ManyNets: An Interface for Multiple Network Analysis and Visualization

[1,2]Manuel Freire, [2]Catherine Plaisant, [2]Ben Shneiderman, [2]Jen Golbeck
[1]Universidad Autónoma de Madrid
28049 Madrid, Spain
manuel.freire@uam.es

[2]University of Maryland
College Park, MD 20742
{plaisant,ben,golbeck}@cs.umd.edu

ABSTRACT
Traditional network analysis tools support analysts in studying a single network. ManyNets offers these analysts a powerful new approach that enables them to work on multiple networks simultaneously. Several thousand networks can be presented as rows in a tabular visualization, and then inspected, sorted and filtered according to their attributes. The networks to be displayed can be obtained by subdivision of larger networks. Examples of meaningful subdivisions used by analysts include ego networks, community extraction, and time-based slices. Cell visualizations and interactive column overviews allow analysts to assess the distribution of attributes within particular sets of networks. Details, such as traditional node-link diagrams, are available on demand. We describe a case study analyzing a social network geared towards film recommendations by means of decomposition. A small usability study provides feedback on the use of the interface on a set of tasks issued from the case study.

Author Keywords
Network analysis, Exploratory analysis, Table interface, Interaction, Information Visualization, Graphical User Interface

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous

INTRODUCTION
The field of Social Network Analysis (SNA) has recently gained visibility as social networking sites have increased in relevance, numbers and participants. Sites with millions of users are now commonplace. Analyzing these networks helps our understanding of how people interact and communicate. Additionally, insights into how users interact within these systems are important to diagnose and improve them, and to develop better systems in the future.

Exploratory analysis of large networks, such as those found in SNA, often starts with visual overviews. Whole-network overviews, such as those generated with node-link diagrams or matrix representations, are hard to interpret, especially in the case of large, dense networks. Comparing these overviews to gain insights regarding a set of networks is even more difficult.

The need to visualize multiple networks at once arises naturally in many situations. For instance, analysts may wish to compare social networks for county managers in 3140 U.S. counties to gain insights into the strength of their collaborations. Subdivision of larger networks is also an important source of multiple networks; for instance, [23] describes the subdivision of an evolving social network by temporal slices, which can then be examined to locate temporal patterns or regions and periods change. In biological networks, the distribution of "motifs" (small patterns of connectivity) has been suggested as an indicator of their functional significance [22]. More generally, analysts can subdivide networks into closely-knit communities or clusters; and, in Social Network Analysis, the use of ego-networks to look at individual neighborhoods is well-established. Indeed, going one step further, analysts may need to compare groups of networks. They may want to discover if the ego-networks of one social network are similar to those of a different social network, and whether they exhibit similar temporal characteristics.

The main contribution of this paper is our approach to network analysis: it is the first approach that attempts to visualize many networks (up to several thousands) at once. We represent these groups of networks in a table, where each row represents a single network, generally a part of a larger network that has previously been split. Configurable columns contain user-defined statistics. Typical columns include link count, degree distribution, or clustering coefficient. In this sense, each row of this interface represents the fingerprint of a network, the combination of its cell values for each of the currently-displayed attribute columns. The use of a table allows easy comparisons between rows and columns, and can benefit from focus+context techniques such as those found in TableLens [29]. Our table visualization is tightly integrated with a node-link diagram network visualization tool, SocialAction [27], allowing on-demand network inspection. A second contribution is in the general

Submitted for review to CHI 2010.

