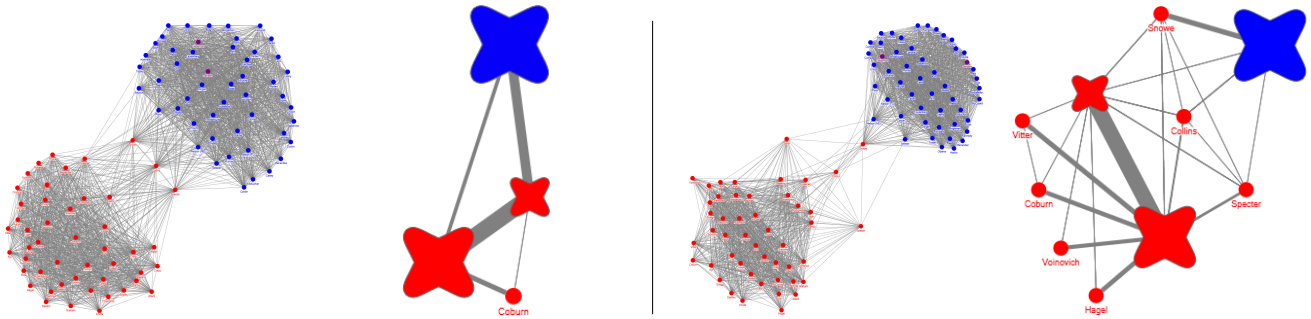
Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs

Cody Dunne, Ben Shneiderman
Department of Computer Science and Human-Computer Interaction Lab
University of Maryland, College Park, MD 20742
{cdunne, ben}@cs.umd.edu



(a) 65% (b) 65% simplified (c) 70% (d) 70% simplified

Figure 1: U.S. Senate 2007 co-voting network at 65% and 70% agreement cutoffs, simplified using clique motif glyphs. Key features are visible, such as the moderate Republican clique around McCain with "wildcards" at the periphery.

ABSTRACT

Analyzing networks involves understanding the complex relationships between entities, as well as any attributes they may have. The widely used **node-link diagrams** excel at this task, but many are difficult to extract meaning from because of the inherent complexity of the relationships and limited screen space. To help address this problem we introduce a technique called **motif simplification**, in which common patterns of nodes and links are replaced with compact and meaningful glyphs. Well-designed glyphs have several benefits: they (1) require less screen space and layout effort, (2) are easier to understand in the context of the network, (3) can reveal otherwise hidden relationships, and (4) preserve as much underlying information as possible. We tackle three frequently occurring and high-payoff motifs: **fans** of nodes with a single neighbor, **connectors** that link a set of anchor nodes, and **cliques** of completely connected nodes. We contribute design guidelines for motif glyphs; example glyphs for the fan, connector, and clique motifs; algorithms for detecting these motifs; a free and open source reference implementation; and results from a controlled study of 36 participants that demonstrates the effectiveness of motif simplification.

CHI 2013, April 27–May 2, 2013, Paris, France.

Author Keywords

Motif simplification; network visualization; graph drawing; node-link diagram; visual analytics.

ACM Classification Keywords

H.5.2. User Interfaces (D.2.2, H.1.2, I.3.6)

INTRODUCTION

Networks of entities and their ties have long been common data structures in computer science, but have only recently exploded into popular culture with publishers like the New York Times including elaborate and interesting networks with their articles. Online communities like Facebook, MySpace, Twitter, Flickr, and mailing lists (to name only a handful) enjoyed enormous growth over the last few years and provide incredibly rich datasets of interpersonal relationships, which social scientists are now fervently exploring. Networks have also found applications in such diverse disciplines as bioinformatics, scientometrics, urban planning, politics, and archeology.

Analysis of network data requires understanding clusters, connectivity, and centrality. Statistical analysis and conventional visualization tools like bar and pie charts are often inadequate when faced with these varied and oftentimes immense datasets. visualcomplexity.com provides almost 800 network visualizations, but most are variations of node-link diagrams, where nodes represent entities and the links or edges indicate ties connecting them. Node-link diagrams only recently became widely available but have already been put to great effect, such as detecting social roles in online newsgroups [32] or studying U.S. political blog ties during an election [2].

However, there is a huge array of possible layouts of the nodes and links in any given network, many of which can be misleading or incomprehensible. Network visualizations are only useful to the degree they "effectively convey information to the people that use them" [3]. In fact, the spatial layout of a node-link diagram can have a profound impact on the detection of communities in the network and the perceived importance of actors [23]. Significant thought must be given to proper visualizations so that analysts will be able to understand and effectively communicate data like clusters, the paths between them, and the importance of individual actors.

As manual layout of nodes in the node-link diagram is incredibly time consuming to do well, a lot of effort has been put into developing automated network layout algorithms and filtering tools. As the optimization of many readability metrics is NP-hard [3], layout algorithms often use heuristics that produce suboptimal visualizations quickly. However, the results of applying a layout algorithm can vary greatly depending on the size and topology of the network, and the layout generated is highly dependent on the algorithm used. We believe that state of the art layout algorithms alone are insufficient to consistently produce understandable network visualizations.

One way forward is the use of aggregation, specifically by aggregating common network structures or subnetworks called **motifs**. Large, complex network visualizations often have motifs repeated throughout because of either the network structure or how the data was collected. Regardless of their cause, some frequently occurring motifs contain little information compared to the space they occupy in the visualization. Existing tools may highlight certain motifs, allow users to filter them out manually, or replace them with meta-nodes.

We improve on these approaches with **motif simplification**, in which network motifs are automatically replaced with compact, representative glyphs. Well-designed glyphs have several benefits: they (1) require less screen space and layout effort, (2) are easier to understand in the context of the network, (3) can reveal otherwise hidden relationships, and (4) preserve as much underlying information as possible. In this paper we discuss three high-payoff motifs that plague network analysts, shown in Fig. 2: **fans**, **connectors**, and **cliques**. We contribute the design of representative and combinable glyphs for these motifs, algorithms for detecting them, and a supporting task-based controlled study with 36 participants. These techniques are all implemented and made publicly available as part of the free and open source NodeXL network analysis tool [27], which is available from nodexl.codeplex.com.

Specifically, the contributions of this paper are:

- A technique for simplifying node-link diagrams by replacing common network motifs with representative glyphs,
- A set of design guidelines for these glyphs to show the motif contents and underlying attributes,
- The design of glyphs for fans, connectors, and cliques,
- Algorithms for detecting these three motifs,
- A supporting task-based study with 36 participants, and
- A free and open source implementation as part of NodeXL.

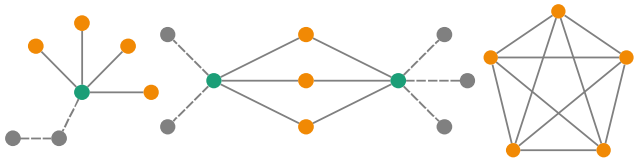


Figure 2: From left to right: fan, connector, and clique motifs.

RELATED WORK

Network analysis tools generally use node-link diagrams, as in NodeXL [27], or matrix representations like MatrixExplorer [13]. Both node-link and matrix diagrams show the topology of small networks well but can be unreadable with a few thousand nodes. We can reduce the visualization complexity by showing an aggregate version of the network, based on any number of criteria. NetLens [17] groups nodes by their attributes and can pivot between connected groups of two different types, while PivotGraph [31] uses attribute groupings but shows ties between aggregates using arcs. GraphTrail [5] combines these approaches with familiar charts, arc diagrams, and a many-to-many pivot between several node types. However, these approaches focus on attribute comparisons at the expense of showing topology.

Alternatively, we can use a hierarchical topologic clustering to show a network of meta-nodes like ASK-GraphView [1] and van Ham & van Wijk [30]. Rather than letting meta-nodes overlap, van Ham & van Wijk used semantic fisheye views to show clusters as merging spheres. Other approaches to creating overview networks include graph summarization [25] and aggregating nodes by shared neighbor sets [21]. [21] also provide a topologic clustering tool, and a level of detail option to split meta-nodes apart to better see the underlying topology. ManyNets [7] takes a different tack, showing statistical comparisons of a network partitioned by topology, attributes, or time. In each of these techniques it can be difficult to understand the topology of the individual aggregates, often because of the ambiguous nature of clustering algorithms.

