Efficient Pattern Detection in Web-Scale Graphs by Subcore-Tree Decomposition and Subset Scanning

Daniel B. Neill, Ph.D.
Event and Pattern Detection Laboratory
Carnegie Mellon University
E-mail: neill@cs.cmu.edu

Joint work with Chunpai Wang, Feng Chen, and Daniel Hono (SUNY Albany).

Motivation for this work

- “Scaling up” pattern detection to large, real-world graphs
  - Online social networks such as Twitter, Amazon purchases, etc.
- Extending our past work on Twitter event detection (Chen and Neill, 2014, 2015).
  - Can accurately detect, and sometimes predict, specific event types of interest (civil unrest, rare disease outbreaks, human rights…)
  - Dramatically reduces effective graph size– but what if we don’t know what we’re looking for *a priori*?

Motivation for this work

Identifying and addressing challenges of big graphs:

Computational: efficiently identifying the highest-scoring connected subgraph

Challenging because there are $2^{|V|}$ subsets of nodes.

Performance metrics:

Run time
(as a function of graph size and other characteristics)

Approximation ratio
(detected subgraph score / maximum subgraph score)

Statistical: distinguishing affected subgraphs from random noise

Challenging because there are many ($\alpha|V|$) significant nodes even under H_0.

Performance metrics:

Detection power
(distinguishing graphs with and without an affected subgraph)

Precision and **recall**
(precisely identifying the affected subset of nodes)

Problem setting

- Standard problem setup, as in Chen and Neill (2014):
  - Known graph G = (V,E)
  - One or more quantities monitored at each graph node v_i → anomalousness of each node represented by empirical p-value p_i.
  - p_i ~ Uniform[0,1] under H_0.
  - Under $H_1(S)$, some subset of nodes $S \subseteq V$ will have a higher than expected number of low (significant) empirical p-values.
- Likelihood ratio score: $F(S) = P(Data \mid H_1(S)) / P(Data \mid H_0)$.
- Report the subset with highest score, $S^* = \arg\max_S F(S)$, and compute its statistical significance by randomization.

Nonparametric scan statistics

Number of nodes in S with p-values ≤α.

Subgraph

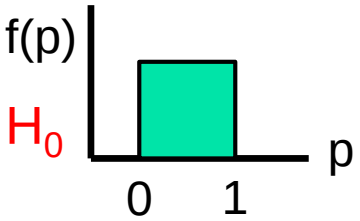
$$F(S) = \max_{\alpha \le \alpha_{max}} F_\alpha(S) = \max_{\alpha \le \alpha_{max}} \phi(\alpha, N_\alpha(S), N(S))$$

Significance level

Number of nodes in S

Berk-Jones (BJ) statistic:

$$\phi_{BJ}(\alpha, N_\alpha(S), N(S)) = N(S) K\left(\frac{N_\alpha}{N}, \alpha\right)$$



Kullback-Liebler divergence:

$$K(x, y) = x \log\left(\frac{x}{y}\right) + (1 - x) \log\left(\frac{1 - x}{1 - y}\right)$$

Real-world graphs

Very large and very sparse, with a distinct core-periphery structure.

Network	Vertices	Edges	Density	Core Nodes	Core Density	Tree Density
slashdot	82168	504230	1.49×10^{-4}	10591	4.61×10^{-3}	4.48×10^{-5}
twitter	81306	1342296	4.06×10^{-4}	17337	4.10×10^{-3}	1.76×10^{-4}
dblp	317080	1049866	2.09×10^{-5}	22093	5.57×10^{-4}	1.42×10^{-5}

1) We can use core-tree decomposition to partition the graph into a denser core and a low-treewidth periphery.

2) We can further detect communities or “sub-cores” within the core. These tend to have densities an order of magnitude larger than the core’s density.

Q1: How can we *efficiently* detect patterns in such large graphs?

Q2: How can we *accurately* detect patterns in such large graphs?

Web-Scale Subset Scan

Our current algorithmic approach, WSSS, first performs subcore-tree decomposition (core-tree + community detection).

