SAND2012-5832C

Sandia National Laboratories

DARPA

Fast Algorithms for Evolving Graphs via Assays, Sampling & Theory (FEAST)

Tamara G. Kolda (PI)

Sandia National Laboratories

University of Minnesota

U.S. Department of Energy

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

FEAST Team

- Tammy Kolda (Sandia – CA)
  - Numerical algorithms, scientific computing
- Ali Pinar (Sandia – CA)
  - Combinatorial scientific computing
- Cindy Phillips (Sandia – NM)
  - Discrete mathematics
- Jaideep Srivastava (UMN)
  - Social network analysis
- Karthik Subbian (UMN grad student)
  - Social network analysis

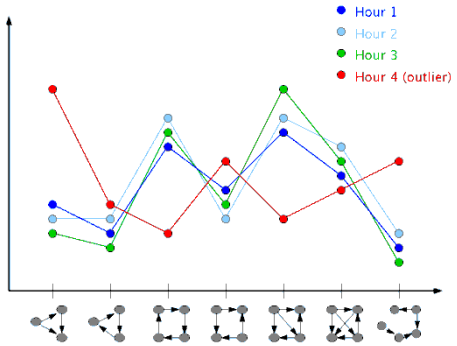
Not here today…

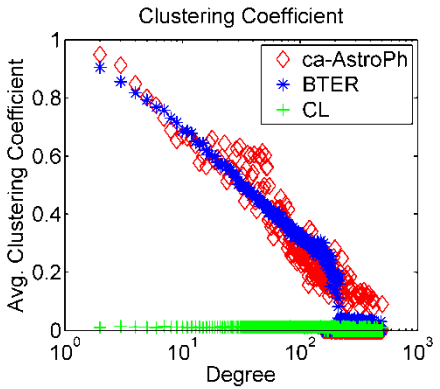
- Jon Berry (Sandia – NM)
  - Graph algorithms
- Todd Plantenga (Sandia – CA)
  - MapReduce for graph algorithms
- C. “Sesh” Seshadhri (Sandia – CA)
  - Theoretical computer science

FEAST Driving Principles

Objective: Measure, reproduce, and exploit quantifiable characteristics of real-world social network and computer traffic graphs

1. Graph Assays: Sampling-based techniques to compute frequency of common patterns in large-scale graphs
   - Ex: Estimate number of each type of directed triangle
2. Generative Models: Reproduce characteristics shown in the assays
   - Ex: Reproduce clustering coefficient per degree (i.e., triangle behavior)
3. Efficient Algorithms: Take advantage of structure of social networks
   - Ex: Triangle enumeration is provably linear for power law graphs with exponent < 7/3





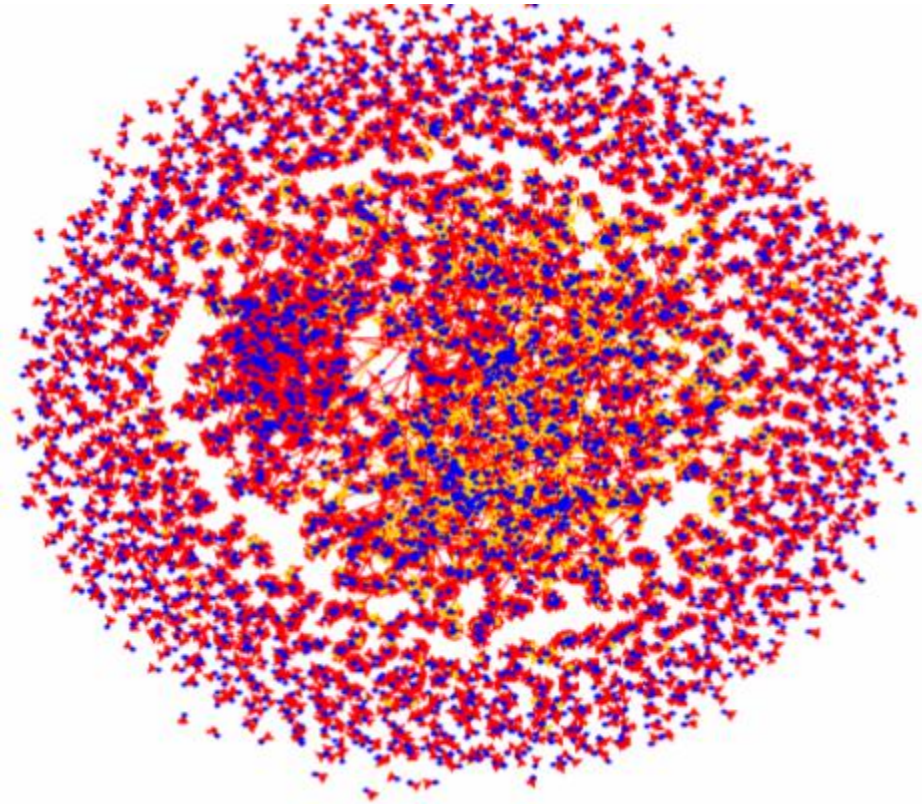
Sandia National Laboratories

University of Minnesota

GENERATIVE MODELS

Why Model Networks?

- Insight into…
  - Generative processes and principles
  - Properties such as eigenvalue distribution, diameter, etc.
  - Changes over time
- Enable sharing of realistic but non-sensitive data
  - Computer network traffic
  - Social networks
- Statistics on graphs
  - Drive sampling strategies
  - Inform anomaly detection
  - Determine nodes/edge properties
- Test graph algorithms
  - Various scales
  - Various degree distributions
  - Future versions of today's networks



Desiderata for Undirected Graphs

- Reproduce a **variety** of heavy tailed degree distributions
  - Power law, Pareto-lognormal, etc.
  - Can also consider joint degree distribution
- Reproduce global and degree-wise **clustering coefficients**
  - Global clustering coefficient: C = probability that a random "wedge" is closed (to form a triangle)
  - Degree-wise clustering coefficient: C_d = probability that a random "wedge" centered at a node of degree d is closed
  - Consider also the 3-way degree distribution of triangles
- ***Scalable***
  - Fast generation, i.e., O(m log n)
  - Independent edge generation, i.e., no history
  - 2^{42} nodes and 2^{46} edges for Graph 500

Most existing models fail here – cannot reproduce observed degree distribution.

Challenging to have both high clustering coefficients and scalability!

Stochastic Kronecker Graphs (SKG) is the leader in scalable generators

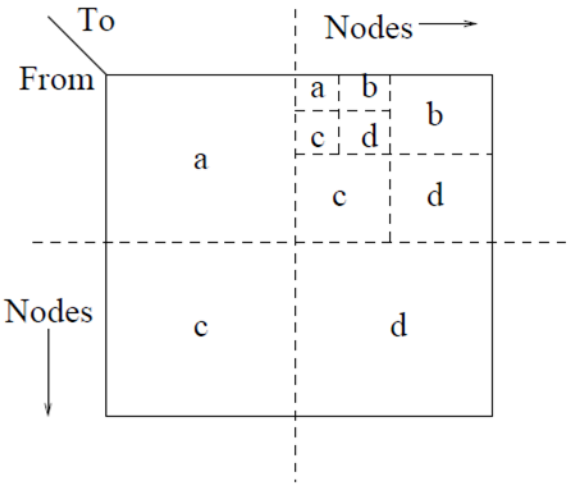
Sandia National Laboratories

University of Minnesota

- Generator for GRAPH500 Benchmark
  - Parallel edge generation
  - Only requires 5 parameters
- SKG Inputs
  - L = # of levels → number of nodes is N = 2^L
  - T = 2 x 2 generator matrix (entries sum to 1)
  - M = # edges
- Notes
  - Some edges may be duplicates or self-links and are ignored
  - Fitting to real data using "KronFit" takes between 7 minutes for 20K nodes to 4 hours for 500K nodes
- References
  - R-MAT: Chakrabarti , Zhan, & Faloutsos, SDM04
  - SKG: Leskovec et al., JMLR, 2010
- **3 issues with SKG…**

Graph 500 Parameters:

- *T = [0.57, 0.19, 0.19, 0.05]*
- *L ∈ {26, 29, 32, 26, 39, 42}*
- *M = 16 · 2^L*



SKG Edge Insert Procedure:

- *Choose a quadrant of the adjacency matrix proportional to entries of T*
- *Repeat for a total of L times to land at a single entry of the matrix*

Issue #1: SKG degree distribution is not realistic (oscillations)

Sandia National Laboratories

University of Minnesota

Seshadhri, Pinar & Kolda, arXiv:1102.5046, 2011; short version in ICDM11

Fixing the degree distribution

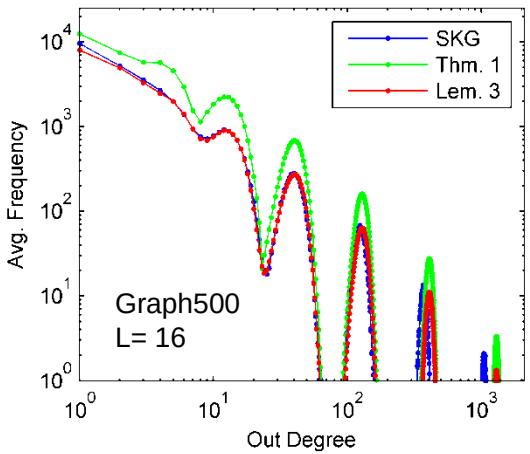
- Thm: SKG degree distribution oscillates between lognormal and exponential tail
- Fix: Choose fixed random noise value μ_i for each of the L levels

$$T_i = \begin{bmatrix} a - \frac{2\mu_i a}{a+d} & b + \mu_i \\ b + \mu_i & d - \frac{2\mu_i d}{a+d} \end{bmatrix}$$

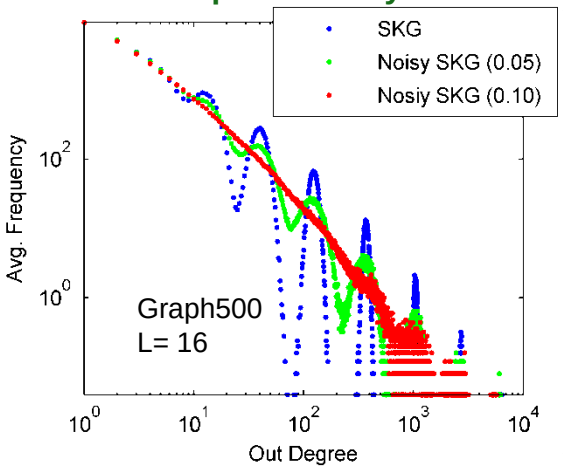
Naïve addition of noise fails

- Naïve: "*To smooth out fluctuations in the degree distributions, we add some noise to the (a, b, c, d) values at each stage of the recursion and then renormalize (so that a+b+c+d = 1).*" – CZF04
- Compare naïve noise (blue) to our method (red) for Graph500 with L=26

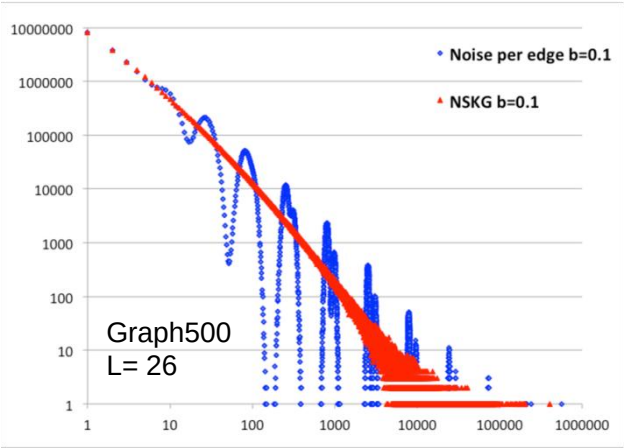
Theoretical Prediction of Deg. Dist.



