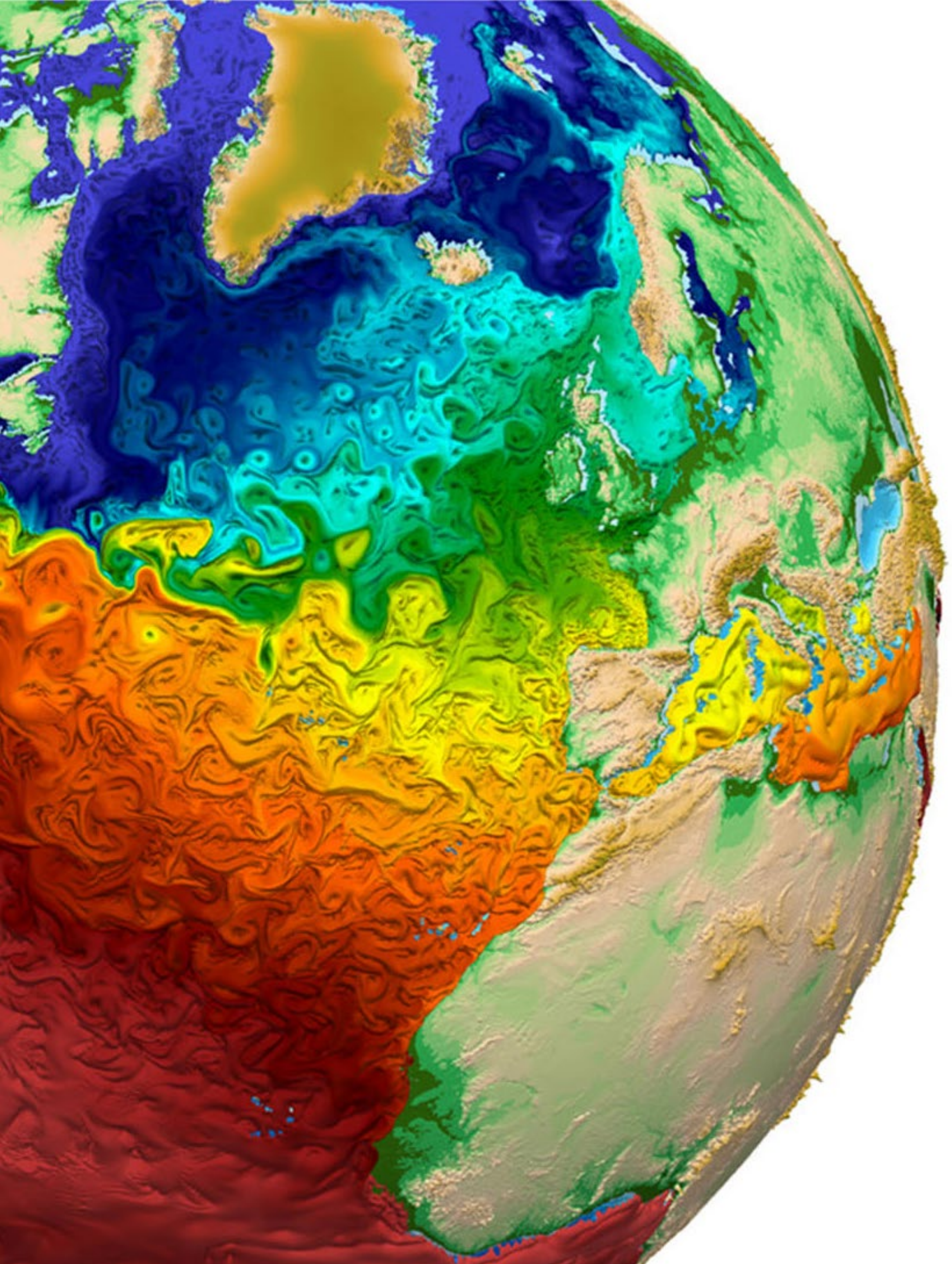


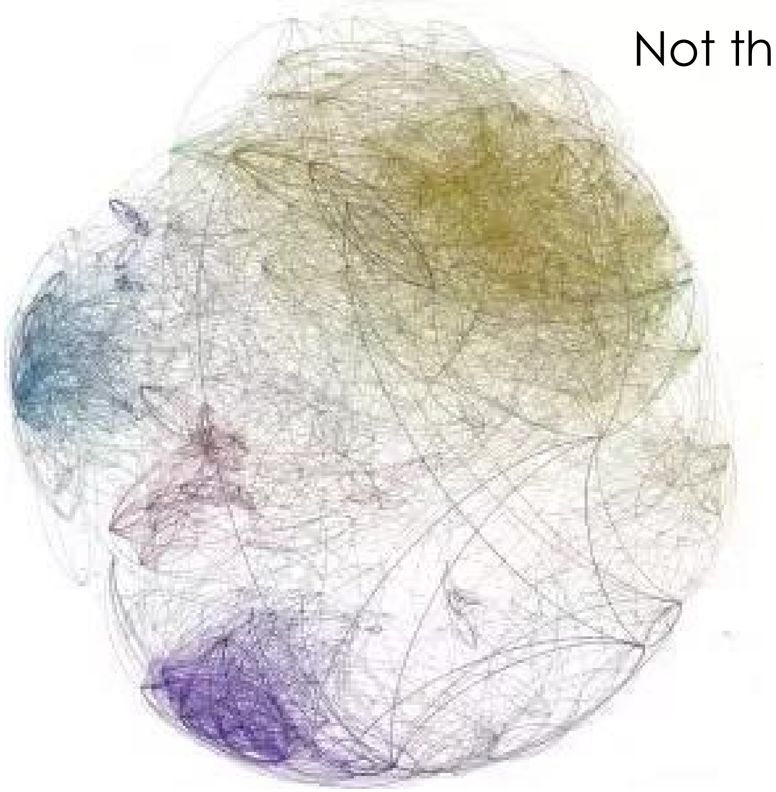
Large Graph Visualization Edge Filtering

Scientific Visualization

Professor Eric Shaffer

ILLINOIS

Large Networks Are Problematic



Not the webgraph

- 2012 webgraph:
- 3.5 billion pages 128 billion links
- Probably don't have enough pixels
- Even if we did, probably don't have enough cognitive capacity

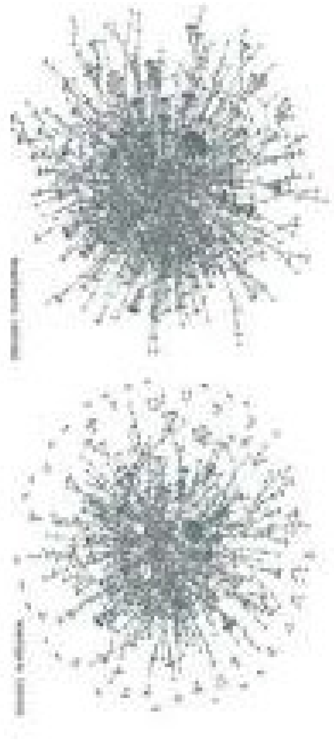
ILLINOIS

Graph Preprocessing

Idea: We can visualize smaller graphs

Let's make the big graph into a small graph