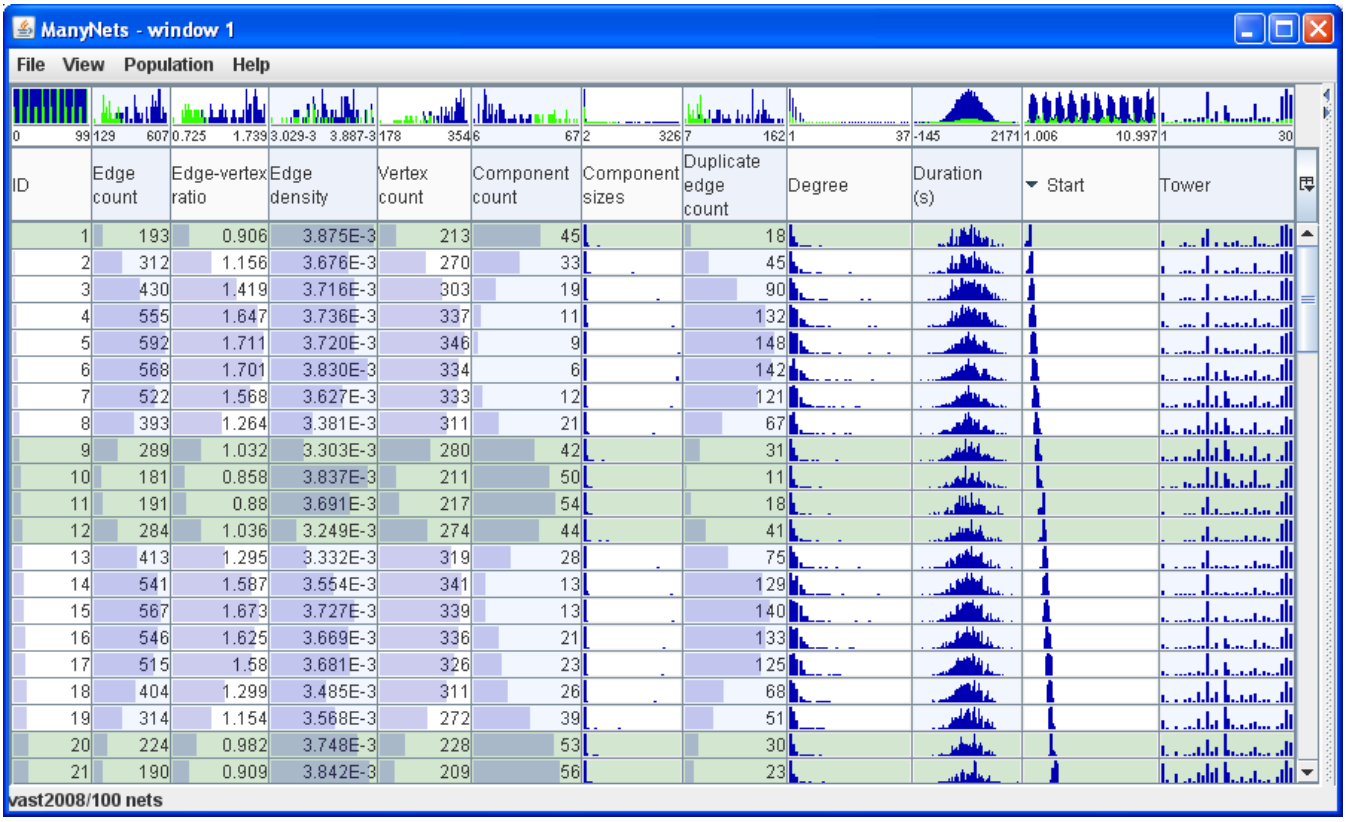

Figure 1. ManyNets displaying a time-sliced cell-phone call network. Each row represents the network of calls in a 5-hour period, with a 50% overlap with the previous row (10 rows cover an entire day). Rows with more than 40 connected components have been selected by dragging on the "Component count" column summary histogram, and are highlighted with a light-green background in the table. Each column summary highlights in bright green those values that correspond to currently-selected rows. In the left-most column summary, equally-spaced highlights reveal a temporal pattern. This synthetic dataset is from the VAST 2008 Challenge [15], and contains 400 nodes and 9834 links.

field of table-based interfaces, in the form of "column summaries". These are special cells, placed on top of the column headers, that provide an abstract of the contents of their column. The summaries also support direct user interaction, and reflect application state by highlighting values that correspond to currently-selected rows.

The next section presents our approach and describes the interface in general terms, using a temporal network dataset from the VAST 2008 Challenge [15] as a guiding example. After a section on related work, we describe an in-depth case-study using data from the FilmTrust social network [13]. Finally, we present the results of usability study with 7 participants.

DESIGN OF MANYNETS

We follow the Visual Information Seeking Mantra [32] to introduce the ManyNets approach to network visualization. The first step of this mantra calls for *Overview first*. Analysts can access two distinct types of overviews. First, rows themselves act as overviews of the networks that they represent. Scalar values, such as vertex (node) count and edge (link) count, are represented with horizontal bars. This allows easy comparisons between rows and columns, and in the case of Figure 1, can reveal several temporal patterns. When there is a distribution of values *within* each network, such as in the node degree column, the distribution is shown as a histogram in the corresponding cell. For example, in Figure 1, the histograms that represent the distributions of phone call durations can be seen to be roughly bell-shaped. On top of each column name, *column summaries* provide the second type of overview. Each column summary summarizes the contents of an entire column. In this sense, the set of column summaries is an overview of all the networks at once. Within summaries, we use miniature histograms to represent distributions of values for both scalar values and distributions. For example, in Figure 1, the column summary histogram for call duration shows a much smoother bell curve that considers all the distribution values in each network. In the same figure, a histogram also shows the distribution of scalar values, like Edge Count, over all networks.

Analysts can *zoom and filter* (the second step of the Mantra) collections of networks in several ways. Columns can be resized by dragging their left and right bounds to increase their size, and filtered by hiding them using the "column control" (displayed as a small button on top of the scrollbar). Sorting can be seen as a special type of filtering. All columns, even those that contain distributions, can be sorted on, and additional "multiple-column" sorting orders can be used. In a semantical sense, it is possible to "zoom" into a given

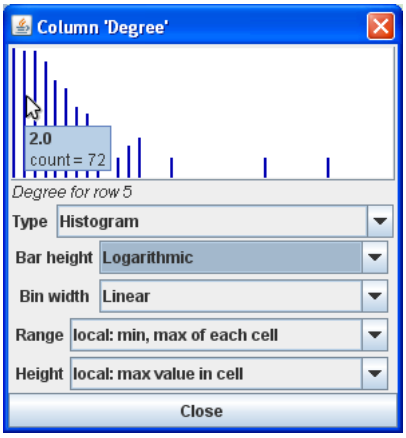

Figure 2. Details on demand for a degree distribution histogram. In all visualizations in ManyNets, hovering the pointer over a value displays a tooltip.

row or set of rows by slicing the network further, revealing finer-grained details. In the case of the dataset of Figure 1, it would be interesting to subdivide the network further into node neighborhoods ("ego networks"). This would allow us to focus on the neighborhood of node #200 (a caller that is present in many of the slices, but not directly visible in the table), described in the original dataset as an important lead (see [15]). Analysts can filter out uninteresting networks by selecting rows to be removed or retained. Selections can be specified in three ways: by clicking on the corresponding rows, through interaction with column summaries, or by specifying a custom filter. We expect analysts to be proficient in the use of spreadsheets; the expressions that are used as filters are similar in complexity to those used in spreadsheet formulas. In Figure 1, the expression `Column['Component count'] > 40` results in the currently-displayed selection; direct manipulation on the column summary for "Component count" could yield this selection filter.

Finally, *details on demand* are available by "mousing" (hovering the mouse pointer for a few seconds, without clicking) over any part of the interface: this will display a small tooltip, describing the value or values under the pointer. Larger, more detailed views of cell contents or column summaries can be obtained by right-clicking on them: this will display a pop-up window with a larger view, and controls that allow manipulation of view settings (Figure 2). Left-clicking any cell selects the corresponding row, and displays its contents in a detail panel. The same detail panel keeps track of the currently-selected rows. Last but not least, ManyNets is tightly integrated with SocialAction, and any row or set of rows can be opened in SocialAction for further inspection as a node-link diagram. SocialAction's interface provides its own facilities for visualizing network overviews, zooming and filtering, and providing details on demand (see [27] for details). For instance, SocialAction can use colors to highlight node or links based on attribute values supplied by ManyNets, and can perform interactive filtering based on these same attributes.

