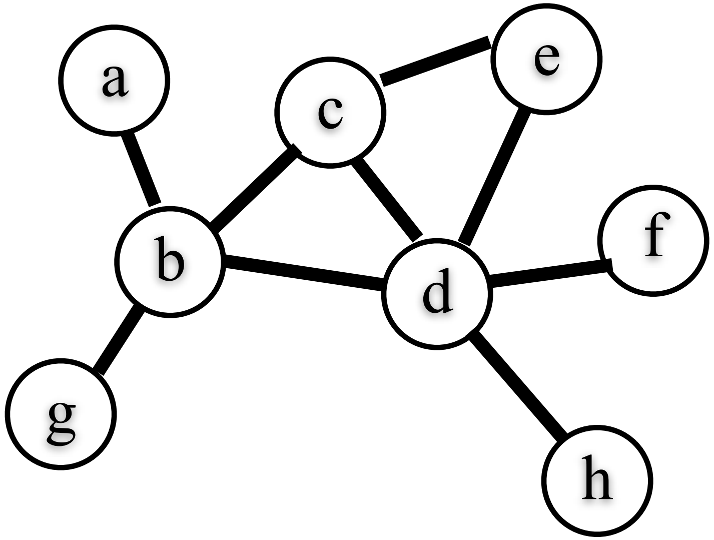
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



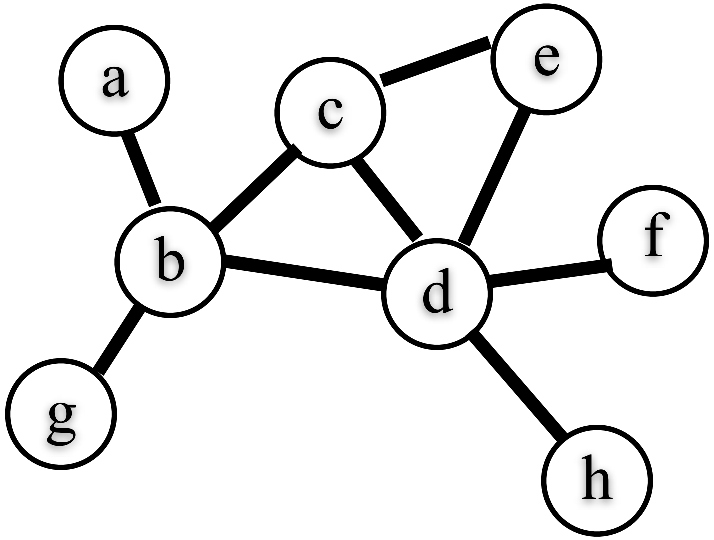
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



a	b	b	b	b	c	c	c	...

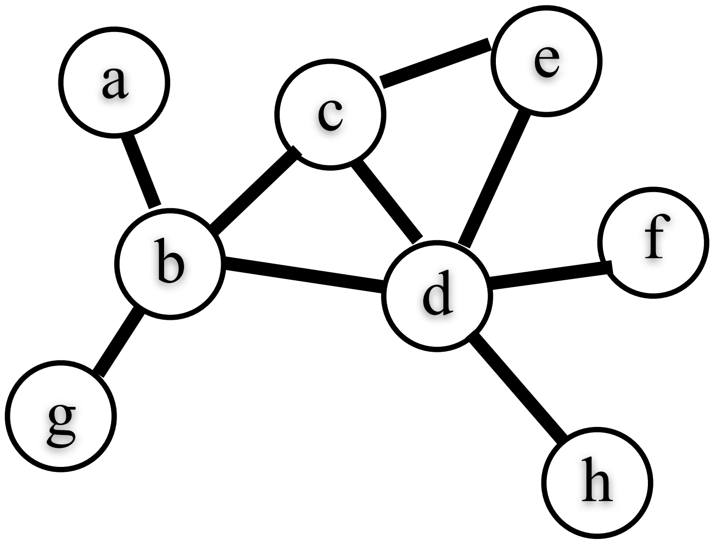
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P((v_i, v_j) \in \mathbf{E}) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - “Edge-by-Edge”

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



a	b	b	b	b	c	c	c

...

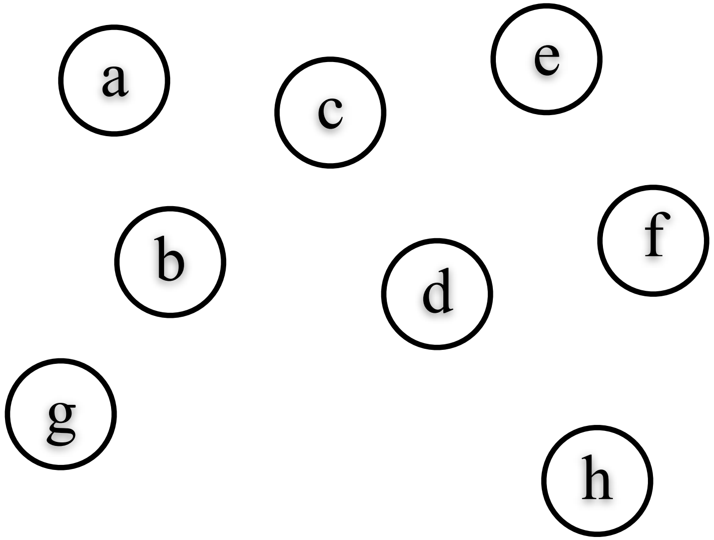
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - "Edge-by-Edge"

p_{11}	p_{12}	…	…	…	…	p_{17}	p_{18}
p_{21}	p_{22}	…	…	…	…	p_{27}	p_{28}
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
p_{71}	p_{72}	…	…	…	…	p_{77}	p_{78}
p_{81}	p_{82}	…	…	…	…	p_{87}	p_{88}



a	b	b	b	b	c	c	c	…

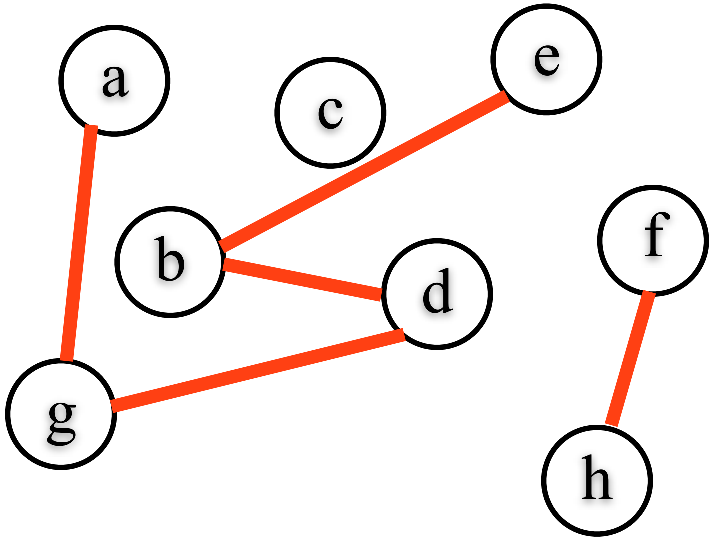
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - “Edge-by-Edge”

p11	p12	...	...	...	...	p17	p18
p21	p22	...	...	...	...	p27	p28
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p71	p72	...	...	...	...	p77	p78
p81	p82	...	...	...	...	p87	p88



a	b	b	b	b	c	c	c	...

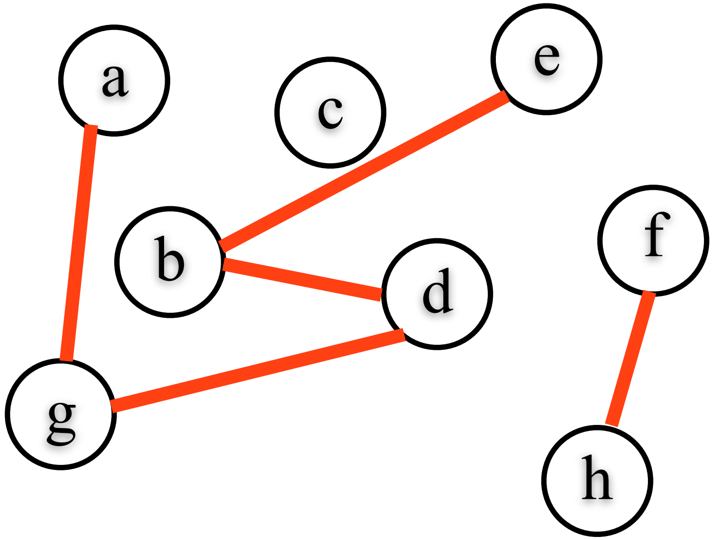
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - “Edge-by-Edge”
  - Draw twice from degree distribution and place an edge

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



a	b	b	b	b	c	c	c	...

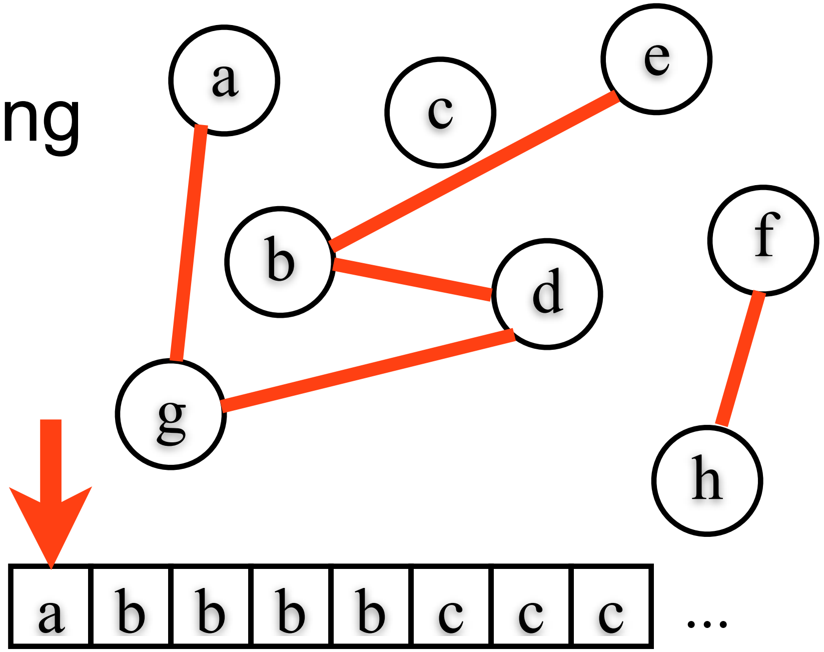
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - "Edge-by-Edge"
  - Draw twice from degree distribution and place an edge

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



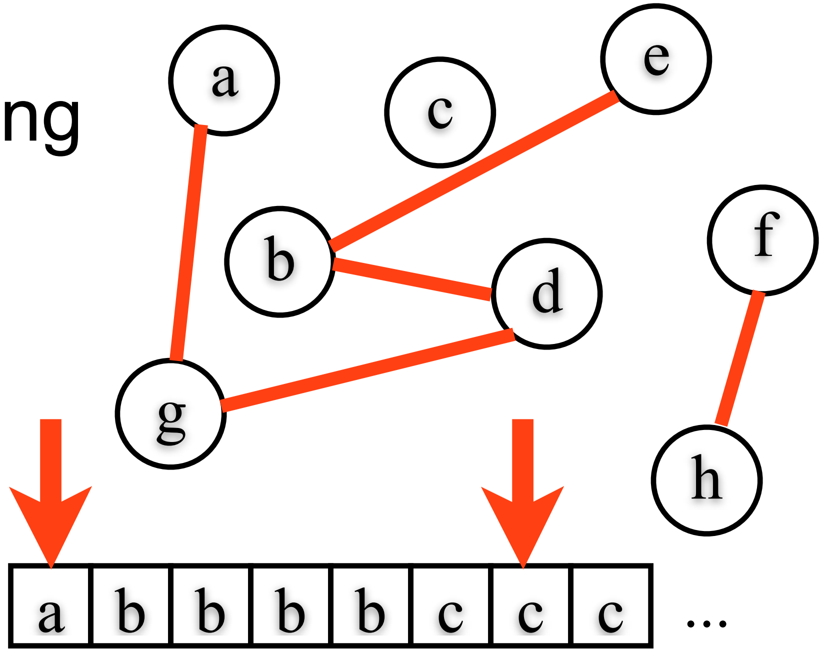
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - “Edge-by-Edge”
  - Draw twice from degree distribution and place an edge

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



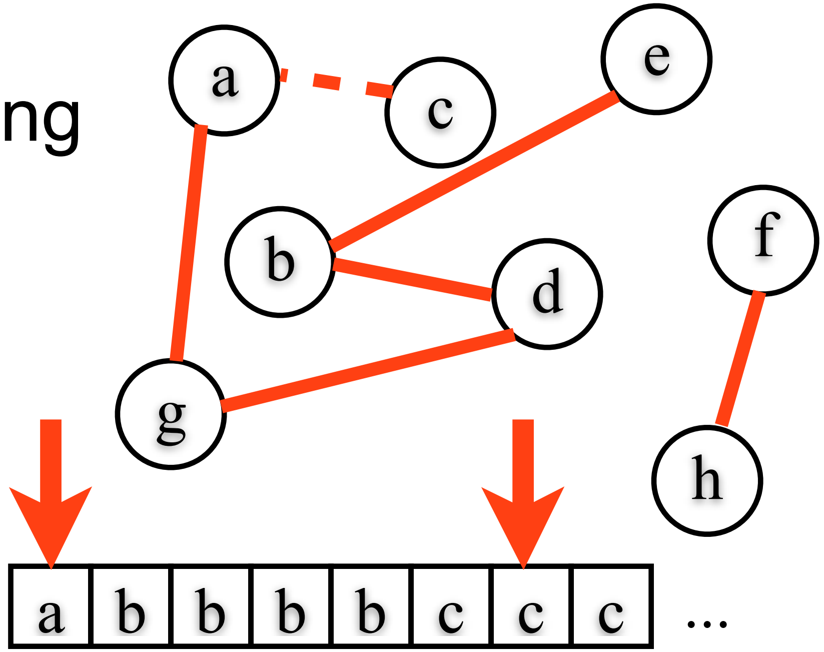
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - “Edge-by-Edge”
  - Draw twice from degree distribution and place an edge

p_{11}	p_{12}	...	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}

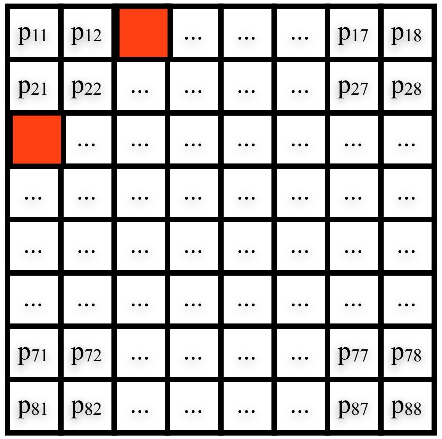


Chung Lu Graph Model

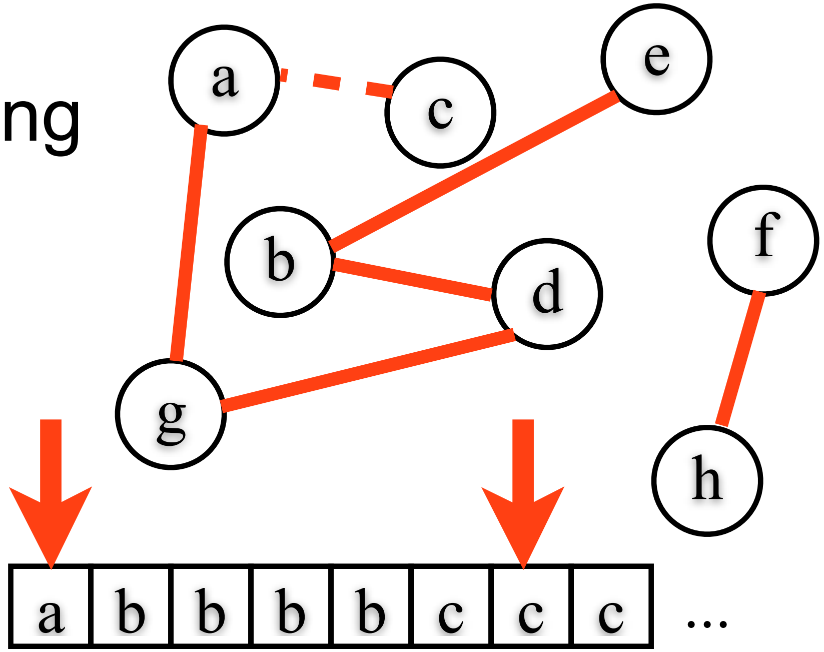
- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - “Edge-by-Edge”
  - Draw twice from degree distribution and place an edge



p_{11}	p_{12}	🟥	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
🟥	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}

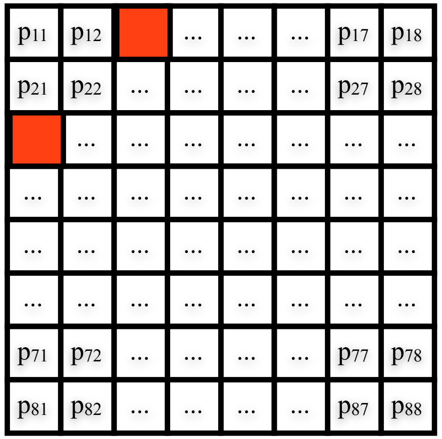


Chung Lu Graph Model

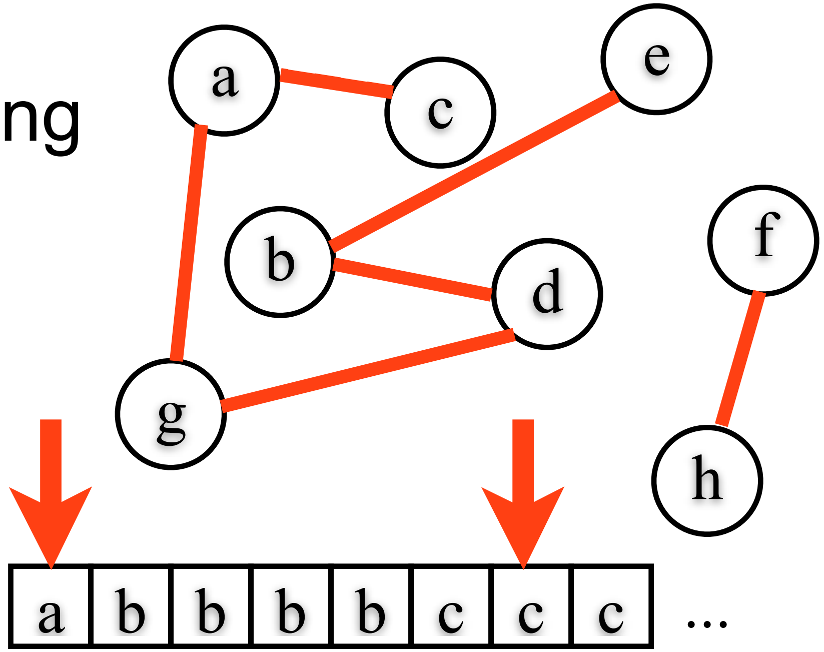
- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - "Edge-by-Edge"
  - Draw twice from degree distribution and place an edge



p_{11}	p_{12}	🟥	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
🟥	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



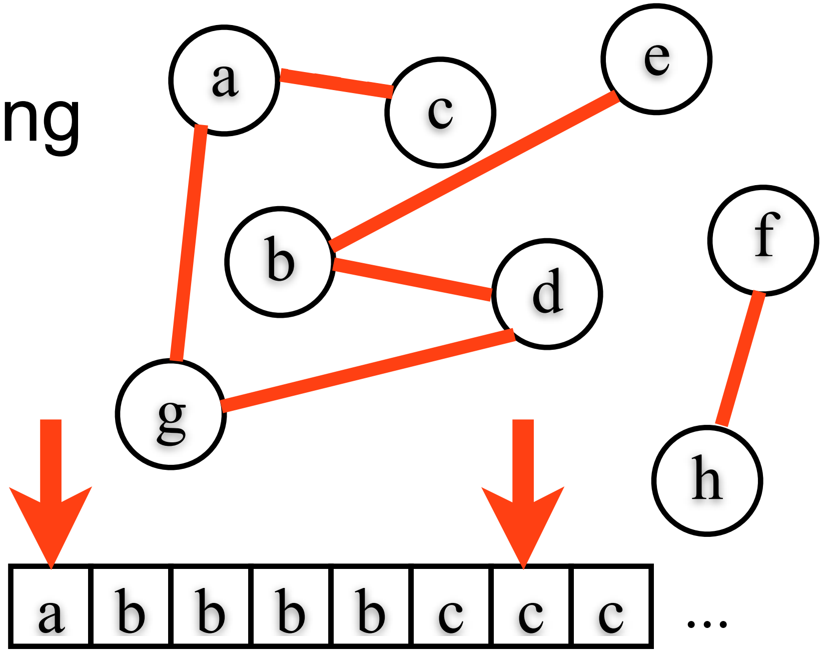
Chung Lu Graph Model

- Produces Networks with Given Expected Degree Distribution
- Edges drawn with probability

$$P\left((v_i, v_j) \in \mathbf{E}\right) = \frac{\theta_{d_i}\theta_{d_j}}{\sum_k \theta_{d_k}} = \frac{\theta_{d_i}\theta_{d_j}}{2N_e}$$

- Expected degree for v_i is θ_{d_i}
- Naive Sampling $O(N_v^2)$
- Structural assumption for sampling
  - Construct Degree Distribution
  - “Edge-by-Edge”
  - Draw twice from degree distribution and place an edge
- Scalable: $O(N_e)$

p_{11}	p_{12}	🟥	...	...	...	p_{17}	p_{18}
p_{21}	p_{22}	...	...	...	...	p_{27}	p_{28}
🟥	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
p_{71}	p_{72}	...	...	...	...	p_{77}	p_{78}
p_{81}	p_{82}	...	...	...	...	p_{87}	p_{88}



Background: Scalable Generative Graph Models

- Erdos-Renyi
  [Erdos & Renyi, 1960]
- Chung Lu (FCL)
  [Chung & Lu, 2002]
- Kronecker Product (KPGM)
  [Leskovec et al., 2010]
- Transitive Chung Lu (TCL)
  [Pfeiffer et al., 2012]
- BTER
  [Kolda et al., 2012]

Scalable Sampling

$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}}) < O(N_v^2)$$

Variable	Definition
N_v	Number Vertices
N_e	Number Edges
$\tau_{\mathcal{E}}$	Construction Cost
$\kappa_{\mathcal{E}}$	Sample Cost

Background: Scalable Generative Graph Models

- Erdos-Renyi
  [Erdos & Renyi, 1960]
- Chung Lu (FCL)
  [Chung & Lu, 2002]
- Kronecker Product (KPGM)
  [Leskovec et al., 2010]
- Transitive Chung Lu (TCL)
  [Pfeiffer et al., 2012]
- BTER
  [Kolda et al., 2012]

Scalable Sampling

$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}}) < O(N_v^2)$$

Scalable Learning

Variable	Definition
N_v	Number Vertices
N_e	Number Edges
$\tau_{\mathcal{E}}$	Construction Cost
$\kappa_{\mathcal{E}}$	Sample Cost

Background: Scalable Generative Graph Models

- Erdos-Renyi
  [Erdos & Renyi, 1960]
- Chung Lu (FCL)
  [Chung & Lu, 2002]
- Kronecker Product (KPGM)
  [Leskovec et al., 2010]
- Transitive Chung Lu (TCL)
  [Pfeiffer et al., 2012]
- BTER
  [Kolda et al., 2012]

🟩 Scalable Sampling

$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}}) < O(N_v^2)$$

🟩 Scalable Learning

🟥 No Attributes

Variable	Definition
N_v	Number Vertices
N_e	Number Edges
$\tau_{\mathcal{E}}$	Construction Cost
$\kappa_{\mathcal{E}}$	Sample Cost

Background: Scalable Generative Graph Models

- Erdos-Renyi
  [Erdos & Renyi, 1960]
- Chung Lu (FCL)
  [Chung & Lu, 2002]
- Kronecker Product (KPGM)
  [Leskovec et al., 2010]
- [illegible]
  [Pfeiffer [illegible]
- BTER
  [Kolda et al., 2012]

Scalable Sampling

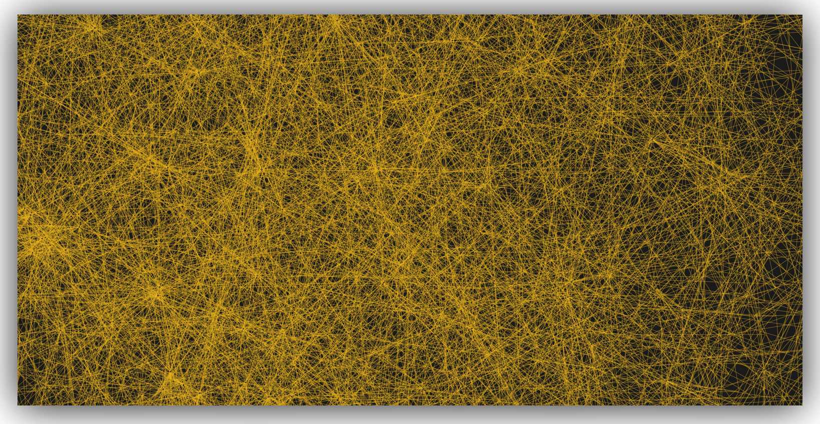
$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}}) < O(N_v^2)$$

AGM incorporates attributes and retains subquadratic learning and sampling

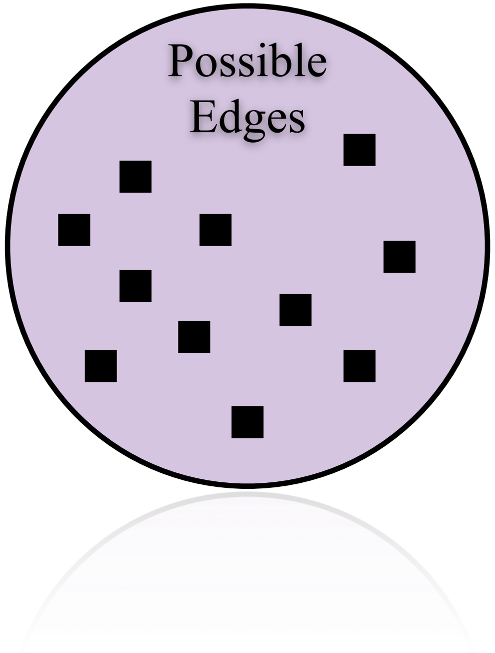
Variable	Definition
N_v	Number Vertices
N_e	Number Edges
$\tau_{\mathcal{E}}$	Construction Cost
$\kappa_{\mathcal{E}}$	Sample Cost

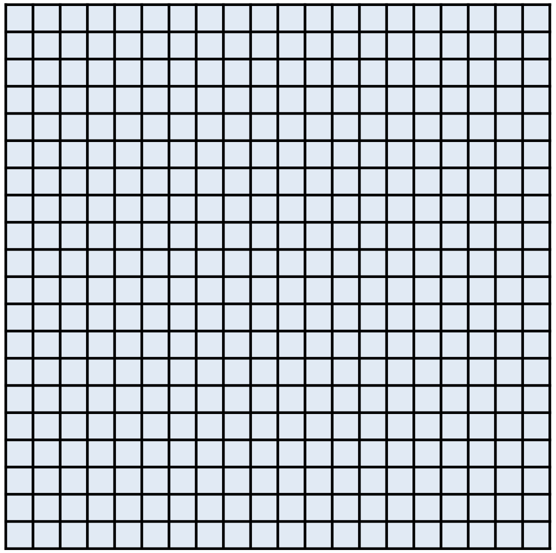
Outline:

- Background
- **Scalable Graph Sampling**
- Attributed Graph Models
    - Sampling
    - Theoretical Results
    - Learning From Data
- Experiments
- Conclusions /
  Future Directions

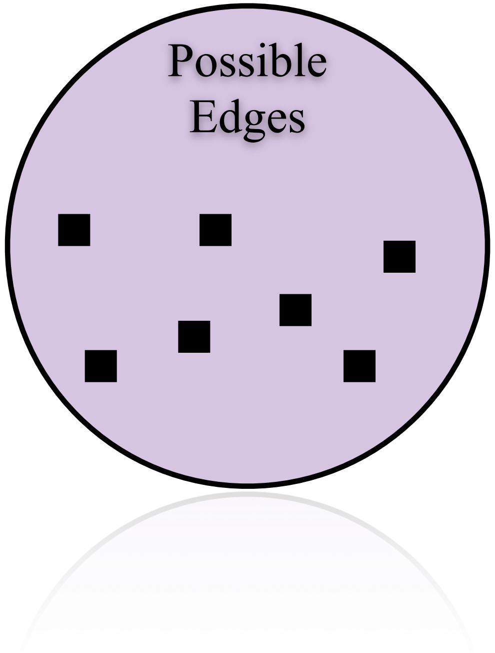


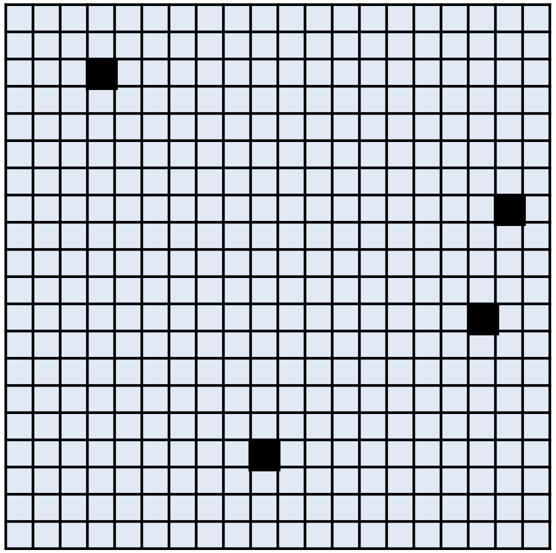
Scalable sampling in practice





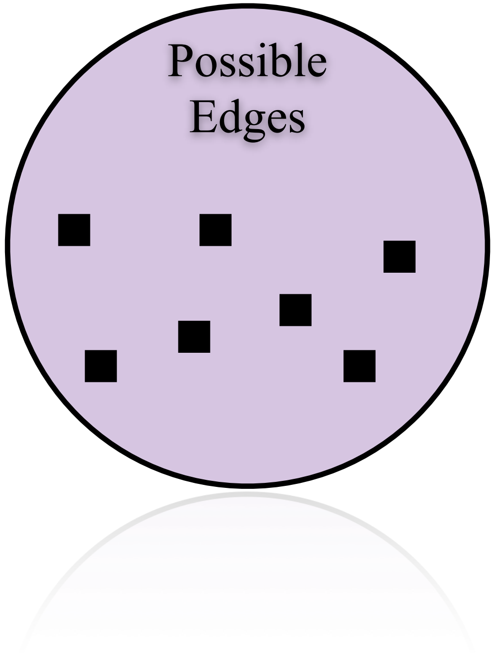
Scalable sampling in practice

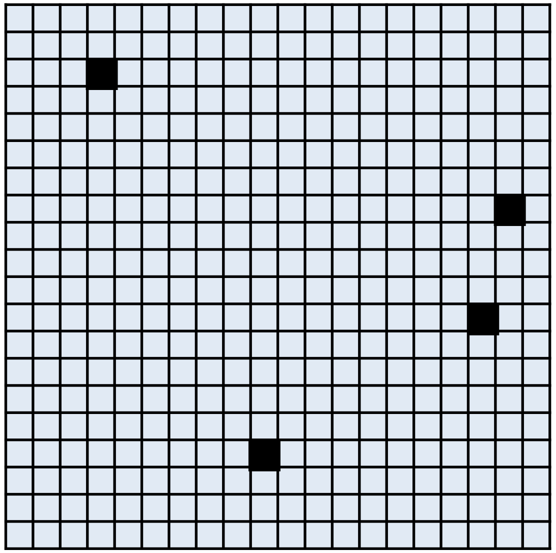




Scalable sampling in practice

```
while not enough edges:
   draw (vi,vj) from Q’ (the model)

   put (vi, vj) into the edges

return edges
```

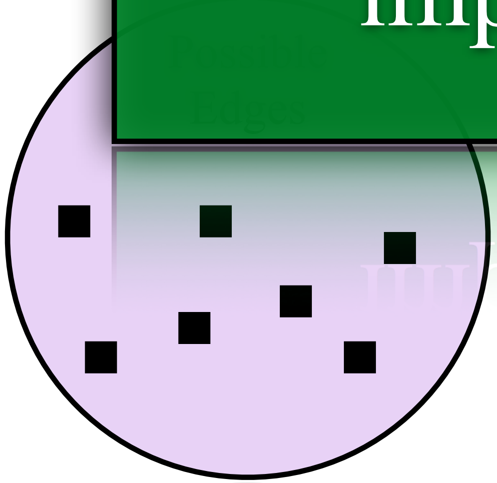


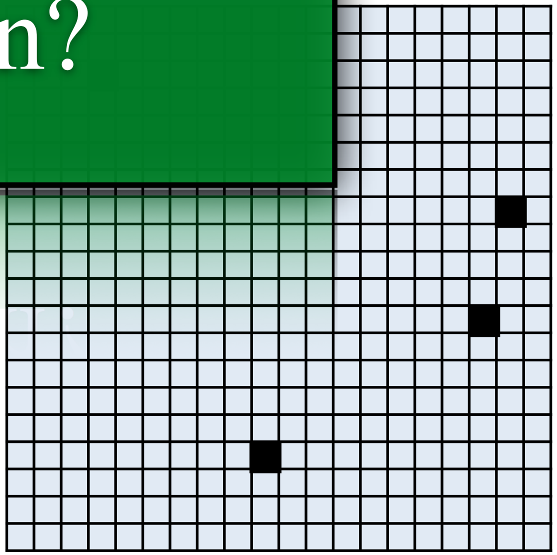


Scalable sampling in practice

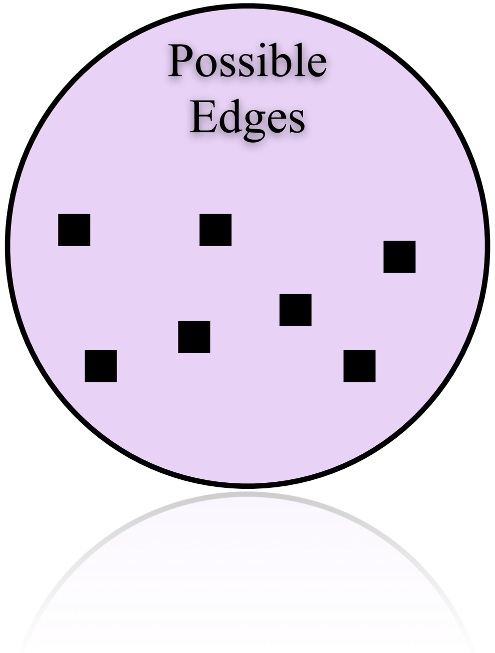
```
while not enough edges:
   draw (vi,vj) from Q’ (the model)

   put (vi, vj) into the edges

return edges
```

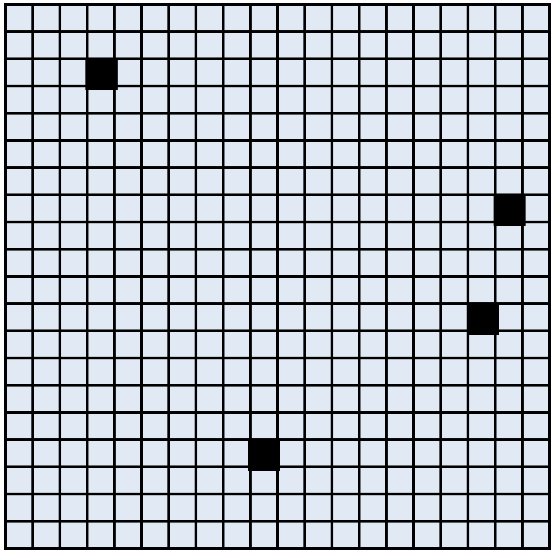
Close, but how do we actually implement this function?