…try to keep the most important parts



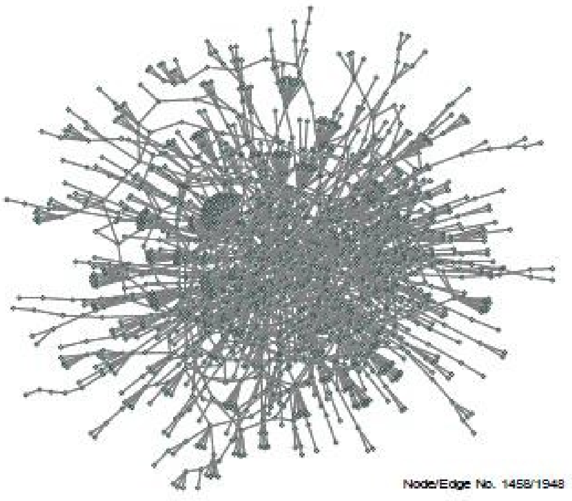
Two approaches:

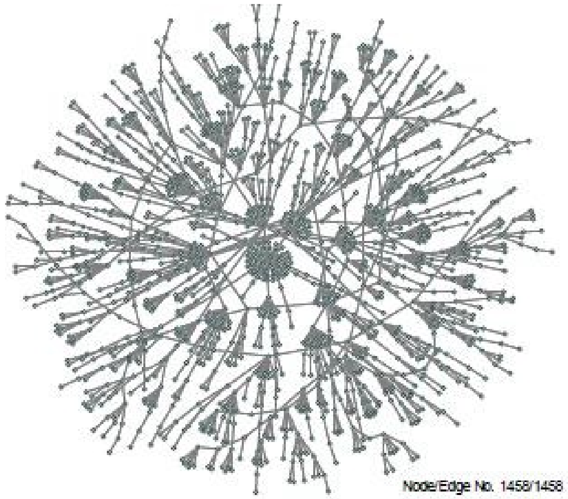
- Graph filtering: remove unimportant parts
- Graph aggregation: merge similar graph elements together

Graph Filtering

JIA Y., HOBEROCK J., GARLAND M., HART J.:
On the visualization of social and other scale-free networks.
IEEE Transactions on Visualization and Computer Graphics (2008)

- Removes edges in order of increasing betweeness centrality
- Preserves connectivity
- Preserves graph features (e.g. cliques)





