DOI: 10.1111/j.1467-8659.2011.02037.x

COMPUTER GRAPHICS *forum*
Volume 30 (2011), number 8 pp. 2314–2327

Social Network Clustering and Visualization using Hierarchical Edge Bundles

Yuntao Jia[1], Michael Garland[2] and John C. Hart[1]

[1]University of Illinois, USA
wdragon521@gmail.com, jch@illinois.edu
[2]NVIDIA Research, USA
mjgarland@acm.org

Abstract

The hierarchical edge bundle (HEB) method generates useful visualizations of dense graphs, such as social networks, but requires a predefined clustering hierarchy, and does not easily benefit from existing straight-line visualization improvements. This paper proposes a new clustering approach that extracts the community structure of a network and organizes it into a hierarchy that is flatter than existing community-based clustering approaches and maps better to HEB visualization. Our method not only discovers communities and generates clusters with better modularization qualities, but also creates a balanced hierarchy that allows HEB visualization of unstructured social networks without predefined hierarchies. Results on several data sets demonstrate that this approach clarifies real-world communication, collaboration and competition network structure and reveals information missed in previous visualizations. We further implemented our techniques into a social network visualization application on facebook.com and let users explore the visualization and community clustering of their own social networks.

Keywords: visualization, network clustering, edge bundles, betweenness centrality

ACM CCS: I.6.9 [Simulation, Modeling, and Visualization]: Visualization–Information visualization.

1. Introduction

Socialnetworks have grown larger, denser and more interconnected. For example, the popular social network Facebook now has 500 million users, each averaging 130 friends [Fac11], which implies a network of 65 billion edges. Visualizing even small portions of such massive networks has become a formidable task. Straight-line graph drawings are cluttered by too many edge crossings to effectively reveal structure, which motivates the investigation of alternative graph drawing techniques. Curve-based or edge-bundle-based graph drawings [DEGM05, PXY*05, Hol06, BD07, CZQ*08] can more effectively present communication in a network by depicting edges as curved pathways that adhere to similar pathways to reveal an underlying graph 'control' structure.

Because the control structure is often much simpler than the graph itself, edge bundling draws curved edge sharing the same part of the control structure near each other to form bundles that drastically reduce the distracting visual clutter of edge crossings. For example, the hierarchical edge bundle (HEB) method [Hol06] bundles graph edges based on a predefined hierarchical clustering of the nodes. After positioning the hierarchy (and nodes) via standard tree layout methods, each edge is drawn as a B-spline curve adhering to the path up and down the hierarchy from one node endpoint to the other.

The effectiveness of these edge bundling approaches relies critically on the hierarchical control structure used to order the nodes and to lay out the curved edges connecting them. The original HEB approach [Hol06] assumes this control structure is provided with the input graph as a node hierarchy that organizes it into a compound graph. Existing methods for automatically building such hierarchies are based on repeated clustering, and the state-of-the-art is represented by small-world clustering based on edge strength

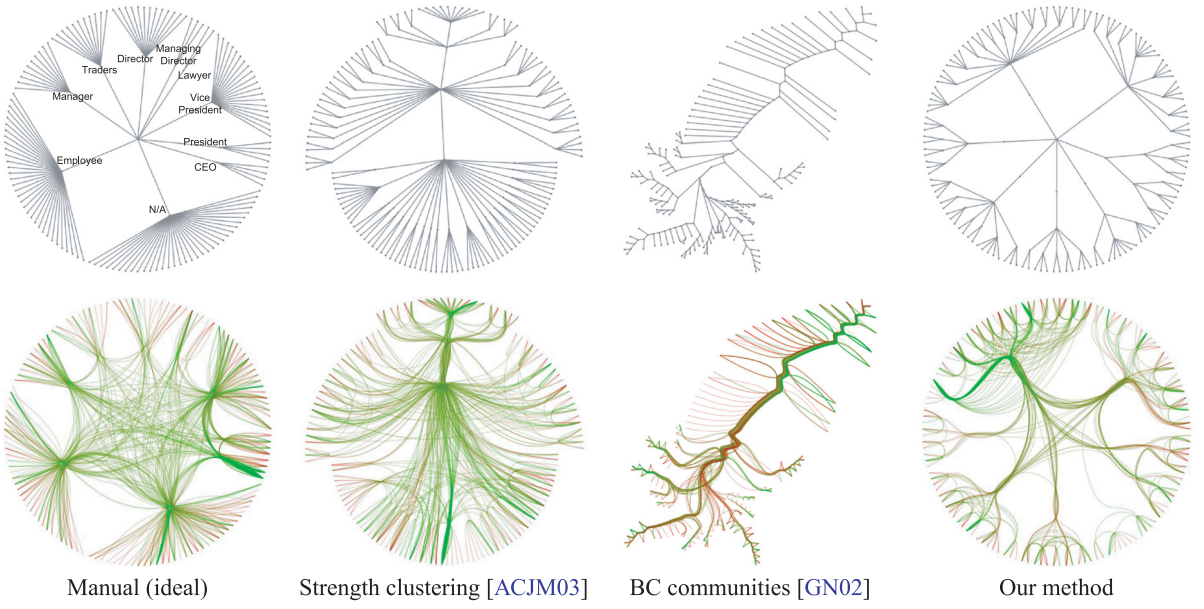


Figure 1: *Comparison between different community hierarchies (top) and their hierarchical edge bundle visualizations (bottom) on the graph 'enron-email-2001.08'.*

[ACJM03] and community clustering based on betweenness centrality (BC) [GN02].

Figure 1 demonstrates HEB visualizations of Enron e-mails using several control structures. The first one is a hierarchy created manually by a user based on the employee status reports [SA04]. Against this, we compare previous automatic hierarchical clustering methods based on edge strength [ACJM03] and low-BC communities [GN02]. The edge strength approach results in large hierarchy fanouts and its community clustering yields many inter-community overlapping edge curves in the bottom HEB visualization. Low-BC clustering detects better community clusters than edge strength but yields an unbalanced binary tree of lists, a 'dendrogram'. Our proposed hierarchical clustering generates more balanced communities designed specifically for effective HEB visualization. Its hierarchies are more evenly distributed, which yields more visually organized edge bundle pathways that keep the curved edges away from the otherwise crowded middle of the HEB disk layout.

Section 3 details this new clustering approach. Like BC communities [GN02], it first computes the BC of every edge in the social network, then removes edges in order of decreasing BC, then reintroduces these edges in community merging steps to form a hierarchy of links between isolated communities. Our new community clustering algorithm builds on this basic approach with (1) modified edge-removal constraints to find a collection of small isolated communities, (2) new community merging rules to yield better organized (flatter, more balanced) hierarchies, and (3) new hierarchy adjustments applied after clustering is completed to further improve the community structure for visualization purpose.

Section 4 demonstrates the results, and Section 5 shows that this new method generates quantitatively better clusters for social networks, with superior modularization qualities (MQ) [MMCG99], when compared to a few existing clustering methods, especially on large graphs. Section 5.2 introduces a relative community strength measure called the 'BC differential' that we use to indicate when community merging is indicated by social structure or a necessary but arbitrary choice needed to maintain tree structure quality. We integrate this new BC differential measure as bundle strength in the resulting HEB visualization.

Section 6 describes interactive implementations, including a Facebook social network visualization application that lets users visualize their own social networks using the proposed new hierarchical community clustering approach for HEB visualization. Section 8 concludes with ideas for future work on larger graphs and time-varying networks.

