Visualization of Social and other Scale-Free Networks

Yuntao Jia, Jared Hoberock, Michael Garland, John C. Hart

Technical Report UIUCDCS-R-2008-2955

April 2008

Abstract

This paper proposes novel methods for visualizing specifically the large power-law graphs that arise in sociology and the sciences. In such cases a large portion of edges can be shown to be less important and removed while preserving component connectedness and other features (e.g. cliques) to more clearly reveal the network's underlying connection pathways. This simplification approach deterministically filters (instead of clustering) the graph to retain important node and edge semantics, and works both automatically and interactively. The improved graph filtering and layout is combined with a novel computer graphics anisotropic shading of the dense crisscrossing array of edges to yield a full social network and scale-free graph visualization system. Both quantitative analysis and visual results demonstrate the effectiveness of this approach.

1 Introduction

Social and other scale-free networks are graphs with few nodes of higher degrees and many of lower degrees, such that the number of nodes of degree k follows a power-law distribution [4]. This distribution is ubiquitous, natural and commonly found in the relationships studied in sociology, networking, biology and physics.

These graphs are rarely planar, and even the best layout methods yield a space-filling jumble of edge crossings for even medium-scale graphs. For example, Fig. 1(a) displays the scale-free network of 1,948 interactions between 1,458 yeast proteins.

Such large graphs can be more effectively visualized in a simplified form so long as the simplification preserves the important structures and features of the original. Sec. 2 reviews a variety of graph simplification methods.

Node clustering simplifies graphs by merging neighboring nodes, which when repeated organizes the graph into a hierarchy [8]. Clustering works well on planar graphs. When applied to non-planar graphs, it can actually increase edge density which makes the layout less flexible and the display more jumbled with more edge crossings. For example, the geodesic clustering used for the visualization in Fig. 1(c) increases the edge density from 1.34 to 1.46 edges/node and the effect worsens with the increased clustering of larger graphs. Furthermore, the merged nodes and edges created by node clustering lose their original semantics.

Filtering methods retain edge and node semantics by ensuring the simplified graph is a subgraph of the original. Stochastic filtering approaches statistically sample the graph, scaling well and preserving the expectation of various graph characteristics. But for the visualization of scale-free networks where most nodes are of least degree, stochastic filtering can destroy connectivity and other features, as it did in Fig. 1(b).

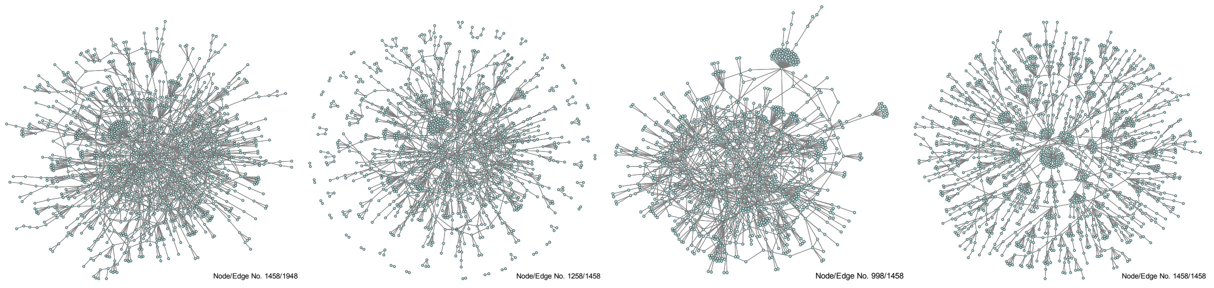
Deterministic filtering methods remove edges based on a metric defined on graph elements. One commonly used metric is "betweenness centrality," [13] which indicates how often a node lies on the shortest (and presumably most used) communication path(s) between other nodes

$$BC(v) = \sum_{u \neq v \neq w \in V} \sigma_{u,w}(v)/\sigma_{u,w} \tag{1}$$

where $\sigma_{u,w}$ counts the number of shortest paths between u and w, and $\sigma_{u,w}(v)$ counts only the ones containing v.

Girvan and Newman [17, 26] compute the betweenness centrality of edges instead of nodes, and removed the highest BC edges from a graph to isolate and cluster its subnetworks. But for visualization, scale-free networks do not cluster well because most of their nodes are minimally connected. For the visualization of scale-free networks, we instead remove the lowest BC edges from a graph, leaving a skeletal substructure of communication pathways. Our filtered simplification in Fig. 1(d) removes 50% of the edges while retaining 80% of its total betweenness centrality, as a measure of its preservation of communication pathways.

This development leads to a novel deterministic filtering approach that improves the simplification, layout and visualization of scale-free networks. This method filters a graph by removing edges in order of increasing betweenness centrality. We constrain this filter to preserve connectivity and other features (e.g. cliques) by marking feature edges in a graph preprocessing pass, and keeping an edge if it is marked or if



(a) Unsimplified (b) Stochastic edge sampling (c) Geodesic clustering (d) Our method

Figure 1: Visualization of the protein interaction graph "bo", laid out with GEM [14], after different simplification methods.

its removal disconnects a connected component of the original graph. The resulting simplified graph thus avoids the distraction of edges seldom utilitized in the propagation of information across the network, while retaining the connectivity and other pre-identified features of the original. Furthermore, removing edges improves the flexibility, convergence and quality of node layout algorithms. Fig. 2 illustrates this approach as detailed in Sec. 3.

