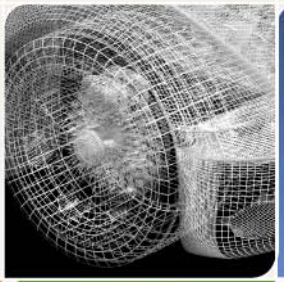
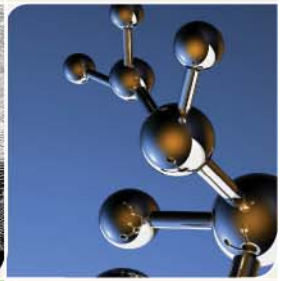
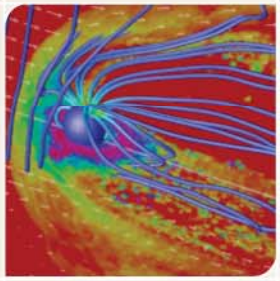
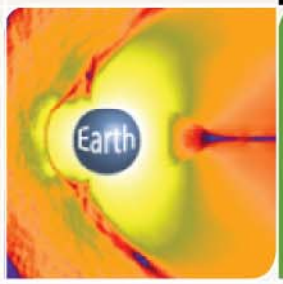


Parallel Algorithms for Small-world Network Analysis and Partitioning (SNAP)

David A. Bader and Kamesh Madduri

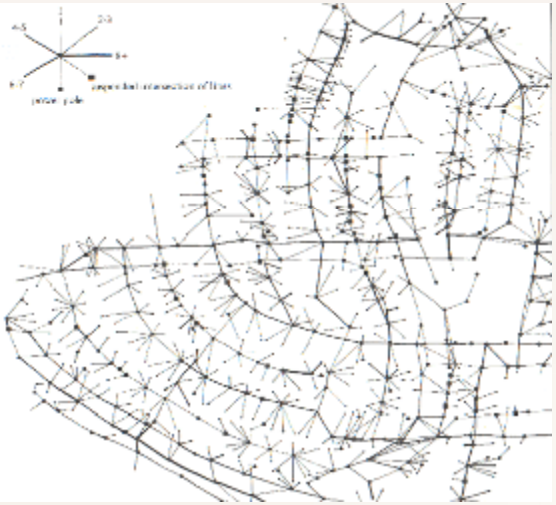
Georgia Tech College of Computing
Computational Science and Engineering


Overview

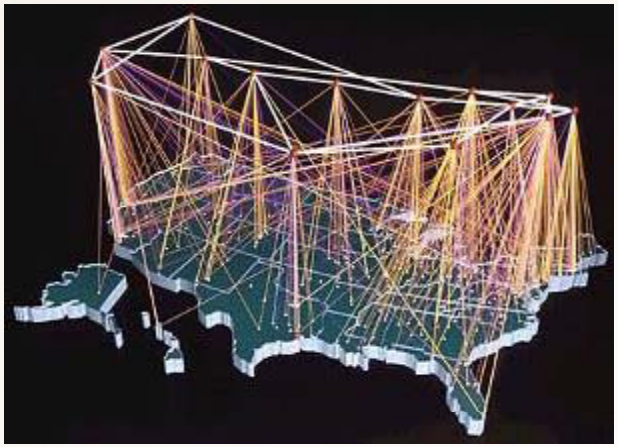
- Informatics networks, small-world topology
- Community Identification/Graph partitioning
- The modularity measure for community structure detection
- SNAP: An open-source, parallel graph framework for exploratory analysis of large-scale networks
- Parallel Community Identification Algorithms for large-scale networks

Georgia Tech College of Computing
Computational Science and Engineering

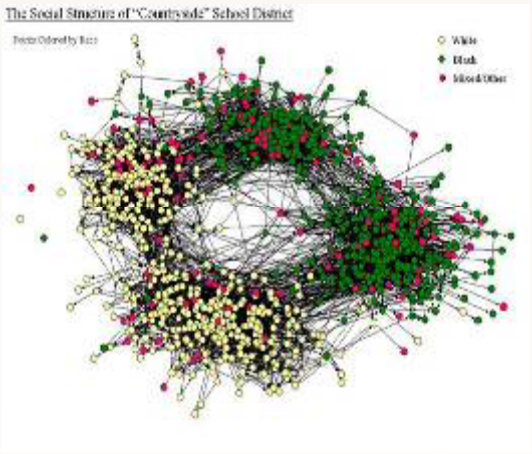
Power Distribution Networks



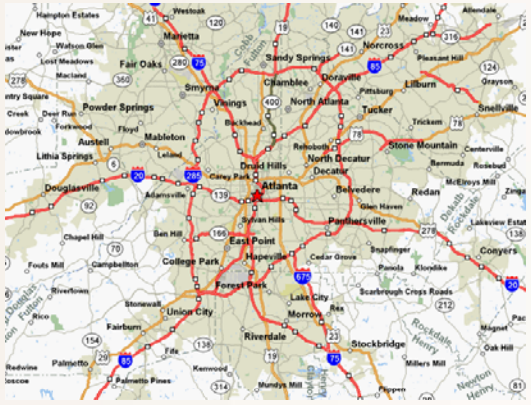
Internet backbone



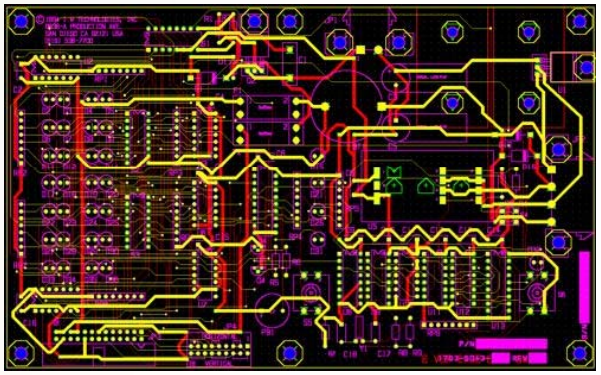
Social Networks



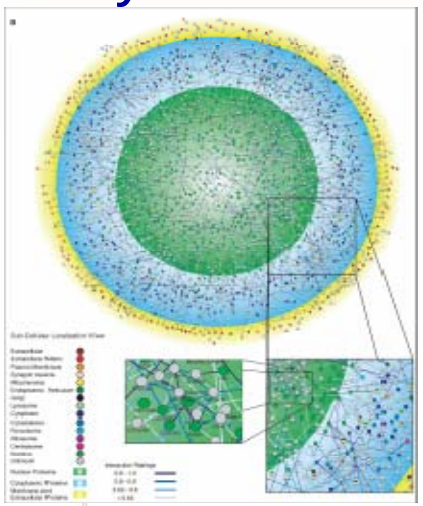
Large scale graph problems arise from a variety of sources



Transportation Networks



VLSI Design



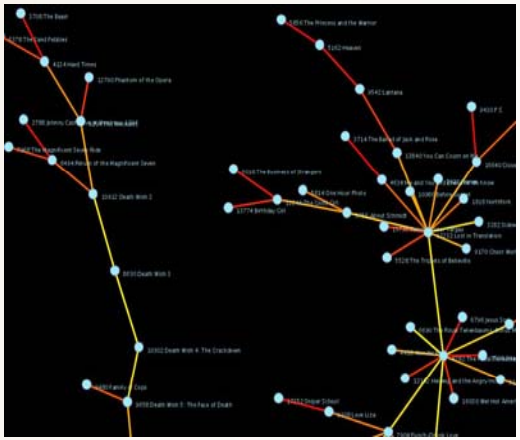
Computational Biology

Georgia Tech College of Computing
Computational Science and Engineering

Small-world Networks

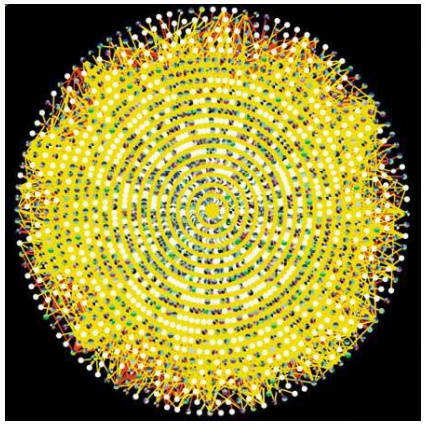
- Graph abstractions and algorithms are extensively used to analyze massive social, technological and biological datasets

Social networks



Community identification, Security and surveillance, viral marketing.

