A SCALABLE METHOD FOR EXACT SAMPLING FROM KRONECKER FAMILY MODELS

Sebastian Moreno, Joseph Pfeiffer, Jennifer Neville, Sergey Kirshner
Introduction

- Kronecker models are probabilistic generative graph models, with a fractal structure, able to capture important network characteristics.
Introduction

• Kronecker models are probabilistic generative graph models, with a fractal structure, able to capture important network characteristics.

• Naive implementations of their sampling process have complexity $O(N\sqrt{V}^2)$, which is intractable for large networks.
• Kronecker models are probabilistic generative graph models, with a fractal structure, able to capture important network characteristics.

• Naive implementations of their sampling process have complexity $O(NV^2)$, which is intractable for large networks.

• Scalable sampling methods, in contrast to prior belief, do not sample from the underlying KPGM distribution, generating unlikely graphs.
Introduction

- Kronecker models are probabilistic generative graph models, with a fractal structure, able to capture important network characteristics.

We create the first $\tilde{O}(N_e)$ sampling algorithms for Kronecker models (KPGM and mKPGM), able to sample accurately from the model probability distribution.

- Scalable sampling methods, in contrast to prior belief, do not sample from the underlying KPGM distribution, generating unlikely graphs.
KRONECKER PROBABILITY GRAPH MODEL

MIXED KRONECKER PROBABILITY GRAPH MODEL
KPGM models a network using Θ, which is extended using $K-1$ Kronecker multiplications with itself.

\[
\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}
\]

$K = 6$

\[
\Theta = \begin{bmatrix}
0.99 & 0.90 \\
0.60 & 0.30
\end{bmatrix}
\]
KPGM models a network using $\Theta$, which is extended using $K-1$ Kronecker multiplications with itself.

$$
\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}
$$

$$
K = 6
$$

$$
\Theta = \begin{bmatrix}
0.99 & 0.90 \\
0.60 & 0.30
\end{bmatrix}
$$
Each cell is a probability between two nodes, which are sampled independently with Bernoulli distributions.

\[ \Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \]

\[ K = 6 \]

\[ \Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix} \]
Each cell is a probability between two nodes, which are sampled independently with Bernoulli distributions.

\[
\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}
\]

\[K = 6\]

\[
\Theta = \begin{bmatrix}
0.99 & 0.90 \\
0.60 & 0.30
\end{bmatrix}
\]
The space of graphs $G^K_o$ is all possible networks able to be generated by KPGM with specific parameters $K$ and $\Theta$. 
The space of graphs $G^K_o$ is all possible network able to be generated by KPGM with specific parameters K and $\Theta$.
• Kronecker Product Graph Model
  • Low number of parameters.
  • Elegant fractal structure.
  • Networks have between 0 to $N\sqrt{2}$ edges.
  • Capture important network characteristics such as: degree and diameter.
• Kronecker Product Graph Model
  • Low number of parameters.
  • Elegant fractal structure.
  • Networks have between 0 to $N\sqrt{2}$ edges.
  • Capture important network characteristics such as: degree and diameter.

• Naive implementations of their sampling process have complexity $O(N\sqrt{2})$, which is intractable for large networks.
Edge by edge generation algorithm generates an edge, sampling $K$ times from a multinomial using $\Theta$.

\[
\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}
\]

\[
K = 6
\]

\[
\Theta = \begin{bmatrix}
0.99 & 0.90 \\
0.60 & 0.30
\end{bmatrix}
\]
Edge by edge generation algorithm generates an edge, sampling $K$ times from a multinomial using $\Theta$. 

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

$$K = 6$$

$$\Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix}$$
Edge by edge generation algorithm generates an edge, sampling $K$ times from a multinomial using $\Theta$.

\[
\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}
\]

$K = 6$

\[
\Theta = \begin{bmatrix}
0.99 & 0.90 \\
0.60 & 0.30
\end{bmatrix}
\]
Edge by edge generation algorithm generates an edge, sampling $K$ times from a multinomial using $\Theta$.

