SAND2018-8934C

Generating Massive Random Graphs that Mimic Real Data

George Slota (Rensselaer Polytechnic Institute)

Jonathan Berry (SNL)

Cynthia A. Phillips (Sandia National Laboratories)

Siva Rajamanickam (SNL)

\

National Nuclear Security Administration

Rensselaer

CCR Center for Computing Research

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Motivation

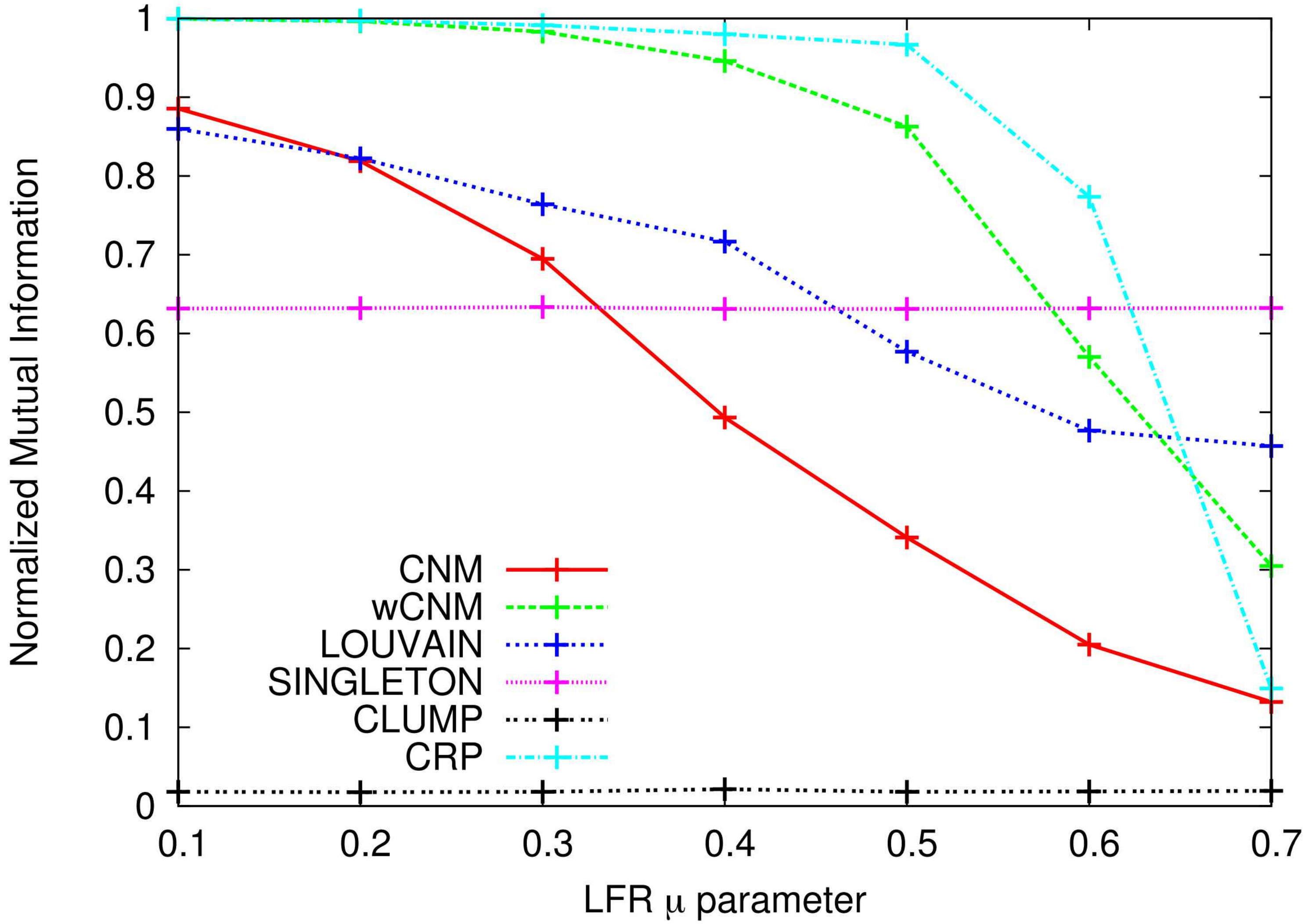
Better evaluate community detection algorithms processing O(Billion)-sized Graphs on HPC resources

- *Small-scale state-of-the-art: "LFR"*
  - Lancichinetti, Fortunato, Radicchi, 2008
  - With > 1600 citations, this is a de facto standard
  - Generates approximate ground truth to test against
  - Has a tunable parameter for community coherence: μ
  - Limited scalability: best implementation takes ~17hrs to generate O(1B) edges (Hamann et. al. 2017)
- *Large-scale state-of-the-art*
  - Without a reliable ground truth, parallel algorithms test with modularity or similar measures