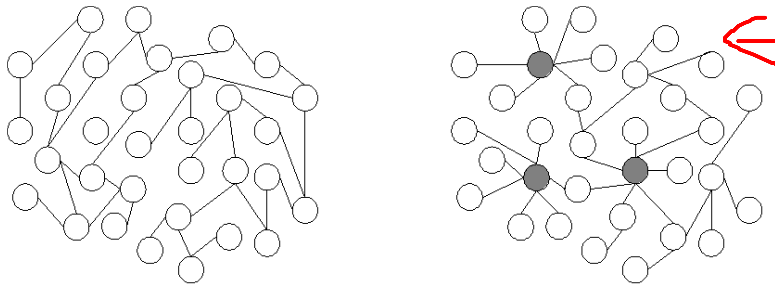
ILLINOIS

Scale-Free Networks

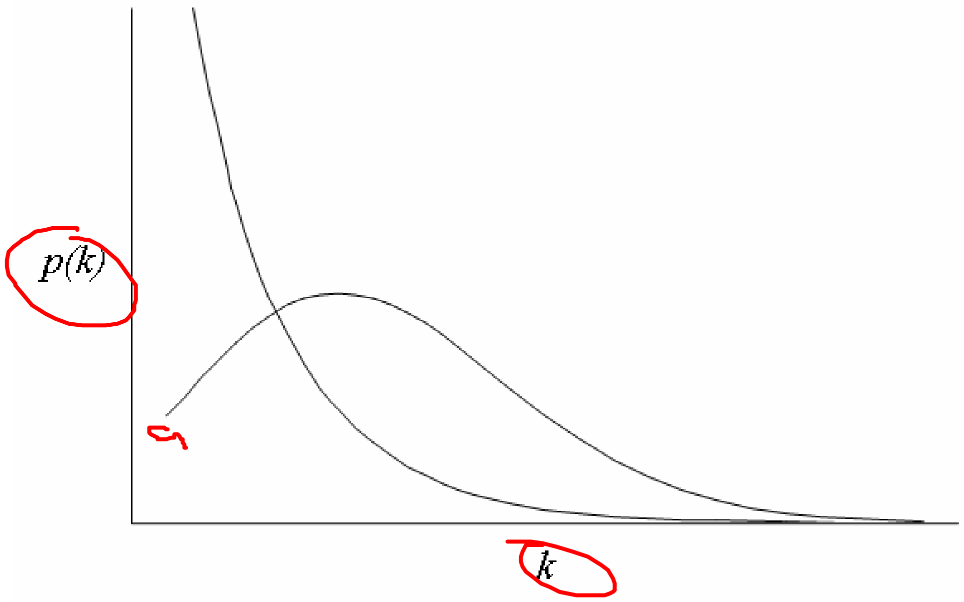
Real-world networks are often claimed to be scale free

Meaning that the fraction of nodes with degree k follows a power law $k^{-\alpha}$

- A few nodes are hubs with many incident edges
- Many nodes have few incident edges
- Social networks were thought to be scale-free



(a) Random network (b) Scale-free network



Current Research in Networks

Article | Open Access | Published: 04 March 2019

Scale-free networks are rare

Anna D. Broido & Aaron Clauset

Nature Communications **10**, Article number: 1017 (2019) | Cite this article

- Statistical analysis has shown that few empirical data sets are truly scale-free
- Social networks currently thought to be weakly scale-free…

Centrality-based graph filtering can be applied to non-scale-free networks…just won't works as well

Betweeness Centrality for Vertices

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

σ_{st} is the total number of shortest paths from node s to node t

$\sigma_{st}(v)$ is the number of those paths that pass through node v

Betweeness Centrality for Edges

$$g(e) = \sum_{s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}}$$

σ_{st} is the total number of shortest paths from node s to node t

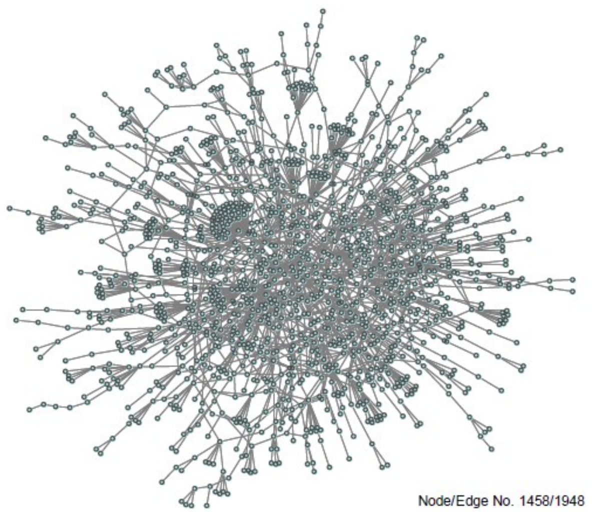
$\sigma_{st}(e)$ is the number of those paths that pass through edge e