Our Proposed Noisy SKG



Our Fix (red) versus Naïve Noise (blue); Hadoop implementation



Issue #2: SKG isolated vertices
Issue #3: No community structure

Sandia National Laboratories

University of Minnesota

Seshadhri, Pinar & Kolda, arXiv:1102.5046, 2011; short version in ICDM11

- Issue #2: SKG may have many **isolated vertices**

$$I = \sum_{r=-L/2}^{L/2} \binom{L}{L/2+r} \exp(-2\lambda\tau^r),$$

$$\tau = (a+b)/(1-(a+b))$$

$$\lambda = \frac{M}{N}[4(a+b)(1-(a+b))]^{L/2}$$

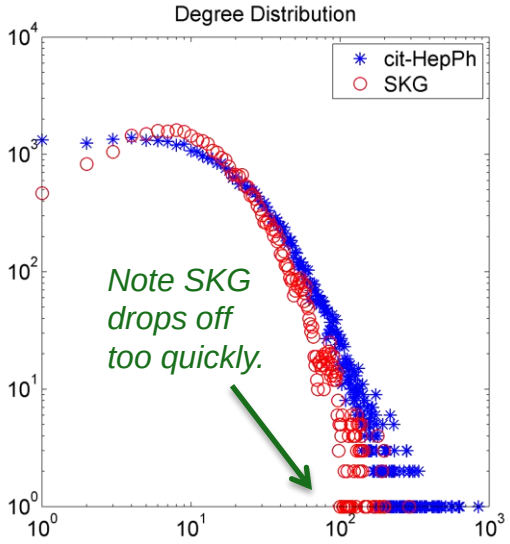
- Issue #3: **Low clustering coefficients**

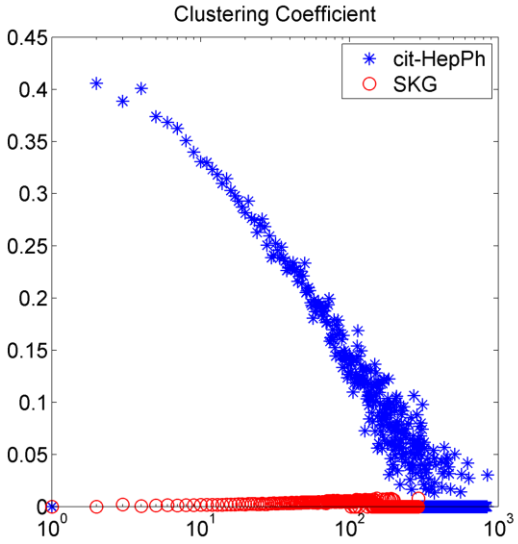
- Noise addition (per Issue #1) doesn't fix these issues

L	Isolated Nodes	Avg. Degree
26	51%	32
29	57%	37
32	62%	41
36	67%	49
39	71%	55
42	74%	62

Graph 500 Parameters:

- *T = [0.57, 0.19, 0.19, 0.05]*
- *M = 16 · 2^L*





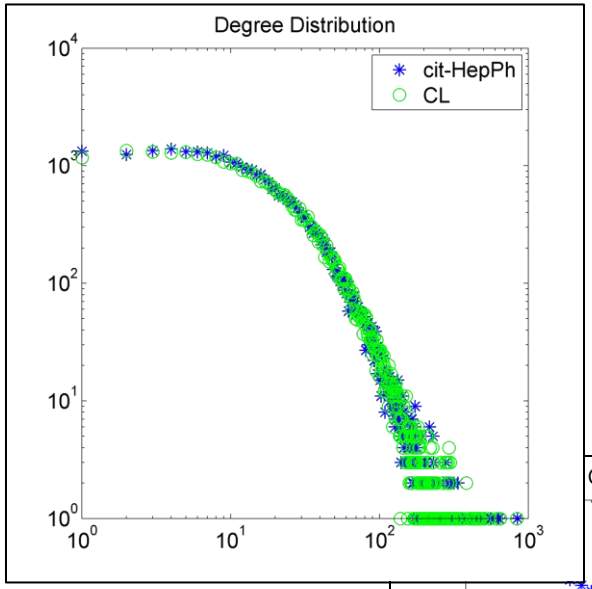
Capturing Degree Distribution: Chung-Lu Model

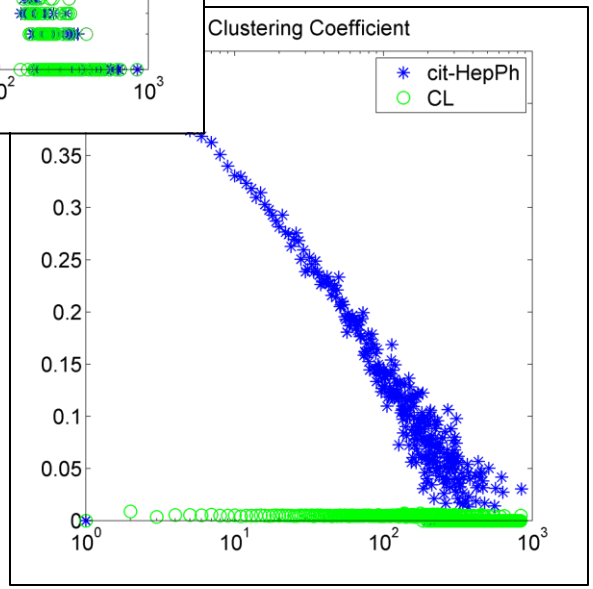
Sandia National Laboratories

University of Minnesota

Aiello, Chung, Lu, *Exp. Maths.* 2001; Chung & Lu, *PNAS*, 2002 and *Annals. Combinatorics*, 2002.

- Premise: Probability of edge is proportional to degrees of endpoints
  - $Pr(edge_{ij}) = d_i d_j / 2M$ (M edges)
- Scalable version: Independently pick end points of each edge proportional to degree
  - Requires only O(n) information
  - Need to be careful in implementation to avoid excessive isolates and/or duplicates
- Only works if $d_i d_j < 2M$, but we think it can be fixed
- Suggested alternative to SKG
  - Pinar, Seshadhri, Kolda, SDM12
- Closely matches real data when clustering coefficient is not an issue

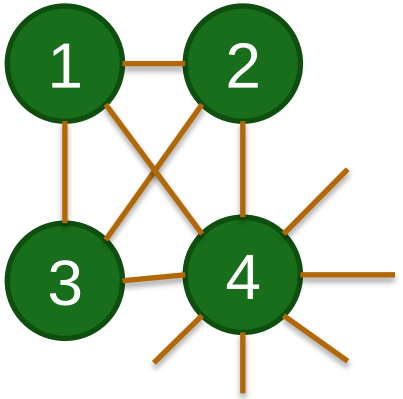




Matching the Clustering Coefficients

Seshadhri, Kolda & Pinar, Phys. Rev. E, 2012

- CL model has no mechanism to close wedges
  - Therefore does not get high clustering coefficients
  - Models that achieve high clustering coefficients require **history** to guide future edge insertions
- How to get high clustering coefficient without history?
  - Random pairings cannot close wedges!
- Must pre-group the nodes into **affinity blocks**
  - Our theory: CL graph with high clustering coefficient must have dense ER block at its core
  - Each affinity block is a near-clique
- Simplifying assumption: Non-overlapping blocks
  - Implies blocks must comprise nodes of equal (or near-equal) degree
  - Can tune clustering coefficient to be appropriate for the degree by changing connectivity of the block

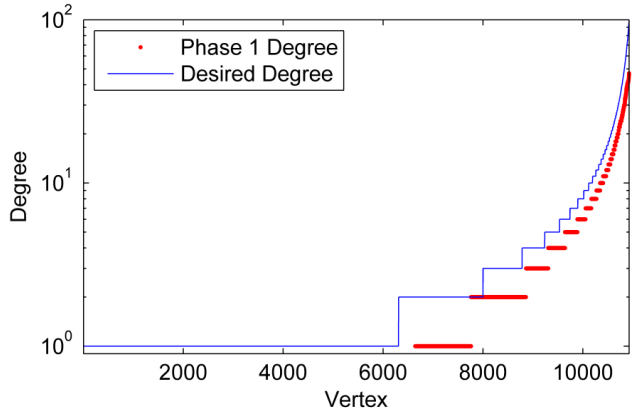


$$c_1 = 1$$
$$c_2 = 1$$
$$c_3 = 1$$
$$c_4 = 0.1$$

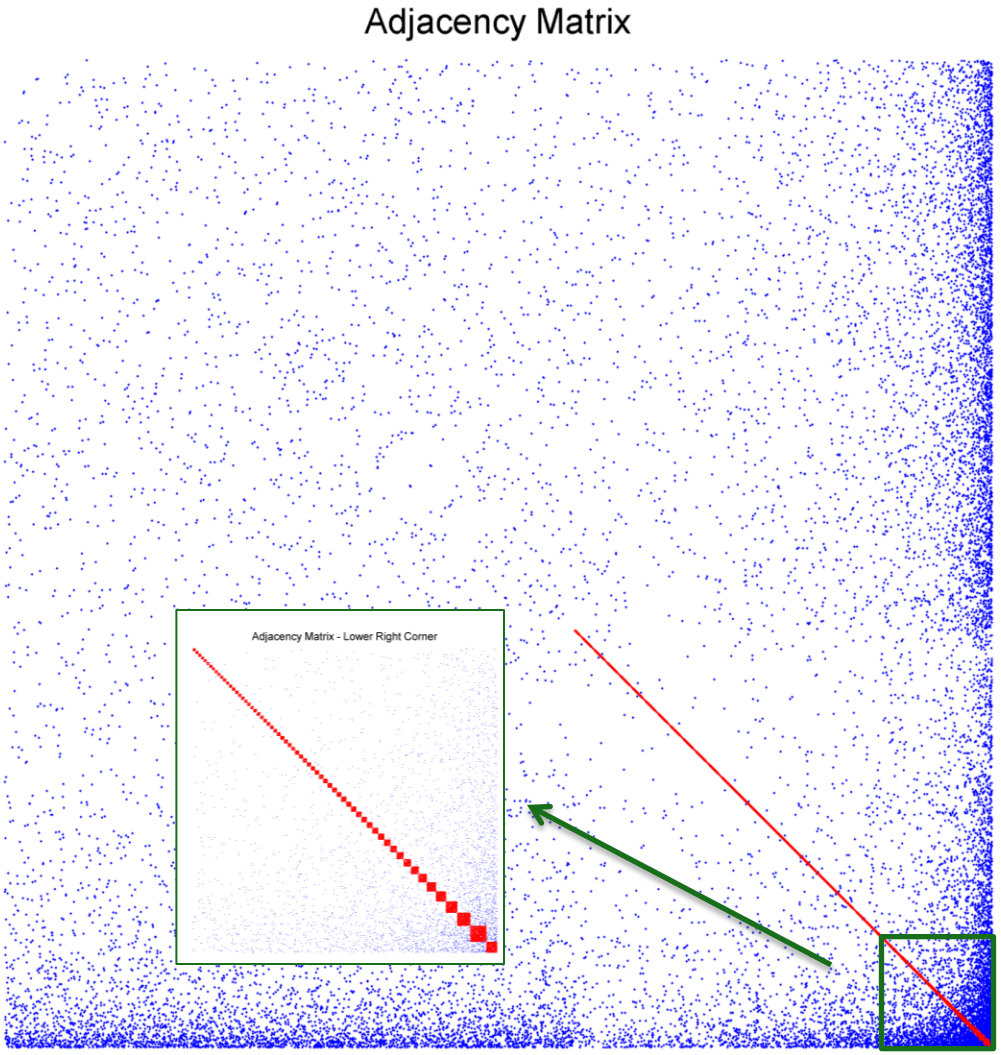
If nodes of different degrees are in the same block, then the high-degree nodes will have low clustering coefficients.

BTER: Block Two-Level ER

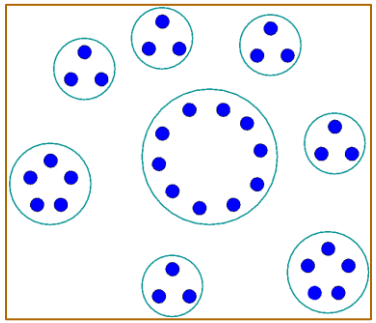
- Phase 1
  - Create near cliques via ER with a high probability such that phase 1 degrees do not exceed desired degrees



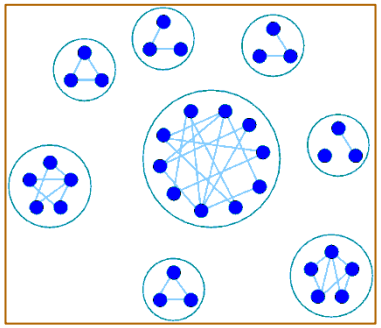
- Phase 2
  - Fill in the remainder of the degree distribution using a CL approach



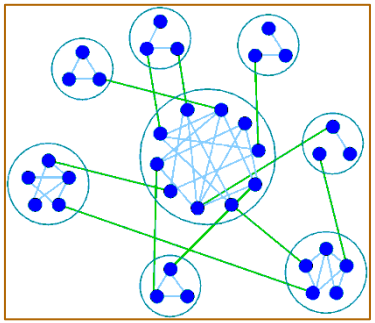
BTER: Block Two-Level Erdős-Rényi



Preprocessing:
Create explicit affinity blocks of nodes with same (or nearly same) degree. The blocks are determined by the degree distribution.



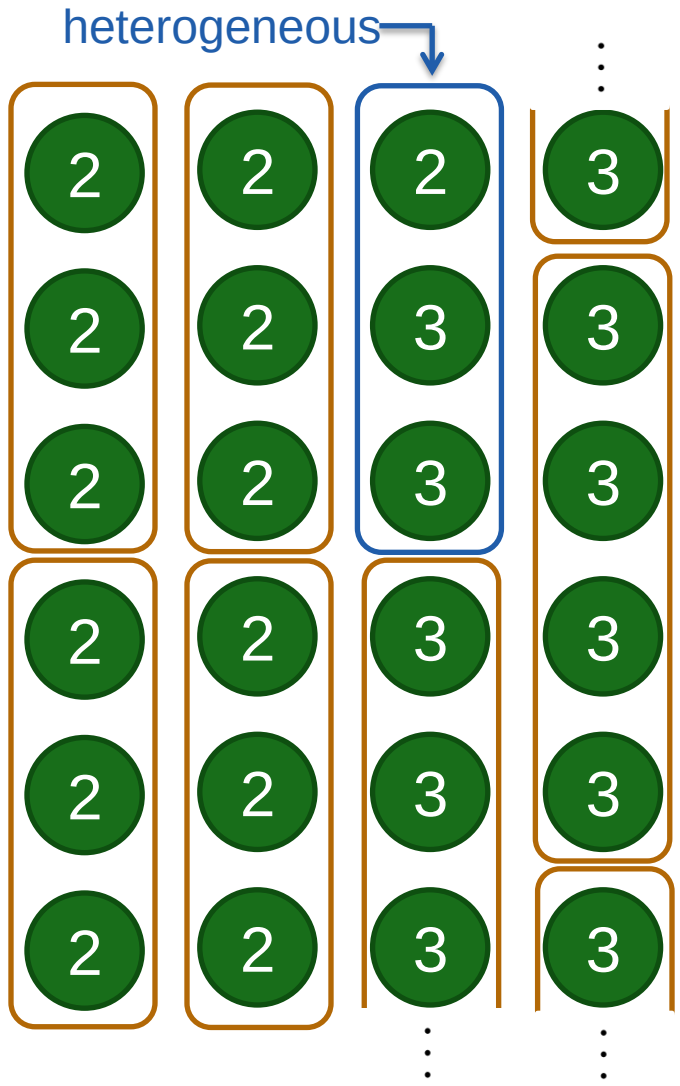
Phase 1:
Erdős-Rényi graphs in each block. Connectivity based on observed C_d determines number of links within each block. Formula depends on lowest-degree node in block.