We then employ this decomposition by:

1) greedily merging significant tree nodes into the core;
2) identifying high-scoring clusters within each subcore;
3) merging high-scoring clusters across subcores.

Step 1 exploits the low treewidth of the periphery and the fact that most tree nodes are adjacent to at least one core node.

Steps 2 and 3 first merge adjacent significant nodes in the core, then use **color coding**, an approximation algorithm for identifying the highest-scoring subset of cardinality <= |k|, requiring time $O(M \log N \times (2e)^k)$ rather than $O(N^k)$.

Since complexity scales rapidly with k, we use a small value (e.g., k = 3).

Each merged node can represent many nodes of the original graph.

Using k > 1 allows us to find subgraphs with some non-significant nodes.

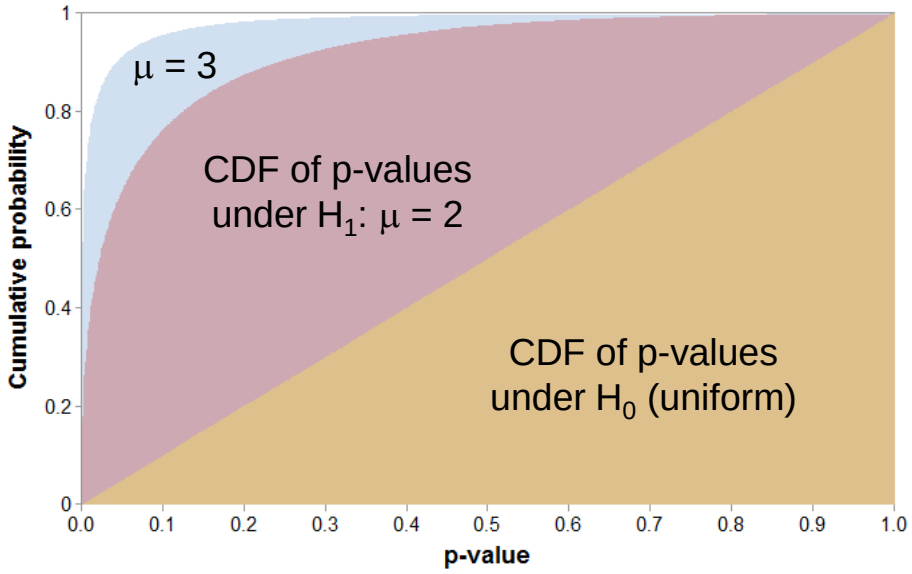
Evaluation setup

For each of the three real-world graphs, we created 150 datasets using that graph structure: 50 each of H_0, H_1 with $\mu = 2$, and H_1 with $\mu = 3$.

Under H_0, all p-values were drawn i.i.d. from U[0,1].

Under H_1, 1000 "affected" vertices were chosen by a random walk on the graph.

Affected p-values were drawn i.i.d. from the distributions shown here.



We used WSSS to identify the highest-scoring subgraph for each dataset, and evaluated run time, detection power, and accuracy (precision, recall, and overlap).

For each of the
world graphs, we
datasets using
structure: 50 ea
with $\mu = 2$, and

Under H_0, all p-
drawn i.i.d. fr

Under H_1, 1000 "affected" vertices were chosen by a random walk on the graph.

Affected p-values were drawn i.i.d. from the distributions shown here.

$$Precision = \frac{|S_{detected} \cap S_{injected}|}{|S_{detected}|}$$

$$Recall = \frac{|S_{detected} \cap S_{injected}|}{|S_{injected}|}$$

$$Overlap = \frac{|S_{detected} \cap S_{injected}|}{|S_{detected} \cup S_{injected}|}$$

We used WSSS to identify the highest-scoring subgraph for each dataset, and evaluated run time, detection power, and accuracy (precision, recall, and overlap).

Results (part 1)

The good news:

Efficiency: WSSS can find very high-scoring subgraphs in a reasonable time.

20 to 40 minutes for k = 1 and exact refinement;
1 to 6 hours for k = 3 and approximate refinement.

Detection power is close to perfect, i.e., we can accurately distinguish between graphs with and without affected subgraphs.

