

Sampling Bipartite Graphs

Stephen Chestnut Rico Zenklusen

January 23, 2014

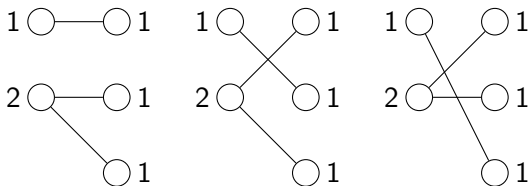
Sampling Bipartite Graphs

Instance

sequences $r = (r_i)_{i=1}^m$ and $c = (c_j)_{j=1}^n$

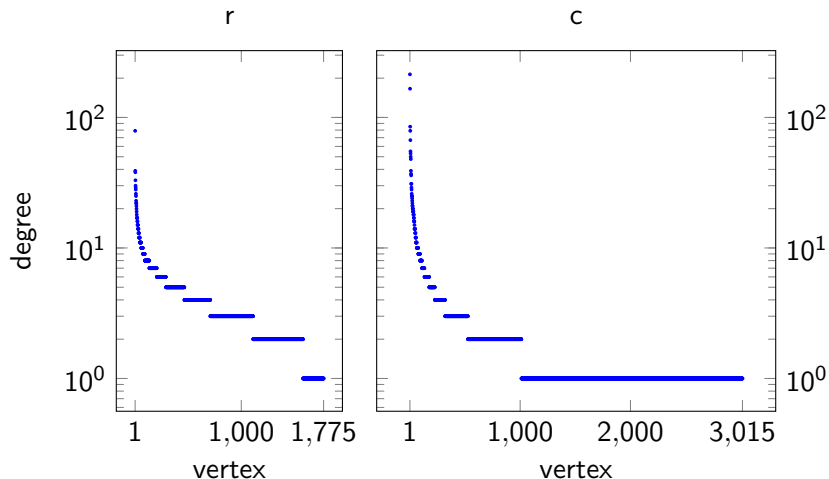
vertices $X = \{u_i\}_{i=1}^m$ and $Y = \{v_j\}_{j=1}^n$

goal Randomly sample a bipartite graph $G = (X \cup Y, E)$ that satisfies $\deg(u_i) = r_i$ and $\deg(v_j) = c_j$, for all i, j



$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Typical Big Instance We Want to Solve



$m = 1775$ users of a forum with $n = 3015$ threads

Prior Work

Algorithms

Regular sequences: Kannan, Tetali & Vempala (1997)

All sequences: Bezáková, Bhatnagar & Vigoda (2007)

↓ **Practically useful** ↓

Very very sparse: Wormald (1984)

c is bounded: Miller & Harrison (2012)

↑ **Theoretical Guarantee** ↑

Old practitioner's standard: Chen, Diaconis, Holmes & Liu (2005)

New practitioner's standard: Miller & Harrison (2013)

Negative Results

Limits on KTV: Bezáková, Bhatnagar & Randall (2009)

CDHL may fail: Bezáková, Sinclair, Štefanovič & Vigoda (2011)

Configuration model and edge weights

initialize Create r_i or c_i “endpoints” for each vertex

repeat Select two endpoints (one at random)

Add an edge between the vertices

Remove the endpoints

reject If the graph is not simple

○ 1

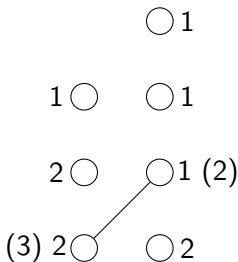
1 ○ ○ 1

2 ○ ○ 2

3 ○ ○ 2

Configuration model and edge weights

- initialize** Create r_i or c_i “endpoints” for each vertex
- repeat** Select two endpoints (one at random)
 - Add an edge between the vertices
 - Remove the endpoints
- reject** If the graph is not simple



$$\frac{2}{6}$$

Configuration model and edge weights

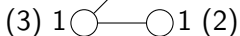
initialize Create r_i or c_i “endpoints” for each vertex

repeat Select two endpoints (one at random)