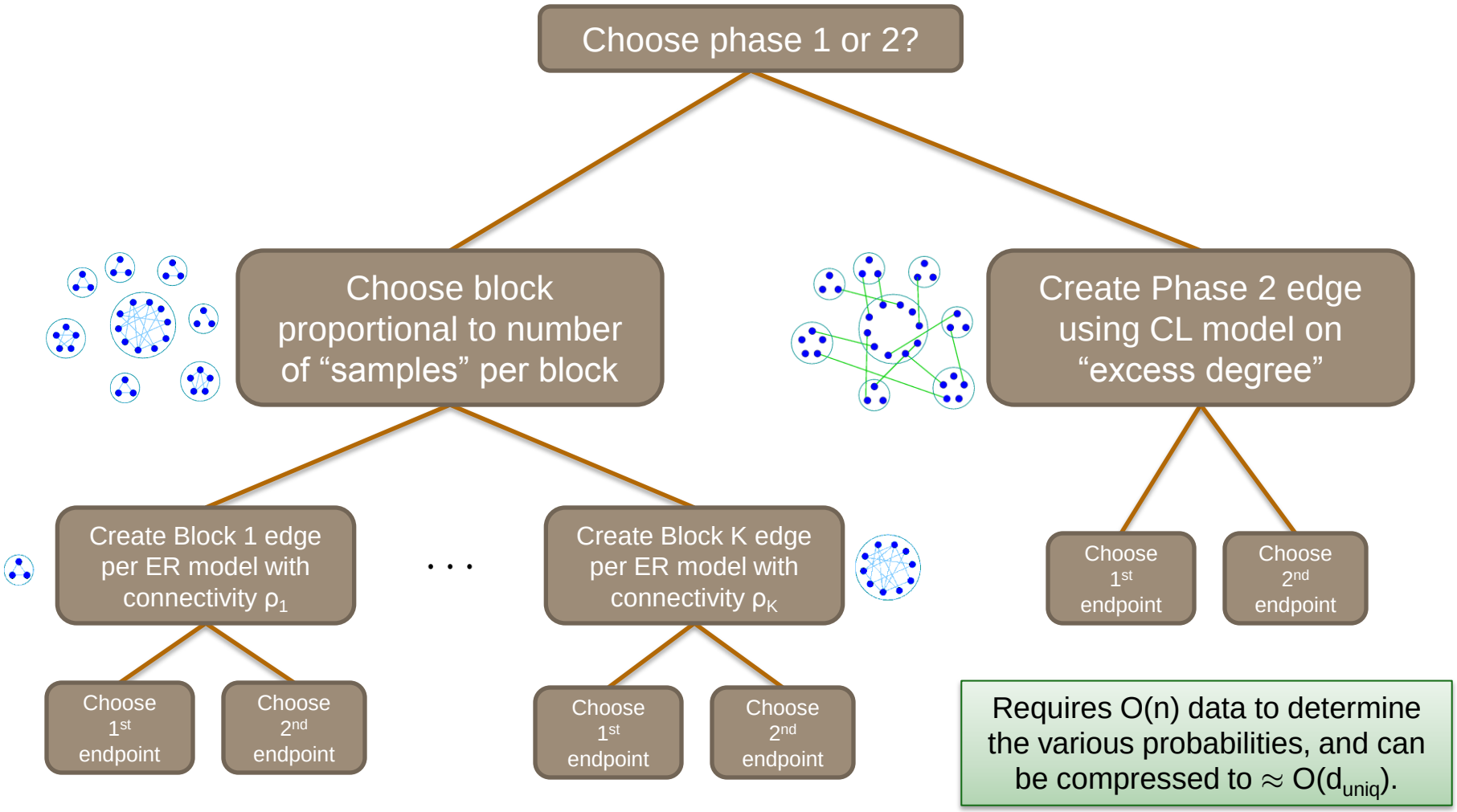
Phase 2:
CL (aka weighted ER) model on "excess" degree, creates connections across blocks. Can run in tandem to Phase 1 by worked with *expected* excess degree.

Preprocessing: Determining Blocks

- Input: Desired degree-distribution
  - Assume nodes sorted by degree, least to greatest
  - Ignore degree-1 vertices in this phase
- Algorithm:
  - Group nodes in order
  - Bulk assign, if possible
- Output: **Compressed** information about blocks
  - Size of each block
  - Starting index of each block
  - "Weight" of each block (depends on connectivity)
  - $O(d_{uniq})$ information, if compressed
- Observe: If degree distribution is power law, then so is the distribution of the block sizes
  - Evocative of Dunbar's number:
    Max block size = 100 for γ=2



Scalable BTER: Independent Edges



Visualization of BTER Graph

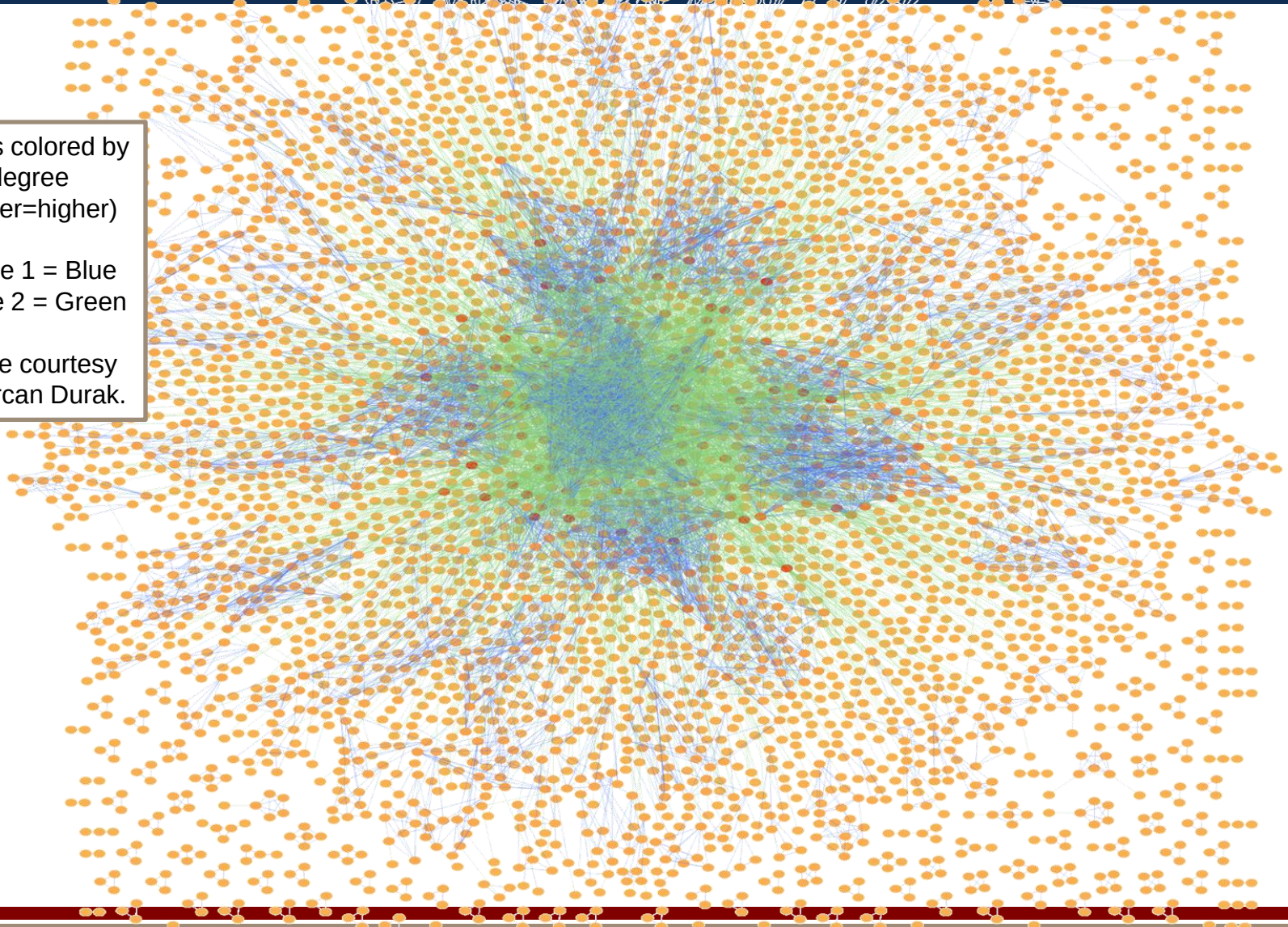
Sandia National Laboratories

University of Minnesota

Nodes colored by degree (darker=higher)

Phase 1 = Blue
Phase 2 = Green

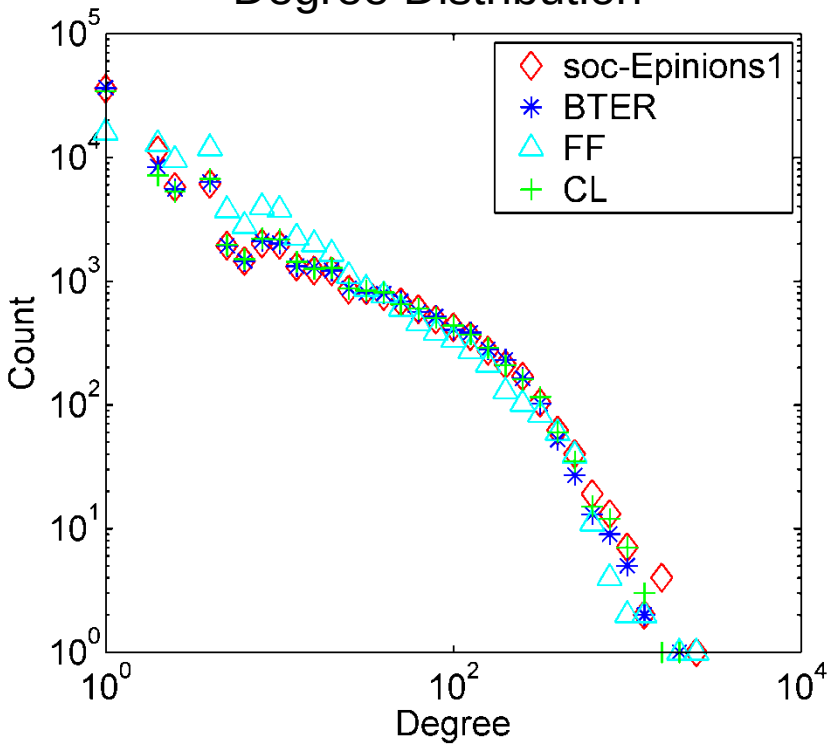
Image courtesy of Nurcan Durak.



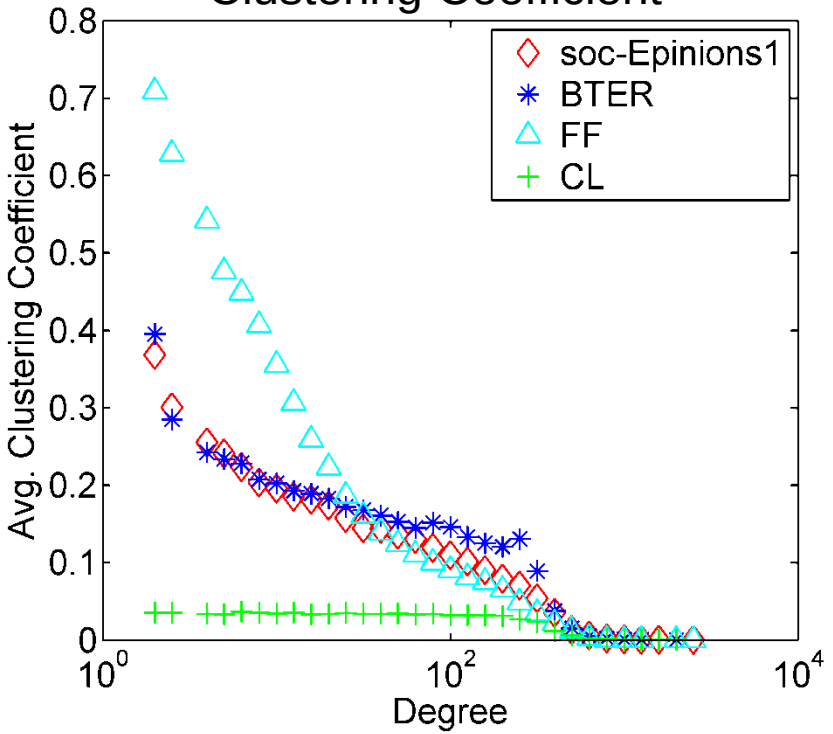
BTER versus CL and Forest Fire (FF)

soc-Epinions1 (sym): 76K nodes, 405K edges (downloaded from SNAP)