2. Previous Work

Given a social network, we construct a social community hierarchy based on BC to automate and improve edge bundle visualization. Here we review the background work in each of these areas.

2.1. Graph hierarchies

Graph hierarchies are useful for visualizing large graphs, and are often created by repeatedly clustering nodes into affinity groups [HMM00]. Wu *et al*. [WGH04] hierarchically clustered nodes by their shortest-path distance from hub nodes (chosen by least BC or highest degree) for data mining and visualizing power-law graphs. Kumar and Garland [KG06] clustered nodes based on an 'authority' metric and stratified the graph into different layers that were overlaid for fast layout and interactive visualization.

Graph hierarchies can also be created by filtering. Auber *et al*. [ACJM03] removed weak edges in a small-world graph to visualize a large graph as a hierarchy of strongly connected components. Chiricota *et al*. [CJM03] applied similar ideas based on edge strength to discover component structure in software systems. Both Newman [New04b] and Blondel *et al*. [BGLL08] clustered nodes by maximizing an 'MQ' metric to find communities in social networks.

Our work proposes new rules for filtering edges and clustering nodes into isolated communities and merging these communities into a balanced hierarchy which yields a flatter and more regular layout than these past approaches.

2.2. Betweenness centrality

BC measures how often an edge is found on all shortest paths from any node to any other node. High-BC edges connect large communities, whereas low-BC edges connect individuals within a community.

Girvan and Newman [GN02] removed high-BC edges to reveal community structures in social and biology networks, and this approach was later surveyed and compared with other approaches by Newman [New04a]. van Ham and Wattenberg [vHW08] explored similar ideas to visualize small-world networks. Heer and Boyd [HB05] removed high-BC edges to reveal communities within social networks, whereas Jia *et al*. [JHGH08] removed low-BC edges to reveal the communication pathways within social networks.

We likewise utilize low-BC edges to detect communities, and simplify high-BC edges to accentuate the communication pathways between communities.

2.3. Edge bundle visualization

Edge bundles render large graphs via edge clustering, by collecting together long edges analogous to the way electric wires are merged into bundles along a shared mutual path segment, fanning out at ends to connect distinct endpoints.

Holten [Hol06] proposed HEBs to visualize a compound graph accompanied by a predefined hierarchy. His approach first drew the hierarchy using an existing tree layout method, such as radial layout [Ead92], balloon layout [CK95] or treemap [Shn92]. It then laid out long and complex graph edges using the nodes of the tree as B-spline control points. Each edge in the original graph was modelled as a single B-spline using the control points along the shortest path in the hierarchy from one end node to the other.

Cui *et al*. [CZQ*08] visualized large graphs via edge clustering through a geographical control structure. Balzer and Deussen [BD07] used edge bundles to simplify edges in a clustered level-of-detail graph visualization. Confluent drawing [DEGM05] displayed non-planar node-link diagrams using curved edges, although not all graphs are confluently drawable. Flow map layouts [PXY*05] route edges through a binary cluster hierarchy, although only for single-source graphs.

Our contributions to edge bundle visualization are based on the automation, integration and colouring provided by our proposed community hierarchy approach.

3. Balanced Community Hierarchy Construction

We cluster a social network into a balanced community hierarchy in two steps, illustrated in Figure 2. We first compute the BC of every edge in the graph, and remove edges in non-increasing order of BC to find the smallest communities that collectively form the base of the hierarchy. We then construct the hierarchy by merging these communities according to the increasing BC of the removed edges, and visualize the result with HEB.

3.1. BC edge removal

BC [Fre77] indicates how often a node lies on the shortest and presumably most used communication paths between other nodes

$$\mathrm{BC}(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}, \tag{1}$$

where $\sigma_{s,t}$ counts the number of shortest paths between s and t, and $\sigma_{s,t}(v)$ counts only the ones containing v. BC of an edge is computed similarly

$$\mathrm{BC}(u, v) = \sum_{s,t,u,v \in V} \frac{\sigma_{s,t}(u, v)}{\sigma_{s,t}}, \tag{2}$$

where $\sigma_{s,t}(u, v)$ counts the number of shortest paths between s and t that pass through edge (u, v).

High-BC edges typically connect communities of nodes within which connections are dense but between which connections are loose [GN02]. The bigger the BC, the larger the communities. In contrast, low BC edges usually connect nodes within the same community. In particular, an edge with BC equal to one (the smallest allowed) indicates it connects two nodes clearly in the same community, and these two nodes form the smallest possible community. Similar to the

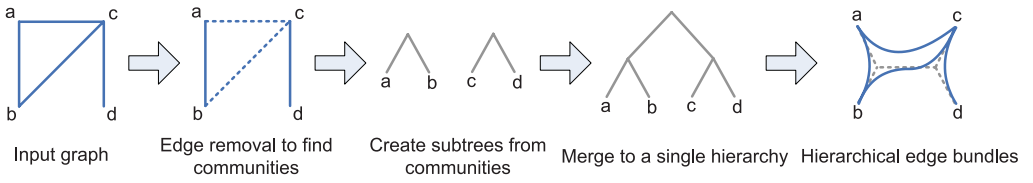
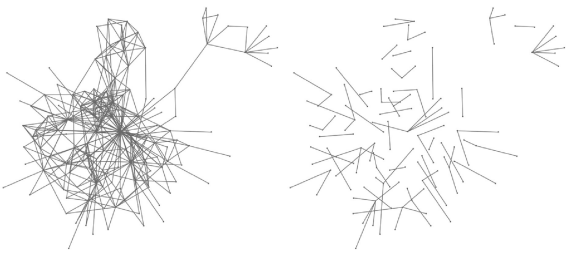


Figure 2: *Work flow of our approach. Different colours are used for graph edges (blue) and hierarchy edges (grey).*



(a) Original graph (b) After edge removal

Figure 3: *Edge removal on graph 'enron-email-2001.08'.*

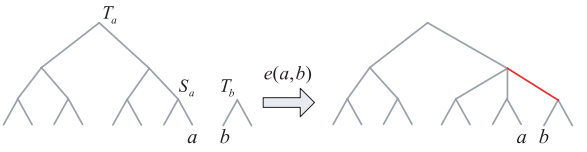


Figure 4: *Merging two communities. Here two communities are connected because node $a \in T_a$ and $b \in T_b$ are connected by an edge e(a, b).*

approach of Girvan and Newman [GN02], we detect communities in a graph by removing edges in descending BC order, leaving small disjoint communities of nodes connected by the remaining edges. The important difference is that their method continues until all edges are removed whereas our method terminates with a collection of isolated communities. We achieve this by implementing the following edge removal constraints.

3.1.1. *Definition 1 (edge removal constraints)*

An edge can be removed if and only if neither its BC, nor the degree of either of its vertices, equals one.

In other words, at the end of the edge removal phase, an edge either has unit BC or has degree-one end node(s). Edges with unit BC connect nodes within the smallest communities. Edges with degree-one end node(s) are the least-BC edges for those nodes, which further indicate relatively the smallest communities to which those nodes belong. By enforcing these two constraints in edge removal, we converge on the smallest communities for all nodes simultaneously. Figure 3 shows the edge removal result on the Enron email network. Each connected component in Figure 3(b) represents the smallest communality relative to nodes within it.

Removed edges are placed on a stack so they can be later reintroduced by a merging phase in last-in-first-out order. Please note that BC is not recomputed during edge removal, as discussed in Section 6.1.