Add an edge between the vertices

Remove the endpoints

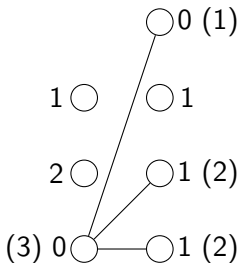
reject If the graph is not simple



$$\frac{2}{6} \cdot \frac{2}{5}$$

Configuration model and edge weights

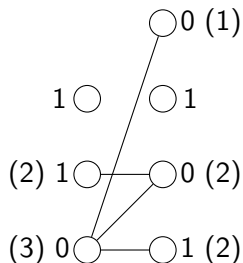
- initialize** Create r_i or c_i “endpoints” for each vertex
- repeat** Select two endpoints (one at random)
 - Add an edge between the vertices
 - Remove the endpoints
- reject** If the graph is not simple



$$\frac{2}{6} \cdot \frac{2}{5} \cdot \frac{1}{4}$$

Configuration model and edge weights

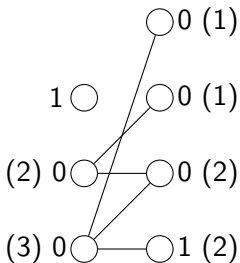
- initialize** Create r_i or c_i “endpoints” for each vertex
- repeat** Select two endpoints (one at random)
 - Add an edge between the vertices
 - Remove the endpoints
- reject** If the graph is not simple



$$\frac{2}{6} \cdot \frac{2}{5} \cdot \frac{1}{4} \cdot \frac{1}{3}$$

Configuration model and edge weights

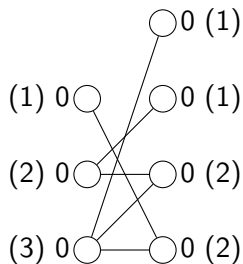
- initialize** Create r_i or c_i “endpoints” for each vertex
- repeat** Select two endpoints (one at random)
 - Add an edge between the vertices
 - Remove the endpoints
- reject** If the graph is not simple



$$\frac{2}{6} \cdot \frac{2}{5} \cdot \frac{1}{4} \cdot \frac{1}{3} \cdot \frac{1}{2}$$

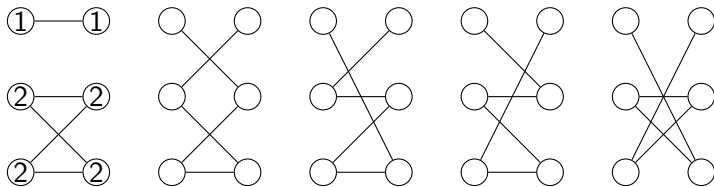
Configuration model and edge weights

- initialize** Create r_i or c_i “endpoints” for each vertex
- repeat** Select two endpoints (one at random)
 - Add an edge between the vertices
 - Remove the endpoints
- reject** If the graph is not simple



$$\frac{2}{6} \cdot \frac{2}{5} \cdot \frac{1}{4} \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{1} = \frac{2!2!1!1!}{6!} = \frac{(\prod_i r_i!)}{(\sum_i r_i)!}$$

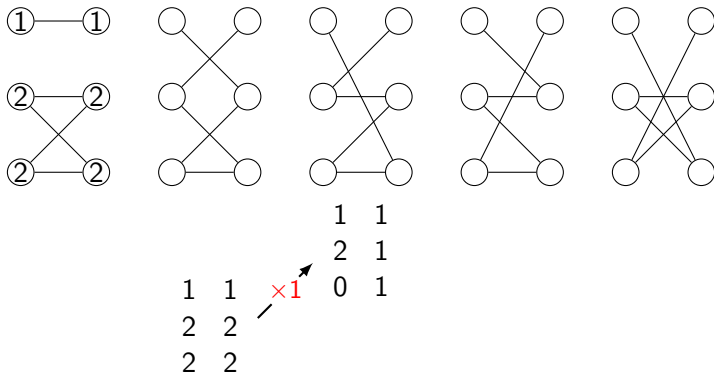
Dynamic Programming Example



1 1
2 2
2 2

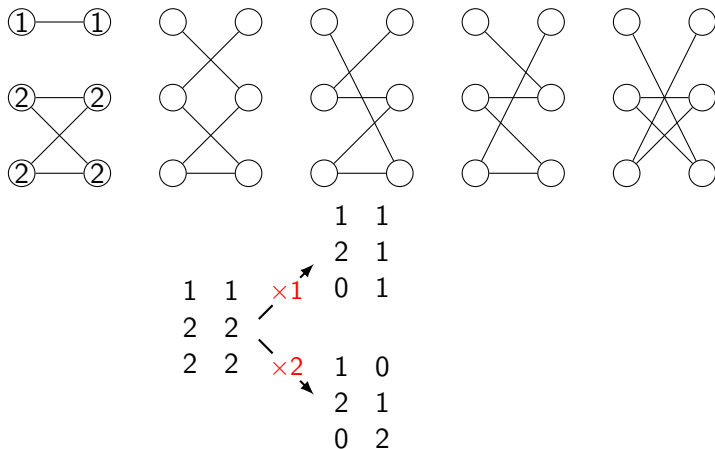
- Only need to store the #vertices of each remaining #endpoints
- Initialization complexity is $O(mn^{\max_j c_j})$

