

STING: Spatio-Temporal Interaction Networks and Graphs for Intel Platforms

David Bader, Jason Riedy, Henning Meyerhenke, David Ediger

29 August 2011

**Georgia
Tech**

College of
Computing

Computational Science and Engineering

Outline



Motivation

Technical

- Overall streaming approach

- Clustering coefficients

- Connected components

- Community detection (in progress)

Related

- Pasqual, a scalable de novo sequence assembler

Plans



Health care Finding outbreaks, population epidemiology
Social networks Advertising, searching, grouping
Intelligence Decisions at scale, regulating algorithms
Systems biology Understanding interactions, drug design
Power grid Disruptions, conservation
Simulation Discrete events, cracking meshes

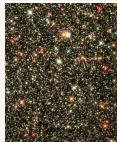
Graphs are pervasive



- Sources of massive data: petascale simulations, experimental devices, the Internet, scientific applications.
- New challenges for analysis: data sizes, heterogeneity, uncertainty, data quality.

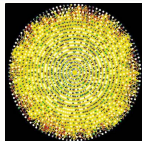
Astrophysics

Problem Outlier detection
Challenges Massive data sets, temporal variation
Graph problems Matching, clustering



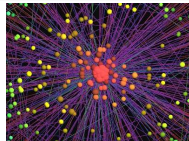
Bioinformatics

Problem Identifying target proteins
Challenges Data heterogeneity, quality
Graph problems Centrality, clustering



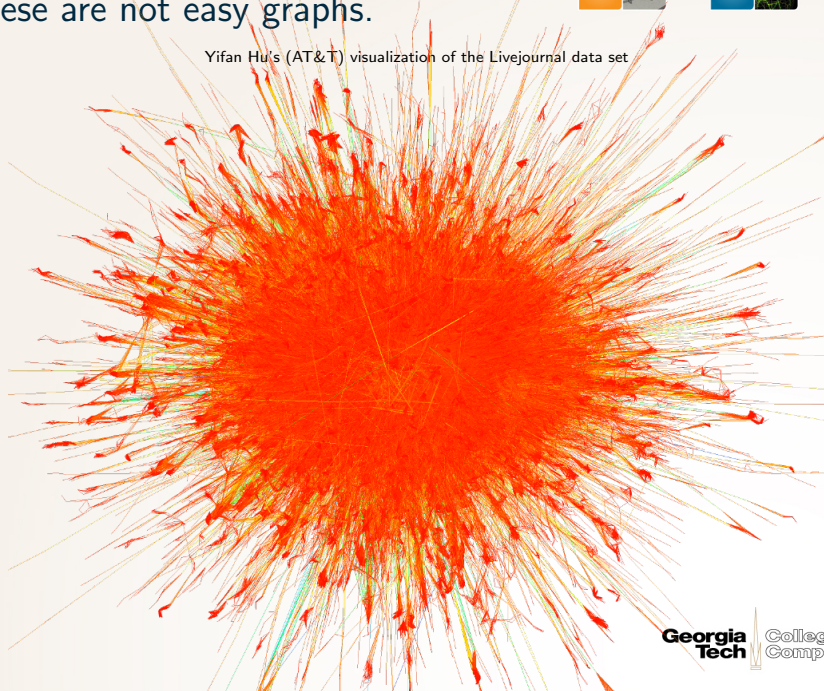
Social Informatics

Problem Emergent behavior, information spread
Challenges New analysis, data uncertainty
Graph problems Clustering, flows, shortest paths

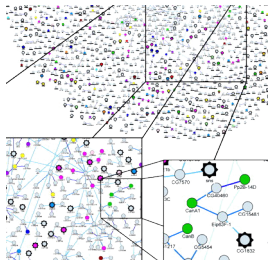


These are not easy graphs.

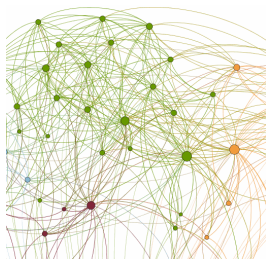
Yifan Hu's (AT&T) visualization of the Livejournal data set



Overall streaming approach



Protein interactions, Giot *et al.*, "A Protein Interaction Map of *Drosophila melanogaster*", Science 302, 1722-1736, 2003.