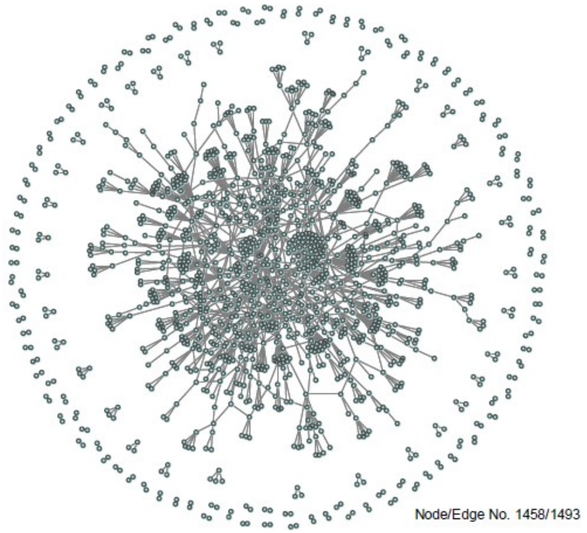
Betweeness Centrality

Betweeness Centrality (BC) ranks edges (or vertices)

- How often they appear on shortest paths
- High BC → important communication tunnels
- Low BC → less important
- Remove low BC edges
- Keeps “back bone” of the graph

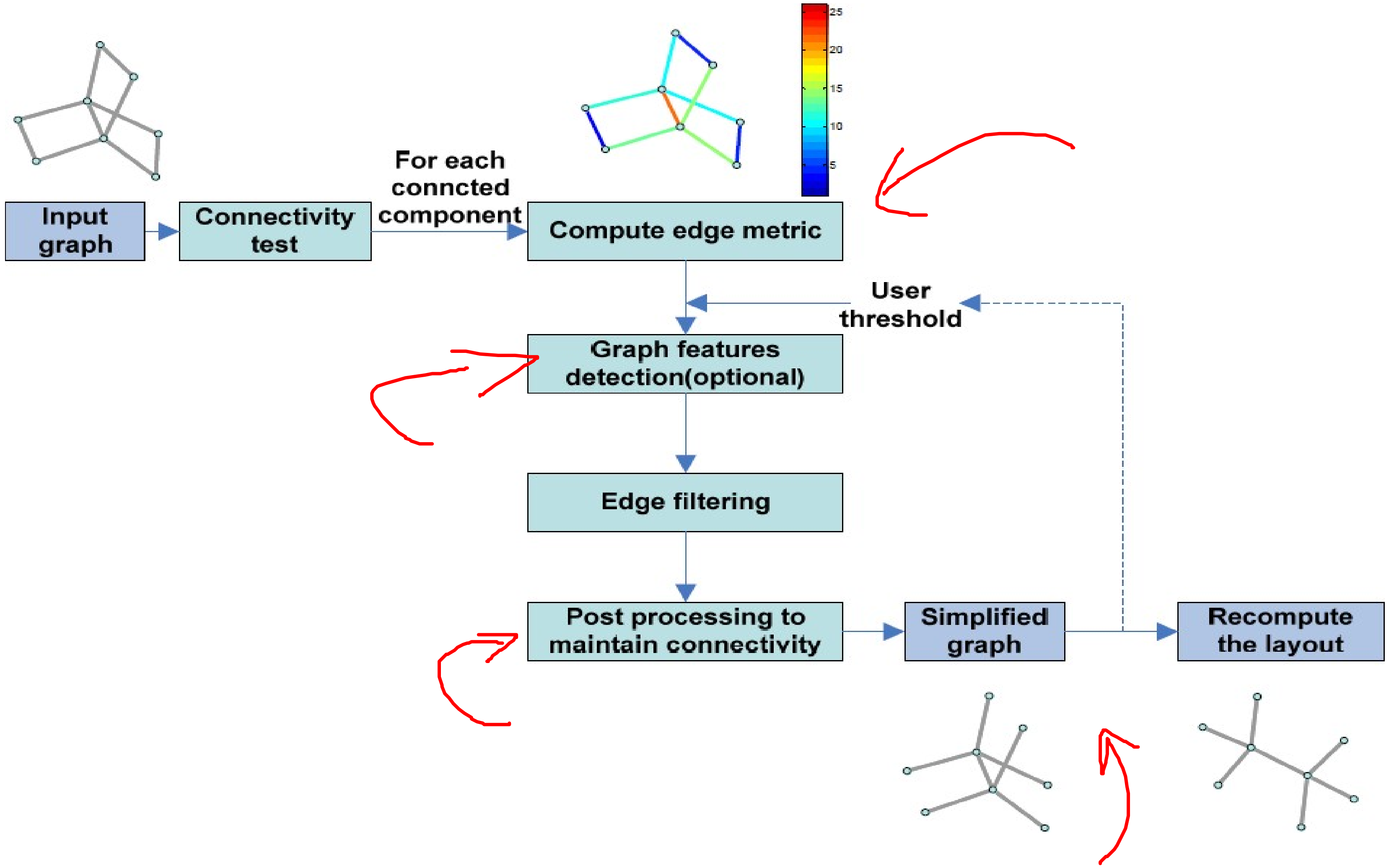
Simple Edge Filtering is Insufficient





Need to maintain connectivity…possibly other important features

Workflow



Input graph
Connectivity test
For each conncted component
Compute edge metric
25
20
15
10
5
User threshold
Graph features detection(optional)
Edge filtering
Post processing to maintain connectivity
Simplified graph
Recompute the layout