Selecting and Adding Columns

We now review selected ManyNets features in greater detail. Basic network statistics, such as link and node counts, link density, and component counts, are calculated and displayed by default. More computationally expensive statistics (e.g. network diameter) are only added if the user explicitly requests them, using the interface shown in Figure 3. Some of these statistics return distributions instead of scalar values; distributions are displayed as miniature histograms embedded in the main table. There can be four sources for columns. Topology-based statistics are those that can be extracted by traversing the network, without any additional domain knowledge. Domain-dependent link attributes and node attributes define two additional sources. In Figure 3, the Tower attribute refers to the particular cell-phone tower from which the call was made. Since these attributes are bound to links or nodes, they naturally result in distributions that will be represented as histograms. The final source for statistics is the user: columns can be defined by entering expressions in Python. For instance, the edge-vertex ratio (ratio of links to nodes) can be added using `Column['Edge count'] / Column['Vertex count']`. This results in a column indistinguishable from the default "Edge-vertex" ratio column (see Figure 4). Different types of references to columns can be inserted by using suitable keywords. Among others, `VarColumn['column']` inserts the variance of a cell in a column that contains distributions, and `MaxColumn['column']` inserts the maximum. A similar interface can be used to specify Python expressions for filters, selections, and sorting-column definitions.

Column Summaries

The column summaries, visible on top of each column of Figure 1, provide an overview of the values of their columns. For instance, in Figure 1, the distribution of phone call durations can be seen to follow a bell-shaped distribution, with a

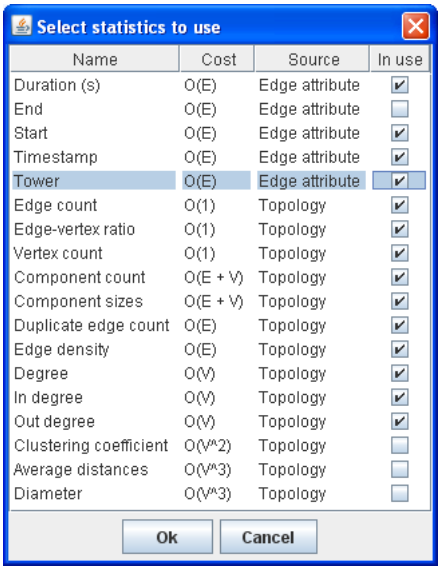

Figure 3. Selecting statistics. Choices correspond to columns displayed in Figure 1.

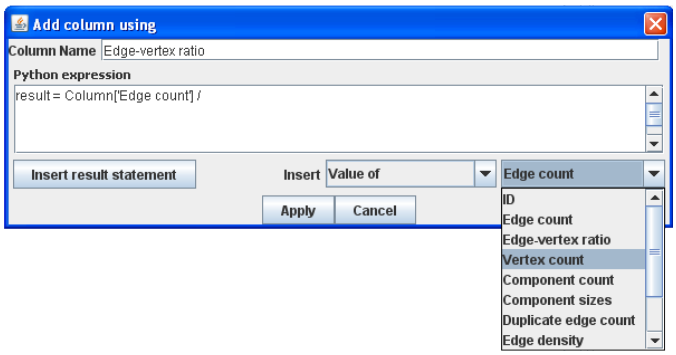

Figure 4. Adding a calculated column using Python. Abundant examples are available to users, lowering the effort of writing these expressions. Column references are substituted by their values before they are evaluated. They can be inserted using the drop-down combo boxes, or typed in directly.

minimum of -145 seconds (these "errors" were present in the dataset itself) and a maximum of 2171. Column summaries display the minimum and maximum column values under a histogram representing the distribution of values within the column, avoiding the need to query the summary (via tooltips) or sort on the column. This is especially useful when the number of rows is large. Histograms are used to represent column overviews; in the case of columns that already contain histograms in their cells, histograms of histograms are generated by adding the corresponding distributions.

Summaries also provide context-dependent information when rows are selected: selecting a set of rows highlights, in all column summaries, the contribution that the network represented by these rows make towards the overall distribution of values. In Figure 1, all slices with more than 40 connected components have been selected (green background). A temporal pattern can be observed in the column summary of slice IDs: the highlighted values (also green) are spaced regularly throughout the summary histogram (which uses the same order as the ID column). Not only do row selections affect column summaries – it is also possible to select rows by interacting directly with a corresponding summary. When dragging (clicking and holding while moving) the mouse pointer across any of these column summary histograms, all rows that have values in the corresponding value range will become selected. This can be used to quickly test for possible dependencies between statistics, and was the method use to select rows in Figure 1.

Ranking Columns

Sorting by a column is an expected feature in table-based interfaces. When the number of networks (and therefore of rows) is large, visually scanning for values is cumbersome and provides no guarantees of not having missed the targets. Sorting does not have these drawbacks. However, it is unclear how best to sort on a column that contains distributions. We use the distribution average by default, but it is possible to use different sorting criteria by adding additional columns to the table. To sort rows by the maximum degree, the user would add a computed column using the expression `max(Column['Degree'])`, and sort this recently-added column instead of the original. To sort on the difference between the maximum and the minimum degree, the expression would be `max(Column['Degree']) - min(Column['Degree'])`.

ManyNets also allows users to sort by several attributes at once (columns or user-defined expressions). For instance, we may wish to find networks that are both large *and* dense. Sorting first by size and then by density will not work when both attributes are not correlated. We address this concern by allowing users to add "multi-criteria sorting columns". Users can create these columns in a similar way to custom columns; instead of being asked for *one* expression to calculate, they have the option to specify several expressions. The resulting column displays the normalized "score" for each expression in a stacked bar (see Figure 5). The use of this additional column-type presents several advantages over inserting a conventional Python expression column. There is visual feedback on the degree to which each criterion has influenced the total sorting score; and scores are normalized, so that users do not need to take into account the different value ranges for each criterion. Additionally, it is possible to adjust the relative weights of each criterion by using the mouse pointer to "stretch" or "shrink" any of the score-bars. The effects of such adjustments are interactively propagated to all other bars.