Instead of clustering, we can use a metric for node importance to filter to an important subset. Skeletal images [15] highlights high-metric nodes, and replaces filtered trees with triangles that take the same space. Tsigkas et al. [29] similarly filtered a security network of events and features on a domain-specific metric, while including a way to aggregate the events joining a subset of features into meta-edges. However, the aggregation is limited to ties between two feature types and obscures the number of connecting nodes and edges.

Our approach is to aggregate the network by the frequently occurring motifs it contains. While the fan, connector and clique motifs we target are quite prominent in social network datasets, there are many other motifs of interest, especially for biologists. Motif census (counting the kinds of motifs) and analysis is used extensively to analyze the behavior of complex biologic networks, looking for repeated patterns that indicate underlying processes. For example, Milo et al. [24] find motifs that appear more frequently than expected in random networks, and provide a chart of small motif frequency.

Knowledge of the motifs present in a network can help predict behavior and the "structural signatures" of individual entities [32], but visualizing these motifs effectively is challenging. Huang et al. [16] detect motifs with fewer than five nodes and draw transparent convex hulls to highlight them. Similarly, in [19] the matches to a chosen 3–5 node motif are colored within the overall visualization and are drawn identically to be easily spotted. Highlighting small motifs can help biologists spot the locations of particular processes, but does little to reduce the clutter of a complex network visualization and can even reduce readability.

Current approaches to reducing complexity aggregate nodes based on their attributes, topology, or metrics but do not provide visible indications on the meta-nodes showing the underlying topology. Moreover, these algorithms usually pay little attention to the motifs present and create a grouping with ambiguous topology. While current tools can highlight small detected motifs, there are few techniques for providing a graphical overview or summary of them. More importantly, we know of no approaches that leverage the motifs present to reduce the visual complexity of the network visualization.

NETWORK MOTIF SIMPLIFICATION

Many common network motifs present little meaningful information, yet can dominate much of the display space and obscure interesting topology. We believe that replacing these motifs with representative glyphs will create more effective visualizations as there will be far fewer nodes and edges for layout algorithms and users to consider. We have chosen three motifs for our initial foray into motif simplification:

- A **fan motif** consists of a **head node** connected to **leaf nodes** with no other neighbors. As there may be hundreds of leaves, replacing all the leaves and their links to the head with a **fan glyph** can dramatically reduce the network size.
- A **D-connector motif** consists of functionally equivalent **span nodes** that solely link a set of D **anchor nodes**. Replacing span nodes and their links with a **connector glyph** can aid in connectivity comparisons.
- A **D-clique motif** consists of a set of D **member nodes** in which each pair is connected by at least one link. Cliques are common in biologic or similarity networks, where swapping for a **clique glyph** can highlight subgroup ties.

These motifs are prime simplification candidates for several reasons. For one, these motifs are quite common in the network datasets we have encountered in several disciplines. While simple to understand on their own, these motifs can account for much of the visual complexity of a node-link diagram. The fan motifs especially can dominate the diagram. While connector motifs usually occupy less space than the fans, they are hard to detect and can contribute substantial complexity. In the densest networks, such as pairwise similarity scores, overall relationships can be hidden in a tangled hairball of edges from overlapping clique motifs as in Fig. 1a.

Glyph Design

For each motif, careful thought must be given to how to represent the simplified version. Arbitrary motifs can be shown as a simple meta-node (e.g., ⊕), possibly with embedded images that show a small node-link diagram of the underlying subnetwork. However, a specially designed representative glyph for a motif can make it easier to understand aggregate topology and attributes with only minimal additional visual clutter. We went through several designs for each of our motif glyphs, though for space reasons we will primarily discuss the final designs and criteria for designing effective glyphs.

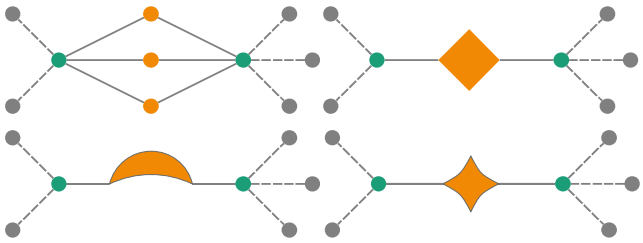


Figure 3: A 2-connector motif with three simplified glyph variants: diamond, crescent, and tapered diamond.

Motif Topology

Foremost each glyph must be representative of the underlying subnetwork topology so that the aggregate relationships in the network can still be understood. As we aim to reduce visual clutter, we must use a small, easily-distinguishable glyph rather than heavy-weight visualizations. An effective way to differentiate the glyphs is to use unique shapes to identify each type, ideally that correspond to the underlying topology.

Several example shapes for a connector motif are shown in Fig. 3. The diamond is a straightforward representation of the outline made by the motif topology, is discernible at scale, and has geometric properties that allow easy area scaling and subdivision. However, they are often used with other shapes for categorical attribute coding. The crescent is not, but our user studies indicated that its asymmetry was visually jarring and that it had poor edge connector properties. We finally chose a symmetric tapered diamond: unique enough to be distinguishable and representative yet symmetric and connectable. We use the same shape regardless of the number of anchor nodes so as to reduce the shape corpus required. The clique motifs were originally represented with a tapered square to indicate the link density, but it was easily confused with the connector motif and has since been replaced with a rounded X (Fig. 6). Like the connector motif, the same shape is used for any number of clique members. For the fan motifs, we chose a sector of a circle (Fig. 4), as it represented the fan of leaf nodes commonly seen in node-link diagrams.

Contained Nodes

In addition to the topology, it is helpful to show information about the nodes contained in the motif. What information we want to show impacts the display mechanism we choose for it. Most useful would be a count of the nodes in the motif. This quantitative value is best expressed by position [22], though in node-link diagrams this is reserved for showing ties. The next best choices would be length, angle, or area [22]. For the fan motif, we can scale the angle of the sector linearly between 10–120° by the number of contained nodes, which

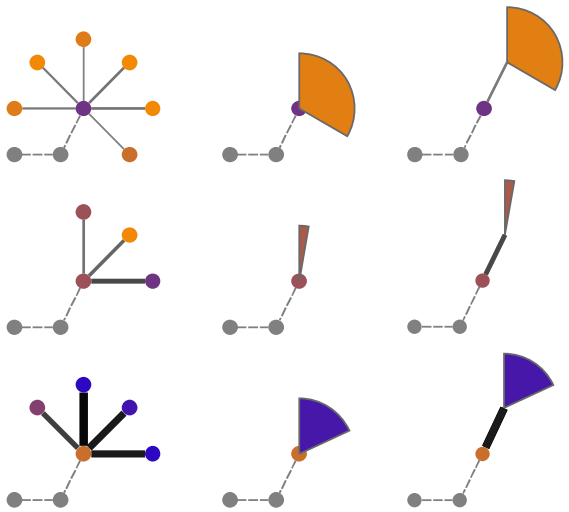


Figure 4: Three fan motifs and two glyph variants of each.

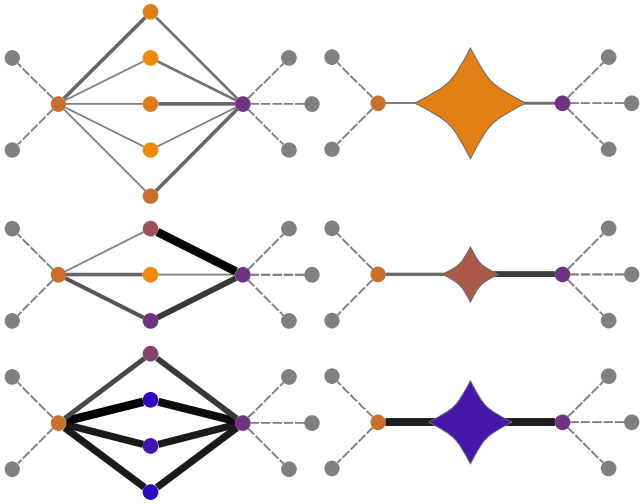


Figure 5: Three 2-connector motifs and their glyphs.

also linearly scales its area (Fig. 4). We chose this range after tests using smaller ranges (20–90°) did not reveal enough size variation. The vertical alignment eases area comparisons and eases glyph subdivision to show edge directionality or attributes. We also scale the area of the other motifs linearly by the number of nodes (Figs. 5 and 6). Designers should ensure the shape is still discernible at its minimum size while not so large at its maximum to occlude edges unnecessarily.