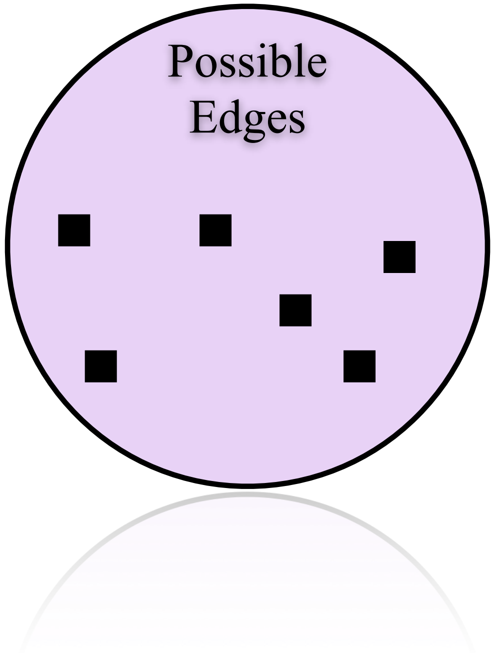


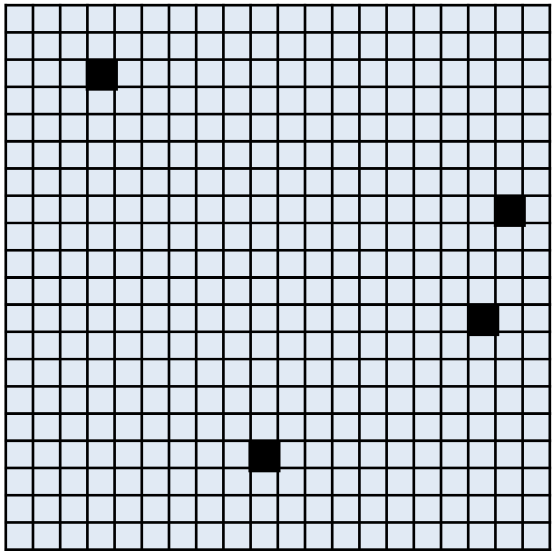
Scalable sampling in practice





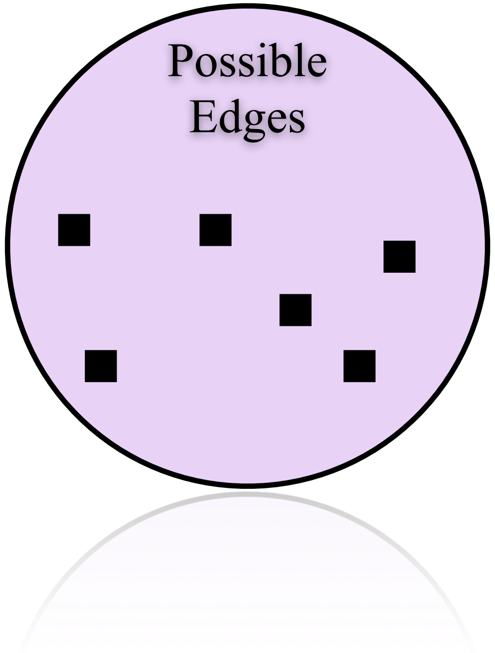
Scalable sampling in practice



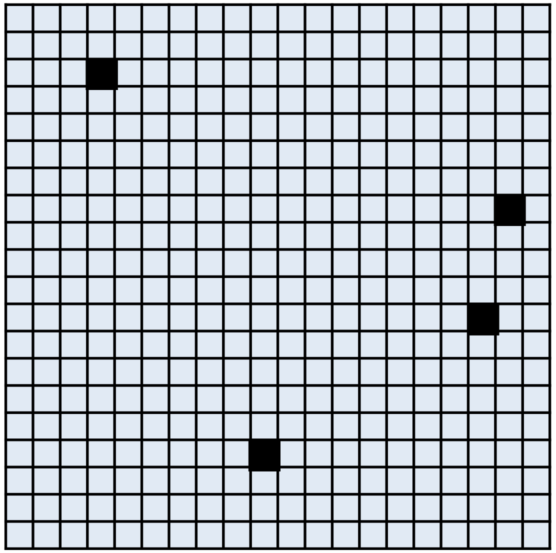


Scalable sampling in practice

```
while not enough edges:
   draw (vi,vj) from Q' (the model)

   if (vi, vj) not in edges
      put (vi, vj) into the edges

return edges
```



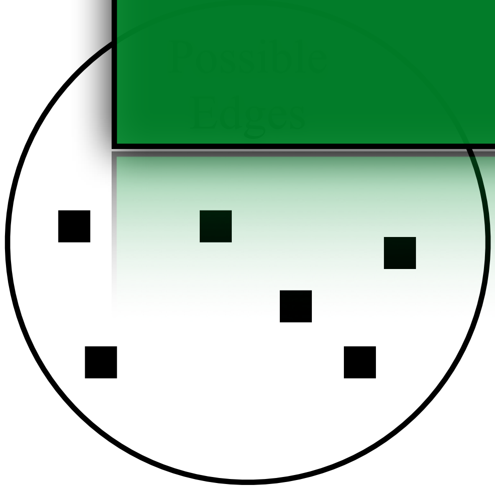
Possible Edges

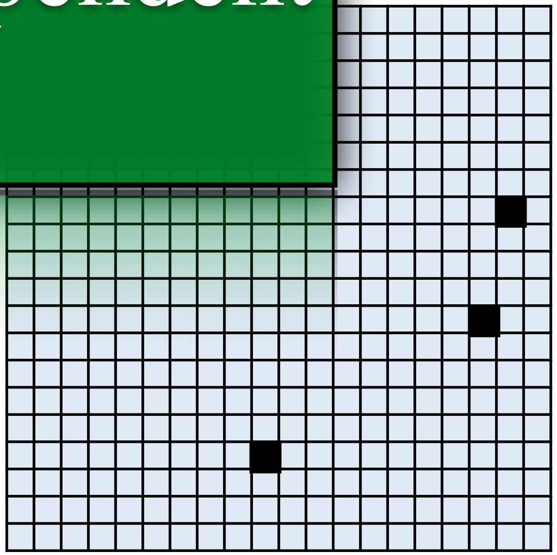


Scalable sampling in practice

```
while not enough edges:
   draw (vi,vj) from Q’ (the model)

   if (vi, vj) not in edges
      put (vi, vj) into the edges


return edges
```

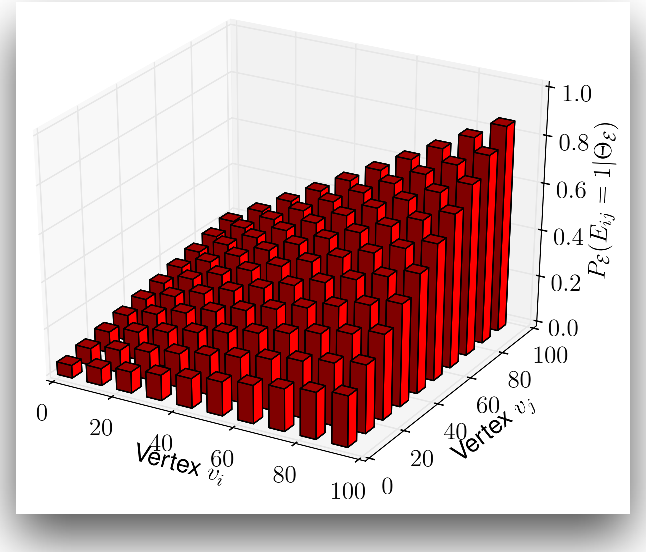
Draws are not actually independent





Examining Scalable Sampling

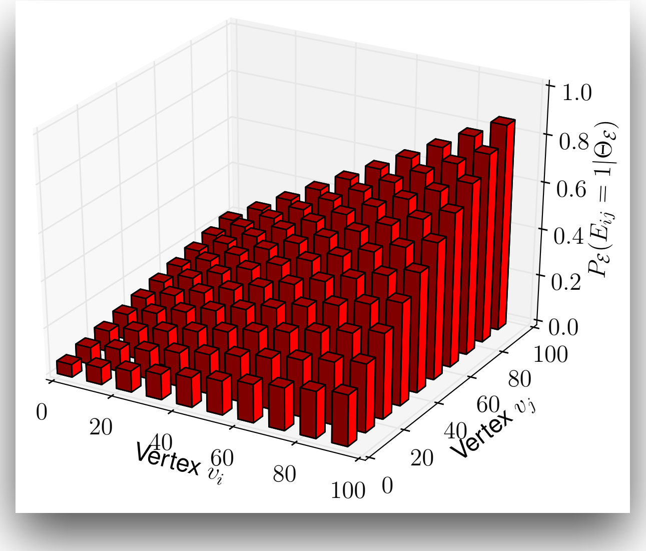
Examining Scalable Sampling



Examining Scalable Sampling

- Scalable sampling algorithms repeatedly sample from a multinomial parameterized by:

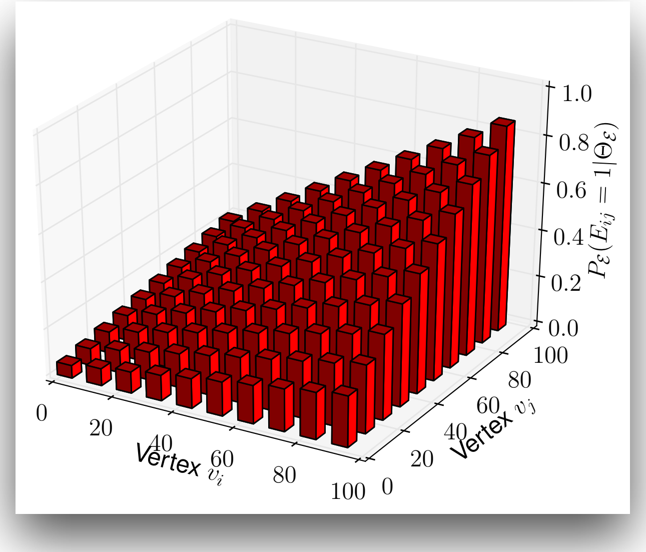
$$Q'_{\mathcal{E}}(i,j) = \frac{P_{\mathcal{E}}\Big((v_i, v_j) \in \mathbf{E}\Big)}{\sum_{k,l} P_{\mathcal{E}}\Big((v_k, v_l) \in \mathbf{E}\Big)}$$

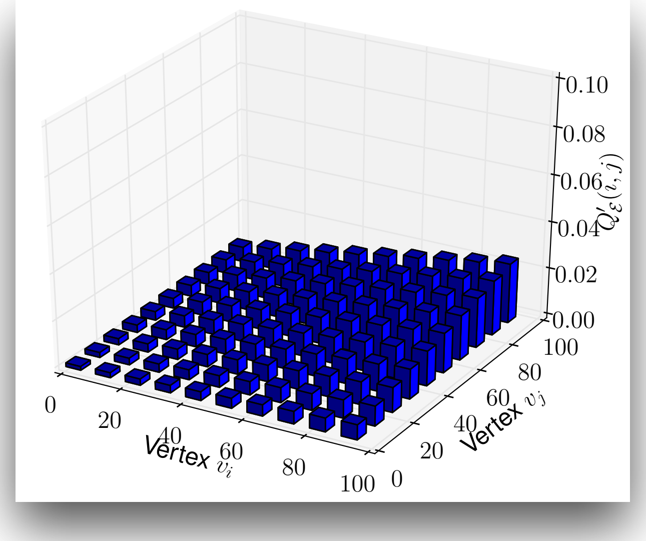


Examining Scalable Sampling

- Scalable sampling algorithms repeatedly sample from a multinomial parameterized by:

$$Q'_{\mathcal{E}}(i,j) = \frac{P_{\mathcal{E}}\Big((v_i, v_j) \in \mathbf{E}\Big)}{\sum_{k,l} P_{\mathcal{E}}\Big((v_k, v_l) \in \mathbf{E}\Big)}$$



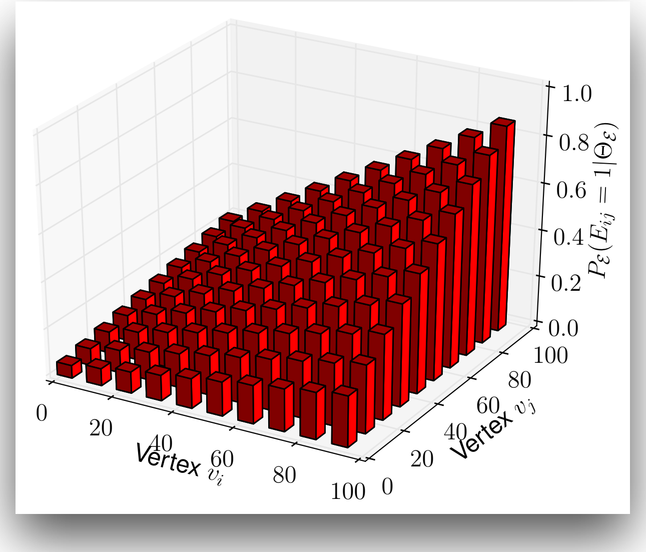


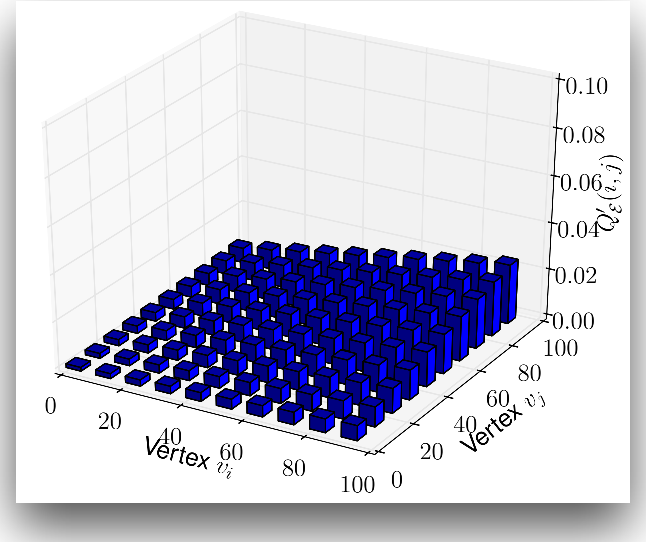
Examining Scalable Sampling

- Scalable sampling algorithms repeatedly sample from a multinomial parameterized by:

$$Q'_{\mathcal{E}}(i,j) = \frac{P_{\mathcal{E}}\Big((v_i, v_j) \in \mathbf{E}\Big)}{\sum_{k,l} P_{\mathcal{E}}\Big((v_k, v_l) \in \mathbf{E}\Big)}$$

- Applies to Chung Lu and Kronecker Product Models
  - Neither explicitly constructs matrix





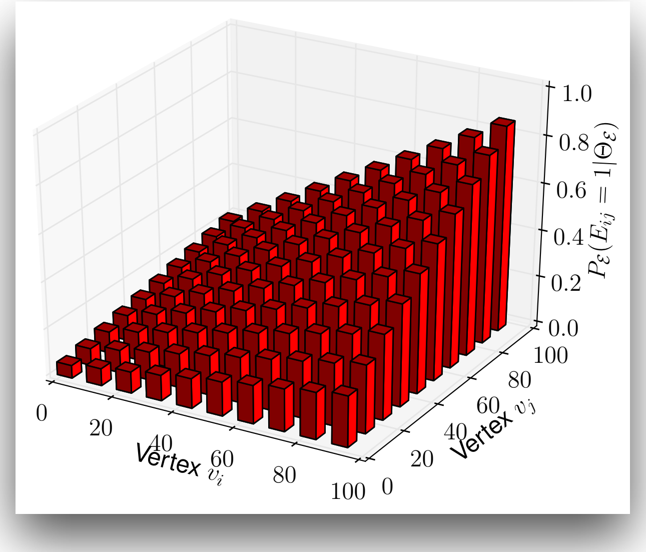
Examining Scalable Sampling

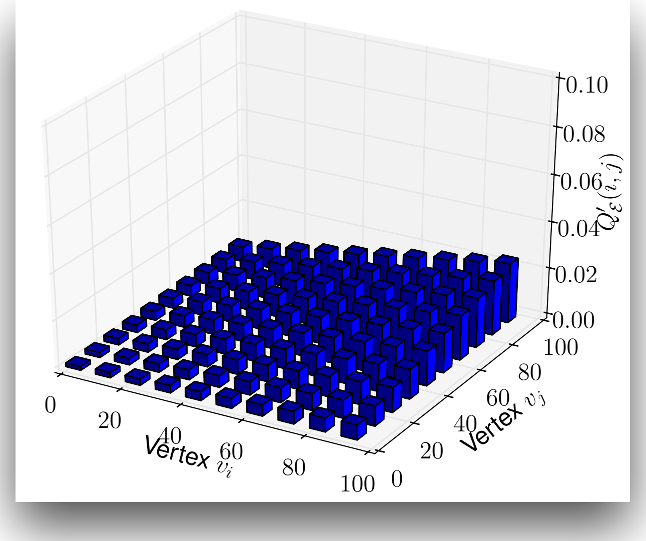
- Scalable sampling algorithms repeatedly sample from a multinomial parameterized by:

$$Q'_{\mathcal{E}}(i,j) = \frac{P_{\mathcal{E}}\Big(\big(v_i, v_j\big) \in \mathbf{E}\Big)}{\sum_{k,l} P_{\mathcal{E}}\Big(\big(v_k, v_l\big) \in \mathbf{E}\Big)}$$

- Applies to Chung Lu and Kronecker Product Models
  - Neither explicitly constructs matrix

$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}}) < O(N_v^2)$$



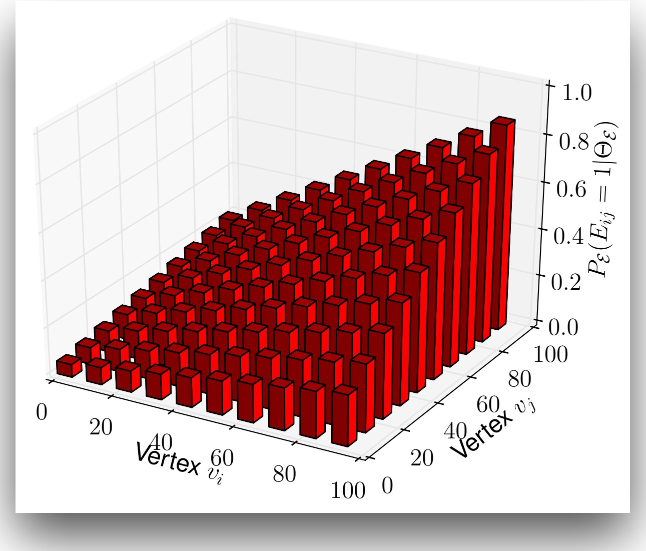


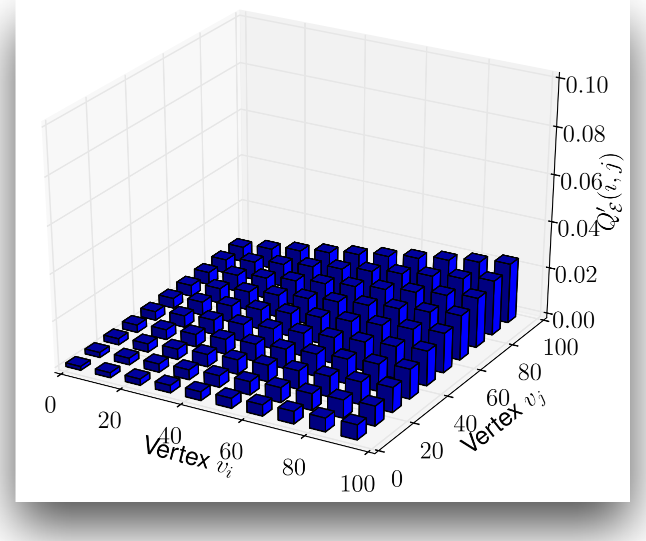
Examining Scalable Sampling

- Scalable sampling algorithms repeatedly sample from a multinomial parameterized by:

$$Q'_{\mathcal{E}}(i,j) = \frac{P_{\mathcal{E}}\Big((v_i, v_j) \in \mathbf{E}\Big)}{\sum_{k,l} P_{\mathcal{E}}\Big((v_k, v_l) \in \mathbf{E}\Big)}$$

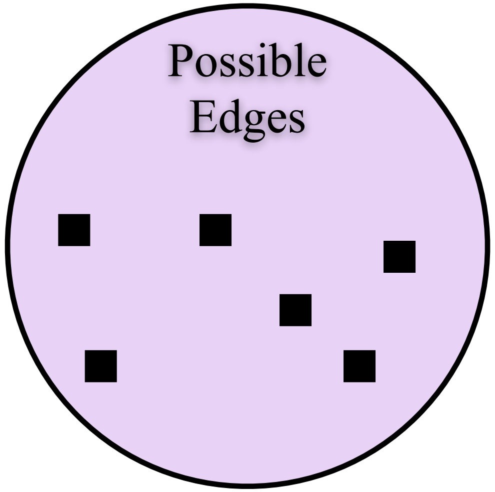
- Applies to Chung Lu and Kronecker Product Models
  - Neither explicitly constructs matrix
    $O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}}) < O(N_v^2)$
- Scalable approximation of true distribution
  - *Better* on *larger* networks

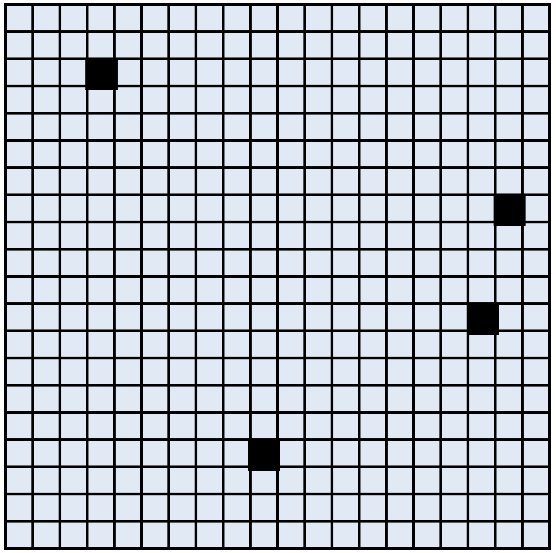




Generalizing and Exploiting

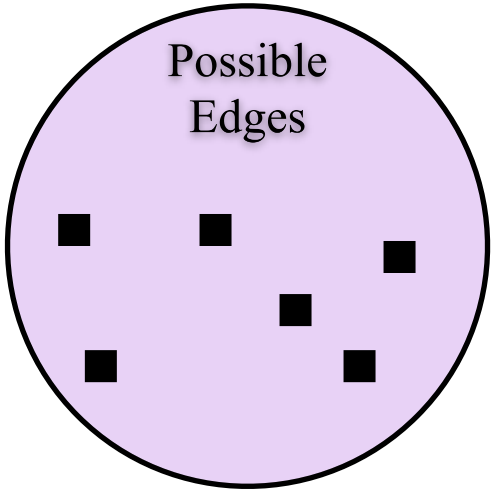
```
while not enough edges:
  draw (vi,vj) from Q’ (the model)

  if (vi, vj) not in edges
    put (vi, vj) into the edges

return edges
```

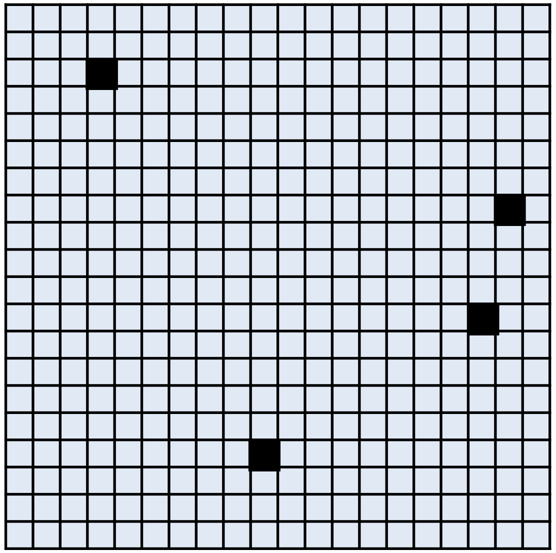




Generalizing and Exploiting

```
while not enough edges:
  draw (vi,vj) from Q' (the model)

  if (vi, vj) not in edges
    put (vi, vj) into the edges

return edges
```

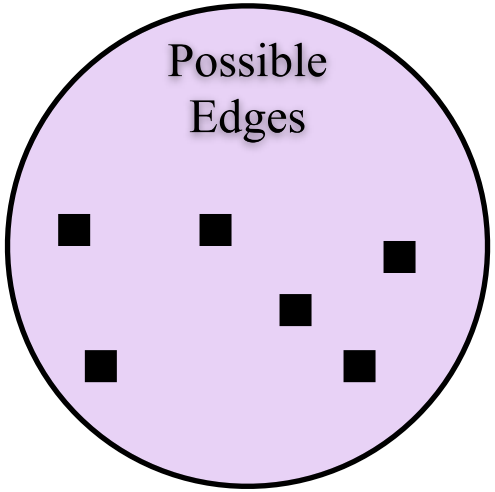


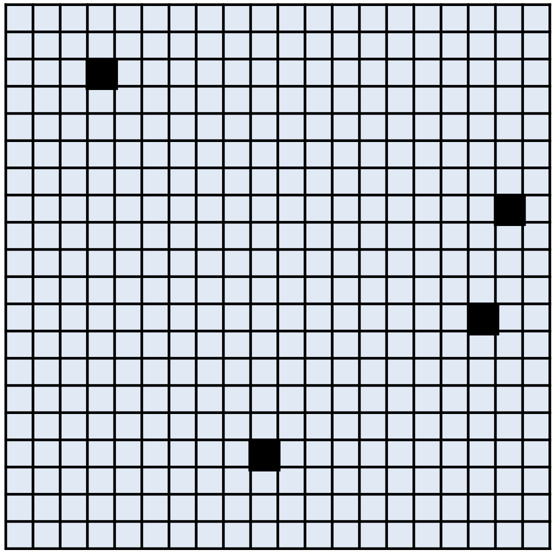


Generalizing and Exploiting

```
while not enough edges:
  draw (vi,vj) from Q’ (the model)

  if (vi, vj) not in edges
    put (vi, vj) into the edges

return edges
```

- Filter: *Rejects* duplicate edges

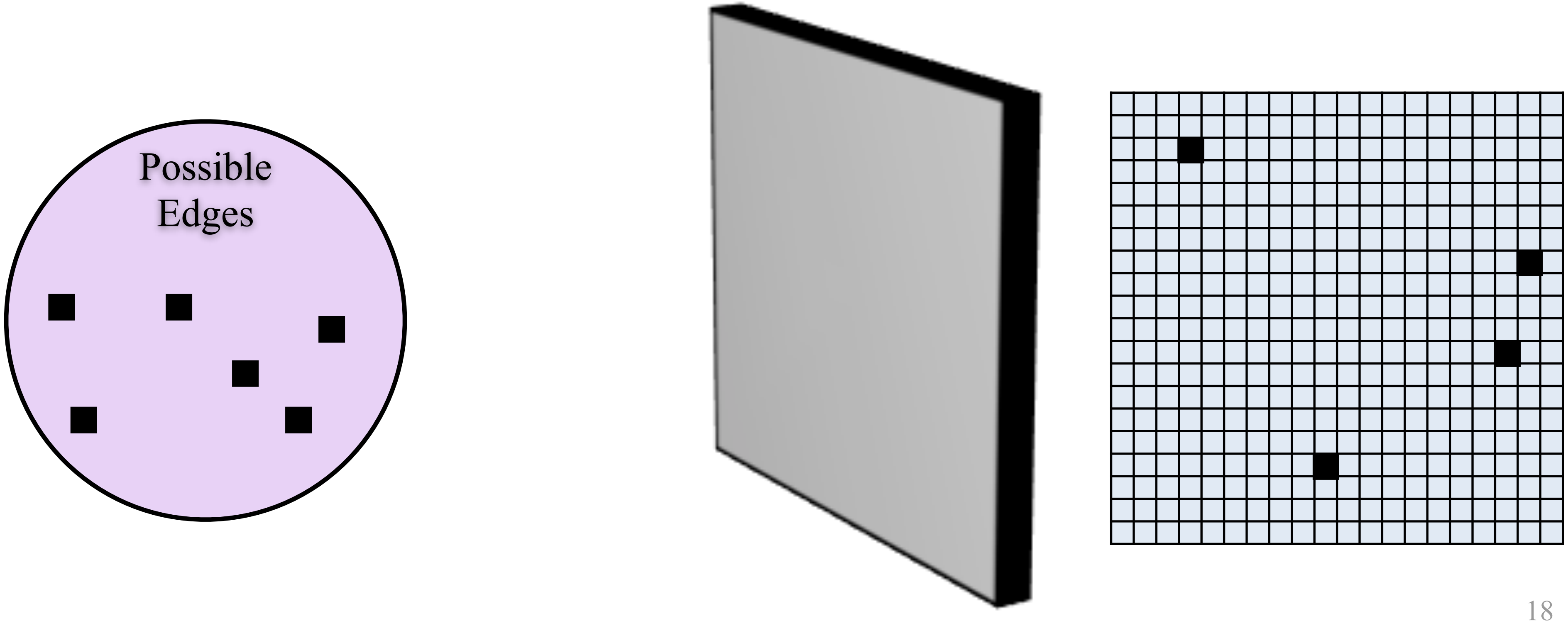




Generalizing and Exploiting

```
while not enough edges:
  draw (vi,vj) from Q’ (the model)

  if (vi, vj) not in edges
    put (vi, vj) into the edges

return edges
```

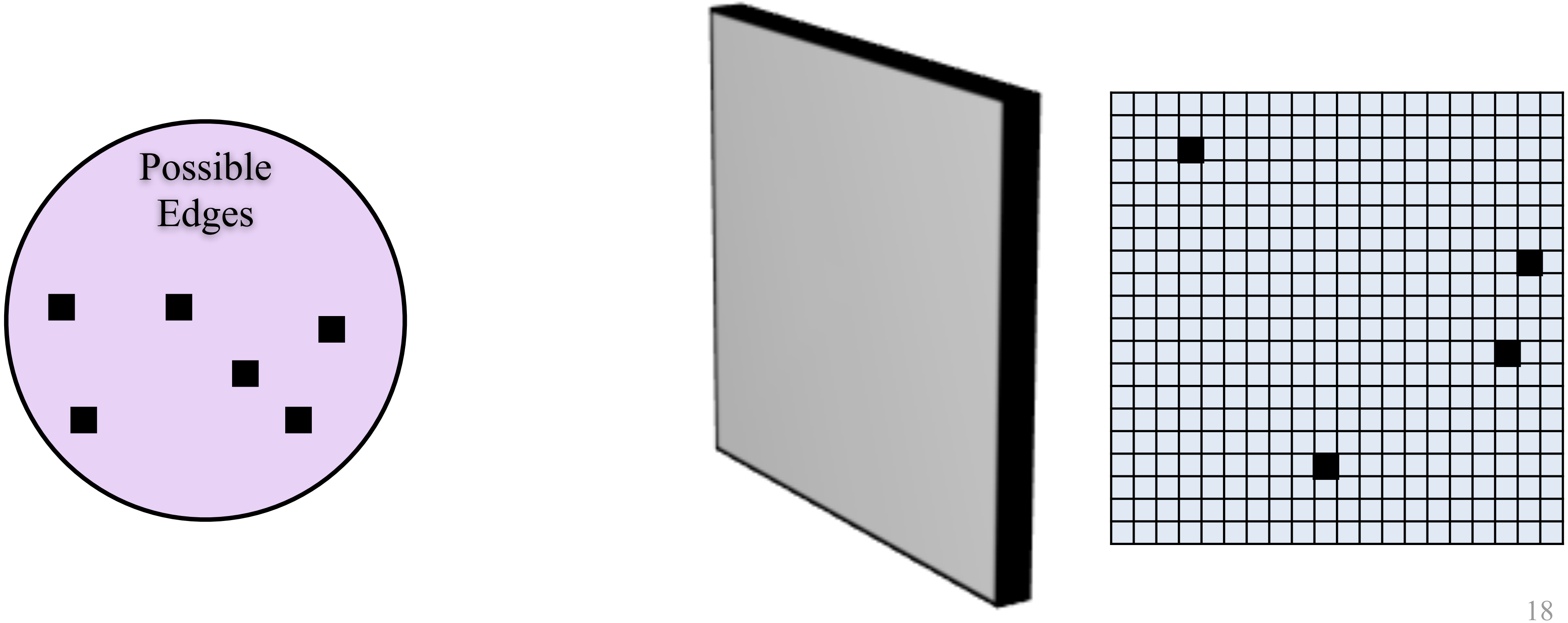
- Filter: *Rejects* duplicate edges



Generalizing and Exploiting

```
while not enough edges:
  draw (vi,vj) from Q' (the model)

  if (vi, vj) not in edges
    put (vi, vj) into the edges

return edges
```

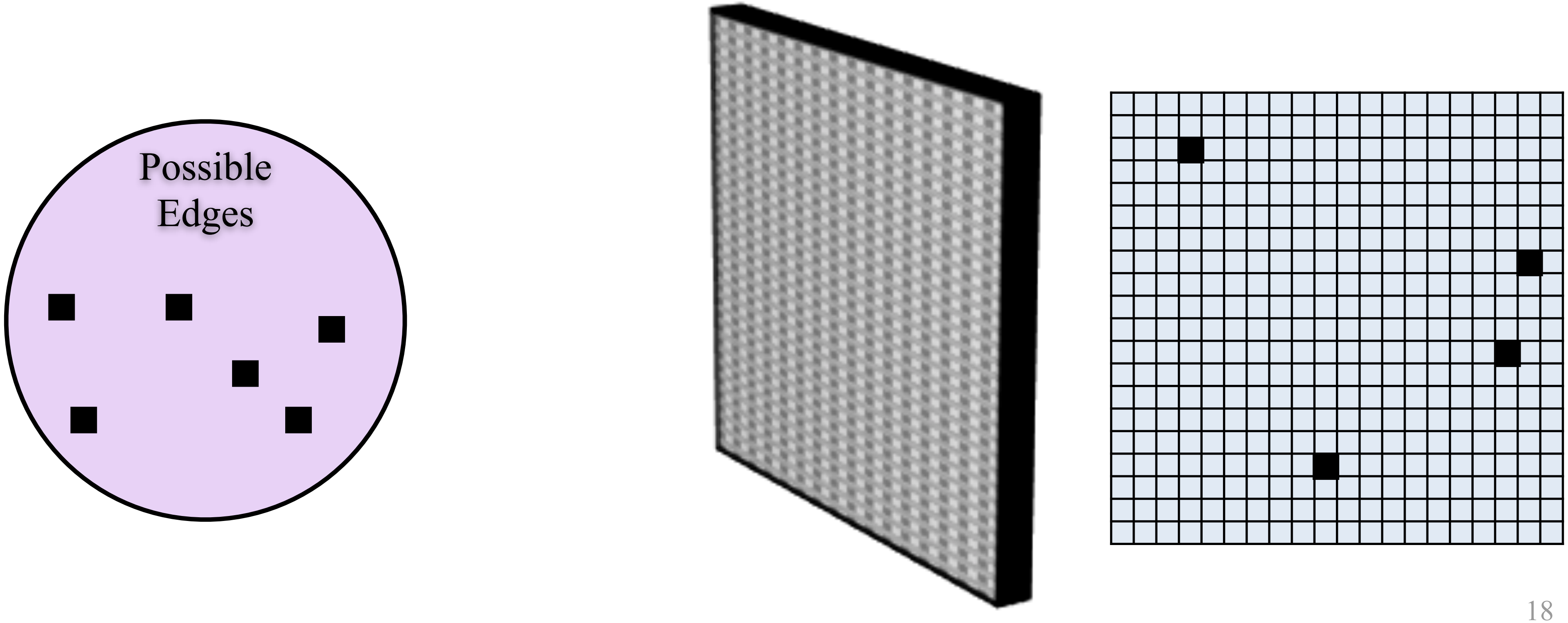
- Filter: *Rejects* duplicate edges
- Generalize to probabilistic rejections



Generalizing and Exploiting

```
while not enough edges:
  draw (vi,vj) from Q’ (the model)

  if (vi, vj) not in edges
    put (vi, vj) into the edges

return edges
```

- Filter: *Rejects* duplicate edges
- Generalize to probabilistic rejections



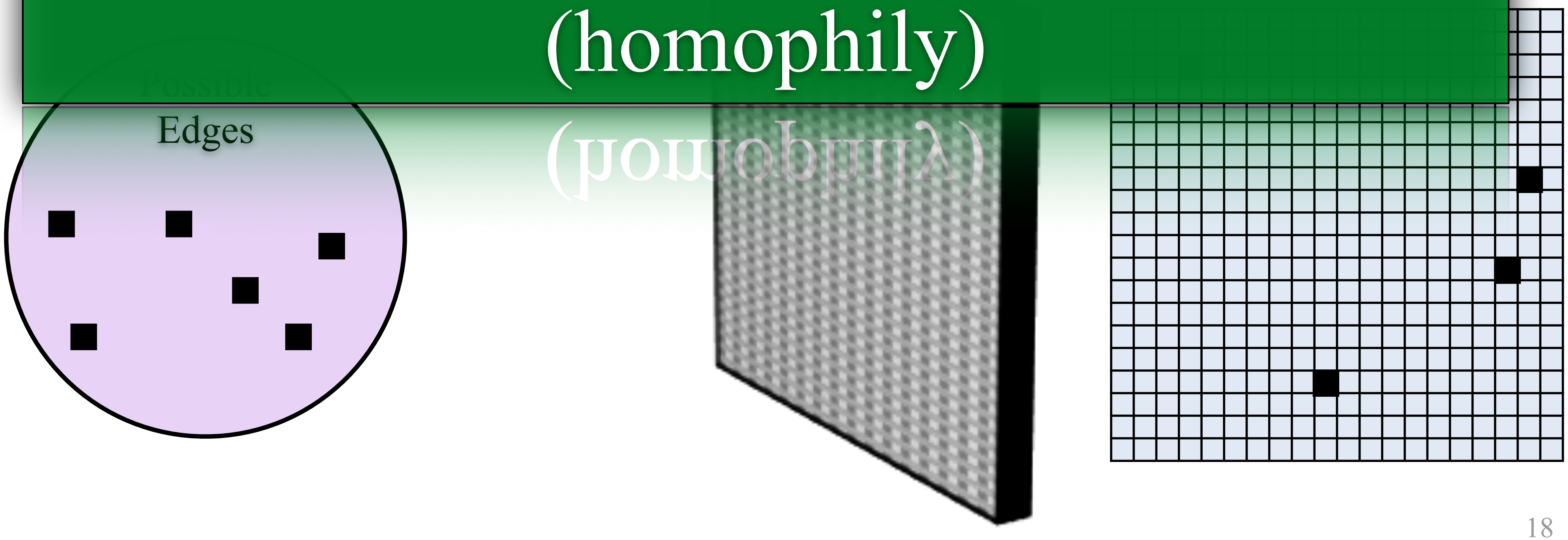
Generalizing and Exploiting

```
while not enough edges:
  draw (vi,vj) from Q' (the model)

  if (vi, vj) not in edges
    put (vi, vj) into the edges
retu
```

- Filter: *Rejects* duplicate edges
- Generalize to

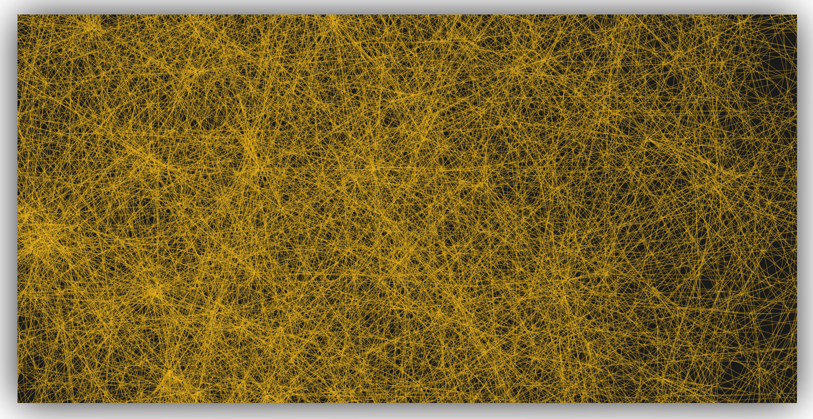
ons

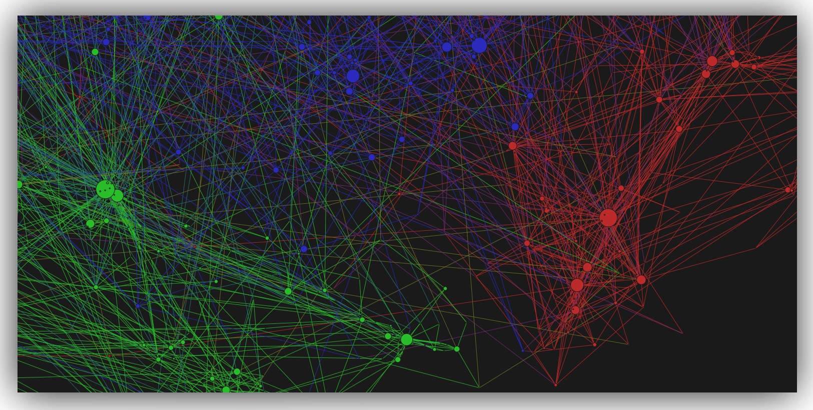
We Define a Probabilistic Filter which Samples Edges *conditioned on* Attributes (homophily)



Outline:

- Background
- Scalable Graph Sampling
- **Attributed Graph Models**
  - **Sampling**
  - Theoretical Results
  - Learning From Data
- Experiments
- Conclusions /
  Future Directions





Naive Approach

Naive Approach

- Assume independence

Naive Approach

- Assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}|\Theta_{\mathcal{E}}, \Theta_X) = P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})P(\mathbf{X}|\Theta_X)$$

Naive Approach

- Assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}|\Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})}P(\mathbf{X}|\Theta_X)$$

Naive Approach

- Assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E} | \Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E} | \Theta_{\mathcal{E}})} P(\mathbf{X} | \Theta_X)$$

- `NaiveApproach`$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$

Naive Approach

- Assume independence
  $$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}|\Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})} P(\mathbf{X}|\Theta_X)$$

- `NaiveApproach`$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$
  - Θ_X=`LearnAttribute`$(\mathbf{X}, \mathcal{X})$

Naive Approach

- Assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}|\Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})} P(\mathbf{X}|\Theta_X)$$

- NaiveApproach$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$
  - Θ_X=LearnAttribute$(\mathbf{X}, \mathcal{X})$
  - $\Theta_{\mathcal{E}}$=LearnStructure$(\mathbf{V}, \mathbf{E}, \mathcal{E})$

Naive Approach

- Assume independence

$$P_{\mathcal{E}}(\mathbf{X},\mathbf{E}|\Theta_{\mathcal{E}},\Theta_{X}) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})}P(\mathbf{X}|\Theta_{X})$$

- `NaiveApproach`$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
  - Θ_{X}`=LearnAttribute`$(\mathbf{X},\mathcal{X})$
  - $\Theta_{\mathcal{E}}$`=LearnStructure`$(\mathbf{V},\mathbf{E},\mathcal{E})$
  - `# Sample`

Naive Approach

- Assume independence
  $$P_{\mathcal{E}}(\mathbf{X},\mathbf{E}|\Theta_{\mathcal{E}},\Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})}P(\mathbf{X}|\Theta_X)$$

- `NaiveApproach`$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
  - Θ_X`=LearnAttribute`$(\mathbf{X},\mathcal{X})$
  - $\Theta_{\mathcal{E}}$`=LearnStructure`$(\mathbf{V},\mathbf{E},\mathcal{E})$
  - `# Sample`
  - $\mathbf{X}'$`=SampleAttribute`$(\mathbf{V},\mathcal{X},\Theta_X)$

Naive Approach

- Assume independence

$$P_{\mathcal{E}}(\mathbf{X},\mathbf{E}|\Theta_{\mathcal{E}},\Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})}P(\mathbf{X}|\Theta_X)$$

```
• NaiveApproach($(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$)
    – $\Theta_X$=LearnAttribute($(\mathbf{X},\mathcal{X})$)
    – $\Theta_{\mathcal{E}}$=LearnStructure($(\mathbf{V},\mathbf{E},\mathcal{E})$)
    – # Sample
    – $\mathbf{X}'$=SampleAttribute($(\mathbf{V},\mathcal{X},\Theta_X)$)
    – $\mathbf{E}'$=SampleStructure($(\mathbf{V},\mathcal{E},\Theta_{\mathcal{E}})$)
```

Naive Approach

- Assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}|\Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\Theta_{\mathcal{E}})} P(\mathbf{X}|\Theta_X)$$