Computational Biology



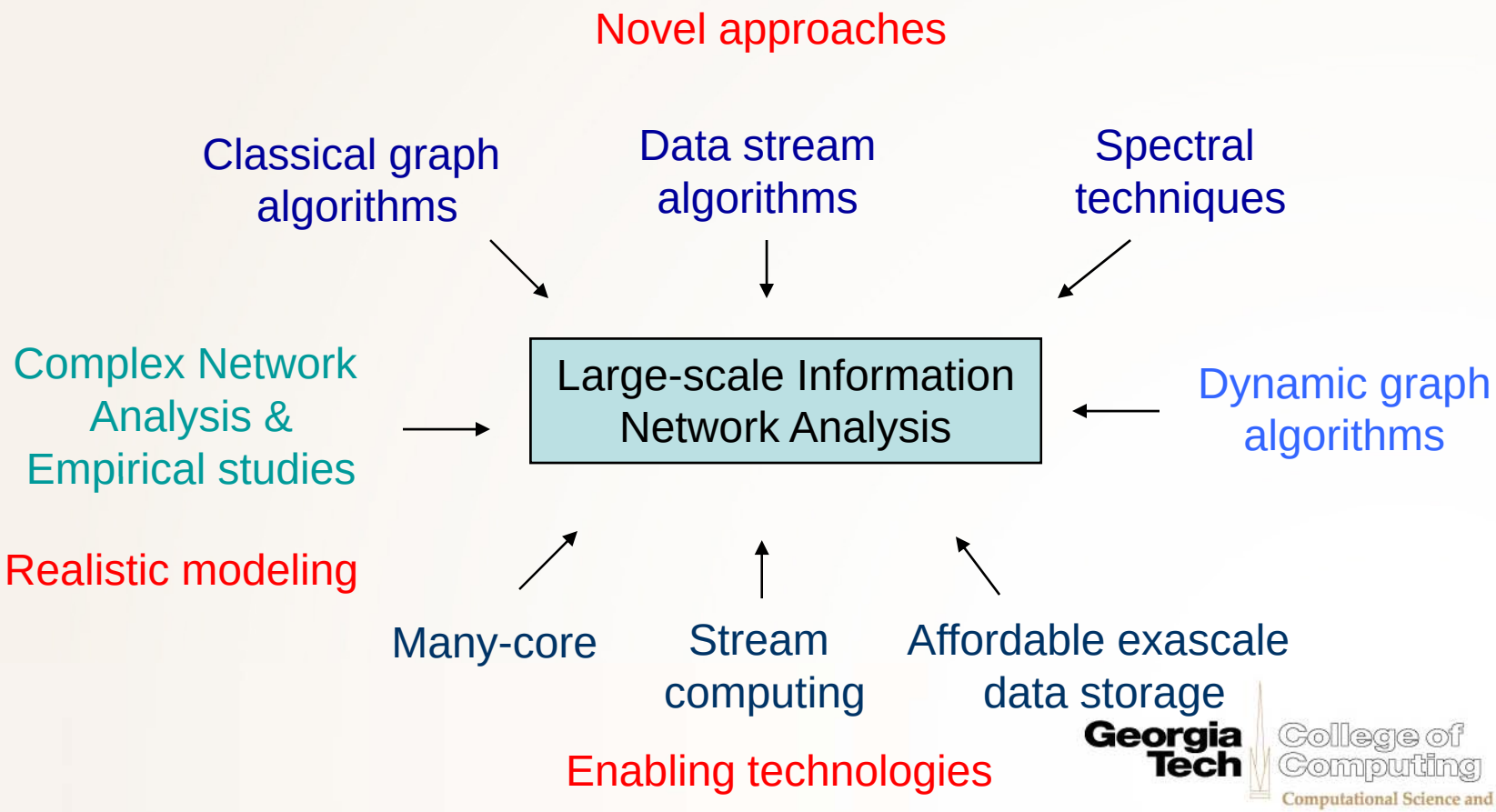
Systems biology, disease modeling, behavioral ecology.

Images sources: visualcomplexity.com

Georgia Tech College of Computing
Computational Science and Engineering

Interaction Networks

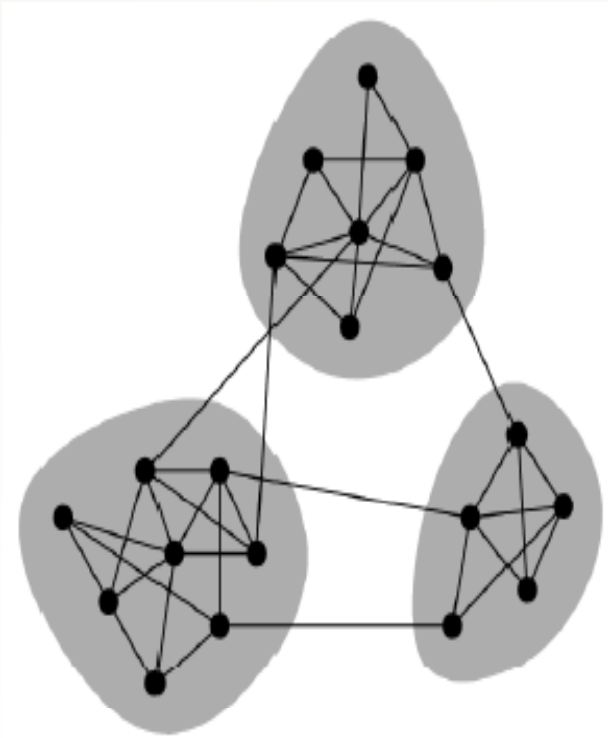
- Analysis of massive small-world interaction networks poses new computational challenges



Georgia Tech College of Computing
Computational Science and Engineering

Community Structure

- Implicit communities in large-scale networks are of interest in many cases
  - WWW
  - Social networks
  - Citation networks
- Community structure detection is usually formulated as a graph clustering problem
- Properties of communities may be different from properties of the entire network



Georgia Tech College of Computing
Computational Science and Engineering



Clustering/Partitioning/Min Cut

PROBLEM	Clustering	MinCut	Graph Partitioning
Objective	Minimize an appropriate measure	Minimize cut size	Minimize cut size
Balance of partition	Sizes may differ	Sizes may differ	Equal sizes
Cardinality of partition	To be computed	To be computed	Input parameter



Community Identification/Graph partitioning

- Graph partitioning is a related problem, but not identical to community identification
- Objective function in graph partitioning: minimize the edge cut, while trying to balance the number of vertices in each partition
  - Metrics: coverage, conductance, performance
- Objective function in clustering: informally, identify/extract “dense” sub-graphs
  - Metrics: intra-cluster vs. inter-cluster edges