The bad news:

Precision of the detected subgraph is very low.

WSSS identifies a subgraph that is much larger and higher-scoring than the true affected subgraph, including many nodes that are significant just by chance.

This happens regardless of parameter settings, and for comparison methods such as the upper level set (ULS) scan as well.

What's happening: huge graphs have many significant p-values even under H_0, and enforcing connectivity does not sufficiently constrain the search from just picking out many significant values.

Results (part 1)

The good news:

Efficiency: WSSS can find very high-scoring subgraphs in a reasonable time.

20 to 40 minutes for k = 1 and exact refinement;
1 to 6 hours for k = 3 and approximate refinement.

Detection power is close to perfect, i.e., we can accurately distinguish between graphs with and without affected subgraphs.

The bad news:

Precision of the detected subgraph is very low.

WSSS identifies a subgraph that is much larger and higher-scoring than the true affected subgraph, including many nodes that are significant just by chance.

This happens regardless of parameter settings, and for comparison methods such as the upper level set (ULS) scan as well.

This overfitting problem is worse for larger α thresholds, so when we scan over α, the largest α value considered often gives the maximum score.

Some possible solutions

Possible solution #1: Consider only values of α up to some lower α_{max}.

Possible solution #2: Eliminate nodes that are not significant at level α_{max}.

Key insight from percolation: randomly removing more than a proportion of nodes $\lambda = 1 - \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$, where $\langle k \rangle$ is avg. node degree and $\langle k^2 \rangle$ is avg. squared degree, leads to the breakdown of the giant component of a graph.

Under H_0, p-values are uniform on [0,1], so removing p-values greater than $\alpha_{max} = 1 - \lambda = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$ should disconnect the graph.

Under H_1, for the affected subgraph the fraction of p-values greater than $1 - \lambda$ should be much lower than λ, keeping the subgraph connected.

This overfitting problem is worse for larger α thresholds, so when we scan over α, the largest α value considered often gives the maximum score.

Some possible solutions

Possible solution #1: Consider only values of α up to some lower α_{max}.

Possible solution #2: Eliminate nodes that are not significant at level α_{max}.

Key insight from percolation: randomly removing more than a proportion of nodes $\lambda = 1 - \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$, where $\langle k \rangle$ is avg. node degree and $\langle k^2 \rangle$ is avg. squared degree, leads to the breakdown of the giant component of a graph.