We may also wish to show quantitative attributes or statistics of the underlying nodes. Showing all the values or their distribution would require complex embedded charts or focusable tooltips. Instead, we show a function of the values such as mean (used for these examples), sum, min, or variance. As size is reserved for node count, we are left with the less effective color saturation, color hue, and density/opacity [22]. While these are less effective encodings, the maximum deviation reported for quantitative tasks is only 13% [4]. Glyphs

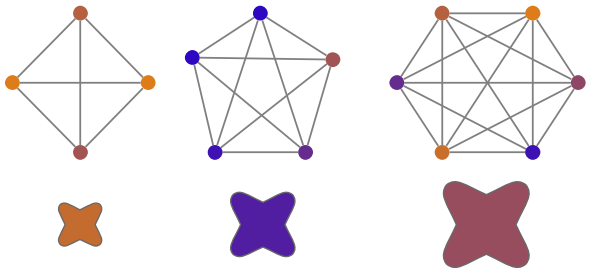


Figure 6: 4-, 5-, and 6-clique motifs and their glyphs.

demonstrating these quantitative attribute or statistic encodings are shown in Figs. 4 to 6, using the same color scale as the underlying nodes in the network. Categorical attributes are more challenging to display without subdividing glyphs or embedding visualizations, increasing the visual clutter. Finally, text attributes such as labels would help reveal the contents of the motif. While a glyph can show a small label, it is challenging to compute a representative one. Instead, we discuss later how interactivity can reveal the underlying nodes.

Connecting Edges

Nodes contained within a motif may have connecting edges, and when the motif is simplified these edges are re-routed to link to the glyph instead. This can result in duplicate, overlapping edges in straight-line drawings, as with the connector motif in Fig. 5. As with nodes, it is useful to show the number of duplicate edges and any attributes they may have. The edges could be drawn independently as curves of varying arcs or stacked in slices with scaled area, but again we strive to avoid visual clutter and show aggregate relationships clearly.

We aggregate these duplicate edges into meta-edges, with width and thus area representing a function of the underlying edges such as the number of edges (Figs. 1 and 8), the average of an attribute value (Figs. 4 and 5), etc. There are options for showing categorical attributes or labels, but these require cluttered embedded visualizations or interactivity. In some cases there are no attributes on the edges to encode, and showing even edge count would be a redundant. One example is the fan motif, in which the number of edges equals the already-encoded number of leaf nodes (in an undirected network without duplicates). Example fan glyphs without meta-edges are shown in the center column of Fig. 4.

Alas, glyph shape impacts how edges connect to them. Ideally, each glyph lies along a straight line with connecting edges so paths can be traced easily. For the 2-connector motif, a crescent would suffice if its corners were aligned along the path (Fig. 3). However, for connectors with three or more anchors our users reported that crescents make edges difficult to follow. Symmetric shapes like the tapered diamond and rounded X are better suited for many connecting edges.

Motif Overlap

Often motifs are non-overlapping and easily transformed into glyphs, though many motifs do not have this luxury. When detecting motifs we can choose a non-overlapping set to display, but motif glyphs will be more effective at reducing com-

plexity when they can be combined to show overlapping motifs. The design of any motif glyphs must thus take overlaps into account. Among our three motifs, fans are the most immune to overlap. The fan leaves have too few edges to participate in the other motifs, though the fan head can be a connector anchor or clique member. As a clique glyph replaces all the clique members, we must exclude the fan head from the fan glyph to allow this combination. Similarly, a connector anchor can be a clique member, which requires its exclusion from the connector glyph. Two example overlaps are shown in Fig. 7 and more on overlap handling is discussed later.

Glyph Interactivity

While the motif glyphs we described can be effective for simplifying a network, we would like to make sure that they are easily understandable and investigable. One important aspect of this is to ensure that users can switch between the original and simplified views interactively. Users can simplify the entire network, or only a selected subset of motifs. Likewise, users can expand the entire network to see the original visualization, or only expand a selected glyph they are interested in exploring. We expose the contents of each glyph with tooltips. It would be possible to expand on this and show details for a glyph via a heavyweight focusable tooltip that contains a chart of attribute distributions or a list of node labels.

Direct manipulation of the motif glyphs and underlying nodes is an effective way of exploring the network. Users can adjust node or glyph placement manually, as well as highlight incident edges or adjacent nodes through simple context menus. Additionally, automatic layout algorithms are available for laying out the simplified network. An ideal layout algorithm would take the shape and size of the glyphs into account, in addition to the number of edges in any meta-edges.

Motif Detection Algorithms

General motif detection can be accomplished with approaches like symmetry-breaking [10], but custom algorithms are more effective for specific motifs that can vary substantially in size. We have implemented algorithms to detect fan, connector, and clique motifs of all sizes, but due to space constraints can only present an overview and refer the interested reader to our supplementary material, tech report,[1] and source code.[2] We will use the terminology of a network or graph G with a set of nodes $G.nodes$ with a size denoted $|G.nodes|$.

Fan Motifs

We use the obvious algorithm for detecting fan motifs which has a run time complexity of $O(|G.nodes| \times$ average neighbor count). Average neighbor count is usually relatively small and can be considered a bounded constant. The main thing to note is that we count the neighbors for each node, rather than its degree, as there may be overlapping or directed edges.

Connector Motifs

Connectors have a **dimension**, denoted D, that indicates the number of anchors it has and is always two or greater. Our algorithm for detecting connector motifs of all dimensions takes parameters $D\text{-}min$ and $D\text{-}max$ to indicate the range of dimensions to search for. The run time complexity of this algorithm is also $O(|G.nodes| \times$ average neighbor count).

Connectors require more algorithmic details to detect than fans, despite having the same run time complexity. The algorithm is broken into several components. First, a pass is made through all nodes searching for span nodes with sets of neighbors that could be anchors and creating or adding to a map of keys to possible motifs. An additional pass is required to traverse the potential motifs and remove those with only one span node, as well as remove all but the most desirable of any overlapping motifs. We choose motifs to keep first by the total number of anchors and spanners, then by the number of spanners, then arbitrarily.

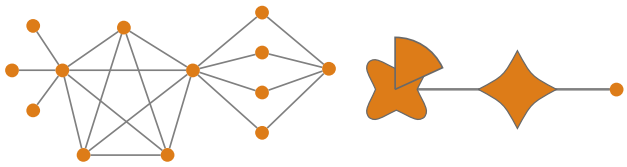


Figure 7: Glyphs for fan, clique, and connector motif overlap.

Clique Motifs

To find all cliques in the graph we use the Tomita et al. algorithm [28], which has a run time complexity of $O(3^{|G.nodes|/3})$. However, this algorithm has high memory requirements and for especially large graphs a new linear-storage algorithm by Eppstein and Strash may be faster or required [6]. Unfortunately cliques in general can have high amounts of overlap. We use a greedy heuristic that chooses the largest non-overlapping clique motifs to keep that is O(number of motifs $\times$ average motif size). This works well on the networks we have analyzed, but may be insufficient for studying dense networks.

Resolving Motif Overlap

When computing motifs, not only can motifs of a type overlap (like cliques), but in general the various types can overlap with each other as well. While our design for fan and connector motifs prevents any ambiguous overlap and allows easy combinations (Fig. 7), the choice of which cliques are simplified can impact user perception of the network. In order to effectively pick a disjoint set of motifs to keep, we would have to rate each motif by desirability and solve the set packing problem, one of Karp's 21 NP-complete problems [18]. Not only is this problem computationally hard to solve exactly, it is also difficult to approximate, hence our use of heuristics.

NodeXL Implementation

We have implemented a reference implementation of motif simplification and made it publicly available as part of the NodeXL network analysis tool [27]. NodeXL is a free and open source template for Microsoft Excel 2007/2010 that is tailored to provide powerful features while being easy to learn. The Excel integration allows rapid data processing

[1] www.cs.umd.edu/localphp/hcil/tech-reports-search.php?number=2012-29

[2] nodexl.codeplex.com/SourceControl/changeset/view/70521#1208172

using standard formulas and macros, but NodeXL also provides calculators for network statistics, automatic layout algorithms, visual attribute encodings, dynamic filters, direct manipulation, coordinated views, and importers from online social networks and common network file formats like GraphML, Pajek, and UCINET. More details are available in our supplementary material. NodeXL is widely used in many disciplines and has a full-time developer as well as a team of volunteer advisors, including the authors of this paper. Many introductory courses on network analysis have used NodeXL and its companion book [11] as part of their curriculum,[3] and user studies have shown NodeXL to be effective in these situations. Given that these users generally have little prior knowledge about network visualization readability, we believe that they will particularly benefit from our interactive motif simplification techniques.