Related Work: Partitioning Algorithms from Scientific Computing

- Theoretical and empirical evidence suggests that existing techniques from scientific computing perform poorly on small-world networks
- Low diameter and heavy-tailed degree distribution pose a challenge
- [Mihail, Papadimitriou '02] Spectral properties of power-law graphs are skewed in favor of high-degree vertices
- [Lang '04] On using spectral techniques, "Cut quality varies inversely with cut balance" in social graphs: Yahoo! IM graph, DBLP collaborations
- [Abou-Rjeili, Karypis '06] Multilevel partitioning heuristics give large edge-cut for small-world networks, new coarsening schemes necessary

Modularity

- [Newman, Girvan '03] Measure based on *optimizing inter-cluster density over intra-cluster sparsity.*
- $C = (C_1, ..., C_k)$: partition of V such that $C_i \neq \phi$ and $C_i \cap C_j = \phi$; $m(C_i)$: no. of intra-cluster edges

$$Q(C) = \sum_i \left[\frac{m(C_i)}{m} - \left(\frac{\sum_{v \in C_i} \deg(v)}{2m} \right)^2 \right]$$

- If a particular clustering has no more intra-cluster edges than would be expected by random chance, we have $Q = 0$
- Values greater than 0.3 indicate community structure in a network
- [Brandes '07] Maximizing modularity is NP-complete



Modularity-maximizing community identification algorithms

Algorithm	Clustering Approach	Algorithmic Technique	Complexity
Newman, Girvan '03	global, divisive	greedy, centrality-based	$O(n^3)$
Clauset et al. '04	global, agglomerative	greedy	$O(n\log^2 n)$
Duch, Arenas '05	global, divisive	extremal optimization	$O(n^2\log^2 n)$
Djidjev '06	global, divisive	greedy, multi-level partitioning	$O(n^2)$
Newman '06	global, divisive	spectral	$O(n^2\log n)$
Brandes '07	agglomerative	greedy	$O(n^2\log n)$

Georgia Tech College of Computing
Computational Science and Engineering


Community Identification

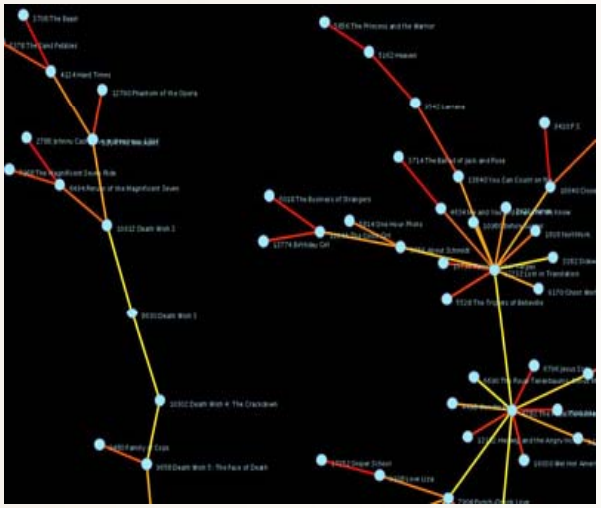
- Known algorithms are compute-intensive: $O(n^2)$, and in some cases $O(n^3)$
- Current approaches do not exploit the small-world network topology
- How beneficial is preprocessing?
- Can we design faster heuristics?

SNAP: A parallel graph framework for small-world networks

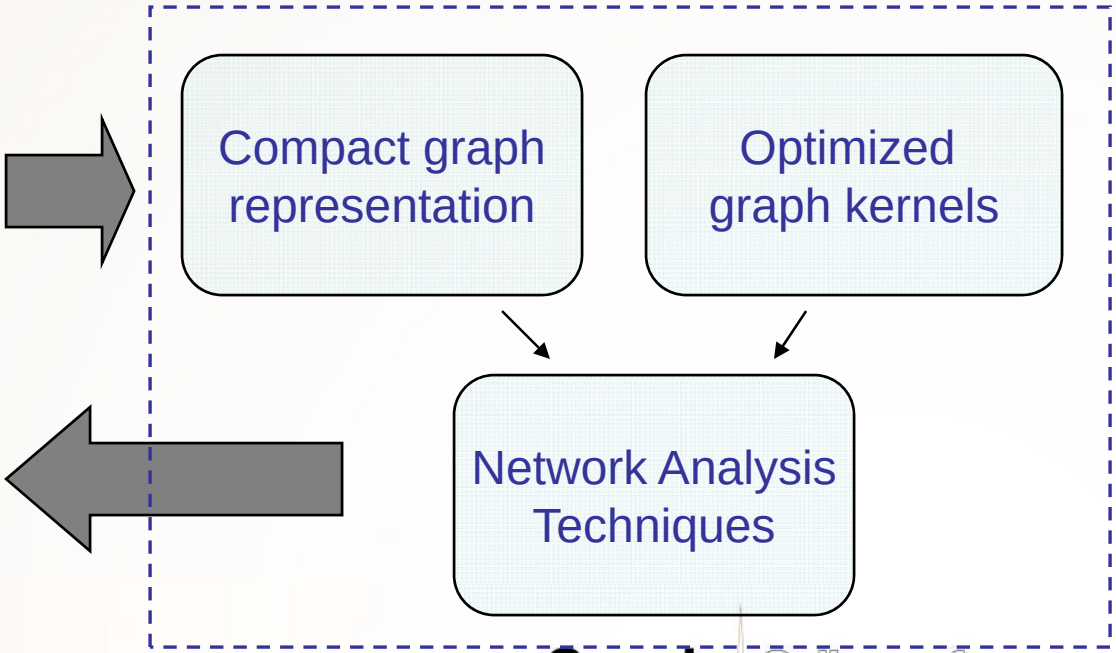
Georgia Tech College of Computing
Computational Science and Engineering

SNAP

- An open-source parallel graph framework for analyzing small-world interaction networks
- Parallel algorithms optimized for shared memory manycore, SMP and multithreaded systems



Interaction data



Georgia Tech College of Computing
Computational Science and Engineering


SNAP Framework

- We have designed fast parallel algorithms and efficient implementations for several graph theoretic problems
  - Graph representation
  - Graph kernels: List ranking, Connected Components [ICPP05], Spanning tree [JPDC06], MST [IPDPS04], Graph traversal [ICPP06], Shortest paths [ALENEX07, MTAAP07]
  - Algorithms: Centrality analysis [ICPP06], community identification [Madduri/Bader 07]
  - Applications: Protein-interaction networks [HiCOMB06], social network analysis [Madduri/Bader 07]