Goal: evaluate at HPC scale against ground truth

Typical Comparison Plot

- For Normalized Mutual Information



Overview

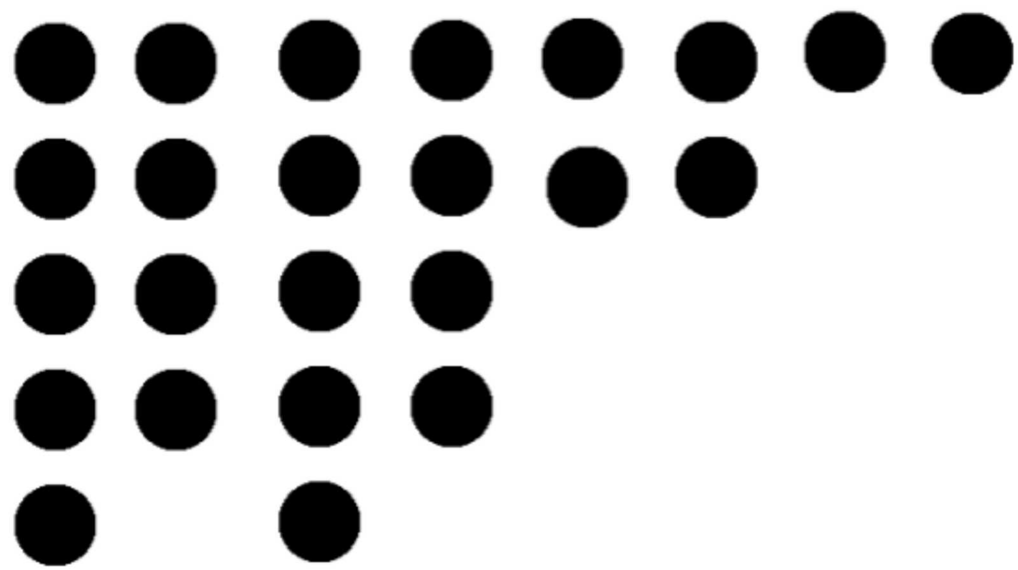
Primary results of this work:

- We develop a novel method for generating large-scale graphs with a tunable ground truth community structure
- We utilize the scalable BTER generator (Kolda et al., 2014) as a core step
- Our approach generates large-scale community benchmarking graphs at a rate of 1B edge/minute on KNL
  – **Orders-of-magnitude faster than state-of-the-art**

BTER: Block Two-Level Erdös-Réyni Graph Generator

- Step 0: Input degree (n_d) and clustering coefficient (c_d) distributions

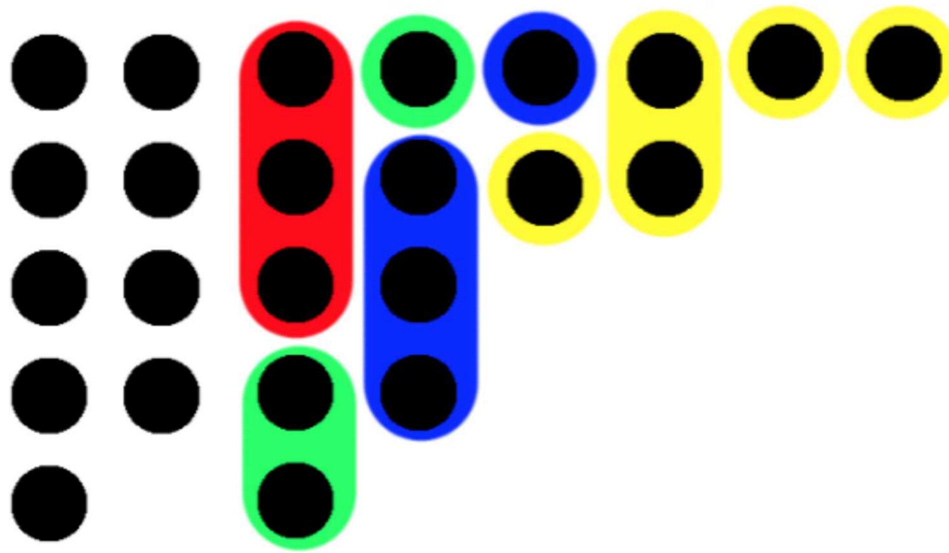
d	1	2	3	4	5	6	7
n_d	9	5	4	2	2	1	1
c_d	--	0.6	0.4	0.2	0.1	0.1	0.1