CASE STUDIES

We explored several networks of interest using motif simplification, in several cases while helping domain experts analyze their data. Overall, motif simplification resulted in vastly reduced network size, reducing the visual complexity faced by the user and easing automatic and manual layout tasks.

U.S. Senate Voting Patterns in 2007

The power of clique motif simplification is shown in an example network of U.S. Senate voting patterns from 2007,[4] shown in Figs. 1 and 8. The percent similarity in voting patterns is an attribute of each one of the 4950 links connecting the 100 Senator nodes. The naive drawing produces a completely connected graph, but filtering the similarity values to show only those with values above 0.65 and applying a force-directed layout produces a revealing portrait (Fig. 1a). We see the willingness of three Republican Senators (center, in red) to vote in support of their Democrat colleagues (top-right, in blue). One of these, Arlen Specter, later switched his affiliation to the Democrats in 2009. However, further insights are not readily visible in the tangled hairball of each party.

After simplifying cliques, several additional features are visible (Fig. 1b). There are three completely connected groups: one with 48 Democrats, the two independents, and a Republican (Snowe); another with 42 Republicans; and a 4-clique of Collins, Smith, McCain, and Specter. We worked with a political scientist to see if these cliques highlighted known behavior, and, in fact, they did. The 4-clique represents moderate Republicans and bridge builders that were often decisive votes, though they have stronger ties to the Republican clique. The only Senator not in a clique is Coburn, a staunch Republican on contentions issues but who often votes his heart.

We increased the cutoff to 0.70 and ran the layout again (Fig. 1c). However, the simplified version (Fig. 1d) has become quite intriguing. While the Democrats and Independents still form a 50-clique, a few members trickled out of Republican cliques. Snowe returns to the middle with high connectivity with her former Democrat clique. Collins and Specter also move to the center, replaced in the McCain clique by Coleman and Lugar – more moderates. The corner outliers are known wildcards that do not follow the party.

Extending this process to higher cutoffs, we begin to see party fragmentation, led by the Republicans (Fig. 8). At 0.80 the network bisects (Fig. 8a), and the Democrats split into three cliques and a solitary Nelson, a Blue Dog moderate (Fig. 8b). The top right 4-clique is the east-coast liberals, while the left 4-clique are moderates. The Republicans splinter further, and by 0.95 only the two Senators from Georgia remain (Fig. 8d).

All told, the political scientist was impressed that motif simplification could highlight many of the features he was already aware of. Moreover, several new insights came from analyzing these visualizations and then checking other sources to provide additional evidence for the pattern.

Lostpedia Wiki Edits

An example of overlapping motif simplification is shown in Fig. 9, which represent the bipartite network for the Lostpedia wiki community collected by Beth Foss. Boxes with labels show wiki pages, linked to the colored discs representing their associated editors. The editors are colored and sized according to two measures of their activity in the wiki. Fig. 9a shows the initial network, while the Fig. 9b shows a simplified version. By combining fan and connector glyphs, we only have 13 nodes to lay out and compare instead of the original 513, only 23 edges instead of 586, and use a fraction of the screen space. While these simplifications are not entirely necessary to understand such a small and well-arranged diagram, they are effective at showing aggregate relationships like the large number of highly active main page editors.

VOSON Web Crawl

A larger dataset we encountered is shown in Fig. 10a, which we modified from the NodeXL book, Fig. 12.9 [11, p. 192]. This network of 3958 web pages and 4380 hyperlinks was collected by crawling sites connected to voson.anu.edu.au. It is immediately evident that large fans of nodes dominate the periphery, in in part because the NodeXL [27] implementation of the Fruchterman-Reingold layout [8] tends to draw elliptical layouts within a rectangular space. However, the fans tend to dominate the visualization regardless of the layout.

Our manual calculations using Gimp showed that 21% of the screen space in Fig. 10a is wasted as blank space in the corners, with 33% showing the core network with its connector motifs, and the remaining 46% used to show the fan motifs. Calculating only for the elliptical visualization region, approximately 58% of the space available is used to show the fan motifs. This is a substantial amount of area dedicated to showing a very common structure in network datasets obtained by crawling web sites or using surveys. Moreover, these fans do not show any information besides the rough number of nodes they contain. The fans in Fig. 10a vary from 17 to 852 nodes, but due to overlap this can be hard to see.

Some of the overlap between motifs and and with other nodes is not visible in the original image, but there is substantial overlap in the bottom-right and many of the smaller fans are

[3] nodexl.codeplex.com/wikipage?title=NodeXL%20Teaching%20Resources

[4] Data by Chris Wilson of Slate magazine, available from above link.

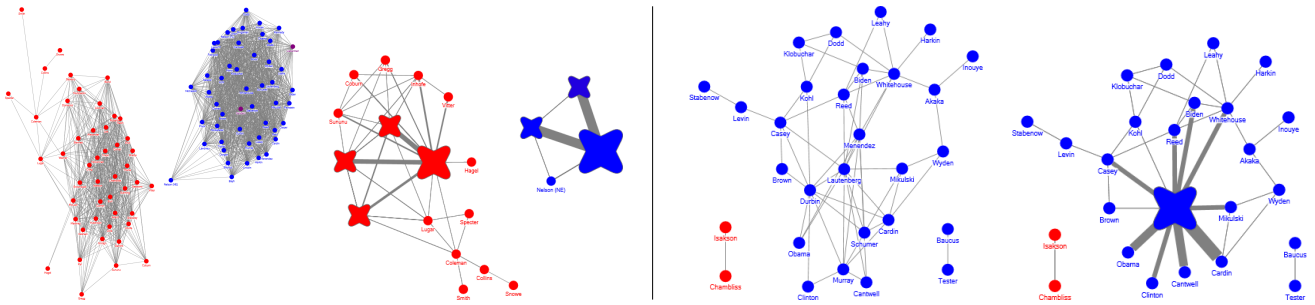


Figure 8: U.S. Senate 2007 co-voting network at 80% and 95% agreement cutoffs, simplified using clique motif glyphs. The east-coast liberals and the Blue Dog Democrats separate at 80%. We see the network decompose at higher cutoffs.

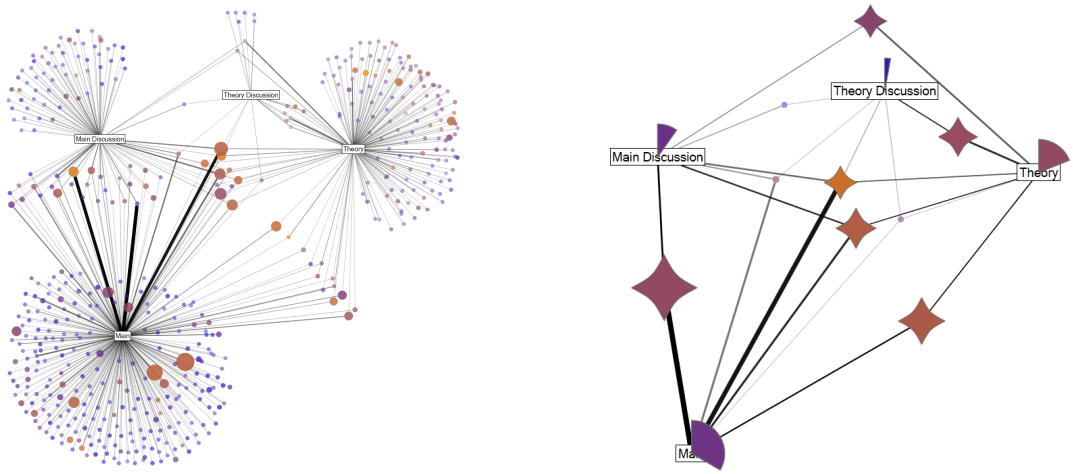


Figure 9: A bipartite network of Lostpedia of wiki edits (left) and a simplified version using fan and connector glyphs (right).