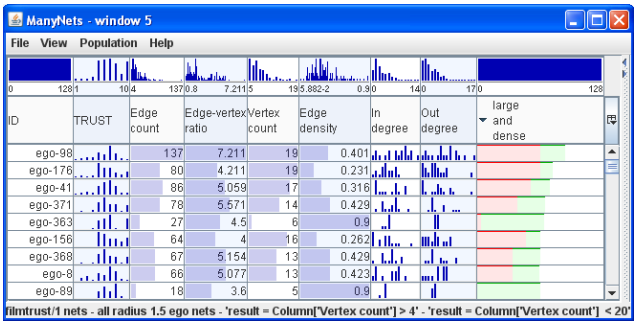

Figure 5. Using a multiple-criteria sorting column (right). In this case, rows have been sorted by node count (red bar) and link density (green bar). The status bar (bottom of figure) displays the steps used to obtain the current set of networks.

Generating Sets of Networks

So far, we have presented the interface from the point of view of a fixed set of networks. There are several ways to arrive at a set of networks to analyze. The most straightforward is to load several networks directly: there are many situations where users need to analyze and compare numbers of independent networks (e.g. citation networks in different scientific domains, or internal communication in different companies). It is also possible to decompose a single network into a set of networks. Finally, subdividing a set of networks into yet more networks also makes sense. For example, time-slicing several temporal networks would allow the temporal patterns of each to be compared. We propose four methods to derive interesting sets of networks from a previous set: ego-based, clustering, attributes, and feature-based subdivisions.

Ego networks are, because of their significance in Social Network Analysis, an important case. The definition of ego networks (a network centered on an individual) allows two degrees of freedom: the radius around the center that should be included, and whether or not connections between non-central neighbors should be included. Both are accepted as a parameters when splitting networks with ManyNets; the default is to split by radius 1, including neighbor-neighbor links. This is often referred to as "radius 1.5".

Clustering-based subdivisions take into account the regions of strong connectivity found in the network. There is considerable literature on graph clustering and community-finding algorithms (for instance, see [35] or [24]). We have implemented network subdivision into connected components, arguably the simplest of clustering algorithms.

In attribute-based subdivision, the network is divided according to the value of an attribute; each resulting subnet contains only a particular value or value-range for the chosen attribute. If a network has a temporal dimension, as is frequent in social networks, it is possible to "slice" the network along the time axis, adopting the terminology of [23]. Slices can be instantaneous, capturing a snapshot of the network at a given moment in time, or they can span time-windows (the value-range scenario), containing the union of all the slices within that time-window. Additionally, it is possible to generate the slices so that they partially overlap. Overlapping slices provide context for evaluating small changes, which would not be present with thinner and non-overlapping slices. A set of time-sliced networks sorted by time allows temporal patterns to be located with the tool, as illustrated in Figure 1.

Local feature-based subdivisions attempt to extract all the instances of a network feature. For instance, it is possible to build collections of all the connected diads or triads in a given graph; these are a particular case of so-called "network motifs". In the context of biological networks, motif distribution has been proposed as an indicator of functional significance [22]. Currently, ManyNets only supports the extraction of triads and diads. In the future, we may want to imitate a motif-specification interface similar to that found in [30].

RELATED WORK

Many systems have been developed to aid with social network analysis, and network analysis in general. Henry and Fekete [17] broadly classify them into those that provide frameworks that need to be extended via programming (such as JUNG[26], GraphViz [11] and Prefuse [16]), and those that can be directly used by means of graphical user interfaces (for instance Pajek [6], UCInet [7], Tulip [3] or SocialAction [27]). Some systems fall in-between, such as GUESS [2] and GGobi [34]. GUESS allows access to a host of filtering and highlighting capabilities through built-in Python scripting, and GGobi can interface with the R statistical package. The above tools rely almost solely on node-link diagrams. Matrix representations are less used, but better for performing certain tasks on dense networks [12]. An interface that is explicitly designed to use both visualizations is described in [17]. When comparing general network structure, however, both matrix and node-link representations make poor overviews: there are too many degrees of freedom, and even two structurally identical networks are likely to result in very different representations. A canonical representation based on network connectivity that addresses this problem is presented in [5].

Efforts to make large, dense node-link diagrams simpler by abstracting the parts of the network furthest from a user-defined focus can be found in [38] and [36], among others. Exploratory analysis is then driven by a top-down methodology, drilling into areas in order to access details. However, this approach makes it difficult to spot low-level patterns that may be distributed throughout the network: they would be hidden under layers of abstractions.

The use of several statistics to represent a network or a subnet is frequent throughout the literature. An example of a "network fingerprint" can be found in [37]; in this case, fingerprints refer to a small node-link diagram of an ego network and bar-charts for in-degree and out-degree during a user-defined time window. Additionally, canonical representations such as the portraits proposed in [5] can be used in the same fashion. Leskovec [20] and Kumar [19] use different types of plots to compare a handful of social networks, and their temporal evolution, to one another. The collection of plots for a single network can be seen as a "fingerprint", and has been used as inspiration in ManyNets.

We have not found any prior art where the network fingerprints are arranged in a table for easy comparison and sorting. However, tables of subnetworks that correspond to a query, without additional attributes, are used in [10]. Outside of the field of network analysis, the Hierarchical Clustering Explorer [31] presents a powerful interface for visualizing relationships between the multiple attributes (columns) of a large set of elements (rows); several features of HCE will be added to future versions of ManyNets (see section on Future Work). Tables are well-known for their information density, and much research has gone into building better table presentations or dealing with screen-size limitations; a well-known example is TableLens [29]. This research is gradually finding its way into commercial applications. For instance, recent versions of Microsoft Excel can display scalars with horizontal bars. To our knowledge, no system uses column summaries similar to those of ManyNets. Within column summaries, our use of overlays to highlight the values of currently-selected rows was partly inspired by [9].