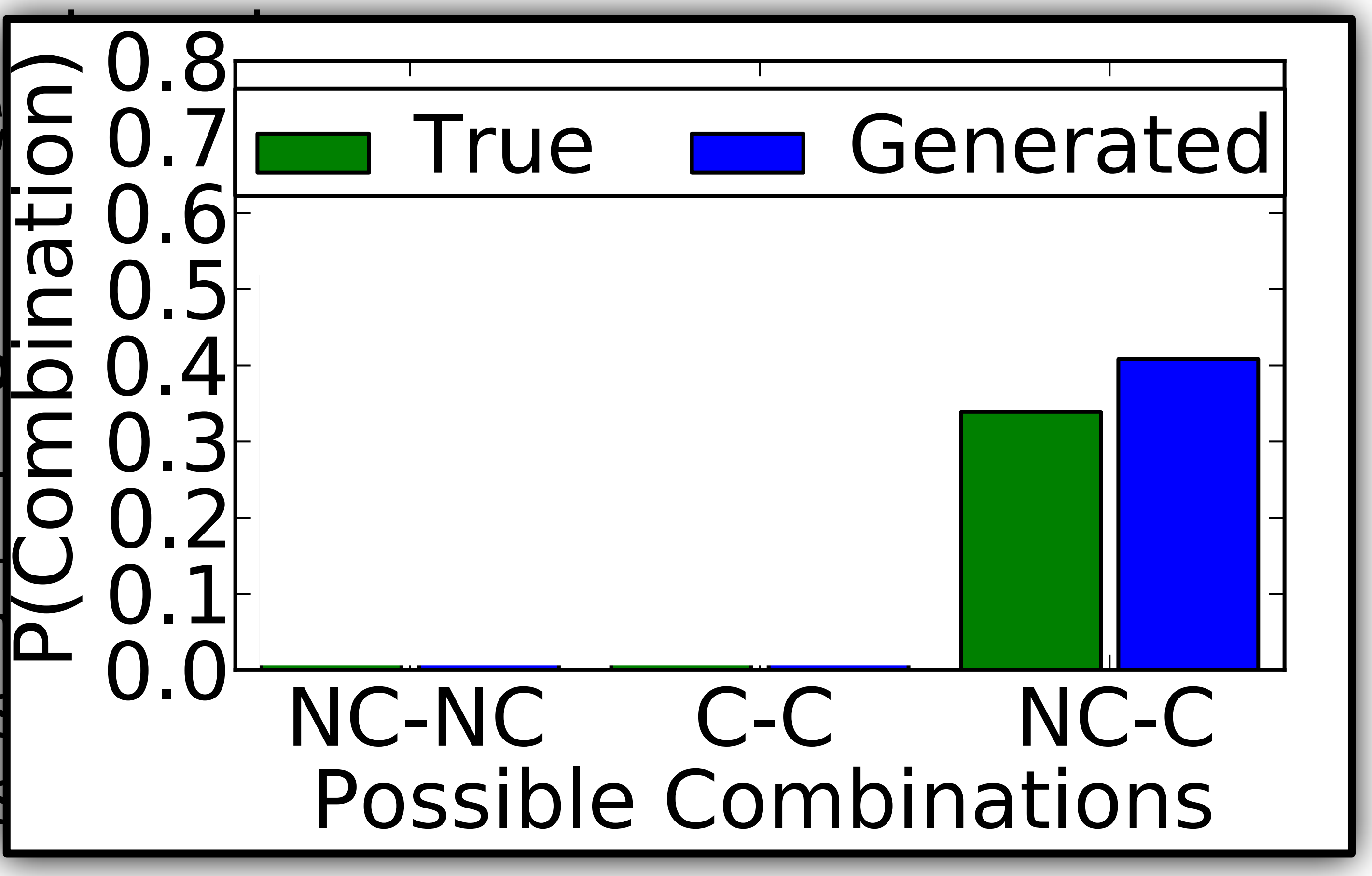
```
• NaiveApproach($\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X}$)
    – $\Theta_X$=LearnAttribute($\mathbf{X}, \mathcal{X}$)
    – $\Theta_{\mathcal{E}}$=LearnStructure($\mathbf{V}, \mathbf{E}, \mathcal{E}$)
    – # Sample
    – $\mathbf{X}'$=SampleAttribute($\mathbf{V}, \mathcal{X}, \Theta_X$)
    – $\mathbf{E}'$=SampleStructure($\mathbf{V}, \mathcal{E}, \Theta_{\mathcal{E}}$)

    – return ($\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X$)
```

Naive Approach

- Assume i

  $P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}$

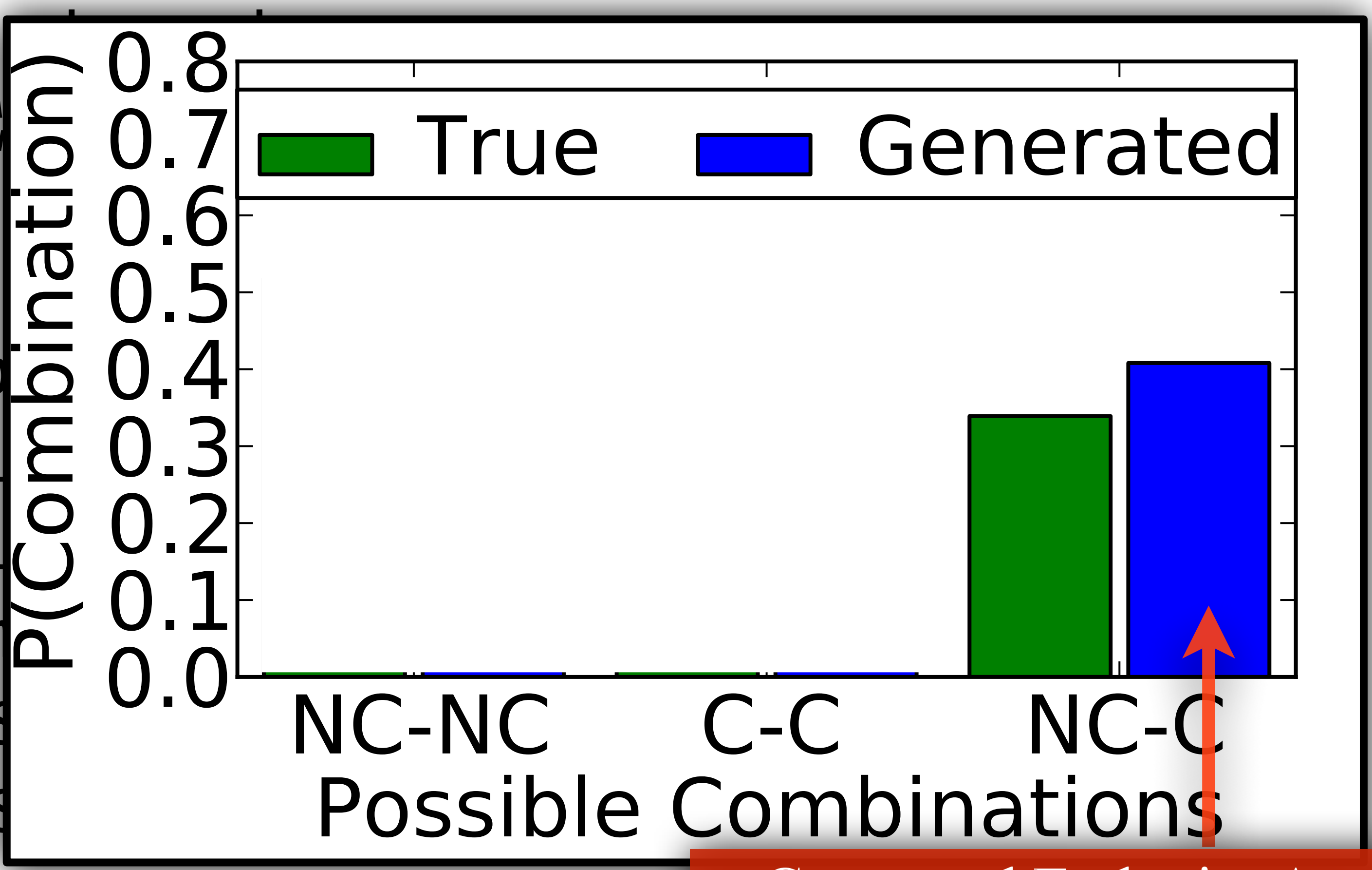
- NaiveAp
  - Θ_X=L
  - $\Theta_{\mathcal{E}}$=L
  - # Sa
  - $\mathbf{X}'$=S
  - $\mathbf{E}'$=S
  - return $(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$



Naive Approach

- Assume i
  $P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}$

- NaiveAp
  - Θ_X=L
  - $\Theta_{\mathcal{E}}$ =L
  - # Sar
  - $\mathbf{X}'$=S
  - $\mathbf{E}'$=S
  - return $(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$



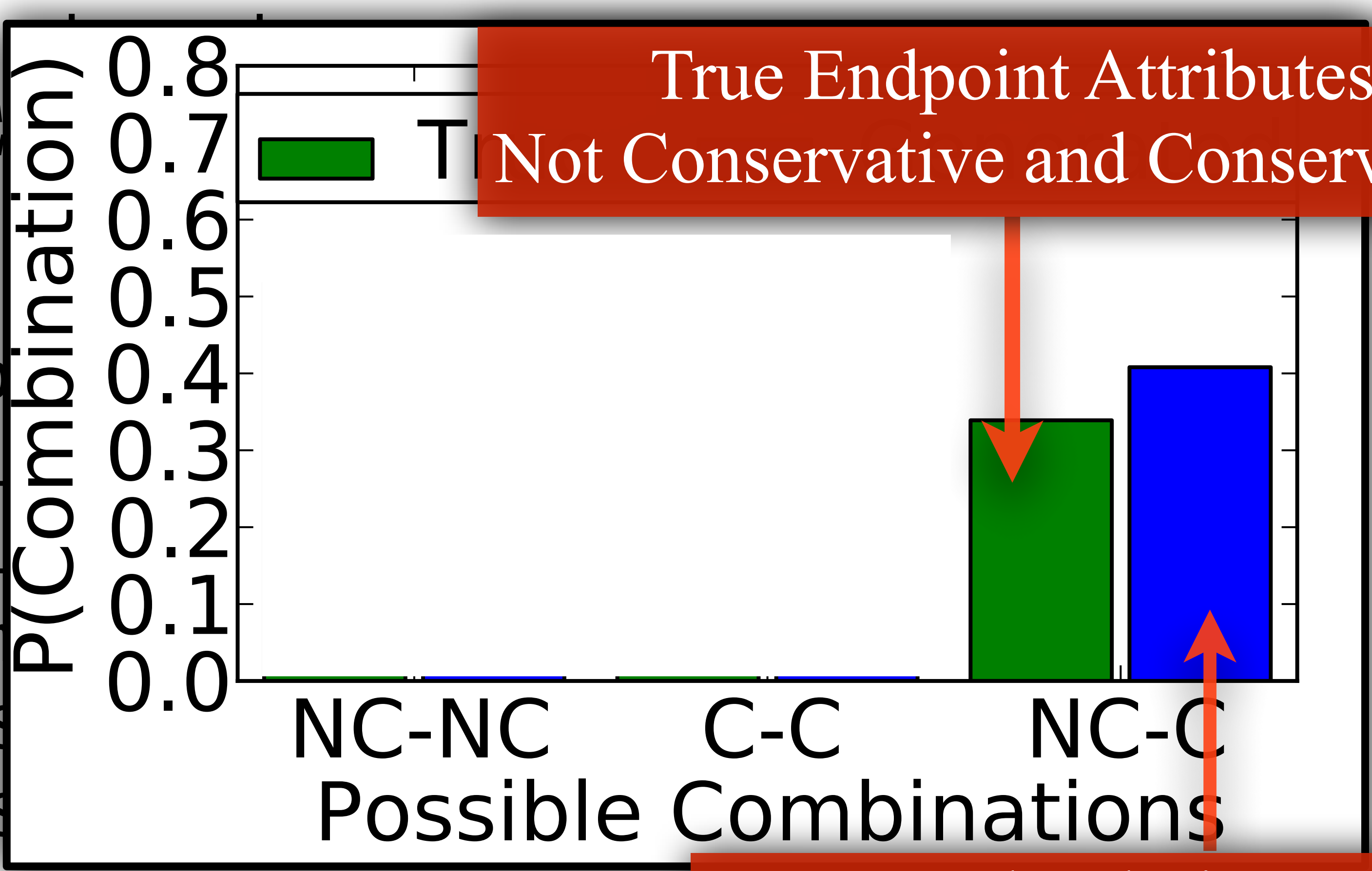
Generated Endpoint Attributes
Not Conservative and Conservative

Naive Approach

- Assume i

  $P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}$

- NaiveAp
  - Θ_X=L
  - $\Theta_{\mathcal{E}}$=L
  - # Sa
  - $\mathbf{X}'$=S
  - $\mathbf{E}'$=S
  - return $(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$



True Endpoint Attributes
Not Conservative and Conservative

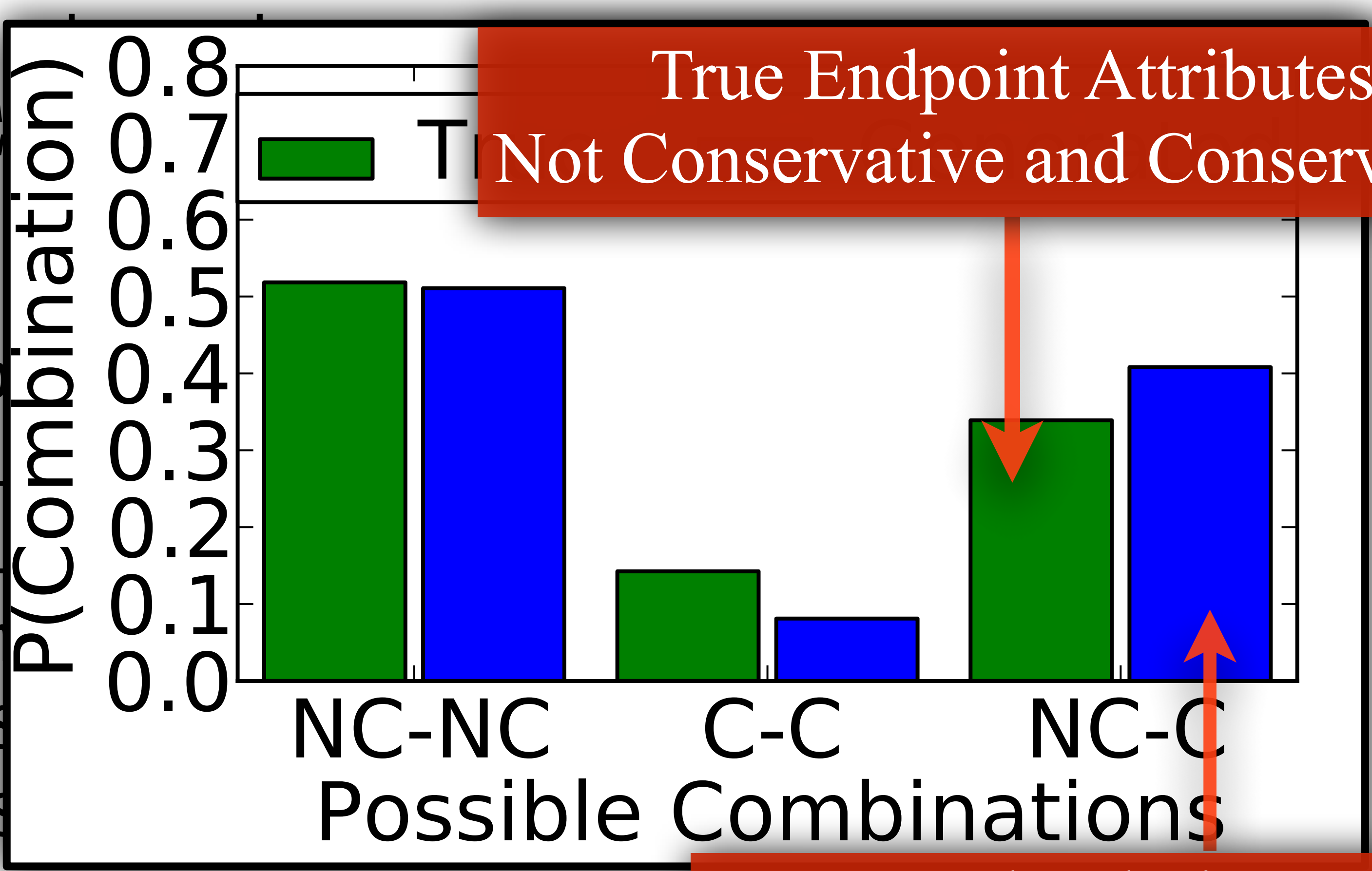
Generated Endpoint Attributes
Not Conservative and Conservative

Naive Approach

- Assume i

  $P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}$

- NaiveAp
  - Θ_X=L
  - $\Theta_{\mathcal{E}}$ =L
  - # Sa
  - $\mathbf{X}'$=S
  - $\mathbf{E}'$=S
  - return $(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$



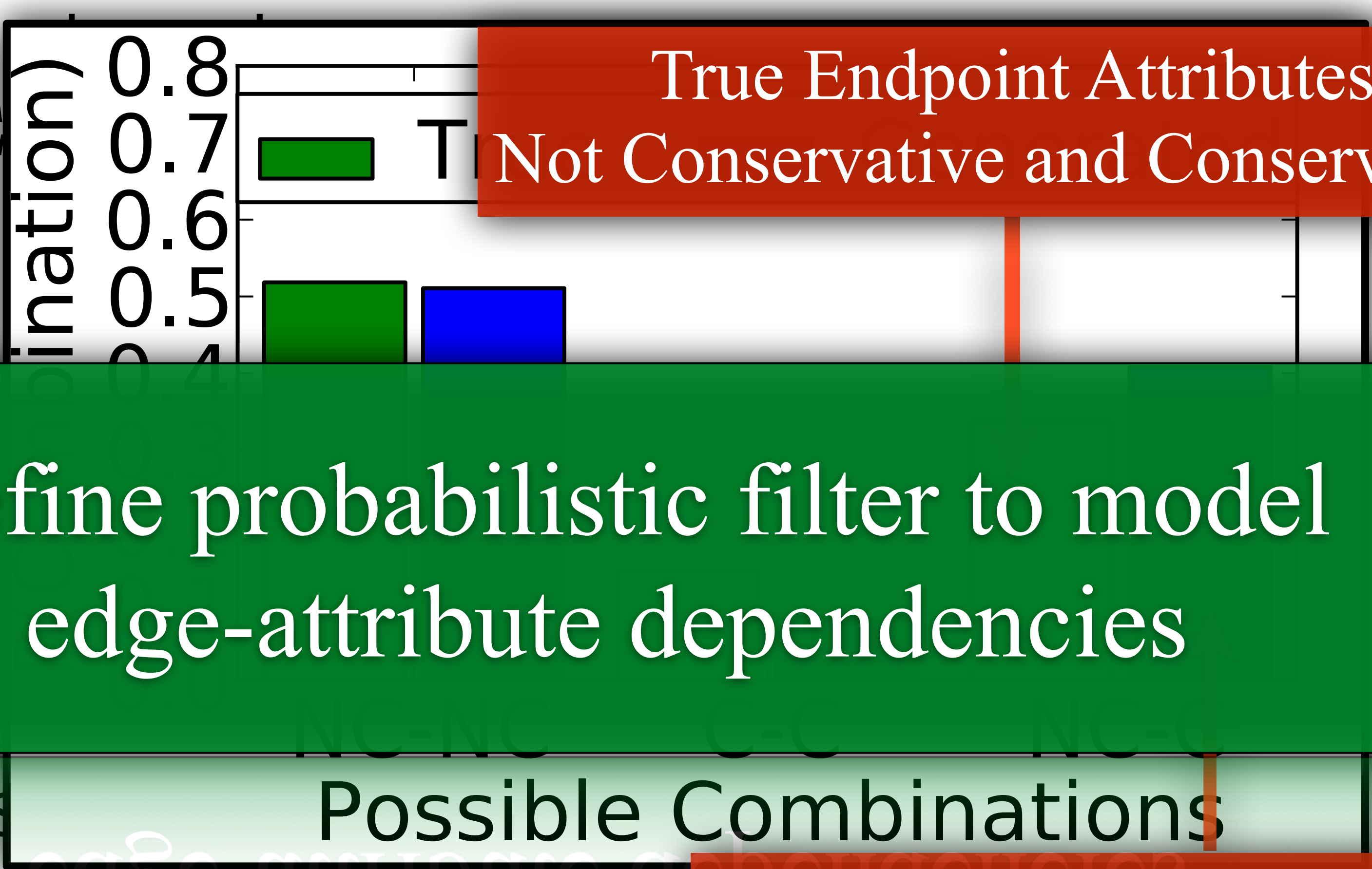
True Endpoint Attributes
Not Conservative and Conservative

Generated Endpoint Attributes
Not Conservative and Conservative

Naive Approach

- Assume i
  $P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}$

- NaiveAp
  - $\mathbf{E}' =$
  - return $(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$



True Endpoint Attributes
Not Conservative and Conservative

Define probabilistic filter to model edge-attribute dependencies

Generated Endpoint Attributes
Not Conservative and Conservative

Attributed Graph Models

Attributed Graph Models

- Do **not** assume independence

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E} | \Theta_{\mathcal{E}}, \Theta_X) = P_{\mathcal{E}}(\mathbf{E} | \mathbf{X}, \Theta_{\mathcal{E}}, \Theta_X) P(\mathbf{X} | \Theta_{\mathcal{E}}, \Theta_X)$$

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X},\mathbf{E}|\Theta_{\mathcal{E}},\Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\mathbf{X},\Theta_{\mathcal{E}},\Theta_X)}P(\mathbf{X}|\Theta_{\mathcal{E}},\Theta_X)$$

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E}|\Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\mathbf{X}, \Theta_{\mathcal{E}}, \Theta_X)} P(\mathbf{X}|\Theta_{\mathcal{E}}, \Theta_X)$$

Represent Using Graphical Models

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E} | \Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E} | \mathbf{X}, \Theta_{\mathcal{E}}, \Theta_X)} P(\mathbf{X} | \Theta_{\mathcal{E}}, \Theta_X)$$

Represent Using Graphical Models

- AGM$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E} | \Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E} | \mathbf{X}, \Theta_{\mathcal{E}}, \Theta_X)} P(\mathbf{X} | \Theta_{\mathcal{E}}, \Theta_X)$$

Represent Using Graphical Models

- `AGM`$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$
  - $(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$= `NaiveApproach`$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E} | \Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E} | \mathbf{X}, \Theta_{\mathcal{E}}, \Theta_X)} P(\mathbf{X} | \Theta_{\mathcal{E}}, \Theta_X)$$

Represent Using Graphical Models

- AGM$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$
  - $(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$= NaiveApproach$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$
  - $\mathbf{A}$ =ComputeAcceptProb$(\mathbf{E}, \mathbf{X}, \mathbf{E}', \mathbf{X}')$

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X},\mathbf{E}|\Theta_{\mathcal{E}},\Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\mathbf{X},\Theta_{\mathcal{E}},\Theta_X)}P(\mathbf{X}|\Theta_{\mathcal{E}},\Theta_X)$$

Represent Using Graphical Models

- AGM$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
  - $(\mathbf{E}',\mathbf{X}',\Theta_{\mathcal{E}},\Theta_X)$= NaiveApproach$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
  - $\mathbf{A}$ =ComputeAcceptProb$(\mathbf{E},\mathbf{X},\mathbf{E}',\mathbf{X}')$
  - $\mathbf{E}' = \emptyset$ # reset edges

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X},\mathbf{E}|\Theta_{\mathcal{E}},\Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\mathbf{X},\Theta_{\mathcal{E}},\Theta_X)} P(\mathbf{X}|\Theta_{\mathcal{E}},\Theta_X)$$

Represent Using Graphical Models

```
• AGM$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
  – $(\mathbf{E}',\mathbf{X}',\Theta_{\mathcal{E}},\Theta_X)$= NaiveApproach$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
  – $\mathbf{A}$ =ComputeAcceptProb$(\mathbf{E},\mathbf{X},\mathbf{E}',\mathbf{X}')$
  – $\mathbf{E}' = \emptyset$  # reset edges
  – while not enough edges
      draw (vi, vj) from Q’ (the model)

      U ~ Uniform(0, 1)
      if U < A(xi, xj)
        put (vi, vj) into $\mathbf{E}'$
```

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X},\mathbf{E}|\Theta_{\mathcal{E}},\Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\mathbf{X},\Theta_{\mathcal{E}},\Theta_X)} P(\mathbf{X}|\Theta_{\mathcal{E}},\Theta_X)$$

Represent Using Graphical Models

- AGM$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$

```
– $(\mathbf{E}',\mathbf{X}',\Theta_{\mathcal{E}},\Theta_X)$= NaiveApproach$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
– $\mathbf{A}$ =ComputeAcceptProb$(\mathbf{E},\mathbf{X},\mathbf{E}',\mathbf{X}')$
– $\mathbf{E}' = \emptyset$  # reset edges
– while not enough edges
    draw (vi, vj) from Q’ (the model)

    U ~ Uniform(0, 1)
    if U < A(xi, xj)
      put (vi, vj) into $\mathbf{E}'$
```

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X},\mathbf{E}|\Theta_{\mathcal{E}},\Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E}|\mathbf{X},\Theta_{\mathcal{E}},\Theta_X)}\, P(\mathbf{X}|\Theta_{\mathcal{E}},\Theta_X)$$

Represent Using Graphical Models

- AGM$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
  - $(\mathbf{E}',\mathbf{X}',\Theta_{\mathcal{E}},\Theta_X)$= NaiveApproach$(\mathbf{V},\mathbf{E},\mathbf{X},\mathcal{E},\mathcal{X})$
  - $\mathbf{A}$ =ComputeAcceptProb$(\mathbf{E},\mathbf{X},\mathbf{E}',\mathbf{X}')$
  - $\mathbf{E}' = \emptyset$ # reset edges
  - while not enough edges

```
while not enough edges
   draw (vi, vj) from Q' (the model)

   U ~ Uniform(0, 1)
   if U < A(xi, xj)
      put (vi, vj) into $\mathbf{E}'$
```

Probabilistic Filter

Attributed Graph Models

- Do **not** assume independence

$$P_{\mathcal{E}}(\mathbf{X}, \mathbf{E} | \Theta_{\mathcal{E}}, \Theta_X) = \boxed{P_{\mathcal{E}}(\mathbf{E} | \mathbf{X}, \Theta_{\mathcal{E}}, \Theta_X)} P(\mathbf{X} | \Theta_{\mathcal{E}}, \Theta_X)$$

Represent Using Graphical Models

- AGM$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$
  - $(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$= NaiveApproach$(\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathcal{E}, \mathcal{X})$
  - $\mathbf{A}$ =ComputeAcceptProb$(\mathbf{E}, \mathbf{X}, \mathbf{E}', \mathbf{X}')$
  - $\mathbf{E}' = \emptyset$ # reset edges
  - while not enough edges

```
while not enough edges
   draw (vi, vj) from Q' (the model)

   U ~ Uniform(0, 1)
   if U < A(xi, xj)
     put (vi, vj) into $\mathbf{E}'$
```

Probabilistic Filter

  - return$(\mathbf{E}', \mathbf{X}', \Theta_{\mathcal{E}}, \Theta_X)$

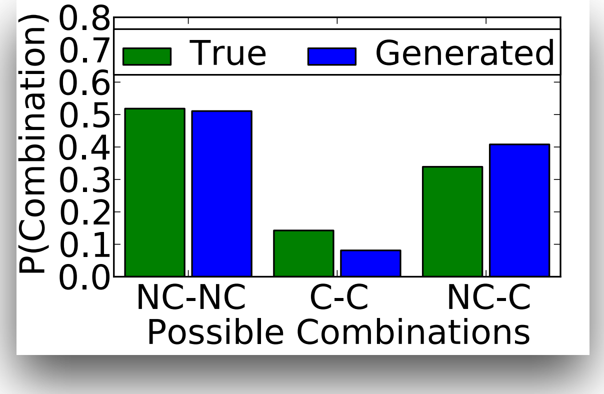
Attributed Graph Models

Attributed Graph Models

- What should the acceptance probabilities be?

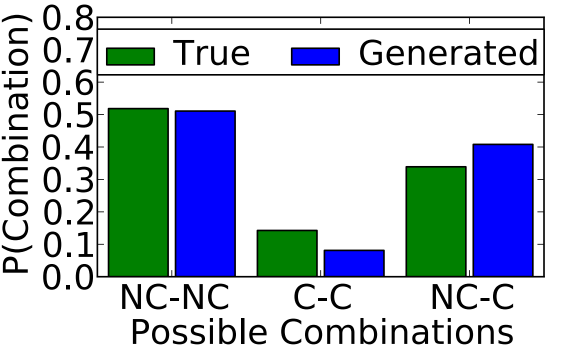
Attributed Graph Models

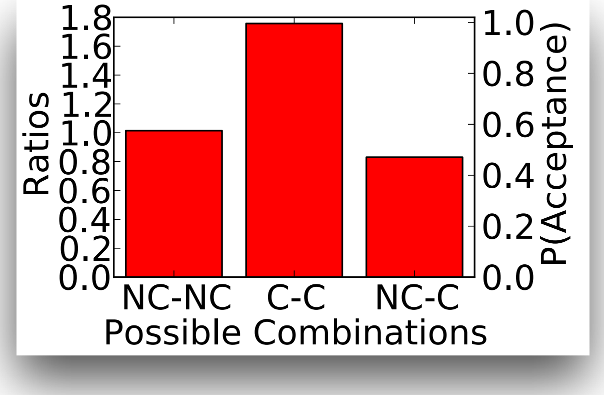
- What should the acceptance probabilities be?



Attributed Graph Models

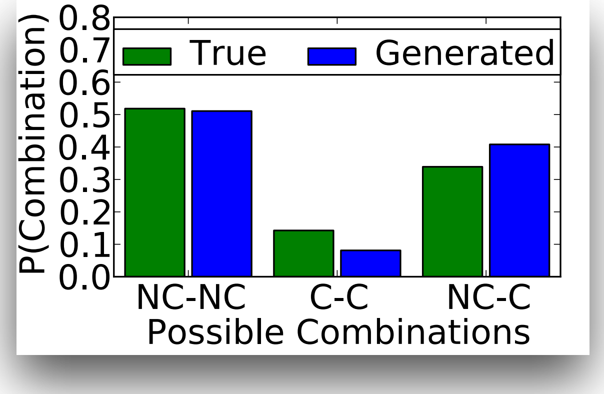
- What should the acceptance probabilities be?

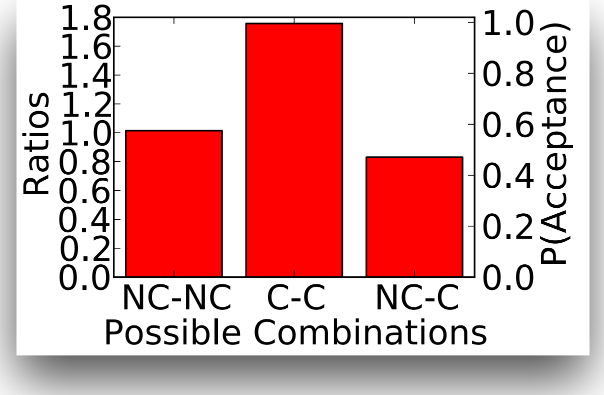




Attributed Graph Models

- What should the acceptance probabilities be?

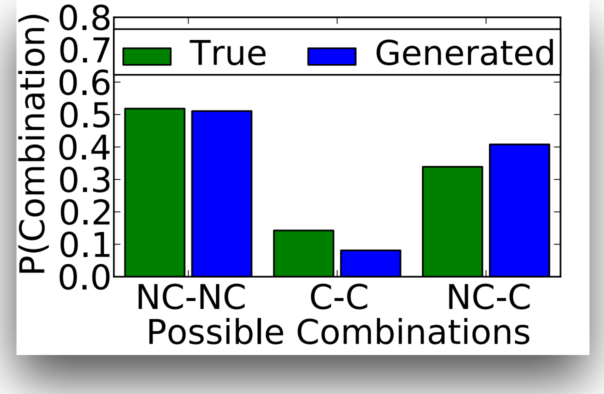


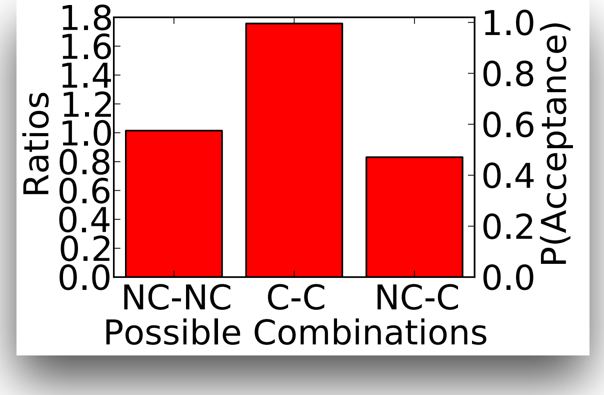


- Why?

Attributed Graph Models

- What should the acceptance probabilities be?

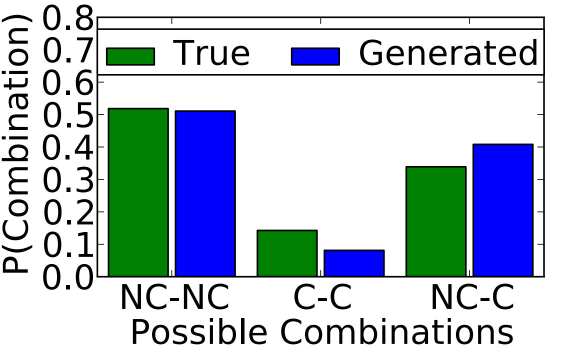


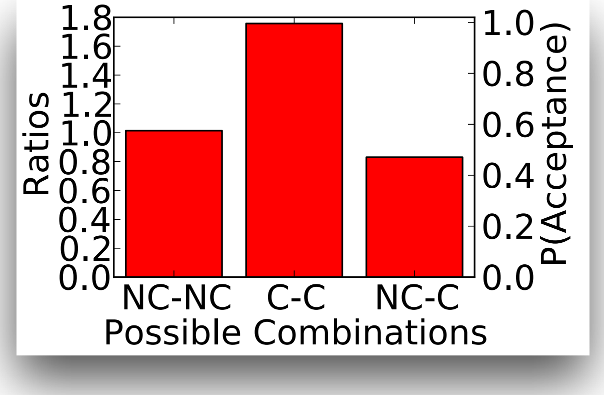


- Why? $P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$ (Thm. 1)

Attributed Graph Models

- What should the acceptance probabilities be?

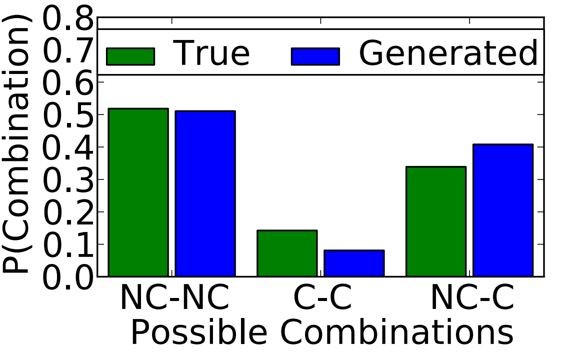


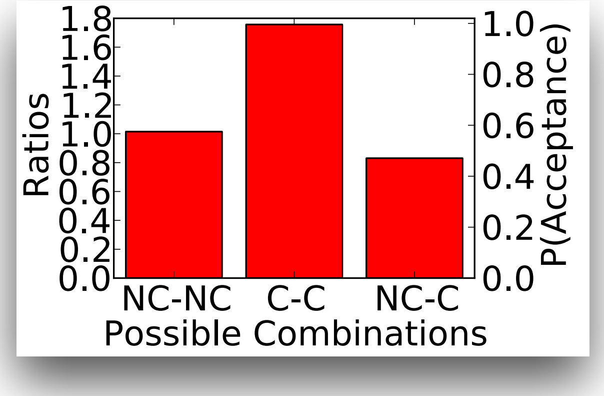


- Why? $P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$ (Thm. 1)
- Corresponds to *Rejection* sampling

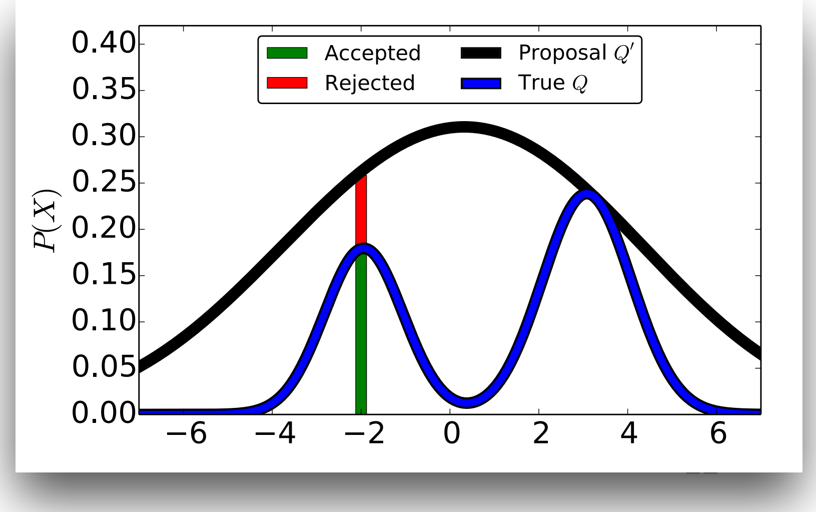
Attributed Graph Models

- What should the acceptance probabilities be?



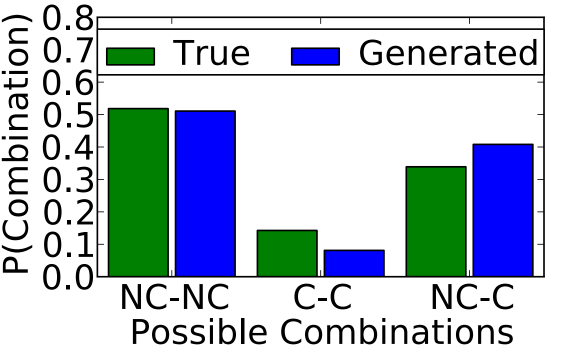


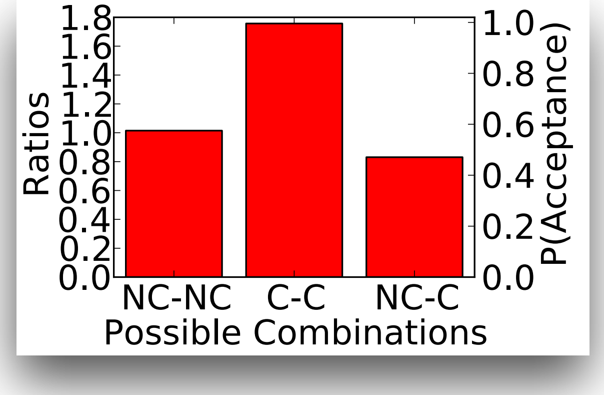
- Why? $P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$ (Thm. 1)
- Corresponds to *Rejection* sampling



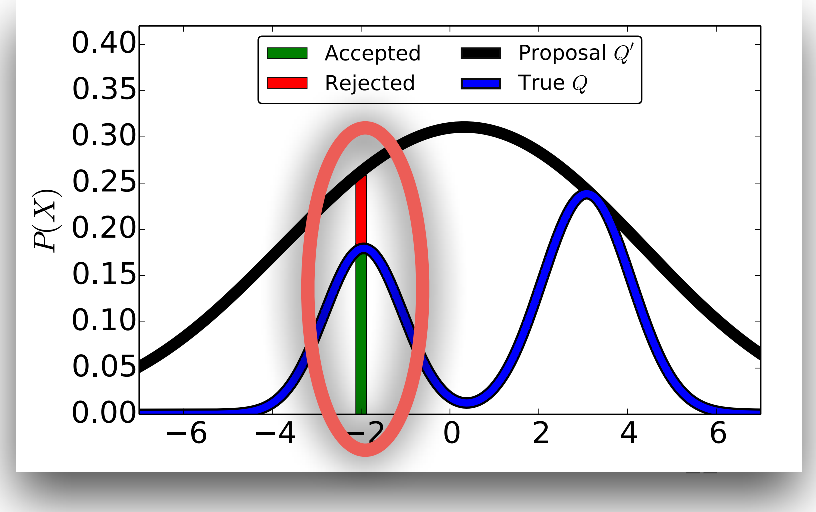
Attributed Graph Models

- What should the acceptance probabilities be?



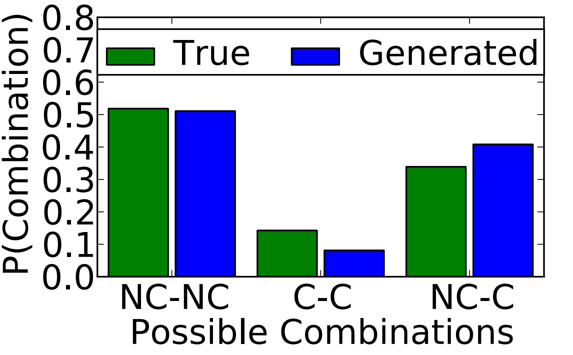


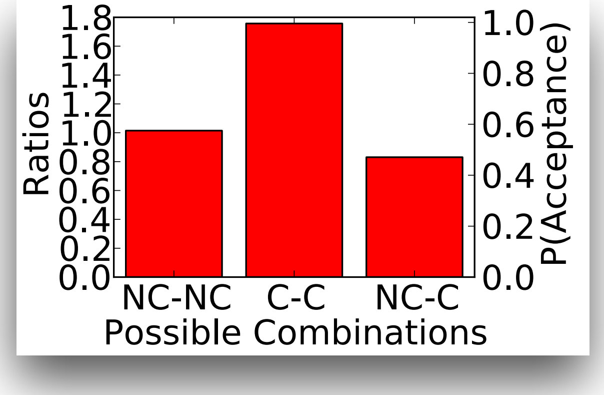
- Why? $P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$ (Thm. 1)
- Corresponds to *Rejection* sampling



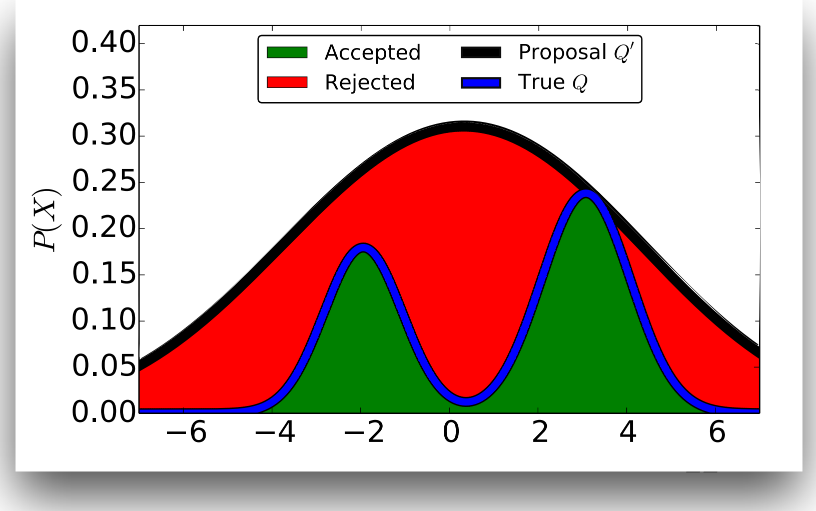
Attributed Graph Models

- What should the acceptance probabilities be?