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

$$K = 6$$

$$\Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix}$$
Edge by edge generation algorithm generates an edge, sampling $K$ times from a multinomial using $\Theta$.

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

$$K = 6$$

$$\Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix}$$
Edge by edge generation algorithm generates an edge, sampling $K$ times from a multinomial using $\Theta$.

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

$$K = 6$$

$$\Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix}$$
X edges are randomly placed in the network. Allowing collisions (multigraphs) in the generated networks

\[ \Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \]

\[ K = 6 \]

\[ \Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix} \]
X edges are randomly placed in the network. Allowing collisions (multigraphs) in the generated networks

\[ \Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \]

\[ K = 6 \]

\[ \Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix} \]
X edges are randomly placed in the network. Allowing collisions (multigraphs) in the generated networks

$$\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}$$

Edge by edge algorithms sample X edges “jointly”, generating collisions among sampled edges.
Edge by edge with rejection

X edges are randomly placed in the network. Collisions are rejected and resampled.

\[ \Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \]

\[ K = 6 \]

\[ \Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix} \]
Edge by edge with rejection

X edges are randomly placed in the network. Collisions are rejected and resampled.

\[
\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}
\]

\[
K = 6
\]

\[
\Theta = \begin{bmatrix}
0.99 & 0.90 \\
0.60 & 0.30
\end{bmatrix}
\]
Edge by edge with rejection increases the probability of unlikely edges, by reallocating the probability of previous sampled edges throughout the network.
The space of graphs generated by edge by edge algorithms is limited by the number of edges (X).
The space of graphs generated by edge by edge algorithms is limited by the number of edges ($X$).
The space of graphs generated by edge by edge algorithms is limited by the number of edges (X).

Networks with $N_e$ different than $X$
The space of graphs generated by edge by edge algorithms is limited by the number of edges (X).

Networks with $N_e$ different than $X$
The space of graphs generated by edge by edge algorithms is limited by the number of edges ($X$).
KPGM sampling

The generation of a network is similar to the sampling of a network from the space of graphs $G^K_o$.
The generation of a network is similar to the sampling of a network from the space of graphs.
The generation of a network is similar to the sampling of a network from the space of graphs.
The generation of a network is similar to the sampling of a network from the space of graphs.
The generation of a network is similar to the sampling of a network from the space of graphs.
The generation of a network is similar to the sampling of a network from the space of graphs. Edge by edge methods sample from a different probability distribution. Theorems: 5 and 6. Corollary: 2.
Even sampling the number of edges from $N(\mu, \sigma^2)$ or Poisson($\lambda$), as is suggested, the empirical distributions do not match the original analytical distribution.
Group probability sampling algorithm

- We develop a new scalable algorithm for KPGM
  - Sample network from the original space of graphs.
  - Replicate underlying probability graph distribution.
Group probability sampling algorithm

- We develop a new scalable algorithm for KPGM
  - Sample network from the original space of graphs.
  - Replicate underlying probability graph distribution.

- We develop a grouped sampling process, where we independently sample groups of edges with the same probability.
Group probability sampling, example

Blue “events” are groups of edges with the same probability
Blue “events” are groups of edges with the same probability.
Group probability sampling

- KPGM have multiple elements with the same probability, because of the Kronecker multiplications and the commutative property.
Group probability sampling

- KPGM have multiple elements with the same probability, because of the Kronecker multiplications and the commutative property.