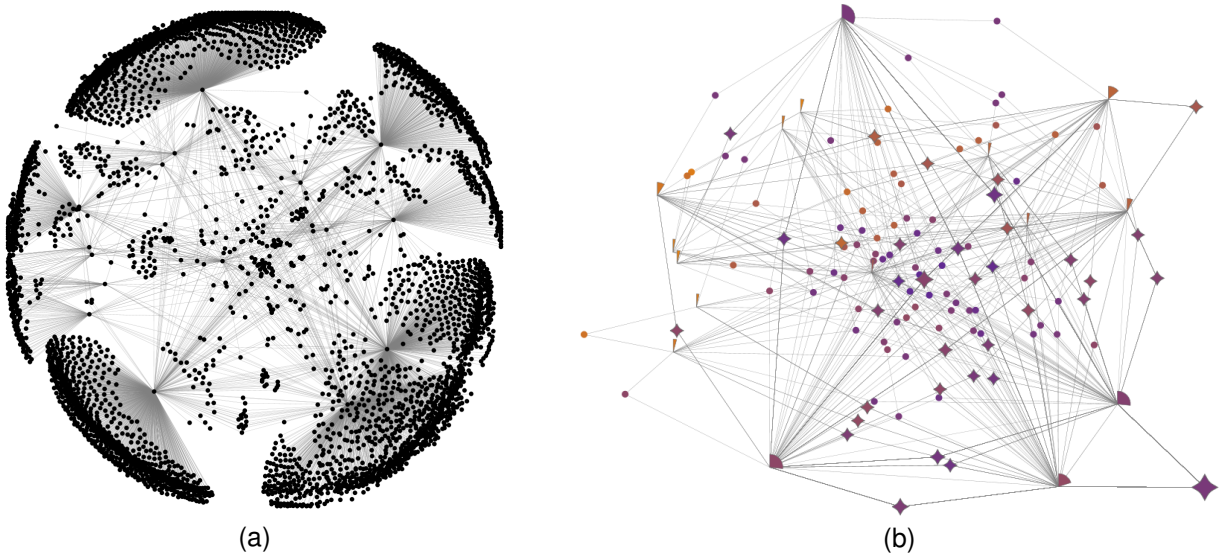


Figure 10: A web crawl starting at voson.anu.edu.au, modified from Fig. 12.9 of the NodeXL book [11], p.192 (left) and the simplified version using fan and connector motifs and colored by eigenvector centrality (right).

spread in several directions or hidden in the interior. Moreover, many of the fans overlap and obscure other more important nodes that are not participating in any fan, such as a huge 2-connector motif with 50 span nodes in the bottom-right. This 2-connector motif, as well as the several others connecting parts of the web page network together, are quite hard to detect among the clutter.

We then simplified these fan and connector motifs, going from 3958 nodes to 559 and 4380 edges to 765, creating a much less cluttered visualization (Fig. 10b) After simplification, it became evident that the large connector motif in the linked the web sites for the Summer Doctoral Programme at the Oxford Internet Institute and the National Center for eSocial Science. Applying a layout algorithm to the simplified network would result in a new layout that makes more effective use of the newfound space. This visualization is much clearer at presenting (1) the size and membership of the various fans motifs and (2) the large connector motifs connecting pairs of fan heads. Moreover, it appears to have minimal loss of information and visual clutter compared to the original.

Larger Networks

We analyzed several other large networks not pictured here. One was a network of innovation and funding ties with 7124 nodes and 16,109 edges. Another showed acquisitions of JP Morgan Chase, with 5766 nodes and 6752 edges. Both were visualized interactively with no performance issues, and had drastic reductions in complexity with motif simplification.

CONTROLLED EXPERIMENT

We ran a controlled experiment to determine the effect motif simplification has on user performance with several common network visualization tasks.

Tasks

We chose a varied set of tasks relating to topology, attributes, and overviews from a taxonomy [20], which demonstrates how all complex tasks can be seen as a series of low-level tasks. These tasks are also used in many recent papers evaluating network visualizations [14, 26, 9]. We asked:

1. About how many nodes are in the network?
2. Which individual node would we remove to disconnect the most nodes from the main network?
3. Which is the largest (fan | connector | clique) motif and how many nodes does it contain?
4. Which node has the label "XXX"?
5. What is the length of the shortest path between the two highlighted nodes?
6. Which of the two highlighted nodes has more neighbors?
7. How many common neighbors are shared by the two highlighted nodes?
8. Which of two pairs of nodes has more common neighbors?

Data

Current random network generators do not produce realistic data [14], which we confirmed trying to generate several networks with similar characteristics. Thus we chose to use three sufficiently interesting networks produced by actual users solving their own problems, which are discussed in the Case Studies section.

Participants

After a two-participant pilot study, we recruited 36 students from our university (19 males and 17 females). The participants were almost entirely graduate students, half from computer science and the balance from eight other departments. 9 had used network visualization tools and an overlapping 9 had seen motif simplification, though none had used it. As we could not generate sufficiently varied datasets with similar properties, we used a between subjects design. We randomly divided participants into two groups, which had similar distributions of gender, department, grade level, and experience.

Procedure

Each 45-minute session began with 5-10 minutes of training on the tool and for the specific tasks, followed by about 35 minutes for answering a total of 31 questions across the three networks and tasks. Each participant received the same order of questions and visualizations. The control group was provided with an interactive node-link diagram in which they could select nodes along with their incident edges, as well as move the nodes. The treatment group received a simplified version of each new visualization, with additional interactive tooltips and the ability to expand and collapse the motifs. Each visualization is presented consistently, originally computed using Harel-Koren [12].

As in [9, 14], users were given one minute to answer each question, told to answer as quickly and accurately as possible, and that they could skip if they could not answer a question. The evaluator spoke each question, gave the participant time to ask for clarification, then revealed the next visualization in turn and began the timer. Users were given $10 plus a $15 bonus for the fastest, most accurate participant in each group.

Analysis

The recorded data was analyzed in several ways. As is common with response time data, the response times were not normally distributed so were normalized using a log transformation. The two groups were then compared using a t-test. Answers to questions consisted of a categorical answer (a specific item), which was recorded correct or not, and/or an integer answer. For questions with categorical answers, the groups were compared with Fisher's exact test instead of the chi-square test as none of the significant group-by-correct matrices had expected values of five or higher in all four cells. For numeric answers we computed $error = (answer - truth)/truth$, skipping any questions that had an incorrect categorical answer the integer answer depended on, and compared error across groups using a t-test.

Results

Here we report only the significant findings. We expect overview tasks like identifying the maximal motif of a type would be easier with the less visual complexity of a simplified network. This was true for all three motifs across all three networks. Cliques, the epitomical clusters, were found in the two networks they occurred in faster ($p<0.01$, -20.82s),

more accurately (p<0.01, 92% vs. 23.5%), and with fewer people giving up (3 vs. 0). Moreover, in the Senate network there was higher accuracy in size estimates (p<0.05, 0% vs. -28% error), which could be true for the web network but we could not measure it as not one control participant detected the maximal 5-clique.

Fans were found in both the networks they occurred in faster (p<0.01, mean -7.77s) and their size was approximated more closely (p<0.01, 2% vs. -62% error). In the large web network the maximal fan was also found more frequently (p<0.01, 95% vs. 35%). Connectors were detected in both their networks faster as well (p<0.01, mean -17.13s). In the web network the largest connector was found more frequently (p<0.01, 79% vs. 6%), and in the wiki network its size was estimated more precisely (p<0.1, -5% vs. -17% error).

These results show that using glyphs for motifs makes the motifs easier to detect and measure, but how does simplifying motifs affect the rest of the network? We would think estimating the number of nodes would be easier in the simplified, interactive view. Our participants could indeed gauge the size of all three networks with significantly more accuracy (p<0.01, -8% vs. -47% error), but for the wiki and web networks users took longer to do so (p<0.01, 21.82s). How about finding a specific node by its label? Logically reducing the number of visual items makes finding a label easier. Our results show that finding labels that are not in motifs is significantly faster (p<0.01, -19.93s), they are found more frequently except in the Senate case (p<0.01, 97.5% vs. 14.5%), and fewer users give up or run out of time (12 did on the plain wiki and web networks). We only saw worse search time for labels in motifs for the Senate clique case (p<0.05, 15.29s).