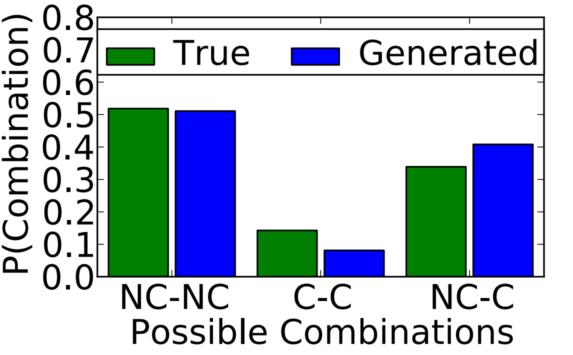


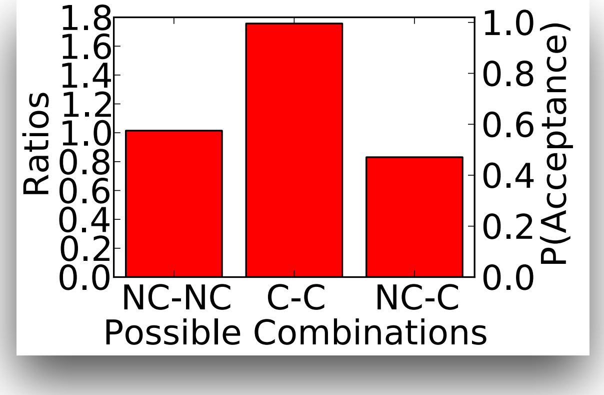
- Why? $P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$ (Thm. 1)
- Corresponds to *Rejection* sampling



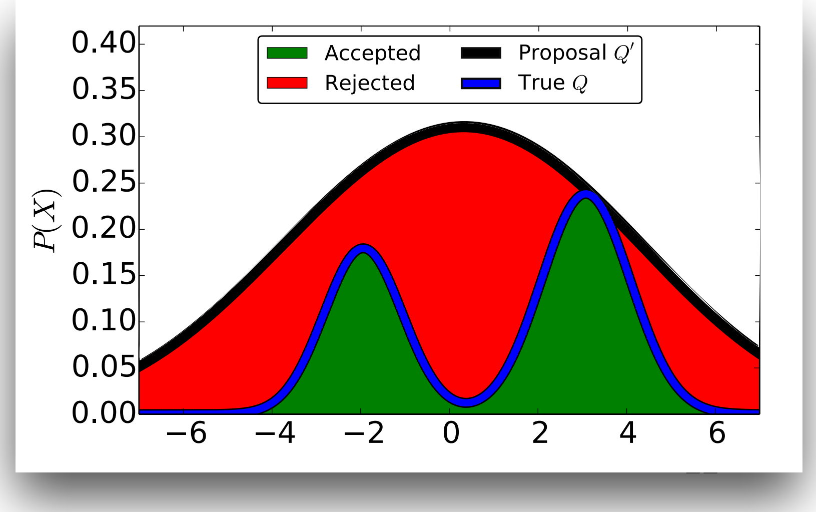
Attributed Graph Models

- What should the acceptance probabilities be?



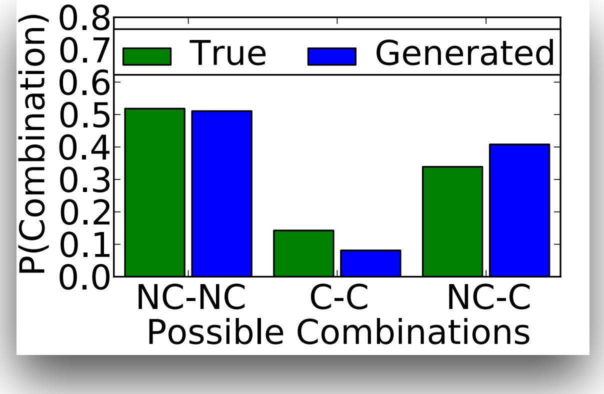


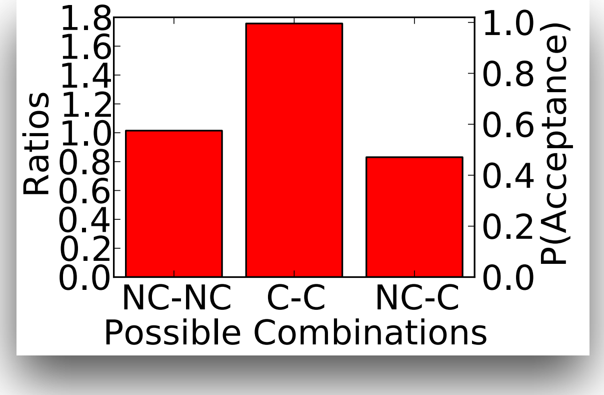
- Why? $P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$ (Thm. 1)
- Corresponds to *Rejection* sampling
- *Proposing Distribution*:



Attributed Graph Models

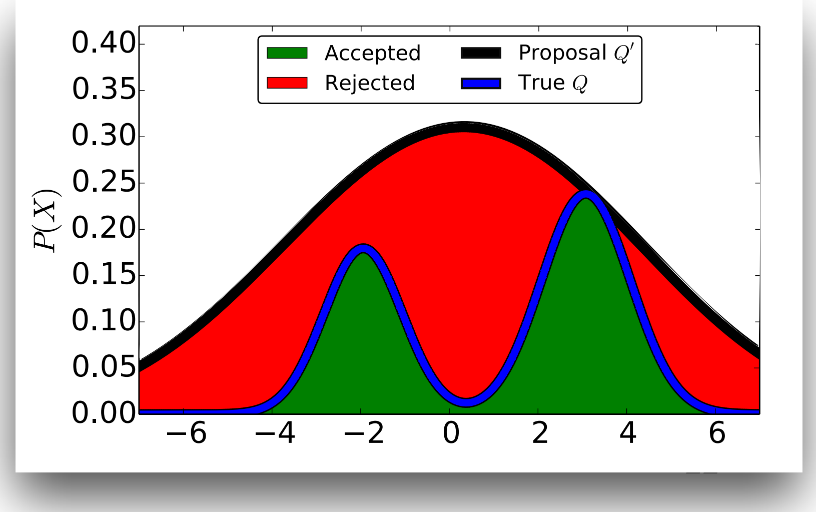
- What should the acceptance probabilities be?





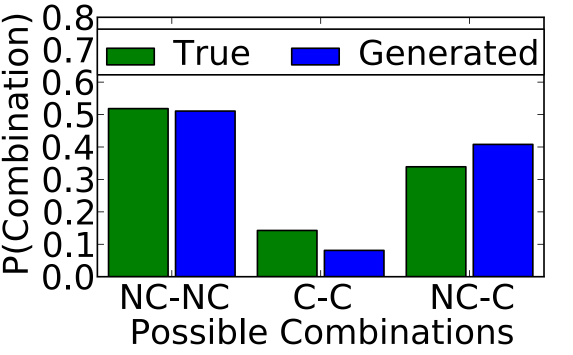
- Why? $P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$ (Thm. 1)
- Corresponds to *Rejection* sampling
- *Proposing Distribution*:

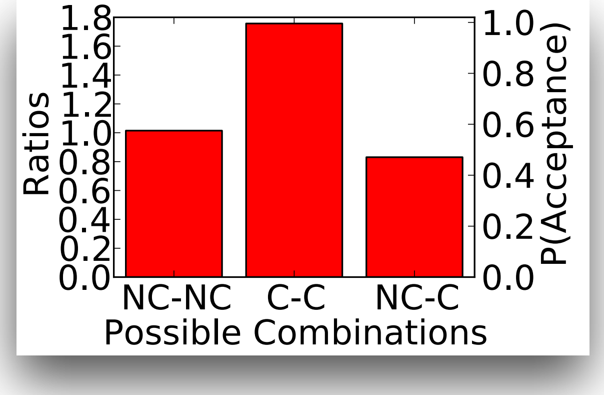
$$P_{\mathcal{E}}(E_{ij} = 1 | \Theta_{\mathcal{E}})$$



Attributed Graph Models

- What should the acceptance probabilities be?

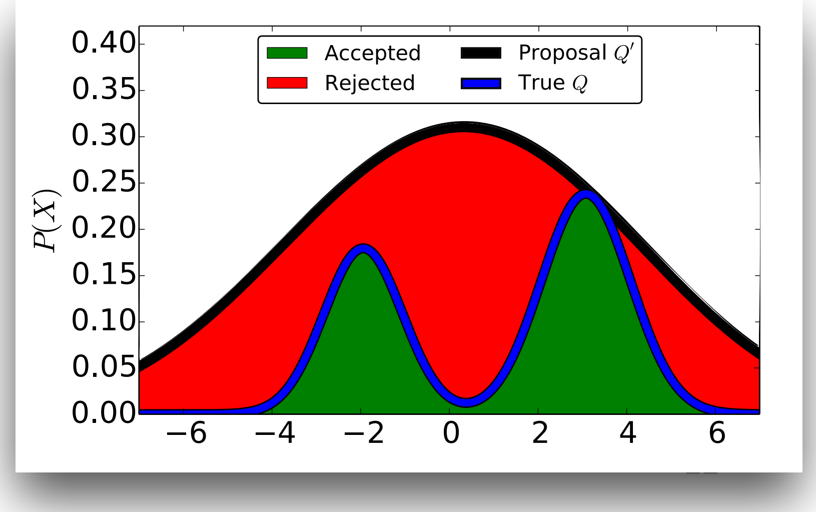




- Why? $P_o(E_{ij}=1|f(\mathbf{x}_i,\mathbf{x}_j),\Theta_{\mathcal{E}},\Theta_X)$ (Thm. 1)
- Corresponds to *Rejection* sampling
- *Proposing Distribution*:

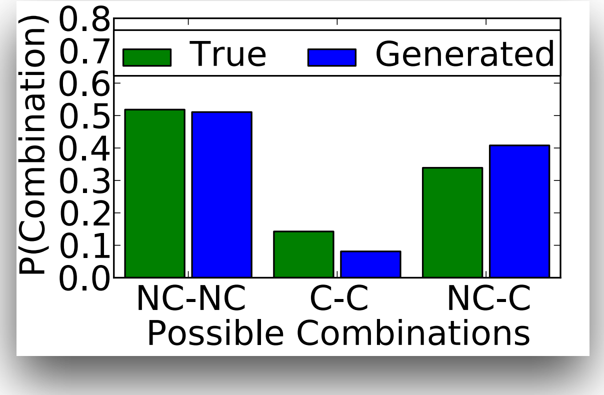
  $$P_{\mathcal{E}}(E_{ij}=1|\Theta_{\mathcal{E}})$$

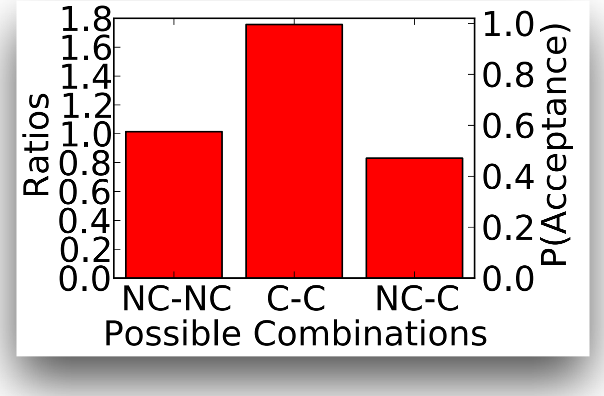
- *True Distribution*:



Attributed Graph Models

- What should the acceptance probabilities be?



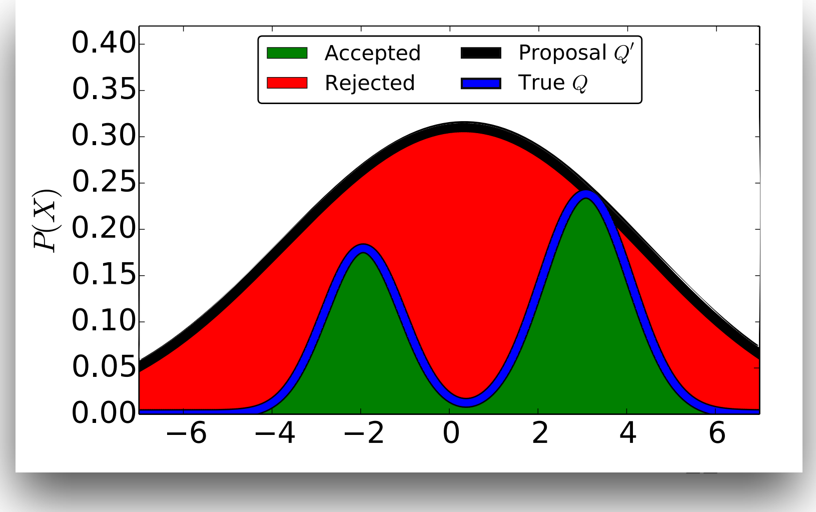


- Why? $P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$ (Thm. 1)
- Corresponds to *Rejection* sampling
- *Proposing Distribution*:

$$P_{\mathcal{E}}(E_{ij} = 1 | \Theta_{\mathcal{E}})$$

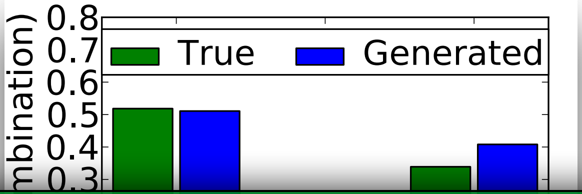
- *True Distribution*:

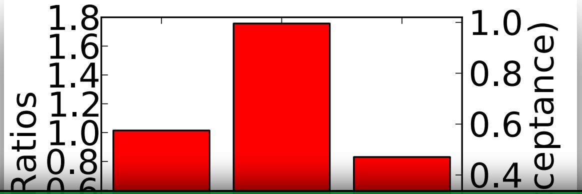
$$P_o(E_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$$



Attributed Graph Models

- What should the acceptance probabilities be?





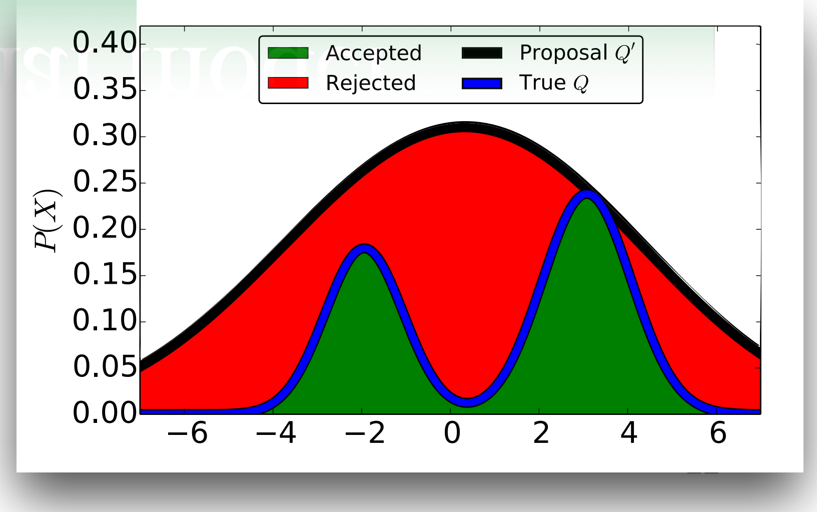
Scalable structural model allows a scalable conditional model

- Why?
- Corresponds to *Rejection* sampling
- *Proposing Distribution*:

$$P_{\mathcal{E}}(E_{ij} = 1|\Theta_{\mathcal{E}})$$

- *True Distribution*:

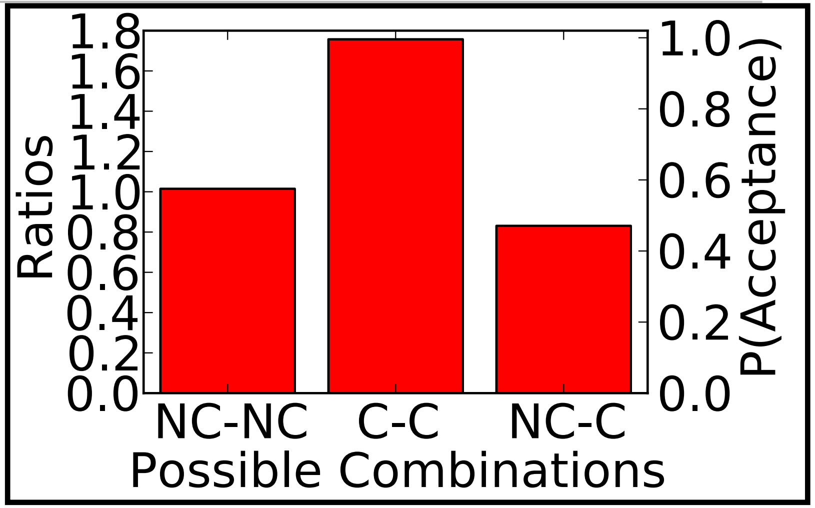
$$P_o(E_{ij} = 1|f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_X)$$

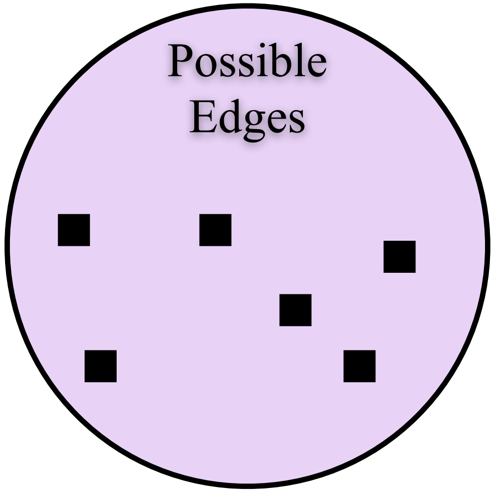


Attributed Graph Models

Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```

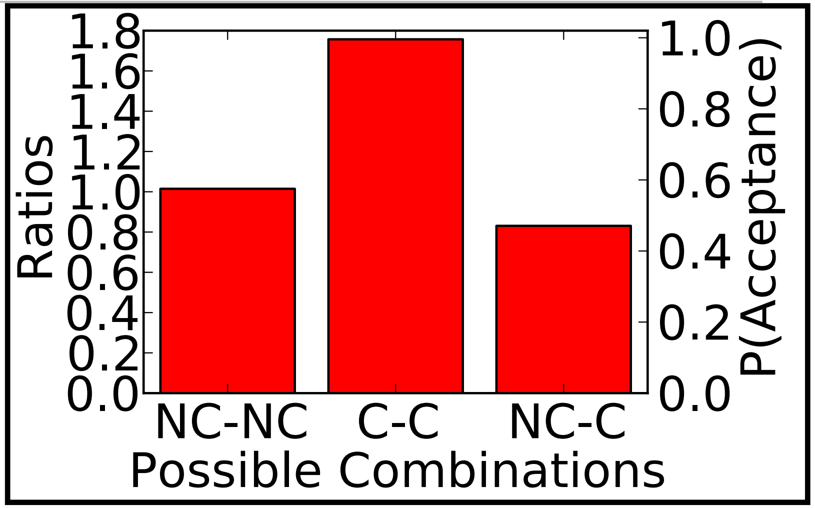


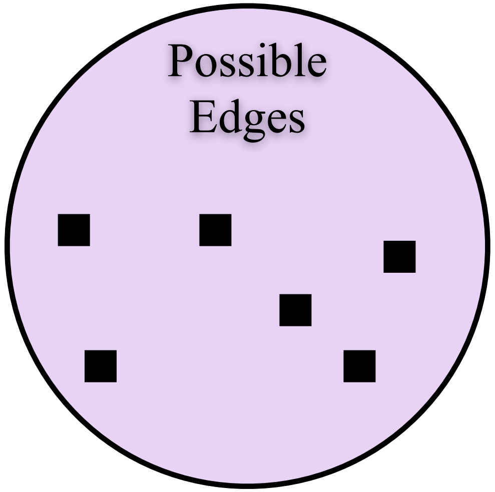


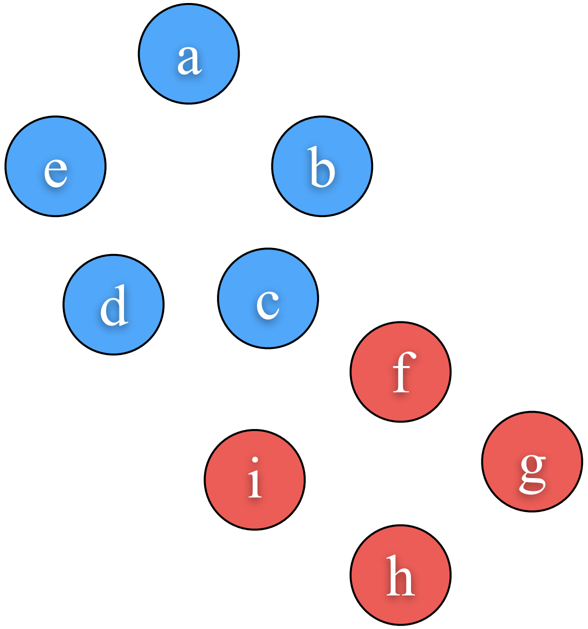
Possible Edges

Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```

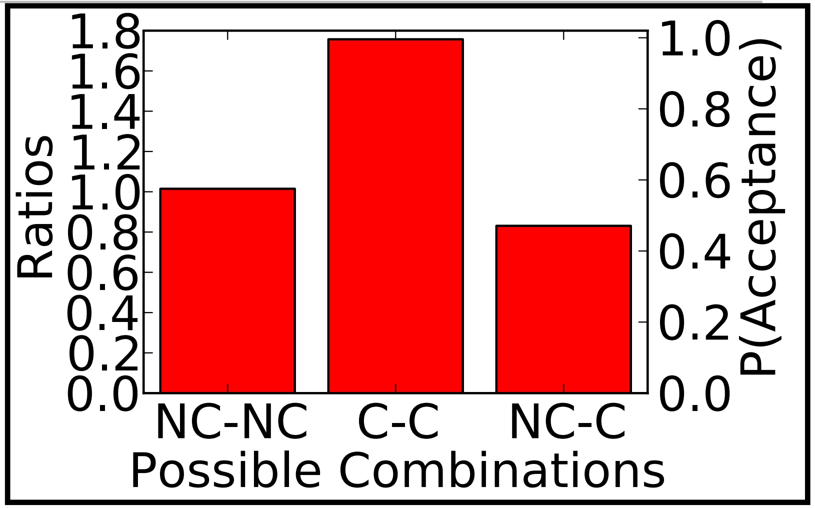


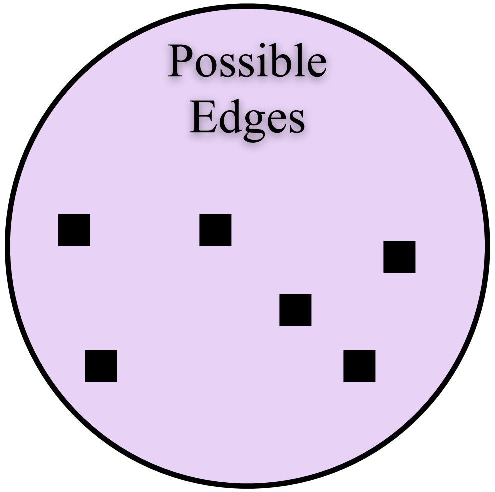


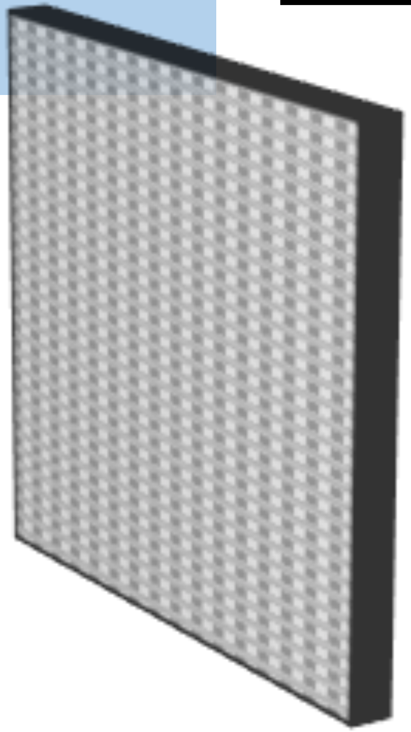


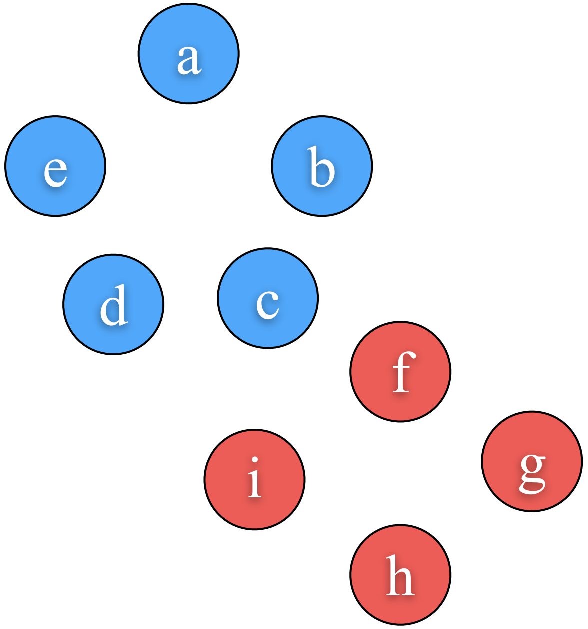
Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



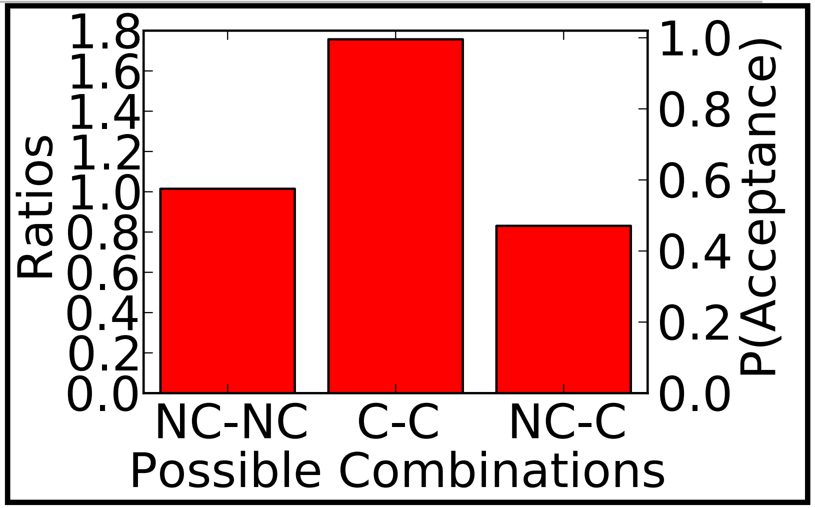


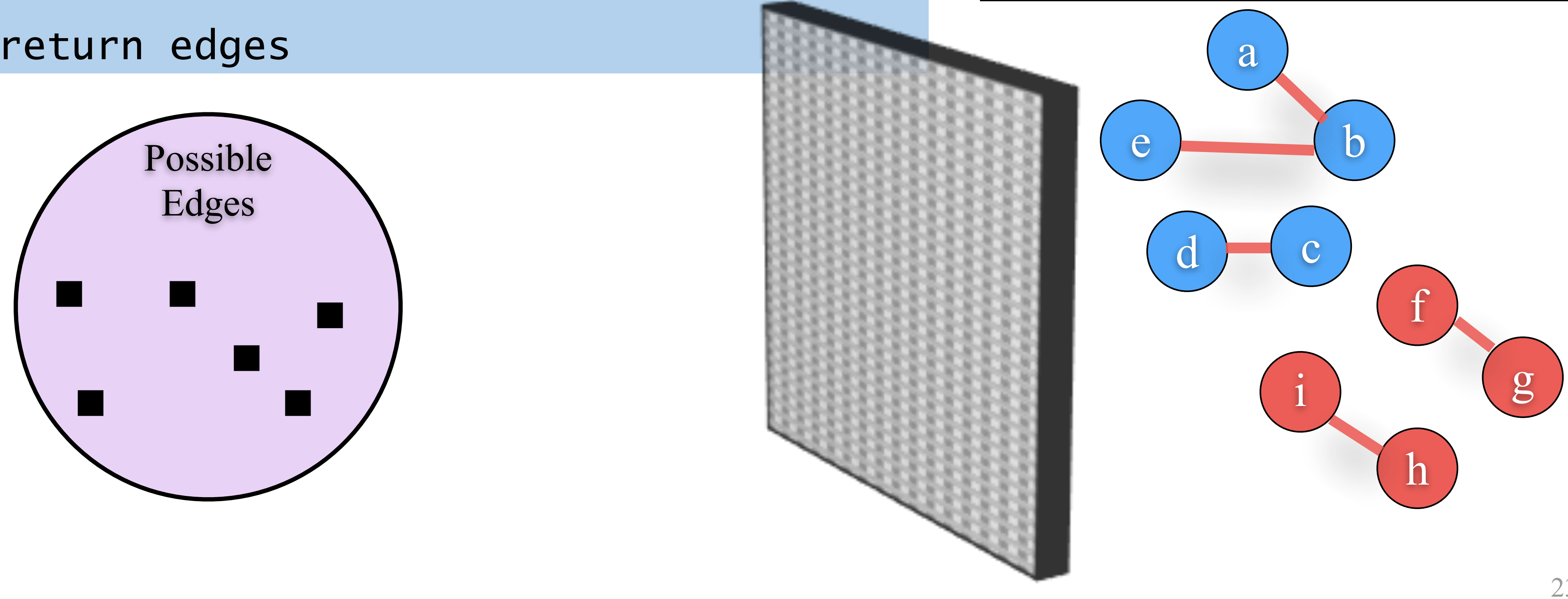




Attributed Graph Models

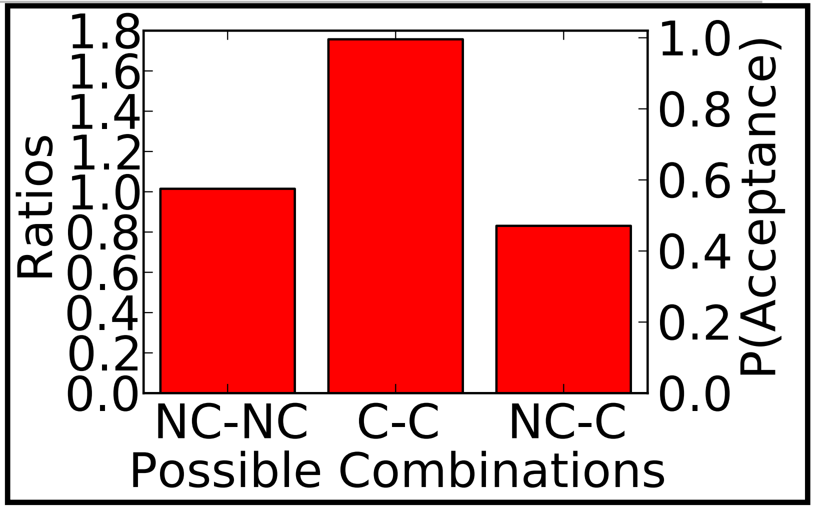
```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```

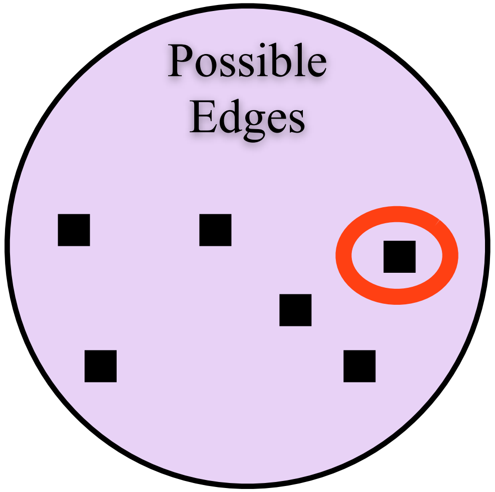


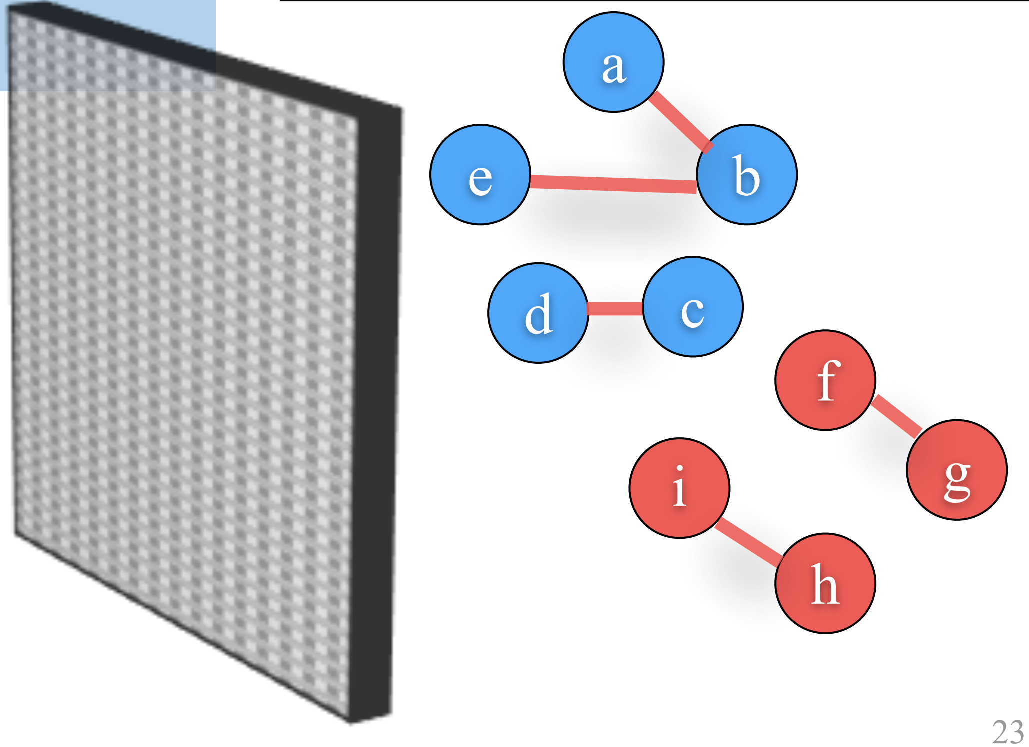


Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q' (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```

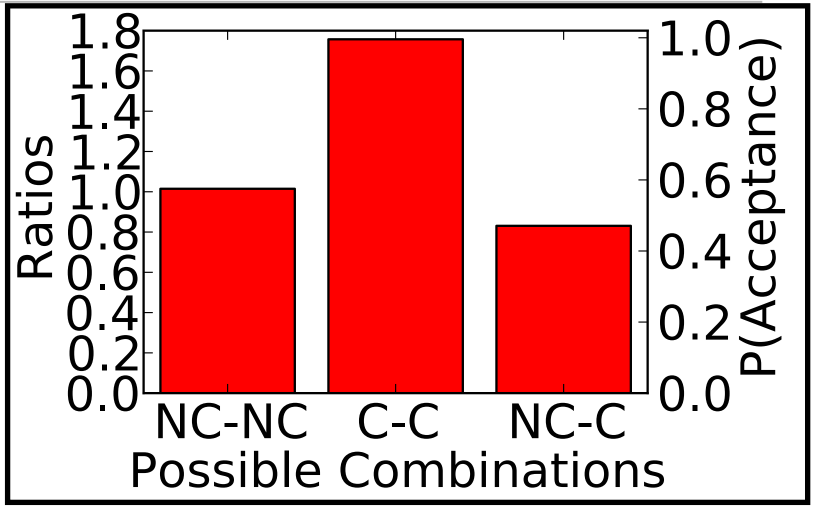


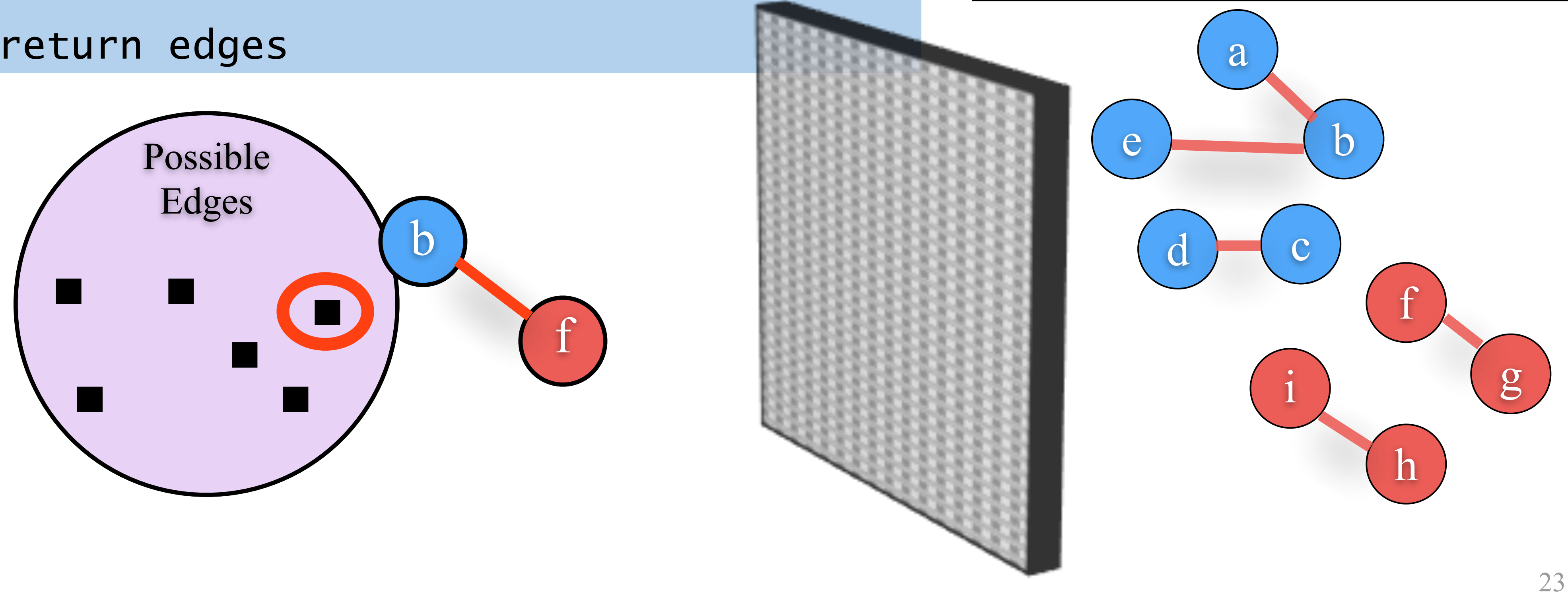




Attributed Graph Models

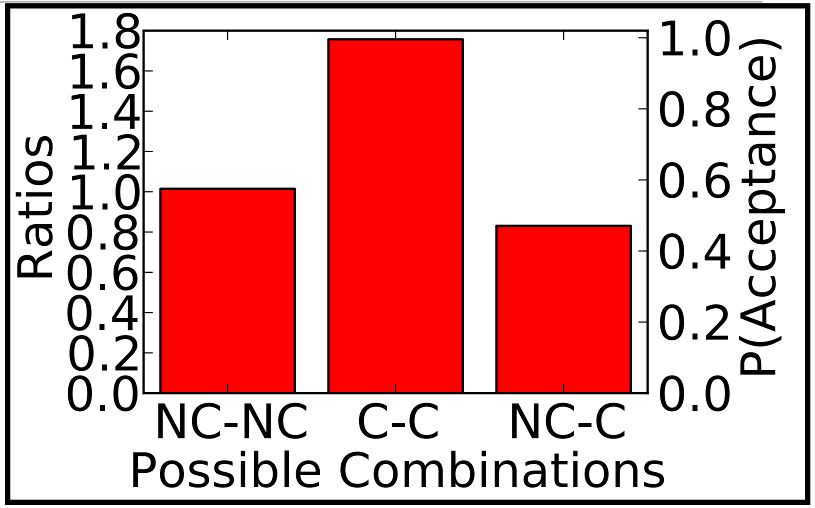
```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q' (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```