Degree Distribution



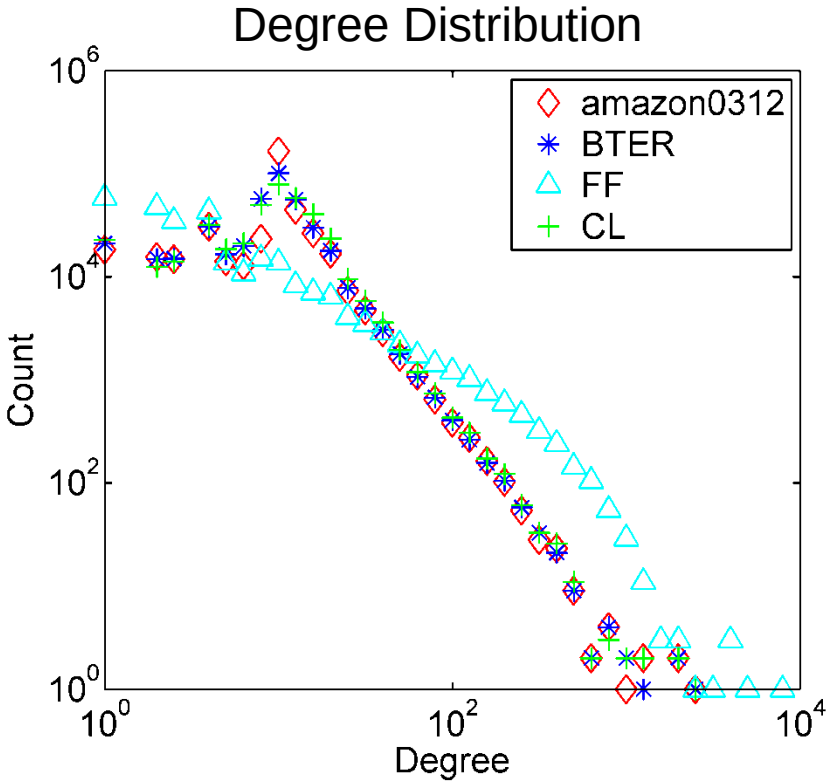
Clustering Coefficient

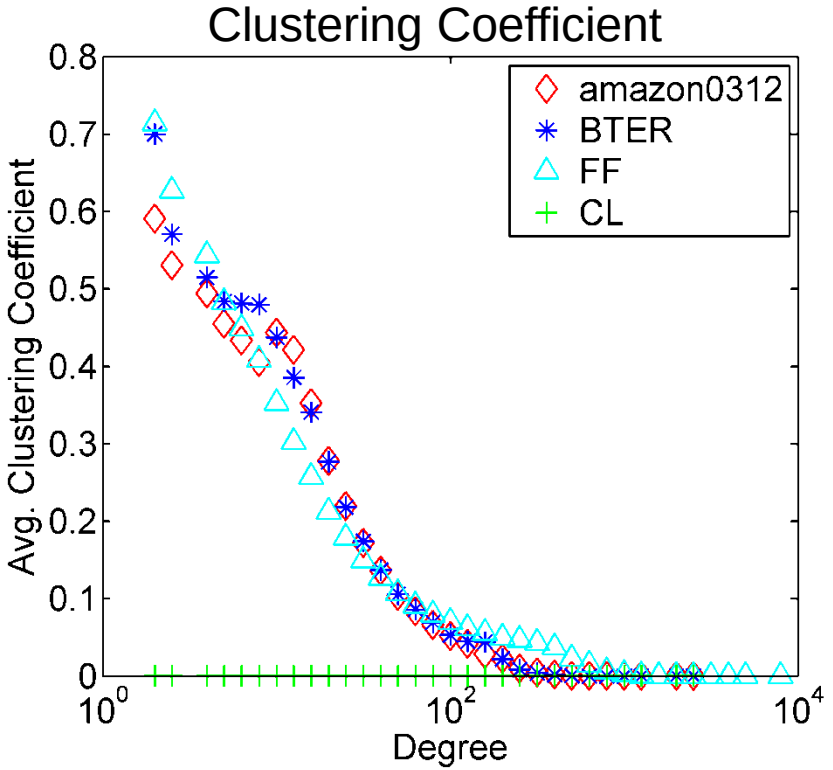


BTER: Seshardhri, Kolda, Pinar, PRE 2012
CL: Chung & Lu, PNAS 2002
FF: Leskovec, Kleinberg, Faloutsos, KDD 2005

BTER versus CL and Forest Fire (FF)

amazon0312 (sym): 400K nodes, 2.3M edges (downloaded from SNAP)

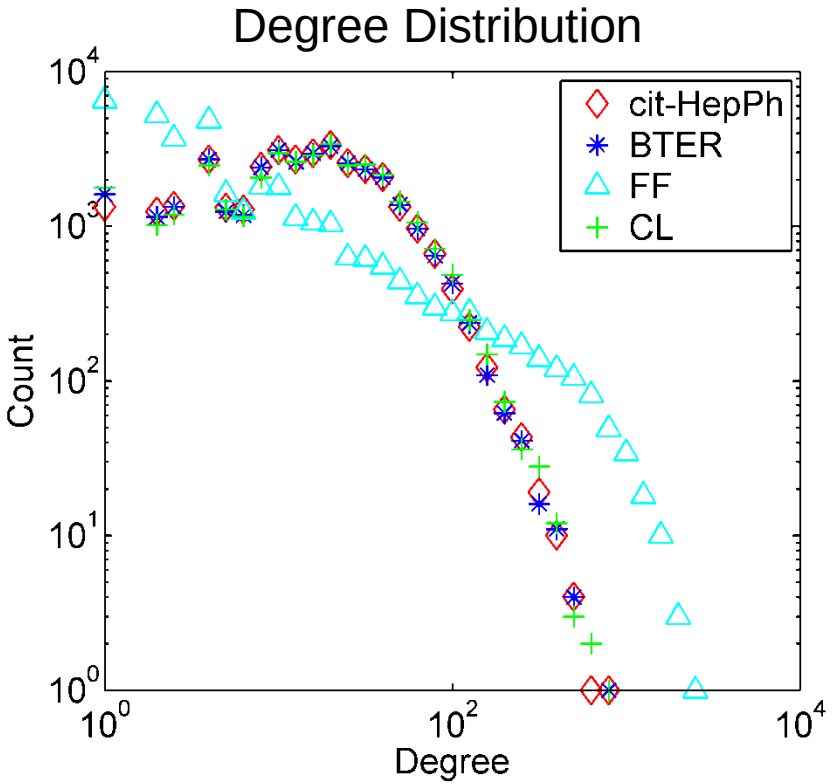


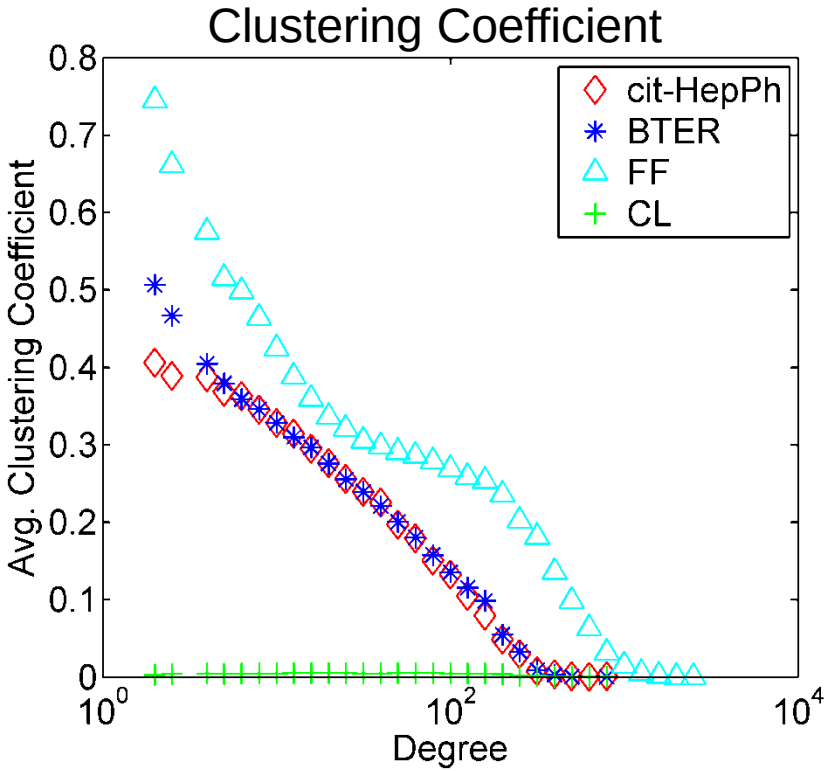


BTER: Seshardhri, Kolda, Pinar, PRE 2012
CL: Chung & Lu, PNAS 2002
FF: Leskovec, Kleinberg, Faloutsos, KDD 2005

BTER versus CL and Forest Fire (FF)

cit-HepPh (sym): 400K nodes, 2.3M edges (downloaded from SNAP)

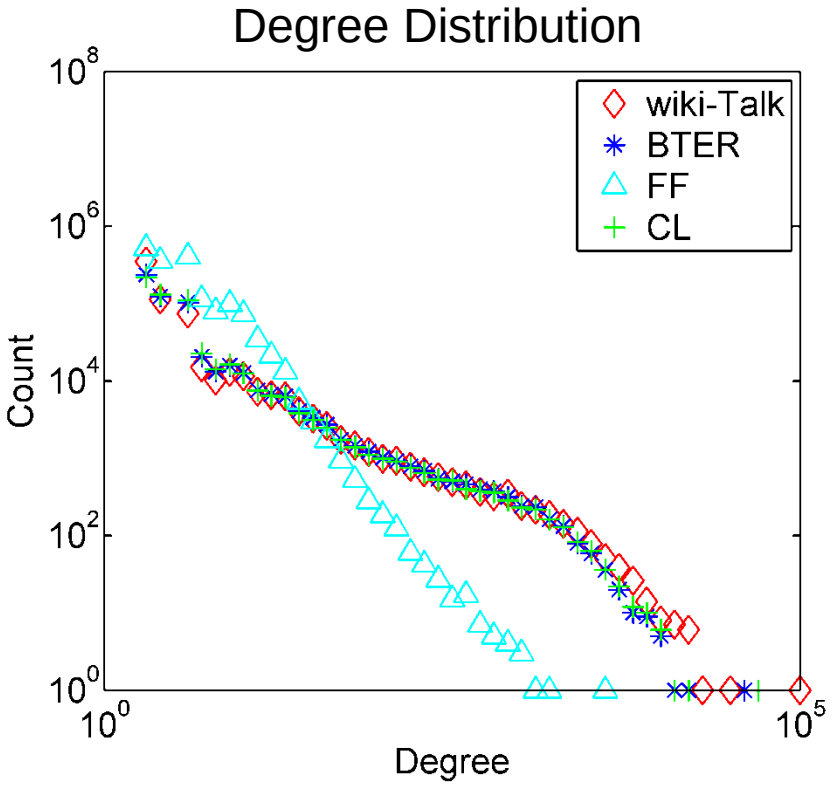


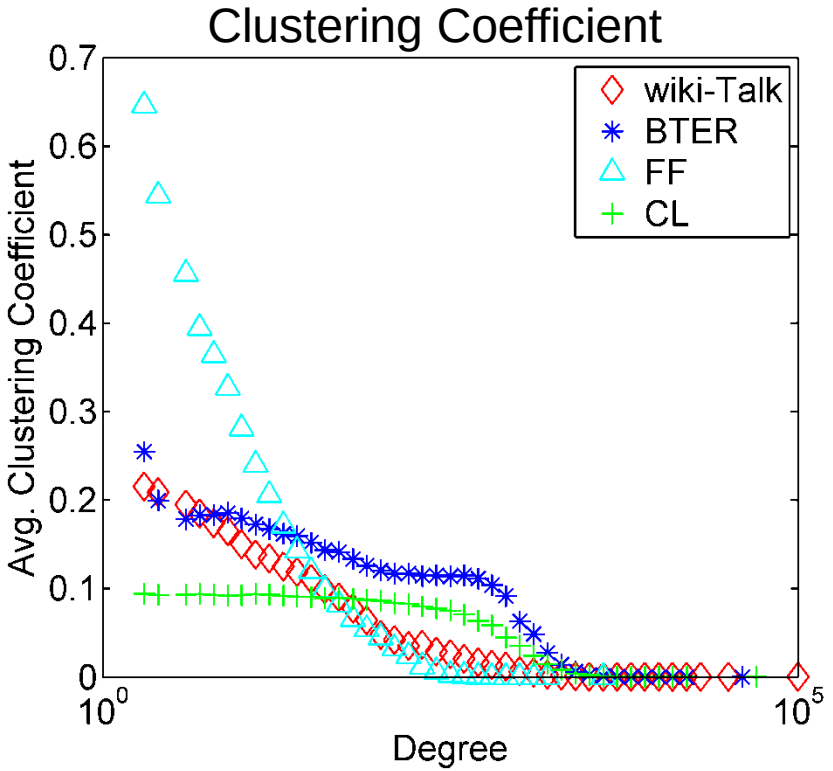


BTER: Seshardhri, Kolda, Pinar, PRE 2012
CL: Chung & Lu, PNAS 2002
FF: Leskovec, Kleinberg, Faloutsos, KDD 2005

BTER versus CL and Forest Fire (FF)

wiki-Talk (sym): 2.3M nodes, 4.7M edges (downloaded from SNAP)

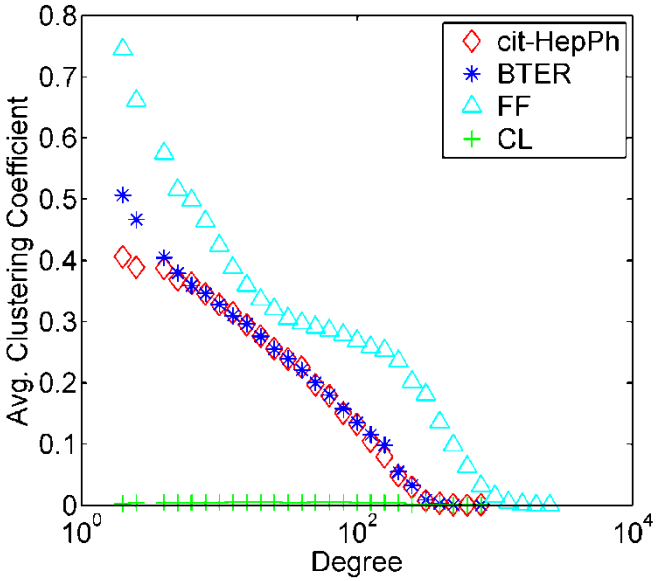




BTER: Seshardhri, Kolda, Pinar, PRE 2012
CL: Chung & Lu, PNAS 2002
FF: Leskovec, Kleinberg, Faloutsos, KDD 2005