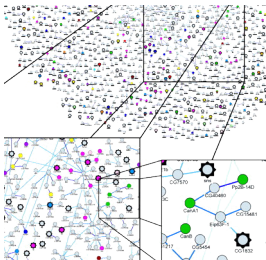


Jason's network via LinkedIn Labs

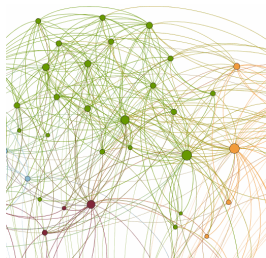
Assumptions

- A graph represents some real-world phenomenon.
 - But **not** necessarily exactly!
 - Noise comes from lost updates, partial information, ...

Overall streaming approach



Protein interactions, Giot *et al.*, "A Protein Interaction Map of *Drosophila melanogaster*", Science 302, 1722-1736, 2003.

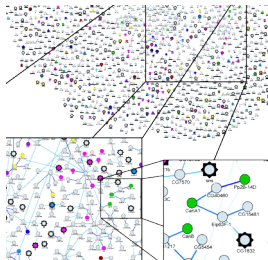


Jason's network via LinkedIn Labs

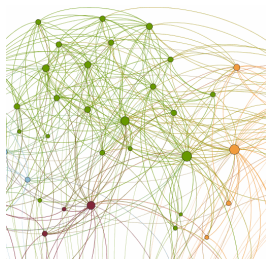
Assumptions

- We target massive, "social network" graphs.
 - Small diameter, power-law degrees
 - Small changes in massive graphs often are unrelated.

Overall streaming approach



Protein interactions, Giot *et al.*, "A Protein Interaction Map of *Drosophila melanogaster*", Science 302, 1722-1736, 2003.



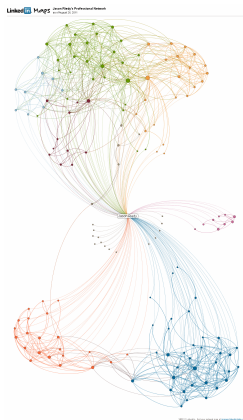
Jason's network via LinkedIn Labs

Assumptions

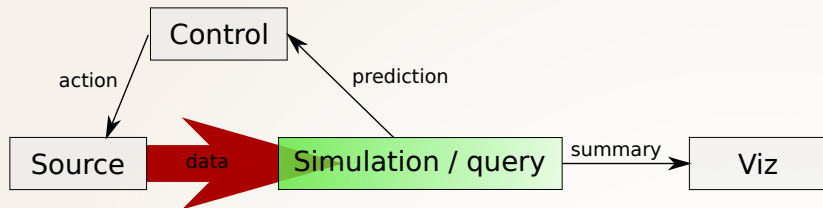
- The graph changes but we don't need a continuous view.
 - We can accumulate changes into batches...
 - But not so many that it impedes responsiveness.

Difficulties for performance

- What partitioning methods apply?
 - Geometric? Nope.
 - Balanced? Nope.
 - Is there a single, useful decomposition? Not likely.
- Some *partitions* exist, but they don't often help with balanced bisection or memory locality.
- Performance needs new approaches, not just standard scientific computing methods.

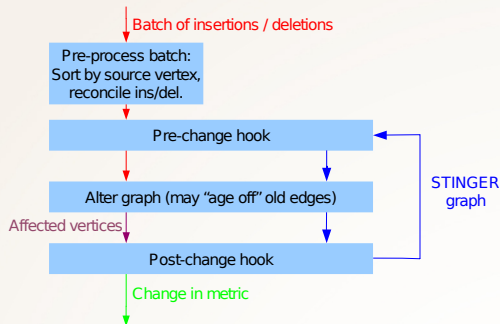


Jason's network via LinkedIn Labs



- STING manages queries against changing graph data.
 - Visualization and control often are application specific.
- Ideal: Maintain many persistent graph analysis kernels.
 - Keep one current snapshot of the graph resident.
 - Let kernels maintain smaller histories.
 - Also (a harder goal), coordinate the kernels' cooperation.

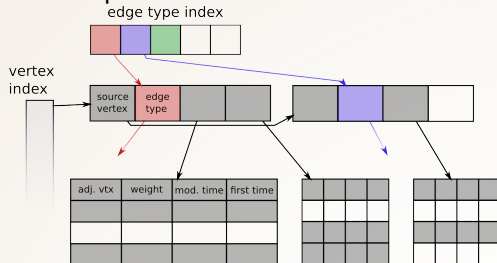
STING and STINGER



- Batches provide dual-level parallelism.
 - Busy loci of change: Know to share the busy points.
 - Scattered changes: Parallel across (likely) independent changes.
- The massive graph is maintained in a data structure named STINGER.



