Eurographics/ IEEE-VGTC Symposium on Visualization 2008
A. Vilanova, A. Telea, G. Scheuermann, and T. Möller
(Guest Editors)

Volume 27 (2008), Number 3

Centrality Based Visualization of Small World Graphs

F. van Ham and M. Wattenberg

IBM Research, Cambridge, USA

Abstract

Current graph drawing algorithms enable the creation of two dimensional node-link diagrams of huge graphs. However, for graphs with low diameter (of which "small world" graphs are a subset) these techniques begin to break down visually even when the graph has only a few hundred nodes. Typical algorithms produce images where nodes clump together in the center of the screen, making it hard to discern structure and follow paths. This paper describes a solution to this problem, which uses a global edge metric to determine a subset of edges that capture the graph's intrinsic clustering structure. This structure is then used to create an embedding of the graph, after which the remaining edges are added back in. We demonstrate applications of this technique to a number of real world examples.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces

1. Introduction

Force directed layout algorithms are a popular method of visualizing graphs. They are straightforward to implement and provide satisfactory results for many different types of networks. Recent research in multi-scale and multi-body algorithms [HJ05] has made it possible to create node link layouts of graphs of tens of thousands of nodes in limited time. However, for dense graphs or sparse graphs with small diameter the images these algorithms produce are often not as informative as one would like, even if the input graph is not very big. One important set of problematic graphs are those with the "small world" property [WS98]. Such graphs exhibit both small average path length compared to their size and a high degree of clustering, compared to a random graph. Precisely because of the small average path length between nodes, node link visualizations of small world graphs often result in a uniform clump of nodes and the resulting tangle makes it hard to identify structural features such as dense clusters or small subtrees. The large number of nodes in a small space further aggravates the problem of overlapping edges, so that finding paths between nodes in the graph becomes virtually impossible.

The underlying cause of this overlap is that the force directed layout algorithm tries to keep all connected nodes close to each other. In practice, a more informative layout can be achieved if we eliminate some of the edges before applying a layout algorithm. The trick lies in deciding which edges should be kept for the initial layout, and which edges should be added afterward. Weighted graphs allow us to make decisions on which edges in the graph are more important than others and it is tempting to simply use edge weights to prune the graph. However, we have no guarantees on the distribution of edge weights over the graph. It might very well be that all dense structures in the graph are formed by higher weight edges. Apart from that, many real world graphs are unweighted. Ideally, we would like to see an edge selection algorithm that meets the following criteria:

1. It should not produce disconnected components if the original input graph was connected.
2. It should strike a balance between reducing the number of edges such that the resulting (reduced) graph is easier to visualize, while maintaining as many of the original edges as possible.
3. It should leave the clustering structure in the original graph intact. If two nodes are in the same structural cluster, they should be positioned closely together in the reduced graph.

In this paper we show that by using a graph metric called edge betweenness centrality we can extract sparse structures from the graph that capture the overall structure well and conform to the criteria above. Section 2 discusses related work in dealing with dense and small world graphs, while

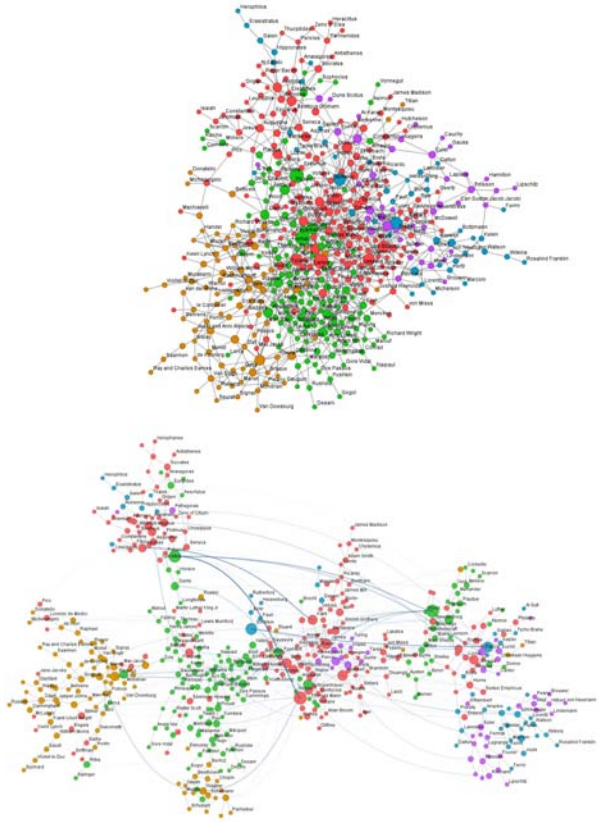


Figure 1: *Layouts of a sparse graph (|V| = 500, |E| = 1032), with data taken from [Lov07]. Colors represent different classes of nodes, sizes represent node centralities. In the top view there is some rough clustering information visible but connectivity information is obscured by the large number of links in the center. The bottom view shows the graph after our method has been applied, making it easier to identify clusters and the connections between them.*