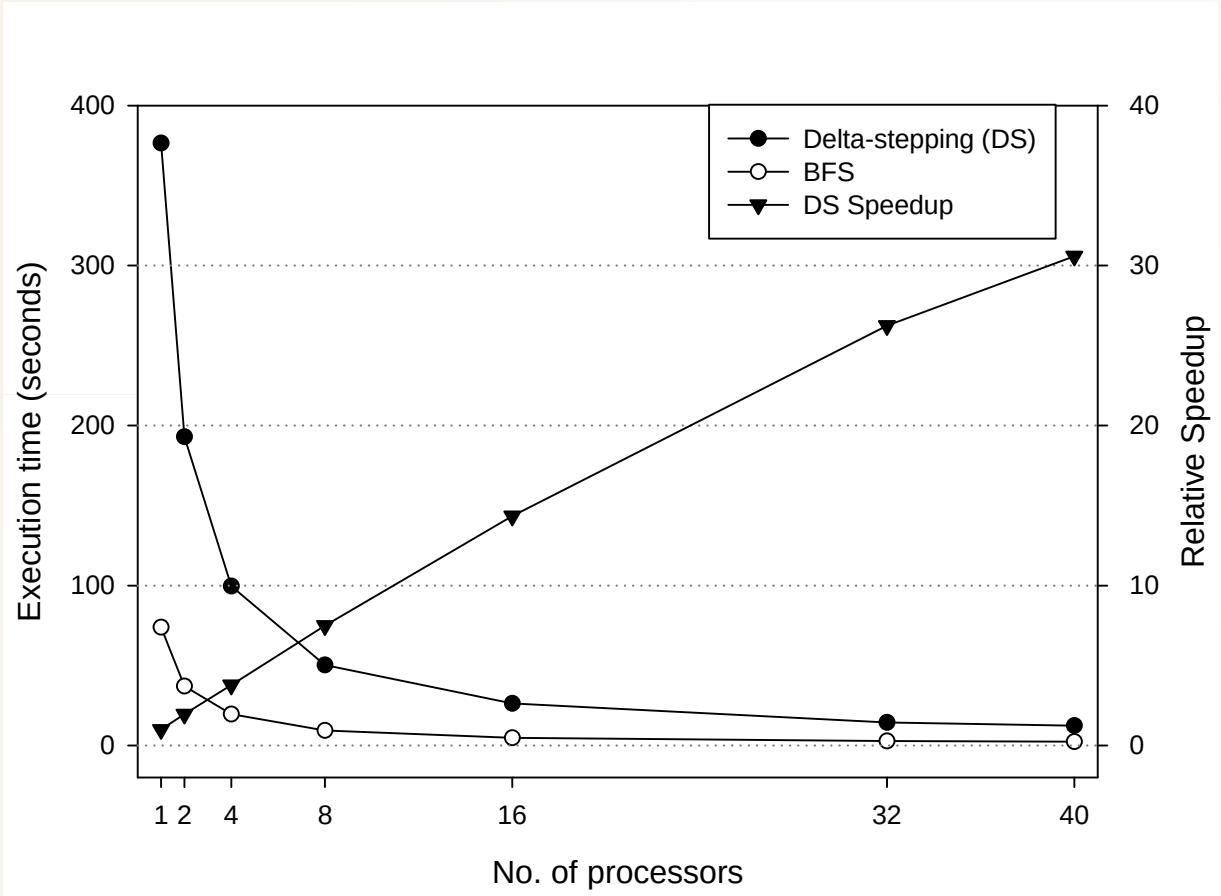
- We integrate these implementations into SNAP, with optimizations for small-world networks



Small-world network optimizations

- Graph representation
  - space-efficient layout
  - different representation for high-degree vertices
- Graph traversal: a level-synchronous approach is suited for low diameter graphs
- Work is assigned to processors with the unbalanced degree distribution in mind
- We try to minimize synchronization overhead in our parallel approaches
- Attention to parallelization granularity

Parallel Performance: BFS and Shortest Paths



synthetic scale-free network of 2^{28} vertices and 2^{30} edges

Parallel performance results on The Cray MTA-2 (1-40 processors)

Georgia Tech College of Computing
Computational Science and Engineering

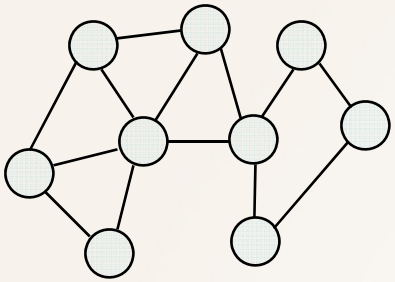
Modularity-maximizing community identification algorithms

Algorithm	Clustering Approach	Algorithmic Technique	Complexity
Newman, Girvan '03	global, divisive	greedy, centrality-based	$O(n^3)$
Clauset et al. '04	global, agglomerative	greedy	$O(n\log^2 n)$
Duch, Arenas '05	global, divisive	extremal optimization	$O(n^2\log^2 n)$
Djidjev '06	global, divisive	greedy, multi-level partitioning	$O(n^2)$
Newman '06	global, divisive	tral	$O(n^2 \log n)$
Brandes '07	agglomerative	dy	$O(n^2 \log n)$
pBD	global, divisive	greedy, centrality-based	$O(n^3)$
pMA	global, agglomerative	greedy	$O(n\log^2 n)$
pLA	local, agglomerative	Greedy, clustering coefficient-based	$O(n\log n)$

Georgia Tech College of Computing
Computational Science and Engineering

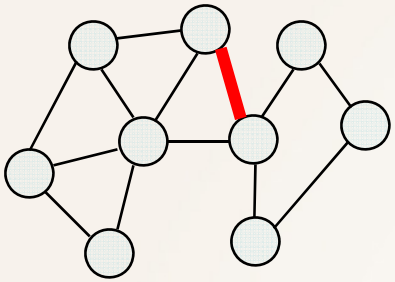
Greedy Divisive Clustering Approach: Illustration

Iteratively remove potential ***inter-community*** edges



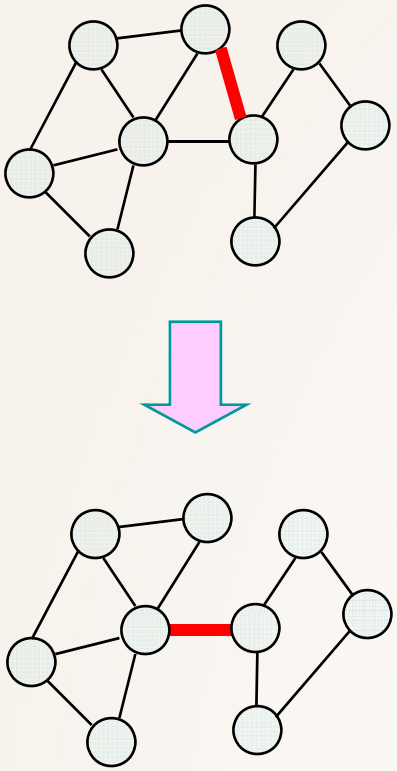
Greedy Divisive Clustering Approach: Illustration

Iteratively remove potential ***inter-community*** edges



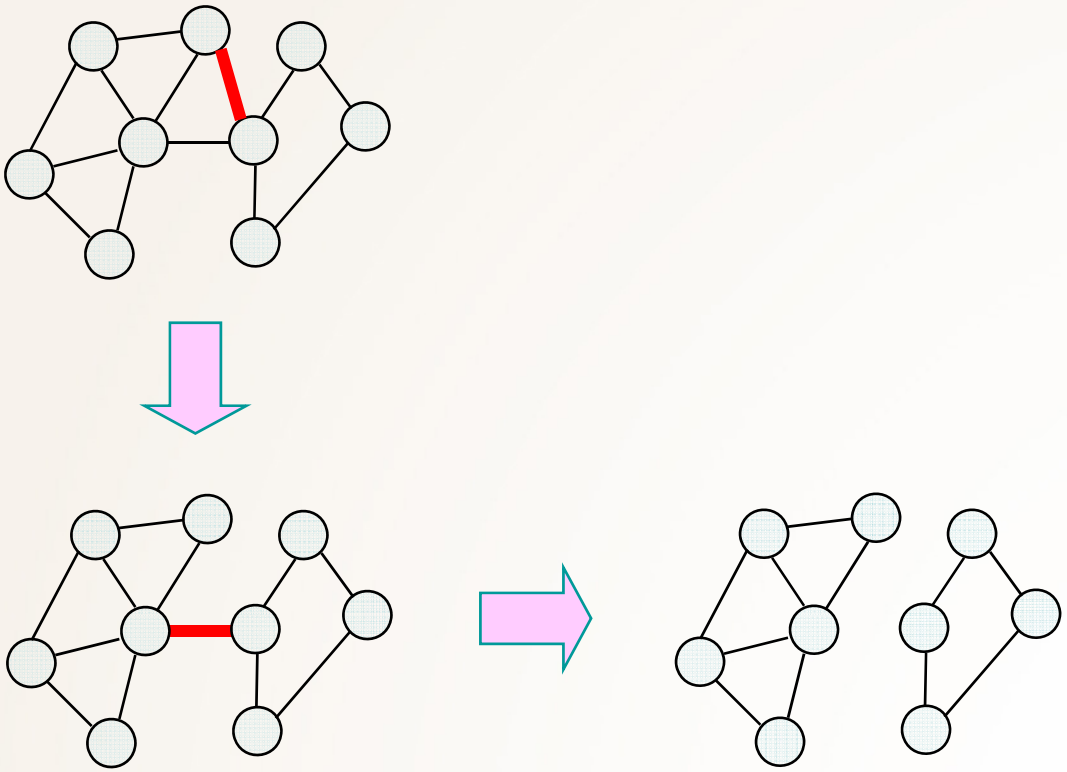
Greedy Divisive Clustering Approach: Illustration