What about topology-based tasks? It seems that with fewer items on the screen tracing edges would be easier. For some questions it did turn out better, like finding the node to cut in the web network correctly (p<0.05, 53% vs. 18%) and the accuracy of the shortest path length between two clique members in the Senate network (p<0.05, -7% vs. 22% error). For others topology questions, the results were mixed to poor. Shortest path length time and accuracy worsened in the web network (p<0.1, 10.06s & 20% vs. 1% error). Comparing the number of neighbors was slower on the wiki (p<0.01, 10.89s) and senate (p<0.05, 9.26s) networks, and the choice accuracy dropped for the senate (p<0.1, 53% vs. 82%) and web (p<0.1, 68% vs 76%). Lastly, the shared neighbor count tasks were slower in the web network (p<0.01, 11.73s), and reduced accuracy in the wiki network (p<0.1, -21% vs. -10%).

Discussion

All told it appears that motif simplification is beneficial for many analysis tasks. Naturally identifying maximal motifs is faster, more accurate, and we can estimate their sizes more accurately when we have glyphs and interaction. Counting nodes in the network turned out to be slower, but more accurate when using the glyphs. Finding unsimplified labels became much quicker, while simplified labels were only slower in one case. Finally, it seems like topology-based tasks are a mixed bag. Finding cut nodes is more accurate, but path-based tasks were better and worse in different circumstances. Comparing the number of neighbors and shared neighbors turned out slower and less accurate in a few cases, while counting them was more error-prone.

We have already implemented additional features to increase user performance on topologic tasks. When we ran the study we did not yet use the sized meta-edges that are shown in Fig. 1. With this simple modification, we believe we can show much of the aggregate connectivity. However, user education is likely the most promising way to improve the glyph performance. Many participants had difficulty understanding the topology inside the collapsed glyphs. With more than the 5-10 minutes of training provided in this study, users may perform much better on these tasks.

LIMITATIONS & FUTURE WORK

Our studies indicate that motif simplification is an effective way of reducing node-link diagram complexity, but it does pose several challenges. There is an extra effort in learning the motif concepts and interpreting the glyphs, which may deter some users, but simplification is a user choice which can be reversed at any time. Heavyweight glyphs with color striping or embedded charts would help users see the content of a motif, but would also add to the visual clutter and reduce the utility of the simplification. While the underlying topology of an individual motif is unambiguous, in some cases the choice of which motifs to simplify can lead to different overviews. The fan and connector motifs prevent ambiguous overlap, but clique motifs can overlap each other substantially. We use a heuristic that picks the largest non-overlapping clique to simplify. A more effective, but computationally hard, approach would be to rate each motif by desirability and find the optimal set of motifs. Alternatively, instead of visualizing exact cliques, we could show the overlap between almost-cliques and the confidence of various clusterings.

CONCLUSION

We present motif simplification, a technique for increasing the readability node-link diagram network visualizations. With motif simplification, common repeating network motifs are replaced with easily understandable motif glyphs that require less space, are easier to understand, and reveal hidden relationships. While users must learn the visual language of motifs and glyphs, there is a dramatic payoff in the usability and readability of the visualization. We contribute design guidelines for motif glyphs; designs of glyphs to replace the high-payoff fan, connector, and clique motifs common in networks; as well as algorithms to identify these motifs. Finally, we have developed a free and open source reference implementation, made publicly available as part of NodeXL.

With case studies and a controlled study we demonstrate the effectiveness of motif simplification as well as areas to focus on for improving glyph design. Motif simplification can result in substantial reductions in visual complexity, allowing easier understanding and manipulation of large network visualizations. There are several avenues for exploration opened up by this work, including additional glyphs for other common motif types, algorithms and glyphs for fuzzy motifs, and methods for showing edge directionality within glyphs.

ACKNOWLEDGEMENTS

We thank Marc Smith, the NodeXL team, and Yiyan Liu for their assistance. This work is supported in part by the Social Media Research Foundation, the Connected Action Consulting Group, and National Science Foundation grant 0915645.

REFERENCES

1. Abello, J., van Ham, F., and Krishnan, N. ASK-GraphView: a large scale graph visualization system. *IEEE TVCG 12*, 5 (2006), 669–676.
2. Adamic, L. A., and Glance, N. The political blogosphere and the 2004 U.S. election: Divided they blog. In *ACM LinkKDD* (2005), 36–43.
3. Battista, G. D., Eades, P., Tamassia, R., and Tollis, I. G. *Graph drawing: Algorithms for the visualization of graphs*. Prentice Hall, July 1998.
4. Cleveland, W. S., and McGill, R. Graphical perception and graphical methods for analyzing scientific data. *Science 229*, 4716 (1985), 828–833.
5. Dunne, C., Riche, N. H., Lee, B., Metoyer, R. A., and Robertson, G. G. GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *ACM CHI* (2012), 1663–1672.
6. Eppstein, D., and Strash, D. Listing all maximal cliques in large sparse real-world graphs. In *SEA*, vol. 6630 (2011), 364–375.
7. Freire, M., Plaisant, C., Shneiderman, B., and Golbeck, J. ManyNets: An interface for multiple network analysis and visualization. In *ACM CHI* (2010), 213–222.
8. Fruchterman, T. M. J., and Reingold, E. M. Graph drawing by force-directed placement. *SPE 21*, 11 (1991), 1129–1164.
9. Ghoniem, M., Fekete, J.-D., and Castagliola, P. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE InfoVis* (2004), 17–24.
10. Grochow, J., and Kellis, M. Network motif discovery using Subgraph Enumeration and Symmetry-Breaking. In *RECOMB* (2007), 92–106.
11. Hansen, D., Shneiderman, B., and Smith, M. *Analyzing social media networks with NodeXL: Insights from a connected world*. Morgan Kaufmann, 2011.
12. Harel, D., and Koren, Y. A fast multi-scale method for drawing large graphs. *JGAA 6*, 3 (2002), 179–202.
13. Henry, N., and Fekete, J.-D. MatrixExplorer: A dual-representation system to explore social networks. *IEEE TVCG 12*, 5 (2006), 677–684.
14. Henry, N., and Fekete, J.-D. MatLink: Enhanced matrix visualization for analyzing social networks. In *INTERACT* (2007), 288–302.
15. Herman, I., Marshall, M. S., Melançon, G., Duke, D. J., Delest, M., and Domenger, J.-P. Skeletal images as visual cues in graph visualization. In *Data Visualization* (1999), 13–22.
16. Huang, W., Murray, C., Shen, X., Song, L., Wu, Y. X., and Zheng, L. Visualisation and analysis of network motifs. In *IEEE InfoVis* (2005), 697–702.
17. Kang, H., Plaisant, C., Lee, B., and Bederson, B. B. NetLens: Iterative exploration of content-actor network data. In *IEEE VAST* (2006), 91–98.
18. Karp, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, 85–103.
19. Klukas, C., Schreiber, F., and Schwöbbermeyer, H. Coordinated perspectives and enhanced force-directed layout for the analysis of network motifs. In *APVis* (2006), 39–48.
20. Lee, B., Plaisant, C., Parr, C. S., Fekete, J.-D., and Henry, N. Task taxonomy for graph visualization. In *ACM BELIV* (2006), 1–5.
21. Liao, Q., Shi, L., and Sun, X. Anomaly analysis and visualization through compressed graphs. In *IEEE LDAV Poster Session* (2012).
22. Mackinlay, J. Automating the design of graphical presentations of relational information. *ACM TOG 5*, 2 (1986), 110–141.
23. McGrath, C., Blythe, J., and Krackhardt, D. The effect of spatial arrangement on judgments and errors in interpreting graphs. *Social Networks 19*, 3 (1997), 223–242.
24. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. Network motifs: Simple building blocks of complex networks. *Science 298*, 5594 (2002), 824–827.
25. Navlakha, S., Rastogi, R., and Shrivastava, N. Graph summarization with bounded error. In *ACM SIGMOD* (2008), 419–432.
26. Shneiderman, B., and Aris, A. Network visualization by Semantic Substrates. *IEEE TVCG 12*, 5 (2006), 733–740.
27. Smith, M., Shneiderman, B., Milic-Frayling, N., Rodrigues, E. M., Barash, V., Dunne, C., Capone, T., Perer, A., and Gleave, E. Analyzing (social media) networks with NodeXL. In *ACM C&T* (2009), 255–264.
28. Tomita, E., Tanaka, A., and Takahashi, H. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science 363*, 1 (2006), 28–42.
29. Tsigkas, O., Thonnard, O., and Tzovaras, D. Visual spam campaigns analysis using abstract graphs representation. In *ACM VizSEC* (2012), 64–71.
30. van Ham, F., and van Wijk, J. J. Interactive visualization of small world graphs. In *IEEE InfoVis* (2004), 199–206.
31. Wattenberg, M. Visual exploration of multivariate graphs. In *ACM CHI* (2006), 811–819.
32. Welser, H. T., Gleave, E., Fisher, D., and Smith, M. Visualizing the signatures of social roles in online discussion groups. *JOSS 8*, 2 (2007).

Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs – Supplementary Material

Cody Dunne, Ben Shneiderman
Department of Computer Science and Human-Computer Interaction Lab
University of Maryland, College Park, MD 20742
{cdunne, ben}@cs.umd.edu

ABSTRACT

This document provides supplementary material for the CHI 2013 paper "Motif simplification: improving network visualization readability with fan, connector, and clique glyphs". It presents algorithms for detecting two common network motifs: **fans** of nodes with a single neighbor and **connectors** that link a set of anchor nodes. The other motif the full paper deals with are **cliques** of completely connected nodes, but the clique detection is fully described in the full paper and makes use of the Tomita et al. algorithm [3]. This document also provides more of the implementation details for motif simplification in NodeXL [2].

NETWORK MOTIF DETECTION

There are two particular motifs we will discuss:

- A **fan motif** consists of a **head node** connected to **leaf nodes** with no other neighbors. As there may be hundreds of leaves, replacing all the leaves and their links to the head with a **fan glyph** can dramatically reduce the network size.
- A **D-connector motif** consists of functionally equivalent **span nodes** that solely link a set of D **anchor nodes**. Replacing span nodes and their links with a **connector glyph** can aid in connectivity comparisons.

General motif detection can be accomplished with approaches like symmetry-breaking [1], but custom algorithms are more effective for specific motifs that can vary substantially in size. We have implemented algorithms to detect fan and connector motifs of all sizes, but due to space constraints the main paper only presents an overview with the details presented here, in the tech report,[1] and source code.[2] We use the terminology of a network or graph G with a set of nodes $G.nodes$, and each node n has a set of adjacent nodes $n.neighbors$. The size of each of these node sets, say s, is denoted as $|s|$.

Fan Motifs

Our approach to detecting all the fan motifs in a network is detailed in Algorithm 1, which has a run time complexity of $O(|G.nodes| \times$ average neighbor count). Average neighbor count is usually relatively small and can be considered a bounded constant. The algorithm first passes through all the nodes in the network, searching for potential fan heads. Each fan head must have two or more neighbors to exclude the degenerate barbell case (Line 3), though this criteria could be increased to find larger fans. For each potential fan head, we then search through the set of its neighbors to find any leaf nodes connected only to it (Line 5). Each of these leaf nodes are added to the set of potential leaves. If two or more leaves have been detected in the neighbor set, the found fan motif is acceptable and recorded (Line 10).

Algorithm 1 Fan motif detection algorithm.
Time complexity: $O(|$G.nodes$| \times$ average neighbor count)

```
1:  procedure DetectFans
2:      for all n ∈ G.nodes do
3:          if |n.neighbors| ≥ 2 then
4:              leaves ← {∅}
5:              for all nbr ∈ n.neighbors do
6:                  if |nbr.neighbors| = 1 then
7:                      leaves.add(nbr)
8:                  end if
9:              end for
10:             if |leaves| ≥ 2 then
11:                 RecordFan(n, leaves)
12:             end if
13:         end if
14:     end for
15: end procedure

16: procedure RecordFan(head, leaves)
17:     ...                          ▷ Record a given fan motif
18: end procedure
```

The differing neighbor count criteria for head and leaf nodes in Algorithm 1 prohibit any overlapping motifs from being detected. However, please note that we are using $|n.neighbors|$ to show the size of the neighbor set of n, which may differ from n's degree if there are overlapping edges. For example, in a network with directed edges a leaf node may have two overlapping edges connecting it to the head node, one for each direction. Moreover, an undirected network with several edge types may have overlapping edges of differing types. Some algorithms for computing degree would return higher values in these cases than the actual number of neighboring nodes.

[1] www.cs.umd.edu/localphp/hcil/tech-reports-search.php?number=2012-29
[2] nodexl.codeplex.com/SourceControl/changeset/view/70521#1208172

Algorithm 2 D-Connector motif detection algorithm. [D-min, D-max] is the range of dimensions of the connector motifs to find (the number of anchors). Time complexity: $O(|\text{G.nodes}| \times \text{average neighbor count})$.

```
1:  procedure DETECTCONNECTORS(D-min, D-max)
2:      found ← Map⟨STRING, CONNECTOR⟩
3:      detectLoop:
4:      for all n ∈ G.nodes do
5:          if |n.neighbors| ∈ [D-min, D-max] then
6:              for all nbr ∈ n.neighbors do
7:                  if |nbr.neighbors| < 2 then
8:                      continue detectLoop
9:                  end if
10:                 ADDSPAN(n.neighbors.sorted, n, found)
11:             end for
12:         end if
13:     end for
14:     out ← {∅}
15:     used ← Map⟨NODE, CONNECTOR⟩
16:     filterLoop:
17:     for all c ∈ found.values do
18:         if |c.spanners| ≥ 2 then
19:             for all s ∈ c.spanners do
20:                 if s ∈ used.keys then
21:                     c′ ← used[s]
22:                     cTotal ← |c.spanners| + |c.anchors|
23:                     c′Total ← |c′.spanners| + |c′.anchors|
24:                     if cTotal ≥ c′total or
                          (cTotal = c′total and
                          |c.spanners| > |c′.spanners|) then
25:                         out.remove(c′)
26:                         used.removeAll(c′.spanners)
27:                         used.removeAll(c′.anchors)
28:                         ADDCONNECTOR(out, used, c)
29:                     end if
30:                     continue filterLoop
31:                 end if
32:             end for
33:             ADDCONNECTOR(out, used, c)
34:         end if
35:     end for
36:     for all c ∈ out do
37:         RECORDCONNECTOR(c.anchors, c.spanners)
38:     end for
39: end procedure
```

Algorithm 3 Continuation of Algorithm 2

```
40: procedure ADDSPAN(anchors, spanner, found)
41:     key ← string(anchors)
42:     if key ∉ found then
43:         found[key] ← new CONNECTOR(anchors)
44:     end if
45:     found[key].spanners.add(spanner)
46: end procedure

47: class CONNECTOR
48:     anchors ← {∅}, spanners ← {∅}
49:     procedure CONNECTOR(new-anchors)
50:         anchors ← new-anchors
51:     end procedure
52: end class

53: procedure ADDCONNECTOR(out, used, c)
54:     out.add(c)
55:     for all spanner ∈ c.spanners do
56:         used[spanner] ← c
57:     end for
58:     for all anchor ∈ c.anchors do
59:         used[anchor] ← c
60:     end for
61: end procedure

62: procedure RECORDCONNECTOR(anchors, spanners)
63:     ...                          ▷ Record a given connector motif
64: end procedure
```

Connector Motifs

Connectors have an **dimension**, denoted D, that indicates the number of anchors it has. D can be any integer two or greater, though the frequency of the motifs generally decreases proportional to D. Our algorithm for detecting connector motifs of all dimensions is shown in Algorithm 2, and takes parameters $D\text{-}min$ and $D\text{-}max$ to indicate the range of dimensions to search for. The run time complexity of this algorithm is also $O(|G.nodes| \times \text{average neighbor count})$. Again, average neighbor count can be considered a bounded constant.

Connector motifs are not as straightforward to detect as fan motifs, despite the algorithms having the same run time complexity. First, a pass is made through all nodes searching for span nodes with sets of neighbors that could be anchors and creating or adding to a map of keys to possible motifs. An additional pass is required to traverse the potential motifs and remove those with only one span node, as well as remove all but the most desirable of any overlapping motifs. We choose motifs to keep first by the total number of anchors and spanners, then by the number of spanners, then arbitrarily.

