"Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest

Frank van Ham and Adam Perer

Abstract—A common goal in graph visualization research is the design of novel techniques for displaying an overview of an entire graph. However, there are many situations where such an overview is not relevant or practical for users, as analyzing the global structure may not be related to the main task of the users that have semi-specific information needs. Furthermore, users accessing large graph databases through an online connection or users running on less powerful (mobile) hardware simply do not have the resources needed to compute these overviews. In this paper, we advocate an interaction model that allows users to remotely browse the immediate context graph around a specific node of interest. We show how Furnas' original degree of interest function can be adapted from trees to graphs and how we can use this metric to extract useful contextual subgraphs, control the complexity of the generated visualization and direct users to interesting datapoints in the context. We demonstrate the effectiveness of our approach with an exploration of a dense online database containing over 3 million legal citations.

Index Terms—Graph visualization, network visualization, degree of interest, legal citation networks, focus+context.

1 Introduction

Visualizations of very large graph datasets typically aspire to present the user with an overview of the entire graph, so no information is missing and the data can speak for itself. However, this lofty goal often results in visualizations that focus exclusively on providing a global view of the structure of the graph. While topology-based graph visualizations ideally may allow analysts to deduce central actors or clusters in a graph, in practice the density and sheer size of many graphs make it hard to create an effective visual representation of the whole graph. Visualizations that faithfully try to render all the data often suffer from cluttering and make it difficult for users to accomplish even very basic tasks such as following edges or counting in-degree of nodes [19]. Structural clustering techniques can reduce the complexity of the data under analysis, but often result in abstract high-level diagrams in which the semantics of the clusters are not always clear.

Users that wish to obtain overviews of very large graphs are typically trying to learn something about a particular dataset. For example, a social scientist may want to identify cliques in a very large social graph and see how they interact, while a software architect may want to deduce the decomposition of a program from its call graph. These use cases typically deal with global patterns like clusters and outliers. However, there is also a very large class of users that frequently deal with large network data but are not interested in global patterns in this data. Instead they are trying to learn something more about a particular datapoint in the dataset and how this point relates to the rest of the data. As a concrete example, financial fraud analysts typically try to understand the pattern of connections associated with a specific fraudulent bank account, while programmers need to understand the dependencies of a specific piece of code if they want to understand the impact of potential changes.

A second argument against presenting users with global overviews is of a more practical nature. Typically, multiple analysts look at a single large dataset that is being maintained at a single centralized location. Many of these graph datasets are huge even by today's standards, and transferring them from a centralized server to multiple client machines is not an option. Apart from that, even the visualization designers might not have access to the full data because of privacy reasons or other access restrictions, making it impossible to pre-compute overviews for these giant graphs.

For these two reasons, we explore an alternative to the traditional 'overview, zoom, details on demand' browsing model [18], which can be loosely characterized as 'search, show context, expand on demand'. In this model, users pick a particular datapoint as a focus for analysis and the system then computes and displays an 'optimal' relevant context given the users' current interests. Users can then direct the visualization system to expand this context in a direction he or she deems interesting. This model is somewhat similar to the idea of "Plant a seed and watch it grow" [12]. However, whereas that model relied solely on topology for context, our approach is extensible to include other types of relevant context based on inherent attributes, associated content, and user interactions.

To determine this relevant context, we extend the well-known concept of degree of interest (DOI) [4] from trees to general graphs. By using a DOI function to assign a measure of relevance to each node in the graph, we can extract a maximal interest subgraph around the point of interest. Users can then interact with this initial subgraph and expand it in any direction. Our degree of interest functions use both embedded attributes and topology of the graph, as well as recorded actions of users. As user tasks are often exploratory and not well-defined, users have full control over the interest sources and can adapt them during exploration. We demonstrate our approach on approximately 15 gigabytes of legal documents containing information on all US federal and Supreme Court cases to date, resulting in a citation graph of over 300,000 nodes and 3.3 million edges. Our system allows multiple analysts to simultaneously explore this online citation graph through an interactive web-based client. Concretely, we claim the following contributions:

- An adaptation of DOI from tree to graph datasets.
- An augmentation of DOI to include inferred interest from the users' search process.
- Methods for diffusing interest over an entire graph to mitigate local minima.
- Novel visualization and interaction techniques that use DOI functions to reduce complexity of very high degree nodes and to provide visual clues to help guide users to hidden, yet interesting results.