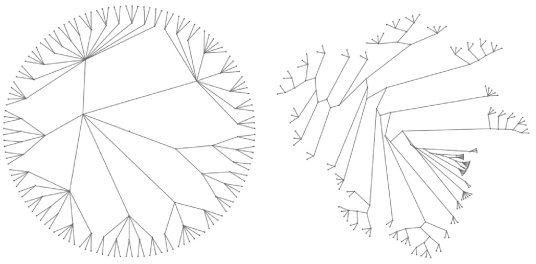
3.2. Merging communities

The numerous small communities that result from edge removal are detected by a simple connected component sweep. These small components are converted into a forest of small subtrees to form the bottom of the hierarchy, illustrated in the third step of Figure 2. Subtrees are then merged according to the previously removed edges in order of increasing BC. If a removed edge connected a pair of nodes from two subtrees, then those two subtrees belong to a bigger community in the original graph. Because removed edges are pushed onto a stack in non-increasing BC order, popping these edges from the stack produces non-descending BC edges that merge communities in the correct order. Our new method to merge two subtrees works as follows.

3.2.1. *Definition 2 (community merging rules)*

Let removed edge $e(a, b)$ connect two graph nodes a and b belonging to two different communities, represented in the hierarchy by subtrees T_a and T_b. If T_a and T_b share the same height, then they can be merged as children of the same new parent tree node. Otherwise assume without loss of generality that T_a is taller than T_b. Let S_a be the unique (lowest) subtree of T_a that contains a and shares the same height as T_b. Then the communities are merged by assigning the parent of S_a as the parent of T_b.

If two subtrees have the same height, we simply merge them by joining their roots. Otherwise, the two nodes are in two communities (subtrees) of different sizes (heights). We thus seek the smallest common community while retaining community tree balance. In the example shown in Figure 4, the smallest common community for a and b is the union of S_a and T_b. This subtree merging rule



(a) With Definition 2 (b) Without Definition 2

Figure 5: *Creating hierarchies by merging communities in graph 'enron-email-2001.08'.*

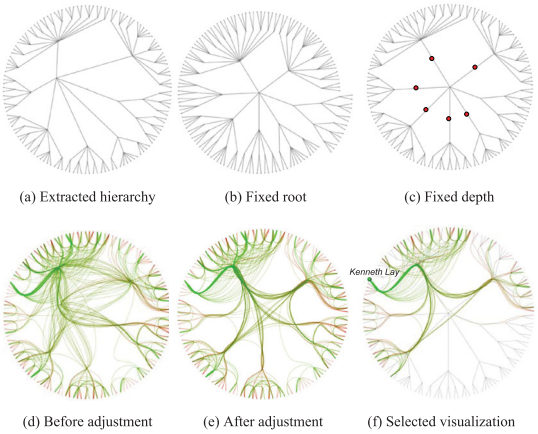


Figure 6: *Root adjustments (b and c) of the community hierarchy (a) of graph 'enron-email-2001.08' to improve HEB visualization from (d) to (e).*

differentiates our method significantly from previous approaches [GN02, New04b], demonstrated in Figure 5 with results on the Enron email network. In the previous approaches, shown in Figure 5(b), every merge increases the height of the result, and can produce a tree of worst-case $O(n)$ height for n nodes, whereas the worst case height under our rules is $O(\log n)$ by induction. If there are two nodes, the height is one. If there is a subtree of height k, then by inductive hypothesis it has $O(2^k)$ nodes. Under the merging rule, its height would not increase unless there is another tree of at least that height. Even if there was, the merged tree would then have height $k+1$ and consist of $O(2^{k+1})$ nodes.

These new merging rules make more sense for both community detection and HEB visualization. For community detection, a reintroduced graph edge (a, b) defines a common community by joining two communities that contain a and b. However, node a may belong to several communities, represented as nested subsets by an ancestry path in the community hierarchy. We must select one of these communities of a to merge with the community of b, so we merge the largest communities of both that share the same level in the hierarchy with the assumption that they represent the same kind of community. This represents the smallest common community of a and b.

With our method, the resulting hierarchy tends to be balanced, whereas previous approaches, including Girvan and Newman [GN02], can merge a small community with a large community, leading to the rather jagged, lopsided hierarchies shown in Figure 1.

For HEBs, a Girvan-Newman hierarchy routes communication from one community to another through a shared parent (the current root during the tree merging process). Sending all communications that far up and then back down unnecessarily increases the edge density in the interior of the radial layout of the HEB visualization, as shown in Figure 1. Merging subtrees at lower heights yields simpler and shorter communication pathways in the HEB visualization, leaving the interior to highlight more significant cross-community communications.

3.3. Hierarchy root adjustment

Radial layout [Ead92] is commonly used to lay out trees and HEBs [Hol06]. Such layouts work best for trees of uniform depth, otherwise multiple radii are needed to position the leaves. For large depth difference, the layout could be skewed, as demonstrated by the third hierarchy in Figure 1. Although our community hierarchy construction rules strive for uniform depth, they do not always achieve it. Furthermore, the root of the tree might not represent the mass centre of the graph nodes at its leaves. In severe cases, this can lead to a lopsided drawing and an off-centre HEB visualization, as shown in image (a) and (d) of Figure 6, respectively.

To overcome these problems, only two simple adjustments are needed in the preparation of the community hierarchy for radial HEB visualization. First, different root node choices are explored to re-centre the radial tree layout so that no child of the root node represents more than half of the original graph nodes (or more than half-plus-epsilon for a binary tree). If a root's child is found to represent too strong a majority of nodes, then it is set to be the new root and the hierarchy is retested, as shown in image (b) of Figure 6.

The radial layout of HEB visualization works best when the leaf nodes on its perimeter all share the same depth. For community hierarchies of varying depth (e.g. Figure 6b), we can improve the appearance of the HEB layout by inserting 'dummy' nodes via a bottom-up depth-balancing process. Figure 6(c) shows the hierarchy after the adjustment, with all of the inserted 'dummy' nodes marked using red dots.

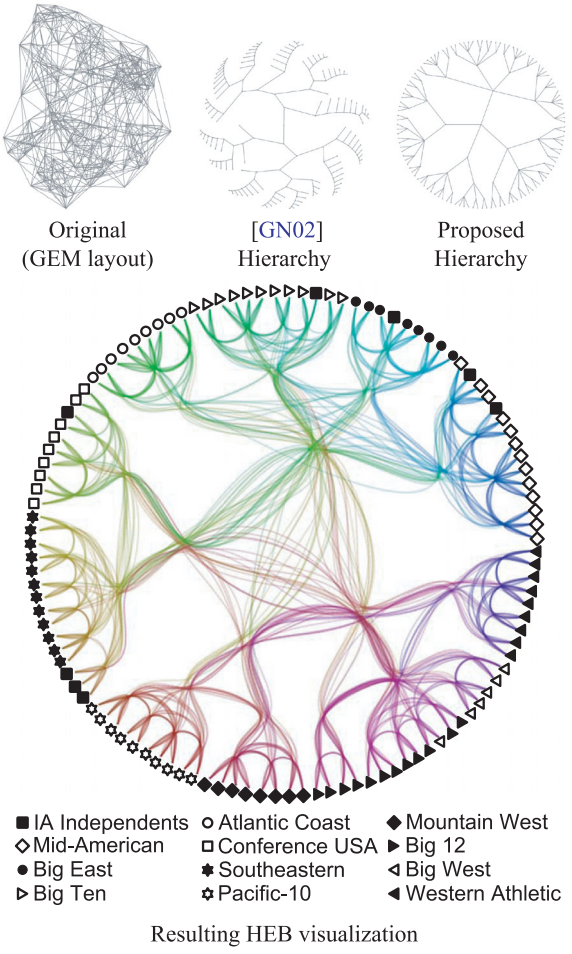


Figure 7: *HEB visualization of graph 'college-football-2000' resulting from the community hierarchy discovered by our proposed automatic method.*