Generative Models

- BTER Model
  - Model based on theoretical observation
    - High clustering coefficient in a CL (random) graph is only possible if there is an ER subgraph
    - Affinity blocks = ER subgraphs.
  - Matches degree distribution and clustering coefficient
    - Superior to CL, SKG (aka RMAT), and FF
  - Hadoop MapReduce and C++/MPI implementation in progress
    - Will propose as replacement for RMAT/SKG in Graph 500
- Even BTER model…
  - Current version is only for undirected simple graphs
  - Plan to add edge direction, attributes, and evolution
  - Goal is that the models should capture the patterns observed in real data

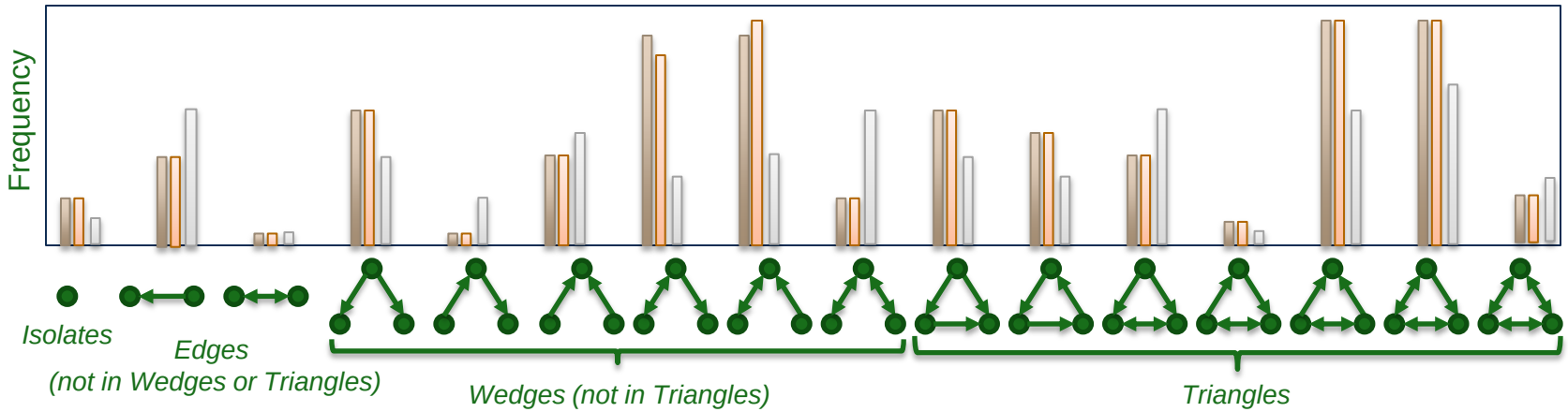


Sandia National Laboratories

University of Minnesota

GRAPH ASSAYS

Graph Assays



- Graph assays measure frequency of common patterns
  - A.k.a. structural signatures, graphlets, etc.
  - For directed graphs, just looking up thru 3-patterns yields rich information
  - Will also add attributes (e.g., degrees of neighbors, time dependencies, etc.) and other features to the patterns
- Capture local interactions within the graph
- Graph assays can be used for validation of generative model – the assays for the real data and the generated data should match
- Can also be used to monitor status of an evolving network, e.g., via hourly snapshots
- **Difficulty: Counting patterns such as triangles can be prohibitively expensive (in time and space) for large-scale graphs. Even linear time/space may be too much.**

Challenge of Triangle Counting

$d_i = \text{degree of node } i$

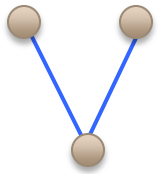
$w_i = \binom{d_i}{2} = \#\text{ wedges centered at node } i$

$t_i = \#\text{ triangles containing node } i$

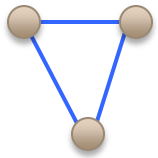
$c = \dfrac{3 \times \#\text{ triangles}}{\#\text{ wedges}} = \dfrac{\sum_i t_i}{\sum_i w_i} = \text{clustering coefficient}$

$c_i = \frac{t_i}{w_i} = \text{clustering coefficient of node } i$

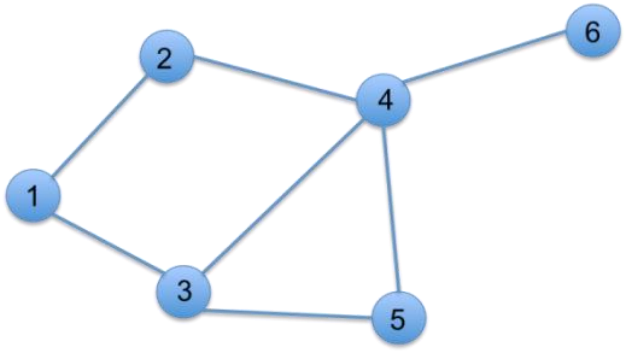
$c_d = \dfrac{1}{n_d} \sum_{i \in V_d} c_i = \text{mean for degree } d$



Open wedge



Closed wedge (i.e., triangle)



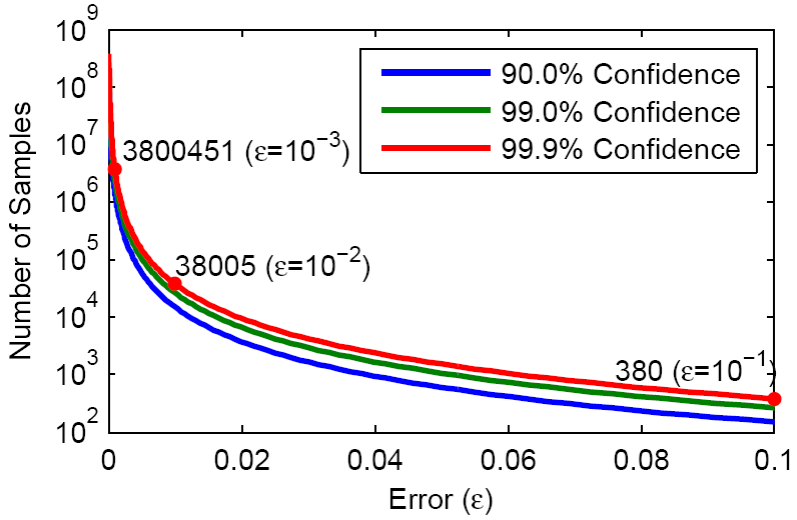
Example with 13 wedges and 1 triangle

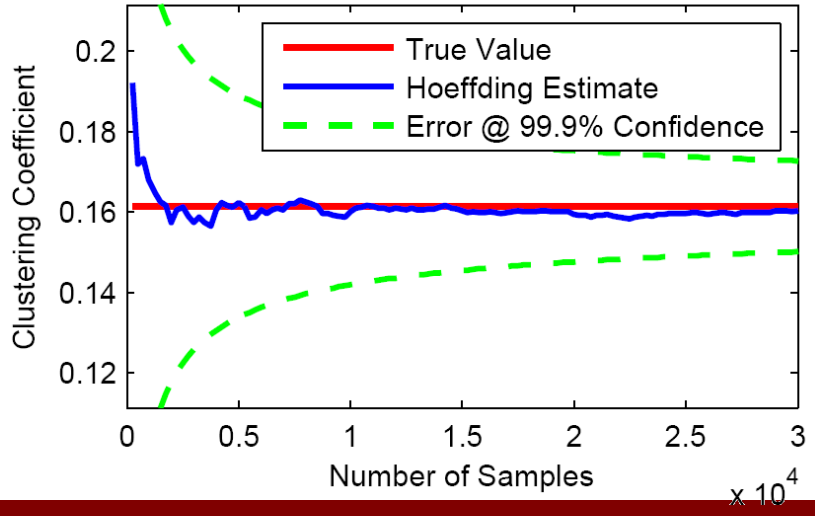
- Naïve algorithm: Enumerate every wedge and check for closure.
- Clever algorithm: Only consider wedges whose lowest degree node is at the center.
- *But, even the clever algorithm still fails for large graphs if the number of wedges is large.*

Sampling Approach

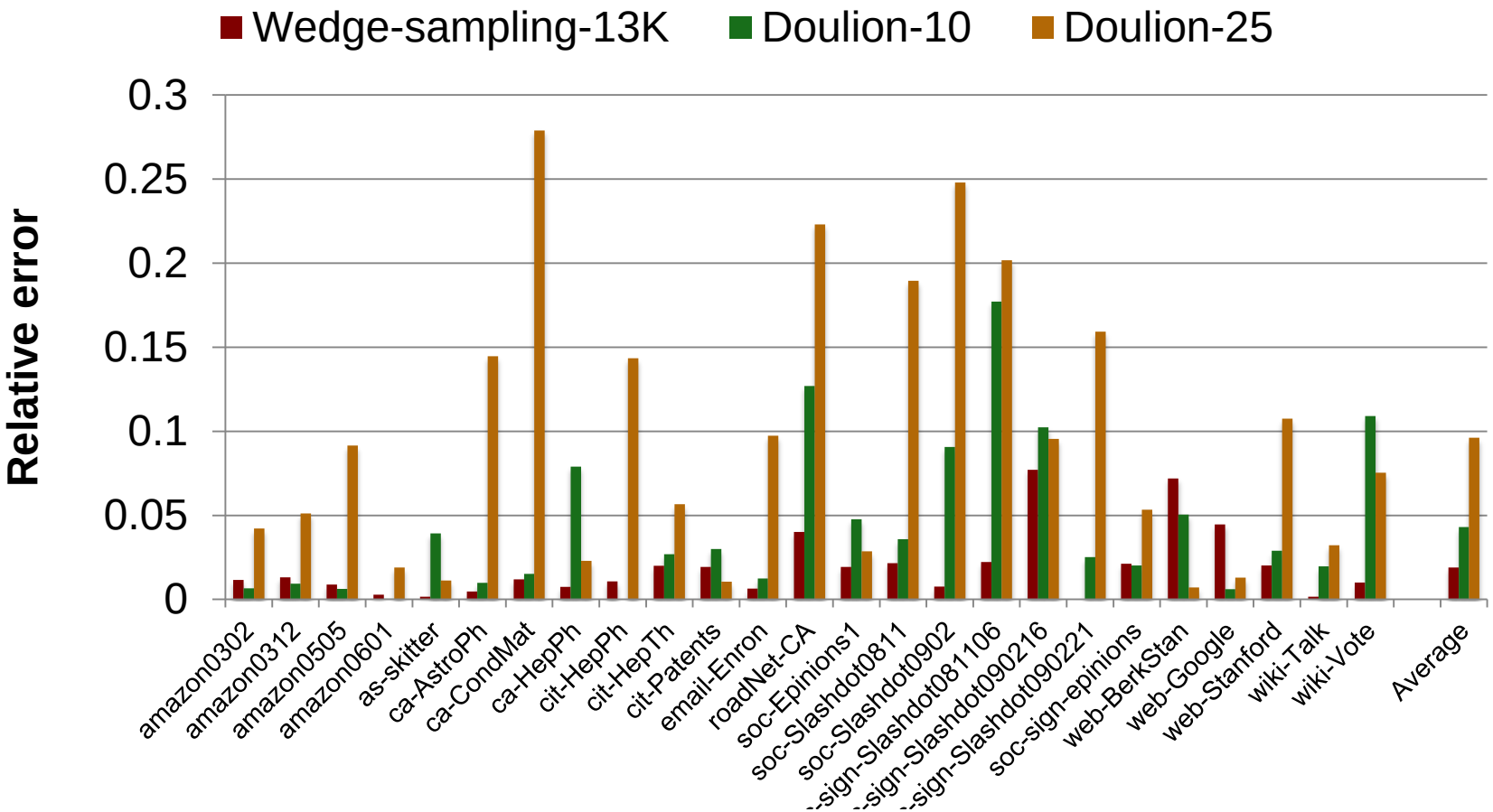
Seshadhri, Pinar & Kolda, arXiv:1202-5230, 2012

- c = clustering coefficient
  = Pr(random wedge is closed)
- p = total # wedges (calculated explicitly)
- t = total # triangles
- c = 3t/p
- Hoeffding bound
  - c' = sample mean for k random wedges
  - $P\{|c' - c| \geq \epsilon\} \leq \delta$
  - t' = 1/3 c'·p
  - $P\{|t' - t| \geq 1/3\ \epsilon \cdot p\} \leq \delta$
- For δ = 99.9% confidence and ϵ = 1% error, need only k = 38,005 samples
- Number of samples (k) is independent of graph size