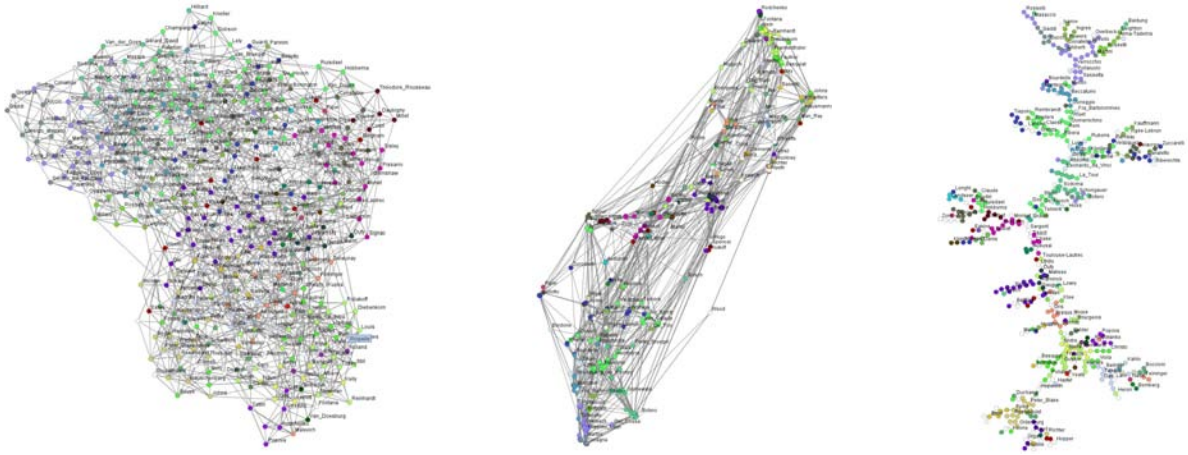
Section 3 outlines the main idea behind our method. In Section 4 we generate different representations of a small diameter graph, each representation having increasing density.In Section 5 we evaluate the method by determining how resistant it is to random perturbations of the graph. Finally, section 6 presents conclusions and recommendations for further work.

2. Related work

There are several existing approaches to handling the problems inherent in force directed layouts for low diameter graphs. One method is to avoid node link diagrams altogether by using an adjacency matrix representation. Although adjacency matrices can deal with graphs of arbitrary density and can show internal clustering structure [vH03], they do not convey the internal connectivity structure of a graph well and are unsuitable for path finding [GFC04]. This is mostly due to the fact that connected edges in a matrix-based visualization need not be close in the visual representation. Nevertheless, for very dense graphs adjacency matrices are usually preferable to node link diagrams. A second approach is to compute a clustering on the input graph and then render this clustered graph structure, in the hope that the clustered structure will be easier to visualize [ACJM03]. Although this approach scales to handle large graphs, it has its drawbacks. The first problem is that the concept of a graph cluster is ill-defined. Graph clusters can exist at many different scales and finding the right scale it not always easy. Second, the abstraction created by clustering a graph with small diameter will itself be a graph with small(er) diameter again. Third, it is often hard for the user to relate the clusters created by a clustering algorithm to the contents of the original graph without some way of giving a concise overview of the contents of a graph cluster. A partial solution to this problem was offered by Noack [Noa05], who used a modified force algorithm to generate a layout of the whole graph, such that graph clusters show up as dense clusters of nodes. Because all nodes are still visible in the resulting visualization a user can more easily form hypotheses as to why this cluster was formed by overlaying attribute information. A third approach is to not render all of the edges, but extract a subset of edges that are a good approximation of the graph structure. Extracting such a subset is especially easy if edges in the graph are weighted, giving us some indication which edges in the structure are more important. Straightforward removal of the lowest weighted edges allows us to transform a dense graph into another graph with an upper limit on the amount of edges. The major disadvantage of this approach is there is no guarantee that the highest weight edges are uniformly distributed throughout the graph, so the generated graph might not capture the structure well. More advanced approaches such as pathfinder networks [SDD89,CM03], do a decent job of capturing the connectivity structure. Unfortunately pathfinder networks are not effective for unweighted networks, since they only include edges that are part of the shortest path between a node pair which, in the undirected case, includes all edges in the graph.