The betweenness centrality edge metric relies on an expensive all-pairs shortest path computation. We discovered that on scale-free networks, betweenness centrality can be accurately approximated by computing shortest paths only from hubs, of which there are only $O(\log n)$ according to the power-law distribution, as described in Sec. 4, which also compares other approximations to establish the novelty of our hub-subset BC. We believe this approach is novel, as a recent survey of BC variations [7] includes choosing a random subset of shortest path endpoints but not specifically targeting the hubs.

Because edge-filtered graphs can still be large and visually complex, our visualization method also incorporates rendering techniques suited to dense fibrous materials to aid in understanding. The anisotropic shading of thin threadlike materials is a familiar cue of fibrous directionality. Shading a scale-free network's dense mass of edges in this manner accentuates hubs which appear like the poles of wound-thread holiday ornaments, as shown in Sec. 5.

The results in Sec. 6 demonstrate the effectiveness of this approach at simplifying and displaying scale-free networks, followed by conclusions in Section 7.

2 Previous Work

Recent previous work relevant to the simplification and visualization of power-law graphs is briefly summarized below.

Graph Layout. Force-directed approaches are easy to implement and work well for most graphs [11, 12, 15, 21]. Recent, more powerful layout methods have been proposed, including GEM [14], GRIP [16], ACE [22], FM3 [18] and Topolayout [1], but work better on near planar graphs. None of these work well for large power-law graph visualization because of the dense edges crossings they produce, which prompted our investigation into (edge) simplification. Our contribution makes these algorithms practical for scale-free graphs by providing a simpler domain for layout. In implementation, our results are laid out using either force-directed FR method [15] or GEM [14].

Node Clustering. Brandes et al. [8] survey a variety of clustering approaches. Wu et al. [30] hierarchically clustered nodes by their shortest-path distance from hub nodes (chosen by min BC or max degree) for data mining and visualizing power-law graphs. Kumar and Garland [23] used clustering to stratify a graph into different layers for independent, faster layout and overlayed for interactive visualization.

Stochastic Filtering. Rafiei and Curial [27] used stochastic and focus-based filtering schemes to simplify large graphs for visualization. Leskovec and Faloutsos [25] analyzed these for a variety of graph properties using random node/edge selection and random walks.

Deterministic Filtering. As mentioned in the introduction, Girvan and Newman [17, 26] removed high-BC edges to isolate hubs to facilitate their clustering. Lee et al. [24] proposed an interactive filtering

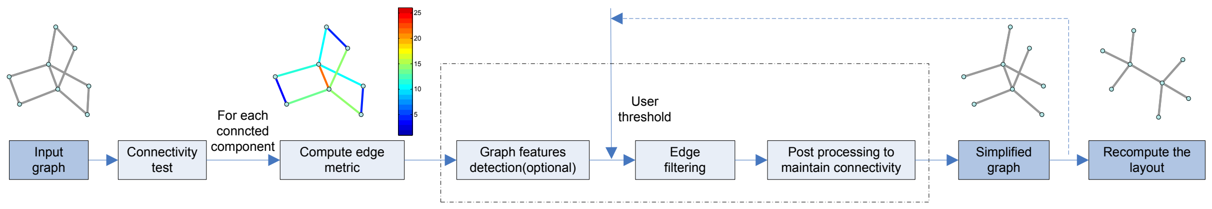


Figure 2: Work flow of our edge simplification program.

approach to reveal local structure when visualizing large graphs by displaying a tree of nodes and edges grown from a user selected root-node. Auber et al. [2] filtered out weak edges in a weighted graph to display a clustered hierarchy of strongly connected graph components.

3 Edge Filtering

Given a graph $G = \{V, E\}$, we find a simplified subgraph $G' = \{V, E' \subseteq E\}$ such that nodes connected by a pathway in G remain connected by some pathway in G'. We also identify and mark feature edges, (specifically those that form cliques) in G so they can be preserved in G'. We compute and assign an edge metric $bc(e)$ (approximating betweenness centrality). We remove only unmarked edges in order of non-decreasing $bc(e)$, and only those edges whose removal does not increase the number of connected components.

Figure 2 illustrates the implementation of this process. The initial connectivity test determines the number of connected components in the initial graph. An input graph with multiple components is handled by processing each of its connected component individually. For the sake of discussion, we assume G contains a single connected component.

We then compute and assign the edge metric. We use a faster approximation of betweenness centrality described in Sec. 4. We also optionally detect graph features and mark relevant edges to prevent them from being filtered. Feature detection can happen concurrently with edge metric assignment unless the feature is based on the metric. In Sec. 3.1 we describe a simple clique detection.

For edge filtering, we store all unmarked edges in a priority queue sorted by non-decreasing metric. With each node, we also store its degree. Before we remove an edge, we check the degree of its two nodes. Only if both degrees exceed two do we remove the edge and decrement them. This degree test helps but does not guarantee that the graph remain connected. To preserve graph connectivity we perform a post-processing pass described in Sec. 3.2.

We continue to remove edges in priority queue order until we reach a user-specified number of edges or a desired edge-metric threshold. The result is a simplified subgraph. We run a layout algorithm on the simplified graph G', which generally produces better results than it would on the original G.

3.1 Feature Detection

The edge filtering system supports feature preservation by marking edges and preventing their removal. Such features can be local or global graph features or user-defined external features. We demonstrate this ability by detecting and preserving cliques, which are maximally connected subgraphs that often represent nodes equally and closely related. For example, stocks that form a clique in a correlation financial network indicate different companies in the same financial sector [5].