BTER: Block Two-Level Erdös-Réyni Graph Generator

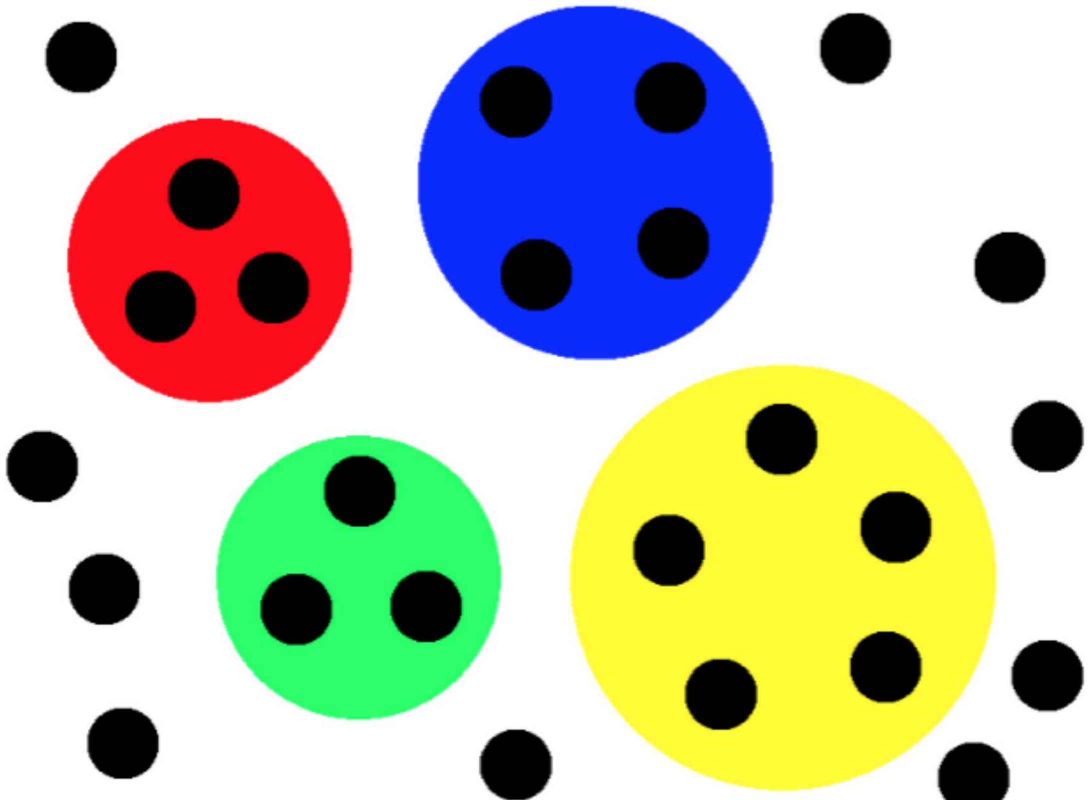
- Step 0: Input degree (n_d) and clustering coefficient (c_d) distributions
- Step 1: With ordered degree sequence, group $d+1$ vertices v of degree $d(v) >= d$ into *affinity blocks*

d	1	2	3	4	5	6	7
n_d	9	5	4	2	2	1	1
c_d	--	0.6	0.4	0.2	0.1	0.1	0.1



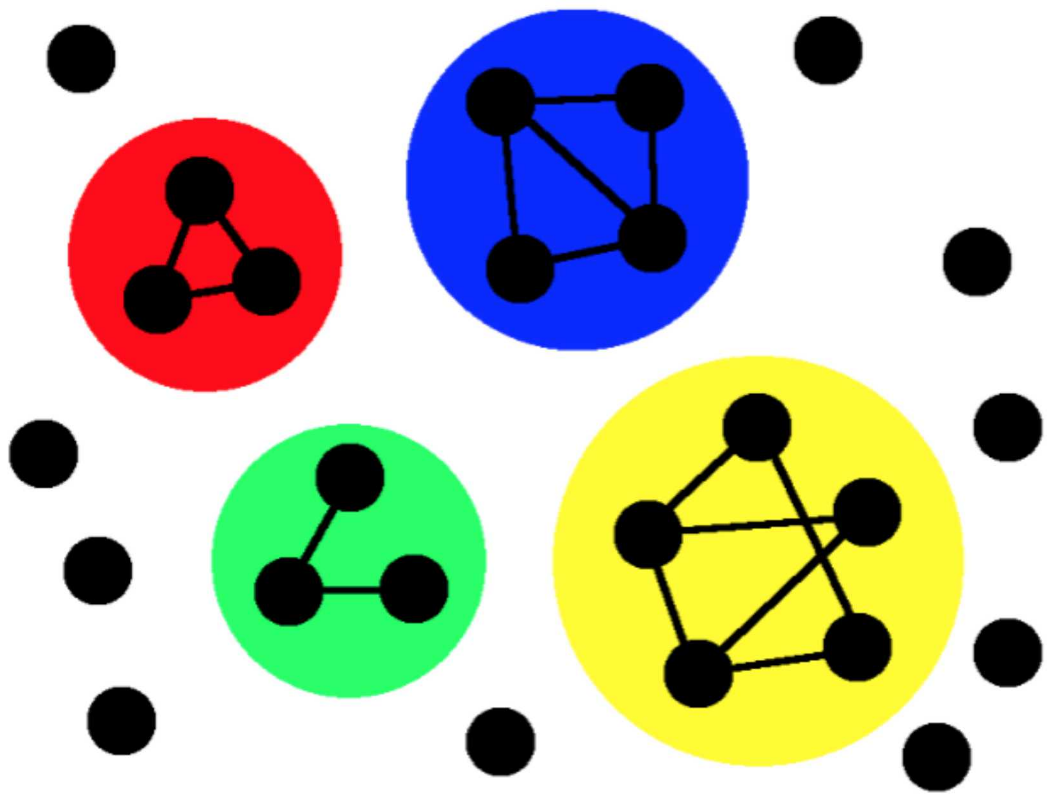
BTER: Block Two-Level Erdös-Réyni Graph Generator