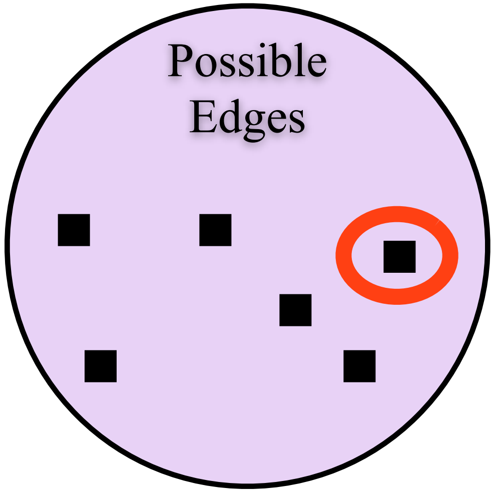


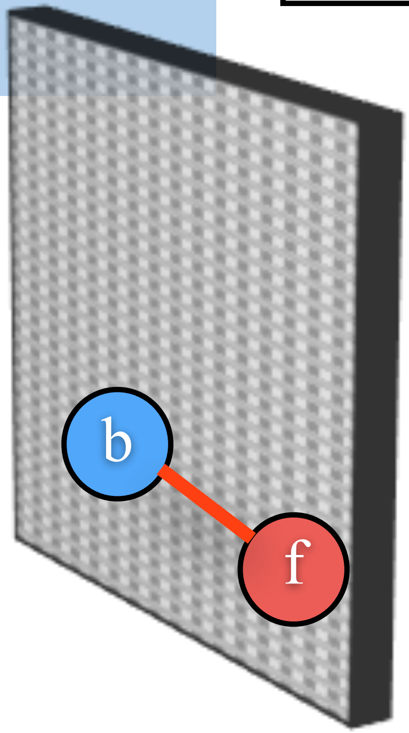


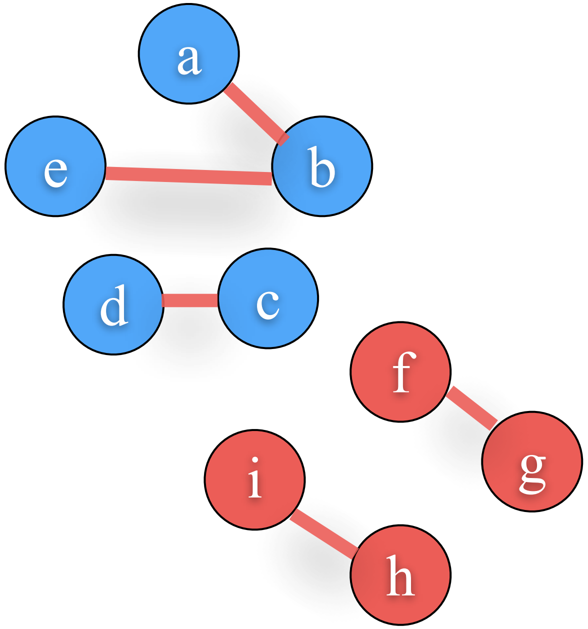
Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q' (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



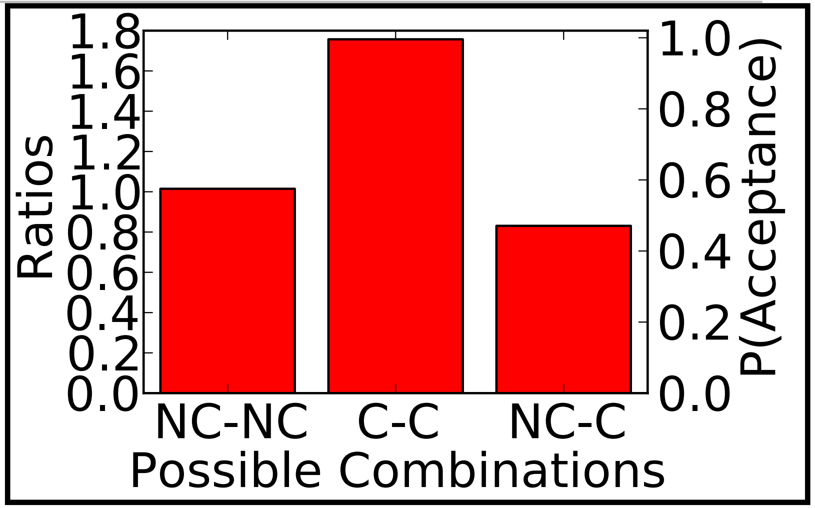


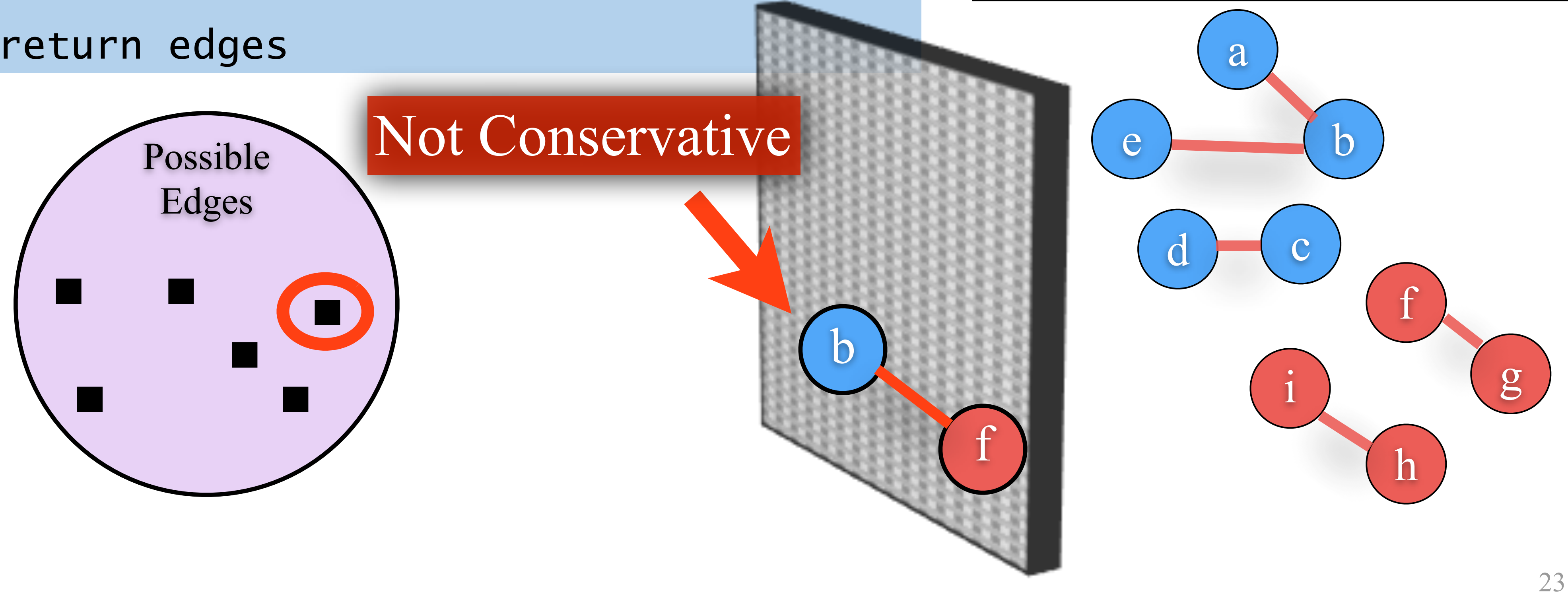




Attributed Graph Models

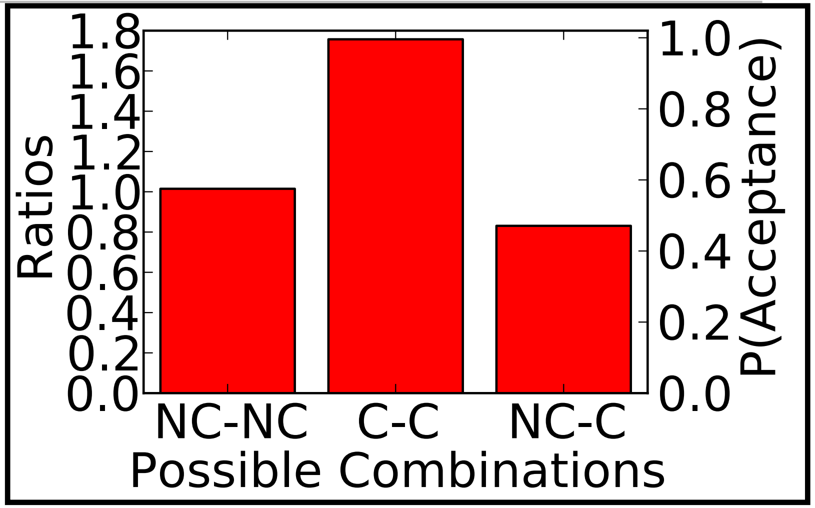
```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q' (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```

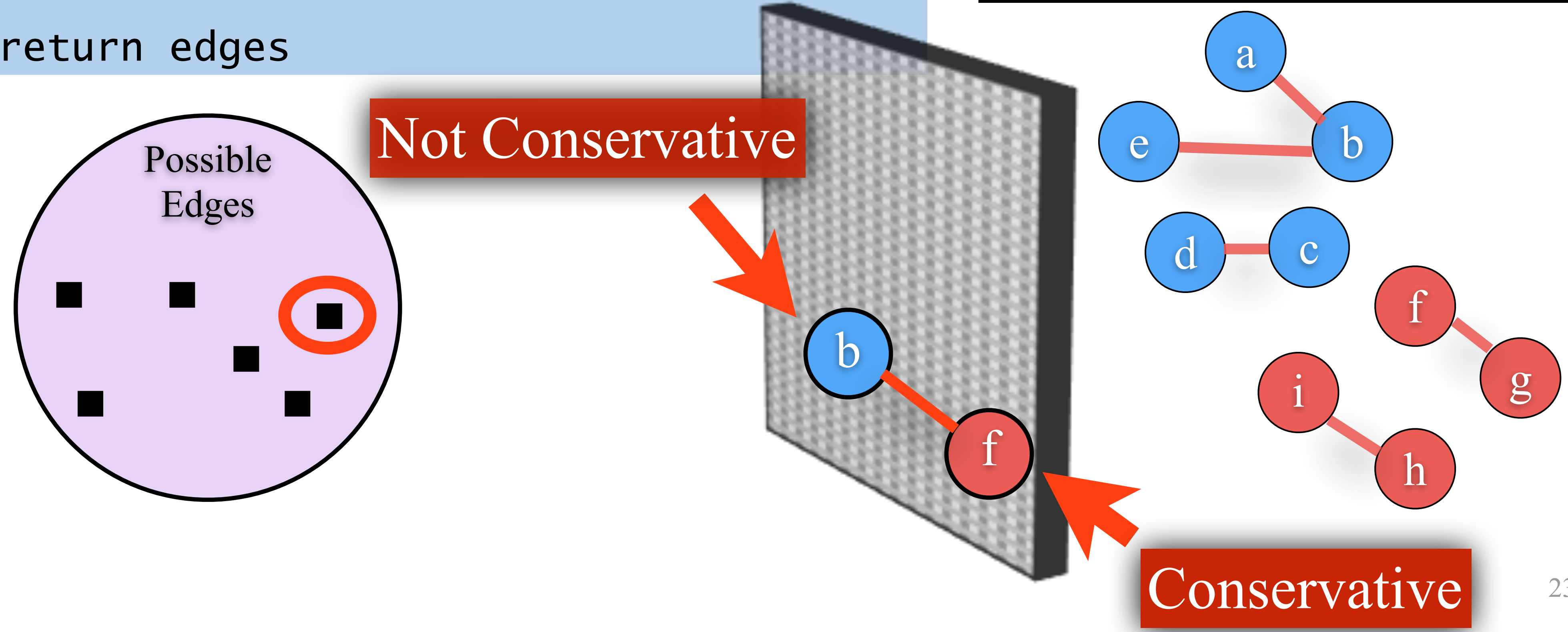



Not Conservative

Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



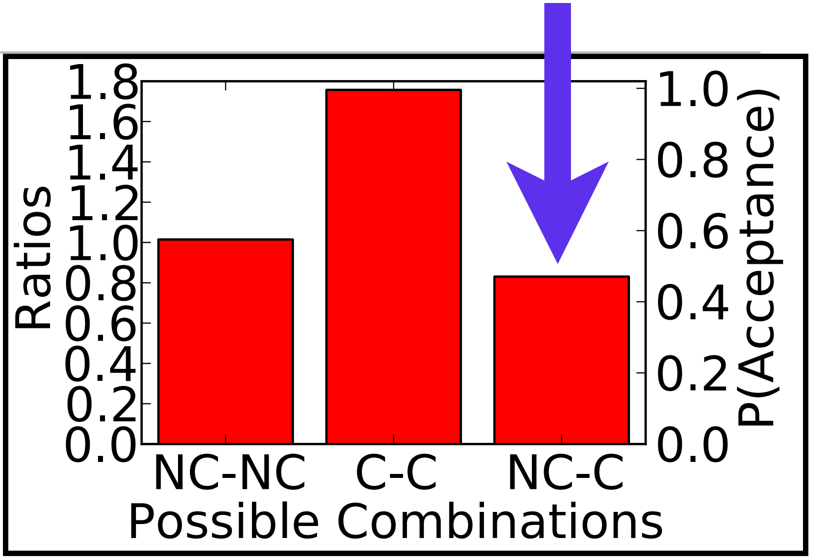


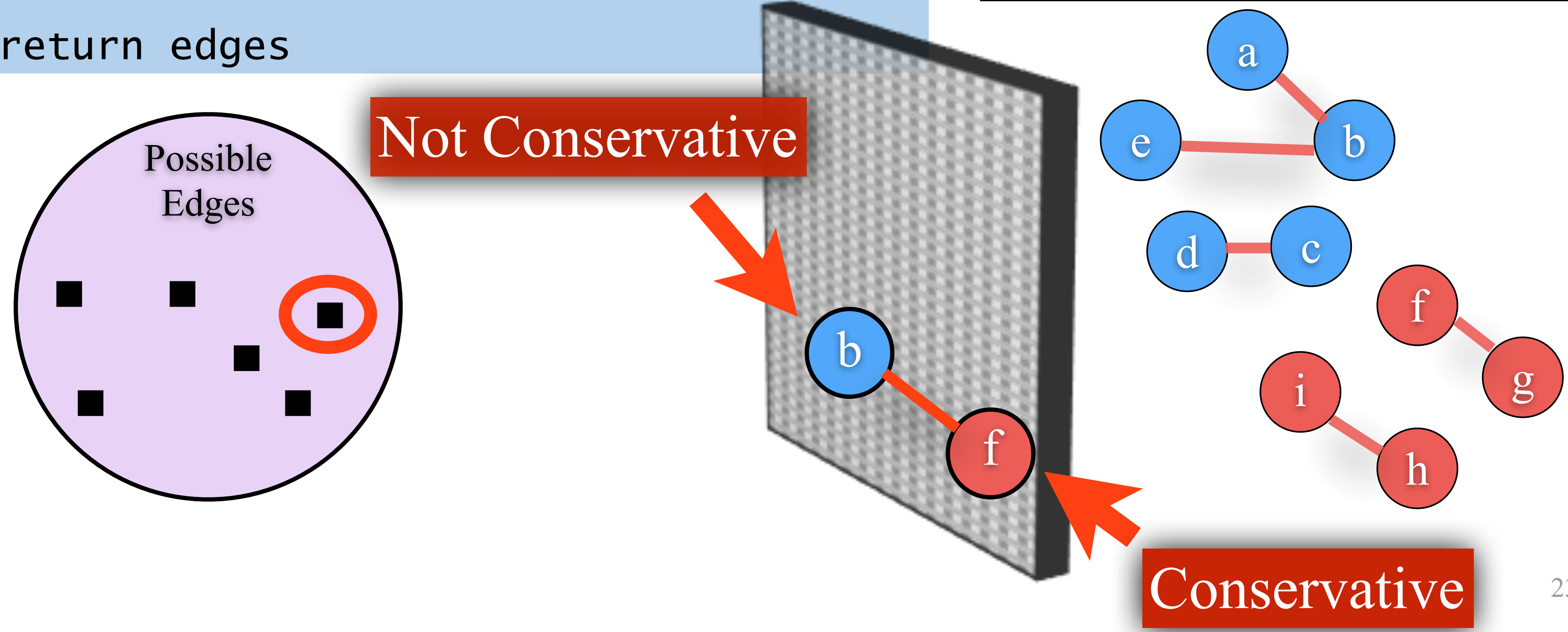
Not Conservative

Conservative

Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q' (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



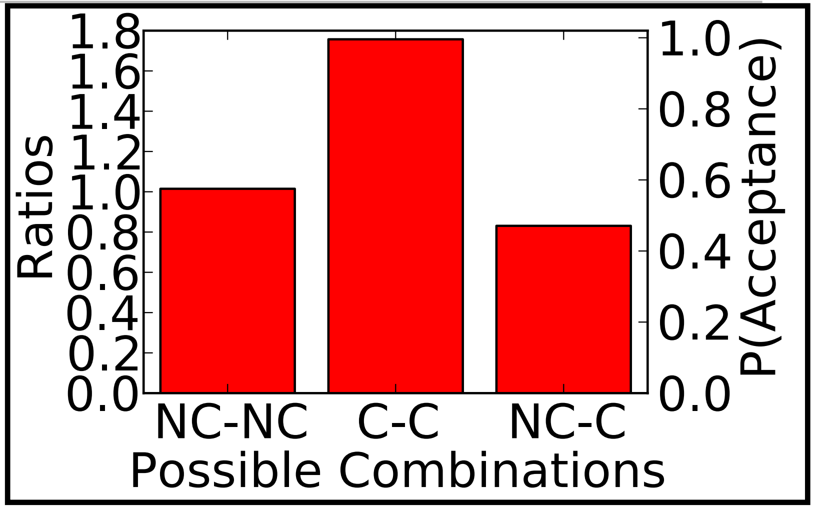


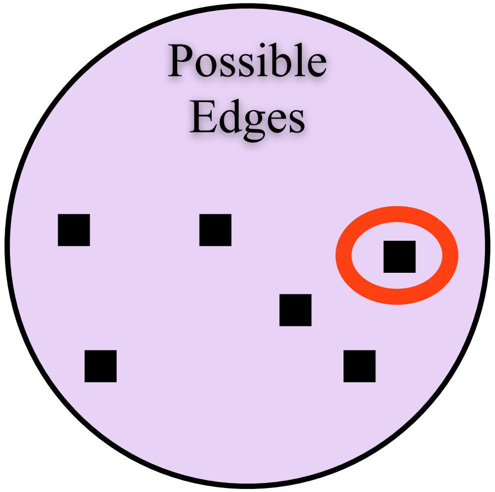
Not Conservative

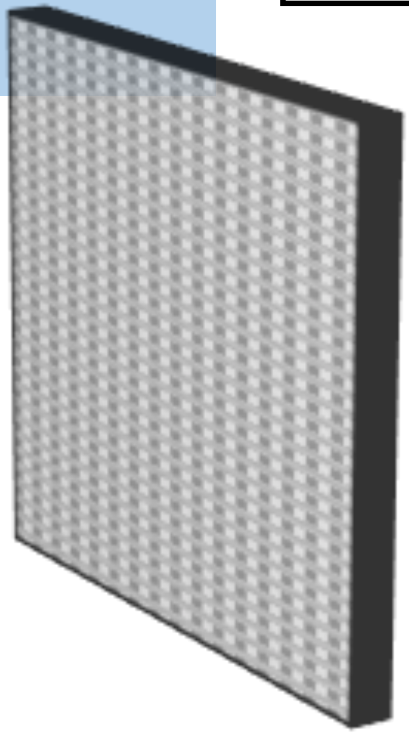
Conservative

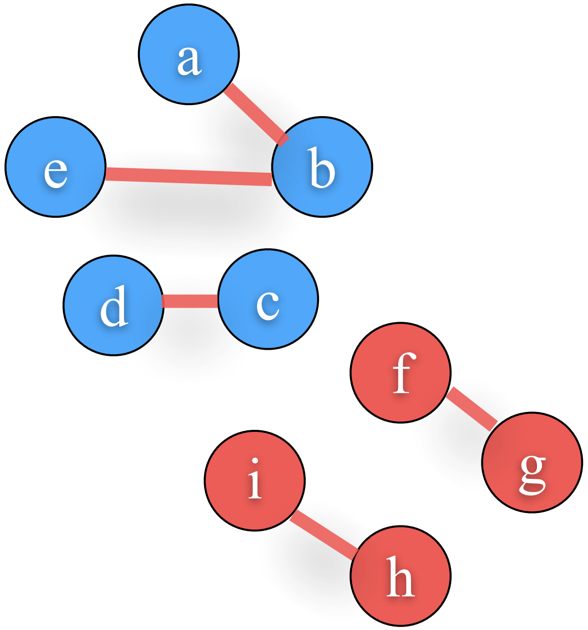
Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q' (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



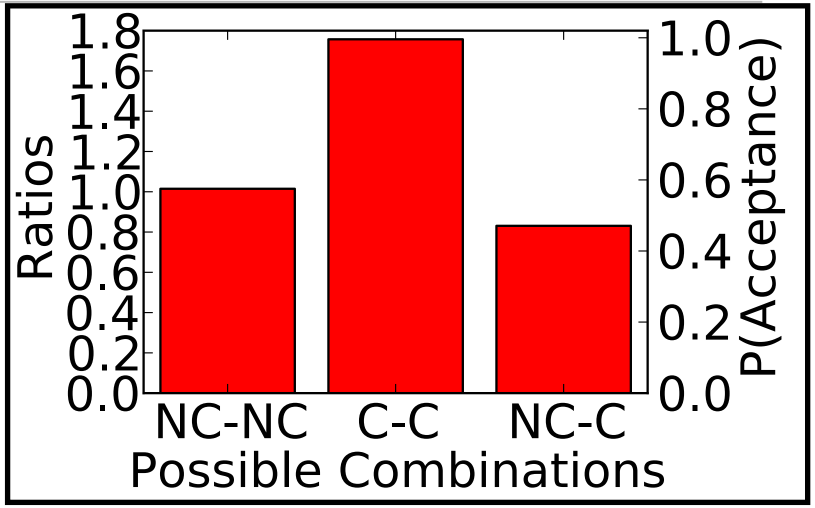


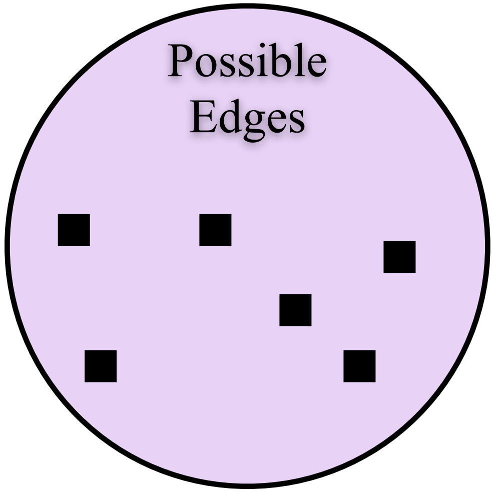


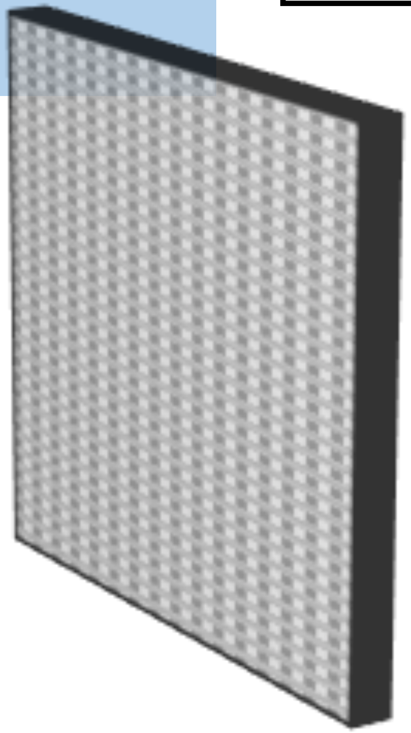


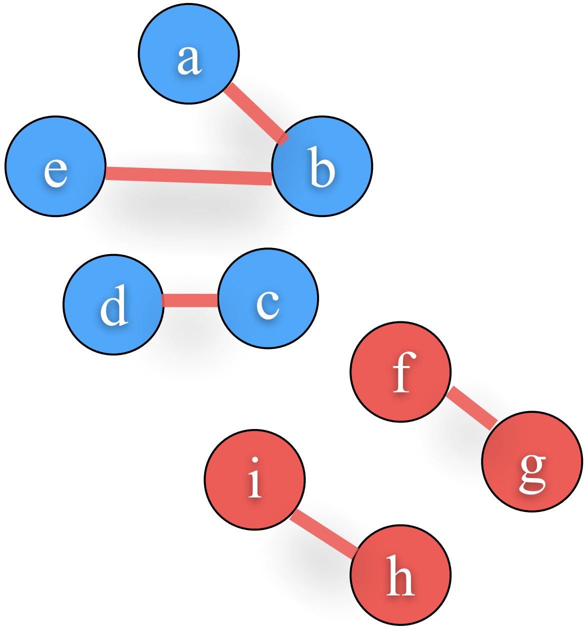
Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q' (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



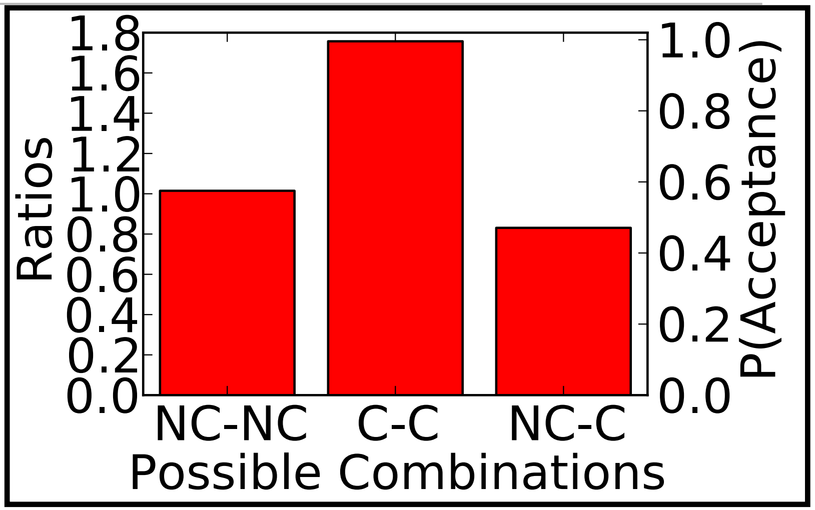


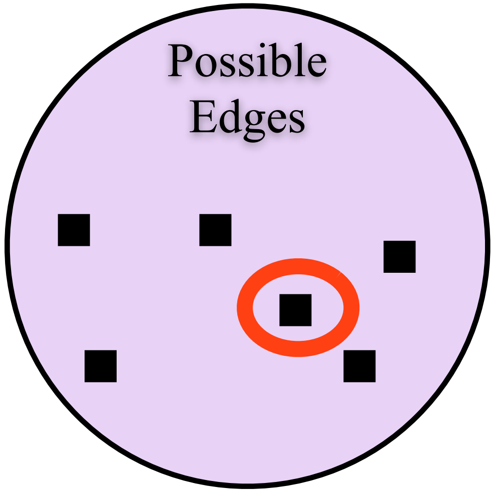


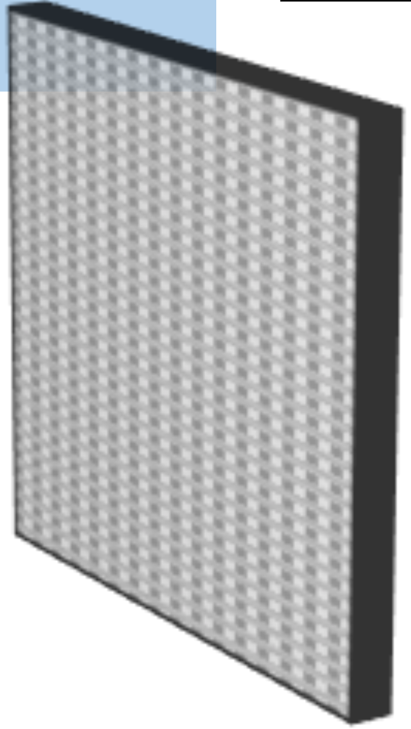


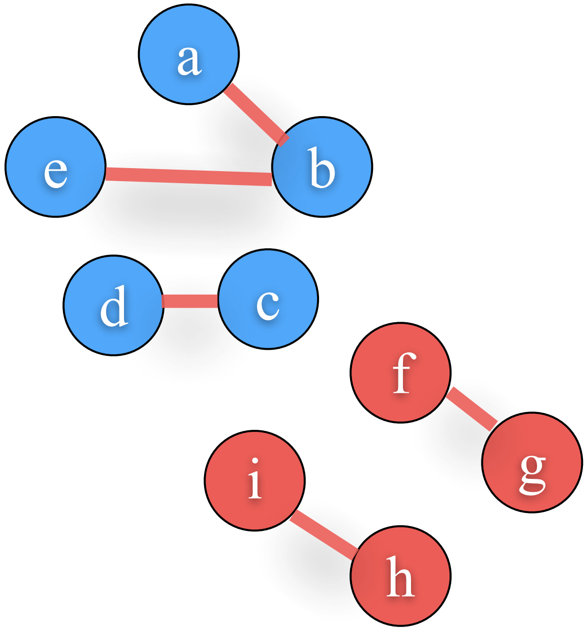
Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



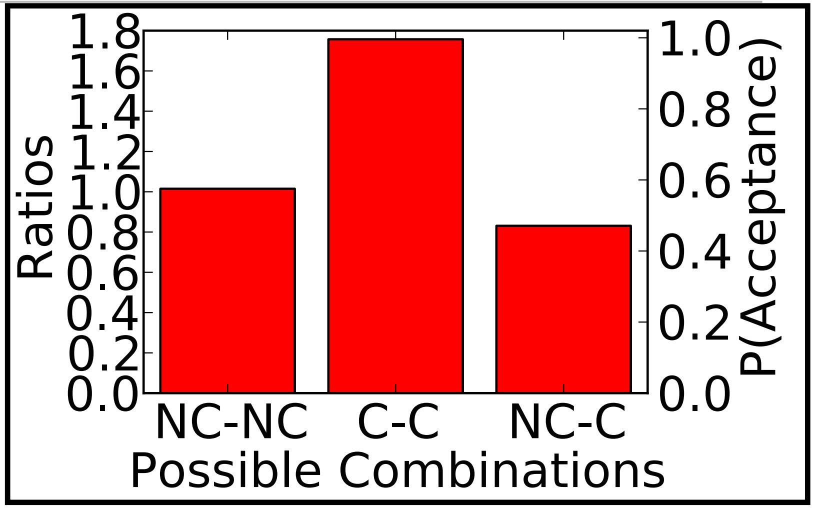


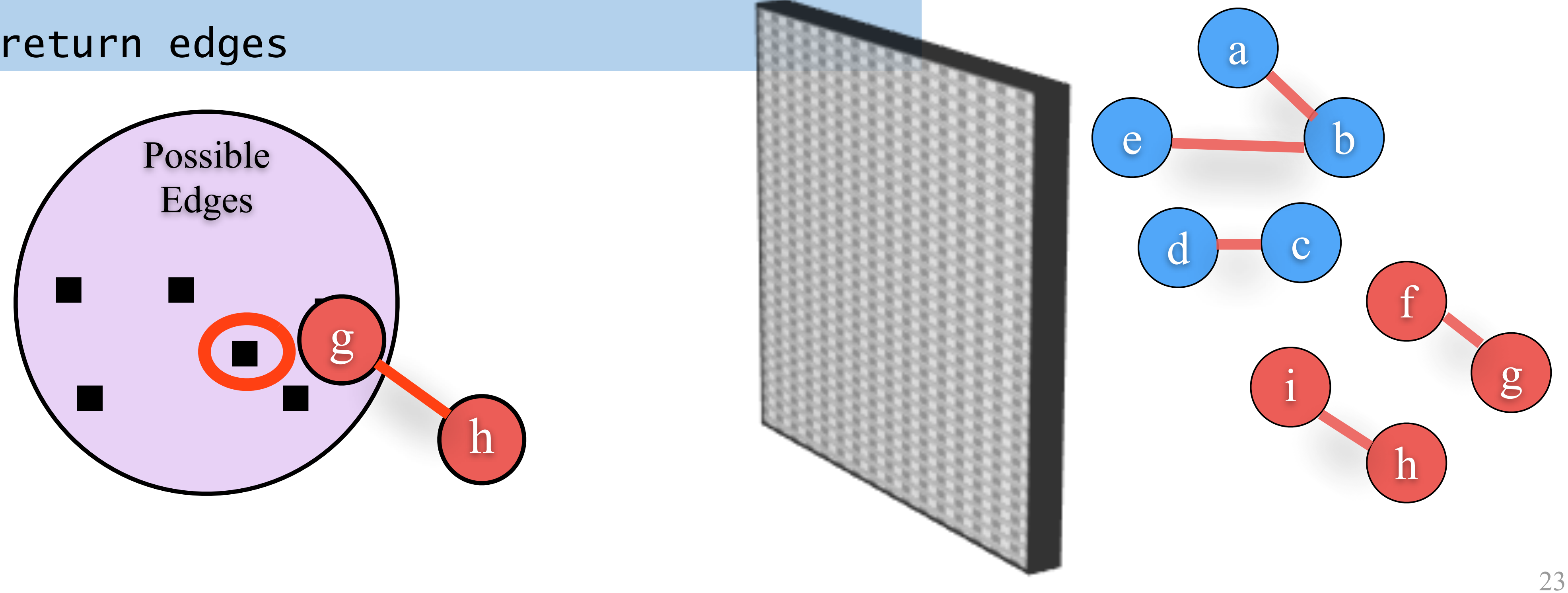




Attributed Graph Models

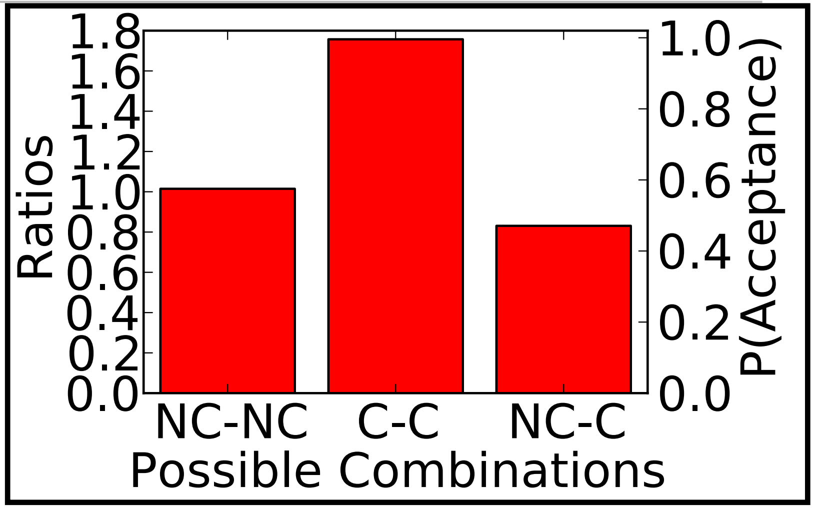
```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```