Detecting maximum cliques in graphs is an NP-Complete problem. We implemented a fast heuristic, described elsewhere [10], to find highly-connected components in graphs. It was originally used to find clusters in software systems [10] and applied to visualize small world networks [2], but only recently shown to detect graph cliques [1] in time $O(m \deg(G))$ where $\deg(G)$ is the maximum node degree of the graph G. Since G is connected, its degree is at least two, and at most $n - 1$. Given their degree distribution, clique detection in scale-free networks is thus likely $O(mn)$.

Clique detection and preservation is demonstrated by the five highly connected subgraphs of the financial graph visualized in Figure 6(left).

3.2 Connectivity Post Processing

An inefficient method to preserve graph connectivity during edge filtering would be to check whether an edge removal separated the graph into two components, but this approach would take $O(n(m - m'))$ time, where $m - m'$ represents the number of edges removed during edge filtering.

Instead, we use a post processing procedure to recover graph connectivity after edge filtering is finished. We maintain an ordered list recording all of the edge removals. (The list order should coincide with its edge's metric order). If G' contains more than one component, then we label each node in G' according to its connected component. We then iterate through the removed edges in reverse order of removal. If an edge links a pair of nodes belonging to different components of G', then we restore that edge to G' and relabel the nodes in the two newly joined components to indicate it is now a single component. The iteration continues until G' contains a single component, and takes $O(n+m-m')$ time.

The preservation of connectivity, when combined with the betweenness centrality metric, tends to retain shortest paths which depict likely communication pathways for visualization and also aid shortest distance queries for data mining applications.

4 Betweenness Centrality Approximation

Though many edge metrics currently exist, we focus specifically on metrics based on shortest paths to accentuate the communication pathways in a scale-free network. The shortest path, a.k.a. *geodesic path*, is a conventional choice for measuring the relative importance of edges [29, 30]. Our edge metric is a fast approximation of betweenness centrality which counts the number of shortest paths through an edge.

Computing the shortest paths from all nodes to measure edge BC can be very expensive for large graphs. The all pairs shortest path (APSP) can be computed in time $O(nm + n^2)$ by computing BFS from every node [19]. One can also compute APSP in $O(n^3)$ time via the Floyd-Warshall algorithm by initializing a shortest path length function

$$d(u,v) = \begin{cases} w_{u,v} & \text{if } < u,v > \in E \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

where $w_{u,v}$ is the weight of edge $< u,v >$, then set

$$d(u,v) = \min(d(u,v), d(u,x) + d(x,v)) \qquad \forall u,x,v \in V^3. \tag{3}$$

Brandes's algorithm [6] for computing node BC runs similarly in $O(nm)$ time for unweighted graphs and $O(nm + n^2 \log n)$ time for weighted graphs.

Betweenness centrality can be estimated more efficiently. By looking at a subset of vertices proportional to $logn/\epsilon^2$, BC can be estimated to at worst $\epsilon n(n-1)$ with probability $1/n$ in $O((m+n)(\log n)/\epsilon^2)$ on an unweighted graph or $O((m + n\log n)(\log n)/\epsilon^2)$ on a weighted graph[19]. Lloyd's algorithm can also be used to distribute random "pivots" from which to compute BC [9].

Adaptive sampling improves the estimate, where the number of samples needed is affected by the results of the samples. The betweenness centrality of a vertex can be estimated as n^2/t for some $t \geq 1$, within a factor of $1/\epsilon$, for $epsilon < 1/2$, by running single source shortest path (SSSP) from only $\epsilon \times t$ nodes [3].

For scale-free networks, we find that the betweenness centrality of an edge can be approximated sufficiently well by counting the number of shortest paths between only the highest degree hub nodes. Restricting the shortest-path end nodes reduces the absolute BC but since we use it as a metric for edge filtering, it only needs to construct a relative edge ordering.

Since the node degree distribution is logarithmic, we select only $c * log(n)$ of the highest degree hub nodes, where $c = 10$ in our implementation. This BC approximation thus runs in time $O((m+n)\log n)$ for unweighted graphs, and $O((m+n)\log^2 n)$ for weighted graphs. An example of the fidelity of this approximation is shown in Fig. 4 (left).

5 Rendering

The conventional depiction of a graph via constant color line rasterization does not work well for large graphs, as shown in Figure 3(left), as the number of edges can overwhelm display resolution, visual

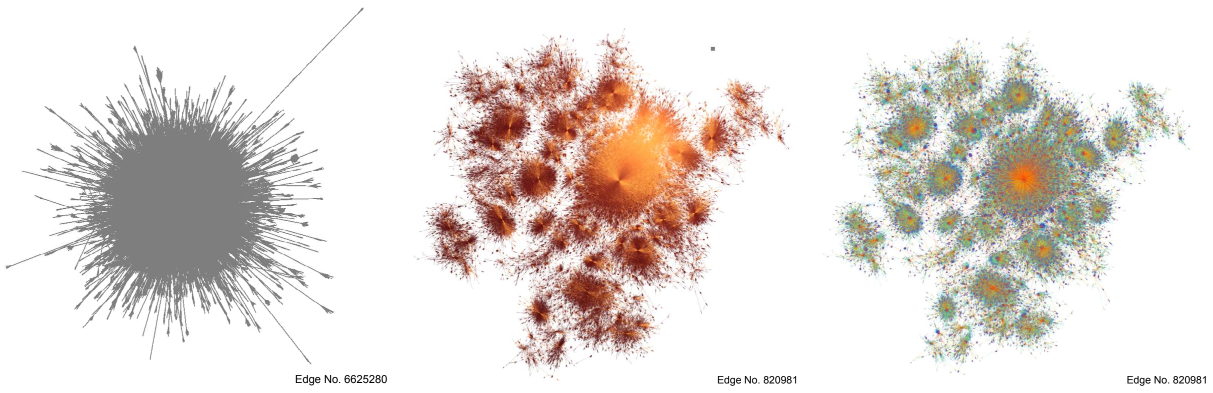


Figure 3: (Left) Unsimplified graph “flickr” rendered with lines of constant color. (Middle) Edge filtered graph “flickr” rendered with anisotropic shading and (Right) edges colored by degree. All graphs were laid out with FR [15].

acuity and perceptual processing. We have thus adapted several approaches from photorealistic computer graphics to help with the visual depiction of large power-law graphs.