- We propose a new group probability sampling for KPGM
  - Aggregate events for unique probability combination

\[
\begin{bmatrix}
\theta_{11}\theta_{11} & \theta_{11}\theta_{12} & \theta_{12}\theta_{11} & \theta_{12}\theta_{12} \\
\theta_{11}\theta_{21} & \theta_{11}\theta_{22} & \theta_{12}\theta_{21} & \theta_{12}\theta_{22} \\
\theta_{21}\theta_{11} & \theta_{21}\theta_{12} & \theta_{22}\theta_{11} & \theta_{22}\theta_{12} \\
\theta_{21}\theta_{21} & \theta_{21}\theta_{22} & \theta_{22}\theta_{21} & \theta_{22}\theta_{22}
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\pi_k \\
T_k
\end{bmatrix}
\begin{bmatrix}
\theta_{11}\theta_{11} & 1 \\
\theta_{11}\theta_{12} & 2 \\
\theta_{11}\theta_{21} & 2 \\
\theta_{11}\theta_{22} & 2 \\
\vdots & \vdots \\
\theta_{22}\theta_{21} & 2 \\
\theta_{22}\theta_{22} & 1
\end{bmatrix}
Group probability sampling

• KPGM have multiple elements with the same probability, because of the Kronecker multiplications and the commutative property.

• We propose a new group probability sampling for KPGM
  – Aggregate events for unique probability combination
  – Sample the aggregate event from a Binomial distribution

<table>
<thead>
<tr>
<th>( \pi_k )</th>
<th>( T_k )</th>
<th>( Bin(T_k, \pi_k) )</th>
<th>( x_k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta_{11} \theta_{11} )</td>
<td>1</td>
<td>( Bin(1, \theta_{11} \theta_{11}) )</td>
<td>1</td>
</tr>
<tr>
<td>( \theta_{11} \theta_{12} )</td>
<td>2</td>
<td>( Bin(2, \theta_{11} \theta_{12}) )</td>
<td>2</td>
</tr>
<tr>
<td>( \theta_{11} \theta_{21} )</td>
<td>2</td>
<td>( Bin(2, \theta_{11} \theta_{21}) )</td>
<td>0</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( \theta_{22} \theta_{21} )</td>
<td>2</td>
<td>( Bin(2, \theta_{22} \theta_{21}) )</td>
<td>1</td>
</tr>
<tr>
<td>( \theta_{22} \theta_{22} )</td>
<td>1</td>
<td>( Bin(1, \theta_{22} \theta_{22}) )</td>
<td>0</td>
</tr>
</tbody>
</table>
Group probability sampling

- KPGM have multiple elements with the same probability, because of the Kronecker multiplications and the commutative property.

- We propose a new group probability sampling for KPGM
  - Aggregate events for unique probability combination
  - Sample the aggregate event from a Binomial distribution
  - Place edges uniformly at random among positions with value $\pi_k$

<table>
<thead>
<tr>
<th>$\pi_k$</th>
<th>$T_k$</th>
<th>$Bin(T_k, \pi_k)$</th>
<th>$x_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_{11}\theta_{11}$</td>
<td>1</td>
<td>$Bin(1, \theta_{11}\theta_{11})$</td>
<td>1</td>
</tr>
<tr>
<td>$\theta_{11}\theta_{12}$</td>
<td>2</td>
<td>$Bin(2, \theta_{11}\theta_{12})$</td>
<td>2</td>
</tr>
<tr>
<td>$\theta_{11}\theta_{21}$</td>
<td>2</td>
<td>$Bin(2, \theta_{11}\theta_{21})$</td>
<td>0</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$\theta_{22}\theta_{21}$</td>
<td>2</td>
<td>$Bin(2, \theta_{22}\theta_{21})$</td>
<td>1</td>
</tr>
<tr>
<td>$\theta_{22}\theta_{22}$</td>
<td>1</td>
<td>$Bin(1, \theta_{22}\theta_{22})$</td>
<td>0</td>
</tr>
</tbody>
</table>
**Algorithm**  Group Probability Sampling