Dynamic Programming Example



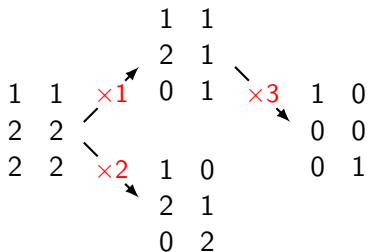
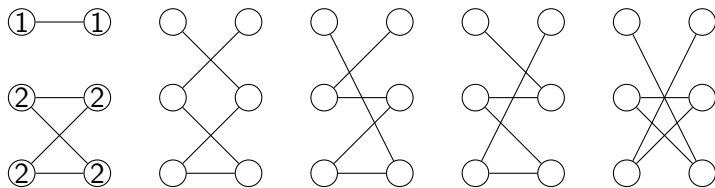
- Only need to store the #vertices of each remaining #endpoints
- Initialization complexity is $O(mn^{\max_j c_j})$

Dynamic Programming Example



- Only need to store the #vertices of each remaining #endpoints
- Initialization complexity is $O(mn^{2 \max_j c_j})$

Dynamic Programming Example



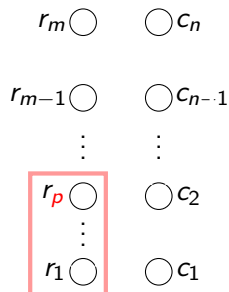
- Only need to store the #vertices of each remaining #endpoints
- Initialization complexity is $O(mn^{2 \max_j c_j})$

Dynamic Programming + Configuration Model

Hybrid Algorithm

choose Integer $p \geq 1$

1. Dynamic programming of the p special vertices
2. Configuration model for remainder



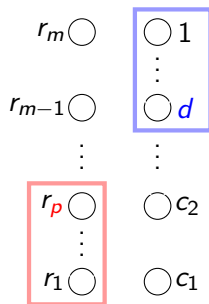
Effect

Samples from configuration model **conditionally given** no repeated edges to p special vertices.

Approximate Dynamic Programming + Config. Model

Hybrid Approximation Algorithm

- choose Integers $p \geq 1$ and $d \geq 0$
- approx If $\deg(v_j) > d$ approx. #endpoints of v_j by c_j
- Dynamic programming for p special vertices
 - Rejection step
 - Configuration model for remainder



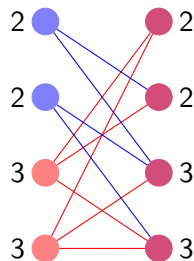
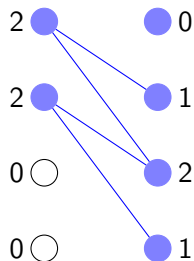
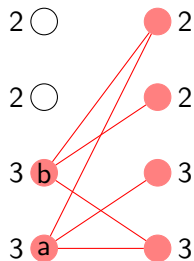
Analysis

complexity Initialization complexity is $O(pn^{2(d+1)})$

output Sample distribution is exactly uniform

Example $p = d = 2$

$$\frac{\overset{\text{vertex a}}{(3 \cdot 3 \cdot 2)} \overset{\text{vertex b}}{(3 \cdot 2 \cdot 1)}}{1980} \underset{\text{accept}}{\binom{2}{3}} \overset{\text{config. model}}{\binom{2!^3}{4!}} = \frac{3!^2 2!^4}{1980 \cdot 4!}$$



Example continued

	$P\left(\begin{array}{c} \bullet \\ \bullet \\ \bullet \\ \bullet \end{array} \begin{array}{c} \bullet \\ \bullet \\ \bullet \\ \bullet \end{array}\right)$	$P(\text{Reject})$
Configuration Model	$\frac{3!^4 2!^4}{10!} \approx 0.0057$	0.81
Hybrid algorithm	$\frac{3!^2 2!^4}{1980 \cdot 4!} \approx 0.012$	0.59
Uniform	$\frac{1}{34} \approx 0.029$	0

Conclusions

- Hybrid algorithm extends the Configuration Model and Dynamic Programming algorithms.
- Doesn't work (yet) for the big instance shown earlier
- Best chance is with sequences with many low degrees, a few high degrees