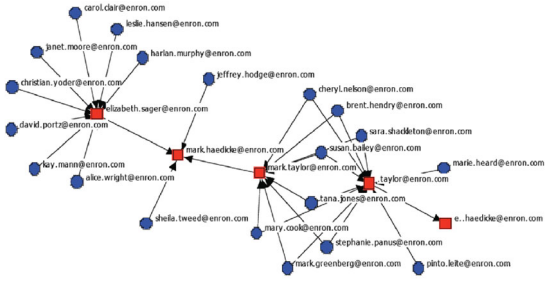
4. Results

We have tested our algorithm with several real data sets to demonstrate its effectiveness at both community detection and visualization. Unless explicitly stated otherwise, we use the GEM method [FLM95] to lay out the original graph and the radial method [Ead92] to lay out the hierarchy and edge bundles. For directed graphs, we follow the colouring scheme by Holten [Hol06] which colours *source* nodes with green and *target* nodes with red. For undirected graphs, we colour nodes in a hue colour mapping based on their polar angles in the radial layout. In both cases, the colour is linearly interpolated along the B-spline that is used to visualize the edge.

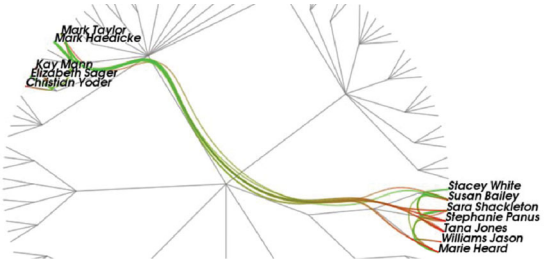
Figure 7 visualizes the undirected graph 'college-football-2000', which represents 616 matches between 115 Division IA college football teams during regular season fall 2000 [GN02]. What makes this data set interesting is that its community structure is known [NCA00]. In particular, those teams belong to 11 conferences except a few independent teams that do not belong to any conference. Games were played more frequently between teams in the same conference. The hierarchy constructed by our approach discovers these conferences and clusters teams in the same conference as siblings to each other. The independent teams are placed in conferences they played more with. This confirms that our method correctly clusters the communities within this graph. Girvan and Newman [GN02] also captures the team conferences, but their unbalanced hierarchy poorly describes relations between teams in the same conference and does not fit well with the hierarchial edge bundle visualization.

Figure 6 visualizes the directed graph 'enron-email-2001.08', which represents 389 emails between 132 Enron employees in August 2001, extracted from the cleansed Enron email data [SA05]. Green represents senders and red represents recipients. We only considered the 'TO' recipients and ignored the 'CC' or 'BCC' recipients. When compared to their straight line drawing counterparts (i.e. Figure 3a), HEBs more effectively convey different communities and the communications between them. Relevant employees are user labelled and emails between them are visualized by interactive user selections. For example, Kenneth Lay was named the CEO of Enron during this data set's time interval. Selecting his node, as shown in image (f) of Figure 6, reveals many emails sent from him to employees in various groups. The SocialRank project [MDPP08], shown in Figure 8(a), serves as a partial ground truth organizational structure, resembles the community structure found by our method, as highlighted in Figure 8(b). In particular, the organization structure indicates that both Kay Mann (kay.mann@enron.com) and Christian Yoder (christian.yoder@enron.com) worked with Elizabeth Sager (elizabeth.sager@enron.com). Our method clusters them into the same community shown on the left. Similarly, Susan Bailey (susan.bailey@enron.com), Sara Shackleton (susan.shackleton@enron.com), Stephanie Panus (stephanie.panus@enron.com), Tana Jones (tana.jones@enron.com) and Marie Heard (marie.heard@enron.com) are correctly clustered together, as all of them worked with Mark Tylor (mark.tylor@enron.com).

Figure 9 visualizes the undirected graph 'myfacebook' which represents the network between the first author's 165 facebook friends. By the nature of social networks, the graph is highly connected with 1803 edges. As shown in Figure 9(c), our method clusters this friends network into different groups, including friends in the subject's academic department, and the method automatically discovers further sub-categories including lab mates, previous college alumni, seniors, juniors, a registered student organization colleagues, soccer teammates and fellow interns.

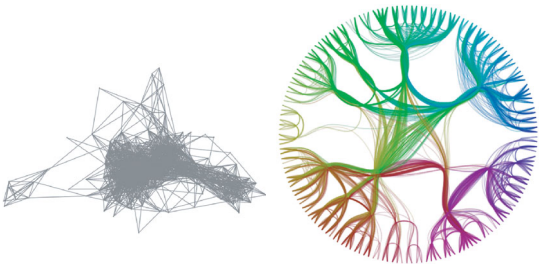


(a) Partial Enron organization structure

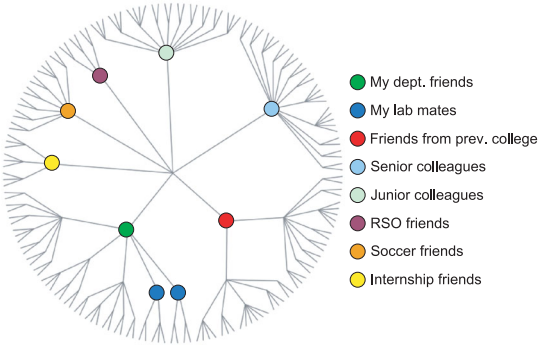


(b) Our clustering result

Figure 8: *The organization ground truth (top) of Enron employees during January 2000 and November 2001. Similar groups (bottom) are discovered with our community clustering method.*



(a) Original graph (b) Edge Bundles



(c) Generated Hierarchy (with user supplied labels)

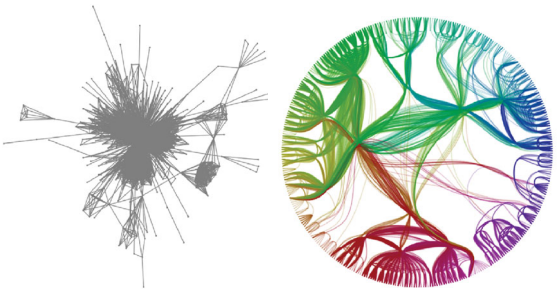
Figure 9: *Graph 'myfacebook', including user supplied labels of groups discovered by our community clustering method.*

The HEB visualization is shown in Figure 9(b) and leads to some discovery. First, the internship friends do not know other friends because they are from different colleges, with a few exceptions. Secondly, there are many connections between current department friends and friends from the previous college. This is simply because after graduation they joined the same college and same department as the author.

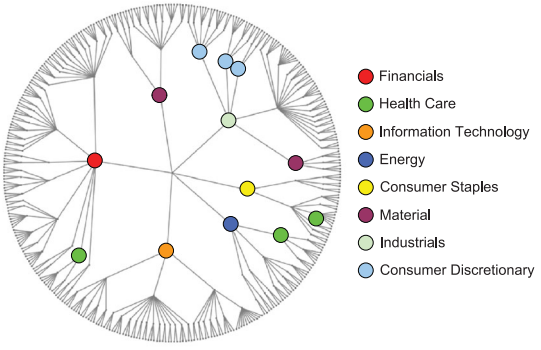
Figure 10 visualizes the undirected graph 'sp500-38', which represents 3206 cross correlations of price fluctuation of 365 stocks from the S&P 500. Our method is able to recognize different stock sectors and clusters them near to each other in the hierarchy. The HEB visualization reveals that financial stocks affect all other kind of stocks except energy, consumer staples, and health stocks, which are relatively independent. For example, there are many edge bundles connecting to the financial stocks on the left of the circle, but very few of them connecting to health and energy stocks on the bottom right.