- Step 0: Input degree (n_d) and clustering coefficient (c_d) distributions
- Step 1: With ordered degree sequence, group $d+1$ vertices v of degree $d(v) >= d$ into *affinity blocks*



BTER: Block Two-Level Erdös-Réyni Graph Generator

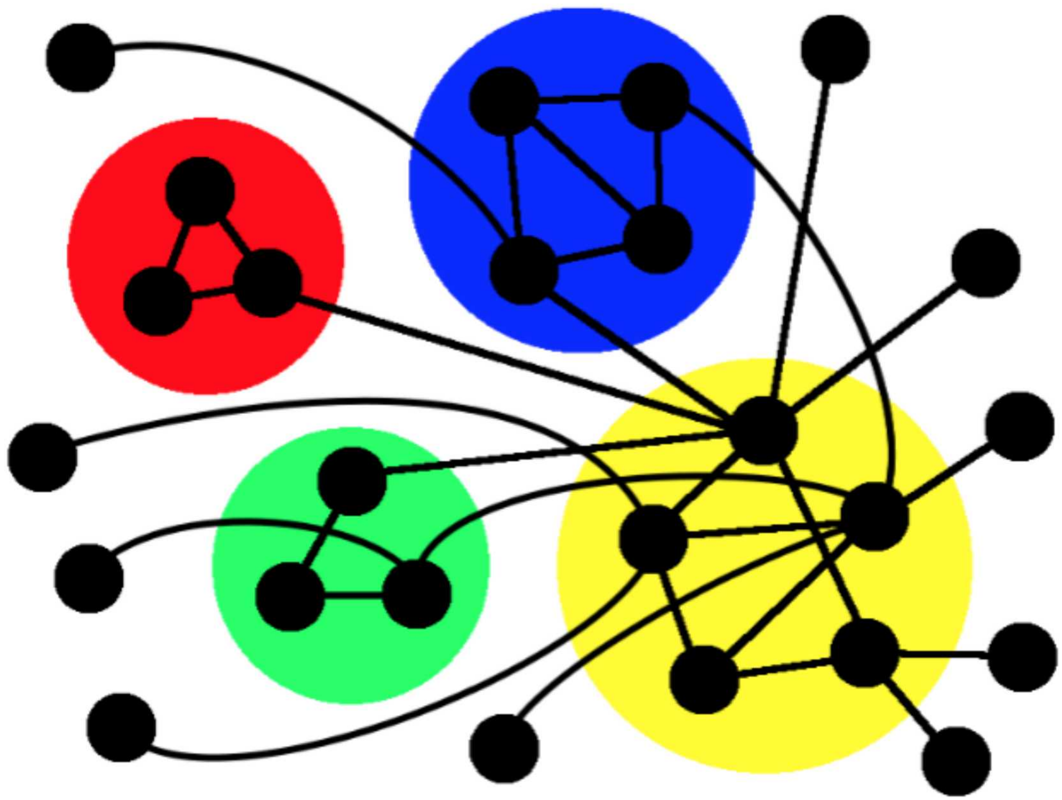
- Step 0: Input degree (n_d) and clustering coefficient (c_d) distributions
- Step 1: With ordered degree sequence, group $d+1$ vertices v of degree $d(v) >= d$ into *affinity blocks*
- Step 2: Use Erdös-Rényi probability $p_d = \sqrt[3]{c_d}$ to create intra-block edges via $G(n, p)$ process



Sandia National Laboratories

BTER: Block Two-Level Erdös-Réyni Graph Generator

- Step 0: Input degree (n_d) and clustering coefficient (c_d) distributions
- Step 1: With ordered degree sequence, group $d+1$ vertices v of degree $d(v) >= d$ into *affinity blocks*
- Step 2: Use Erdös-Rényi probability $p_d = \sqrt[3]{c_d}$ to create intra-block edges via $G(n, p)$ process
- Step 3: Create inter-block edges via Chung-Lu process