We use alpha blending to rasterize very thin lines. For conventional two-dimensional layouts, we display edges in order of increasing edge metric, such that more important edges occlude less important ones.

We use color to highlight the degree distribution of power-law graphs, as shown in Fig. 3, to accentuate hubs among a mass of criss-crossing edges. We use a cold-to-warm color map indexed by node degree, which perceptually brings warm-colored high-degree hubs to the foreground and allows the remaining cool-colored low-degree elements to be slightly less distracting in the background. Since we display edges instead of nodes, we interpolate node colors along each edge.

Since node degree is distributed according to a power-law, mapping colors linearly to degree does not adequately differentiate node degree. We instead borrow a tone-mapping technique, originally developed for converting physical power to perceptual brightness [28], to get the map

$$t(v) = \deg(v)\frac{1+\deg(v)d/\deg(G')^2}{d+\deg(v)} \tag{4}$$

where $t(v)$ is the logarithmic color index used to color node v, $\deg(G')$ is the maximum degree in the (simplified) displayed graph G', and d is a user parameter indicating the degree that should map to $t = 1/2$.

For dense graphs such as scale-free networks, anisotropic shading conveys directionality and allows the user to distinguish individual edges. We have applied a technique originally developed for shading fibrous materials such as fur to the similarly dense lines present in power law graph visualizations [20]. The middle of Fig. 3 demonstrates how anisotropic shading conveys form and directionality by highlighting edges based on their orientation.

6 Results

In this section, we verify our method with error analyses and visual results. These results are based on 2-D layouts computed by either the GEM layout method or 100 iterations of the FR force-directed layout method.

6.1 Error Analysis

To verify our sampling method quantitatively, we measured the errors of various graph metrics introduced by various simplification approaches, namely random edge sampling [27, 25] and geodesic clustering [30]. We averaged the errors generated by random edge sampling over three separate executions. We also

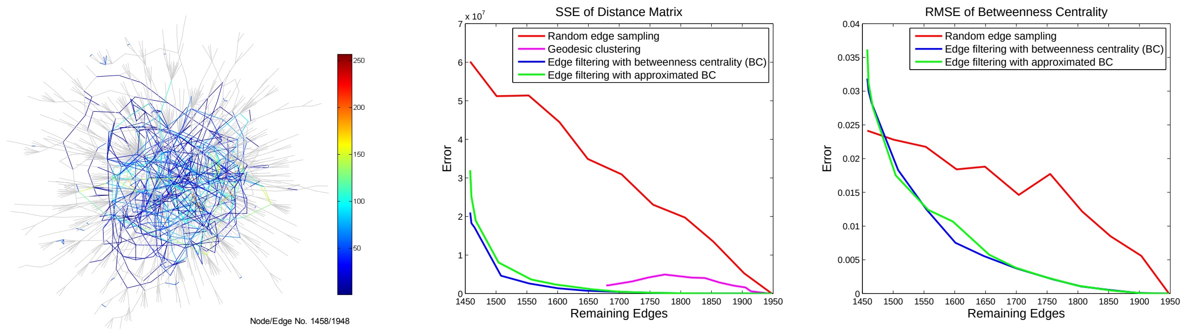


Figure 4: (Left) Plot of the relative error of our BC approximation on the graph "bo." Errors tend to occur closer to cluster centers, and fall within a factor of two of actual BC in almost all edges. (Center) Summed squared error between the geodesic distance matrix of the original graph and that of the graph simplified by random sampling, geodesic clustering and our method, for the graph "bo." (Right) Root-mean-square error of node betweenness centrality when comparing that of the original graph to that of graphs simplified by random sampling and our method, for the graph "bo."

compared our edge filtering performance using the exact edge BC computed on the original graph and our approximation of edge BC computed using shortest paths only between the highest-degree hubs.

Fig. 4 (center) indicates how simplification distorts the shortest paths of a graph. For each graph including the original, we compute an n^2 distance matrix containing the length of the shortest path between the row's node and the column's node. This distance matrix is used for graph mining [30], and is sensitive to changes in the graph. We then measure the sum of the squared differences of these distance matrix elements from those of the original's, for graphs whose edges are filtered from the original m edges to the minimum $n-1$ spanning-tree edges needed to maintain a connected subgraph.

Fig. 4 (right) indicates how removing edges in order of the original graph's edge betweenness centrality affects the betweenness centrality of the nodes in the simplified graph. Node BC is less sensitive than the distance matrix to changes in the graph. Note that our filtering order relies on the edge BC of the original graph and the filtering method does not recompute the BC ordering when each edge is removed. In this RMS error graph, we compare the recomputed node BC against the original node BC to determine the node BC error introduced by simplification. Because BC is not defined for cluster hierarchies, we did not compare geodesic clustering.

Both comparisons show our method generates fewer errors than previous methods for a reasonable number of edges in the simplified graph, which is at least 1500 for graph "bo". Moreover, the approximate BC metric filtering performs about as well as exact BC metric filtering, which further justifies the approximation.