Figure 11 visualizes a bipartite human disorder-gene network [GCV*07] 'diseasome', which represents 1550 associations between 1419 disorders and genes. A disorder is connected to a gene if the disorder arises from the mutation of the gene. Figure 11(a) shows a visualization of the network published in the New York Times [Pol08] and interactively [BC08], in which squares correspond to genes and circles correspond to disorders, coloured by disorder types and sized by the number of gene links. We visualization this network using HEBs as follows. We first repeatedly cluster to form a community hierarchy, shown in Figure 11(b). This hierarchy is then laid out using an adaptive tree drawing technique [JH08], shown in Figure 11(d). All first level internal nodes are drawn on different layout circles to emphasize the communities they represent. A few disorders are also labelled to show that the same type of disorders is likely clustered into the same community. Finally, all the disorder–gene associations are drawn on top of the hierarchy as edge bundles, which is shown in Figure 11(c). There are seldom connections between communities whereas disorders and genes within the same community are more strongly associated with each other. One benefit of the visualization is that the layout differentiates communities better with well separated node clusters without overlapping or interleaving. In addition, it is easy to tell large and dense communities from small, sparse ones.

The steps in our method have linear complexity with two exceptions. One superlinear step is the BC computation which can be performed for n nodes and m edges in $O(nm)$



(a) Original graph (b) Edge Bundles



(c) Generated Hierarchy (with user supplied labels)

Figure 10: *Graph 'S&P500', including user supplied labels of stock sectors discovered by our community clustering method.*

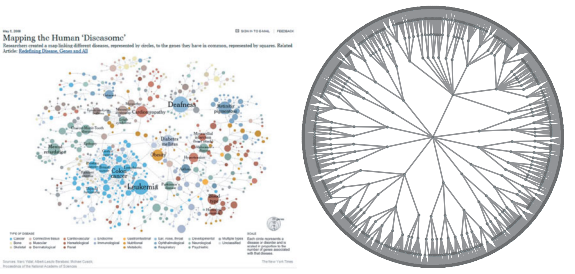
time for unweighted graphs and $O(nm + n^2 \log n)$ time for weighted graphs [Bra01]. The other superlinear step sorts the edges in non-increasing order of BC before edge removal, which requires $m\log(m)$ running time. All presented layouts and drawings were computed in a matter of seconds for the graphs shown.

5. Discussion

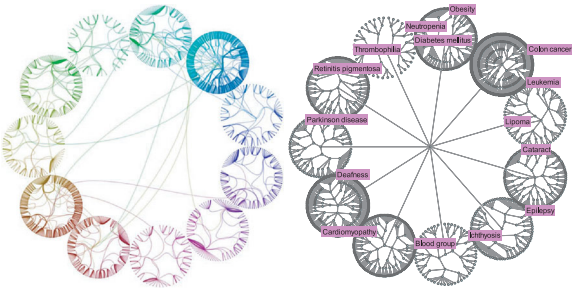
The results show our proposed HEB layouts to be an effective approach based on qualitative visual comparison. Here we evaluate the layouts based on quantitative measurements.

5.1. Comparing cluster quality

To further justify our clustering method, we measured the modularization quality (MQ) of communities found in the network and compared it against results from other methods. MQ was first introduced as a partition cost function in the field of software reverse engineering [MMCG99], but has also been applied to measure cluster qualities for small-world networks [ACJM03]. Given a set of clusters $C = \{C_1, C_2, \ldots, C_p\}$ of a graph G, MQ measures the average edge density within clusters versus the average edge density between clusters as



(a) "diseasome" (b) Generated hierarchy



(c) Edge bundles (d) Adaptive hierarchy drawing

Figure 11: *Visualizations of a human disorder-gene network 'diseasome'. Labels in (d) are aligned on the right side of the corresponding nodes.*

$$\mathrm{MQ}\{C;G\} = \frac{1}{p}\sum_{i=1}^{p} s(C_i, C_i) - \frac{1}{p(p-1)/2}\sum_{i<j} s(C_i, C_j), \quad (3)$$

where $s(C_i, C_j)$ measures the edge density between nodes in C_i and C_j as

$$s(C_i, C_j) = \frac{|e(C_i, C_j)|}{|C_i||C_j|}, \quad (4)$$

where $|e(C_i, C_j)|$ denotes the number of edges connecting a node in C_i to a node in C_j and $|C_i|$ represents the number of nodes within C_i. Hence a larger MQ should indicate a better clustering.

When creating a hierarchy, the graph is divided into a unique number of clusters at each step. To measure the quality of these clusters, we measure and plot the MQ at each step to show the change of MQ versus the number of clusters.

Figure 12 shows the MQ measured on networks 'enron-email-2001.08' and 'myfacebook' clustered by four methods: 'strength clustering' [ACJM03], 'BC communities'

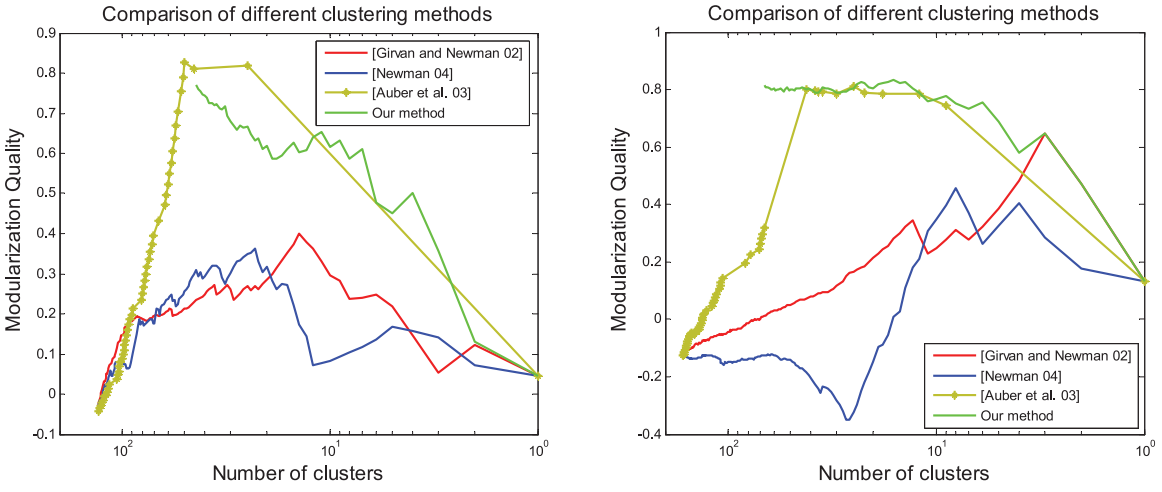


Figure 12: *Clustering quality comparison on networks 'enron-email-2001.08' (left) and 'myfacebook' (right).*

[GN02], 'fast communities' [New04b] and our method. The three previous methods begin with every node in its own cluster, whereas our method starts with nodes in small community clusters. Hence, our method begins with 44 instead of 132 clusters for the Enron network and 67 instead of 165 clusters for the facebook example. Furthermore, 'strength clustering' is a top down approach that repeatedly divides clusters into sub-clusters, but does not generate all possible numbers of clusters. We marked the available cluster numbers with '∗'s.