Initial Thoughts

- Native μ for BTER input

$$\mu_{\text{BTER}} = \frac{\sum_{d\in D} d \cdot n_d(1 - \sqrt[3]{c_d})}{\sum_{d\in D} d \cdot n_d} = 1 - \frac{\sum_{d\in D} d \cdot n_d(\sqrt[3]{c_d})}{2m}$$

- Where n_d = # nodes of degree d and c_d is CC for degree d
- Transform to new clustering coefficient (CC) distribution:

$$c'_d = \left(\frac{(1-\mu_g)}{R}\right)^3 \cdot c_d$$

- This gives a desired goal LFR parameter μ_g.
- Issue: This can make some $c_d > 1$

Sandia National Laboratories

Our Implementation for Community Detection. wBTER = wrapped BTER

How we wrap the baseline BTER process for generating graphs for community detection benchmarking:

- Treat affinity blocks as ground truth communities
- We have a *native* μ_n, based on ratio of inter- to intra-block edges generated from the original distributions
- Can shift μ_n to some target goal μ_g via a Linear Program solve (to be described) – we use Pyomo and CBC
- Our BTER implementation: fully-parallelized in shared-memory with OpenMP/C++

Linear Program: Shifting the Native μ of a Graph's CC Distribution

Minimally shift the input clustering coefficient (CC) distribution such that the output graph has a desired goal μ_g considering both definitions:

$$\mu_g = \frac{1}{N}\sum_d \frac{d_{inter}}{d} \qquad \mu_g = \frac{1}{2M}\sum_d n_d d_{inter}$$

$$\begin{aligned}
\text{minimize} \quad & \sum_d |\hat{p}_d - p_d| \\
\text{subject to} \quad & \sum_d n_d \hat{p}_d = N(1-\mu_g) \\
& \sum_d d n_d \hat{p}_d = 2M(1-\mu_g) \\
& 0 \le \hat{p}_d \le 1 \\
\text{output} \quad & \hat{c}_d = \hat{p}_d^3
\end{aligned}$$

- p_d is $G(n,p)$ probabilities per degree from CC distribution c_d, $p_d = \sqrt[3]{c_d}$
- $\hat{p}_d$ is output probabilities to get new CC distribution $\hat{c}_d$, $\hat{c}_d = \hat{p}_d^3$
- n_d is degree distribution, n vertices of d degree
- d_{inter} is expected number of inter-community edges for vertex of degree d
- N is number of vertices in graph, M is number of edges

Experimental Setup
Test System and Test Graphs

Test System: *Bowman* at Sandia Labs – each node has a KNL with 68 cores, 96 GB DDR, and 16 GB MCDRAM

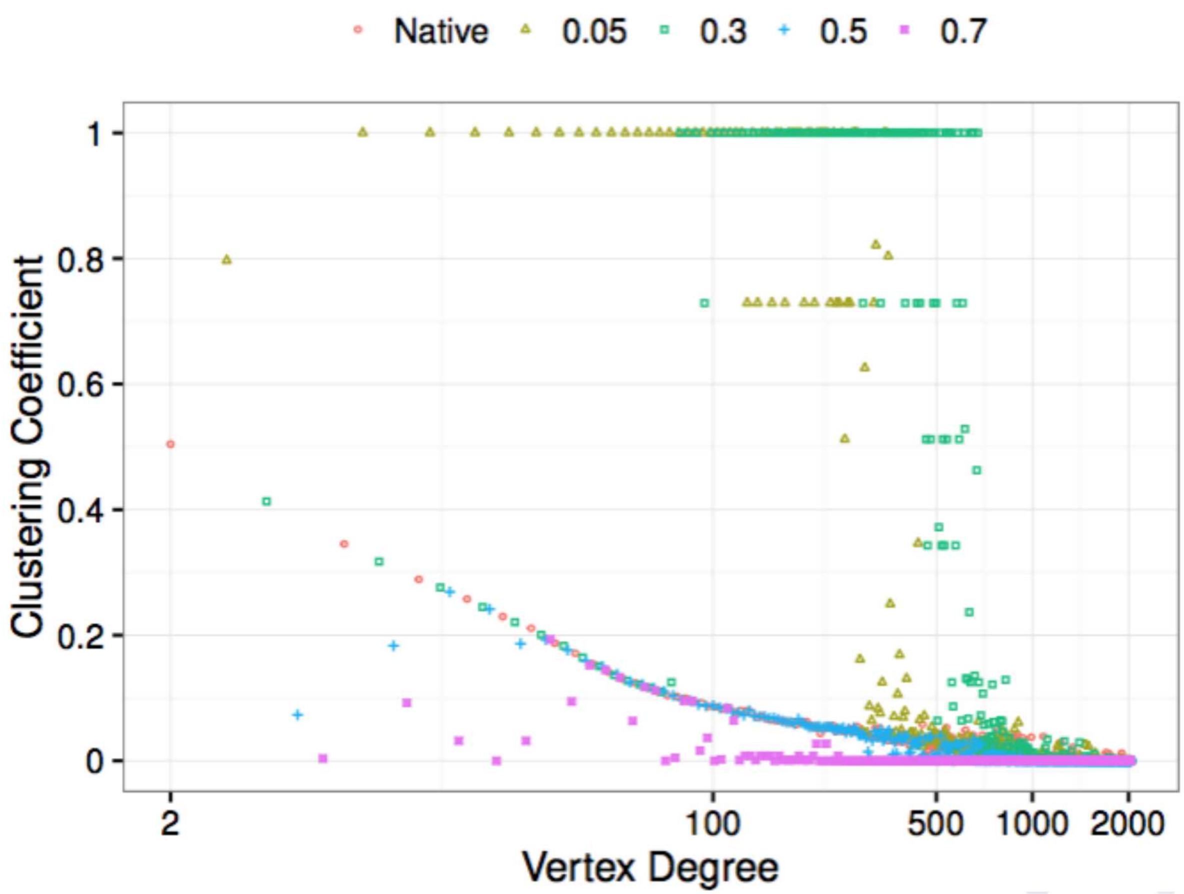
Test Graphs:

Network	n	m	$davg$	$dmax$	$\tilde{D}$
LJ-fp	4.2 M	27 M	18	20 K	18
uk-2002	18 M	261 M	28	195 K	28
Wikilinks	26 M	332 M	23	39 K	170
RMAT_26	67 M	1.1 B	16	6.7 K	8
Friendster	66 M	1.8 B	27	5.2 K	34

Graphs are from the SNAP, Koblenz, and LAW databases.
LiveJournal-fp is a parsed version of LiveJournal from SNAP.

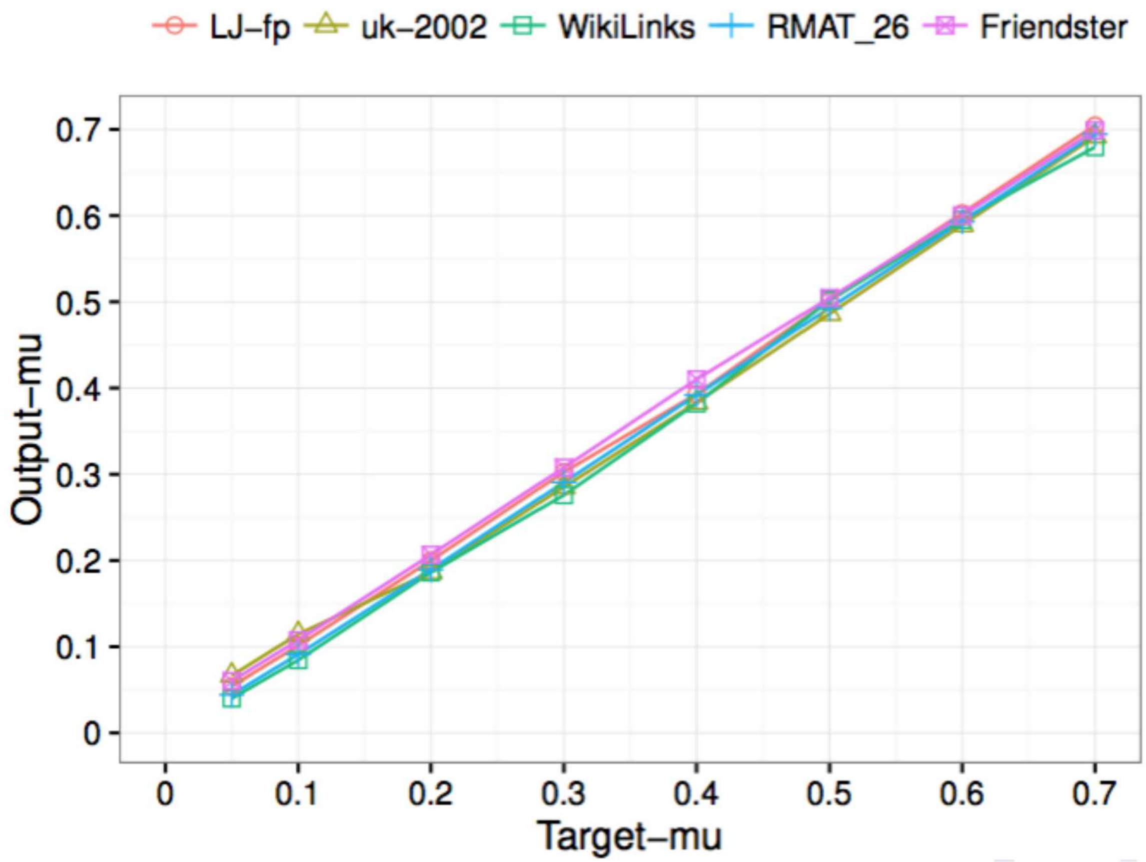
Shifting Distribution

- Only every 5^{th} value plotted for better visualization
- Generally, distribution is most "accurate" near *native* μ
- Better *smoothing* of distribution via LP constraints is future work