As the number of remaining edges approaches the number of nodes, the error of our approach increases dramatically, likely because low BC edges are preserved to maintain connectivity, causing even higher BC edges to be removed instead.

Fewer edges means easier visualization, so we provide a slider that allows the user to interactively specify the portion of edges to retain to best compromise visual clarity with graph accuracy. We found empirically that this compromise worked best when retaining 3% more edges than nodes (plus any feature preserved edges).

6.2 Visual results

We have applied our method to simplify and visualize several graphs from different areas whose nodes and edges range from hundreds to millions.

6.2.1 Comparing with clustering methods

Fig. 5 compares our method to geodesic clustering [30] on the graph "bo." In both cases, nodes "224" and "1183" are important. Both results find similar neighboring nodes for "224." The main differences between the methods are that our method retains all of the original nodes and its simplified edges also

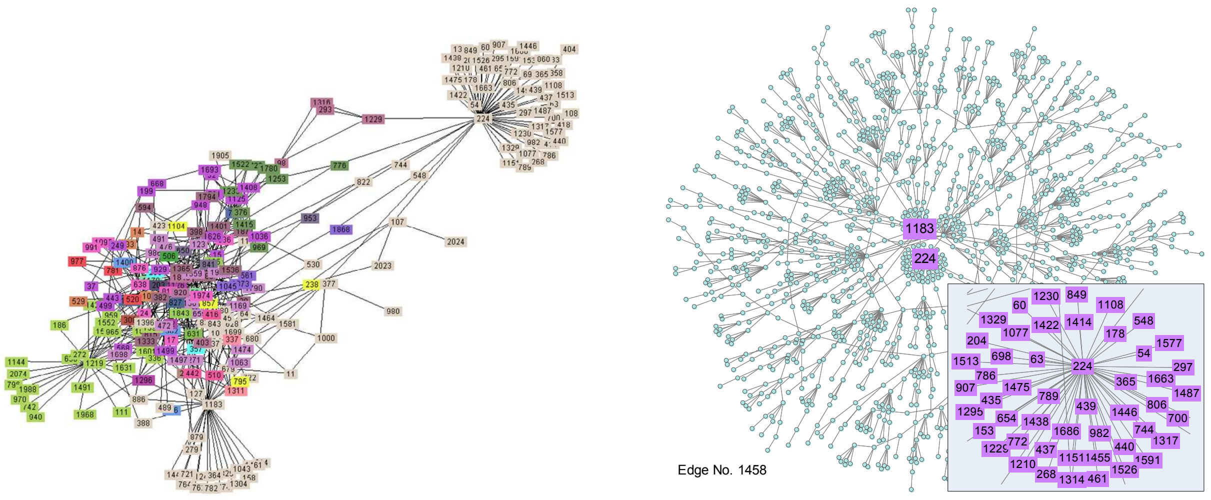


Figure 5: Graph "bo" simplified with geodesic clustering and a spring layout (top) and our method with a GEM layout (bottom).

appear in the original, such that the elements of the simplification retain the original's semantics. The edge between nodes "108" and "224" in the geodesic clustering implies these two proteins interact, but they do not in the original dataset. However, they do not in the original data. Second, our method provides a clear overview of all nodes in the graph while their method only shows a subset.

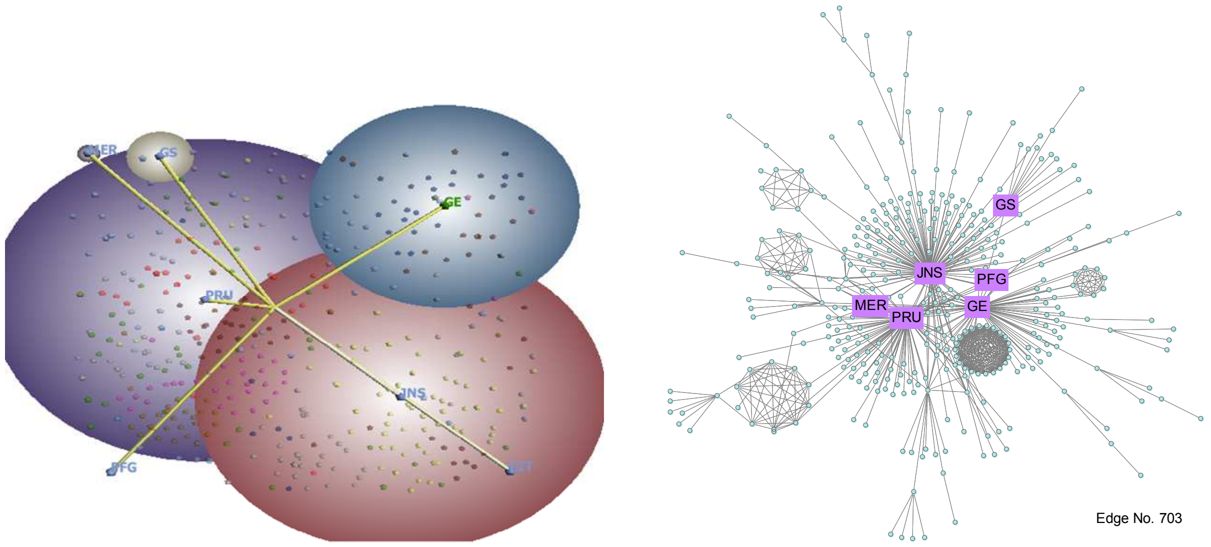


Figure 6: Graph "sp500-38" simplified for top-level view by interactive graph stratification (left) and our method with a GEM layout (right).