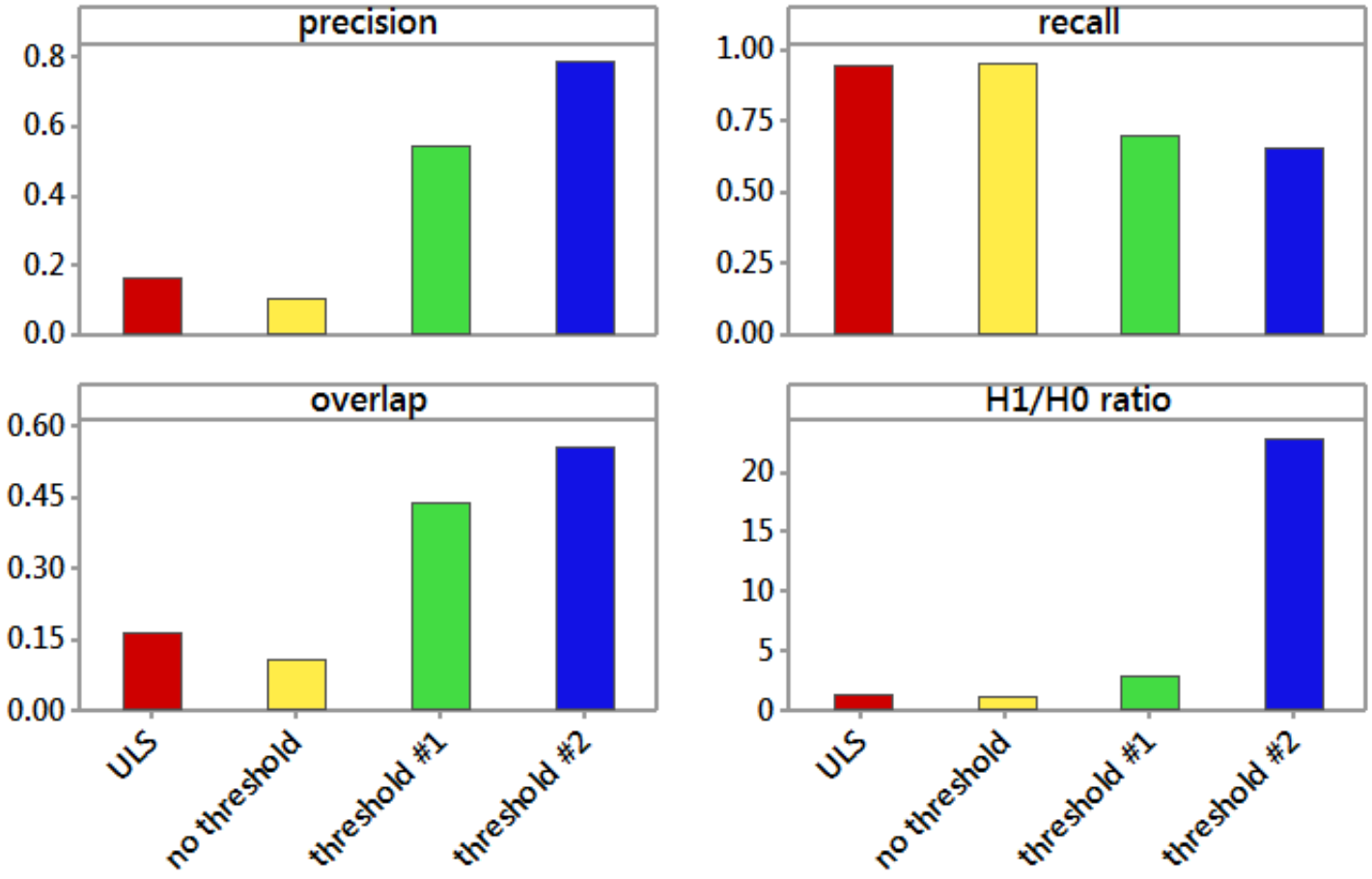
Under H_0, p-values are uniform on [0,1], so removing p-values greater than $\alpha_{max} = 1 - \lambda = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$ should disconnect the graph.

Under H_1, for the affected subgraph the fraction of p-values greater than $1 - \lambda$ should be much lower than λ, keeping the subgraph connected.

Resulting α_{max} values are .006, .008, and .05 for Twitter, Slashdot, and DBLP graphs respectively.