Iteratively remove potential ***inter-community*** edges



Greedy Divisive Clustering Approach: Illustration

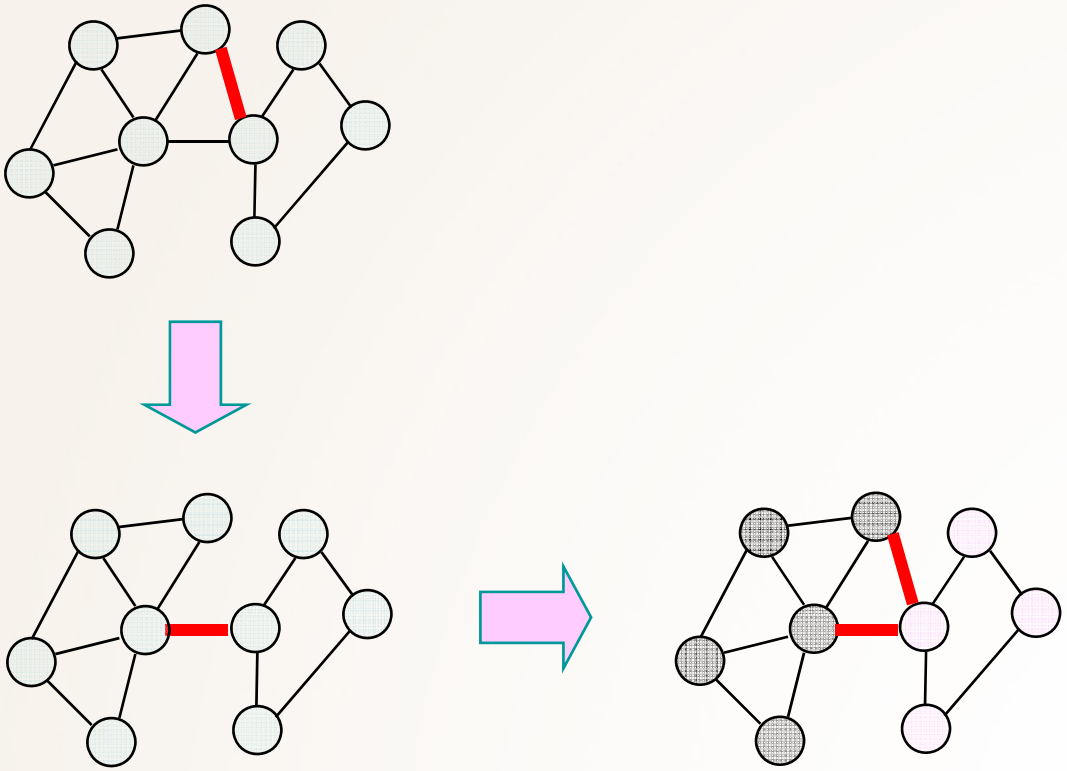
Iteratively remove potential ***inter-community*** edges



Greedy Divisive Clustering Approach: Illustration

Iteratively remove potential ***inter-community*** edges

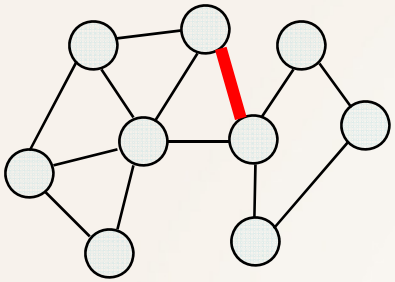
- How do we identify these edges?

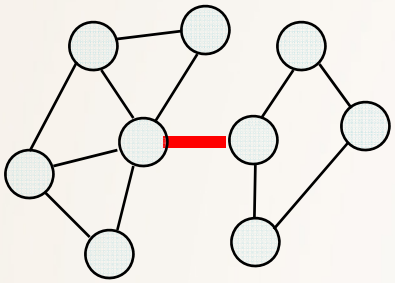


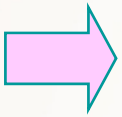
Greedy Divisive Clustering Approach: Illustration

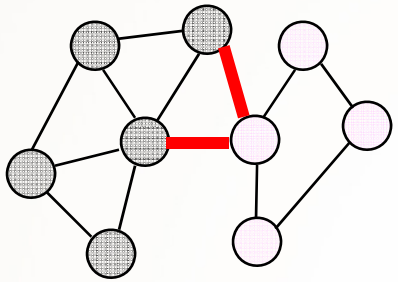
Iteratively remove potential ***inter-community*** edges

- How do we identify these edges?



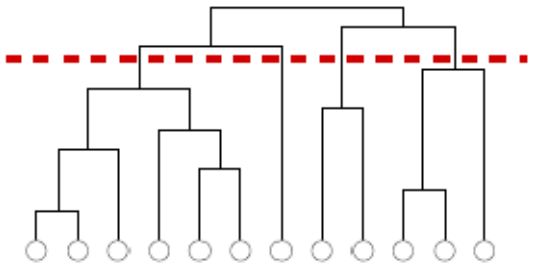






Newman-Girvan '03

- Compute *edge betweenness* scores to determine cut-edges
- Use the modularity metric to quantify community structure

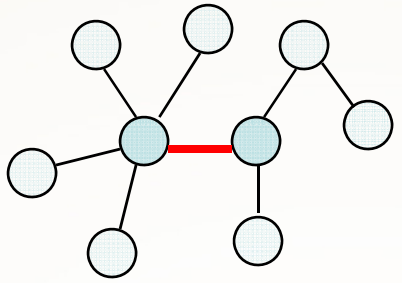


Georgia Tech College of Computing
Computational Science and Engineering

Betweenness Centrality (BC)

- Key metric in social network analysis
  [Freeman '77, Goh '02, Newman '03, Brandes '03]

$$BC(e) = \sum_{s \neq t \in V} \frac{\sigma_{st}(e)}{\sigma_{st}}$$



- σ_{st} -- No. of shortest paths between vertices *s* and *t*
- $\sigma_{st}(v)$ -- No. of shortest paths between vertices *s* and *t* passing through edge *e*

- Exact BC is compute-intensive: O(mn)