Betweeness Centrality is Expensive

Graph $G = (V, E)$, $|V| = n$, $|E| = m$

- Betweenness centrality [Freeman 1977]
- Relies on computing All-Pairs Shortest Paths
- Complexity $O(m*n)$ for unweighted graph [Brandes 01]

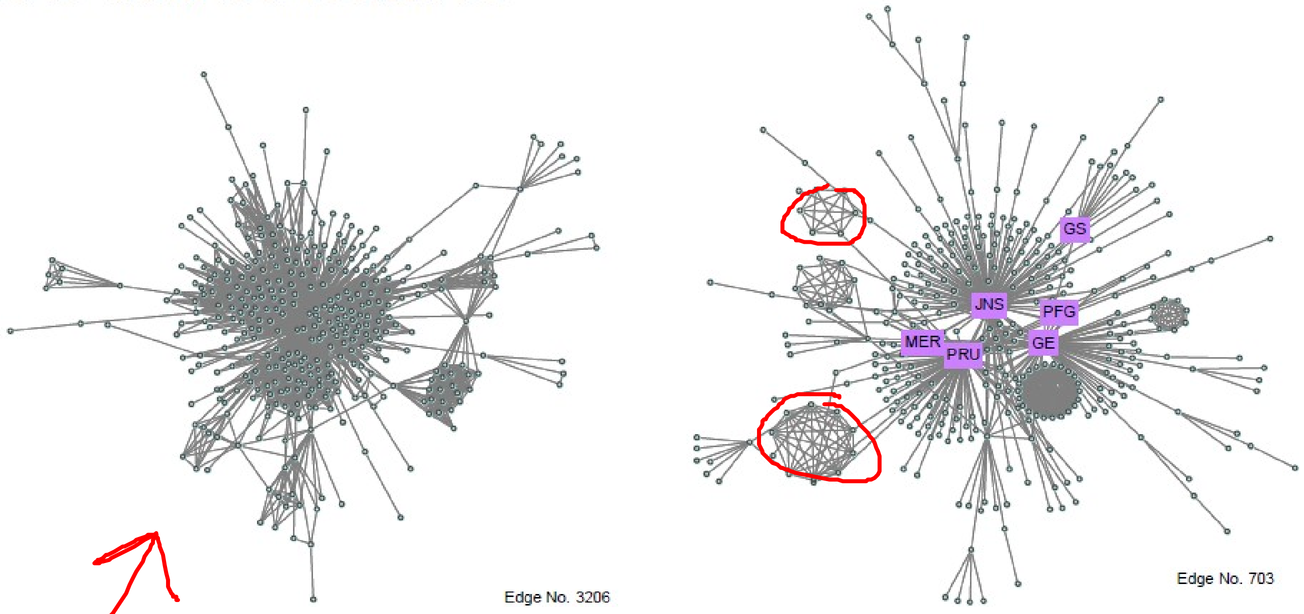
For huge graphs

- Approximated with random sampling [Jacob et al. 05]
  - $O((m+n)*log(n))$ with $C*log(n)$ samples where C is a constant

- For our edge filtering purpose
  - Only relative orders of BC are needed
  - Select $C*log(n)$ highest degree hub nodes

Graph Feature Detection

- Graph features
  - Cliques
    - NP-Complete problem
    - Fast approximation *O(m*n)* [Chiricota et al. 03]
- User defined features