The spreadsheet metaphor is a well-known success story in Human-Computer Interaction. It allows users to easily extend an analysis by allowing any cell to build on the contents of other cells; in this sense, we allow columns to build on the contents of existing columns. Most spreadsheet systems limit cell contents to single elements, generally numbers or strings. Several research systems have gone beyond these limitations, allowing display and computation with complex elements as cell values [28, 8]. In [8], different thumbnail-

sized views of a single evolving network are displayed as node-link diagrams in individual cells. A different approach is that of NodeXL [1], where a commercial spreadsheet has been extended to provide single-network visualization, directly exposing tables of nodes and edges to users as spreadsheet pages.

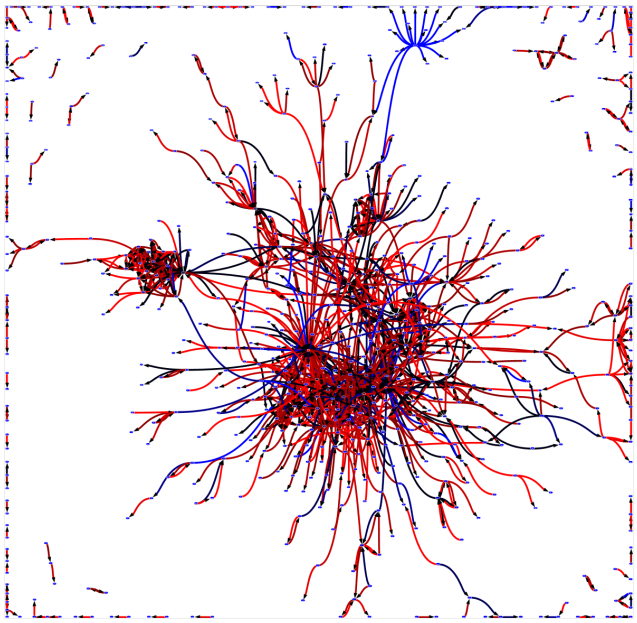

Figure 6. Overview of the FilmTrust network (described below), using SocialAction. Directed links are colored according to their "trust" attributes, with red highest and blue lowest.

CASE STUDY: FILMTRUST

Trust networks are social networks augmented with trust information. Links are directed and labeled with the trust that the source user has for the sink user. Trust can be binary or scored on a discrete or continuous scale. These networks have been used in the computing literature in two ways. First, trust inference is an important problem; when two users are not directly connected, trust inference algorithms approximate a trust value between them (see [14] for a survey). Secondly, trust has been used to improve the ways users interact with information. It is a tool for personalization and has been applied to generating movie recommendations [13, 25], to assigning posting permissions in online communities [21], to finding safe mountain ski routes [4], to improving the performance of P2P systems [18], and to sorting and filtering blog posts [21].

For this analysis we used the trust network that underlies the FilmTrust website [13]. FilmTrust is an online social network where users rate and review movies and also rate how much they trust their friends about movies. Trust is assigned on an integer scale from 1 (low trust) to 10 (high trust). Friendship and trust are not required to be mutual relationships; user A can add user B as a friend without B's approval or reciprocation. Trust ratings are also asymmetric; users' trust values for one another are independent and can be very different. The FilmTrust network has been online since 2005 and contains 666 nodes and 1396 links with one giant component and 69 much smaller components. Figure 6 is an overview of the network generated by loading it into ManyNets and then displaying it with SocialAction. A few interesting patterns are visible, but there is no way to analyze the dense central cluster without decomposing it into smaller networks.

Analysis

The analysis was performed by one of the authors, with a long experience in the domain of trust networks. First, she examined the node-link diagram. The next step was to split the network into all "1.5" ego networks (each containing a center node, its immediate neighbors, and all links from center to neighbors or between neighbors), and examine the results within the tool. A look at trust distributions reveals a high rate of "maximum" values, and generally low rates of low-to-mid values, as predicted by previous research in ratings. When sorting the ego networks by size, she observed that large ego networks are overrepresented among the low ID numbers (see highlights in the ID column summary in Figure 7); this is to be expected: early adopters have had more time to create connections, and newcomers are more likely to connect to well-connected members (this phenomenon is often called "preferential attachment").

The analysis was split into two sessions of 4.5 and 3 hours. During the first session, she mainly used sorting and filtering to browse for insights in distributions, and to look for "interesting" networks. For instance, user #345 (second-to-last row of Figure 7) stands out, with connections to 16 neighbors, all of them strangers to each other, and the minimum trust value assigned to each of these neighbors.

After sorting by trust and filtering out the smallest ego networks, it became apparent that few ego-networks of more than 5 nodes had high trust values throughout; this prompted a hypothesis: does trust tend to be symmetric between users? This is most easily observed in cliques, which can be located by filtering by edge density. All 1.5 ego-network cliques were of size 2 (dyads) except for 2 triplets.

Examining the trust differences showed very similar trust values in these little pairs. However, by starting with ego-networks, only pairs of users disconnected from the main component were shown; this did not represent other dyads present in the graph. For the next analysis session, we added the possibility to split a network by dyads and triads, and a mechanism to visualize many network-rows simultaneously within SocialAction. Her next step was a comparison of trust in isolated pairs with trust in all pairs of nodes in the network.