1: $V = \{1, \ldots, N_v\}, E = \{\}$

2: Construct $U$, the set of unique probability values $\pi'_k$

3: for $k = 1; k++; k \leq |U|$ do

4: Obtain $\pi'_k$, the $k$-th unique probability of the set $U$

5: Calculate $T_k$

6: Draw $x_k \sim Bin(T_k, \pi'_k)$

7: countEdge=0

8: while countEdge < $x_k$ do

9: Generate new edge $E_{uv}$

10: if $E_{uv} \notin E$ then

11: countEdge++

12: $E = E \cup \{E_{uv}\}$

13: Return $G = (V, E)$
Group probability sampling, pseudocode

Algorithm: Group Probability Sampling

1: \( V = \{1, \cdots, N_v\}, E = \{\} \)
2: Construct \( U \), the set of unique probability values \( \pi'_k \)
3: for \( k = 1; k++; k \leq |U| \) do
4: Obtain \( \pi'_k \) the k-th unique probability of the set \( U \)
5: Calculate \( T_k \)
6: Draw \( x_k \sim Bin(T_k, \pi'_k) \)
7: countEdge=0
8: while countEdge < \( x_k \) do
9: Generate new edge \( E_{uv} \)
10: if \( E_{uv} \notin E \) then
11: countEdge++
12: \( E = E \cup \{E_{uv}\} \)
13: Return \( G = (V, E) \)
Group probability sampling, pseudocode

Algorithm Group Probability Sampling

1: \( V = \{1, \cdots, N_v\}, E = \{\} \)
2: Construct \( U \), the set of unique probability values \( \pi'_k \)
3: \textbf{for} \( k = 1; \ k++; \ k \leq |U| \) \textbf{do}
4: \textbf{for} \( k = 1; \ k++; \ k \leq |U| \) \textbf{do}
5: \textbf{Calculate} \( T_k \)
6: \textbf{Draw} \( x_k \sim Bin(T_k, \pi'_k) \)
7: \textbf{countEdge} = 0
8: \textbf{while} \( \text{countEdge} < x_k \) \textbf{do}
9: \textbf{Generate new edge} \( E_{uv} \)
10: \textbf{if} \( E_{uv} \notin E \) \textbf{then}
11: \textbf{countEdge}++
12: \( E = E \cup \{E_{uv}\} \)
13: \textbf{Return} \( G = (V, E) \)

Uniform sampling among group of edges
avoid sampling unbias

The rejection process
avoid multigraphs
(space of graphs= \( G_o^K \))
**Algorithm**  Group Probability Sampling

1: \( V = \{1, \cdots, N_v\}, E = \{\} \)
2: Construct \( U \), the set of unique probability values \( \pi'_k \)
3: for \( k = 1; k ++ ; k \leq |U| \) do
4: Obtain \( \pi'_k \) the \( k \)-th unique probability of the set \( U \)
5: Calculate \( T_k \)
6: Draw \( x_k \sim Bin(T_k, \pi'_k) \)
7: countEdge=0
8: while \( \text{countEdge} < n \) do
9: \( \text{countEdge}++ \)
10: \( E = E \cup \{E_{uv}\} \)
11: \( E = E \cup \{E_{uv}\} \)
12: Return \( G = (V, E) \)

**Group probability KPGM samples from the original probability distribution**

**Theorems 1 and 2**
Group probability sampling, pseudocode

Algorithm: Group Probability Sampling

1: \( V = \{1, \ldots, N_v\}, E = \{\}\) 
2: Construct \( U \), the set of unique probabilities, \( u_k \) 
3: for \( k \) = 1 to \( K \) do 
4: \( \text{Omit } U \) 
5: Calculate \( T_k \) 
6: Draw \( x_k \sim \text{Bin}(T_k, \pi'_k) \) 
7: countEdge = 0 
8: while \( x_k > 0 \) do 
9: \( \text{countEdge} += 1 \) 
10: \( E = E \cup \{E_{uv}\} \) 
11: Return \( G = (V, E) \)
The CDF of **group probability KPGM** has a KS distance of 0.03% with respect to the analytical CDF of KPGM.
Group probability sampling, time generation

KPGM generates a network, in Matlab, with approximately 15 millions edges in 25 seconds.
Group probability sampling, time generation

Our C++ implementation generates a network with approximately half billion edges, in 234 seconds.