STING Extensible Representation:



- Rule #1: **No explicit locking.**
 - Rely on atomic operations.
- Massive graph: Scattered updates, scattered reads rarely conflict.
- Use time stamps for some view of time.



Prototype STING and STINGER

Monitoring the following properties:

- 1 clustering coefficients,
- 2 connected components, and
- 3 community structure (in progress).

High-level

- Support high rates of change, over 10k updates per second.
- Performance scales somewhat with available processing.
- Gut feeling: Scales as much with *sockets* as cores.

<http://www.cc.gatech.edu/~bader/code.html>



Unless otherwise noted

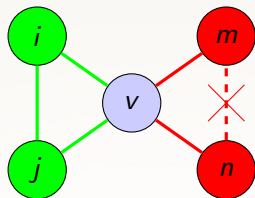
Line	Model	Speed (GHz)	Sockets	Cores
Nehalem	X5570	2.93	2	4
Westmere	E7-8870	2.40	4	10

- Westmere loaned by Intel (*thank you!*)
- All memory: 1067MHz DDR3, installed appropriately
- Implementations: OpenMP, gcc 4.6.1, Linux \approx 3.0 kernel
- Artificial graph and edge stream generated by R-MAT[Chakrabarti, et al.]
 - Scale x , edge factor $f \Rightarrow 2^x$ vertices, $\approx f \cdot 2^x$ edges.
 - Edge actions: 7/8th insertions, 1/8th deletions
 - Results over five batches of edge actions.
- **Caveat:** No vector instructions, low-level optimizations yet.

Clustering coefficients



- Used to measure “small-world-ness” [Watts and Strogatz] and potential community structure
- Larger clustering coefficient \Rightarrow more inter-connected
- Roughly the ratio of the number of actual to *potential* triangles



- Defined in terms of **triplets**.
- $i - v - j$ is a **closed triplet** (triangle).
- $m - v - n$ is an **open triplet**.
- Clustering coefficient:
$$\frac{\# \text{ of closed triplets}}{\text{total } \# \text{ of triplets}}$$
- Locally around v or globally for entire graph.

Updating triangle counts



Given Edge $\{u, v\}$ to be inserted (+) or deleted (-)

Approach Search for vertices adjacent to both u and v , update counts on those and u and v

Three methods

Brute force Intersect neighbors of u and v by iterating over each, $O(d_u d_v)$ time.

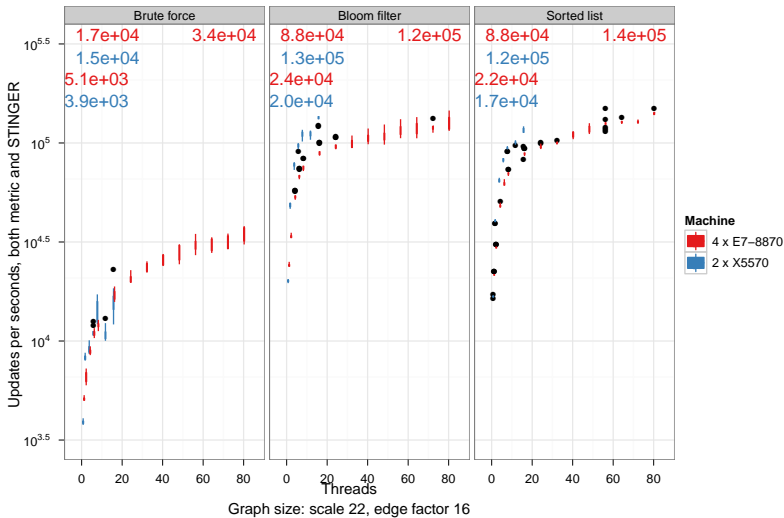
Sorted list Sort u 's neighbors. For each neighbor of v , check if in the sorted list.

Compressed bits Summarize u 's neighbors in a bit array. Reduces check for v 's neighbors to $O(1)$ time each.