While trust assigned by isolated pairs (from ego networks) follows the same distribution as trust assigned among all pairs, isolated pairs seem to assign fewer middle range values. An interpretation is that participants either trust each other's criteria regarding films or they do not, with little need for middle ground. The same experiment, comparing 3-vertex ego networks to all triplets, yields stronger results: isolated groups of three tend to have more trust with each

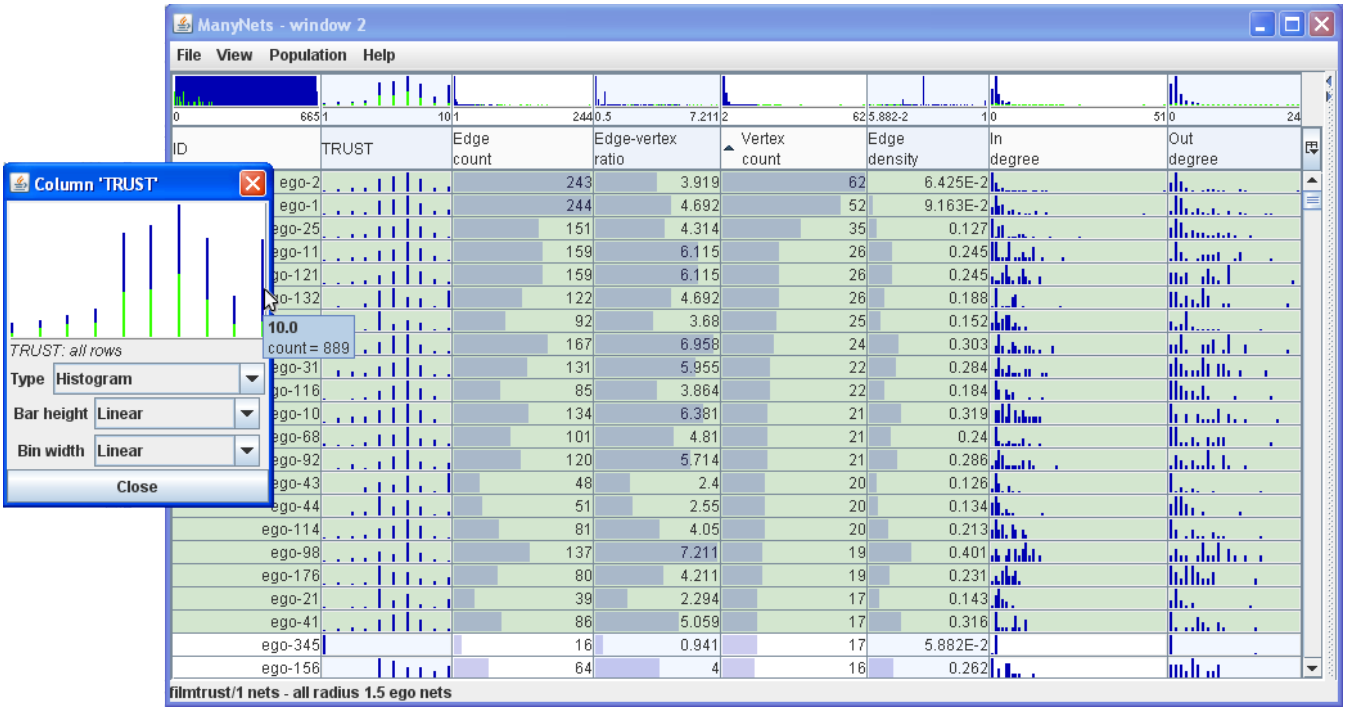

Figure 7. The "1.5" ego networks in the FilmTrust dataset ranked by size. The 20 largest have been selected, and the pop-up window is displaying a larger version of the summary column for the trust attribute. Notice the different distribution in trust among the selected rows as compared with the global distribution. The summary for the ID column shows that the selected rows are unevenly distributed, due to preferential attachment.

other than do groups of 3 embedded within larger components (see Figure 8).

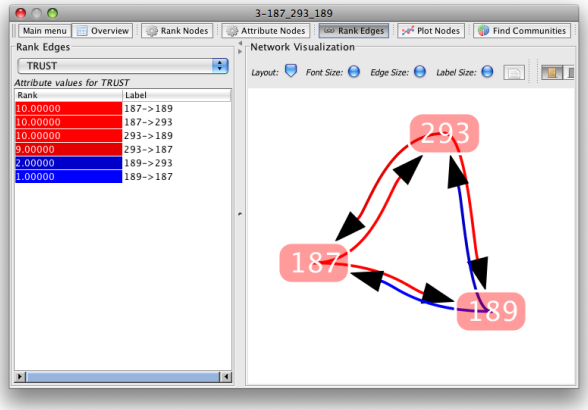

Figure 8. A triplet with asymmetric trust values. Red arrows indicate high trust, blue arrows indicate low trust.

This lead to a similar question about how trust was distributed in tightly clustered groups. Indeed, there appears to be no correlation between trust and edge density, except when the group is isolated. By selecting the top-20 largest ego-networks (Figure 7), the trust distribution can be seen to be significantly lower (especially regarding high ratings) than the general distribution.

A second hypothesis was to test the "transitivity" of trust. If *A* trusts *B* and *C* very highly, trust between *B* and *C* might be expected to be high as well. In this analysis, she noted the need for sorting by distribution "similarity", prompting us to develop of additional statistics for user-defined columns (it is now possible to, for instance, to create a column for the trust variance). Using a combination of the table view and SocialAction visualization, she found that trust did not follow any sort of transitivity. Trust values between *B* and *C* in this example varied widely and independently from the values assigned by *A*.

Outcome

Our analysis of the FilmTrust network began with two major questions that follow from conventional wisdom. First, since trust reflects a similarity in tastes, do nodes trusted by the same people trust one another? Secondly, and a related point, does trust increase in strongly connected sub-groups? As we pursued answers to these questions, we made several additional insights. We found that conventional wisdom does not hold in this trust network. Two nodes that are trusted by a person do not necessarily trust one another (see figure 8 for an example). Secondly, tightly clustered groups do not have higher levels of trust than the general population; within these groups trust tends to follow the same distribution as it does overall. These results have significant implications for research into trust networks and will be directly applicable to improving trust inference algorithms, building trust network simulators, and designing better trust-based interface enhancements.

In a matter of a few hours our trust network expert was able

to learn to use the tool, answer dozens of questions, and make significant discoveries in a dataset she was already familiar with. While results of case studies are difficult to generalize, we believe, in the spirit of [33], that they are a valuable tool when evaluating complex analytic tools such as ManyNets.