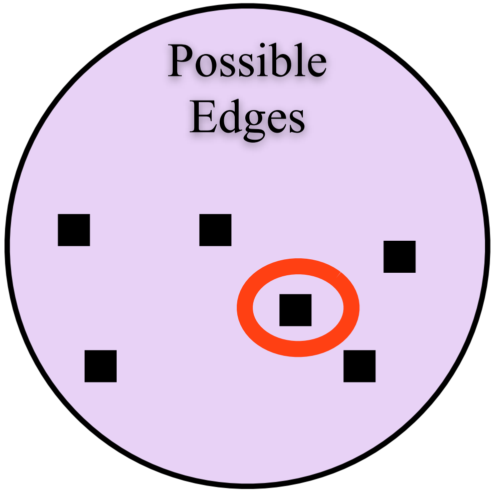


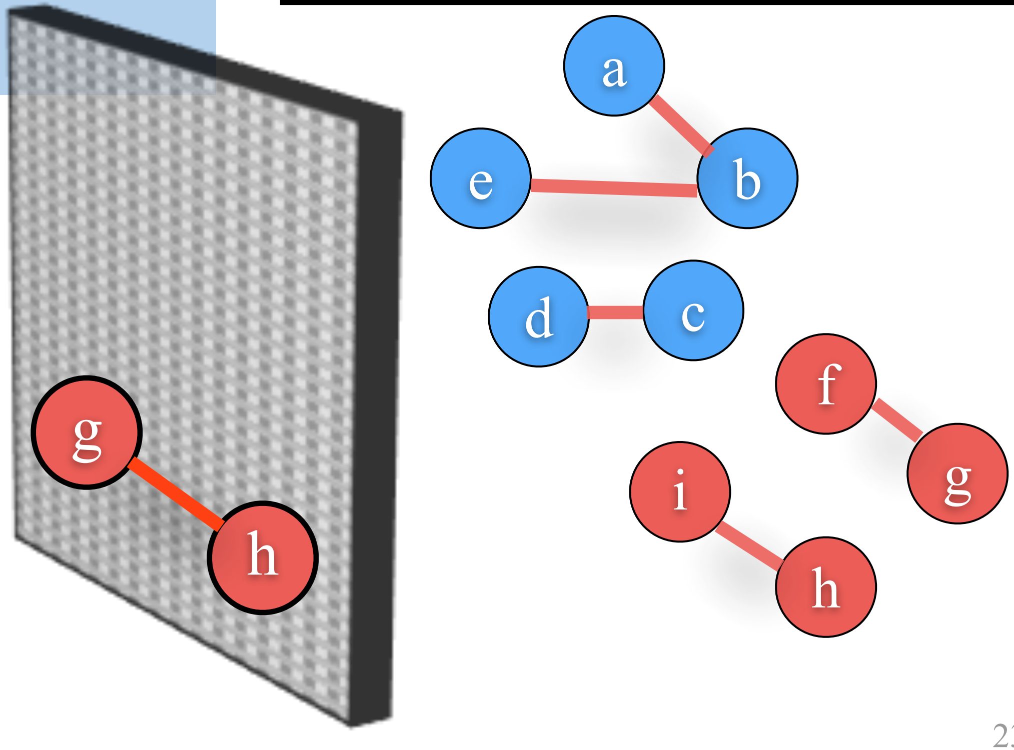


Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```

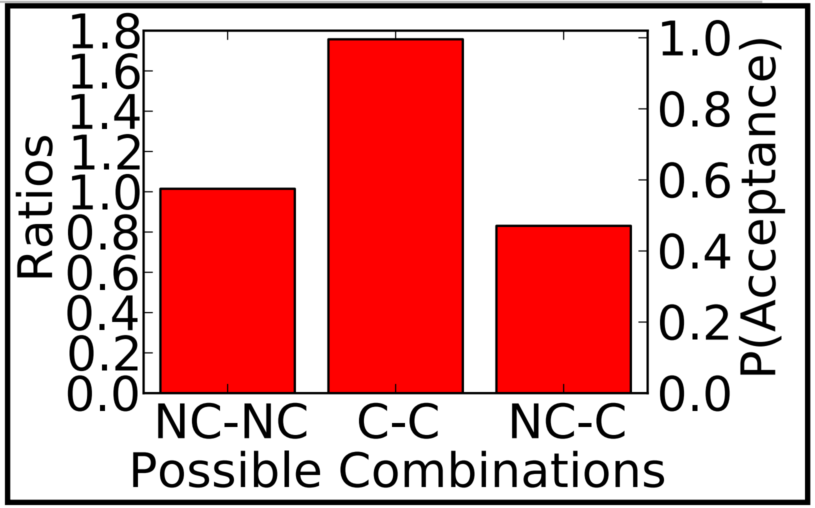


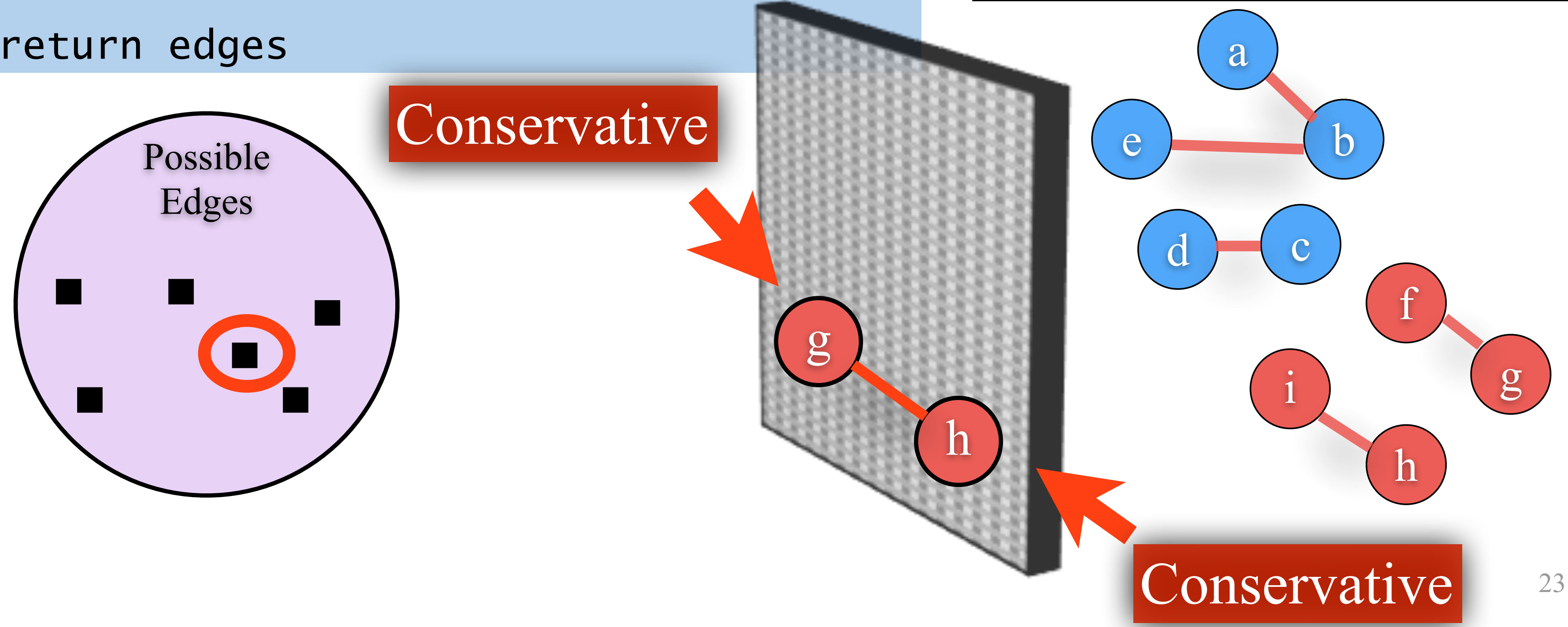




Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



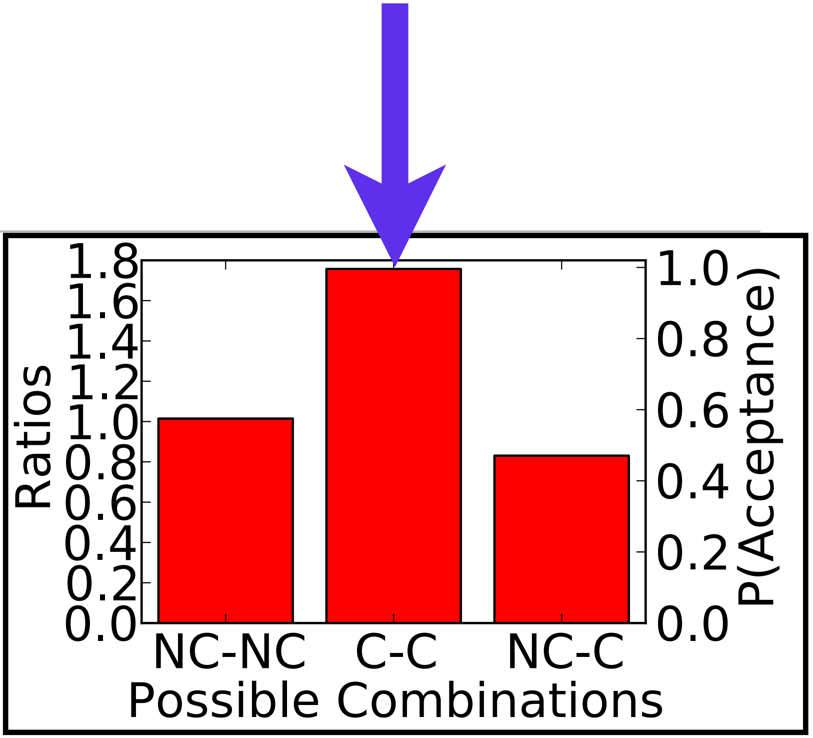


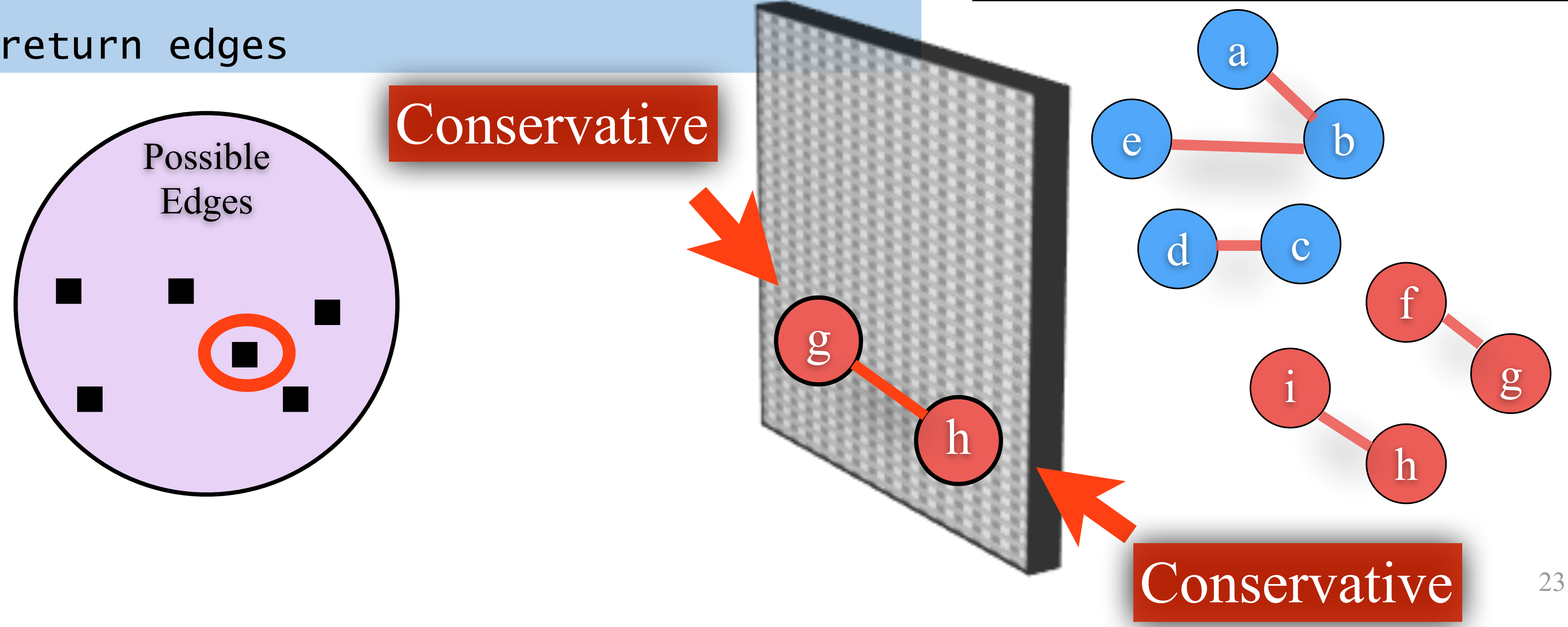
Conservative

Conservative

Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q' (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



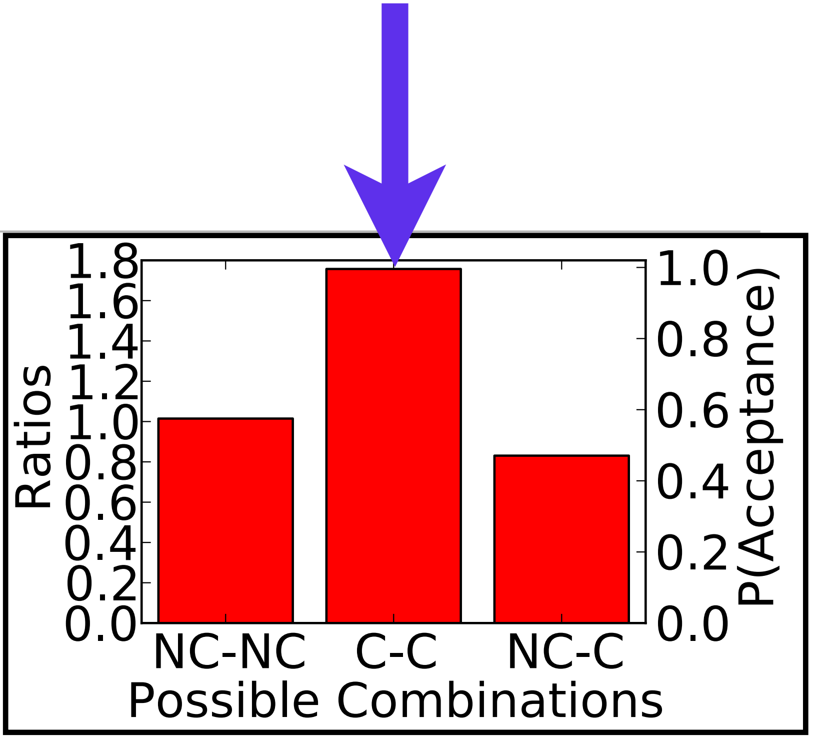


Conservative

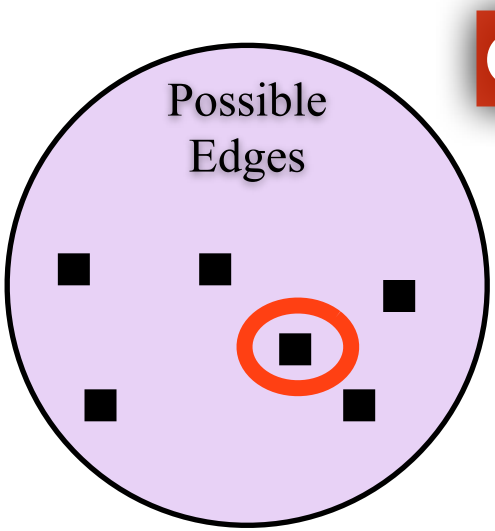
Conservative

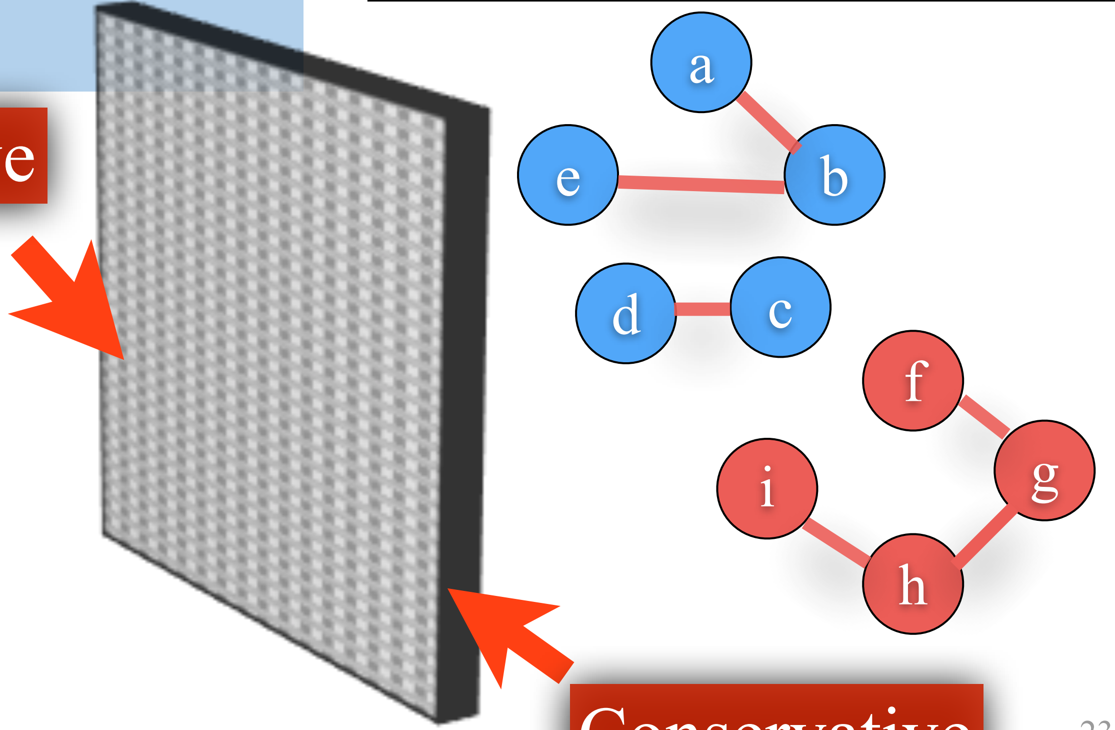
Attributed Graph Models

```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
  draw (vi,vj) from Q’ (the model)

  U ~ Uniform(0,1)
  if U < A(xi, xj)
    put (vi, vj) into the edges

return edges
```



Conservative

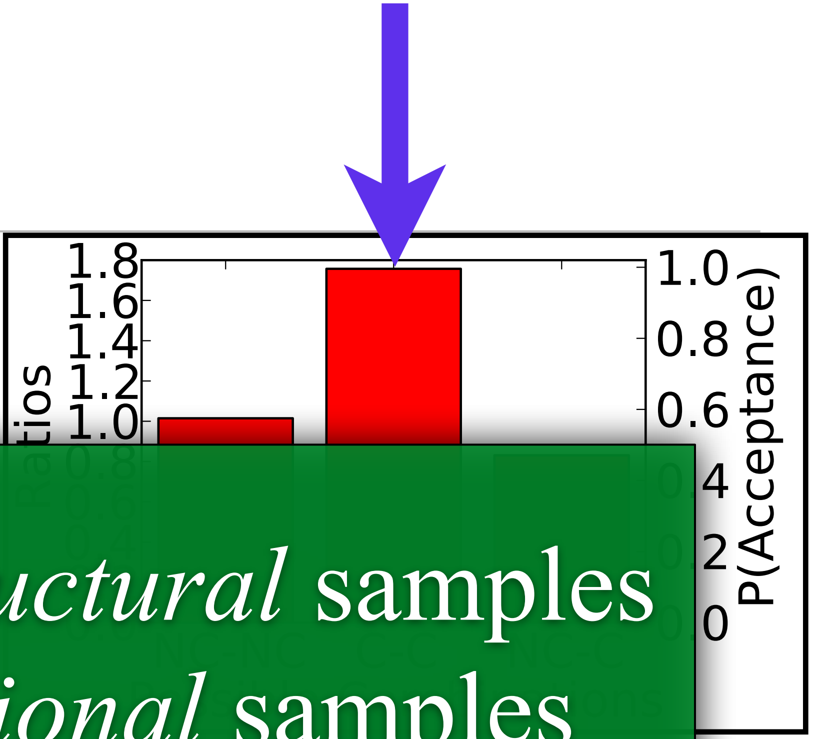




Conservative

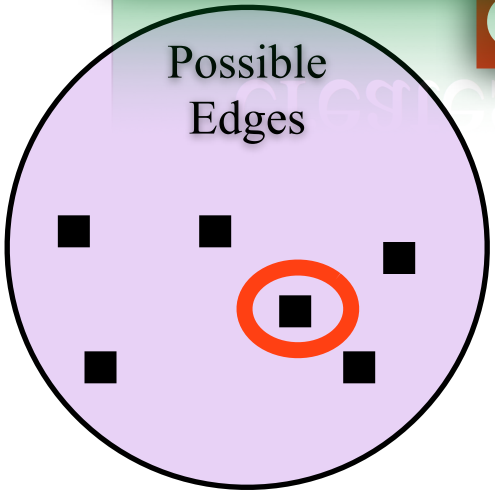
Attributed Graph Models

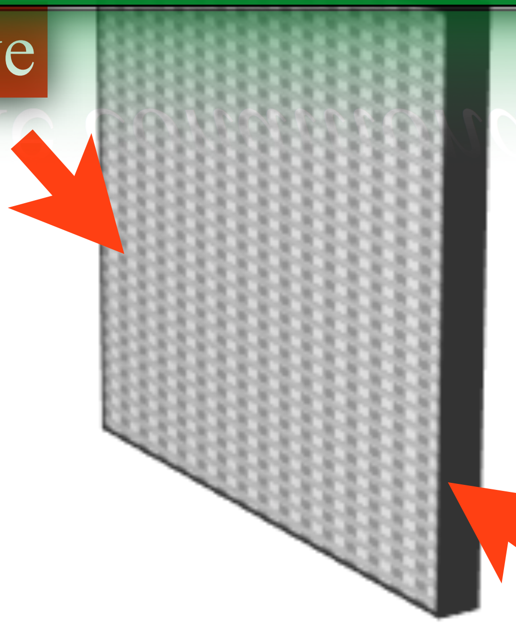
```
# … Learn parameters from network…
# … Sample attributes …

while not enough edges:
   draw (vi,vj) from Q' (the model)

   U ~
   if
      put

return edges
```

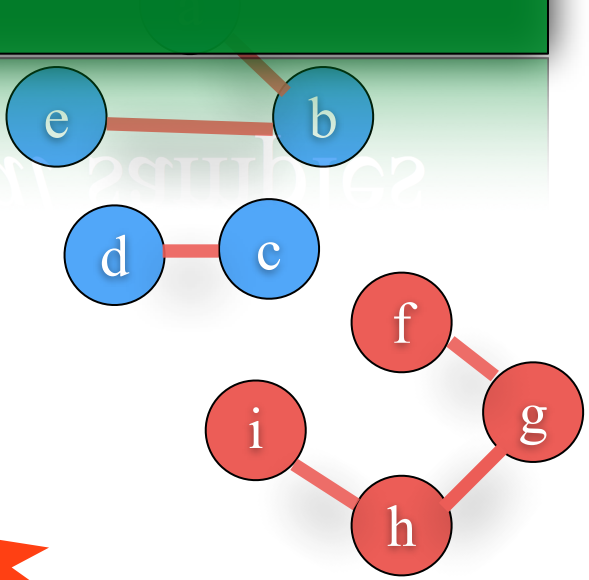


Filtering the scalable *structural* samples creates scalable *conditional* samples

Conservative



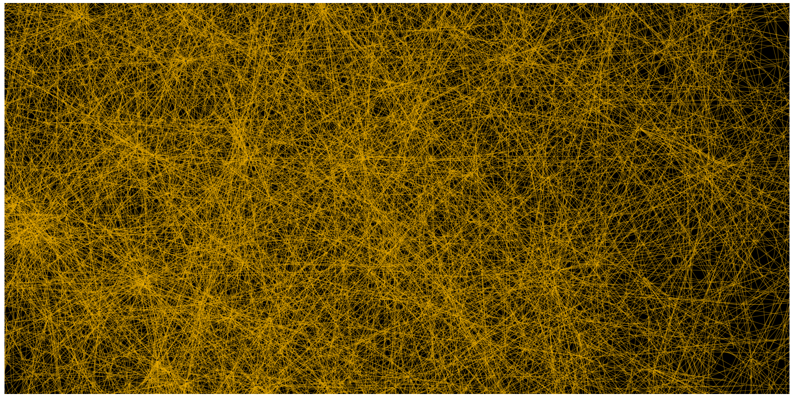


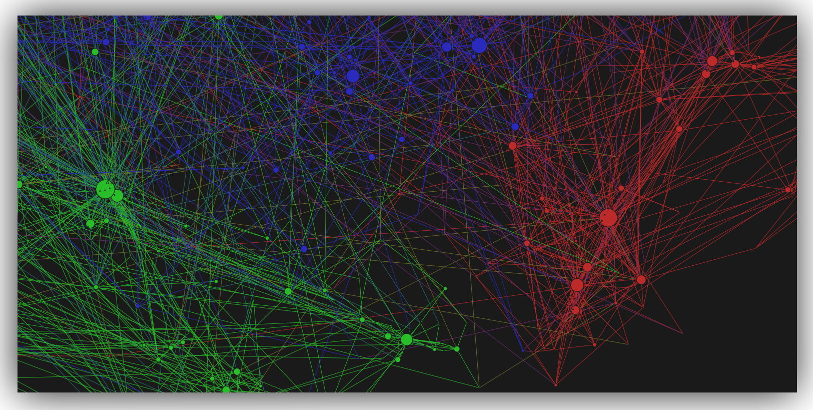


Conservative

Outline:

- Background
- Scalable Graph Sampling
- **Attributed Graph Models**
  - Sampling
  - **Theoretical Results**
  - Learning From Data
- Experiments
- Conclusions / Future Directions





Theorem 1: AGM samples from the joint distribution of edges and attributes

$$P(\mathbf{E}_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_F) P(\mathbf{x}_i, \mathbf{x}_j | \Theta_X)$$

Theorem 1: AGM samples from the joint distribution of edges and attributes

$$P(\mathbf{E}_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_F) P(\mathbf{x}_i, \mathbf{x}_j | \Theta_X)$$

Theorem 1: AGM samples from the joint distribution of edges and attributes

$$P(\mathbf{E}_{ij} = 1 | f(\mathbf{x}_i, \mathbf{x}_j), \Theta_{\mathcal{E}}, \Theta_F) P(\mathbf{x}_i, \mathbf{x}_j | \Theta_X)$$

Corollary 1: Expected Degree equals Expected Degree of structural model

When is AGM Scalable?

When is AGM Scalable?

- Given a structural model $\mathcal{E}$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$

$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}} \cdot \lambda) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:

$$O(\tau_{\mathcal{E}} + N_e \cdot \kappa_{\mathcal{E}} \cdot \lambda) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:

$$O(\underset{\uparrow}{\tau_{\mathcal{E}}} + N_e \cdot \kappa_{\mathcal{E}} \cdot \lambda) < O(N_v^2)$$

Construction

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

$$O(\tau_{\mathcal{E}}) < O(N_v^2)$$

$$O(\underset{\underset{\text{Construction}}{\uparrow}}{\tau_{\mathcal{E}}} + N_e \cdot \kappa_{\mathcal{E}} \cdot \lambda) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

$$O(\tau_{\mathcal{E}}) < O(N_v^2)$$

$$O(\underset{\underset{\text{Construction}}{\uparrow}}{\tau_{\mathcal{E}}} + N_e \cdot \underset{\underset{\text{Draw}}{\uparrow}}{\kappa_{\mathcal{E}}} \cdot \lambda) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

    $$O(\tau_{\mathcal{E}}) < O(N_v^2)$$

  - Draws are *sublinear*

    $$O(\kappa_{\mathcal{E}}) < O(N_v)$$

$$O(\underset{\underset{\text{Construction}}{\uparrow}}{\tau_{\mathcal{E}}} + N_e \cdot \underset{\underset{\text{Draw}}{\uparrow}}{\kappa_{\mathcal{E}}} \cdot \lambda) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

    $$O(\tau_{\mathcal{E}}) < O(N_v^2)$$

  - Draws are *sublinear*

    $$O(\kappa_{\mathcal{E}}) < O(N_v)$$

- Constraints on λ

$$O(\underset{\substack{\Big\uparrow \\ \text{Construction}}}{\tau_{\mathcal{E}}} + N_e \cdot \underset{\substack{\Big\uparrow \\ \text{Draw}}}{\kappa_{\mathcal{E}}} \cdot \lambda) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

    $$O(\tau_{\mathcal{E}}) < O(N_v^2)$$

  - Draws are *sublinear*

    $$O(\kappa_{\mathcal{E}}) < O(N_v)$$

- Constraints on λ

$$O(\underset{\substack{\uparrow \\ \text{Construction}}}{\tau_{\mathcal{E}}} + N_e \cdot \underset{\substack{\uparrow \\ \text{Draw}}}{\kappa_{\mathcal{E}}} \cdot \overset{\substack{\text{Ratio} \\ \swarrow}}{\lambda}) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

    $$O(\tau_{\mathcal{E}}) < O(N_v^2)$$

  - Draws are *sublinear*

    $$O(\kappa_{\mathcal{E}}) < O(N_v)$$

- Constraints on λ
  - Must be sublinear

    $$O(\lambda) < O(N_v)$$

Ratio

$$O(\underset{\uparrow}{\tau_{\mathcal{E}}} + N_e \cdot \underset{\uparrow}{\kappa_{\mathcal{E}}} \cdot \overset{\swarrow}{\lambda}) < O(N_v^2)$$

Construction   Draw

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

    $$O(\tau_{\mathcal{E}}) < O(N_v^2)$$
  - Draws are *sublinear*

    $$O(\kappa_{\mathcal{E}}) < O(N_v)$$
- Constraints on λ
  - Must be sublinear

    $$O(\lambda) < O(N_v)$$
- Constraint on $\kappa_{\mathcal{E}} \cdot \lambda$

$$O(\underset{\overset{\uparrow}{\text{Construction}}}{\tau_{\mathcal{E}}} + N_e \cdot \underset{\overset{\uparrow}{\text{Draw}}}{\kappa_{\mathcal{E}}} \cdot \overset{\overset{\text{Ratio}}{\swarrow}}{\lambda}) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

    $$O(\tau_{\mathcal{E}}) < O(N_v^2)$$

  - Draws are *sublinear*

    $$O(\kappa_{\mathcal{E}}) < O(N_v)$$

- Constraints on λ
  - Must be sublinear

    $$O(\lambda) < O(N_v)$$

- Constraint on $\kappa_{\mathcal{E}} \cdot \lambda$
  - Must be sublinear

    $$O(\kappa_{\mathcal{E}} \cdot \lambda) < O(N_v)$$

$$O(\underset{\overset{\uparrow}{\text{Construction}}}{\tau_{\mathcal{E}}} + N_e \cdot \underset{\overset{\uparrow}{\text{Draw}}}{\kappa_{\mathcal{E}}} \cdot \overset{\overset{\text{Ratio}}{\swarrow}}{\lambda}) < O(N_v^2)$$

When is AGM Scalable?

- Given a structural model $\mathcal{E}$
- Q' (defined by $\mathcal{E}$) must have:
  - Construction is *subquadratic*

    $$O(\tau_{\mathcal{E}}) < O(N_v^2)$$

  - Draws are *sublinear*

    $$O(\kappa_{\mathcal{E}}) < O(N_v)$$

- Constraints on λ
  - Must be sublinear

    $$O(\lambda) < O(N_v)$$

- Constraint on $\kappa_{\mathcal{E}} \cdot \lambda$
  - Must be sublinear

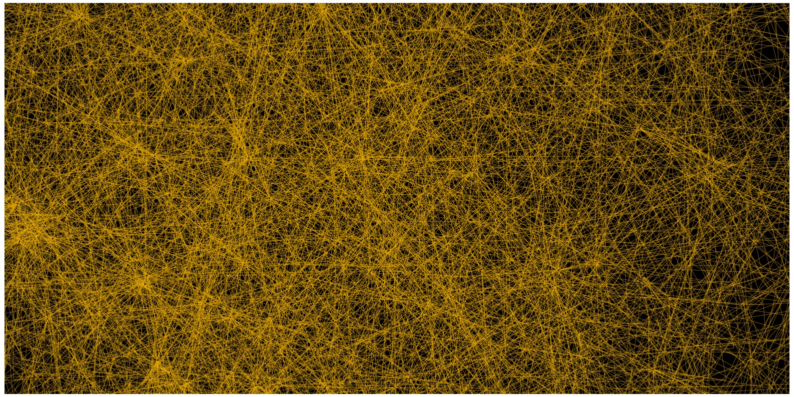
    $$O(\kappa_{\mathcal{E}} \cdot \lambda) < O(N_v)$$

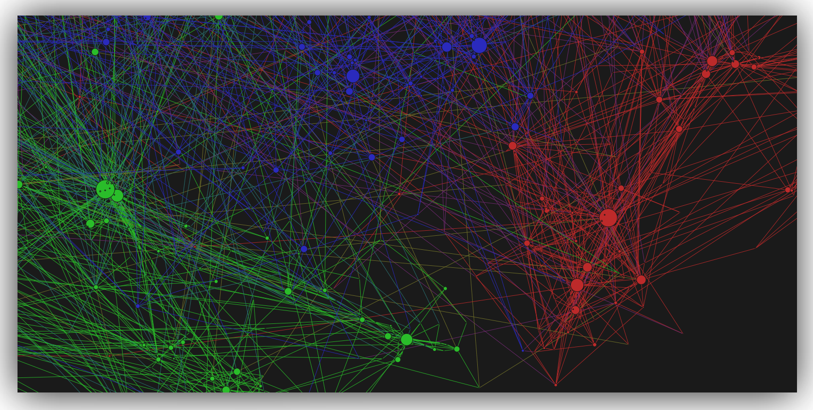
Model	$\tau_{\mathcal{E}}$	$\kappa_{\mathcal{E}}$
FCL	$O(N_e)$	$O(1)$
TCL	$O(N_e)$	$O(\log d)$
KPGM	$O(1)$	$O(\log N_v)$

$$O(\underset{\text{Construction}}{\underset{\uparrow}{\tau_{\mathcal{E}}}} + N_e \cdot \underset{\text{Draw}}{\underset{\uparrow}{\kappa_{\mathcal{E}}}} \cdot \overset{\text{Ratio}}{\overset{\swarrow}{\lambda}}) < O(N_v^2)$$

Outline:

- Background
- Scalable Graph Sampling
- **Attributed Graph Models**
  - Sampling
  - Theoretical Results
  - **Learning From Data**
- Experiments
- Conclusions /
  Future Directions





Learning

Learning

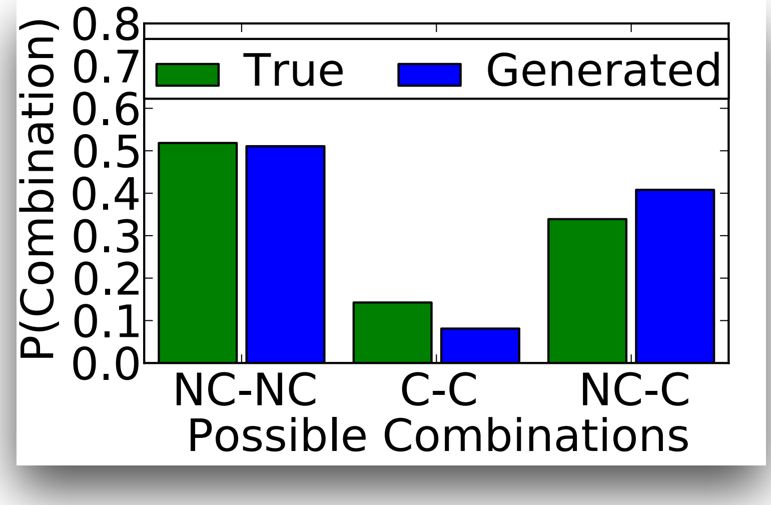
- Given a network, learn

$$P(f(\mathbf{x}_i, \mathbf{x}_j)|E_{ij} = 1, \Theta_{\mathcal{E}}, \Theta_F)$$

Learning

- Given a network, learn

$$P(f(\mathbf{x}_i, \mathbf{x}_j)|E_{ij} = 1, \Theta_{\mathcal{E}}, \Theta_F)$$

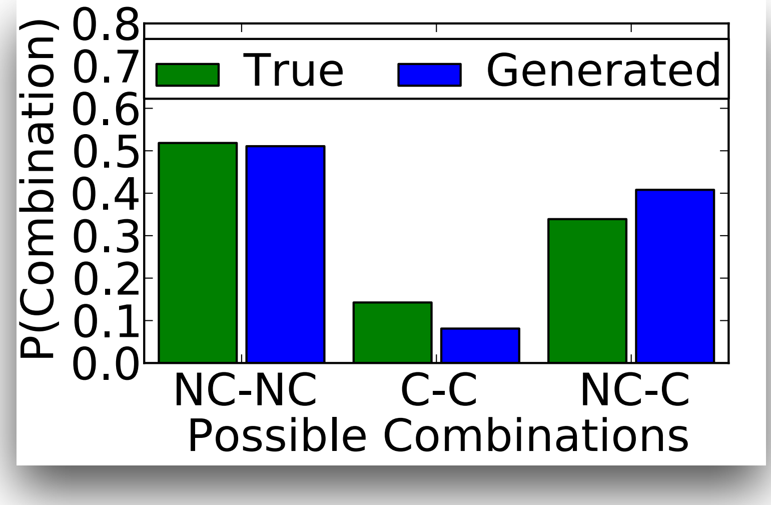


Learning

- Given a network, learn

$$P(f(\mathbf{x}_i, \mathbf{x}_j)|E_{ij} = 1, \Theta_{\mathcal{E}}, \Theta_F)$$

  - Observed network and the model

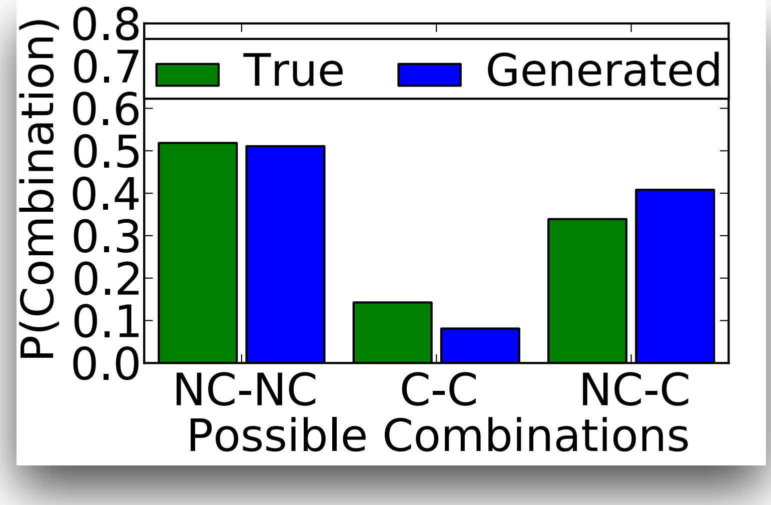


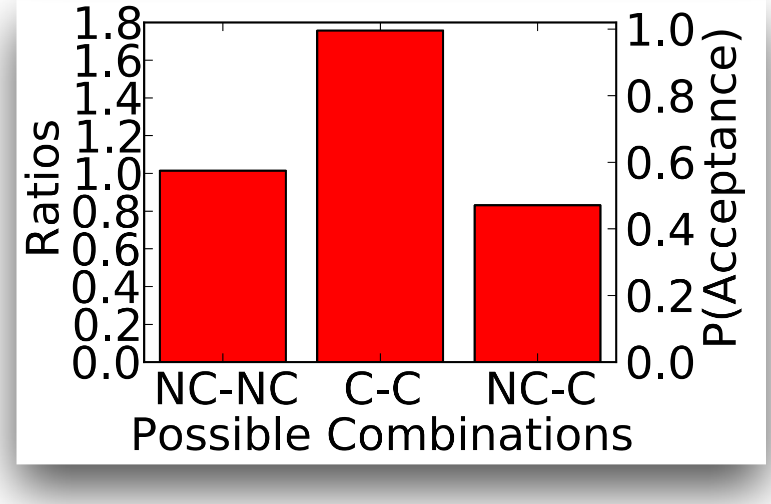
Learning

- Given a network, learn

$$P(f(\mathbf{x}_i, \mathbf{x}_j) | E_{ij} = 1, \Theta_{\mathcal{E}}, \Theta_F)$$

  - Observed network and the model



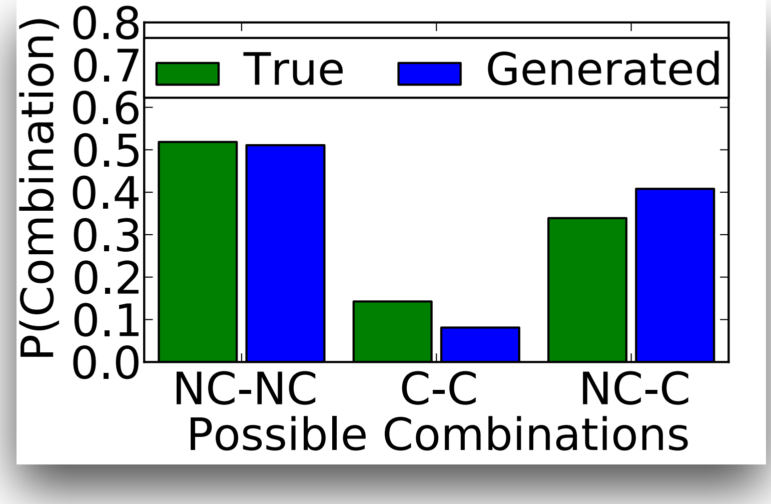


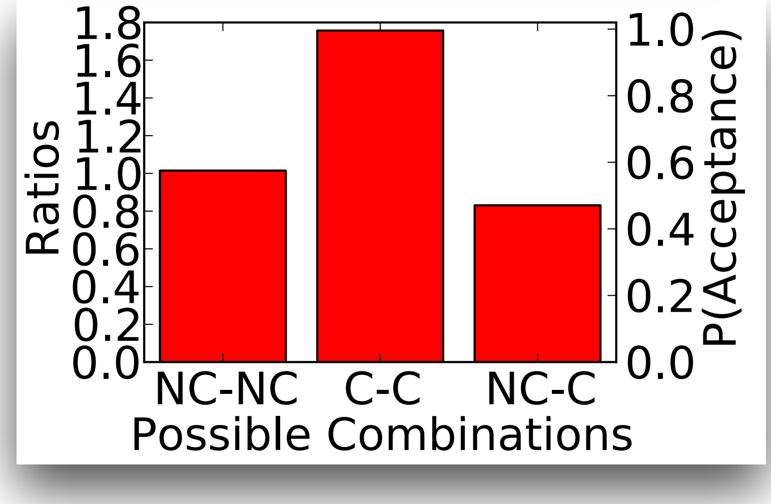
Learning

- Given a network, learn

$$P(f(\mathbf{x}_i, \mathbf{x}_j)|E_{ij} = 1, \Theta_{\mathcal{E}}, \Theta_F)$$

  - Observed network and the model
- Maximum Likelihood Estimation
  - Single feature: count instances



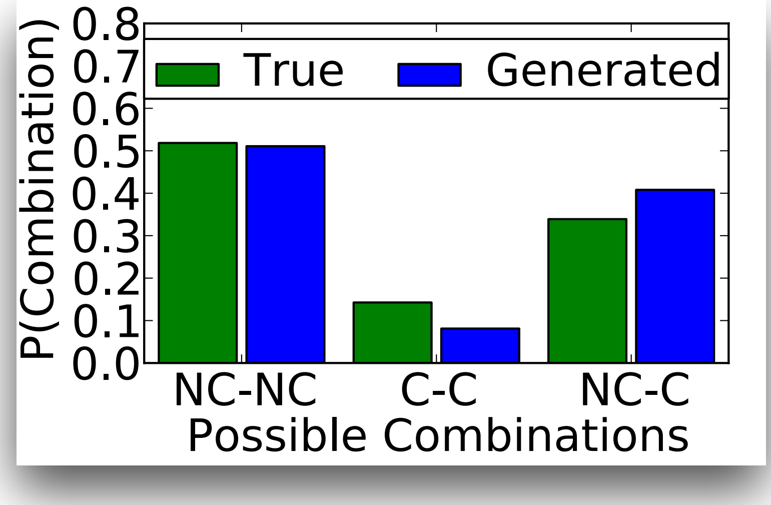


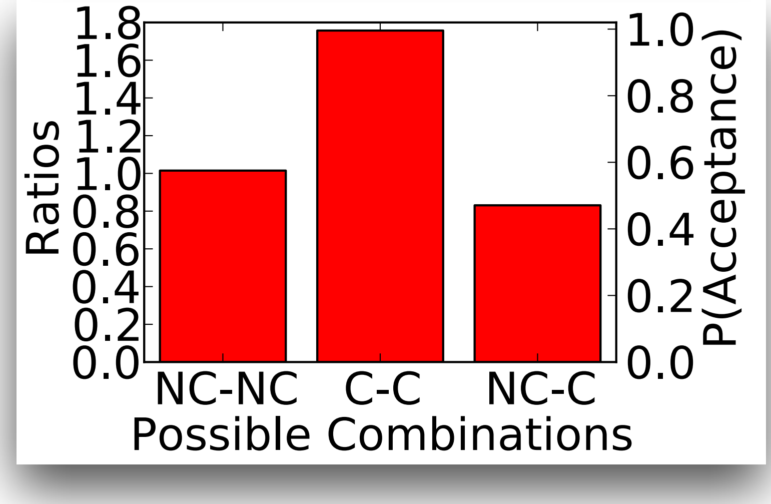
Learning

- Given a network, learn

$$P(f(\mathbf{x}_i, \mathbf{x}_j) | E_{ij} = 1, \Theta_{\mathcal{E}}, \Theta_F)$$

  - Observed network and the model
- Maximum Likelihood Estimation
  - Single feature: count instances
- Observed Graph: Estimate directly