3. Approach

Given an undirected graph $G = (V,E)$, we denote an edge $e \in E$ connecting vertices $x \in V$ and $y \in V$ with $e(x,y)$. Here we will consider the graph to be undirected, i.e. $e(x,y) = e(y,x)$. Any directed graph can always be made undirected by ignoring the arc's directions. Each undirected edge has an associated numerical weight $w(e(x,y))$, if we are dealing with unweighted graphs, we assume $w(e) = 1$ for all $e \in E$. A path between two vertices x and y is an alternating sequence of nodes and edges $< x, e(x,v1), v1, e(v1,v2), v2, ..., vn, e(vn,y), y >$. The length of a path is equal to the sum of the weight of its edges. A short-



(a) **(b)** **(c)**

Figure 2: *Different layouts of the same graph, representing 500 artists and their mutual influences. Coloring is done by artistic movement.* ***(a)*** *Shows a the result after a standard force directed layout* ***(b)*** *Shows the result after a force directed layout using a Lin-Log model [Noa05]* ***(c)*** *Shows the result after computing a minimum edge centrality spanning tree and performing a force directed layout.*

est path between two vertices s and t is a path with minimal length. We define $\sigma_{st} = \sigma_{ts}$ as the total number of unique shortest paths that connect vertices s and t. By convention we assume that $\sigma_{ss} = 1$. Let $\sigma_{st}(v)$ be the number of shortest paths that contain a vertex v (note that $\sigma_{st}(t) = \sigma_{st}(s) = \sigma_{st}$). We can now define the betweenness centrality $C_v(v)$ of a vertex v as [Ant71, Fre97]:

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

Similar to the betweenness centrality of a vertex we can define the edge betweenness centrality (we will simply refer to it as edge betweenness or betweenness from here on) of an edge e. Let $\rho_{st}(e)$ be the total number of shortest paths from s to t that contain an edge e. Then the betweenness $C_e(e)$ of an edge $e \in E$ can be expressed by:

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

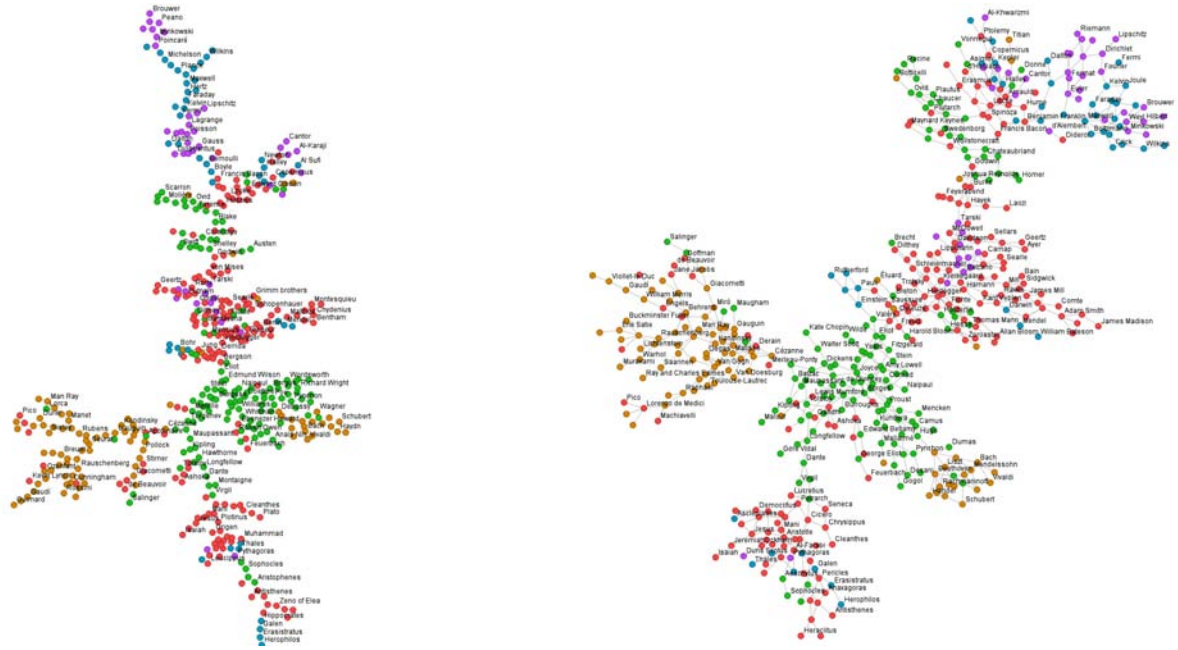
Although naïve implementations of the betweenness metric can compute betweenness for all nodes and edges in $O(N^3)$, more efficient algorithms were proposed by [Bra01] and, recently, [BP07] . For an unweighted graph, exact betweenness centrality for both nodes and edges can be computed in $O(VE)$, for weighted graphs complexity increases to $O(VE + V^2 \log V)$.