USABILITY STUDY

We carried out a pilot usability study with 7 participants to identify opportunities for improvement. All had computer science background but no previous network analysis experience. Our goal was to understand the use of the interface by novice users, and to guide the development of training materials for future users. We asked our participants to think aloud while performing actions, and to describe any problems as they encountered them. Before attempting the actual tasks, participants were asked to watch an 8-minute video describing the problem domain and the general use of the tool. Participants were then asked to familiarize themselves with the tool by performing a set of common operations (split a network, rank columns, and write a filter), and were provided with reference documentation that they could use at any time during the actual tasks.

Our tasks were based on those that our analyst performed when studying the FilmTrust dataset. Using this same dataset, participants were asked to:

1. Find a pair of nodes that have rated each other with the largest difference in their trust values, and estimate how many such pairs can be found in the whole network.

2. Find the densest 1.5 ego network with more than 3 nodes.

3. Find the 1.5 ego network with more than 3 nodes that has the lowest average trust among the nodes.

4. Find a group of three nodes (not necessarily an ego network) with at least one high trust rating (> 8) and at least one low trust (< 3) rating.

5. Describe any difference in the the distribution of trust values in 1.5 ego networks with 4 nodes compared to the distribution among all 1.5 ego nets.

6. Find a group of at least three nodes where one node has given a rating > 5 to a neighbor and that neighbor's reciprocal rating is < 5.

Results

Early participants required around 30 minutes of hands-on experience before becoming proficient with the tool, as evidenced by faster task execution and greater feeling of control when faced with the last tasks. This prompted us to extend our initial training, and underlines the need for providing real-world examples of tool use in training materials, as opposed to context-free descriptions of tool functionality. Early participants also requested more context on the significance of our choice of tasks for real-world analysis; a motivation section was provided to later participants.

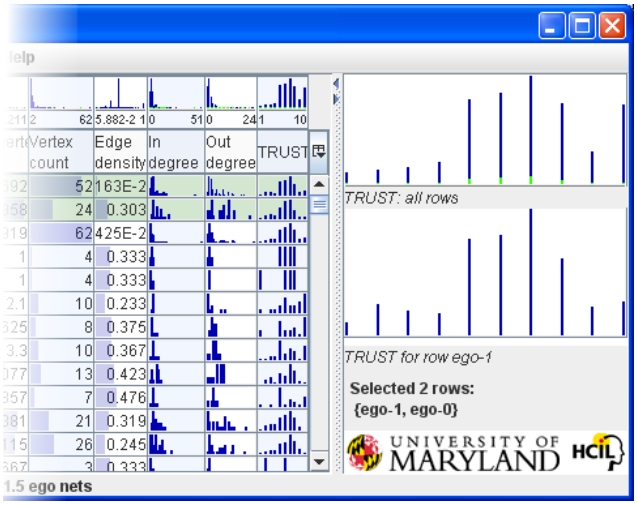

Figure 9. Closeup of the details-on-demand panel, shown after selecting the first two rows of by dragging on their trust values.

Participants chose different ways of accomplishing the same tasks. For instance, some participants preferred to filter networks into different views, and then work with the filtered views, while others added user-defined columns and ranked on the new column's values. Once they found a satisfying approach, participants tried to extend it to any further tasks, even though alternatives would have proven more effective. This suggests that future training materials should expose users to different styles of analysis, highlighting the most effective methods for different tasks.

Several usability problems were identified. Most participants preferred the use of a details-on-demand panel that was always visible (Figure 9) to the pop-up details-on-demand dialog (Figure 2) which had to be manually dismissed. As a result, we are considering merging both features. Several participants found the use of histograms for cells that contained less than 3 values to be confusing. Dragging on column summaries proved to be difficult to do accurately; one participant suggested allowing the drag to be performed on a detail view of that same summary, which could additionally allow several "selection regions" to be specified. Finally, as expected, participants encountered frequent problems while learning the syntax of the expressions used when filtering or specifying user-defined columns, although they became more proficient with time. In this regard, two participants asked for spreadsheet-like assistant to build and debug formulas. Multi-criteria sorting columns were seldom used, due to their poor discoverability; however, the one user that successfully used them commented positively on their functionality, and quickly saw ways of using them for multiple tasks.

Three participants asked for the possibility of accessing the individual components of networks. In particular, for task 6, it seemed more natural to access the attributes of individual nodes by querying for them directly, instead of having to deal with a distribution of values as seen from a network perspective. Addressing this concern would require us to allow direct access to the underlying data, which could

again be represented as a table of nodes or edges, similar to those used in NodeXL [1]. A more involved solution would require adding a pattern-matching language, so that filter and user-defined column expressions could refer to particular nodes that met a pattern, and then access the attributes of "matched" nodes or edges. This would allow within-network queries, and would open the way to user-defined pattern matching and ranking.

Finally, one participant suggested the use of ManyNets for datasets that were not networks, arguing that column summaries, the ability to generate filtered views, and the availability of spreadsheet-like user-defined columns could be useful to inspect any tabular data.

FUTURE WORK AND CONCLUSION

One of the most natural behaviors when confronted with a set of elements is to look for similarities and differences. It would be useful to calculate and display intra-group similarity, based on user-defined criteria (for instance, graph edit distance). Graphically displaying similarities (as seen in HCE [31]) would allow identification of clusters of related networks, and provide interesting overviews of their internal structure. Clusters could then be used as criteria for network subdivision, allowing additional observations to be made.

We plan to add support for bipartite networks; this would allow us to compare the trust values in the FilmTrust network with the actual ratings that the users gave to the films in the system's database. Exposing film ratings together with trust values may help to understand the way trust is assigned. Additionally, our analysis were done considering only a recent snapshot of the network. Using timestamp-data may provide further insights.

We have described ManyNets, an interface to visualize sets of networks, and presented a case study where the interface was used to make new discoveries in the FilmTrust network dataset. This case study demonstrated the effectiveness of our interface in a representative Social Network Analysis task, leading to insights and suggesting new research avenues within trust networks. The usability study has identified several areas for improvement, and suggested additional features. Many of the comments received during both study have are finding their way into ManyNets. We believe that ManyNets opens new ways of analyzing large social networks, and other collections of complex networks.