Georgia Tech College of Computing
Computational Science and Engineering

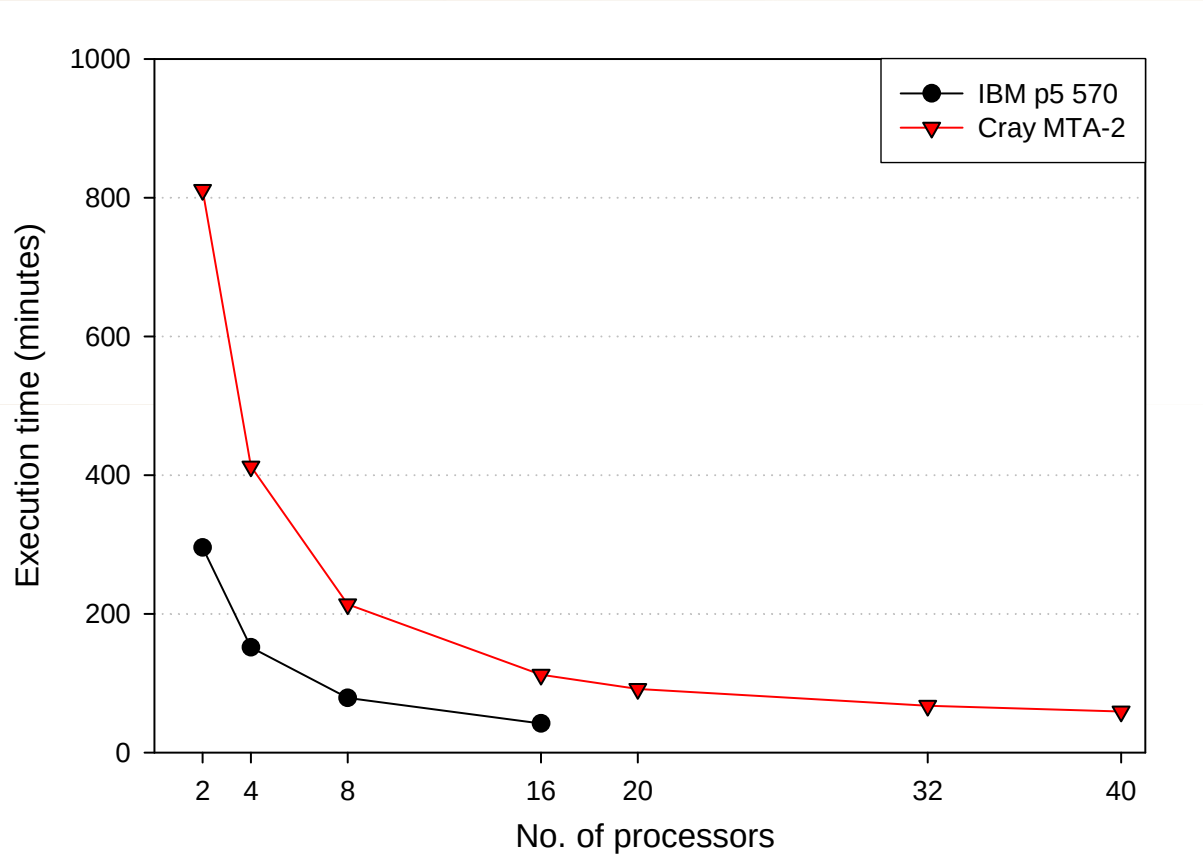


Centrality analysis: previous work

- Design and implementation of *parallel algorithms* for evaluating Betweenness and other Centrality metrics, *optimized for scale-free sparse graphs*
- Capability to solve real-world instances more than three orders of magnitude larger than current SNA packages!
- We analyze BC scores for *several large-scale real datasets*: patent citation networks, movie-actor, and protein-interaction networks

Georgia Tech College of Computing
Computational Science and Engineering

Parallel Performance: Betweenness Centrality



IMDB movie-actor interaction network: 392K vertices and 31.7M edges

Parallel performance results on the Cray MTA-2 and IBM p5 570

Georgia Tech College of Computing
Computational Science and Engineering



Approximate Betweenness-based Divisive Algorithm (pBD)

m Iterations:

1. Calculate ***approximate*** betweenness scores of all the edges in the graph **in parallel**
2. Find the edge with the highest score and remove it from the network
3. Extract connected components of the graph
3. Compute modularity of resulting clustering **in parallel**

- Note: Betweenness scores are recalculated in every iteration
- We need to only determine the edge with the highest betweenness in each iteration
- *Preprocessing routines*: biconnected components, strongly connected components, sparsification


Approximating Betweenness Centrality

[Bader, Kintali, Madduri, Mihail '07]

- Novel approximation algorithm for determining the betweenness of a *specific vertex or edge* in a graph
- *Adaptive in the number of samples* (graph traversals) – the work done varies with the information obtained in each sample
- We prove high-probability bounds on the error with a fixed number of samples
- In practice, we observe that high-centrality entities can be estimated with less than 20% error, by sampling just 5% of the vertices
- 20X speedup over exact betweenness centrality



Approximate Betweenness-based Divisive Algorithm (pBD)

- Identify highest centrality edge using the approximate betweenness algorithm
- Run fast auxiliary kernels (biconnected components) to determine critical edges in graph
- Once the graph is broken into a sizeable number of components, compute centrality score of each component in parallel

Georgia Tech College of Computing
Computational Science and Engineering



Greedy Agglomerative Clustering Algorithm (pMA)

- Start with *n* singleton communities
- Iteratively merge the pair of communities that result in the greatest increase in modularity
  - Maintain a priority queue of possible community merges
- Update overall modularity score after merge

- At each iteration, the community-pair costs are computed in parallel. The community modularity update step is also parallelized.

Georgia Tech College of Computing
Computational Science and Engineering


Greedy Local Aggregation Algorithm (pLA)

- Previous approaches to maximizing modularity rely on global heuristics (an vertex/edge-ranking criterion) *on every iteration*
- pLA: we order vertices only once, and then use a local heuristic (clustering coefficient) for forming communities
- Start from randomly chosen vertices in the graph, inspect their neighborhoods, and iteratively build clusters
- Discard community if edge cut is above a threshold