Hitting Target μ
Accuracy of LP for Generating Desired μ

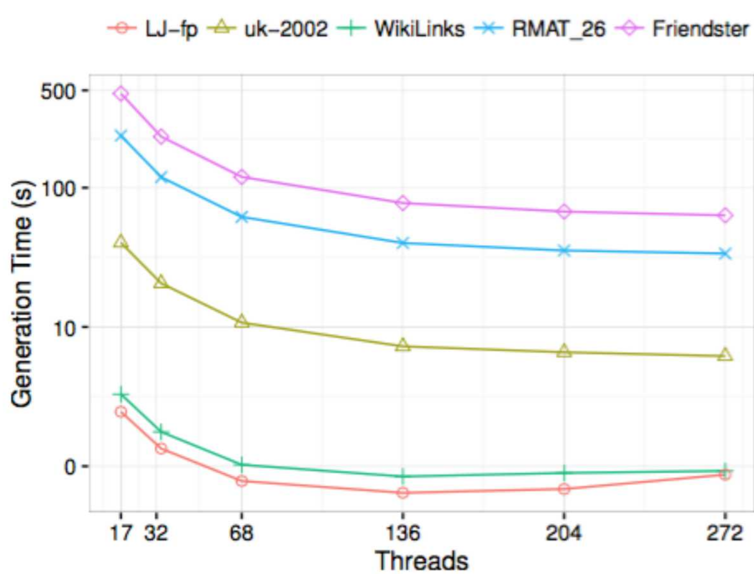
- Generation accuracy is comparable to LFR
- Less than 5% error in most instances
- Error is greatest at lower μ targets

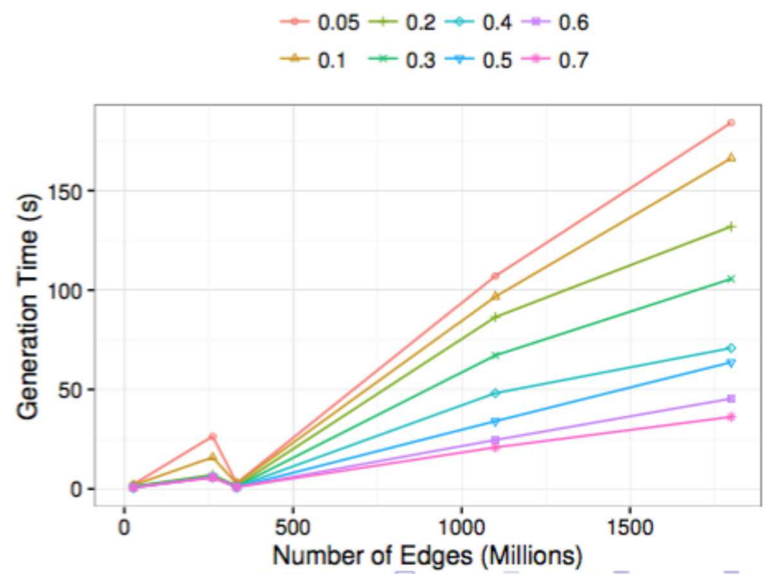


Sandia National Laboratories

Generation Time vs. Target μ
(Left) Time vs μ. (Right) Time vs graph scale

- Strong scaling generally good up to 2 threads/core
- Time decreases with increasing μ, due to *coupon collectors* edge generation scaling - higher CC requires more attempts for each edge
- Generation time a function of scale and complexity (max degree)
- Average ~2 minutes for 1.8B unique edges
  - Original BTER code: ~4 min. for 1.2B edges on 32 node Hadoop cluster
  - Fastest LFR implementation: 17 hours for 1B edges in shared-memory