Edge betweenness captures the connectivity structure of the graph, as edges with high betweenness are present in a large number of shortest paths and can be considered important bridge edges between loosely connected parts of the graph. This fact is exploited by clustering algorithms [NG04] to find strongly connected clusters in the graph. Other researchers [KNJ04] have used the betweenness metric to create spanning trees of complex graphs. The rationale is that high betweenness edges capture most of the connectivity in the graph and we should include as many of them as possible in the spanning tree. However, this approach fails our third requirement as most of the edges in the reduced graph will connect nodes in different clusters.

The main approach and main contribution of this paper rests on the reverse assumption: If edges with high betweenness serve as bridge edges between different clusters, edges with low betweenness will connect nodes that are in the same cluster. From here on, we refer to edges connecting nodes in the same graph cluster as cluster edges. If we want to create a reduced graph that leaves the original clustering structure intact, we can use these low betweenness edges to decide which nodes should be adjacent in the reduced version. Nodes in the same cluster will generally have multiple disjoint shortest paths connecting them (and hence have low betweenness), while nodes in two different clusters have few disjoint shortest paths between them.

4. Graph Reductions

In this section we present a number of graph reductions that use these ideas to create sparse node link representations of our input graph and then add more edges to them. Note that this idea also resonates with the small world model proposed by [WS98]. Starting from a regular lattice, a small world network can be generated by adding a relatively small number of random connections. It is precisely these links that cause



(a) (b)

Figure 3: *Two representations of a graph representing influences between great thinkers, taken from [Lov07]. Colors represent the general class of person, with philosophers in red, writers in green, artists in beige, scientists in blue and mathematicians in purple.* ***(a)*** *Minimum betweenness spanning tree, showing rough clustering structure in the graph* ***(b)*** *minimum betweenness planar graph, showing 81% of the edges in the original graph. Compare with the standard force directed view in the top of Figure 1.*

the small average path length to decrease, while the local clustering structure is present in the lattice. If we can identify and remove most of these irregular connections in the small world network after they have been added, we can reconstruct the original lattice and use it to drive our visualization. In what follows we assume the input graph is connected and that edge and node centralities have been computed. If a graph is not connected we can always create a reduced version by reducing each of its connected components in turn. Our approach can be divided into main two phases: in the first phase we reduce the amount of edges that are fed into the force directed layout algorithm. In the second phase we add the removed edges back into the layout. Sections 4.1 and 4.2 below outline two methods of creating a reduced version of our original graph. In section 4.3 we will show how we can reinsert the edges while keeping the resulting visualization readable.

4.1. Minimum spanning trees

Because we assume our input graph is connected, we also need to create a connected abstraction (see requirement 1 in section 1). The sparsest possible connected abstraction is a tree. Since edge centralities are already computed, we can use these to determine the tree with smallest total edge betweenness. Such a tree is easily generated by a minimum spanning tree algorithm in O($E\ logV$).

As a first test of the hypothesis that removing high-betweenness edges would yield a structurally meaningful abstraction, we decided to generate a minimum spanning tree for a real-world example. We chose a small sized network of painters and sculptors, taken from [vHvW04]. The network contains 500 well-known artists and 2486 connections, representing influences between artists. Previous work [vHvW04, Noa05] has found local clusters in the graph that correspond to artistic movements. A standard node link visualization succeeds in keeping artists with identical movements (indicated by color in Figure 2a) somewhat together, but obscures connectivity information. The Lin-Log layout algorithm [Noa05] (Figure 2b) manages to roughly group nodes in clusters although it is not always clear why some nodes are clustered together and other groups are split apart. Regardless, the large amount of edges in both these representations make them hard to read.

In Figure 2c we ran a minimum spanning tree algorithm using the computed edge centralities, and rendered the resulting graph with radial tree layout, followed by a few iterations of a force directed algorithm. The minimum spanning tree proves a surprisingly good match to the inherent cluster-

ing structure present in the graph. Most movements show up as connected pieces of a single color and the overall ordering of the nodes reflects the ordering from classical to modern artists (which is also present in both other visualizations).

This rudimentary view of the graph is already helpful. Although it does not allow us to spot the cluster borders, we can use it to gauge how a particular attribute is distributed over the structure of the graph, and if it might be a good indicator to predict clusters. Generally speaking, if connected branches in the tree share the same value for an attribute, this attribute will be correlated with the structure. Here we have used a radial tree layout, followed by a few iterations of a force directed layout for aesthetic reasons. However, we could also use a simple tree layout algorithm to convey the same structure and avoid running expensive layout algorithms altogether.