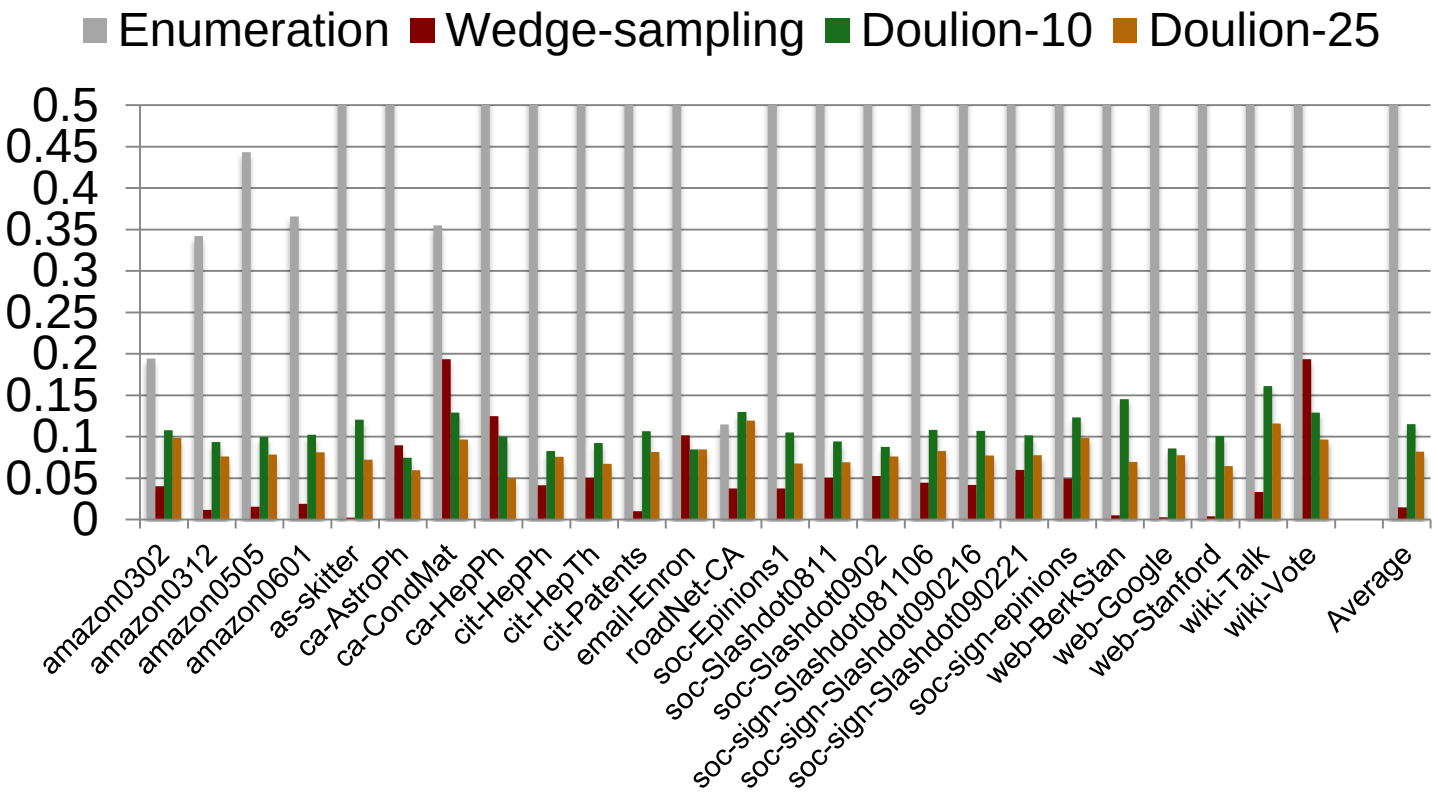


Wedge sampling is more accurate



Doulion [Tsourakakis et al, KDD09]: Generate a smaller graph by removing each edge with probability 1-ρ. Count the number of triangles in the original graph. Divide by ρ^3 to predict the number of triangles in the original graph.

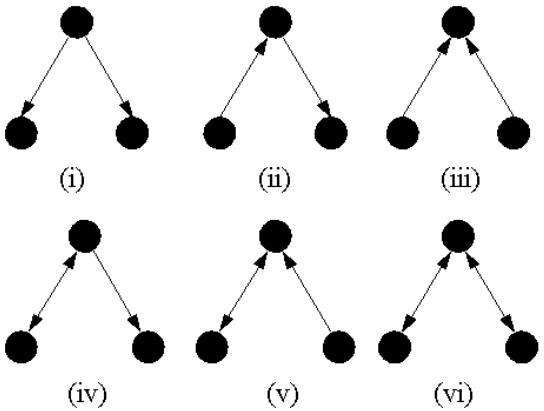
…with big savings in runtime



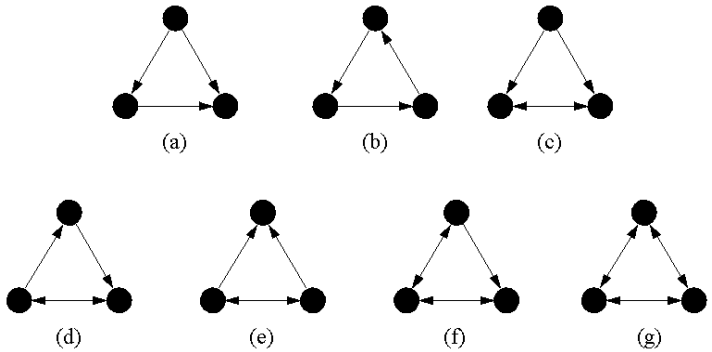
Times normalized with respect to the IO time.

Counting Directed Triangles

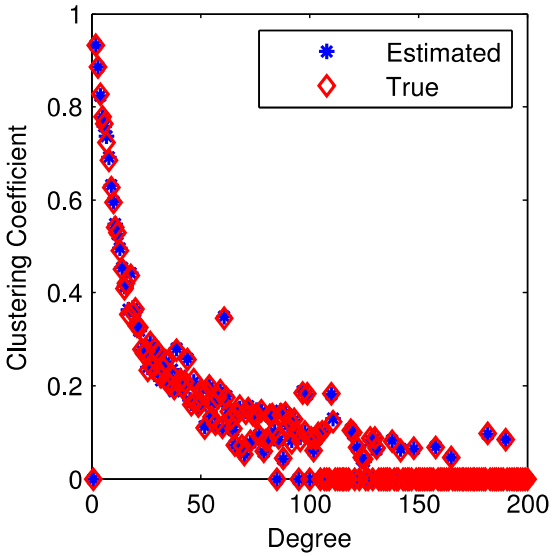
- Multiple occurrences of the same wedge type causes counting the same triangle multiple times.
- Algorithm
  - Pick a wedge-type for the triangle type
  - Compute the success rate
  - t = success rate · w / wedge multiplicity



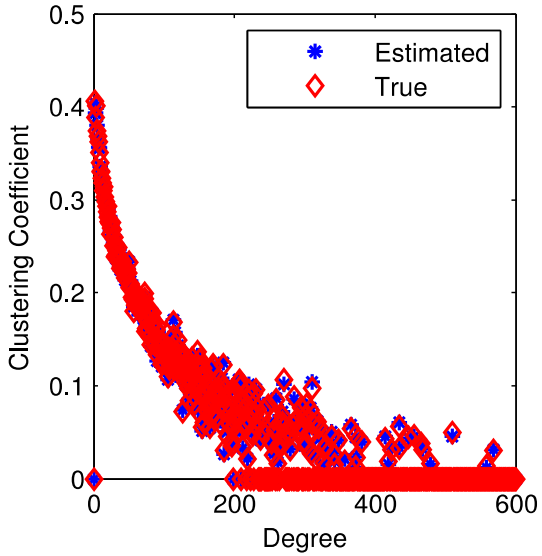
	i	ii	iii	iv	v	vi
a	1	1	1			
b		3				
c	1				2	
d		1		1	1	
e			1	2		
f				1	1	1
g						3



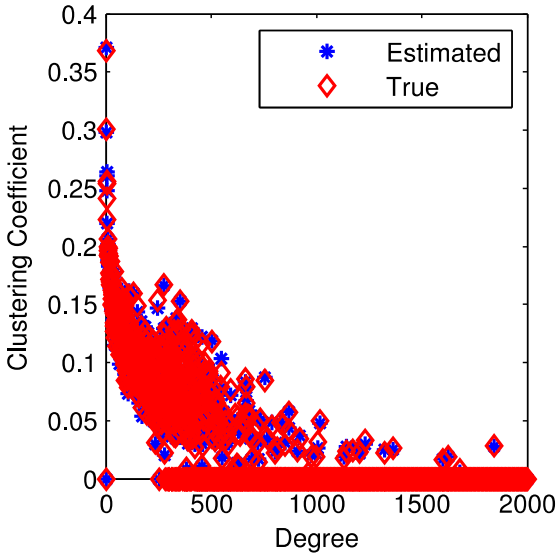
Estimating triangles per degree



ca-CondMat



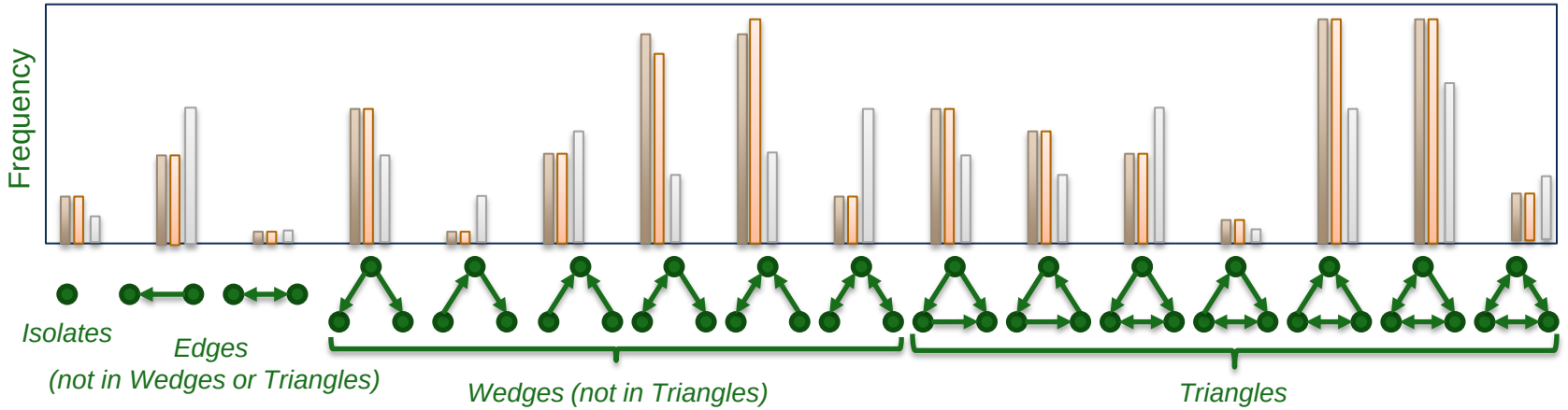
cit-HepPh



soc-Epinions1

- Similar counting strategies apply to counting triangles per degree
- But, we need to adjust the counts based of the number of vertices with the *same degree* in the sampled wedge.

Goal is *Fast* Graph Assays



- Preliminary work on sampling-based approaches
- Clustering coefficient by degree needed to fit/validate BTER model
- MapReduce implementation in progress
- Also have MapReduce attributed subgraph isomorphism code, which can be used for enumeration of infrequent patterns (and validation of sampling method for smaller problems)

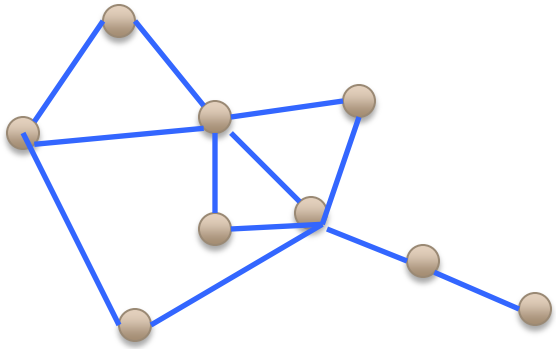
Sandia National Laboratories

University of Minnesota

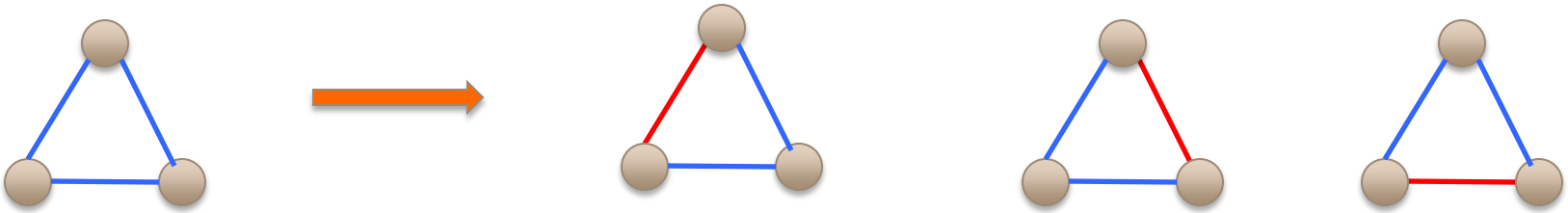
EFFICIENT ALGORITHMS

Enumerating triangles

- Core idea: check whether each wedge is closed.
- Naïve method
  - Runs in roughly quadratic time
  - Redundant work: each triangle is reported 3 times

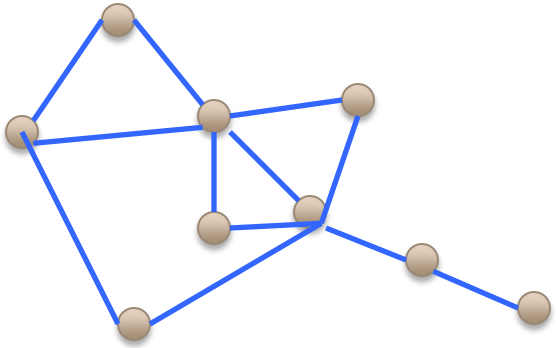


Example with 13 wedges and 1 triangle

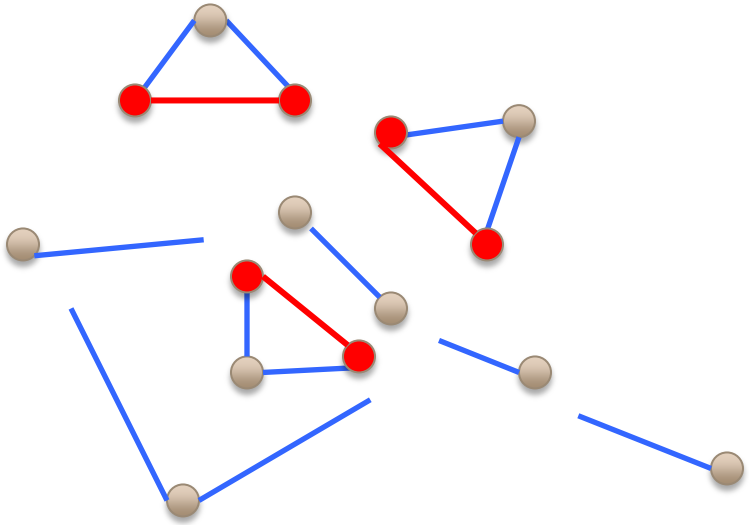


Clever Enumeration

- By imposing an ordering on the vertices (e.g., order by degree), we can check only one wedge per triangle (the one centered on the vertex with min. degree).
- This can be achieved by assigning each edge to its vertex with lower degree.
- Discovered and rediscovered starting in 1985.