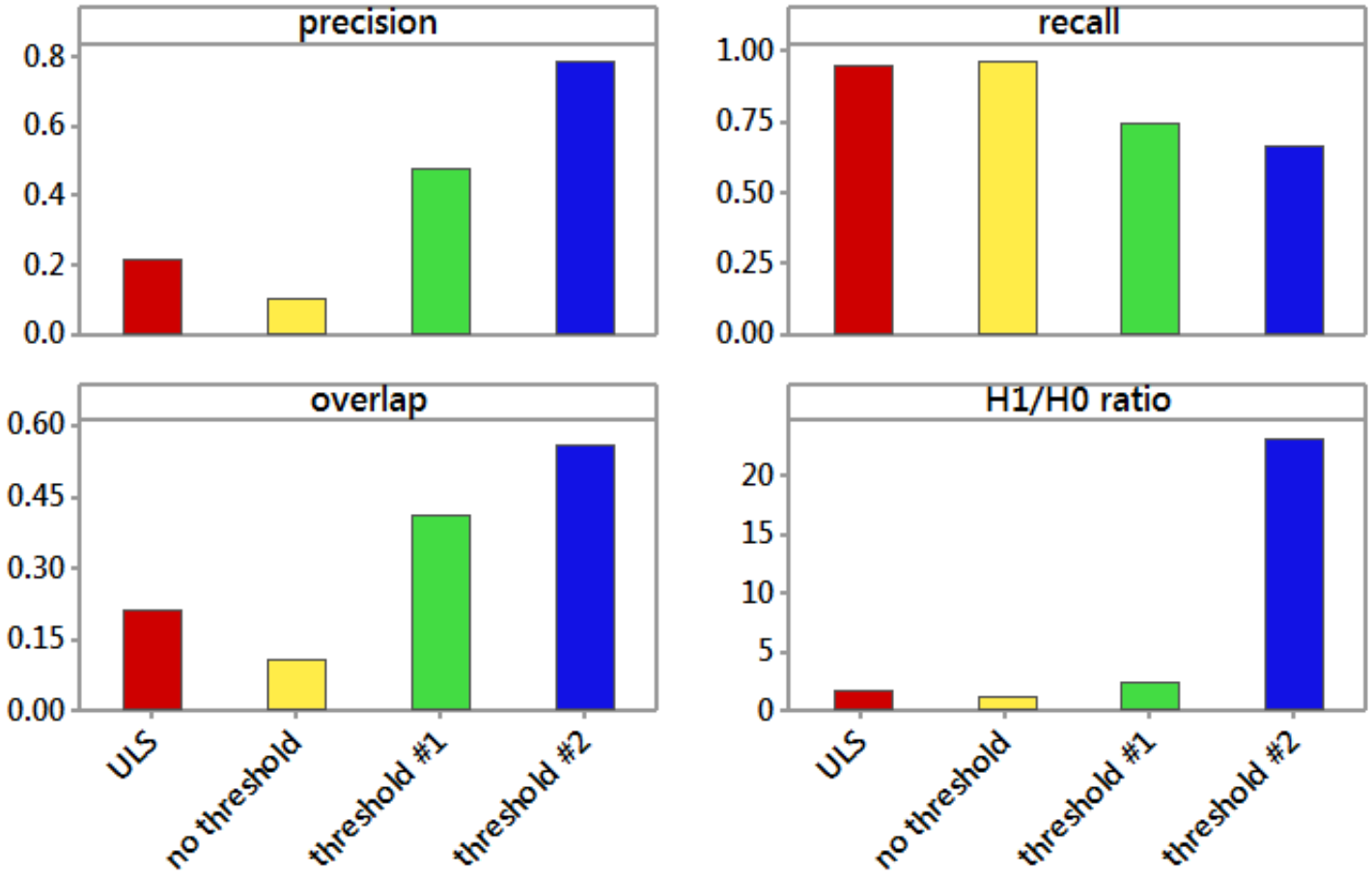
Results (part 2)

Comparison on **Twitter** dataset
(μ = 3; exact refinement with k = 1)