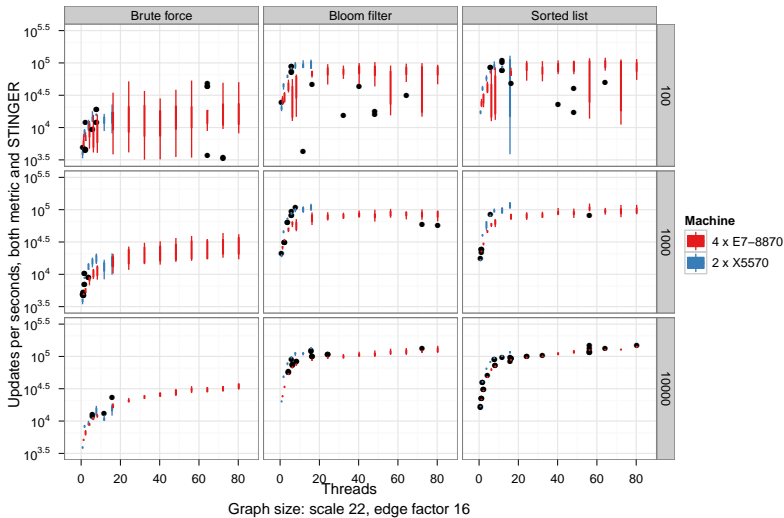
Approximate with Bloom filters. [MTAAP10]

All rely on atomic addition.

Batches of 10k actions



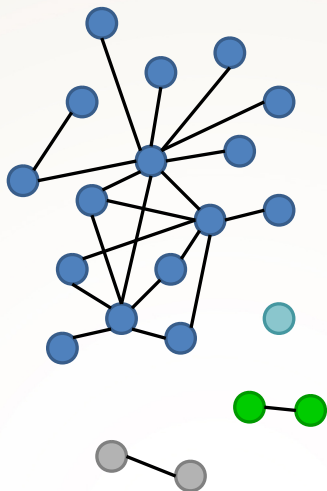
Different batch sizes



Connected components



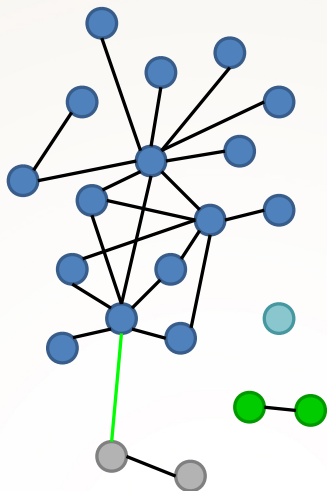
- Maintain a mapping from vertex to component.
- *Global* property, unlike triangle counts
- In “scale free” social networks:
 - Often one big component, and
 - many tiny ones.
- Edge changes often sit *within* components.
- Remaining insertions merge components.
- Deletions are more difficult...



Connected components



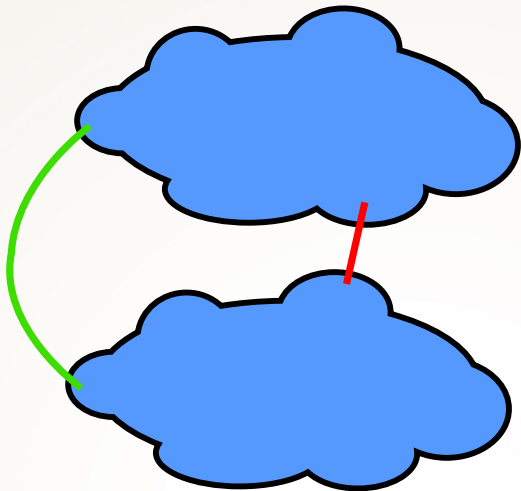
- Maintain a mapping from vertex to component.
- *Global* property, unlike triangle counts
- In “scale free” social networks:
 - Often one big component, and
 - many tiny ones.
- Edge changes often sit *within* components.
- Remaining insertions merge components.
- Deletions are more difficult...





The difficult case

- Very few deletions matter.
- Determining *which* matter may require a large graph search.
 - Re-running static component detection.
 - (Long history, see related work in [MTAAP11].)
- Coping mechanisms:
 - *Heuristics*.
 - Second level of batching.