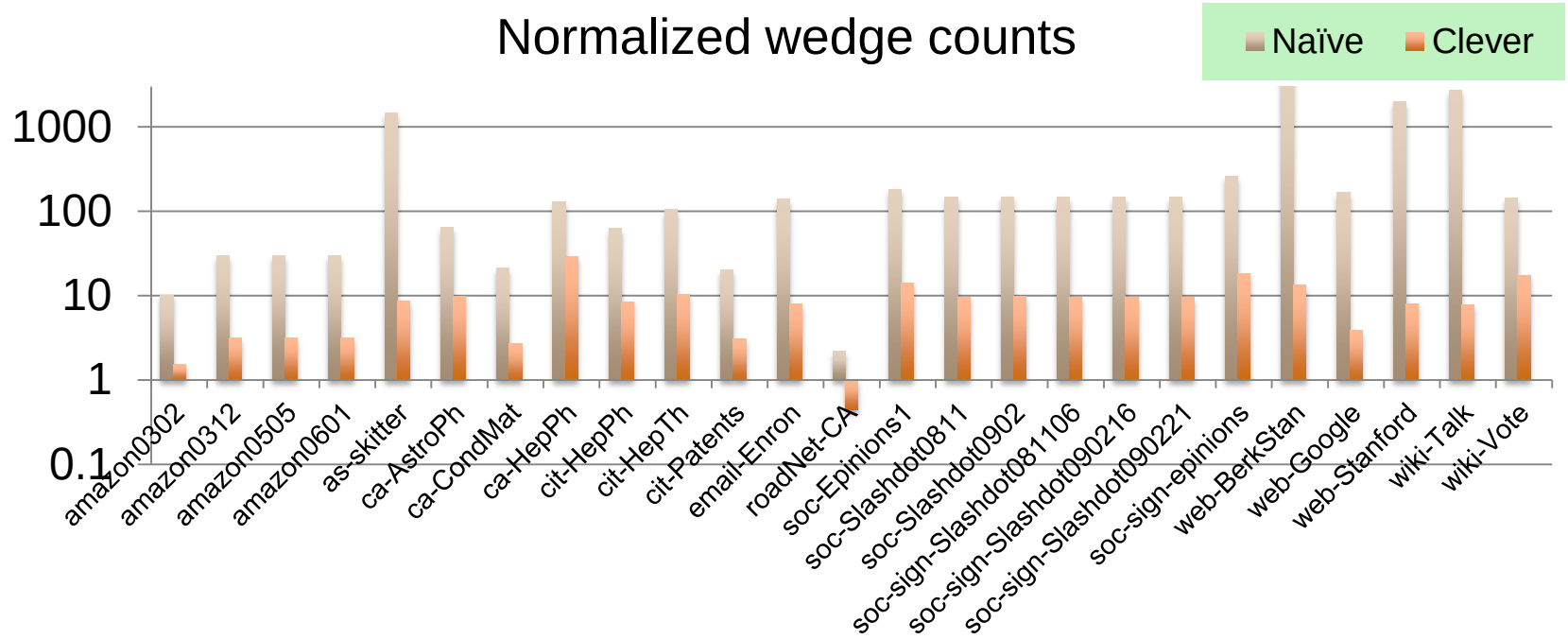


Total wedges: 24



Wedges that need to be checked: 4

Naïve vs. Clever enumeration



- In practice, clever approach is very effective in reducing number of wedges that are checked
- However, it is *surprisingly effective* for many graphs.
- Recent work showed that the clever approach is **linear** in the number of nodes [Berry et al, SAND2010-4474C]

Explicit Triangle Counting Work

- Cleaner version of result is underway…
- Run time of naïve algorithm on CL graphs: $N \cdot \mathbb{E}[d_i^2]$
- Run time of clever algorithm on CL graphs: $M + N \cdot (\mathbb{E}[d_i^{4/3}])^3$
- If degree distribution is power law with exponent γ:
  - If $\gamma > 2$, clever algorithm has faster run time than naïve
  - If $\gamma > 7/3$, run time of clever algorithm is linear!
- Running time related to heaviness of tail
  - A well-studied property becomes useful algorithmically
  - Experiments show that our formula is good
- Tries to formalize intuition that when edges connect disparate degrees, algorithm works well
- Not clear, however, that CL is a good model for real-world graphs…

Sandia National Laboratories

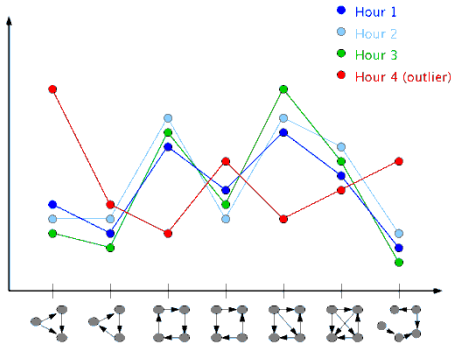
University of Minnesota

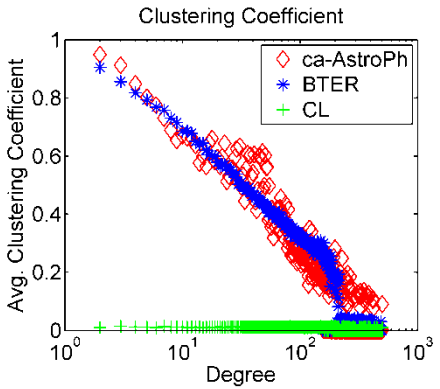
RESEARCH PLAN

FEAST Driving Principles

Objective: Measure, reproduce, and exploit quantifiable characteristics of real-world social network and computer traffic graphs

1. Graph Assays: Sampling-based techniques to compute frequency of common patterns in large-scale graphs
   - Ex: Estimate number of each type of directed triangle
2. Generative Models: Reproduce characteristics shown in the assays
   - Ex: Reproduce clustering coefficient per degree (i.e., triangle behavior)
3. Efficient Algorithms: Take advantage of structure of social networks
   - Ex: Triangle enumeration is provably linear for power law graphs with exponent < 7/3





Validation Data - Cyber

- Sandia collects and stores network traffic data for cyber network defense
- Metadata has been collected for 2 years
  - 10GB/wk (compressed)
  - Stored in Hadoop MapReduce cluster
- Raw pcap also available, but for shorter time period
- Example graph: client IPs with HTTP requests to top-level domains
  - Attributes: request time, URL, server response, etc.
  - Single day graph has 20M edges and 50K vertices (just top level domains)
  - Larger graphs for longer time periods
- Validation
  - Domain experts
  - Geographic IP information
  - Lists of suspicious web sites

Validation Data - MMORPG

- MMORPG: Massively Multiplayer Online Role Playing Games
- People assume characters in a fantasy world
- On average, each player spends 22 hours a week
  - The Psychology of MMORPG, http://www.nickyee.com/~daedalus/gateway_demographics.html
- World-of-Warcraft has 10 million subscribers as of Feb 2012
  - http://investor.activision.com/releasedetail.cfm?ReleaseID=647732
- MMORPG is $20 Billion industry

MMORPG Dataset

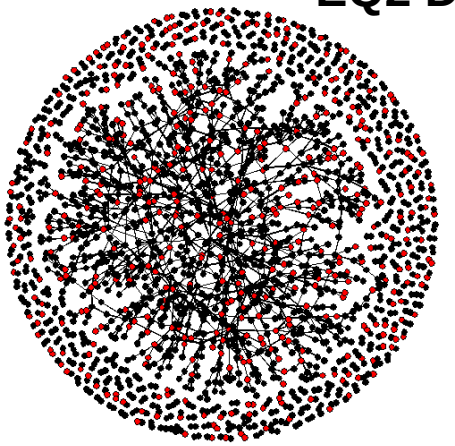
4 month data
100GB data size
Friendship network:
(nodes, edges)
7960447, 76671510

9 month data
2.5TB data size
Trading Network:
(nodes, edges)
54287, 1045521695

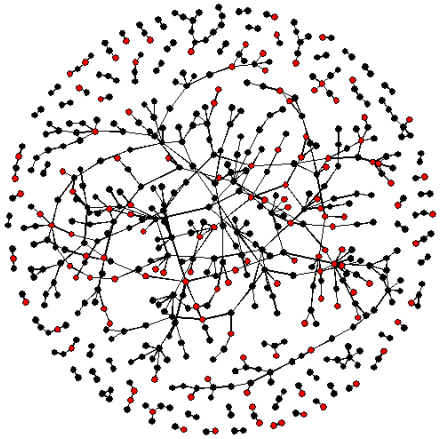
5 month data
200GB data size
Friendship Network:
(nodes, edges)
120690, 6596058

4 month data
160GB data size
Friendship network:
(nodes, edges)
954685, 392851843

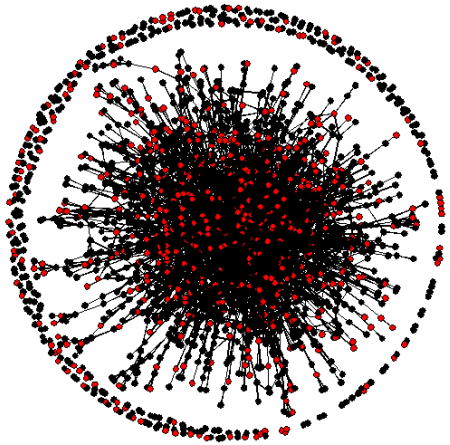
EQ2 Data Set



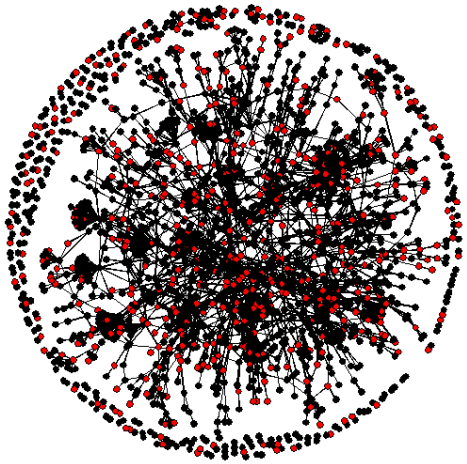
Partnership



Instant messaging

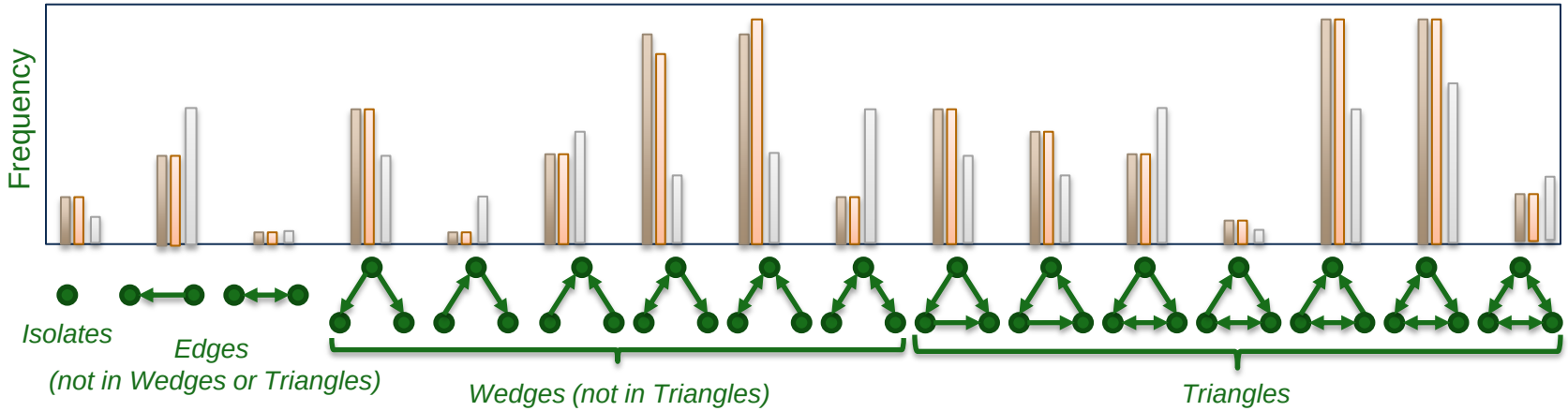


Trade



Mail

Graph Assays



- Graph Assays: Characterization of networks by counting occurrences of specified patterns
  - We will deploy graph assays for evolving networks to Sandia's cybersecurity analysts as a tool for situational awareness
- Context of local interactions must be considered for realistic graph models
  - Directed and/or attributed graphs have a rich structure and hence a larger space of patterns to study
  - Eventually, we will incorporate time dependencies directly into the patterns themselves resulting in time-sensitive assays
- The big challenge is to understand how these various attributes are related to the topology of the graph

Motivation for Assays from Social Sciences Perspective

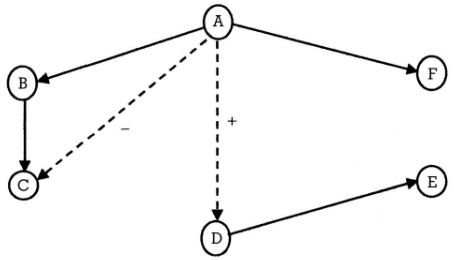
Sandia National Laboratories

University of Minnesota

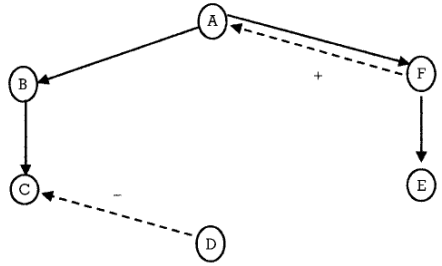
- **Structural signatures corresponds to graph assays**
- Understanding graph assays gives the ability to quantitatively study the human behavior in MMORPG networks
- Macro Analysis:
  - Understand quantitatively the contributions of each structural signature in human relationship networks, such as, friendship, mentorship, housing, trade, etc.
- Micro Analysis:
  - Understand structural signatures of smaller communities
  - Study the relationship of structural signatures across different communities