KPGM generates a network, in Matlab, with approximately 15 millions edges in 25 seconds.

---

**Graph Details:**
- **Y-axis:** Time in seconds
- **X-axis:** Expected number of edges $\times 10^6$
- **Legend:**
  - Red line: original
  - Dashed black line: group probability
  - Green dashed line: edge by edge
  - Blue dashed line: edge rejection

---

**Notes:**
- The graph illustrates the time taken to generate networks of different sizes using various methods.
- The C++ implementation is significantly faster than the Matlab implementation for larger networks.
Group probability sampling gives the same results than KPGM and the edge by edge misestimate the clustering coefficient distribution.

**GRQC:** $N_v=5,242$ $N_e=28,980$
MIXED KRONECKER PROBABILITY GRAPH MODEL
mixed KPGM generation

mixed KPGM generates firsts $\ell$ levels using KPGM, new kronecker multiplications ties edges increasing variance

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

$$K = 6 \quad \ell = 4$$

$$\Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix}$$
mixed KPGM generation

mixed KPGM generates firsts $\ell$ levels using KPGM, new kronecker multiplications ties edges increasing variance

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

$$K = 6 \quad \ell = 4$$

$$\Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix}$$
mixed KPGM generation

mixed KPGM generates firsts $\ell$ levels using KPGM, new kronecker multiplications ties edges increasing variance

$$
\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}
$$

$$
K = 6 \quad \ell = 4
$$

$$
\Theta = \begin{bmatrix}
0.99 & 0.90 \\
0.60 & 0.30
\end{bmatrix}
$$
mixed KPGM generation

New levels are sampled before subsequent Kronecker multiplications until the final network is generated.

\[ \Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \]

\[ K = 6 \quad \ell = 4 \]

\[ \Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix} \]
New levels are sampled before subsequent Kronecker multiplications until the final network is generated.

\[ \Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \]

\[ K = 6 \quad \ell = 4 \]

\[ \Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix} \]
mixed KPGM generation

New levels are sampled before subsequent Kronecker multiplications until the final network is generated.

\[ \Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \]

\[ K = 6 \quad \ell = 4 \]

\[ \Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix} \]
mixed KPGM generation

New levels are sampled before subsequent Kronecker multiplications until the final network is generated.

\[
\Theta = \begin{bmatrix}
\theta_{11} & \theta_{12} \\
\theta_{21} & \theta_{22}
\end{bmatrix}
\]

\[
K = 6 \quad \ell = 4
\]

\[
\Theta = \begin{bmatrix}
0.99 & 0.90 \\
0.60 & 0.30
\end{bmatrix}
\]
New levels are sampled before subsequent Kronecker multiplications until the final network is generated.

$$\Theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

$$K = 6 \quad \ell = 4$$

$$\Theta = \begin{bmatrix} 0.99 & 0.90 \\ 0.60 & 0.30 \end{bmatrix}$$
mixed KPGM

- mixed Kronecker Product Graph Model
  - Low number of parameters.
  - Capture important network characteristics such as: degree, diameter and cluster coefficient.
mixed KPGM

- mixed Kronecker Product Graph Model
  - Low number of parameters.
  - Capture important network characteristics such as: degree, diameter and cluster coefficient.

- There is no fast implementation, because of the tied edge process.
- Naive implementations of their sampling process have complexity $O(N\sqrt{2})$, which is intractable for large networks.
mixed KPGM

- mixed Kronecker Product Graph Model
  - Low number of parameters.
  - Capture important network characteristics such as: degree, diameter and cluster coefficient.

- There is no fast implementation, because of the tied edge process.
- Naive implementations of their sampling process have complexity $O(N\sqrt{V})$, which is intractable for large networks.