Algorithm 2 is broken into several procedures and a class to store the details for each potential connector motif. The detect loop in the algorithm (Line 3) passes through all nodes in the network, searching for potential span nodes. Each span node must have between $D\text{-}min$ and $D\text{-}max$ neighbors, which must be anchor nodes. We require a minimum of two span nodes for the connector motif, so each anchor node must have two or more neighbors itself (Line 7). At least two of the neighbors are span nodes, but the remainder can be connections to the main network or other anchor nodes in the motif.

If all the anchor nodes check out, the span node is added to a connector motif (Algorithm 2, Line 10) using the ADDSPAN procedure (Line 40). This motif can be new or an existing one with the same set of anchors. All existing motifs are stored in a map (Line 2), using a string representation of the anchors as a key and an instance of the **CONNECTOR** class (Line 47)

as the associated value. This allows speedy lookup of each potential motif given a sorted anchor set. Note that the anchor set and its string representation must be sorted so as to avoid having motifs with identical anchor sets but the anchors were found in a different order.

After searching for all potential span nodes, Algorithm 2 requires an additional pass over the detected connector motifs to ensure that (1) they have two or more span nodes and (2) they do not overlap with other connector motifs. The filter loop on Line 16 goes through each potential **CONNECTOR** instance in the map to verify that they pass these two criteria. The first criteria, the minimum number of span nodes, could be increased if only larger higher payoff motifs are of interest (Line 7, 18). An example we have found that matches the second criteria, connector motif overlap, is a ring of four nodes $A-B-C-D-A$ isolated from the rest of the network. In this case it is unclear whether to choose A & C or B & D as the 2-connector motif anchors, as we do not allow overlap.

As there may be other examples of overlap that need to be caught, we chose a general overlap detection approach that compares each span node s in a motif to all span and anchor nodes in already detected motifs (Line 19 – Line 33). If one of the span nodes$s$ of a potential **CONNECTOR** c is also a span or anchor node of an already found motif c', we then compare their sizes. We choose to keep the motif that has the greatest total number of anchors and spanners, and if they are equal we choose the one with more spanners. If both values are equal we keep the first detected. If the prior motif c' is to be replaced, we must first remove its spanners and anchors from the map (Line 25–Line 27). After passing the minimum span count and overlap ranking checks, the detected connector motif c is then stored (Lines 28 and 36) using the ADDSPAN procedure (Line 40). As part of this, the spanners and anchors are all added to the map of used nodes and associated with their **CONNECTOR**. All this bookkeeping process prevents a potential connector motif from overlapping with more than one that was already found. Finally, we record the remaining non-overlapping and valid connector motifs (Line 36).

NODEXL IMPLEMENTATION

We have implemented a reference implementation of our motif simplification approach and made it publicly available as part of the NodeXL network analysis tool [2]. NodeXL is a free and open source template for Excel 2007/2010 that is tailored to provide powerful features while being easy to learn. The basic interface of NodeXL is shown in Fig. 1. The left side provides several worksheets in an Excel workbook that represents the network: one each for the nodes, edges, and any groups. Each worksheet has several columns, including basic information about the network like the nodes and edges between them. Additionally, there are places to insert columns for node or edge attributes and calculated metrics, as well as columns that control the visual display of each network item. These include color, shape, size, label, tooltip, display position, and the like. Any of these visual properties can be automatically filled based on the metric or attribute columns using a special autofill dialog. Moreover, standard Excel formulas or macros can be used for arbitrary calculations and scales. The Excel ribbon is customized with a new tab for many of the common operations users perform on networks, including the autofill feature.

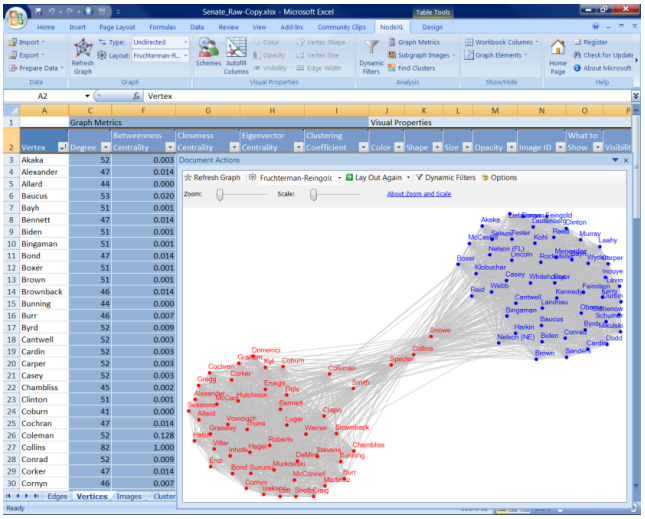


Figure 1: The standard NodeXL workspace, showing U.S. Senate voting patterns from 2007. The left view shows the worksheets that store the network and its attributes, while the right pane shows a node-link diagram of the network.

The visualization pane shown in the right of Fig. 1 displays a node-link diagram based on the network in the workbook. Whenever the contents of the workbook is updated, the visualization pane can be updated using a button. The pane also provides users with several automatic layout algorithms to arrange the network, and any automatic or manual adjustments to the node positions are stored in the workbook as well. Moreover, the contents of the visualization can be filtered using a dynamic filters dialog.

The worksheet view and the visualization pane are connected using brushing, where any selection in one is reflected in the other. Clicking a node in the visualization or dragging a box around several causes the associated rows to be selected in the nodes worksheet. Likewise, any incident edges are selected in the edges worksheet. The reverse is also true. Any nodes or edges selected in the worksheets are highlighted in the visualization pane as well.

Motif Simplification in NodeXL

We have integrated our motif simplifications into the standard NodeXL groups infrastructure, which stores groups using two worksheets: (1) **Groups** which contains a row for each group and its attributes, and (2) **Group Vertices** where each row maps an individual grouped node to its associated group. These worksheets can be populated automatically in a variety of manners, including detection of topological clusters, exact-value attribute groupings, connected components, and now the fan and connector network motifs. The NodeXL group model allows for nodes that are in no group at all, which is important for motif simplification as not every node in the network is part of a motif. Note however that this group

model does not allow overlapping groups, which means that special care must be given to the definition of what members of each motif constitute the group in the worksheets.

In the group worksheets users can interactively edit the labels, attributes, visual encoding, and membership of specific groups; remove groups completely; or even create custom sets of groups by editing the worksheets or visual interaction with the node-link diagram visualization. Moreover, automated statistics can be computed for each group and added to the Groups worksheet, including node & edge counts, geodesic distances, and graph density; as well as the number of edges between pairs of groups in a special **Group Edges** worksheet.

After the groups have been computed or entered into the worksheets manually, users can display them in the visualization pane. When users select a group in the worksheet, all its member nodes are selected in the visualization. Likewise, for any nodes selected in the visualization users can select any groups in the worksheet that contain them using the ribbon menu. By default, groups are shown in their original expanded form based on the current layout algorithm, with categorical color and shape coding so as to distinguish them from each other. However, users can switch between the original expanded form and an alternate collapsed form for specific selected groups or all groups. This is done using the context menu in the visualization pane or the ribbon groups menu.

The default collapsed form for groups is a meta-node representation of a the same categorically coded shape with a plus sign inside to indicate its status (e.g., $\bigoplus$), sized proportional to the number of nodes the group contains and with any associated label next to it. However, the groups for our motifs use their representative glyphs that were described in the full paper. When a collapsed group is selected in the visualization pane it is also selected in the Groups worksheet, and its position in the visualization can be adjusted with the mouse. These collapsed representations are by default colored using the same categorical coloring as for the expanded version so the association between views can be easily identified. Through an option in the groups menu, users can switch from the default categorical colors and shapes to the underlying node attribute encodings the user specified. This updates all collapsed motifs so that they show the aggregate attribute information about the underlying nodes they represent.

REFERENCES

1. Grochow, J., and Kellis, M. Network motif discovery using Subgraph Enumeration and Symmetry-Breaking. In *RECOMB* (2007), 92–106.
2. Smith, M., Shneiderman, B., Milic-Frayling, N., Rodrigues, E. M., Barash, V., Dunne, C., Capone, T., Perer, A., and Gleave, E. Analyzing (social media) networks with NodeXL. In *ACM C&T* (2009), 255–264.
3. Tomita, E., Tanaka, A., and Takahashi, H. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science 363*, 1 (2006), 28–42.