4.2. Minimum Centrality Planar Graphs

Although the minimum spanning tree seems to reflect the cluster structure well, it does not meet requirement 2 in section 1-that is, it throws away more edges than is necessary. Force directed algorithms can provide useful embeddings for denser graphs than trees, and there is no reason why we cannot add more edges to a minimum betweenness spanning tree. In theory, we could keep on adding lowest centrality edges until our graph has the desired density. Adding edges in this manner suffers the same drawback as adding edges with low weight in weighted graphs: Since the distribution of betweenness over the graph is non-uniform, we likely will add too many edges in some areas and too few in others. An alternative to looking for a global cutoff value is to use a structural metric to decide when to stop adding edges. One possible structural metric is to only add edges that connect nodes with a spanning tree distance less than a given threshold, with the threshold a function of the diameter of the spanning tree. This will avoid long edges in the graph by definition and will allow us to create a denser representation without shortening the average path length too much. Disadvantages of this approach are that many edges will be added to any dense clusters present in the graph and that nodes in different subbranches rarely connect even though they might be in the same structural cluster.

Planar graphs are graphs which can be embedded in the two dimensional plane without edge crossings and are generally sparse enough to be drawn nicely by a force directed layout algorithm. Conveniently, planarity testing of a graph can be done in linear time [HT74] . Note that here we use the planarity criterion merely as a local measure of graph complexity, and we do not intend to create a planar embedding of the result, as that might destroy cluster structure. Trying to add as many (weighted) edges as possible while maintaining planarity is known as the (NP hard) *maximum weighted planar subgraph* problem in graph theory, but a number of heuristics exist (see [Lie01] for an overview). Here we use a simple greedy heuristic that inspects edges in order of non-decreasing betweenness and adds them to the spanning tree if their addition does not invalidate planarity, leading to a worst case complexity of $O(VE)$. Incremental planarity testing [WH04] can bring this time down to $O(V \log V)$ and more advanced heuristics [JM96] can provide better overall results, but their implementations are complex and were unavailable to us. Since we add edges in order of non decreasing betweenness, most edges that are added are cluster edges and they connect nodes that will not be far apart in the spanning tree. This means that the structure of our planarized graph will roughly correspond to the structure of the minimum spanning tree (and consequently the clustering structure of our original graph).

Figure 3 shows the result of running this maximal planarization heuristic on a graph taken from [Lov07]. This graph represents 500 great thinkers throughout the ages and their mutual influences. Nodes are classified by their main occupation: artists (beige), philosophers (red), writer (green), scientist (blue) or mathematician (purple). Figure 3a shows the computed spanning tree. Some interesting features that we can already make out from this very basic representation include a separate section of artists split off on the right, which turn out to be all composers. The mixed branch of writers, philosophers and scientists in the bottom right of the image are mostly early Greeks. The purple section of mathematicians in the center contains logicians which would explain their close coupling to the main body of philosophers. The green writer cluster near the top appears to be a combination of playwrights and early 19th century English poets. The above observations show that we can infer a wealth of information from this extremely sparse view of the graph, which would have been harder when looking at its original force directed rendering in Figure 1. Figure 3b shows the result of running the planarization heuristic on the spanning tree in Figure 3a. It is important to note that, although our graph is planar, the straight line force directed embedding generated here is not planar. Here we opted for a straight-line embedding that keeps all edges short, making it easier to identify clusters. Most edges added to the spanning tree are edges with relatively low centrality, and thus connect nodes that were already close to each other, so the main backbone structure that we have computed for the graph has remained intact. Note that very few edges connecting different clusters have been added, although some of the elongated tree structures have been collapsed into more compact clusters. In terms of information content, the planarization procedure has added 81% of the edges back in for this particular graph.

In a way this planar view gives us a good overview of the local interactions in the graph and how different tightly connected groups of nodes are positioned relative to one another. However, in many cases the user will be interested in edges connecting different clusters of nodes. For example, in the sample above one might be interested in non obvious connections between two people from different groups.