Georgia Tech College of Computing
Computational Science and Engineering

Modularity comparison with other approaches

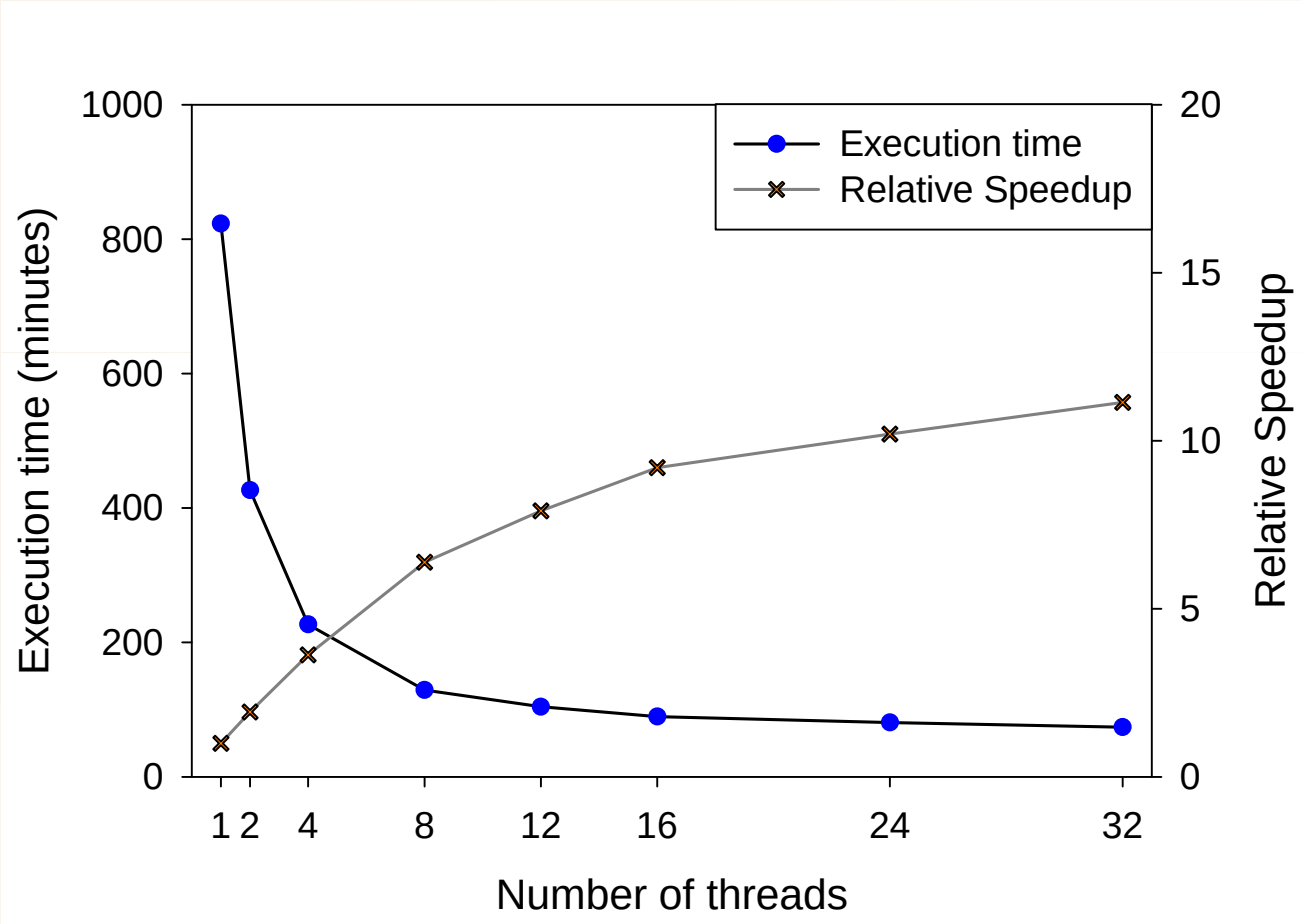
		Modularity Q				
Network	n	GN	pBD	pMA	pLA	Best known
Karate	34	0.401	0.397	0.381	0.397	0.431
Political books	105	0.509	0.502	0.498	0.487	0.527
Jazz musicians	198	0.405	0.405	0.439	0.398	0.445
Metabolic	453	0.403	0.402	0.402	0.402	0.435
E-mail	1,133	0.532	0.547	0.494	0.487	0.574
Key signing	10,680	0.816	0.846	0.733	0.794	0.855


Experimental Study

- Experiments run on the Sun Fire T2000 (8-cores, 4 threads per core, UltraSparc T1 processor)
- Large-scale networks used:

Label	Network	n	m	Type
PPI	human protein interaction network	8,503	32,191	undirected
Citations	Citation network from KDD Cup 2003	27,400	352,504	directed
DBLP	CS publication coauthorship network	310,138	1,024,262	undirected
NDwww	web-crawl (nd.edu)	325,729	1,090,107	directed
Actor	IMDB movie-actor network	392,400	31,788,592	undirected
RMAT-SF	synthetic small-world network	400,000	1,600,000	undirected

pBD Performance (Sun Fire T2000)

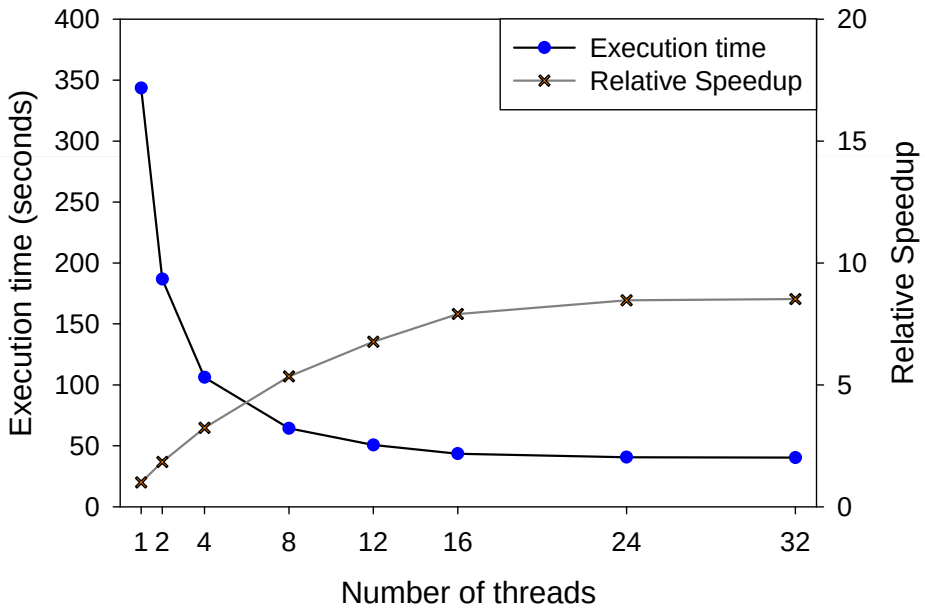


RMAT-SF: 400K vertices and 1.6M edges

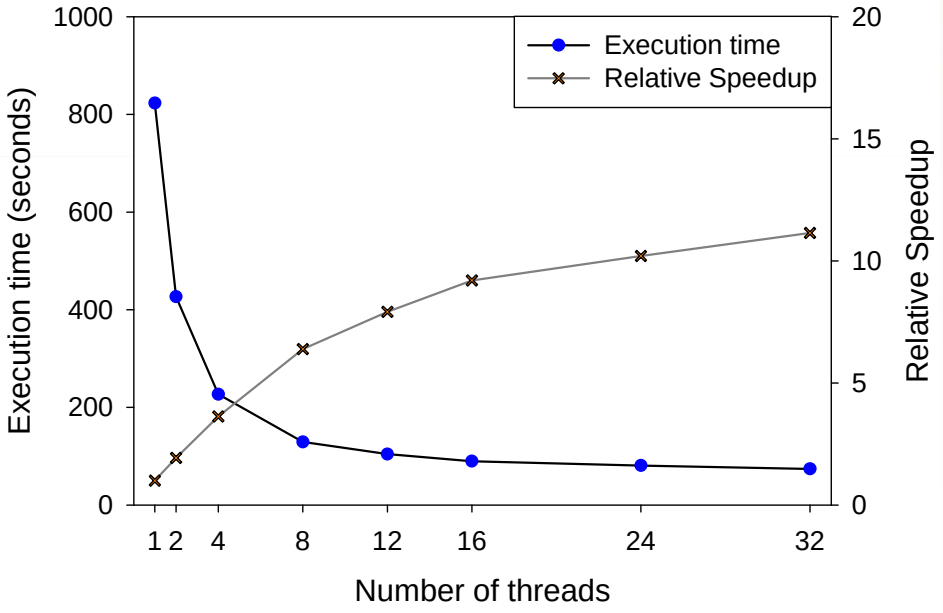
Georgia Tech College of Computing
Computational Science and Engineering

pMA and pLA Performance (Sun Fire T2000)