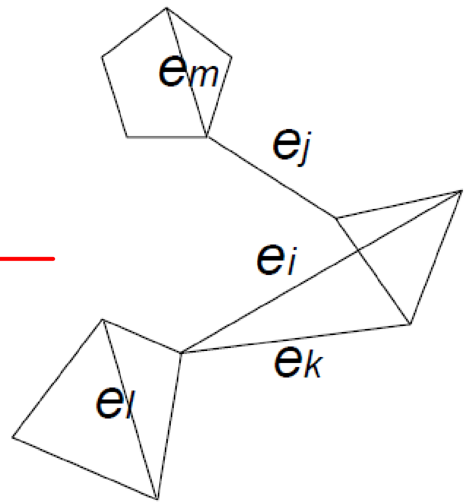
Edge Filtering

Edges	BC Metric
…	…
e_h	1.3
e_i	1.1
e_j	1.2
e_k	1.15
e_l	1.21
e_m	1.09
…	…

Sort

Edges	BC Metric
e_m	1.09
e_i	1.1
e_k	1.15
e_j	1.2
e_l	1.21
e_h	1.3
…	…

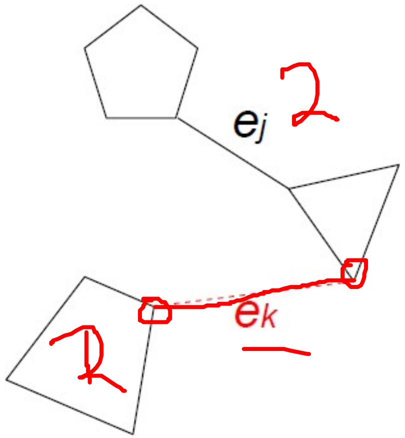
Threshold $t = 1.25$



Recover Connectivity

e_l	1.21
e_j	1.2
e_k	1.15
e_i	1.1
e_m	1.09
Removed Edges	BC Metric

stack



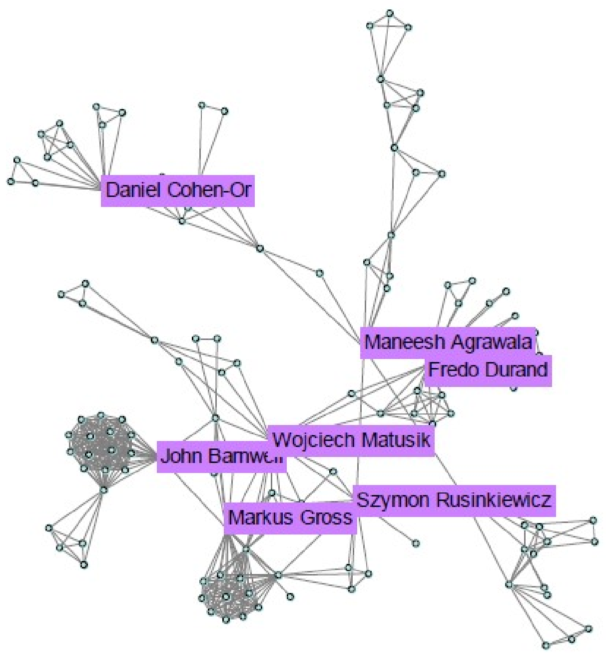
- Compute connected components of graph and label vertices by component
- Iterate through the removed edges in reverse order of removal.
- If an edge links a pair of nodes belonging to different components

  → restore that edge and unify components and labels
- The iteration continues until graph contains a single component