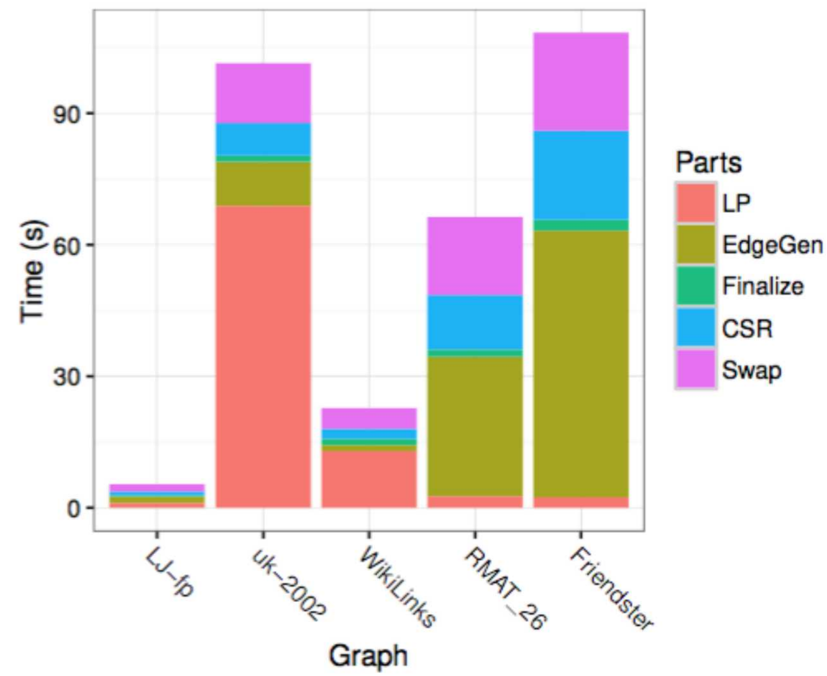
Sandia National Laboratories

A Note on BTER Assortativity

- An issue with our approach so far is the degree homogeneity of communities
- We propose the following addition:
  - Consider intra-comm edge count of each vertex
  - Permute community assignments of all vertices with same count
  - **Observation**: won't affect μ, de-homogenizes communities in terms of degree
- This approach might also be applied to baseline BTER generation

Timing Breakdown
Full wBTER with Community Permutation μ=0.5

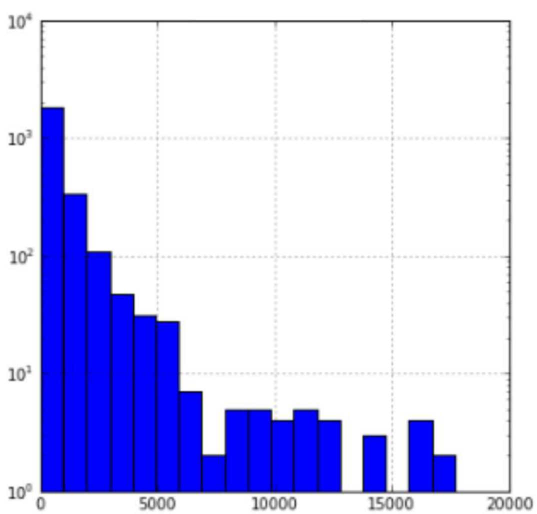
- Time costs of major wBTER steps with community assignment permutation
- **Work Complexity**: $d = D_{max}, n = |V|, m = |E|$
    - LP: expected to scale as $O(d \log d)$
    - EdgeGen: $O(m \log d)$
    - Finalize: $O(n + m)$
    - CSR: $O(n + m)$
    - Swap: $O(n \log n + m)$



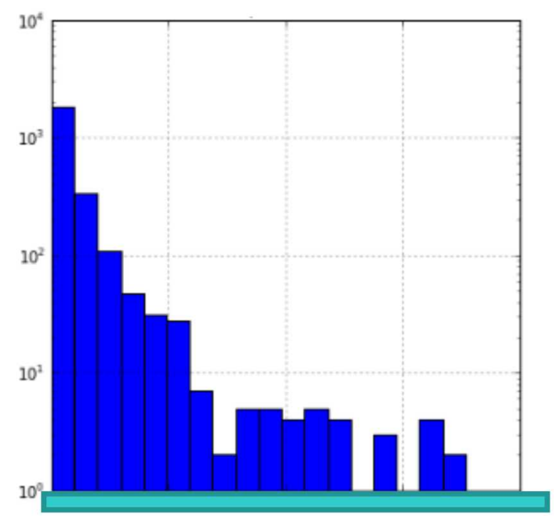
LP: linear program
EdgeGen: primary BTER phase
Finalize: remove 0-degree vertices & cleanup
CSR: create graph representation
Swap: community degree permutation

Open: Scaling a Graph Up

- Goal: Make a graph that is 2x the size of an example graph
- Problem: Given a discrete distribution, make it 2x the size, but "looks" the same



Effect on counts?



2x the domain

https://stats.stackexchange.com/questions/205503/do-histograms-need-to-be-sorted-to-determine-if-the-data-follow-a-power-law-dist

Sandia National Laboratories

Initial Thoughts

- Change of variables
- Graph generation process
  - Run node addition process “twice as long”
  - Deliberately do not initially consider densification

Can we scale graphs down too for faster initial debugging/testing?

Future Work

- Figure out how to do graph scaling (bigger, smaller)
- Better develop LP to reduce noise in output clustering-coefficient distributions
- Generation methods for hierarchical communities