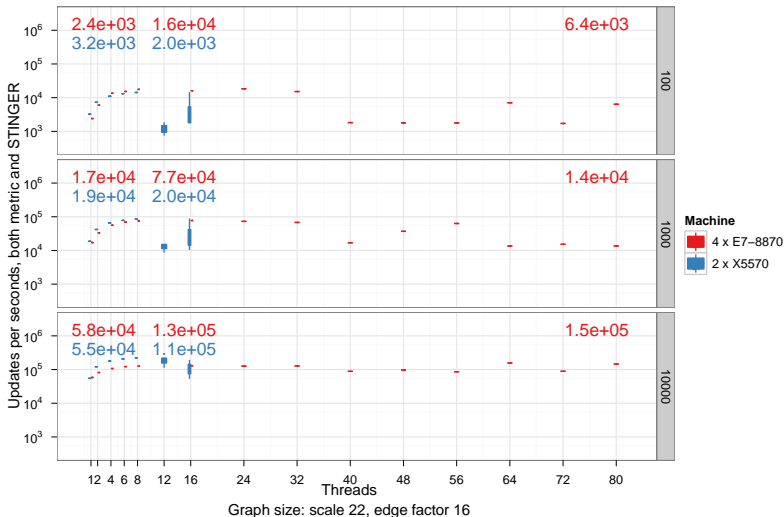


Rule out effect-less deletions

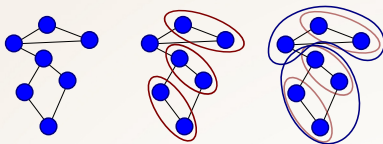
- Use the *spanning tree* by-product of static connected component algorithms.
- Ignore deletions when one of the following occur:
 - ① The deleted edge is not in the spanning tree.
 - ② If the endpoints share a common neighbor*.
 - ③ If the loose endpoint can reach the root*.
- In the last two (*), also fix the spanning tree.

Rules out 99.7% of deletions.

Connected components: Performance

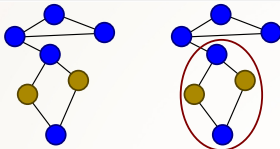


Community detection (work in progress)



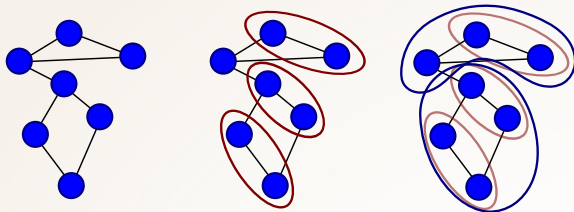
Greedy, agglomerative partitioning

- Partition to maximize modularity, minimize conductance, ...



Seed set expansion

- Grow an optimal / "relevant" community around selection.
- (Work with Jonny Dimond of KIT.)



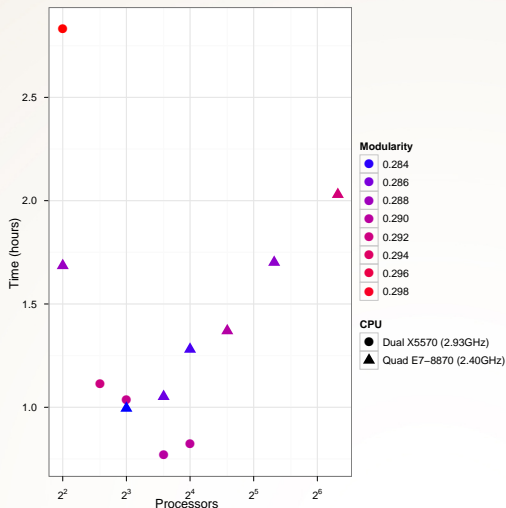
Parallel greedy, agglomerative partitioning [PPAM11]

- Score edges by optimization criteria.
- Chose a maximal, heavy-weight matching.
 - Negate edge scores if minimizing conductance.
- Contract those edges.
- Mimics sequential optimizers, but produces different results.

Performance



- R-MAT on right.
- Livejournal
 - 15M vertex, 184M edge
 - 6-12 hours on E7-8870
- Highly variable performance.
- Algorithm under development.





A scalable *de novo* assembler

Work by Henning Meyerhenke, Xing Liu, Pushkar Pande.

- Next-generation sequencers produce mountains of small gene sequences.
- Assembling into a genome: Yet another large graph problem.
- Pasqual forms a compressed *overlap graph* and traces paths.
- **Only scalable and correct shared-memory assembler.**
 - Faster *and* uses less memory than other existing systems.
 - Evaluation against the few distributed assemblers is ongoing.