Figure 12 (left) shows that for the same number of clusters, that is from 44 to 2, our method achieves higher MQ than does either of Newman's methods. One possible reason is that their methods start by picking one small initial cluster and iteratively merge neighbouring clusters connected to it by low BC edges. Thus it prioritizes merging of well-connected clusters which might not be optimal for the entire graph. While our method finds the smallest clusters for all the nodes simultaneously, and obtains better clusters overall. "Strength clustering" produces higher MQ than our method at 25 clusters, because their method varies the number of clusters to maximize MQ. Eventually, all methods converge on a single cluster with the same MQ. The facebook network (right) is much denser than the Enron network (left). The facebook results show our method achieves the highest MQ ranging from 66 to 3 clusters. Both plots indicate that our method generates high quality, densely connected clusters themselves only loosely connected with each other.

A clustering method with higher MQ could potentially benefit the quality of HEB created using a clustering hierarchy. A higher MQ indicates more intra-cluster edges and fewer inter-cluster edges. In HEB, intra-cluster edges are drawn as *short* B-splines, whereas inter-cluster edges are drawn as *long* B-splines. In comparison to short edge-bundle

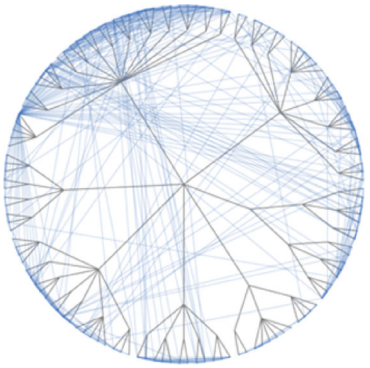


Figure 13: *HEB with edges drawn as straight B-splines.*

B-splines, long edge-bundle B-splines route through higher level hierarchy nodes, and tend to overlap more. A clustering method with higher MQ would generate an HEB with more short B-splines and fewer long ones. As a result, such an HEB would exhibit fewer edge bundle crossings.

Figure 13 shows an HEB visualization of 'enron-email-2001.08'. Grey lines represent the clustering hierarchy while blue lines represent the edge bundles. Here the bundle 'strength' is set to zero so that edges no longer bundle but follow independent straight lines. This figure shows there are only a few long blue lines across the circle centre, which correspond to inter-cluster edges, because our clustering method achieves a high modularization quality.

A 'modularity' measure [NG04] indicates the quality of communities discovered in networks. It sums over each community the difference between observed connectivity and the

expected value of a random graph with the same degree distribution [GdMC10]. For the Enron network, 'fast communities' obtained the best modularity of 0.481, compared to 0.387 obtained by our method, because 'fast communities' clusters the graph by maximizing modularity. Recent studies [FB07, GdMC10] argue that the behavior and accuracy of maximizing modularity is not well understood and modularity itself can exhibit degeneracy. Empirically we found that clustering by high modularity may not produce good HEB visualizations. For example, 'BC communities' obtains a modularity of 0.475 but yields overlap of many B-splines through the centre branch of the hierarchy as shown in the third column of Figure 1.

5.2. Measuring community strength

Communities are detected based on edge BC and low BC edges connect nodes in small communities. The BC of intra-community edges is lower than that of inter-community edges. A community is strong if this margin is large. In other words, the strength of a community can be measured by the BC difference between intra-community edges and inter-community edges. We call this the 'BC differential' and we use it to indicate for each community hierarchy node whether the choice of which two communities to merge was obvious or arbitrary.

Let C be one of the communities at some non-root, non-leaf level in the community hierarchy, representing a set of nodes. Let $p(C)$ indicate the parent community of C, and let C_i indicate its ith child community. We denote by $e(C_i, C_j)$ a portion of the set of 'intra-community' edges of C connecting nodes in sub-community C_i to nodes in C_j, and the number of these edges is denoted by set cardinality $|e(C_i, C_j)|$. We can hence measure the intra-community BC of a cluster C as the total BC of edges between child clusters of C,

$$\underline{\mathrm{BC}}(C) = \frac{\sum_{i \neq j} \mathrm{BC}(e(C_i, C_j))}{\sum_{i \neq j} |e(C_i, C_j)|}, \tag{5}$$

normalized by the number of edges over which the BC was measured. We can similarly define the inter-community BC of a cluster C as the average BC of the edges connecting a node in C to a node in a sibling cluster

$$\overline{\mathrm{BC}}(C) = \frac{\sum_{i:C \neq p(C)_i} \mathrm{BC}(e(C, p(C)_i))}{\sum_{i:C \neq p(C)_i} |e(C, p(C)_i)|}. \tag{6}$$

We can then define the BC differential as the normalized BC difference between inter-community edges and intra-community edges

$$D_{\mathrm{BC}}(C) = \overline{\mathrm{BC}}(C) - \underline{\mathrm{BC}}(C). \tag{7}$$

Figure 14(a) shows the BC-differential measured on the graph 'college-football-2000' and visualized on the hierarchy with pseudo colour mapping. Nodes with high-BC difference indicate strong communities, which coincide with the conferences of those teams. Smaller BC difference nodes indicate weaker, somewhat arbitrary, choices for communities because their intra-community edges and inter-community edges have smaller difference in BC.

Besides measuring the strength of communities, we use the BC differential to control the HEB drawing. Instead of having a single edge bundle strength [Hol06] applied to all the nodes, we set strength for each node individually proportional to its BC difference. A node with high-BC difference represents a strong community and we increase the tension of B-splines through that control point to make the edge bundles more compact near that node.

Figure 14(b) demonstrates the rendering of the BC-differential metric for community strength, depicted by the strength of B-spline control points in the HEB. We marked two nodes with dashed rectangles. One of them has much higher BC difference than the other, and its edge bundles are more compact in the visualization to illustrate that it is a strong community.

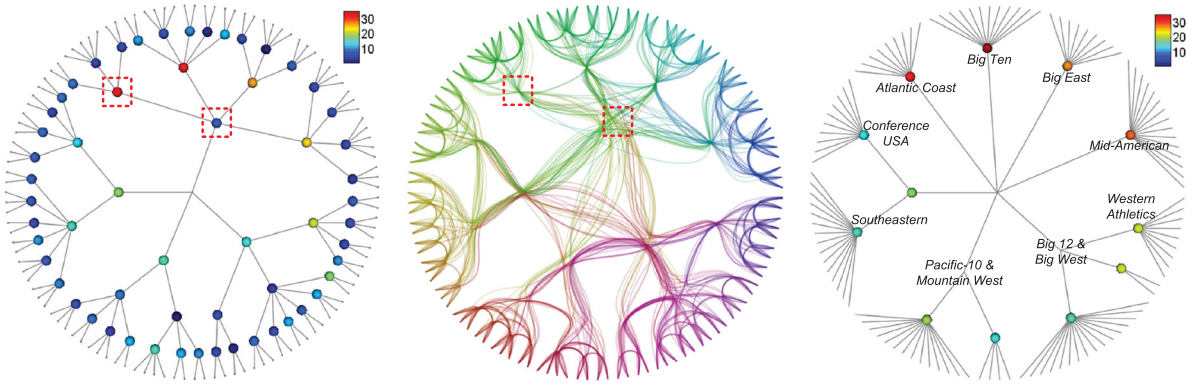
This information could further be used to discover more accurate community information from the graph. In particular, nodes in the hierarchy are removed if their BC differences are lower than a threshold. One result is shown in Figure 14(c) with a threshold of 10 out of the range [0.7, 30.7]. Only strong communities are reserved here and they more closely match to the conferences of those teams. Finding a good threshold requires some experiments, and can be expedited by plotting the histogram of the BC differential.