ACKNOWLEDGMENTS

The first author is supported by a MEC/Fulbright Scholarship (Reference No. 2008-0306). Partial support for this research was provided by Lockheed Martin Corporation. We particularly thank Brian Dennis for providing motivating scenarios for ManyNets and feedback on our designs, and Cody Dunne and John Guerra for technical insights.

REFERENCES

1. Nodexl website. `http:/www.codeplex.com/NodeXL`. Last visited, 2009.
2. E. Adar. Guess: a language and interface for graph exploration. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 791–800, New York, NY, USA, 2006. ACM.
3. D. Auber. Tulip — A huge graph visualization framework. In M. Jünger and P. Mutzel, editors, *Graph Drawing Software*, Mathematics and Visualization, pages 105–126. Springer-Verlag, 2004.
4. P. Avesani, P. Massa, and R. Tiella. A trust-enhanced recommender system application: Moleskiing. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1589–1593, New York, NY, USA, 2005. ACM.
5. J. P. Bagrow, E. M. Bollt, J. D. Skufca, and D. ben Avraham. Portraits of complex networks. *EPL*, 81:68004, Feb. 28 2007. Comment: 6 pages, 9 figures.
6. V. Batagelj and A. Mrvar. Pajek-Program for Large Network Analysis. *Connections*, 21(2):47–57, 1998.
7. S. Borgatti, M. Everett, and L. Freeman. Ucinet for windows: Software for social network analysis. *Harvard: Analytic Technologies*, 2002.
8. E. H. Chi and S. K. Card. Sensemaking of evolving web sites using visualization spreadsheets. In *Proc. IEEE Symposium on Information Visualization (Info Vis '99)*, pages 18–25, 142, 1999.
9. H. Dawkes, L. A. Tweedie, and B. Spence. Vicki: the visualisation construction kit. In *AVI '96: Proceedings of the workshop on Advanced visual interfaces*, pages 257–259, New York, NY, USA, 1996. ACM.
10. P. Durand, L. Labarre, A. Meil, J.-L. Divo1, Y. Vandenbrouck, A. Viari, and J. Wojcik. Genolink: a graph-based querying and browsing system for investigating the function of genes and proteins. *BMC Bioinformatics*, Jan. 17 2006.
11. J. Ellson, E. Gansner, L. Koutsofios, S. North, and G. Woodhull. Graphviz-open source graph drawing tools. *Graph Drawing*, 2265:483–485, 2001.
12. M. Ghoniem, J. Fekete, and P. Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135, 2005.
13. J. Golbeck. Generating predictive movie recommendations from trust in social networks. In *Proceedings of the Fourth International Conference on Trust Management*, pages 93–104. Springer, 2006.
14. J. Golbeck. The science of trust on the web. *Foundations and Trends in Web Science*, 2008.
15. G. Grinstein, C. Plaisant, S. Laskowski, T. OConnell, J. Scholtz, and M. Whiting. VAST 2008 Challenge: Introducing mini-challenges. In *IEEE Symposium on Visual Analytics Science and Technology, 2008. VAST'08*, pages 195–196, 2008.

16. J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM.

17. N. Henry and J. Fekete. MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677–684, 2006.

18. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM Press.

19. R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, editors, *KDD*, pages 611–617. ACM, 2006.

20. J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–470, New York, NY, USA, 2008. ACM.

21. R. Levien and A. Aiken. Attack-resistant trust metrics for public key certification. In *7th USENIX Security Symposium*, pages 229–242, 1998.

22. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298:824–827, 2002.

23. J. Moody, D. McFarland, and S. BenderdeMoll. Dynamic network visualization. *American Journal of Sociology*, 110(4):1206–1241, 2005.

24. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, Aug. 11 2003.

25. J. O'Donovan and B. Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174, New York, NY, USA, 2005. ACM.

26. J. O'Madadhain, D. Fisher, P. Smyth, S. White, and Y. Boey. Analysis and visualization of network data using JUNG. *Journal of Statistical Software*, 10:1–35, 2005.

27. A. Perer and B. Shneiderman. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *CHI '08: Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 265–274, New York, NY, USA, 2008. ACM.

28. K. W. Piersol. Object-oriented spreadsheets: the analytic spreadsheet package. In *OOPLSA '86: Conference proceedings on Object-oriented programming systems, languages and applications*, pages 385–390, New York, NY, USA, 1986. ACM.

29. R. Rao and S. K. Card. Exploring large tables with the table lens. In *CHI 95 Conference Companion*, pages 403–404, 1995.

30. F. Schreiber and H. Schwobbermeyer. MAVisto: a tool for the exploration of network motifs. *Bioinformatics*, 21(17):3572–3574, Aug. 22 2005.

31. J. Seo and B. Shneiderman. Knowledge discovery in high-dimensional data: Case studies and a user survey for the rank-by-feature framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):311–322, May/June 2006.

32. B. Shneiderman. The eyes have it: a task by data type taxonomy for invormation visualization. In *Proceedings of the IEEE Workshop on Visual Language*, pages 336–343, 1996.

33. B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *BELIV '06: Proceedings of the 2006 AVI workshop on BEyond time and errors*, pages 1–7, New York, NY, USA, 2006. ACM.

34. D. F. Swayne, D. T. Lang, A. Buja, and D. Cook. GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43(4):423–444, 2003.

35. S. Van Dongen. Graph clustering by flow simulation. Master's thesis, Center for Mathematics and Computer Science (CWI), 2000.

36. F. van Ham and J. J. van Wijk. Interactive visualization of small world graphs. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, pages 199–206, Washington, DC, USA, 2004. IEEE Computer Society.

37. H. Welser, E. Gleave, D. Fisher, and M. Smith. Visualizing the signatures of social roles in online discussion groups. *The Journal of Social Structure*, 8(2), 2007.

38. P. C. Wong, H. Foote, P. Mackey, G. C. Jr., H. J. Sofia, and J. Thomas. A dynamic multiscale magnifying tool for exploring large sparse graphs. *Information Visualization*, 7(2):105–117, 2008.