Fig. 6 compares our method to graph stratification [23] on the graph "sp500-38," which represents 3,206 cross correlations of price fluctuation of 365 stocks from the S&P 500. Both extract similar information from the original graph. Our result detects and preserves five cliques, and retains the difference between "GS" and "MER" which share few connections yet get clustered in the graph stratification visualization.

Figure 7 visualizes the 773 co-authorship relations among the SIGGRAPH 2007 papers' 328 authors. This graph is small and contains a pair of main components, one of which is examined in more detail. Even for a graph of this smaller size, the edge overlaps produced by the graph's non-planarity make it difficult to visualize, whereas out edge filtering makes the graph planar, allows its layout to be more evenly distributed, but retains a clear depiction of the original's hubs.

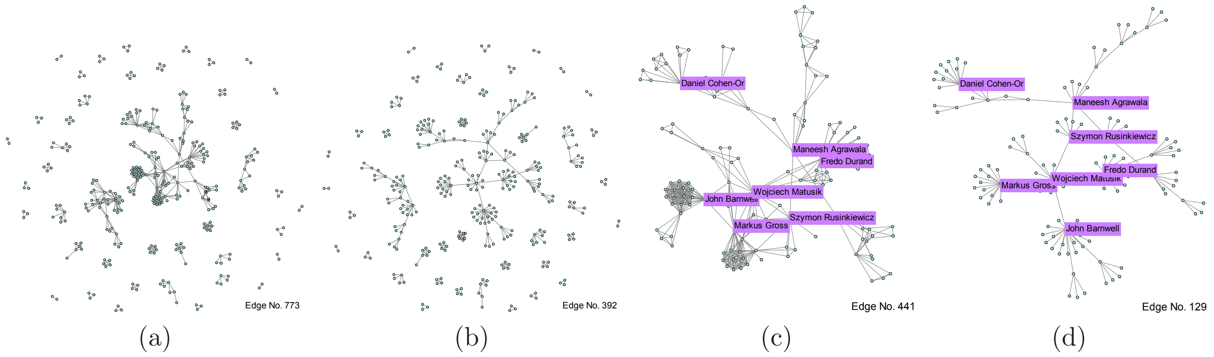


Figure 7: Visualization of SIGGRAPH 2007 paper author collaboration: (a) original dataset, (b) filtered, (c) largest connected component from original, and (d) filtered. Each of them has a GEM layout.

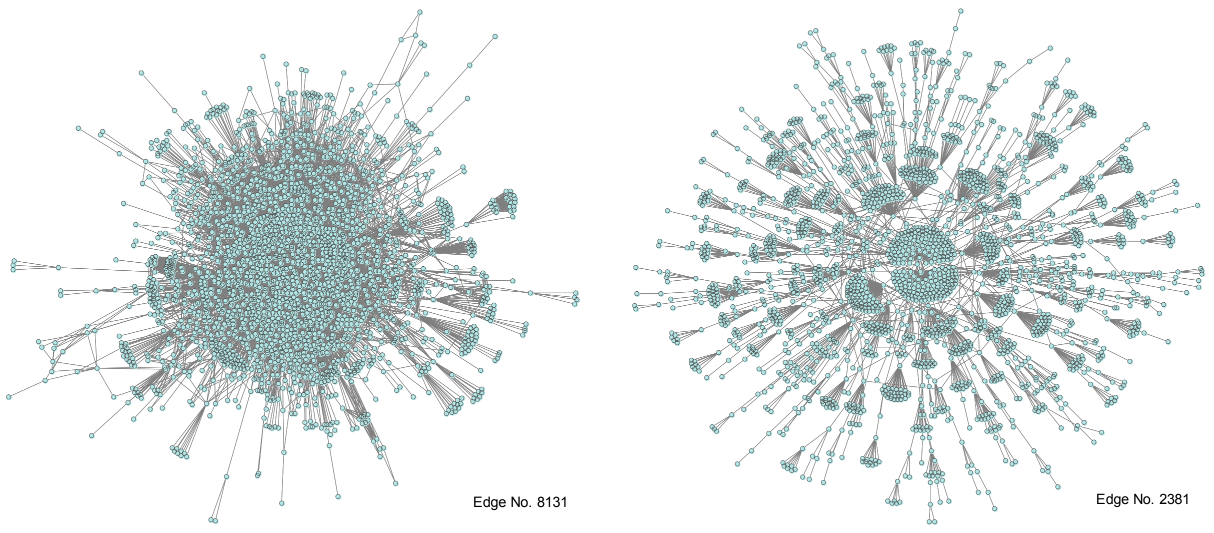


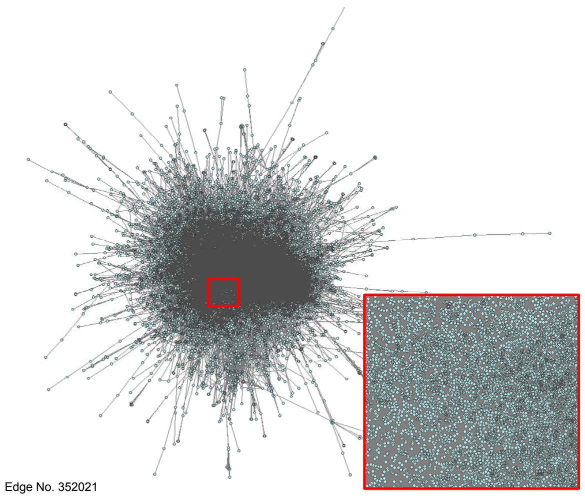
Figure 8: Graph "cg-web" visualized directly (top) and with our edge simplification method (bottom). Each of them has a GEM layout.