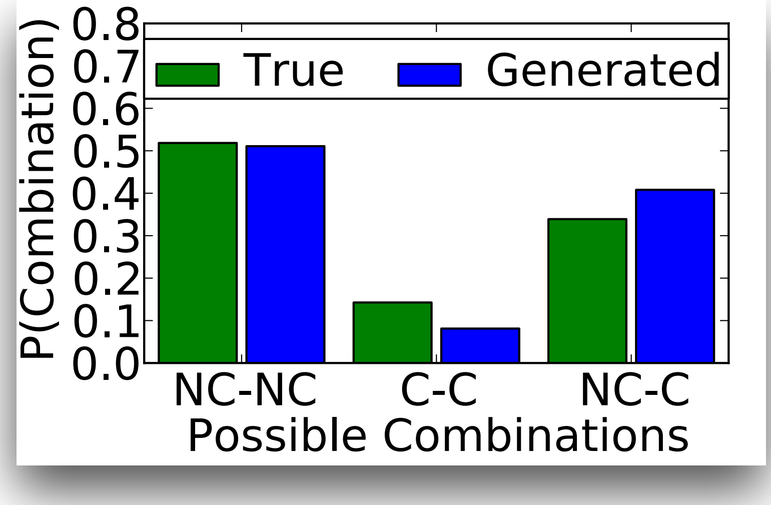


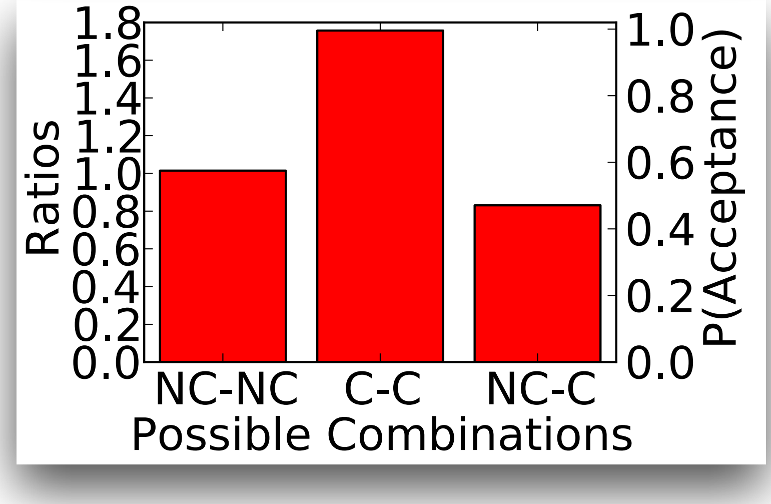
Learning

- Given a network, learn

$$P(f(\mathbf{x}_i, \mathbf{x}_j)|E_{ij} = 1, \Theta_{\mathcal{E}}, \Theta_F)$$

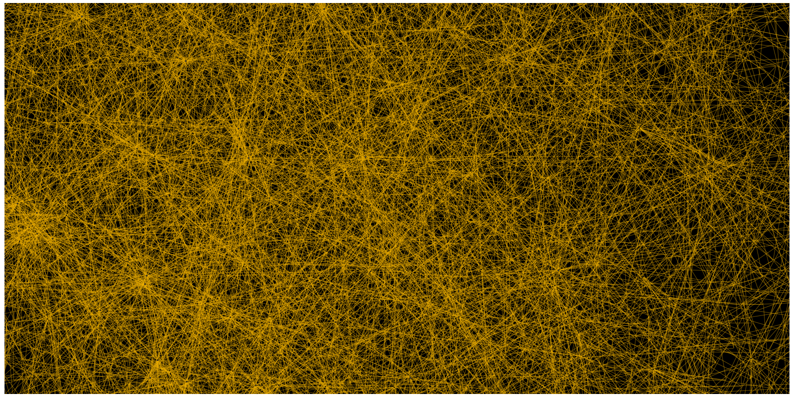
  - Observed network and the model
- Maximum Likelihood Estimation
  - Single feature: count instances
- Observed Graph: Estimate directly
- Model: Draw sample graph

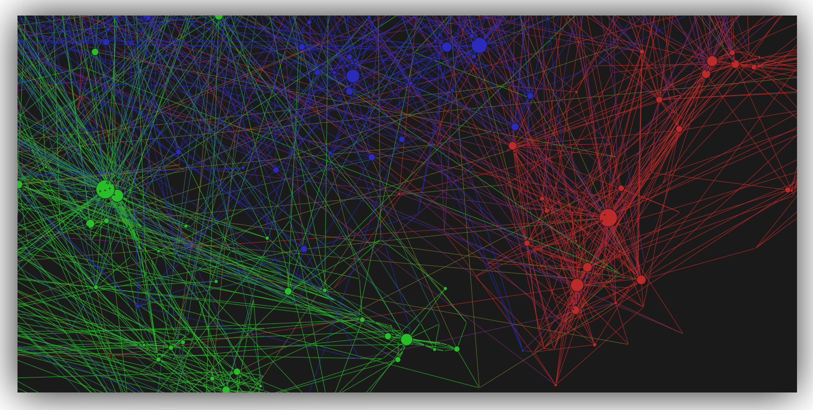




Outline:

- Background
- Scalable Graph Sampling
- Attributed Graph Models
  - Sampling
  - Theoretical Results
  - Learning From Data
- **Experiments**
- Conclusions /
  Future Directions





Evaluation - Models and Data

Evaluation - Models and Data

- Compare 4 Generative Graph Models with and without AGM

Evaluation - Models and Data

- Compare 4 Generative Graph Models with and without AGM

Original Model	AGM Model
FCL	AGM-FCL
TCL	AGM-TCL
KPGM	AGM-KPGM
KPGM	AGM-KPGM

Evaluation - Models and Data

- Compare 4 Generative Graph Models with and without AGM
- Two large attributed networks
  - CoRA and Facebook

Original Model	AGM Model
FCL	AGM-FCL
TCL	AGM-TCL
KPGM	AGM-KPGM
KPGM	AGM-KPGM

Evaluation - Models and Data

- Compare 4 Generative Graph Models with and without AGM
- Two large attributed networks
  - CoRA and Facebook

Original Model	AGM Model
FCL	AGM-FCL
TCL	AGM-TCL
KPGM	AGM-KPGM
KPGM	AGM-KPGM

Name	Nodes	Edges	**Features**
CoRA	11,881	31,482	1
Facebook	449,748	1,016,621	2

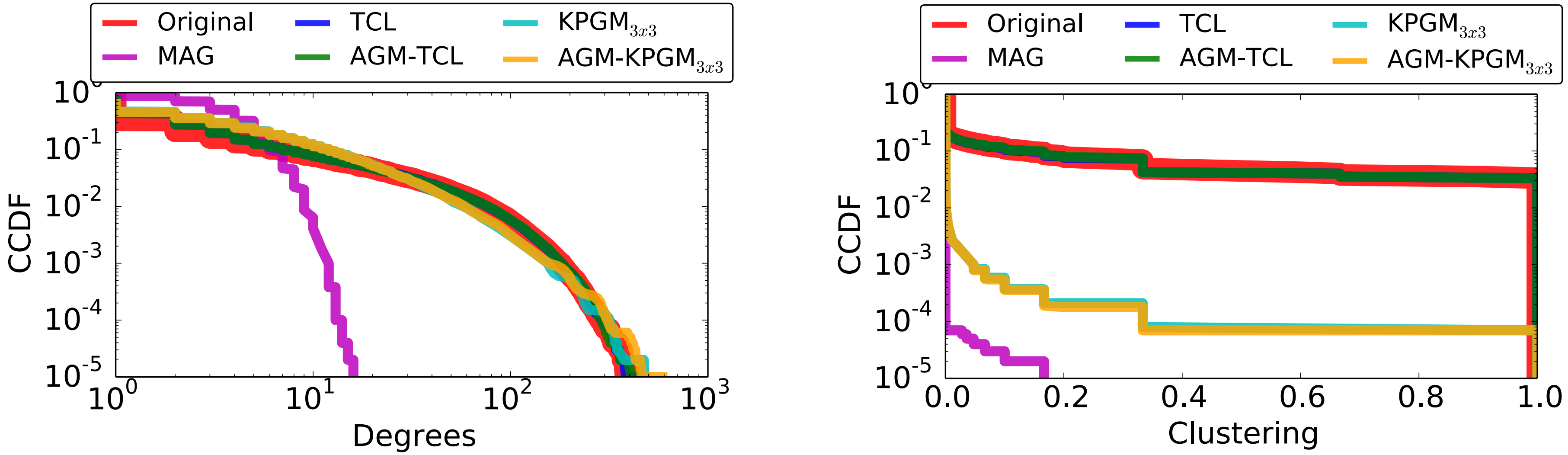
Evaluation - Models and Data

- Compare 4 Generative Graph Models with and without AGM
- Two large attributed networks
  - CoRA and Facebook
- Measured structural features and attribute correlations

Original Model	AGM Model
FCL	AGM-FCL
TCL	AGM-TCL
KPGM	AGM-KPGM
KPGM	AGM-KPGM

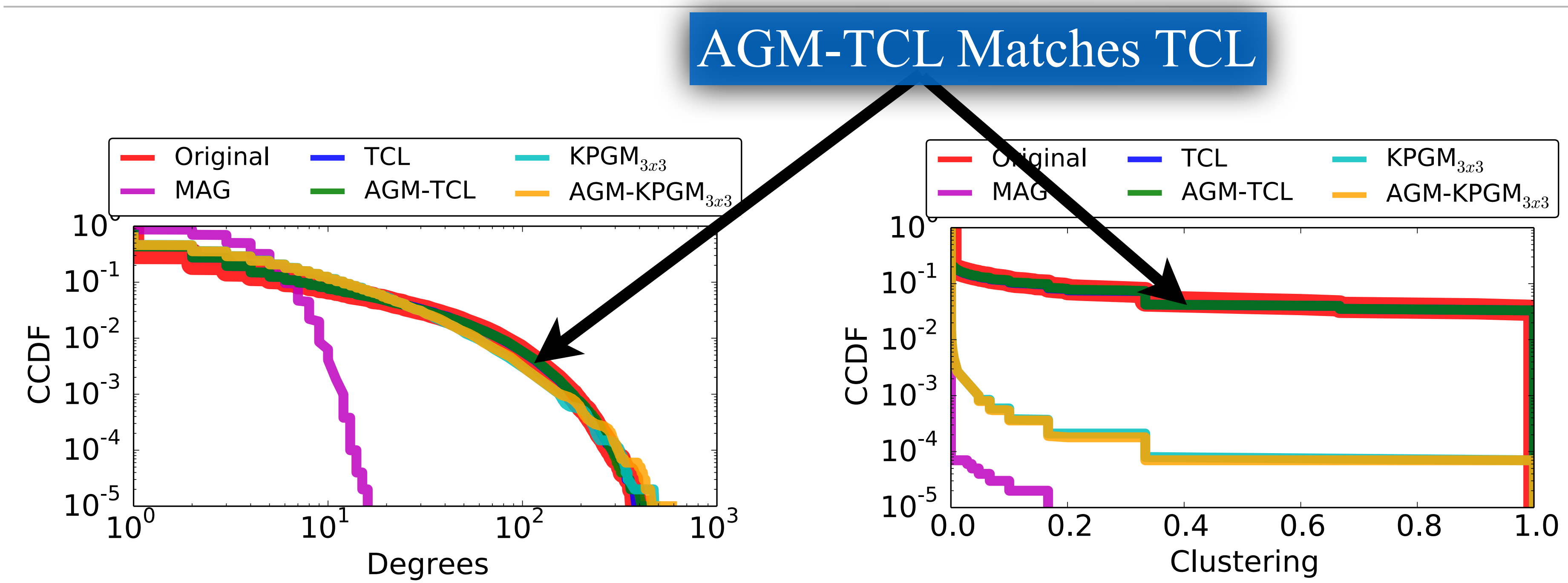
Name	Nodes	Edges	**Features**
CoRA	11,881	31,482	1
Facebook	449,748	1,016,621	2

Structural Features



Facebook

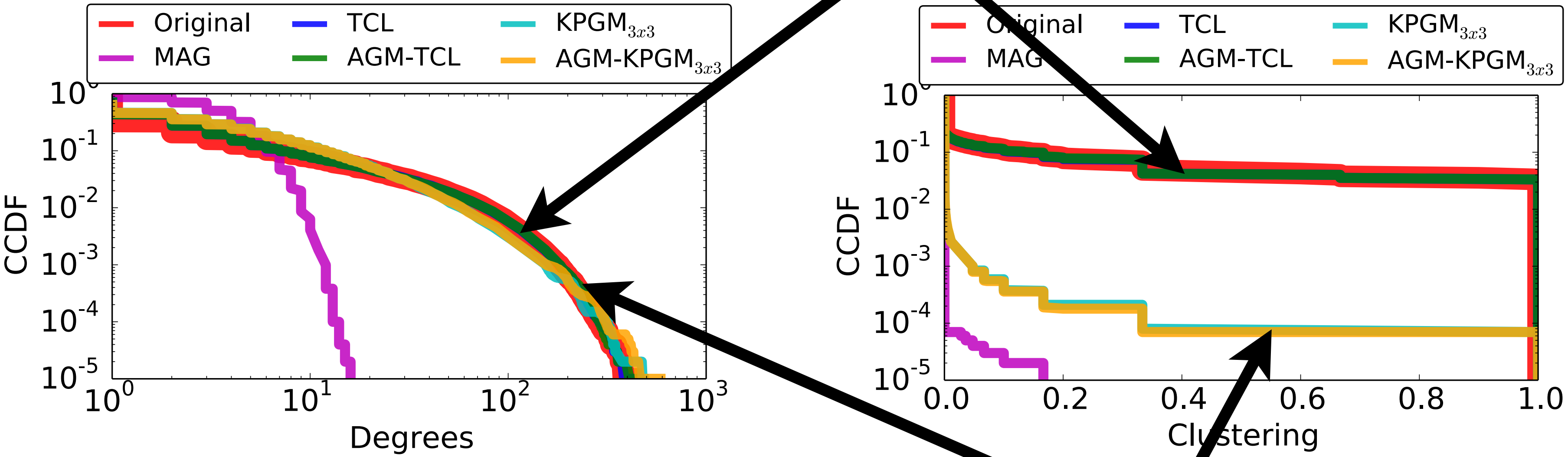
Structural Features


AGM-TCL Matches TCL

Facebook

Structural Features

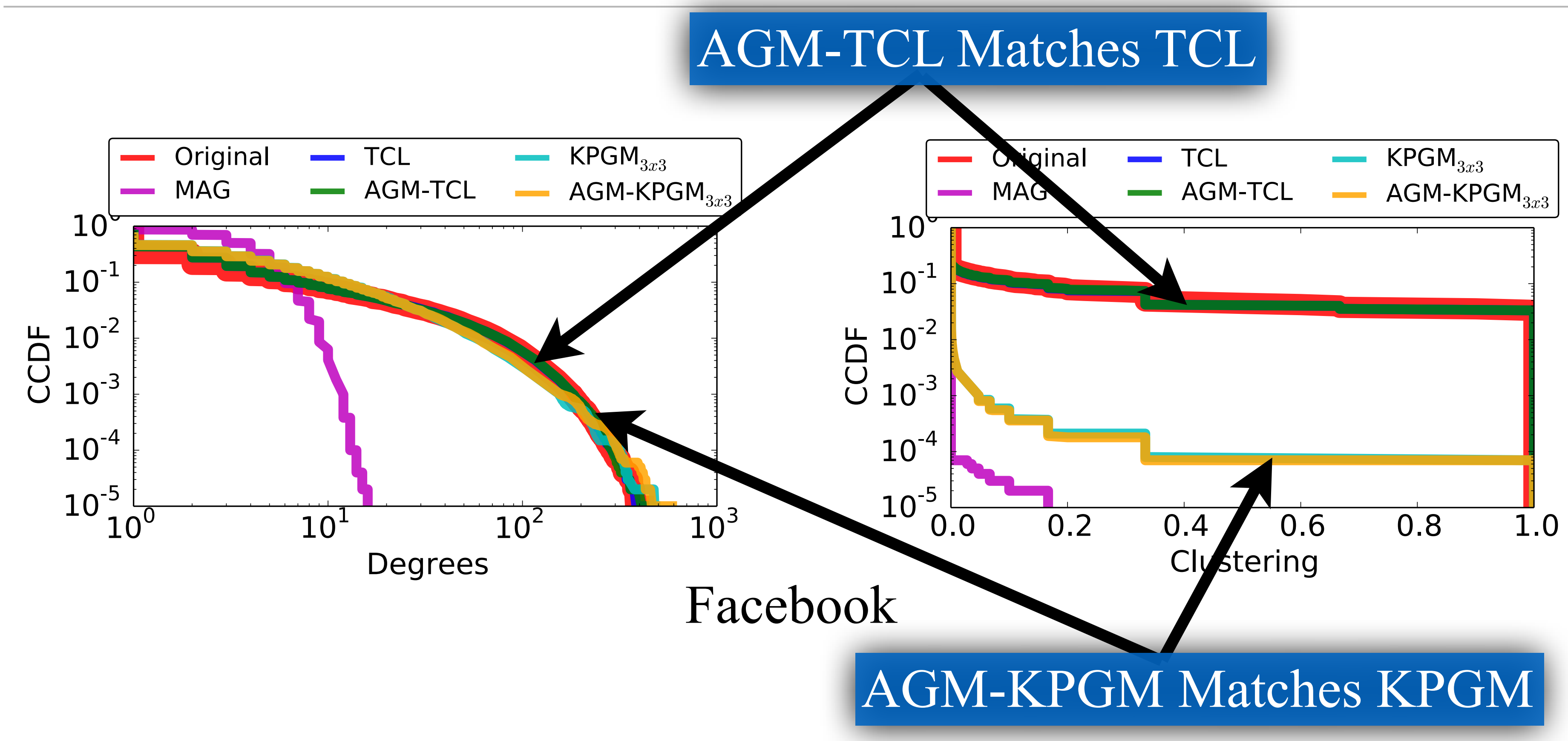
AGM-TCL Matches TCL



Facebook

AGM-KPGM Matches KPGM

Structural Features



AGM-TCL Matches TCL

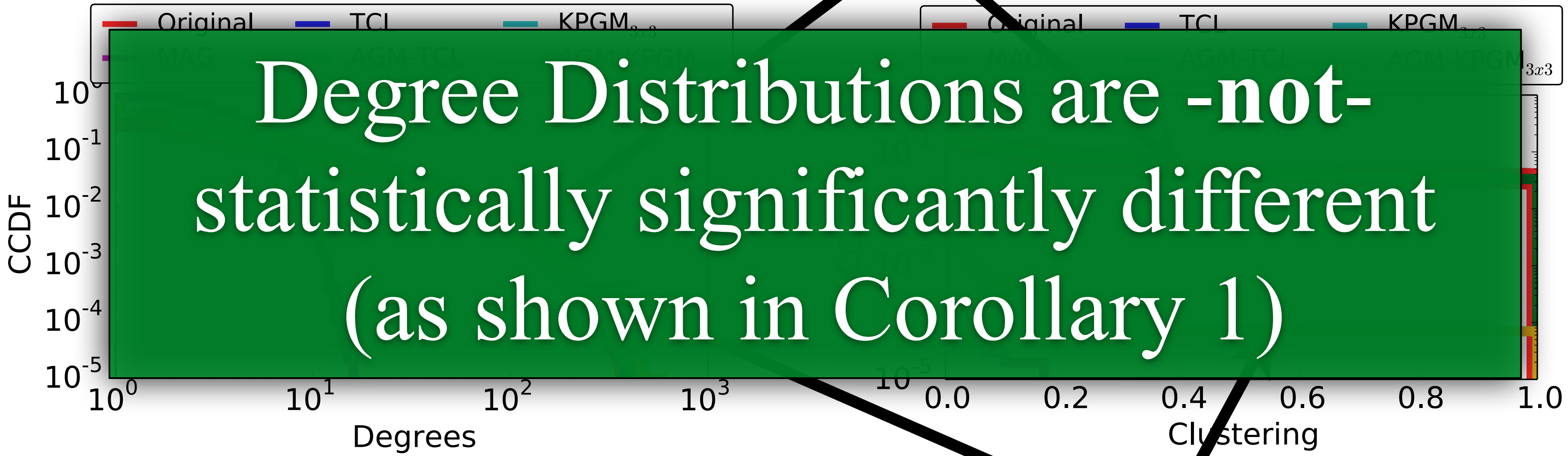
Facebook

AGM-KPGM Matches KPGM

Dataset	Degree Distribution KS Distances			
	FCL	TCL	KPGM	KPGM
Facebook	0.003	0.002	0.004	0.004

Structural Features

AGM-TCL Matches TCL



Degree Distributions are **-not-** statistically significantly different (as shown in Corollary 1)

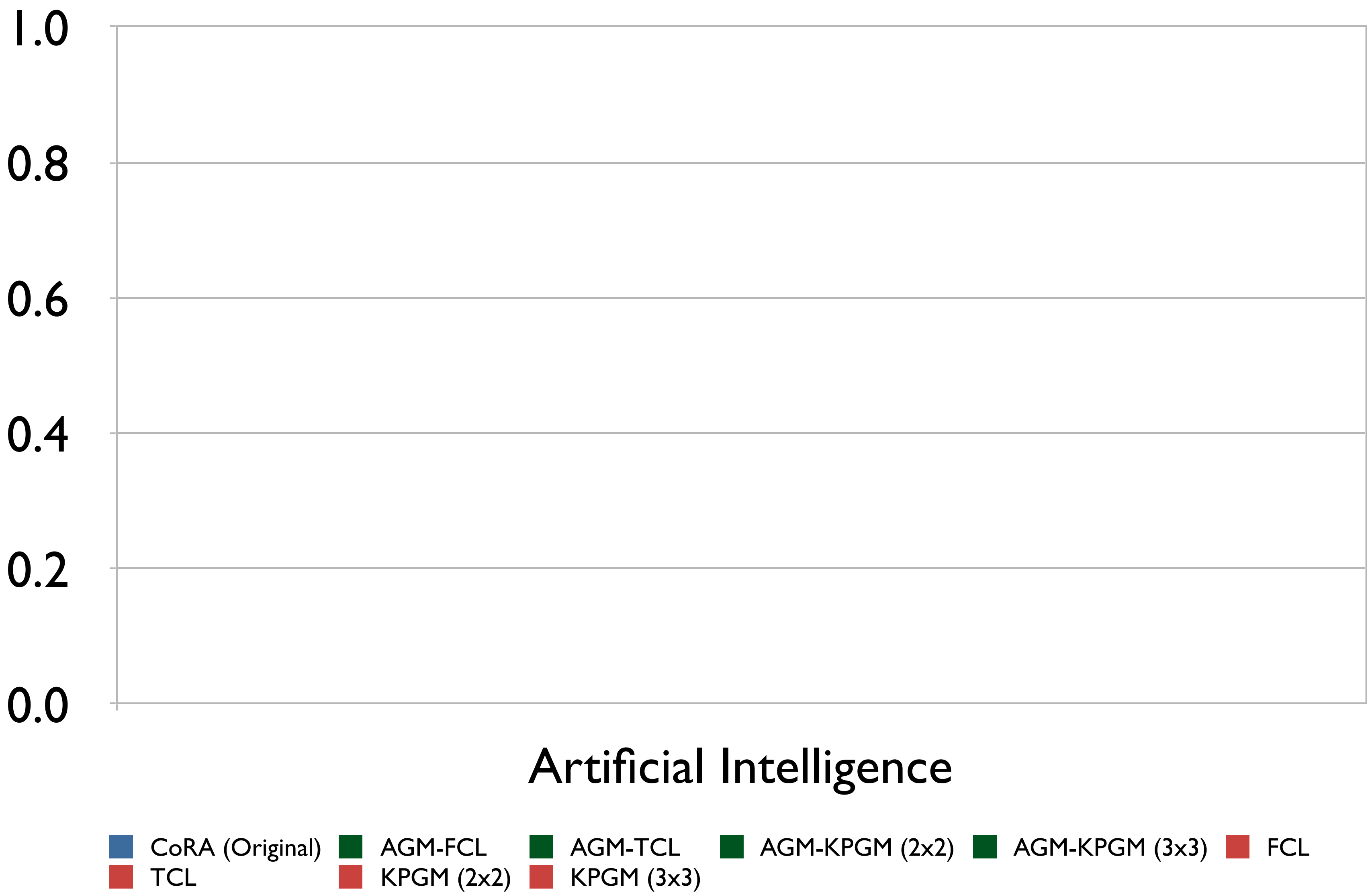
Facebook

AGM-KPGM Matches KPGM

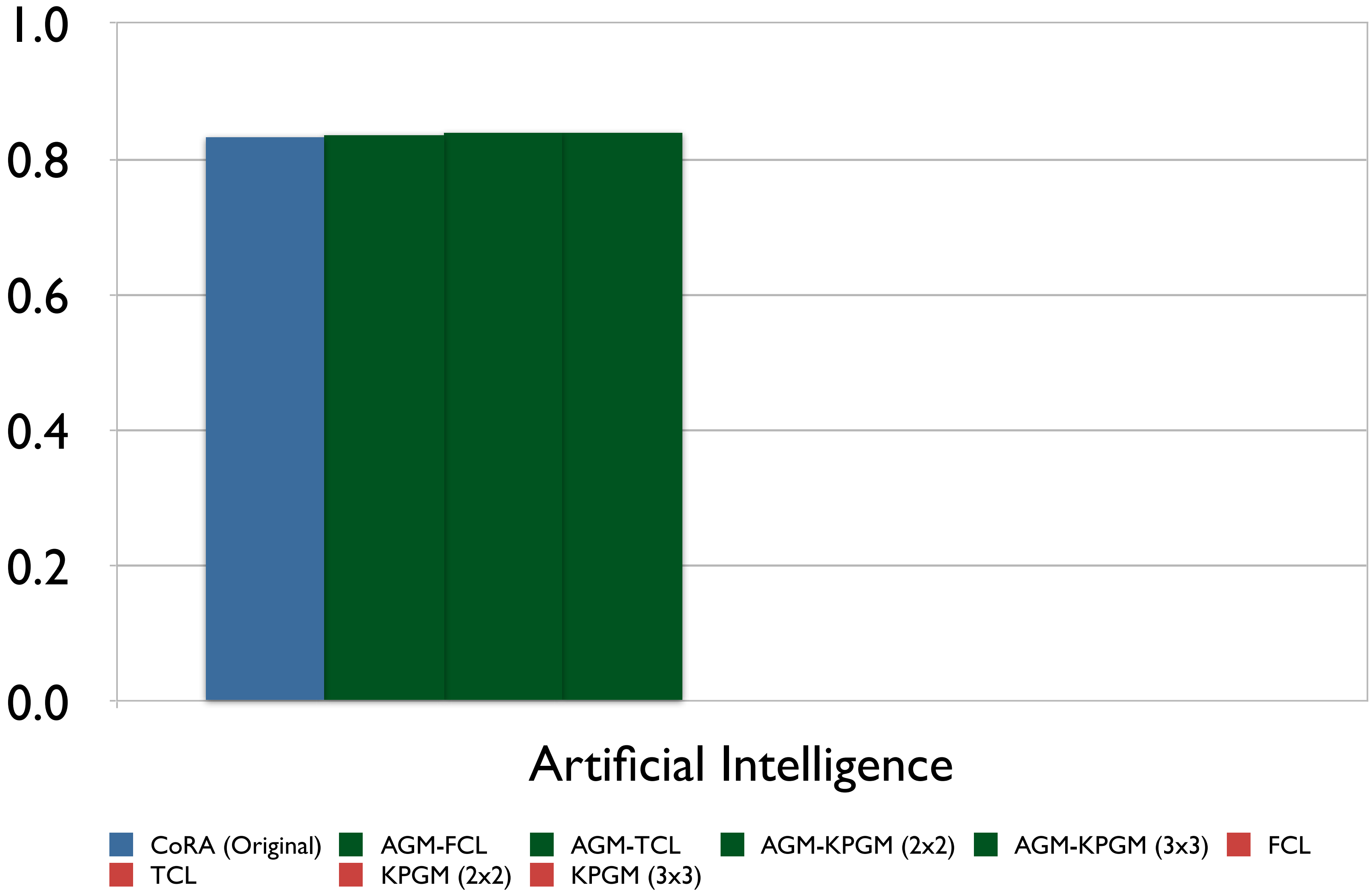
Dataset	Degree Distribution KS Distances			
	FCL	TCL	KPGM	KPGM
Facebook	0.003	0.002	0.004	0.004