6. Interaction

The HEB layout tool can be deployed as an offline graph layout process, but here we discuss how it can be implemented more rapidly for interactive display of dynamic graphs, to allow user correction to the automated community clustering, and our deployment of the layout as a facebook application.

6.1. Accuracy versus speed

Users also have control over the tradeoff between accuracy and speed in our clustering algorithm. In particular, when removing edges, BC is only computed once to determine the order of all edges, which saves computation time. If accuracy is more important, then BC can be recomputed after removing each edge, as done in the approach by Girvan and Newman [GN02]. For 'college-football-2000', in the static BC case, the Big 12 and Big West conference are merged into a bigger community, as shown in the first image in Figure 16. By recomputing a dynamic BC after each edge is removed, those two conferences end up well separated, as shown in the first image in Figure 15. The hierarchy is more accurate and similar to the community ground truth, which is shown



(a) BC differential (b) Bundle strength set to BC differential (c) Remove low BC differential nodes

Figure 14: *Measuring strength of communities with BC differential on the graph 'college-football-2000'.*

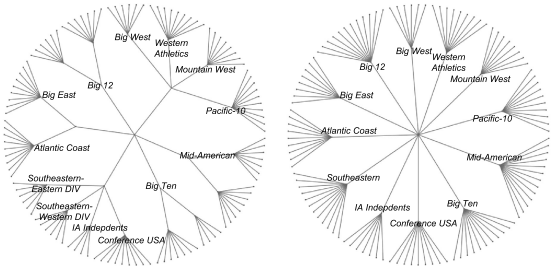


Figure 15: *Community hierarchy constructed with recomputing BC (left) compared to the community ground truth (right) of the graph 'college-football-2000'.*

in the second image. However this incurs the additional time complexity of a factor of m overall.

6.2. User interactions

We have implemented our algorithm in an interactive visualization tool. Users can modify the hierarchy based on their knowledge of the data through two operations: moving and swapping. The moving operation allows users to change the parent of a chosen node whereas the swapping operation allows users to swap two nodes in the hierarchy, with both operations affecting the descendants of the selected node. An example on the graph 'college-football-2000' is shown in Figure 16. The hierarchy constructed automatically by our method intermixed several teams from the Big 12 and Big West conferences, which can be interactively corrected as shown.

Users can also filter displayed edge bundles by selecting or deselecting graph nodes. For the graph 'college-football-2000', a user can perform queries such as 'display only the Illinois' games'. The result is shown in the left image of Figure 17. Users can also select a community of nodes by selecting the corresponding parent node in the hierarchy.

Our tool can also takes advantage of standard HEB visualization features. For example, users can change the strength and transparency of edge bundles, and apply different colour mappings. An example on the graph 'college-football-2000' is shown in the right image of Figure 17, where the strength is changed to 0.65.

7. Visualization Application on Facebook

We have also implemented our clustering method and HEBs in a visualization application on facebook.com, namely 'Friend Insight' [JTH09]. The facebook application allows users to visualize their network of friends, and is written in Flash based on the existing data visualization package 'flare' [Vis09]. A snapshot is shown in Figure 18, anonymized with random names. Users can visualize their networks with our HEB layout, which automatically collects their friends into communities the user might not have previously considered. Users can pan and zoom to inspect the details of the visualization. To facilitate easier visualization of inter-community connections, the user can change the edge bundle strength and can filter the visualization to a single node's connections by either searching for a name or clicking on a friend.

8. Conclusion

We proposed a new social network clustering method, based on BC, that clusters a social network into a community hierarchy. This new method generates a balanced hierarchy that improves HEB graph visualization. We also assessed the strength of the discovered communities, to help users understand the community structure residing in graphs.

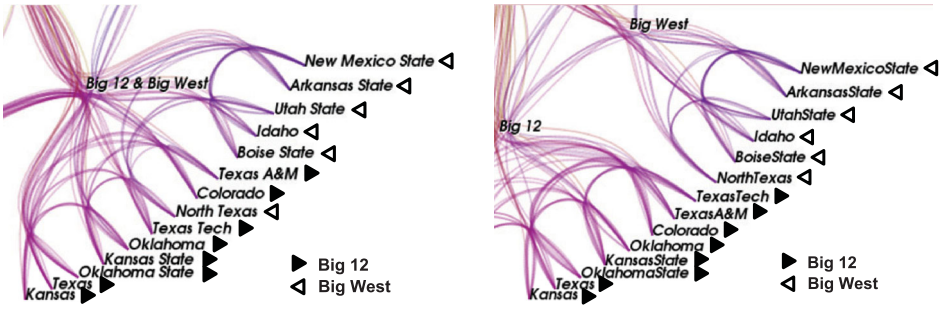


Figure 16: *An interactive user modification of the community hierarchy of graph 'college-football-2000' to separate two conferences.*

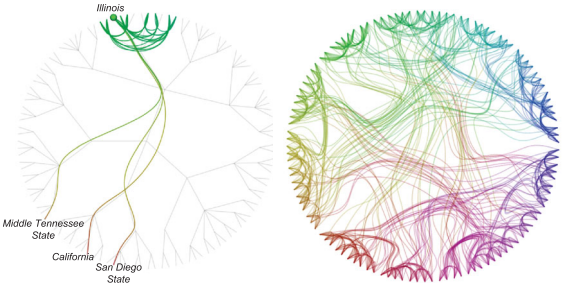


Figure 17: *Manipulating a hierarchical edge bundle visualization of graph 'college-football-2000'.*

We implemented this method as part of an interactive graph visualization tool, on top of the Tulip library and VTK, where users can modify and manipulate visualizations to facilitate knowledge discovery. We also implemented it into a visualization application on facebook to allow potential users to explore their own personal social networks.

This approach has a few limitations. When applying it to larger graphs, the radial layout spaces nodes too closely, and the B-spline paths become unusable, necessitating other tree drawing methods, for example balloon or treemap layouts. Another scalability concern is the superlinear computation time of BC, sometimes addressed by approximation [BKMM07] or parallelism [MEJ*09, JLH*11]. The application of this approach to time-varying or real-time data, such as real time messages or email within a social network, opens up interesting problems of making a dynamic hierarchy that maintains its meaning over time.

We would like to improve our facebook application. Currently, it rasterizes each B-spline to dozens of short line segments for smooth Flash rendering which can become slow for many line segments. It does not support bundle-centric interactions, such as selecting crossing edge bundles. A user study might reveal further strengths and weaknesses of this approach.

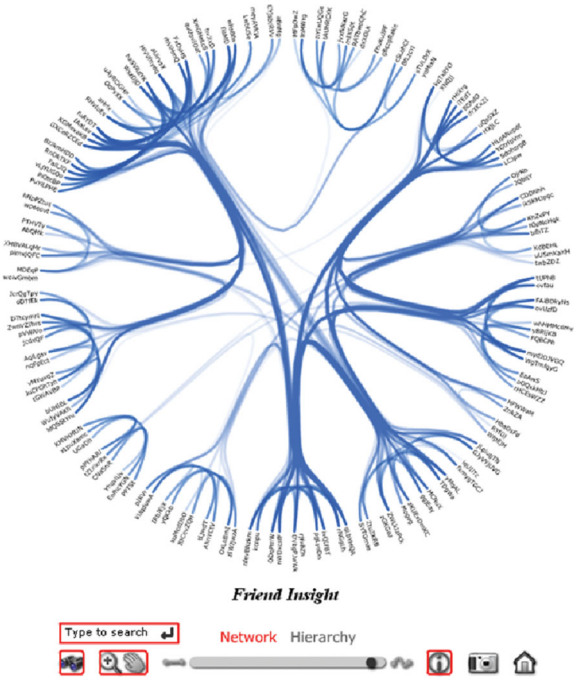


Figure 18: *Our facebook application 'Friend Insight'.*

Acknowledgements

This project was supported by NSF grant IIS-0534485 and NVIDIA Research. Thanks to Victor Lu for the video demo, Harish Tella for implementing the facebook application, and the reviewers for their valuable comments.