Figure 8 visualizes the graph "cg-web," representing 8,131 links between 2,269 computational geometry websites. This very dense collection becomes slightly easier to visualize when our edge filtering reduces it to 2,381 edges, better revealing its hubs and yielding a more even radial structure.

Figure 9 visualizes the graph "hep-th," representing 27,400 citations among 352,021 high-energy theoretical physics preprints on arXiv. The mass of nodes and edges makes visualization impractical, but edge filtering organizes this detail into a connected constellation of clumps to more clearly identify the seminal publications and areas of research within this field.

Figure 10 shows the results on the directed graph data "as-rel.20071008", which is obtained from the CAIDA's ranking of Autonomous Systems. Each of the 26,242 nodes represents the collection of IP networks and routers under the control of a single entity, and each of the 53,174 directed edges represents a service. Our filtering and display ignores edge direction, but edge filtering better organizes the global and local hubs from their default clump into a more spatious configuration.

Figure 3 shows the results on the graph data "flickr", which represents 6,625,280 friendships of 820,878 users on the photo sharing website flickr.com. For such massive graphs containing more edges than pixels, the anisotropic shading techniques better indicate the hubs in the edge filtered version, and also variation among the otherwise unform edge directions emanating from the center. The grey square in right upper corner of the middle image is an interactive widget controlling the lighting direction.



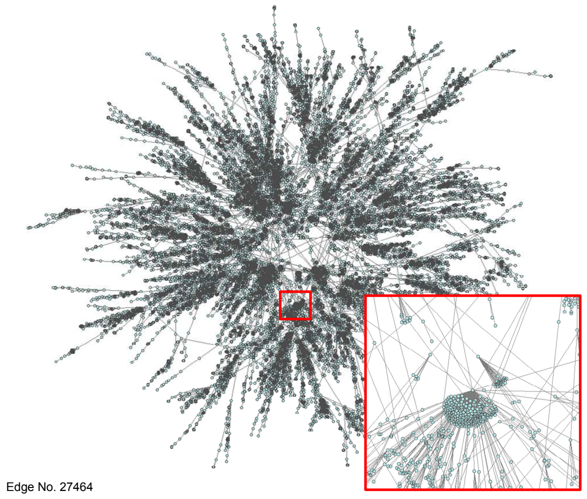


Figure 9: Graph "hep-th" visualized directly (top) and with our edge simplification method (bottom). A comparison in a small portion shows that our method helps user to perceive more graph structures. Each of them has an FR layout.

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

Table 1: Edge metric computation performance.

6.3 Performance

Our experiments validate the complexity analysis of Section 4, with edge metric evaluation performance proportional to $O(log(n) * m)$ on undirected graphs. We found that edge simplification can also improve the performance of the layout algorithm, by 18% on average. Because our simplification does not affect node count, the all pairs repulsive force computation dominates layout time. Experiments were performed on a modern workstation. Table 1 summarizes our results.

7 Conclusion

We have discovered, through the course of filtering edges to remove distracting edge crossings in large non-planar graphs, that reducing edge count while maintaining node count facilitates faster and better appearing layouts. We have also discovered that limiting the computation of betweenness centrality to using shortest paths between high-degree hubs of power-law graphs reduces the time complexity class of its computation with little cost in visual fidelity. Finally, we have implemented these tools in an interactive graph visualization system that supports additional high-quality graphics rendering modes that accentuate the hubs in overview visualizations of large scale-free networks. These discoveries povoke one to consider further the application of these methods to time varying graphs and their integration into a focus-based graph exploration system [27].