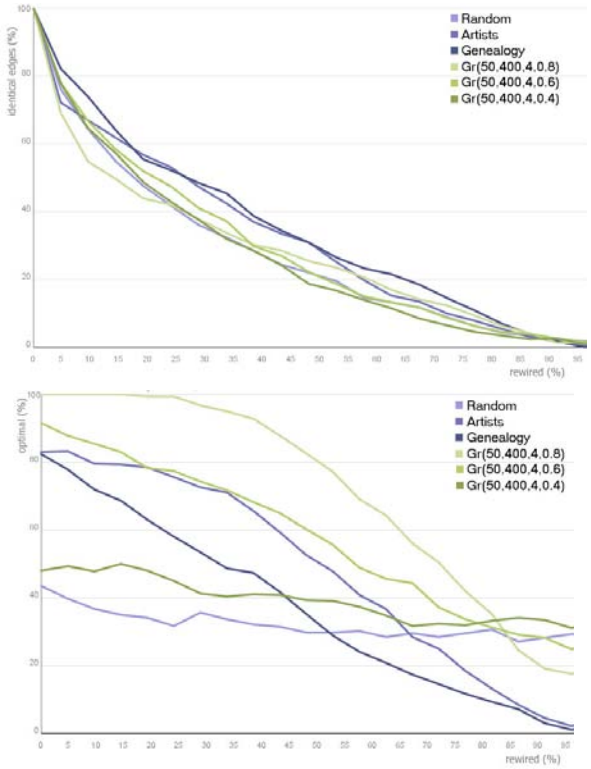


Figure 4: *Stability of the betweenness metric under noise.The top graph shows the correspondence between the minimum betweenness spanning tree of a graph and the minimum betweenness spanning tree of the same graph with an increasing amount of edges rewired. Bottom shows the quality of the computed spanning tree as a function of noise added.*

Or, in another use case, a software engineer looking at a call graph may want to know about connections between high-level modules in the program. Ironically, it is precisely these high betweenness edges we have left out when creating our planar version. In the next section we add back the rest of the edges to the visualization.

4.3. Re-inserting edges

As mentioned above, we want to ensure that edges connecting nodes in different graph clusters remain visible in our final visualization. A first idea might be to use the layout of the planar version of our graph, and simply add the remaining edges to that embedding. However, most of the remaining edges connect distant nodes and hence tend to be longer. Adding all of them leads to a messy representation in which paths are hard to discern.

Luckily, the edge betweenness values that we computed for the graph allow us to decide which non cluster edges carry the most information. In this case we want to emphasize edges that have high edge betweenness, since they carry most of the shortest paths between the nodes. We therefore decided to render the remaining edges as curved arcs, overlaid on the planar layout generated in the previous step. We mapped the computed betweenness values to the alpha channel such that edges with low betweenness show up as fainter connections. We chose arcs because this makes the distinction between edges connecting clusters and edges connecting nodes in the same cluster clearer, while at the same time providing a depth cue (see Figure 3 bottom)

5. Measure stability

The results above suggest that low betweenness edges seem to be a good indicator of local graph connections and clusters. One way to measure this quantitatively is to examine the effect of small perturbations. If there are large changes in the low-betweenness spanning tree when we add only a few edges to the graph, our metric is unstable and might not be a good indicator of global structure. To measure the stability of our metric we therefore prepared a sample consisting of the two real world graphs discussed before and four pseudo-random clustered graphs. A pseudo random clustered graph $G_r(n,e,c,p)$ is a graph with n nodes divided into c equal sized clusters, connected by e edges. The probability that any edge connects two nodes in the same cluster is p. We generated three pseudo random clustered graphs with parameters $n = 50$, $e = 400$ and $c = 4$. The strength of the clustering was varied by setting p to 0.8, 0.6 ,0.4 and $\frac{1}{16}$. Note that $p = \frac{1}{16}$ equals an Erdös-Renyi random graph, since the probability of selecting an edge between two clusters at random is $\frac{1}{c^2}$.

Although the baseline clustering for the pseudo-random graphs is known, we had no baseline clustering for both real world graphs. We decided to use clusters generated by the MCL graph clustering algorithm [vD00] to evaluate how well our spanning tree corresponds to the inherent clusters in the real world graphs. MCL, using its default settings, computed 115 and 163 structural clusters for the artist and genealogy datasets, respectively. For each graph in the sample we then computed the minimum betweenness spanning tree before and after adding a varying amount of noise. Noise was added by randomly rewiring a fraction of the edges in the graph.

We used two different measures to compare the noisy spanning tree to the original. Firstly, we were interested in the percentage of identical edges in both trees. Ideally, the spanning tree should degrade smoothly when we increase the amount of noise. A second measure we looked at was how well the betweenness metric reflects the internal clustering of the graph. We measured the number of cluster edges in the (noisy) spanning tree (i.e. edges that connect nodes in the same cluster) and divided this by the number of edges that an ideal spanning tree should have. We consider an ideal spanning tree a tree that represents every cluster in the graph