<http://www.cc.gatech.edu/pasqual/>



Similar speed, better results

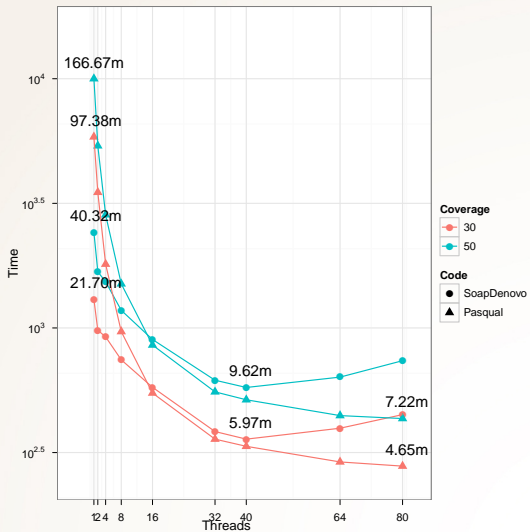
Length	Code	Time (min)	N50 (bp)	Errors
35	Velvet	>12h	1355	683
	Edena	451.15	1375	3
	ABySS	56.52	1412	521
	SOAPdenovo	15.62	1470	485
	<i>Pasqual</i>	15.50	1451	5
100	Velvet	132.68	6635	175
	Edena	466.12	6545	7
	ABySS	92.92	6229	136
	SOAPdenovo	18.05	6879	142
	<i>Pasqual</i>	19.15	7712	6



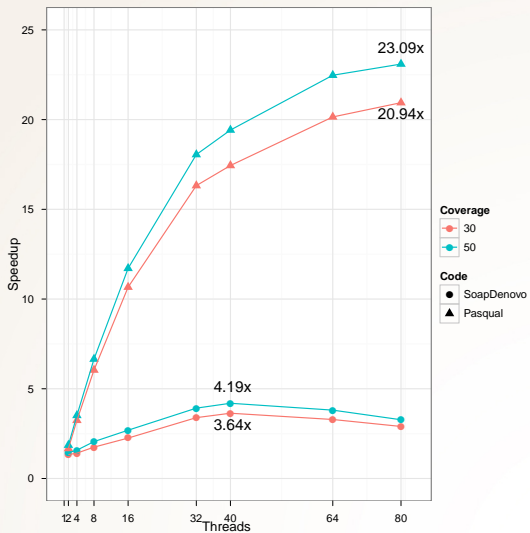
Far better speed *and* results

Length	Code	Time (min)	N50 (bp)	Errors
100	Velvet	256.02	4045	799
	Edena	>12h	2661	0
	ABYSS	—	—	—
	SOAPdenovo	31.83	3998	725
	<i>Pasqual</i>	57.27	4535	0
200	Velvet	—	—	—
	Edena	—	—	—
	ABYSS	—	—	—
	SOAPdenovo	—	—	—
	<i>Pasqual</i>	38.07	7911	0

Performance v. SOAPdenovo



Speed-up





Community detection Improving the algorithm, pushing into streaming by de-agglomerating and restarting.




Seed set expansion Maintaining not only one expanded set, but multiple for high-throughput monitoring.

Microbenchmarks Expand on initial promising work on characterizing performance by peak number of memory operations achieved, find bottlenecks by comparing with microbenchmarks.



Distributed/PGAS STINGER fits a PGAS model well (think SCC). Interested in exploring distributed algorithms.

Packaging Wrap STING into an easily downloaded and installed tool.





-  D. Ediger, K. Jiang, E. J. Riedy, and D. A. Bader.
Massive streaming data analytics: A case study with clustering coefficients.
In Proceedings of the Workshop on Multithreaded Architectures and Applications (MTAAP'10), Apr. 2010.
-  D. Ediger, E. J. Riedy, and D. A. Bader.
Tracking structure of streaming social networks.
In Proceedings of the Workshop on Multithreaded Architectures and Applications (MTAAP'11), May 2011.
-  K. Madduri and D. A. Bader.
Compact graph representations and parallel connectivity algorithms for massive dynamic network analysis.
In 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rome, Italy, May 2009.



-  E. J. Riedy, D. A. Bader, K. Jiang, P. Pande, and R. Sharma. Detecting communities from given seeds in social networks. Technical Report GT-CSE-11-01, Georgia Institute of Technology, Feb. 2011.
-  E. J. Riedy, H. Meyerhenke, D. Ediger, and D. A. Bader. Parallel community detection for massive graphs. In *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics*, Torun, Poland, Sept. 2011.



-  D. Chakrabarti, Y. Zhan, and C. Faloutsos.
R-MAT: A recursive model for graph mining.
In *Proc. 4th SIAM Intl. Conf. on Data Mining (SDM)*, Orlando, FL, Apr. 2004. SIAM.
-  D. J. Watts and S. H. Strogatz.
Collective dynamics of 'small-world' networks.
Nature, 393(6684):440–442, Jun 1998.