Correlations - CoRA

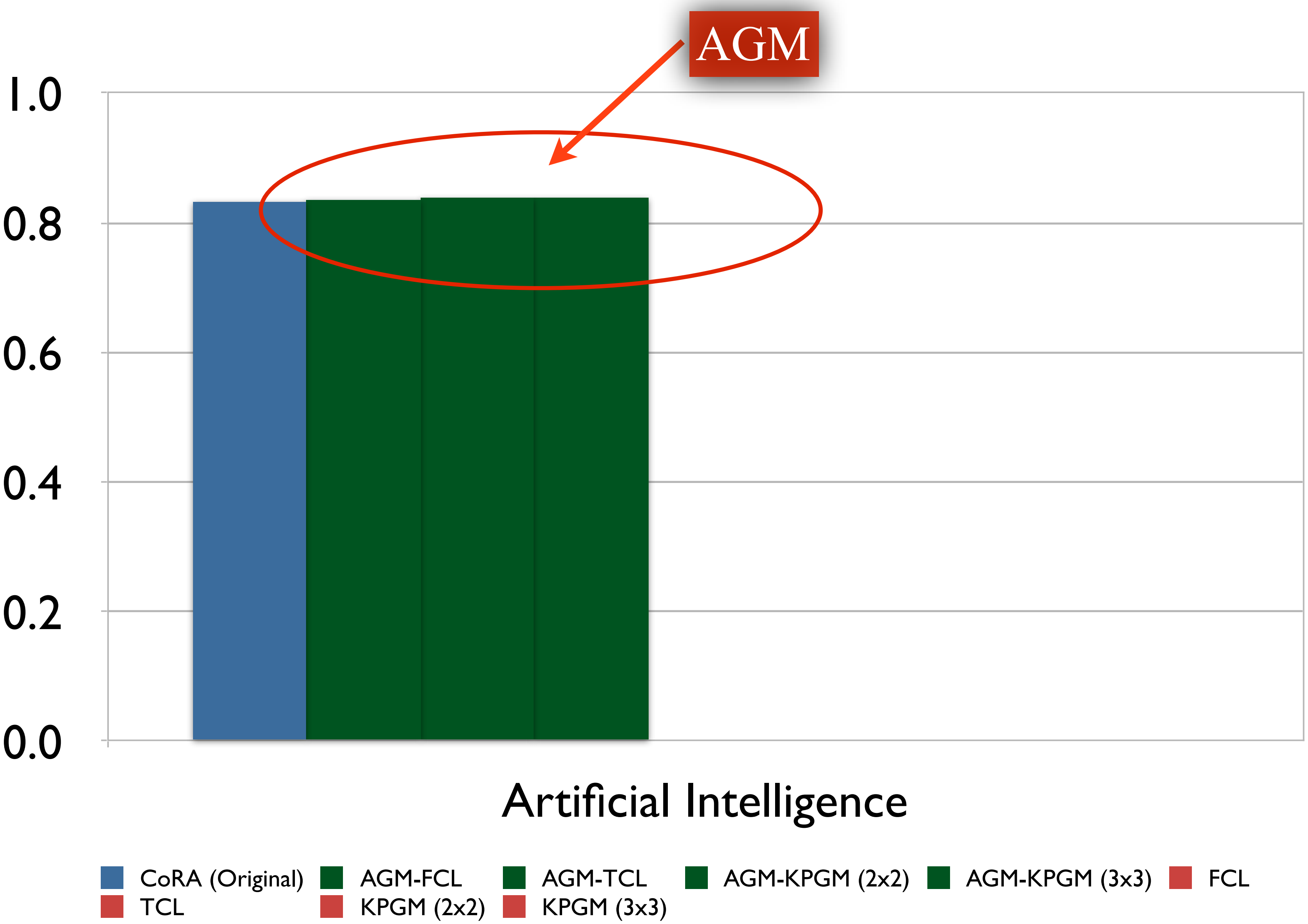
Correlations - CoRA



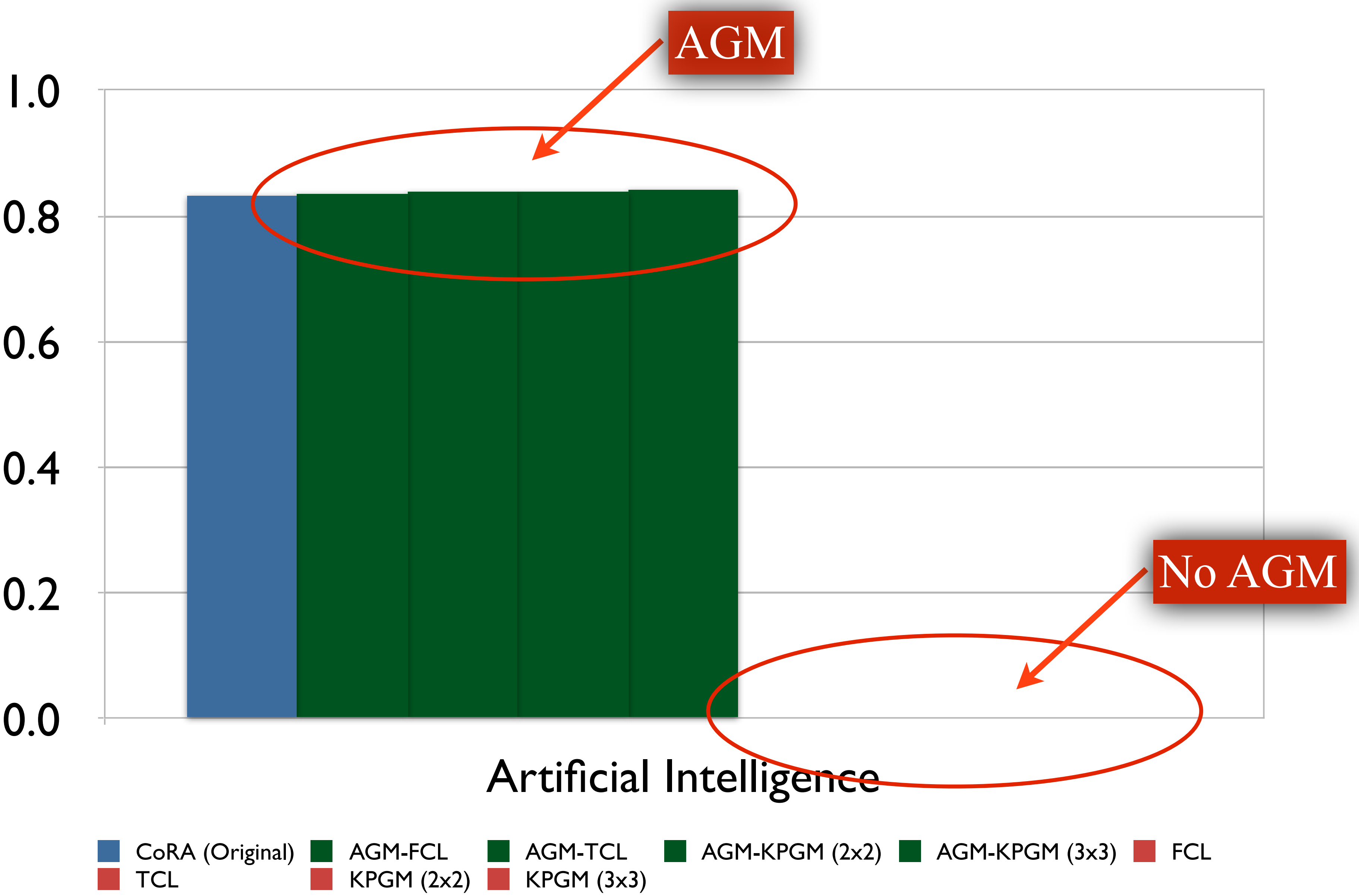
Correlations - CoRA



Correlations - CoRA

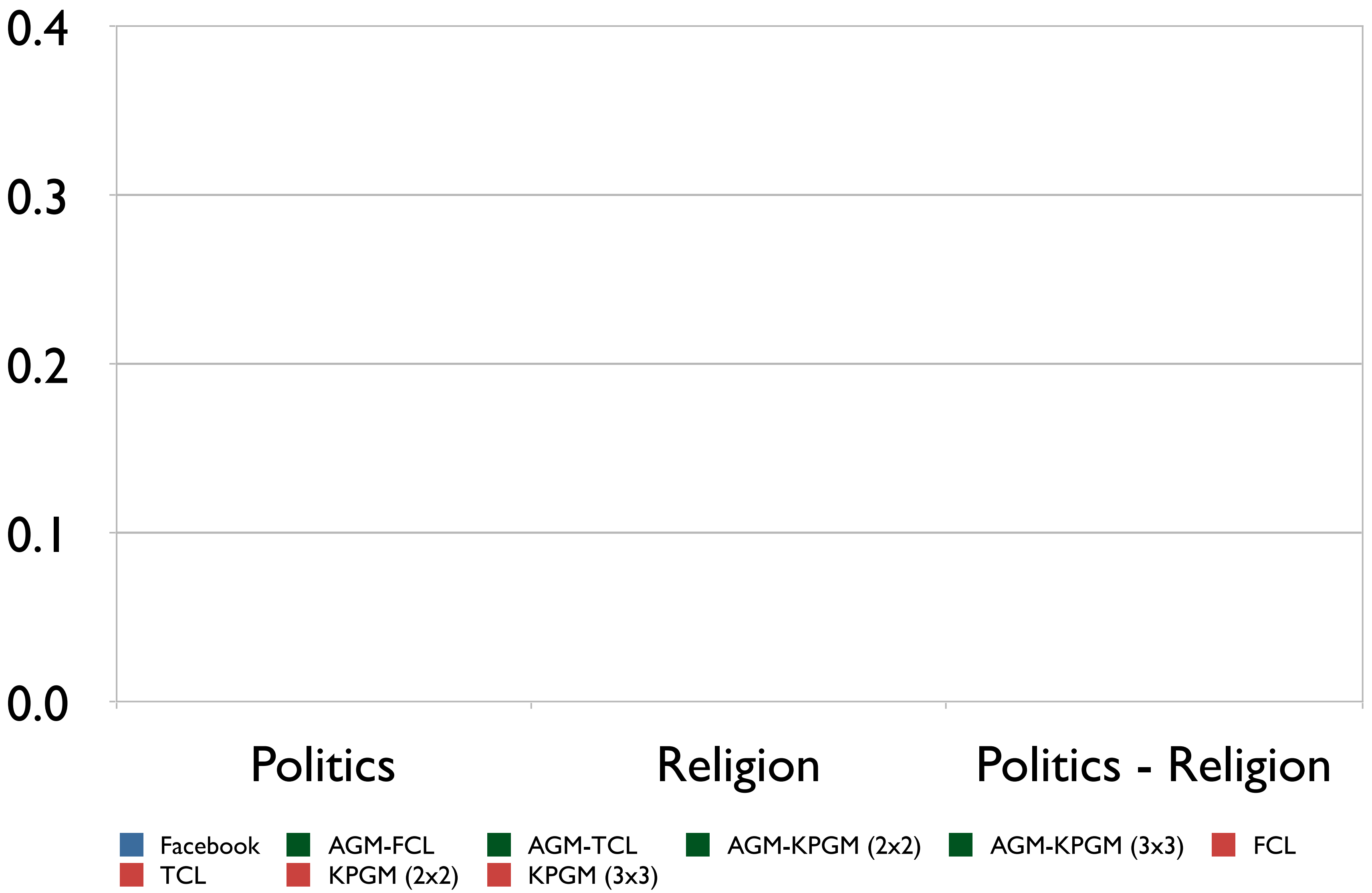


Correlations - CoRA

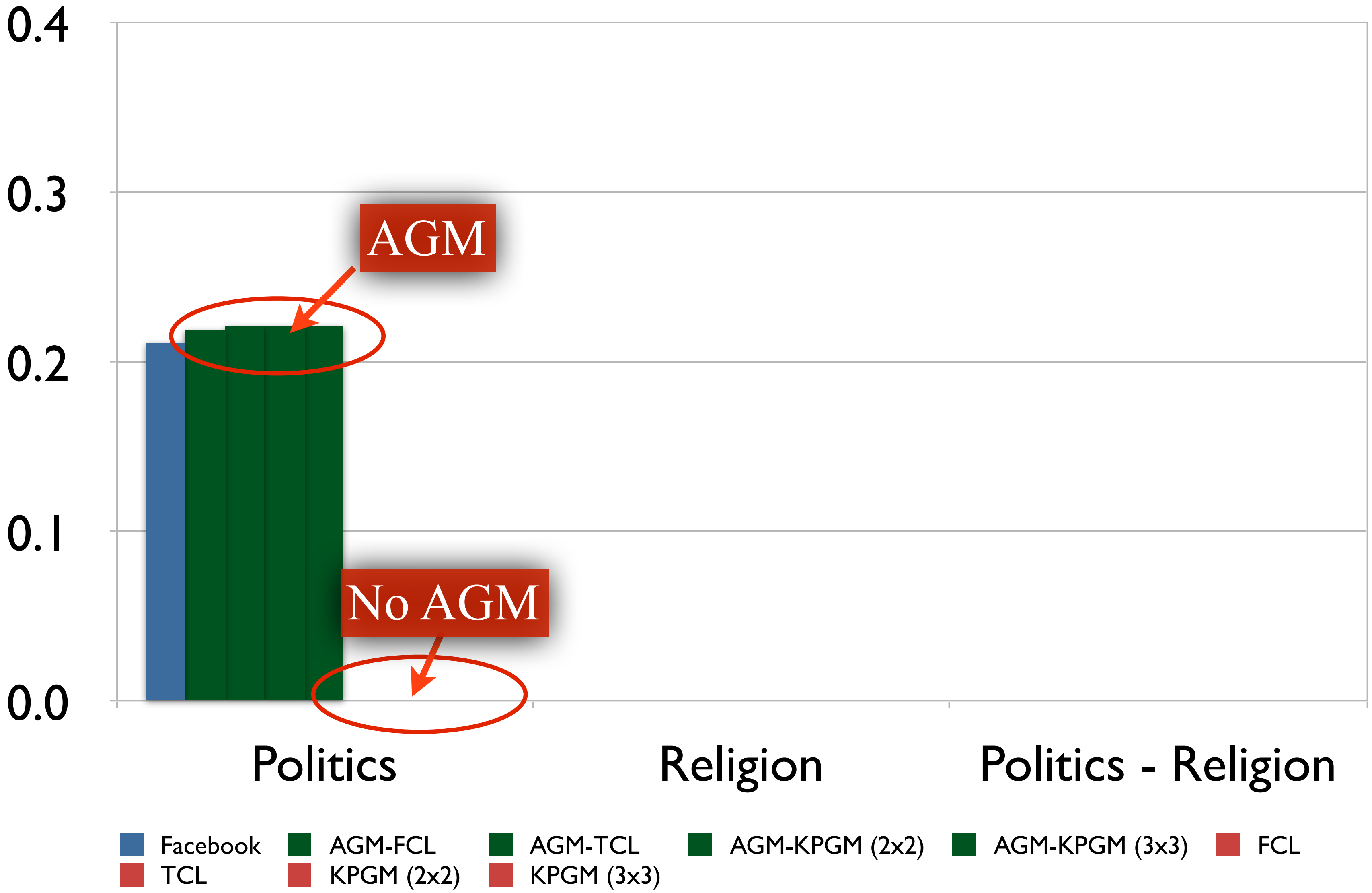


Correlations - Facebook

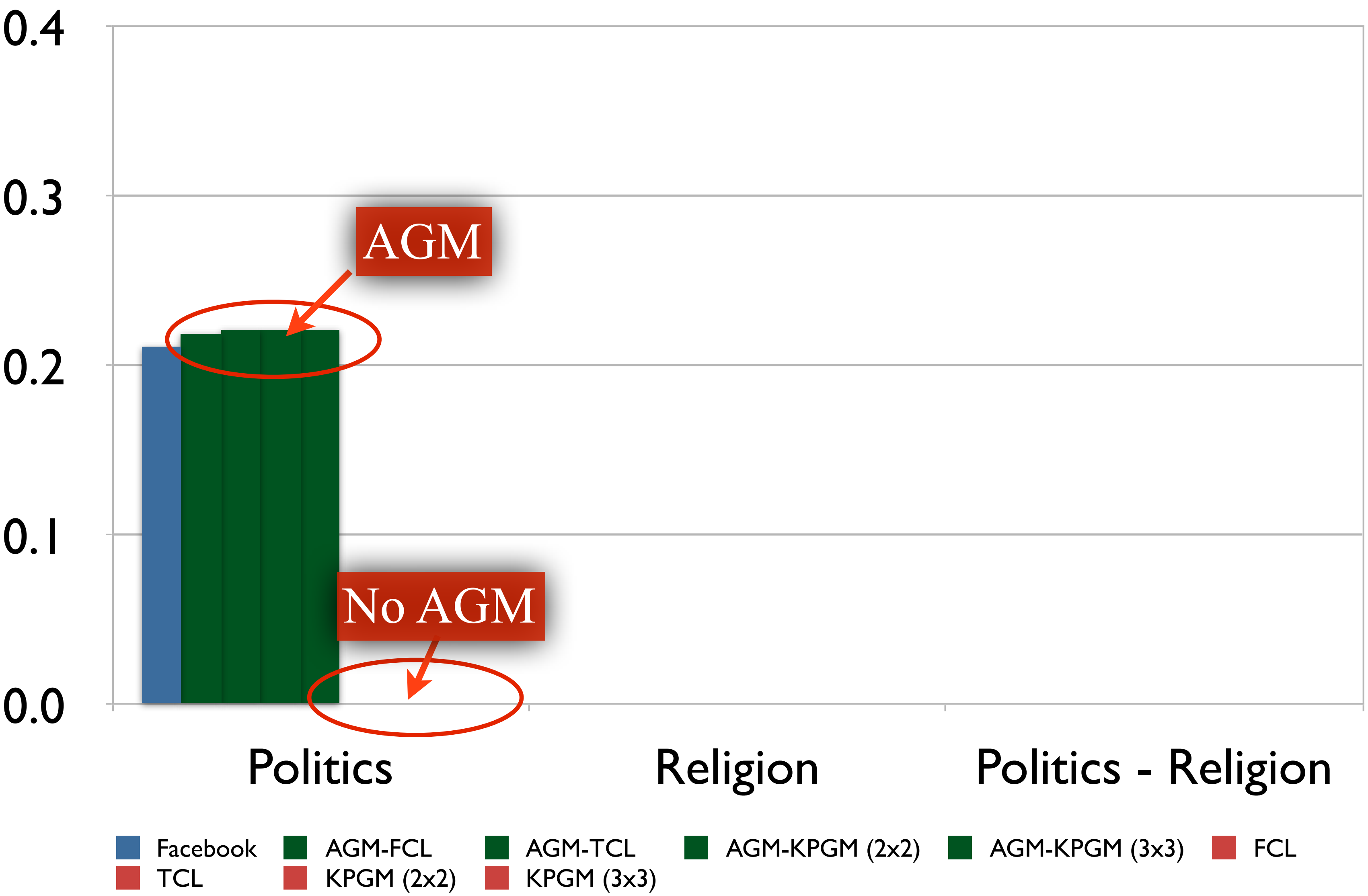
Correlations - Facebook



Correlations - Facebook



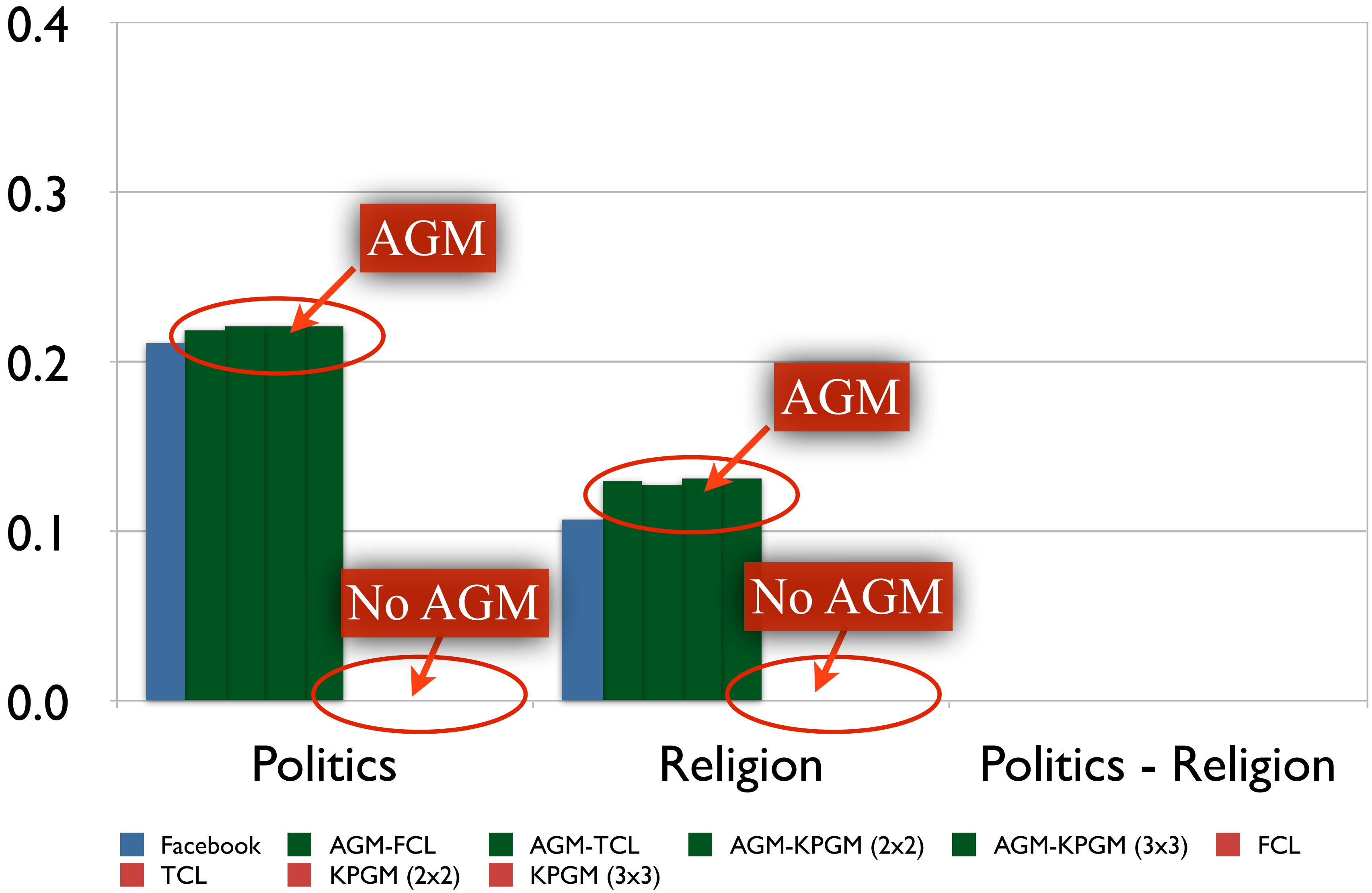
Correlations - Facebook



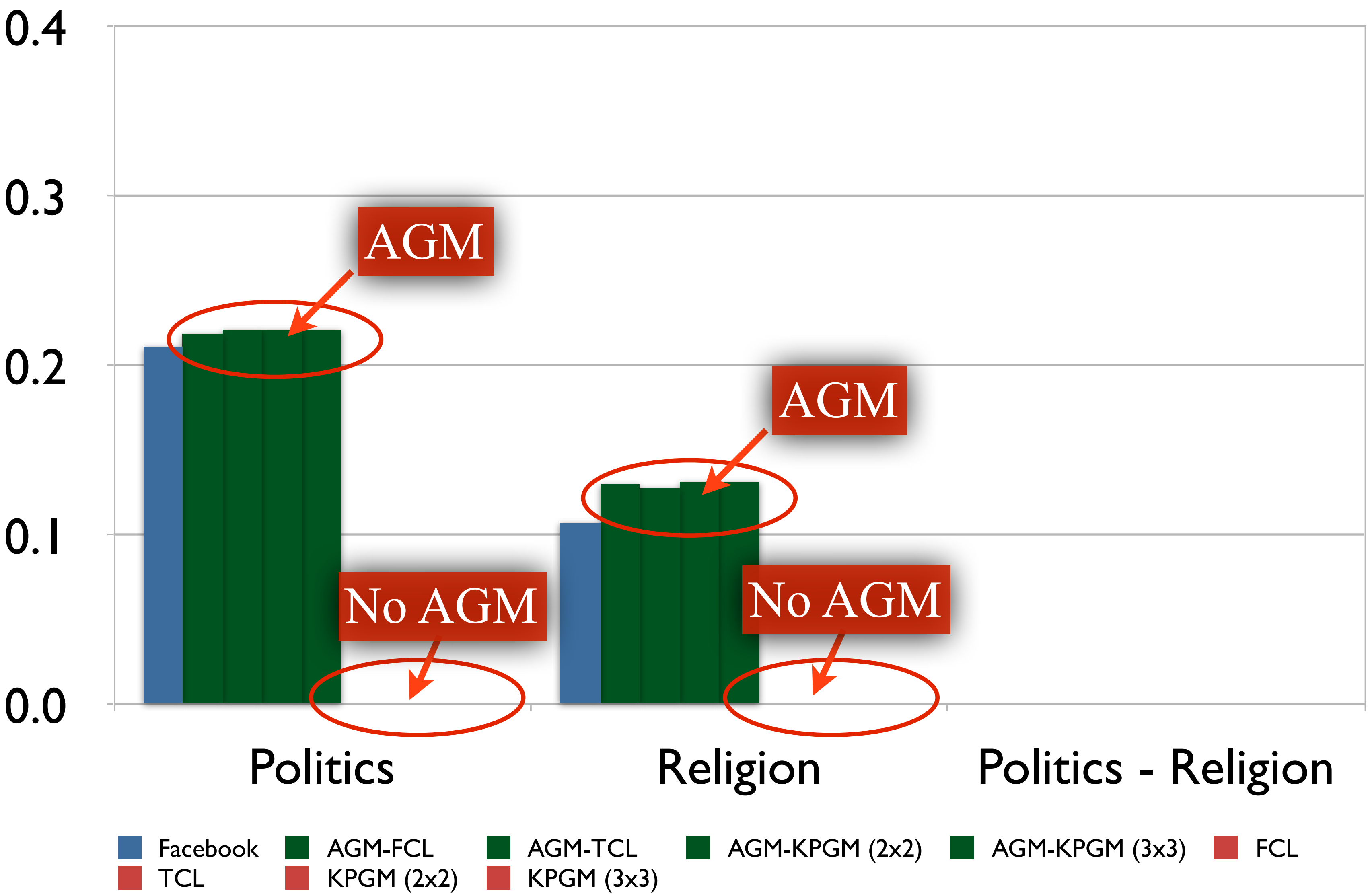
AGM

No AGM

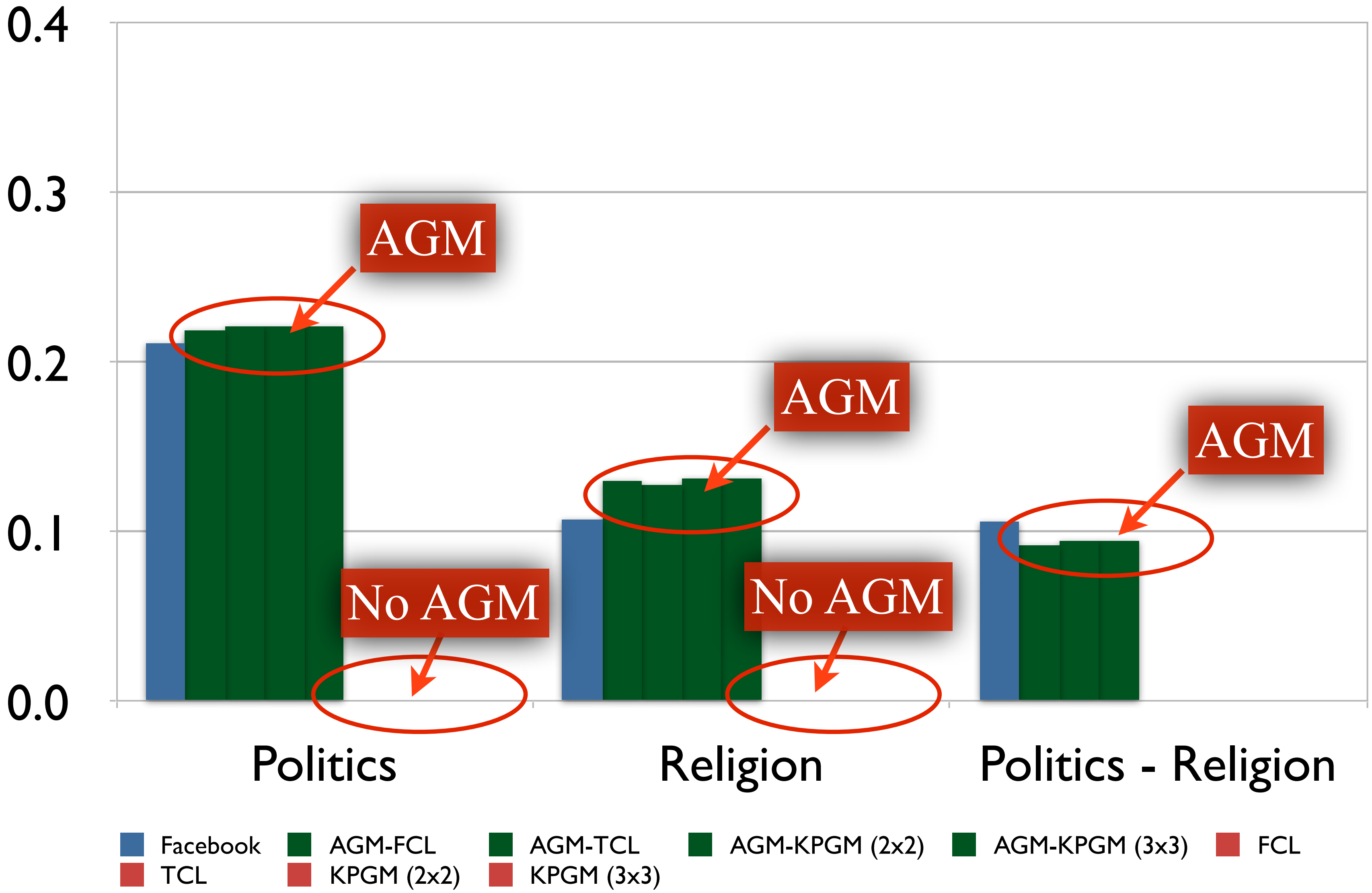
Correlations - Facebook



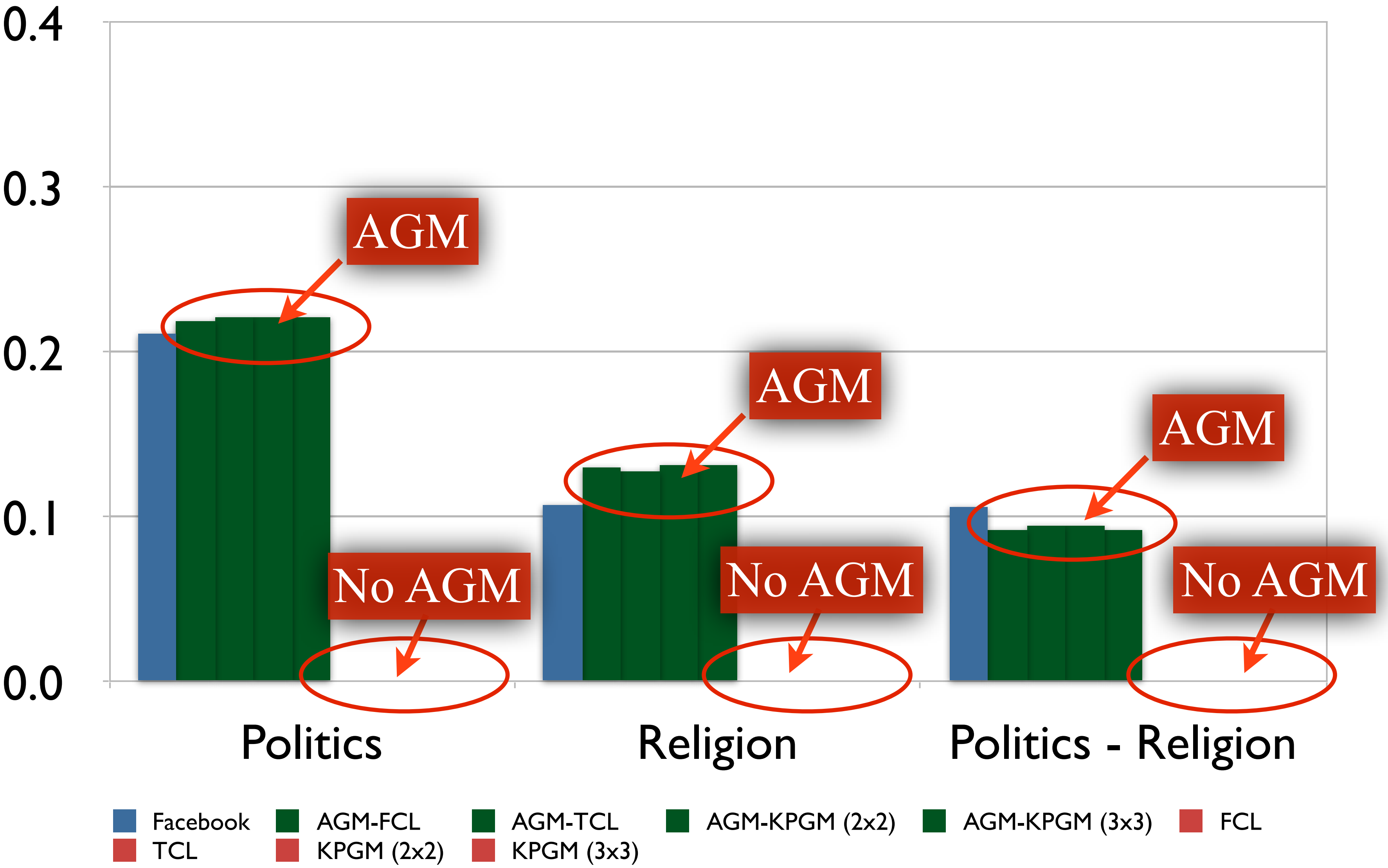
Correlations - Facebook



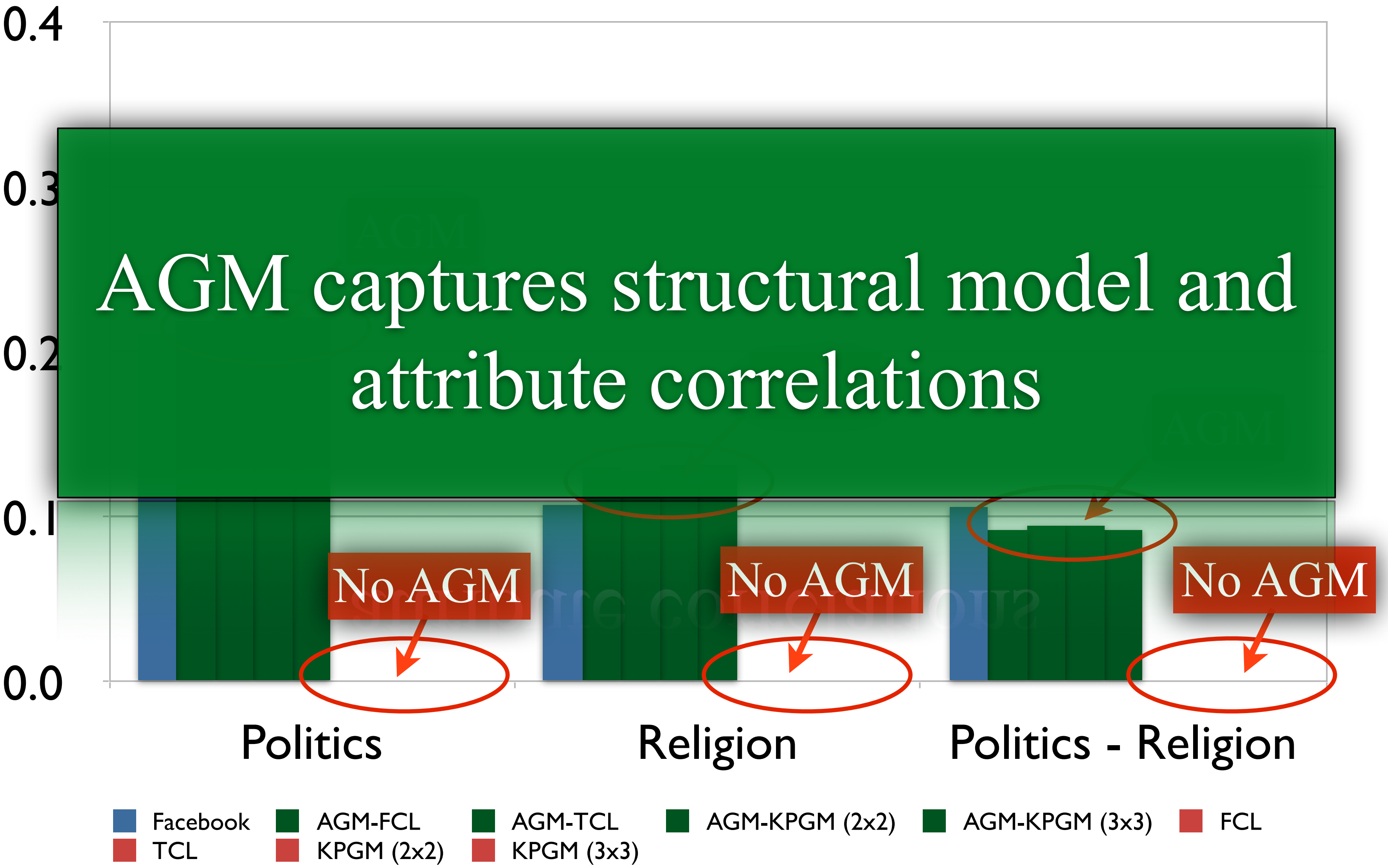
Correlations - Facebook



Correlations - Facebook



Correlations - Facebook



AGM captures structural model and attribute correlations

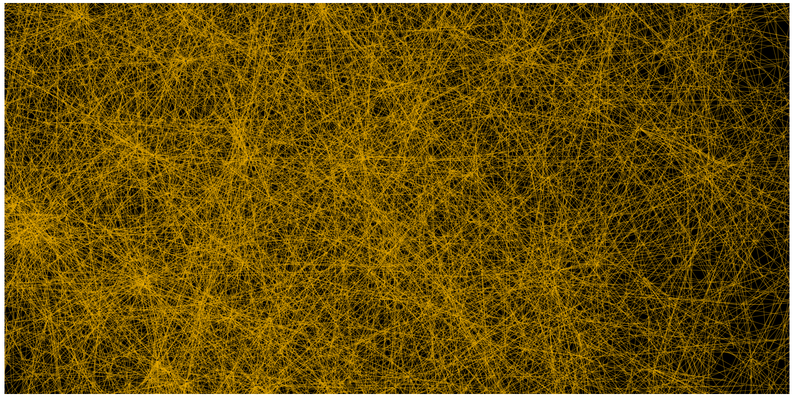
No AGM

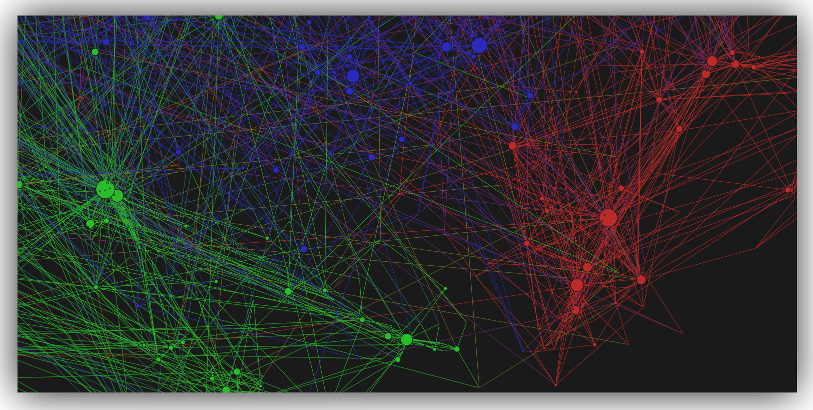
No AGM

No AGM

Outline:

- Background
- Scalable Graph Sampling
- Attributed Graph Models
  - Sampling
  - Theoretical Results
  - Learning From Data
- Experiments
- **Conclusions / Future Directions**





Conclusions / Future Directions

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models
  - Generalized and exploited sampling process to incorporate rejection sampling

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models
  - Generalized and exploited sampling process to incorporate rejection sampling
    - Sample edges conditioned on attributes

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models
  - Generalized and exploited sampling process to incorporate rejection sampling
    - Sample edges conditioned on attributes
  - Preserve structural graph properties provided by generative graph models

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models
  - Generalized and exploited sampling process to incorporate rejection sampling
    - Sample edges conditioned on attributes
  - Preserve structural graph properties provided by generative graph models
- Future Work

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models
  - Generalized and exploited sampling process to incorporate rejection sampling
    - Sample edges conditioned on attributes
  - Preserve structural graph properties provided by generative graph models
- Future Work
  - Extend rejection framework to other types of features

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models
  - Generalized and exploited sampling process to incorporate rejection sampling
    - Sample edges conditioned on attributes
  - Preserve structural graph properties provided by generative graph models
- Future Work
  - Extend rejection framework to other types of features
    - e.g., Triangles, Paths, etc…

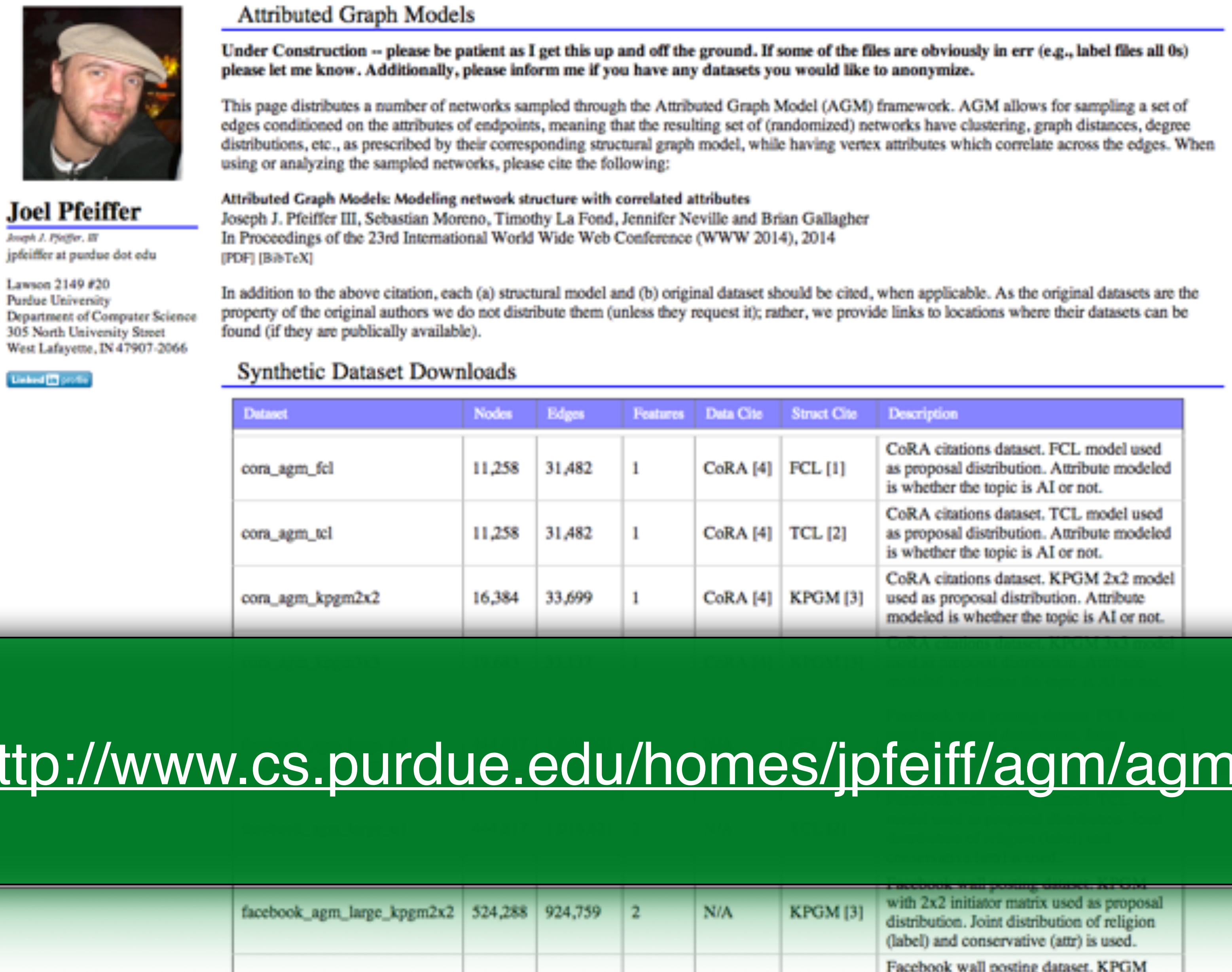
Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models
  - Generalized and exploited sampling process to incorporate rejection sampling
    - Sample edges conditioned on attributes
  - Preserve structural graph properties provided by generative graph models
- Future Work
  - Extend rejection framework to other types of features
    - e.g., Triangles, Paths, etc…
  - Annealing / Gibbs Sampling

Conclusions / Future Directions

- Introduced Attributed Graph Model Framework
  - Analysis of scalable generative graph models
  - Generalized and exploited sampling process to incorporate rejection sampling
    - Sample edges conditioned on attributes
  - Preserve structural graph properties provided by generative graph models
- Future Work
  - Extend rejection framework to other types of features
    - e.g., Triangles, Paths, etc…
  - Annealing / Gibbs Sampling
  - Investigate temporal network domains (homophily)

Datasets



Joel Pfeiffer
Joseph J. Pfeiffer, III
jpfeiffer at purdue dot edu

Lawson 2149 #20
Purdue University
Department of Computer Science
305 North University Street
West Lafayette, IN 47907-2066

Linked in profile

Attributed Graph Models

Under Construction -- please be patient as I get this up and off the ground. If some of the files are obviously in err (e.g., label files all 0s) please let me know. Additionally, please inform me if you have any datasets you would like to anonymize.

This page distributes a number of networks sampled through the Attributed Graph Model (AGM) framework. AGM allows for sampling a set of edges conditioned on the attributes of endpoints, meaning that the resulting set of (randomized) networks have clustering, graph distances, degree distributions, etc., as prescribed by their corresponding structural graph model, while having vertex attributes which correlate across the edges. When using or analyzing the sampled networks, please cite the following:

Attributed Graph Models: Modeling network structure with correlated attributes
Joseph J. Pfeiffer III, Sebastian Moreno, Timothy La Fond, Jennifer Neville and Brian Gallagher
In Proceedings of the 23rd International World Wide Web Conference (WWW 2014), 2014
[PDF] [BibTeX]

In addition to the above citation, each (a) structural model and (b) original dataset should be cited, when applicable. As the original datasets are the property of the original authors we do not distribute them (unless they request it); rather, we provide links to locations where their datasets can be found (if they are publically available).

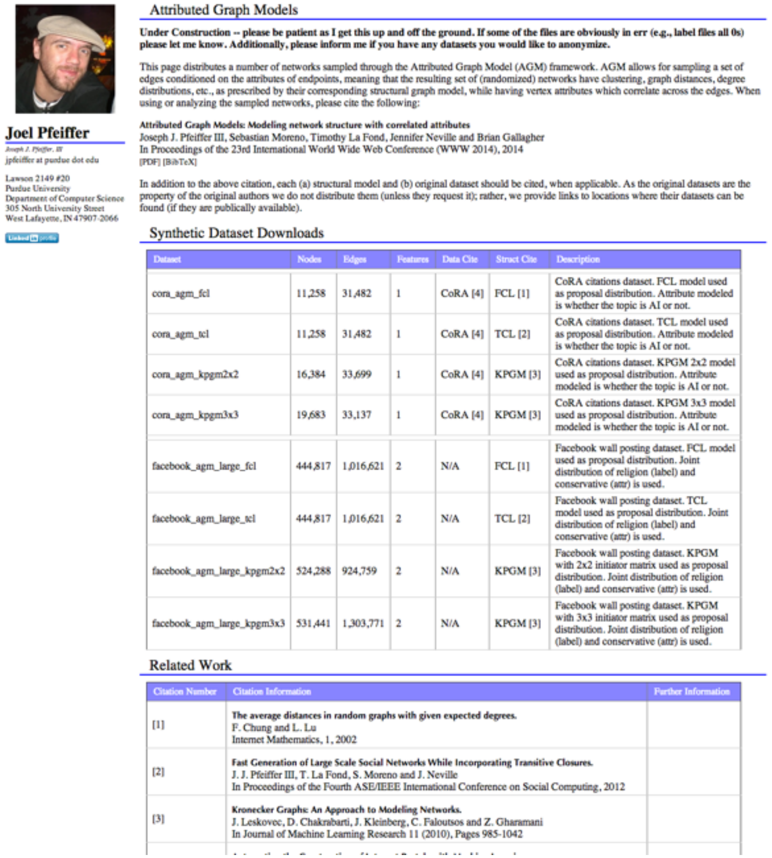
Synthetic Dataset Downloads

Dataset	Nodes	Edges	Features	Data Cite	Struct Cite	Description
cora_agm_fcl	11,258	31,482	1	CoRA [4]	FCL [1]	CoRA citations dataset. FCL model used as proposal distribution. Attribute modeled is whether the topic is AI or not.
cora_agm_tcl	11,258	31,482	1	CoRA [4]	TCL [2]	CoRA citations dataset. TCL model used as proposal distribution. Attribute modeled is whether the topic is AI or not.
cora_agm_kpgm2x2	16,384	33,699	1	CoRA [4]	KPGM [3]	CoRA citations dataset. KPGM 2x2 model used as proposal distribution. Attribute modeled is whether the topic is AI or not.
facebook_agm_large_kpgm2x2	524,288	924,759	2	N/A	KPGM [3]	Facebook wall posting dataset. KPGM with 2x2 initiator matrix used as proposal distribution. Joint distribution of religion (label) and conservative (attr) is used.
						Facebook wall posting dataset. KPGM

http://www.cs.purdue.edu/homes/jpfeiff/agm/agm.html

Thanks!

Email: jpfeiffer@purdue.edu
Twitter: @jjpfeiffer3
Datasets: http://www.cs.purdue.edu/homes/jpfeiff/agm/agm.html



Joel Pfeiffer
Joseph J. Pfeiffer, III
jpfeiffer at purdue dot edu

Lawson 2149 #20
Purdue University
Department of Computer Science
305 North University Street
West Lafayette, IN 47907-2066

Attributed Graph Models

Under Construction -- please be patient as I get this up and off the ground. If some of the files are obviously in err (e.g., label files all 0s) please let me know. Additionally, please inform me if you have any datasets you would like to anonymize.

This page distributes a number of networks sampled through the Attributed Graph Model (AGM) framework. AGM allows for sampling a set of edges conditioned on the attributes of endpoints, meaning that the resulting set of (randomized) networks have clustering, graph distances, degree distributions, etc., as prescribed by their corresponding structural graph model, while having vertex attributes which correlate across the edges. When using or analyzing the sampled networks, please cite the following:

Attributed Graph Models: Modeling network structure with correlated attributes
Joseph J. Pfeiffer III, Sebastian Moreno, Timothy La Fond, Jennifer Neville and Brian Gallagher
In Proceedings of the 23rd International World Wide Web Conference (WWW 2014), 2014
[PDF] [BibTeX]

In addition to the above citation, each (a) structural model and (b) original dataset should be cited, when applicable. As the original datasets are the property of the original authors we do not distribute them (unless they request it); rather, we provide links to locations where their datasets can be found (if they are publically available).

Synthetic Dataset Downloads

Dataset	Nodes	Edges	Features	Data Cite	Struct Cite	Description
cora_agm_fcl	11,258	31,482	1	CoRA [4]	FCL [1]	CoRA citations dataset. FCL model used as proposal distribution. Attribute modeled is whether the topic is AI or not.
cora_agm_tcl	11,258	31,482	1	CoRA [4]	TCL [2]	CoRA citations dataset. TCL model used as proposal distribution. Attribute modeled is whether the topic is AI or not.
cora_agm_kpgm2x2	16,384	33,699	1	CoRA [4]	KPGM [3]	CoRA citations dataset. KPGM 2x2 model used as proposal distribution. Attribute modeled is whether the topic is AI or not.
cora_agm_kpgm3x3	19,683	33,137	1	CoRA [4]	KPGM [3]	CoRA citations dataset. KPGM 3x3 model used as proposal distribution. Attribute modeled is whether the topic is AI or not.
facebook_agm_large_fcl	444,817	1,016,621	2	N/A	FCL [1]	Facebook wall posting dataset. FCL model used as proposal distribution. Joint distribution of religion (label) and conservative (attr) is used.
facebook_agm_large_tcl	444,817	1,016,621	2	N/A	TCL [2]	Facebook wall posting dataset. TCL model used as proposal distribution. Joint distribution of religion (label) and conservative (attr) is used.
facebook_agm_large_kpgm2x2	524,288	924,759	2	N/A	KPGM [3]	Facebook wall posting dataset. KPGM with 2x2 initiator matrix used as proposal distribution. Joint distribution of religion (label) and conservative (attr) is used.
facebook_agm_large_kpgm3x3	531,441	1,303,771	2	N/A	KPGM [3]	Facebook wall posting dataset. KPGM with 3x3 initiator matrix used as proposal distribution. Joint distribution of religion (label) and conservative (attr) is used.

Related Work

Citation Number	Citation Information	Further Information
[1]	**The average distances in random graphs with given expected degrees.** F. Chung and L. Lu Internet Mathematics, 1, 2002	
[2]	**Fast Generation of Large Scale Social Networks While Incorporating Transitive Closures.** J. J. Pfeiffer III, T. La Fond, S. Moreno and J. Neville In Proceedings of the Fourth ASE/IEEE International Conference on Social Computing, 2012	
[3]	**Kronecker Graphs: An Approach to Modeling Networks.** J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos and Z. Gharamani In Journal of Machine Learning Research 11 (2010), Pages 985-1042	

Sebastian Moreno
smorena@purdue.edu

Timothy La Fond
tlafond@purdue.edu

Jennifer Neville
neville@cs.purdue.edu

Brian Gallagher
bgallagher@llnl.gov

Cites

- BA2001: Barabasi and Albert. Emergence of Scaling in Random Networks. *Science*
- CL2002: Chung and Lu. The Average Distances in Random Graphs with Given Expected Degrees. PNAS 2002
- Getoor & Taskar, 2007. An Introduction to Statistical Relational Learning.
- L2010: Leskovec, Chakrabarti, Kleinberg, Faloutsos, Ghahramani. Kronecker Graphs: An approach to modeling networks. JMLR 2010
- LBKT2008: Leskovec, Backstrom, Kumar, Tomkins. Microscopic Evolution of Social Networks. KDD 2008
- SPT2013: Seshadhri, Pinar, Kolda. An In-Depth Analysis of Stochastic Kronecker Graphs. JACM 2013
- WS1998: Watts and Strogatz. Collective Dynamics of Small World Networks. *Nature*