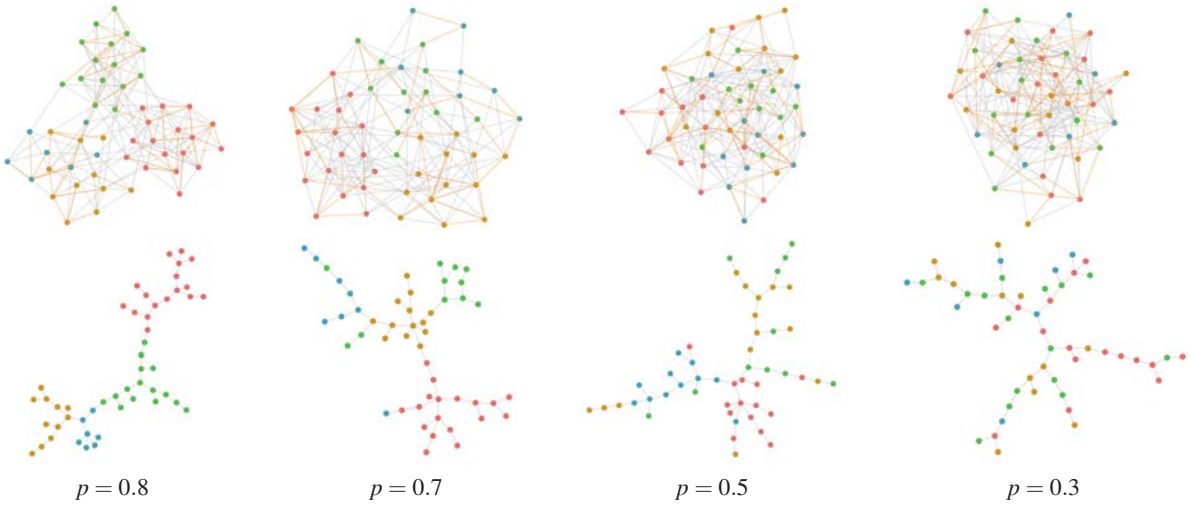


Figure 5: *Four different pseudo random graphs with 50 nodes, 200 edges and 4 clusters. Varying the ratio between cluster edges and non cluster edges lead to different degrees of clustering. The top row shows layouts of the graphs with spanning tree edges highlighted. The bottom row shows the corresponding spanning trees.*

as a connected branch. Since every spanning tree has exactly $|V|-1$ edges and at least $C-1$ of these are needed to connect C different clusters, the maximum possible number of cluster edges is $|V|-C$. Note that the maximum value of this metric is 1 and the expected value for a random graph is approximately $\frac{1}{C}$. We expected the latter measure to be more stable than the first one, since adding even a small amount of noise can significantly influence the shortest path counts for each edge. Although the betweenness metric can identify which edges connect different clusters and which edges connect nodes in the same cluster, the actual cluster edge that gets picked for the spanning tree depends on a potentially small difference in betweenness among the set of internal edges for that cluster.

Figure 4a shows the stability of the spanning tree when adding small amounts of noise. Interestingly enough, the behavior seems largely independent of graph structure and size. The amount of correspondence decreases sharply at first and then gradually decreases to zero as noise is increased. We believe this difference in rate is due to the fact that most edges in the initial spanning tree are low betweenness edges. Since it is precisely these edges that are affected most by a small change in the graphs shortest path structure, most of these edges will be substituted for other edges within the same cluster. This hypothesis can be verified if we look at Figure 4b. The spanning tree generated for $G_r(50,400,4,0.8)$ is the optimal spanning tree with respect to the graph's internal clustering. This metric is relatively stable, suggesting that although a large number of edges has changed (only 50% of the original edges remain when noise is increased to 15%), these edges are substituted for other cluster edges.

The betweenness metric performs better if the difference between the amount of cluster and non cluster edges is more pronounced. For a rate of 0.8 the metric always finds an optimal spanning tree. This quickly decreases as we increase the fraction of non cluster edges in the graph (see Figure 5). The quality of the tree for the random graph hovers around its statistical expectation value of 0.25. The algorithm still performed reasonably well for both real world datasets. The artist and genealogy datasets contain 48% cluster edges and 39% cluster edges, respectively. Given that more than half of the edges in both datasets connect nodes in different clusters, the betweenness metric still produces spanning trees that are above 80% of optimal.

6. Conclusions and Further Work

Instead of trying to visualize small diameter graphs by using a standard force directed layout method, we opt to extract a sparse version of the global graph structure first. Here, we have presented a method that uses a minimum centrality metric to identify edges which connect different graph clusters. By using two low complexity graph algorithms (a spanning tree algorithm and a planarity testing algorithm) we created a planar representation of the graph's overall structure, after which non cluster edges were added back in. The overall complexity of this method is O(VE).