References

[ACJM03] Auber D., Chiricota Y., Jourdan F., Melançon G.: Multiscale visualization of small world networks. In *Proceedings of the Ninth Annual IEEE Conference on Information Visualization* (Seattle, WA, USA, 2003), pp. 75–81.

[BC08] Bloch M., Corum J.: Mapping the human diseasome. tinyurl.com/4jqvty (5 May 2008).

[BD07] Balzer M., Deussen O.: Level-of-detail visualization of clustered graph layouts. In *Proceedings of the Asia-Pacific Symposium on Visualization* (Sydney, Australia, 2007), pp. 133–140.

[BGLL08] Blondel V. D., Guillaume J.-L., Lambiotte R., Lefebvre E.: Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment 2008*, 10 (2008), P10008.

[BKMM07] Bader D. A., Kintali S., Madduri K., Mihail M.: Approximating betweenness centrality. In *Proceedings of the 5th Workshop on Algorithms and Models for the Web-Graph (WAW2007)* (San Diego, CA, USA, December 2007).

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

[CJM03] Chiricota Y., Jourdan F., Melançon G.: Software components capture using graph clustering. In *Proceedings of the 11th IEEE International Workshop on Program Comprehension* (Portland, OR, USA, 2003), pp. 217–226.

[CK95] Carriere J., Kazman R.: Research report: Interacting with huge hierarchies: beyond cone trees. In *Proceedings of the 1995 IEEE Symposium on Information Visualization* (Atlanta, GA, USA, 1995), pp. 74–81.

[CZQ*08] Cui W., Zhou H., Qu H., Wong P. C., Li X.: Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1277–1284.

[DEGM05] Dickerson M., Eppstein D., Goodrich M. T., Meng J.: Confluent drawings: Visualizing non-planar diagrams in a planar way. *Journal of Graph Algorithms and Applications* 9, 1 (2005), 31–52.

[Ead92] Eades P.: Drawing free trees. *Bulletin of the Institute of Combinatorics and its Applications 5* (1992), 10–36.

[Fac11] Facebook: Statitics. http://www.facebook.com/press/info.php?statistics (last visited on April, 2011), April 2011.

[FB07] Fortunato S., Barthélemy M.: Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 104, 1 (2007), 36–41.

[FLM95] Frick A., Ludwig A., Mehldau H.: A fast adaptive layout algorithm for undirected graphs. In *Proceedings of the DIMACS International Workshop on Graph Drawing* (Princeton, NJ, USA, 1995), pp. 388–403.

[Fre77] Freeman L. C.: A set of measures of centrality based upon betweenness. *Sociometry* 40, 1 (1977), 35–41.

[GCV*07] Goh K. I., Cusick M. E., Valle D., Childs B., Vidal M., Barabási A.-L.: The human disease network. *Proceedings of the National Academy of Sciences 104*, 21 (May 2007), 8685–8690.

[GdMC10] Good B. H., de Montjoye Y. A., Clauset A.: Performance of modularity maximization in practical contexts. *Physical Review E* 81, 4 (2010), 046106.

[GN02] Girvan M., Newman M.: Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America 99*, 12 (June 2002), 7821–7826.

[HB05] Heer J., Boyd D.: Vizster: Visualizing online social networks. In *Proceedings of the 2005 IEEE Symposium on Information Visualization* (Minneapolis, MN, USA, 2005), p. 5.

[HMM00] Herman I., Melançon G., Marshall M. S.: Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics* 6, 1 (2000), 24–43.

[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.

[JH08] Jia Y., Hart J. C.: *Drawing Trees: How Many Circles to Use?* Tech. Rep. hdl.handle.net/2142/14068, University of Illinois, Department of Computer Science, 2008.

[JHGH08] Jia Y., Hoberock J., Garland M., Hart J. C.: On the visualization of social and other scale-free networks. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1285–1292.

[JLH*11] Jia Y., Lu V., Hoberock J., Garland M., Hart J. C.: Edge v. node parallelism for graph centrality metrics. In *GPU Computing Gems 2*, Hwu W.-M. (Ed.). Morgan Kaufmann (2011), pp. 15–30.

[JTH09] Jia Y., Tella H., Hart J. C.: Friend Insight (facebook application). http://apps.facebook.com/friendinsight (last visited on April 2011), 2009.

[KG06] Kumar G., Garland M.: Visual exploration of complex time-varying graphs. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 805–812.

[MDPP08] Montemayor J., Diehl C., Pekala M., Patrone D.: Socialrank: An ego- and time-centric workflow for

relationship identification. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology, Interactive Poster* (Columbus, OH, USA, 2008).

[MEJ*09] Madduri K., Ediger D., Jiang K., Bader D. A., Chavarria-Miranda D.: A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing* (Rome, Italy, 2009), pp. 1–8.

[MMCG99] Mancoridis S., Mitchell B. S., Chen Y., Gansner E. R.: Bunch: A clustering tool for the recovery and maintenance of software system structures. In *Proceedings of the IEEE International Conference on Software Maintenance* (Washington, DC, USA, 1999), IEEE Computer Society, pp. 50–59.

[NCA00] NCAA: Team listing, division I football statistics. http://web1.ncaa.org/d1mfb/mainpage.jsp?year=2000 (last visited on April, 2011), 2000.

[New04a] Newman M.: Detecting community structure in networks. *The European Physical Journal B – Condensed Matter 38*, 2 (March 2004), 321–330.

[New04b] Newman M.: Fast algorithm for detecting community structure in networks. *Physical Review E* 69, 6 (2004), 066133.

[NG04] Newman M., Girvan M.: Finding and evaluating community structure in networks. *Physical Review E 69*, 2 (Feb 2004), 026113.

[Pol08] Pollack A.: Redefining disease, genes and all. *New York Times* (May 2008).

[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* (Minneapolis, MN, USA, 2005), pp. 219–224.

[SA04] Shetty J., Adibi J.: *The Enron email dataset database schema and brief statistical report*. Tech. rep., Info. Sci. Inst. Tech. Rep., USC, 2004.

[SA05] Shetty J., Adibi J.: Discovering important nodes through graph entropy the case of Enron email database. In *Proceedings of the 3rd International Workshop on Link Discovery* (Chicago, IL, USA, 2005), pp. 74–81.

[Shn92] Shneiderman B.: Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics* 11, 1 (1992), 92–99.

[vHW08] van Ham F., Wattenberg M.: Centrality based visualization of small world graphs. *Computer Graphics Forum* 27, 3 (2008), 975–982.

[Vis09] VisualizationLab@UCB: Flare – data visualization for the web. UC Berkeley Visualization Lab, http://flare.prefuse.org (last visited on April, 2011), 2009.

[WGH04] Wu A. Y., Garland M., Han J.: Mining scale-free networks using geodesic clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Seattle, WA, USA, 2004), pp. 719–724.