pMA



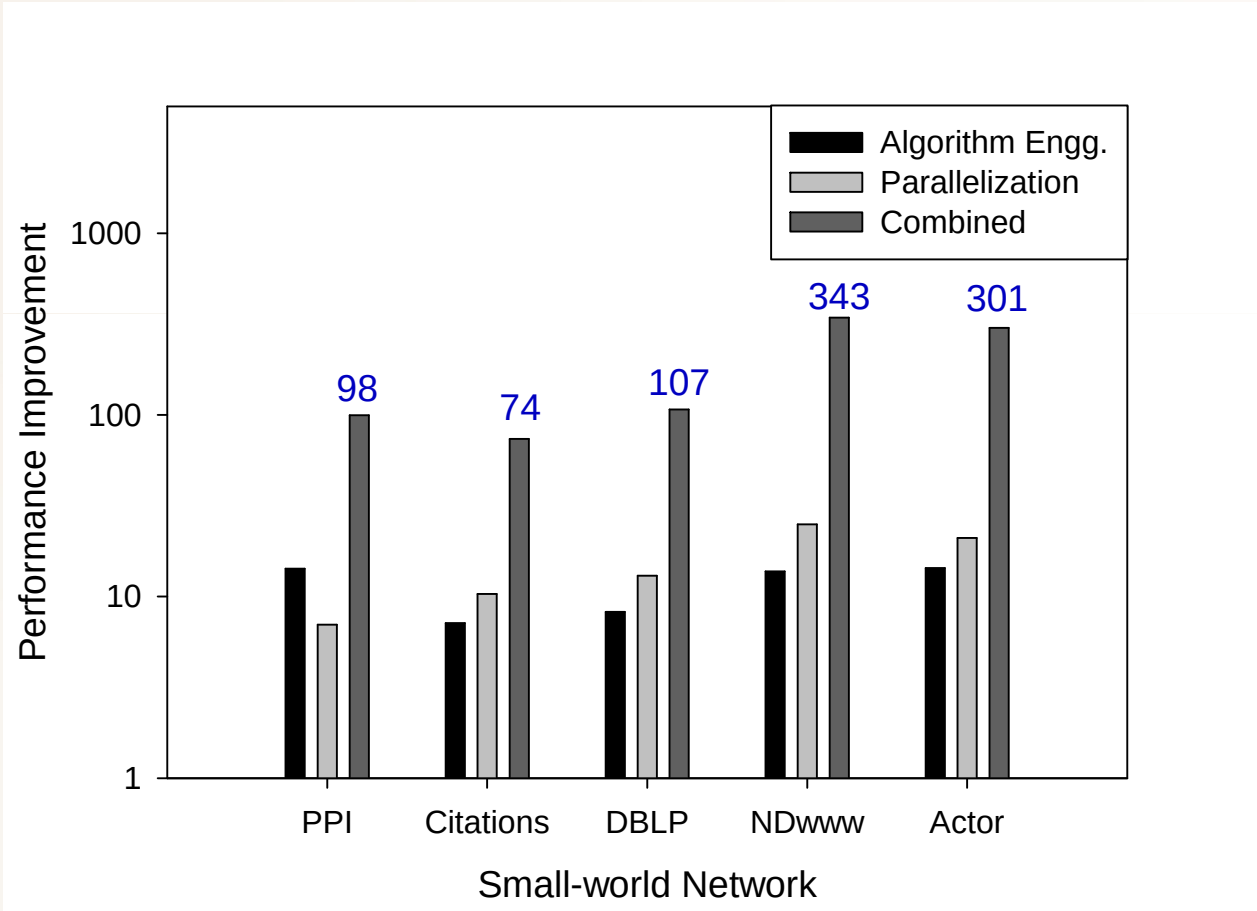
pLA



RMAT-SF: 400K vertices and 1.6M edges

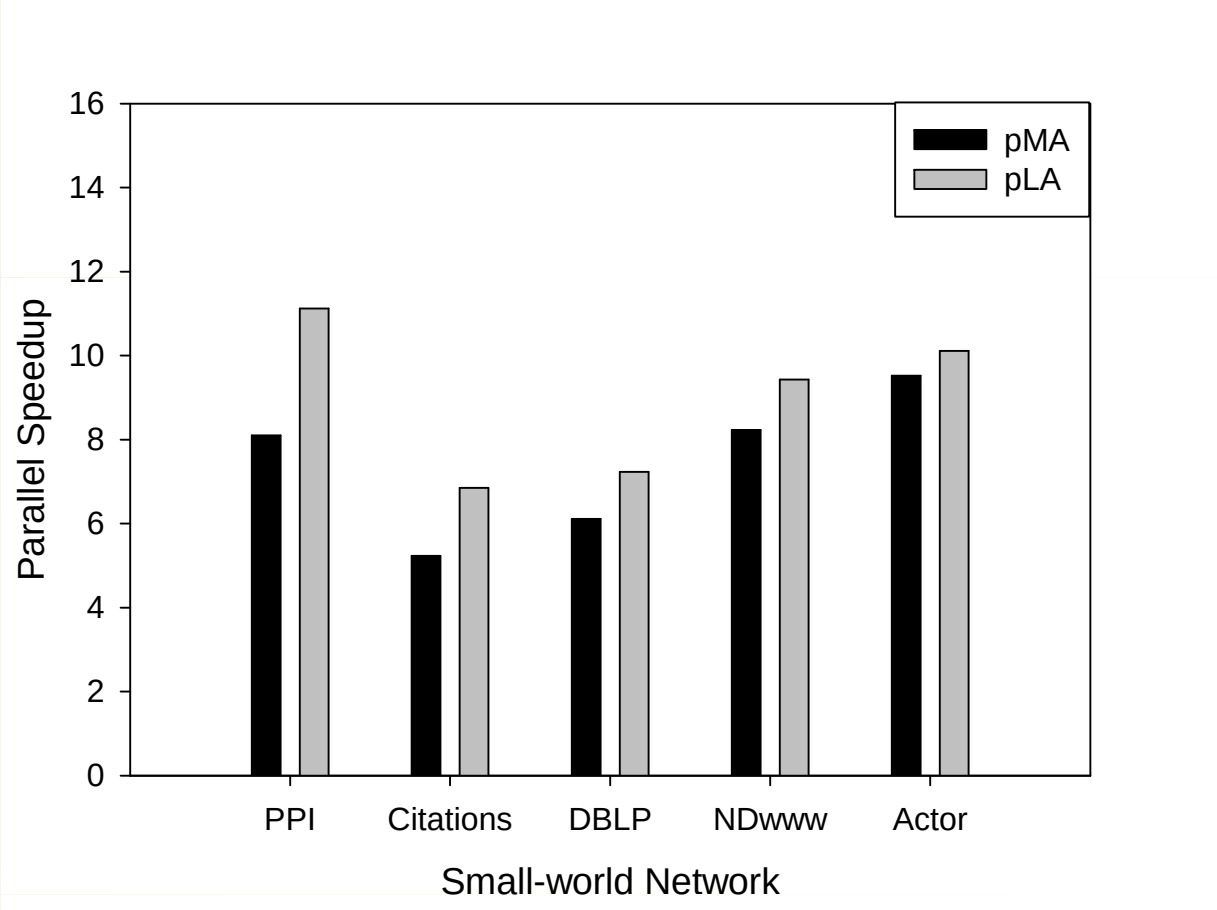
Georgia Tech College of Computing
Computational Science and Engineering

pBD speedup relative to GN (Parallel speedup on Sun Fire T2000)



Georgia Tech College of Computing
Computational Science and Engineering

pLA and pMA Parallel Speedup on large-scale graph instances (Sun Fire T2000)



Georgia Tech College of Computing
Computational Science and Engineering


Observations

- pBD is 10-30x faster than GN with very little loss in quality
- Multithreaded implementations of all three algorithms scale well on the Sun Fire T2000
- Speedup achieved from preprocessing kernels depends on the graph topology
  - NDwww, Citations: significant improvement


Conclusions

- We present three new parallel approaches for community identification in small-world networks

- SNAP: an open-source parallel framework for large-scale network analysis
  - http://www.cc.gatech.edu/~kamesh/SNAP
  - http://www.graphanalysis.org

Georgia Tech College of Computing
Computational Science and Engineering

Acknowledgment of Support