- We develop the first scalable sampling process, by independently sampling groups of edges with the same probability.
Group probability sampling

- Current mKPGM algorithm interleave Kronecker multiplication and Bernoulli sampling

- Group probability sampling for a layer $k$ in mKPGM
  - Aggregate events for unique probability combination $U = \{\theta_{11}, \theta_{12}, \ldots, \theta_{bb}\}$
  - Sample the aggregate event from a Binomial distribution
  - Place edges uniformly at random among positions with value $\pi_k$
Group probability sampling

- Current mKPGM algorithm interleave Kronecker multiplication and Beroulli sampling.
- Group probability mKPGM samples from the original probability distribution.
- Aggregate events for unique probability combination: \( \{ \theta_{11}, \theta_{12}, \ldots, \theta_{bb} \} \)
- Sample the aggregate event from a Binomial distribution.
- Place edges uniformly at random among positions with value \( \pi_k \).

Theorems 3 and 4

Corollary 1

\[
\begin{array}{c|c|c|c|c|c|c}
\pi_{ij} & T_k & Bin(T_k, \pi_{ij}) & Bin(N_{e_{k-1}}, \theta_{11}) & Bin(N_{e_{k-1}}, \theta_{12}) & Bin(N_{e_{k-1}}, \theta_{21}) & Bin(N_{e_{k-1}}, \theta_{22}) \\
\hline
\theta_{11} & N_{e_{k-1}} & \text{Binomial} & \text{Binomial} & \text{Binomial} & \text{Binomial} \\
\theta_{12} & N_{e_{k-1}} & \text{Binomial} & \text{Binomial} & \text{Binomial} & \text{Binomial} \\
\theta_{21} & N_{e_{k-1}} & \text{Binomial} & \text{Binomial} & \text{Binomial} & \text{Binomial} \\
\theta_{22} & N_{e_{k-1}} & \text{Binomial} & \text{Binomial} & \text{Binomial} & \text{Binomial} \\
\hline
\end{array}
\]
Group probability sampling

- Current mKPGM algorithm interleave Kronecker multiplication and Bernoulli sampling.

- Group probability mKPGM samples from the original probability distribution
  - Aggregate events for unique probability combination
  - Sample the aggregate event from a Binomial distribution
  - Place edges uniformly at random among positions with value \( \pi_k \)

Theorems 3 and 4
Corollary 1

Group probability mKPGM avoid explicit Kronecker multiplications, resulting in algorithm with time complexity \( O(b^2 \cdot N_e) = \tilde{O}(N_e) \)
group probability mKPGM has a KS distance of 0.08% with respect to the empirical mKPGM.

group probability mKPGM can generate a network with approximately 15 millions edges in 5 seconds.
Group probability sampling, results

**Group probability mKPGM**

- Has a KS distance of 0.08% with respect to the empirical mKPGM.
- Can generate a network with approximately 15 millions edges in 5 seconds.

Our **C++** implementation generates a network with over one billion edges, in 402 seconds.
Group probability sampling, results

**Email:** $N_v=6,503$ $N_e=14,756$

**group probability mKPGM** generates similar networks than the original mKPGM sampling process.
Conclusions

• Previous scalable algorithms for KPGM do not reproduce the probability distribution over the space of graphs
Conclusions

• Previous scalable algorithms for KPGM do not reproduce the probability distribution over the space of graphs

• We develop a new representation of Kronecker models, facilitating the first scalable and exact sampling algorithms for KPGM and mKPGM
  – Reproduce the probability distribution over the space of graphs (KS distances less than 0.1%)
  – Replicate the characteristics of the networks generated by the original algorithms
  – Efficiently generate networks with time complexity $\tilde{O}(N_e)$.
  – Notably, we can generate a network with over 16 millions nodes and over 1 billion edges in 402 seconds.
Thanks for your attention

Code available at
www.cs.purdue.edu/homes/smorenoa/codes.html