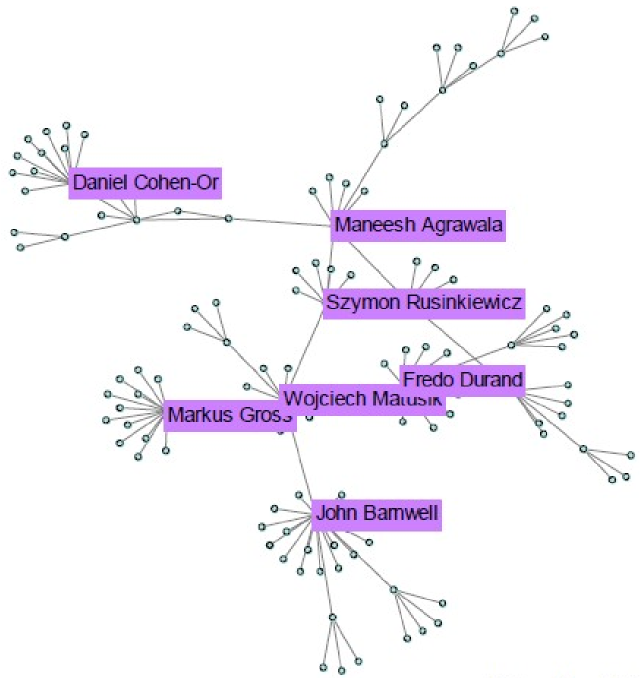
Recompute the Layout

After filtering, apply a force-directed layout

Fewer edges…should be more efficient and less visually cluttered

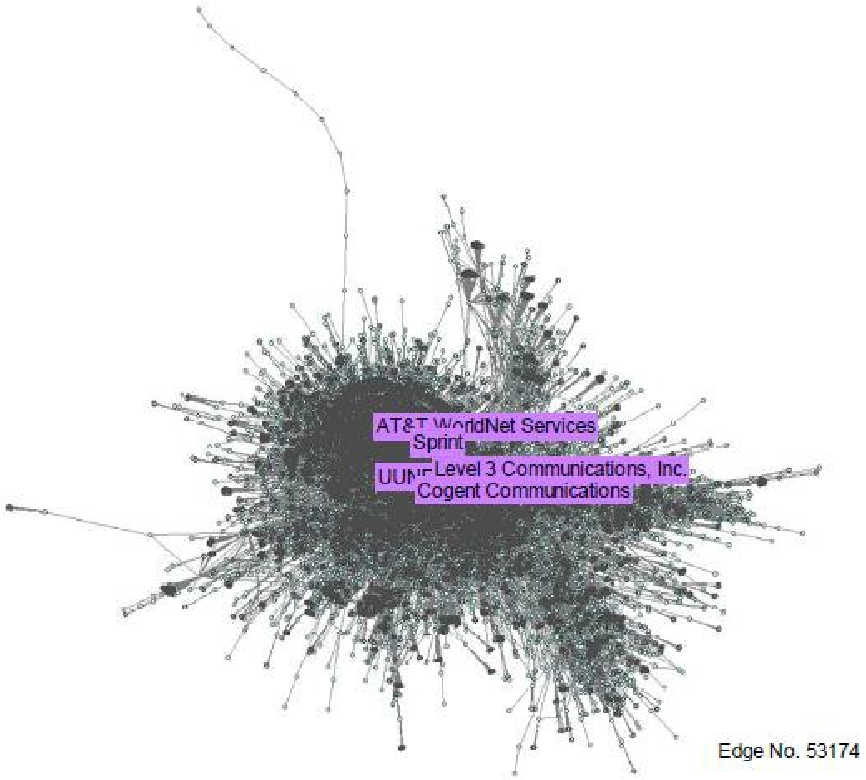


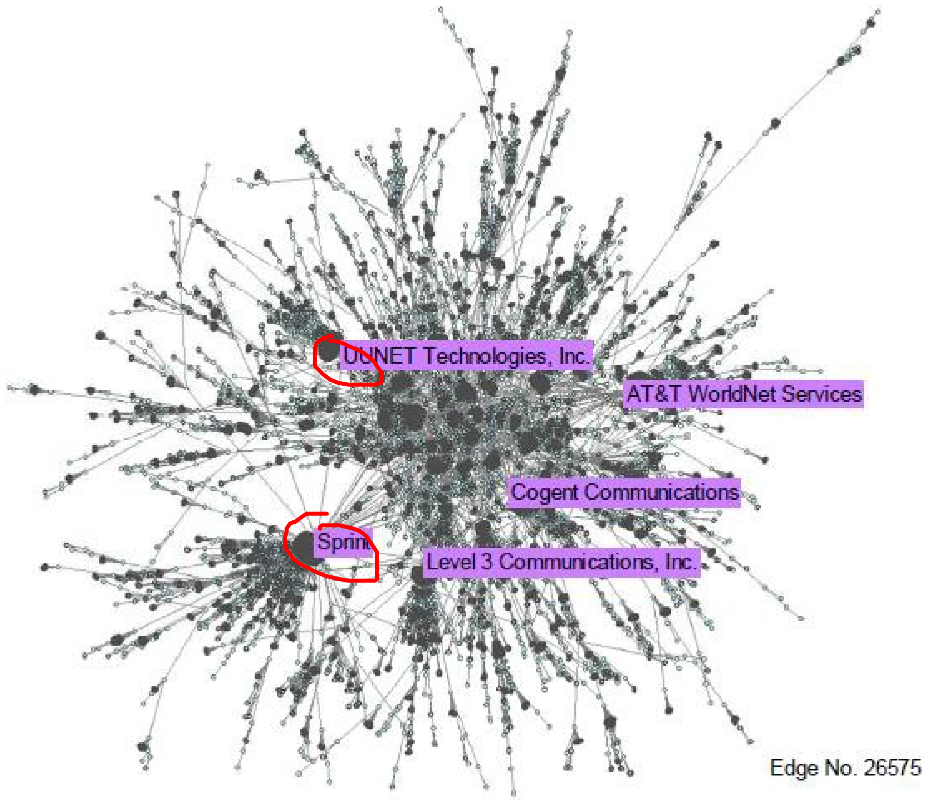
Edge No. 441



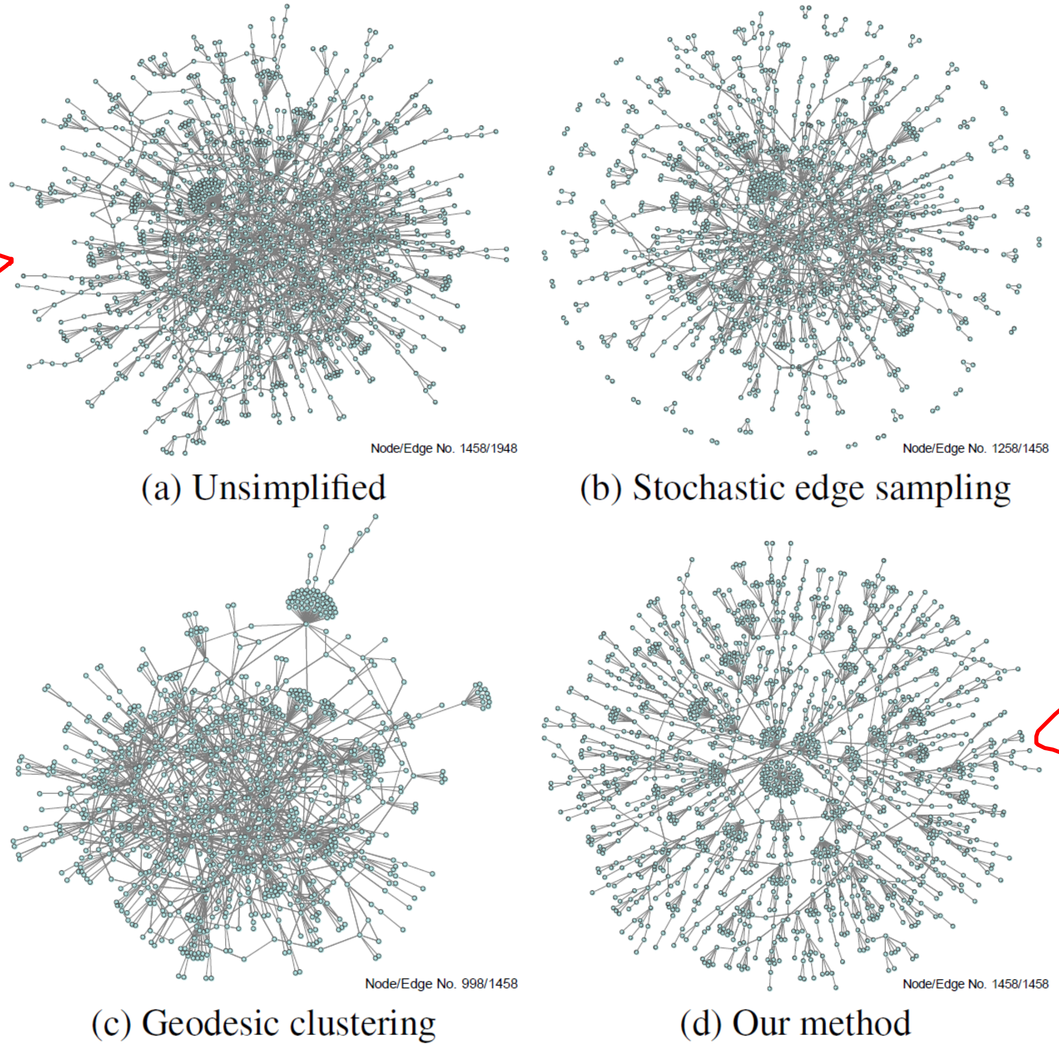
Edge No. 129

Fixing the Hairball....





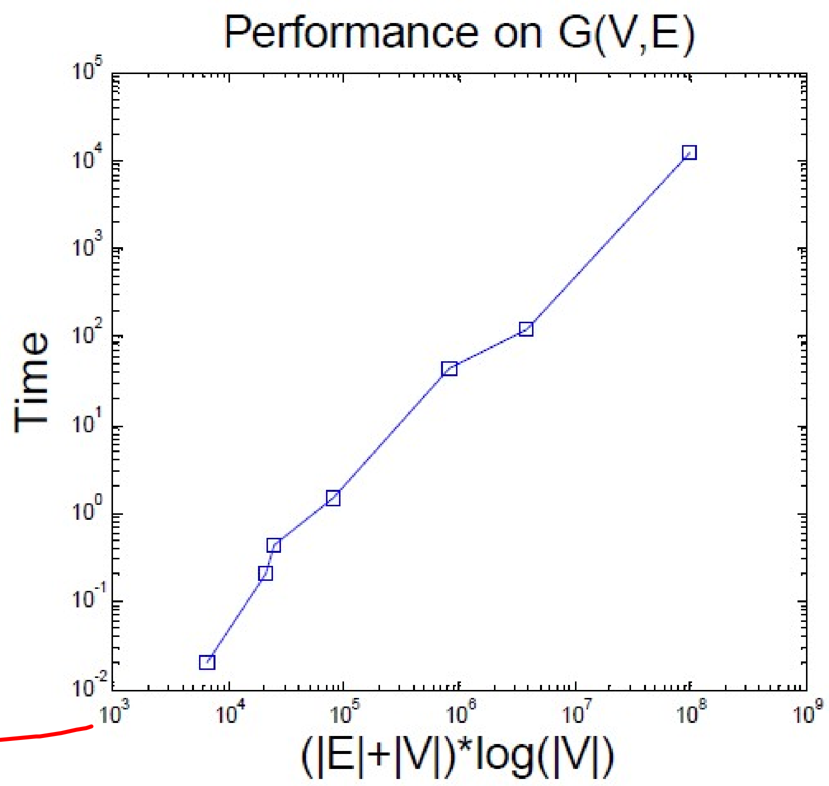
Comparison



ILLINOIS

Performance

Graph	Nodes	Edges	Timing
siggraph07	328	773	0.02s
sp500-038	365	3206	0.20s
bo	1458	1948	0.44s
cg_web	2269	8131	1.50s
as-rel.071008	26242	53174	43.66s
hep-th	27400	352021	120.72s
flickr	820878	6625280	12442.70s



Limitations

- Doesn’t work well for non-power law graphs
  - Including planar graphs
  - Clustering may be a better choice than filtering
- Obviously doesn’t show entire data set