In this visualization we have not used any node attribute information and the resulting visualization only reflects the structural properties of the graph. This makes it possible to correlate node properties with structural properties, as we have done with the samples presented in this paper. Another tempting approach might be to use the minimum between-

ness metric to generate explicit clustering hierarchies on the graph by iteratively merging nodes with lowest betweenness. However, to obtain a correct merge order, one would have to recompute the betweenness after each merge step. This then leads to a similar approach as proposed in [NG04], albeit bottom-up instead of top-down. Here, we have used the betweenness metric merely to extract a structural ordering in the nodes, not to compute an explicit decomposition of the graph. A number of important areas for improvement still remain. One involves the visual representation of the arcs we used to indicate inter cluster connections. The visual complexity of the overall view might be reduced if we can group arcs that roughly point in the same direction, for example by using algorithms proposed by [Hol06] or [PXY*05]. The two bottom arcs in Figure 5b might be good candidates for such a reduction for example. Finally, as the betweenness metric is a global graph metric, it is therefore expensive to compute. We have experimented with other more local measures, such as the edge clustering metric proposed by [ACJM03], but these produced worse results. In the future we plan to investigate this further and look at other possible metrics that might produce similar results in less time.

References

[ACJM03] AUBER D., CHIRICOTA Y., JOURDAN F., MELANÇON G.: Multi-scale visualization of small world networks. In *Proceedings of the 2003 IEEE Symposium on Information Visualization* (2003), pp. 75–81.

[Ant71] ANTHONISSE J.: *The rush in a directed graph, Technical Report BN 9/71*. Tech. rep., Stichting Mathematisch Centrum, Amsterdam, 1971.

[BP07] BRANDES U., PICH C.: Centrality estimation in large networks. *International Journal of Bifurcation and Chaos 17*, 7 (2007), 2303–2318.

[Bra01] BRANDES U.: A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology 25*, 2 (2001), 163–177.

[CM03] CHEN C., MORRIS S.: Visualizing evolving networks: Minimum spanning trees versus pathfinder networks. In *Proceedings of the 2003 IEEE Symposium on Information Visualization* (2003), pp. 67–74.

[Fre97] FREEMAN L.: A set of measures of centrality based on betweenness. *Sociometry 40* (1997), 35–41.

[GFC04] GHONIEM M., FEKETE J.-D., CASTAGLIOLA P.: A comparison of the readability of graphs using node-link and matrix-based representations. In *Proceedings of the 2004 IEEE Symposium on Information Visualization* (2004), pp. 17–24.

[HJ05] HACHUL S., JÜNGER M.: An experimental comparison of fast algorithms for drawing large general graphs". In *Proceedings Graph Drawing 2005 (GD'05)* (2005), pp. 235–250.

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 741–748.

[HT74] HOPCROFT J., TARJAN R.: Efficient planarity testing. *Journal of the ACM (JACM) 21*, 4 (1974), 549–568.

[JM96] JÜNGER M., MUTZEL P.: Maximum planar subgraphs and nice embeddings: Practical layout tools. *Algorithmica 162* (1996), 33–59.

[KNJ04] KIM D.-H., NOH J.-D., JEONG H.: Scale-free trees: The skeletons of complex networks. *Phys. Rev. E 70*, 046126 (2004).

[Lie01] LIEBERS A.: Planarizing graphs. *JGAA 5*, 1 (2001), 1–74.

[Lov07] LOVE M.: Genealogy of influence, 2007. http://mike-love.net/, accessed March 2007.

[NG04] NEWMAN M., GIRVAN M.: Finding and evaluating cluster structure in networks. *Phys. Rev. E69 026113* (2004).

[Noa05] NOACK A.: Energy-based clustering of graphs with nonuniform degrees. In *Proceedings Graph Drawing 2005* (2005), pp. 309–320.

[PXY*05] PHAN D., XIAO L., YEH R., HANRAHAN P., WINOGRAD T.: Flow map layout. In *Proceedings of the 2005 IEEE Symposium on Information Visualization* (2005), pp. 219–224.

[SDD89] SCHVANEVELDT R., DURSO F., DEARHOLT D.: Network structures in proximity data. *The psychology of learning and motivation: Advances in research and theory 24* (1989), 249–284.

[vD00] VAN DONGEN S.: *Graph Clustering by Flow Simulation*. PhD thesis, Universiteit Utrecht, 2000.

[vH03] VAN HAM F.: Using multilevel call matrices in large software projects. In *Proceedings of the 2003 IEEE Symposium on Information Visualization* (2003), pp. 227–232.

[vHvW04] VAN HAM F., VAN WIJK J.: Interactive visualization of small world graphs. In *Proceedings of the 2004 IEEE Symposium on Information Visualization* (2004), pp. 199 – 206.

[WH04] WILKINSON D., HUBERMAN B.: A method for finding communities of related genes. *Proceedings of the National Academy of Sciences (PNAS) 101*, 1 (2004), 5241–5248.

[WS98] WATTS D., STROGATZ S.: Collective dynamics of 'small-world' networks. *Nature 393* (1998), 440–442.