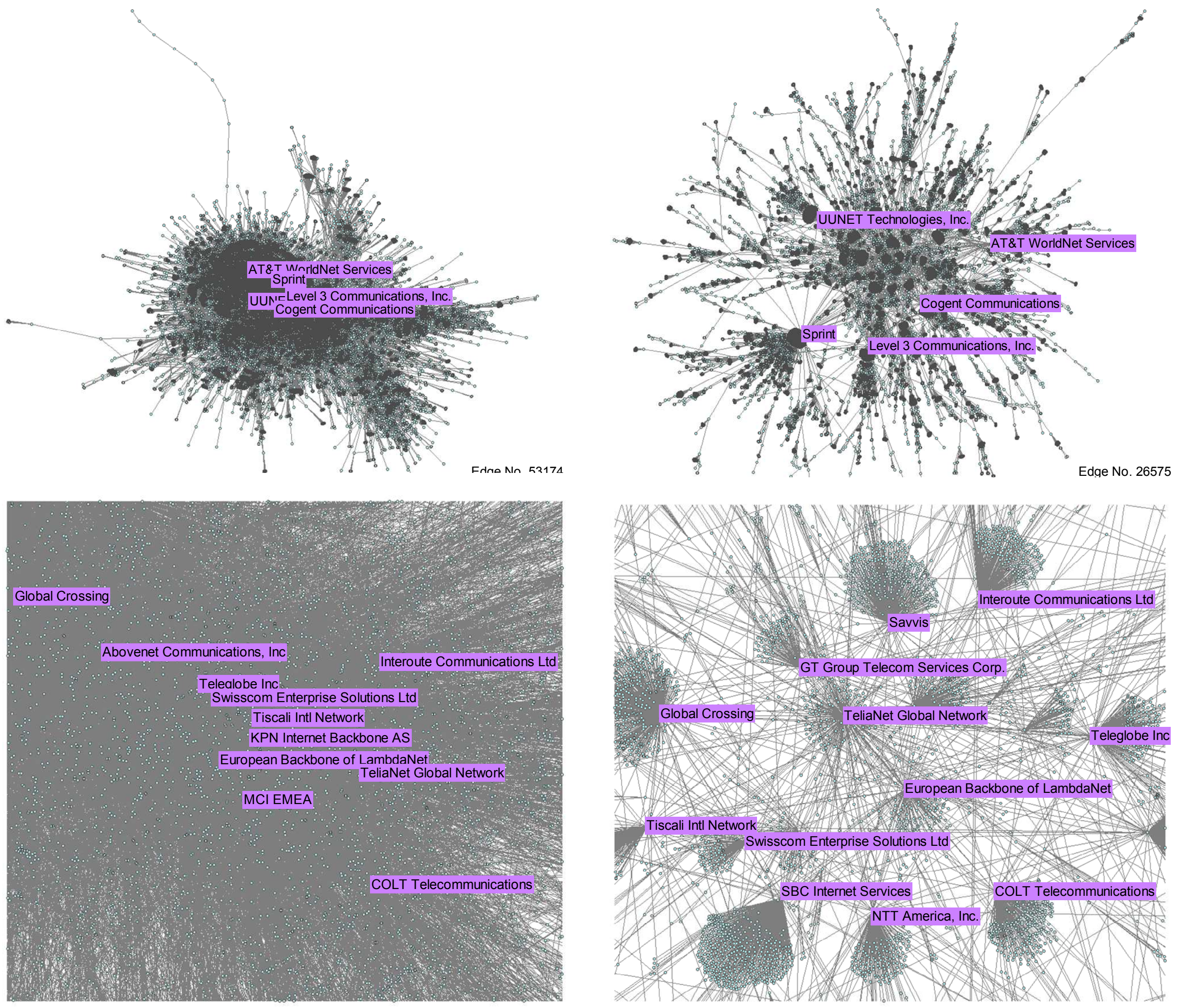


Figure 10: Graph “as-rel.20071008” visualized directly (left column) and with our edge simplification method (right column). Labels are drawn for important nodes in the dataset. Each of them has an FR layout. More details about this dataset can be found at http://as-rank.caida.org/ and its ranking on Oct. 8 2007.

References

[1] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[2] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon. Multiscale visualization of small world networks. In *IEEE Symposition on Information Visualisation*, pages 75–81, 2003.

[3] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail. *Approximating Betweenness Centrality*, pages 124–137. Springer, 2007.

[4] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[5] V. Boginski, S. Butenko, and P. M. Pardalos. Statistical analysis of financial networks. *Computational Statistics & Data Analysis*, 48:431–443, 2005.

[6] U. Brandes. A faster algorithm for betweenness centrality. *J. Math. Soc.*, 25(2):163–177, 2001.

[7] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *In review: Social Networks*, 2007.

[8] U. Brandes, M. Gaertler, and D. Wagner. *Experiments on Graph Clustering Algorithms.* Springer Berlin / Heidelberg, 2003.

[9] U. Brandes and C. Pich. Centrality estimation in large networks. *Intl. J. Bifurcation & Chaos*, 17(7):2303–2318, 2007.

[10] Y. Chiricota, F. Jourdan, and G. Melancon. Software components capture using graph clustering. In *Program Comprehension, 2003. 11th IEEE International Workshop on*, pages 217–226, 2003.

[11] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.

[12] P. A. Eades. A heuristic for graph drawing. In *Congressus Numerantium*, volume 42, pages 149–160, 1984.

[13] L. C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40(1):35–41, 1977.

[14] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs. In *GD '94: Proceedings of the DIMACS International Workshop on Graph Drawing*, number 894, pages 388–403, 1994.

[15] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[16] P. Gajer and S. G. Kobourov. GRIP: Graph dRawing with intelligent placement. In *GD '00: Proceedings of the 8th International Symposium on Graph Drawing*, pages 222–228, 2000.

[17] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[18] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *Graph Drawing*, pages 285–295, 2004.

[19] R. Jacob, D. Koschutzki, K. A. Lehmann, L. Peeters, and D. Tenfelse-Podehl. *Algorithms for Centrality Indices*, pages 62–82. Springer, 2005.

[20] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph.*, 23(3):271–280, 1989.

[21] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989.

[22] Y. Koren, L. Carmel, and D. Harel. Drawing huge graphs by algebraic multigrid optimization. *Multiscale Modeling & Simulation*, 1(4):645–673, 2003.

[23] G. Kumar and M. Garland. Visual exploration of complex time-varying graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):805–812, 2006.

[24] B. Lee, C. S. Parr, C. Plaisant, B. B. Bederson, V. D. Veksler, W. D. Gray, and C. Kotfila. Treeplus: Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426, 2006.

[25] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.

[26] M. E. J. Newman. Detecting community structure in networks. *European Physical Journal*, B 38:321–330, 2004.

[27] D. Rafiei and S. Curial. Effectively visualizing large networks through sampling. In *IEEE Visualization*, page 48, 2005.

[28] E. Reinhard. Parameter estimation for photographic tone reproduction. *J. Graph. Tools*, 7(1):45–52, 2002.

[29] S. White and P. Smyth. Algorithms for estimating relative importance in networks. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 266–275, 2003.

[30] A. Y. Wu, M. Garland, and J. Han. Mining scale-free networks using geodesic clustering. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 719–724, 2004.