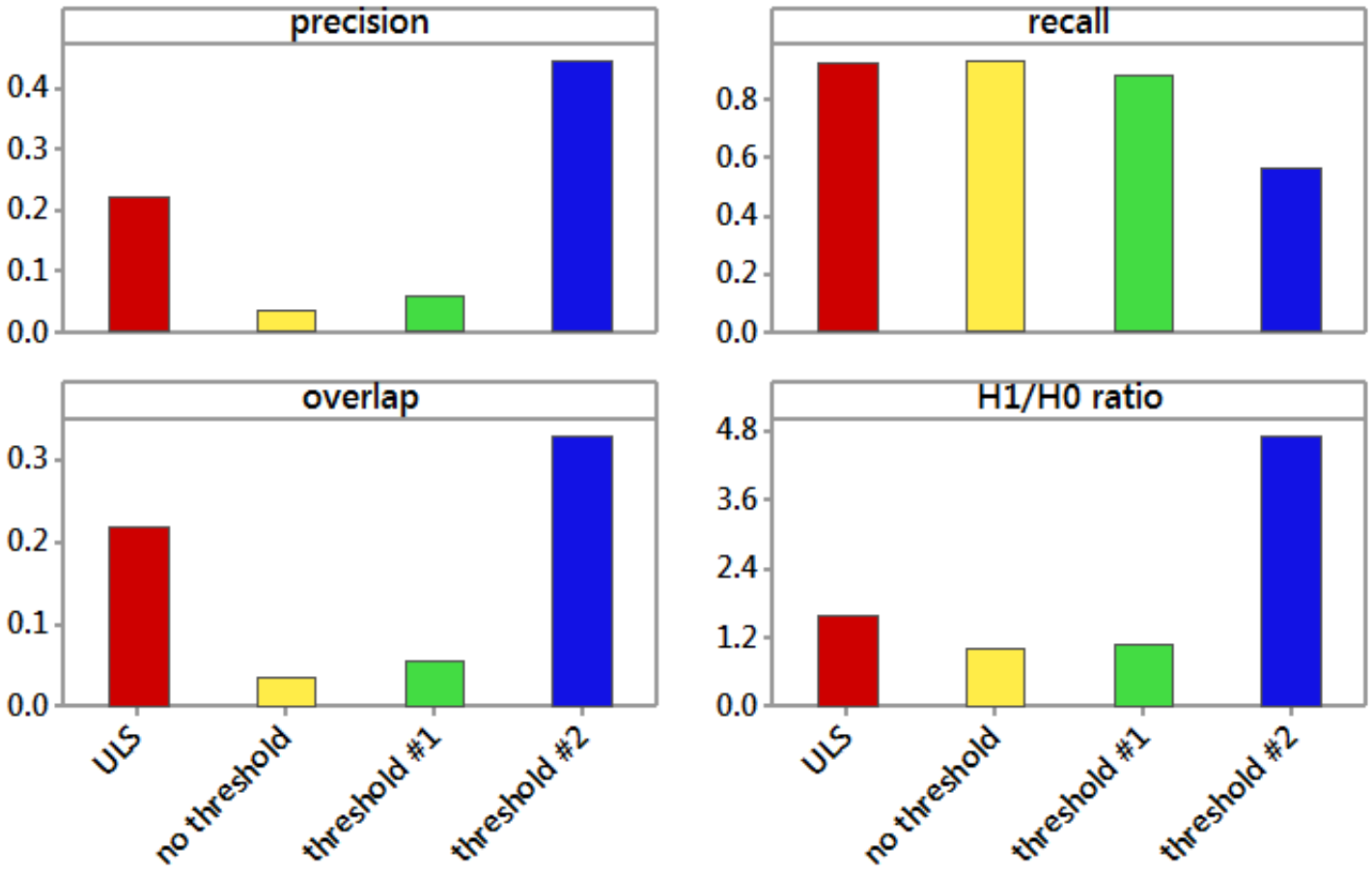
Results (part 2)

Comparison on **Slashdot** dataset
(μ = 3; exact refinement with k = 1)



Results (part 2)

Comparison on **DBLP** dataset
(μ = 3; exact refinement with k = 1)



Conclusions

Pattern detection in huge real-world graphs is challenging because of both computational complexity and the large number of individually significant “false positive” nodes.

We can exploit the **core-periphery structure** of real-world graphs to enable efficient (approximate) maximization of a likelihood ratio statistic over connected subgraphs.

Naïve maximization of the likelihood ratio statistic leads to detecting overly large, low precision subgraphs.

Using the **percolation threshold** of the graph to reduce overfitting can enable more accurate detection.

References

Non-parametric scan statistics and subset scanning:

- D.B. Neill. Fast subset scan for spatial pattern detection. *Journal of the Royal Statistical Society (Series B: Statistical Methodology)* 74(2): 337-360, 2012.
- E. McFowland III, S. Speakman, and D.B. Neill. Fast generalized subset scan for anomalous pattern detection. *Journal of Machine Learning Research*, 14: 1533-1561, 2013.
- F. Chen and D.B. Neill. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. *Proc. 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1166-1175, 2014.
- F. Chen and D.B. Neill. Human rights event detection from heterogeneous social media graphs. *Big Data* 3(1): 34-40, 2015.

Core-periphery structure: Leskovec et al., *Internet Math.,* 2009.

Core-tree decomposition: Maehara et al., *VLDB*, 2014.

Color coding: Cadena, Chen, and Vullikanti, SDM 2017.

Graphs obtained from SNAP: https://snap.stanford.edu/data/index.html