Understanding structural signatures will enhance our understanding of human behavior in network relationships

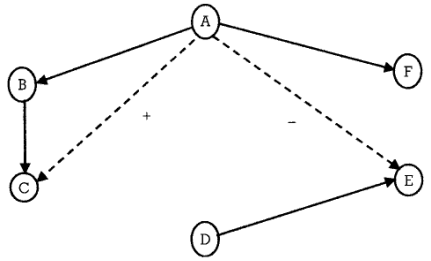
Structural Signatures



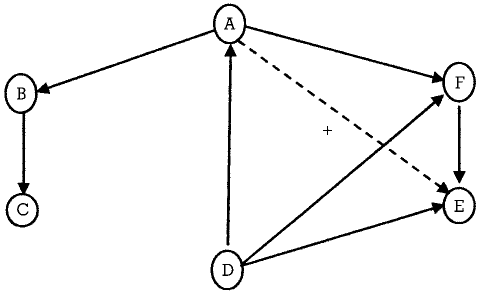
Theory of Structural Holes (Burt, 1992)



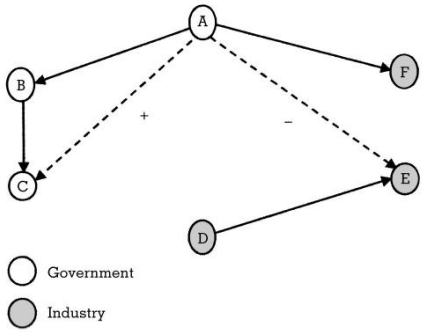
Theory of Social Exchanges (Blau, 1964)



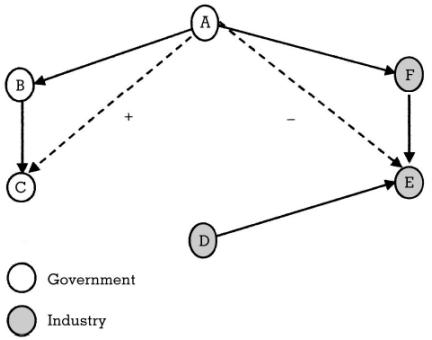
Theory of Cognitive Balance (Heider, 1958)



Theory of Collective Action (Coleman, 1973)



Theory of Homophily (Coleman, 1957)



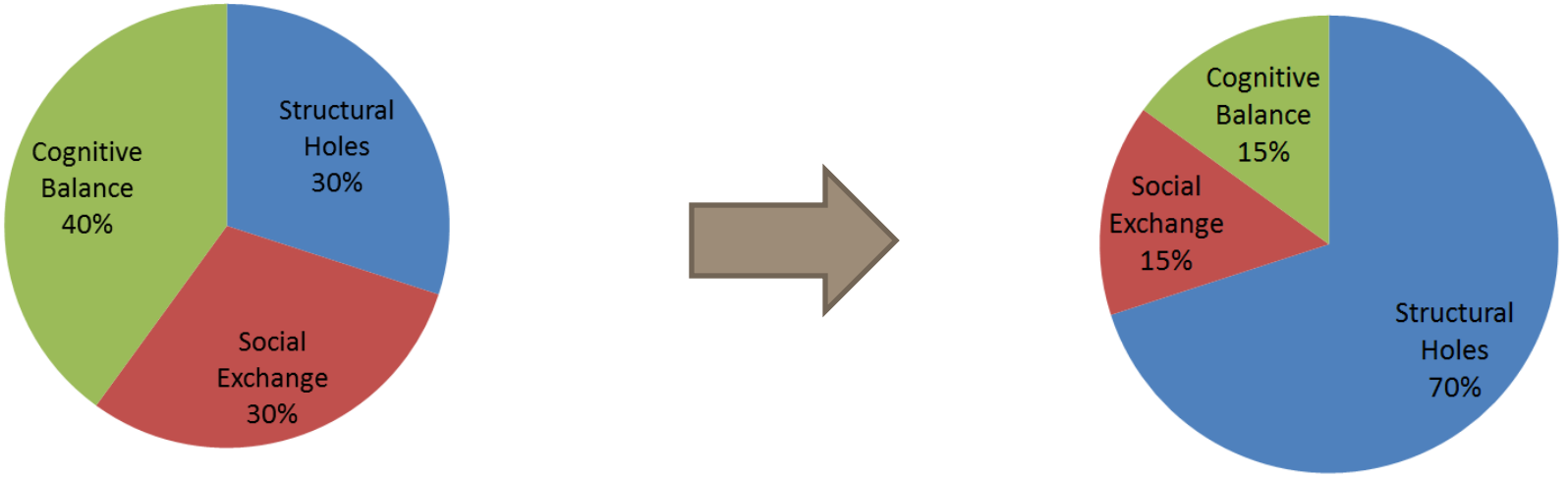
Theory of Balance (Smith-Lovin & Cook, 2001)

Understanding Evolving Community Behavior

Sandia National Laboratories

University of Minnesota

- *Understanding evolving graph assays can tell us how community behavior evolves over time*



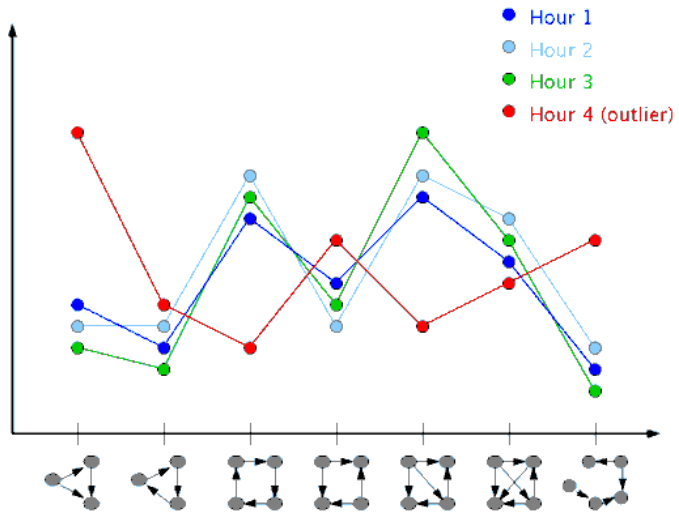
E.g., increase in information exchange points and reduction in overall information capacity

Graph Assay Tasks

- Phase 1
  - Task 1: (SNL) Adapt existing MapReduce software for subgraph isomorphism for graph assays and install at UMN and Sandia
  - Task 2: (UMN) Develop undirected, directed, time-dependent, and attributed assays for MMORPG social networks
  - Task 3: (SNL) Develop undirected, directed, time-dependent, and attributed patterns for computer tracffic networks
  - Task 4: (SNL) Theory explaining undirected and directed observations
- Phase 2
  - Task 11: (SNL) Deploy assays in operational cyber security

Fast Graph Assays

- Major roadblock in using assays is computational challenge of enumerating small subgraphs
  - Likely reason they have not received much attention until now
- Goal: Use sublinear random sampling to develop fast algorithms for creating sampling-based approaches for assays
- Efficient assay computation gives us a simple method for tracking behavior of these subgraph occurrences over time



Fast Graph Assay Tasks

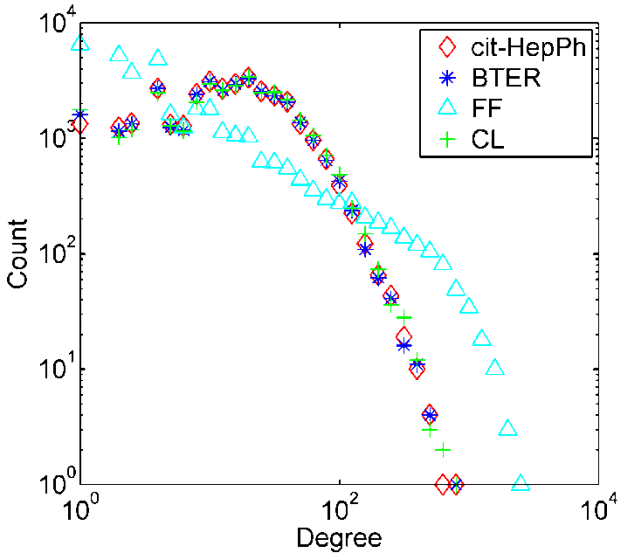
Sandia National Laboratories

University of Minnesota

- Phase I
  - Task 5: (SNL) Methods and theory for undirected patterns
  - Task 6: (SNL) Methods and theory for directed patterns
- Phase II
  - Task 12: (SNL) Methods/theory for time-dependent patterns
  - Task 13: (SNL) Methods/theory for patterns with auxiliary information

From Assays to Models

- Most existing generative graph models have not been carefully validated against real data
- As we discover properties from graph assays, we will formalize them as theoretical properties ("assay statements")
  - Expressed asympotically as densities over certain subgraphs
  - Example: social network graph on n nodes contains a union of O(n) dense ER subgraphs
- This is a principled approach for designing new models, validating them, and pinpointing precise deficiencies
- Scalability: Without resorting to a methodology that grows a network one node or edge at a time (barrier to scalability), we intend to derive a fundamental understanding that enables us to model evolving networks
- We need generative models that add labels and timestamps that make contextual sense as compared to real-world data



Modeling Tasks

- Phase I
  - Task 7: (SNL) Scalable generative model with directed edges
  - Task 8: (SNL, sub UMN) Validate directed model on static network snapshots
- Phase II
  - Task 9: (SNL) Evolving generative model, with directed edges
  - Task 10: (SNL, sub UMN) Validate evolving model on evolving networks
- Phase III
  - Task 14: (SNL) Capture time dependencies in scalable generative model
  - Task 15: (SNL, sub UMN) Validate time-dependencies in model
  - Task 16: (SNL) Capture attributes in scalable generative model
  - Task 17: (SNL, sub UMN) Validate attributes in model

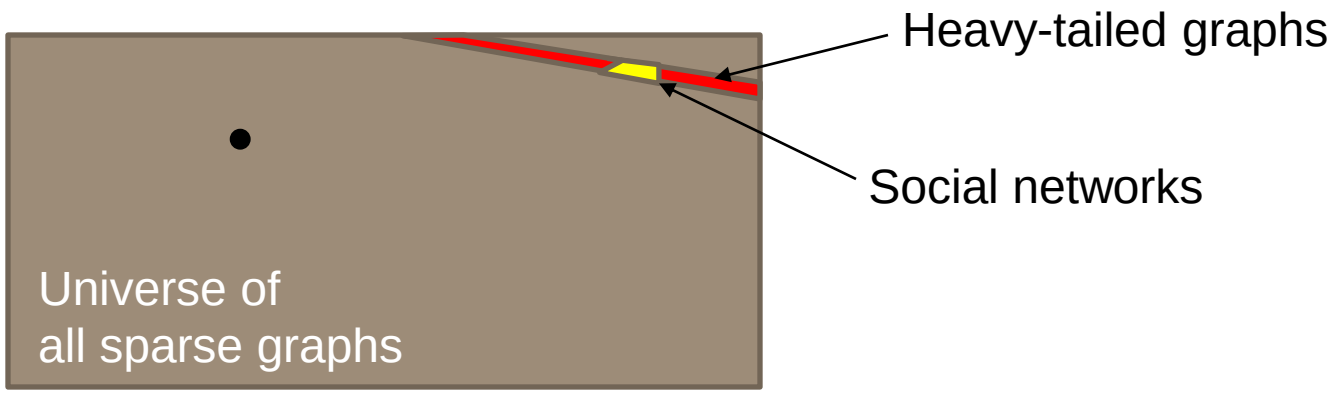
Sampling using Random Walks

- Deep math problem: How does one correctly choose a small "random" sample of a graph that reveals information about its properties?
- We have already shown that by tracking collision properties of a sublinear number of short random walks, we can detect bottlenecks in a graph
- Need to build a theory of random walks for **evolving** graphs

Random Walk Tasks

- Phase II:
  - Task 18: (SNL) Random-walk analysis for heavy-tailed graphs
  - Task 19: (SNL) Random-walk analysis for evolving, heavy-tailed graphs
  - Task 22: (SNL) Notion and algorithms based on random walks
- Phase III
  - Task 27: (SNL) Regularity lemma for social networks

Exploiting Structure



- Conjecture that finding min-conductance cuts may be poly-time solvable for social graphs
- Even though community detection has received much attention, there is no accepted objective function that captures what humans believe to be correct in all cases
- Plan to develop first community detection method to feature graph assays as key idea
- Will consider whether the specific structure of social networks makes it easier to find “large” near cliques efficiently
- We seek a type of regularity lemma for social networks

Structure Tasks

- Phase I:
  - Task 21: (SNL) Prove certain patterns (4-cliques, directed patterns, etc.) can be enumerated efficiently for social networks
- Phase II
  - Task 20: (SNL) Prove minimum-conductance is polynomial on social networks
  - Task 23: (SNL, sub UMN) Hypothesize and validate definition of community that is relevant to social science and validated by surveys on MMORPG participants.
- Phase III
  - Task 25: (SNL, sub UMN): Community-finding methods (not based on random walks) that exploit structures detected in assays
  - Task 26: (SNL) Near max-clique algorithms for social networks.

Sandia National Laboratories

University of Minnesota

BACK-UP SLIDES

Similarity of CL to SKG for Graph 500

Fit CL to the degree distribution produced by SKG for Graph 500 with L = 18