- *Frank van Ham is at IBM-ILOG Research in Gentilly, France and can be reached by e-mail at frankvanham@fr.ibm.com*
- *Adam Perer is at IBM Research in Haifa, Israel and can be reached by e-mail at adamp@il.ibm.com*

Manuscript received 31 March 2009; accepted 27 July 2009; posted online 11 October 2009; mailed on 5 October 2009.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org .

Our paper begins with a reflection on related work, where we categorize and discuss Degree of Interest based research as well as current approaches to large graph visualization. In Section 3 we explain our methods for adapting DOI to graphs and our resulting implementation. Section 4 demonstrates our system on a legal corpus of documents and highlights a few examples of how our approach succeeds. Finally, we discuss the positive and negative aspects of DOI graphs and suggest future work in Section 5, while we conclude in Section 6.

2 Related Work

This section discusses previous work in the area of degree of interest visualization and summarizes the state of current large graph visualization techniques that do not focus on displaying a structural overview.

2.1 Degree of Interest functions in Visualization

Furnas first introduced the concept of Degree of Interest (DOI) in [4], realizing that information items have different levels of importance to different observers, depending on their viewpoint. Instead, an observer's initial point of interest (or focus) can be used to compute a numerical value for each data item that indicates its degree of interest. These values can then be used to create compact elided or abstracted views of the full data, by only displaying information items above a certain 'interestingness' threshold. Furnas' original paper showed how these ideas can be applied to trees or calendars and he later revisits them in [5] offering theories of the importance of DOI for real user's tasks.

Card and Nation used this concept to implement DOITrees, a tree visualization that uses DOI calculations and focus+context interaction techniques [1]. Heer and Card [9] extends this work with an efficient, space-constrained, multi-focal tree layout.

Although DOI functions are well established to display trees, applications to general graphs are limited. Two notable techniques [7][8] use an explicit degree of interest function to create multilevel abstractions of a whole graph by more aggressively clustering nodes further away from a focal node. In practice however, they tend to generate very abstract views where the meaning of a cluster of nodes to the user is not always clear and the actual clustering structure changes substantially with adjustment of the focus.

Graphical distortion techniques are related to the concept of degree of interest, but instead of explicitly assigning an interest value to nodes they perform information compression in screen space. That is, nodes further away from a predefined focus are assigned less screen space, resulting in a graphical fisheye [17]. Other approaches do not necessarily assign additional screen space, but use visual cues such as font size and color to focus user attention on interesting items [2]. Although graphical information compression techniques can be useful in some cases, they still require access to and rendering of the entire graph, which is impractical for our case.

2.2 Large graph visualization

As mentioned in the introduction, many graph visualization techniques try to present the user with a holistic view of the entire graph. This typically involves very efficient layouts, multiscale clustering techniques or matrix representations. Although providing a structural overview of the graph is a laudable goal, there are many cases where the user is simply not interested in a global view of the whole graph, but wants to solve a particular concrete task on the graph instead. Thus, we focus the discussion here to related work that does not focus on showing the overall graph topology. We discern three basic categories that all use different approaches to reduce the complexity

2.2.1 Attribute based abstractions

The first category of approaches proposes to use the node attributes in multivariate graphs as parameters for abstraction. PivotGraph [23] compresses multivariate graph in two dimensions by rolling up nodes into metanodes if they have the same attribute value for a particular dimensions. Semantic substrates [19] use node attributes to provide a basic layout of the graph based on space subdivision and then superimpose the edges. Approaches in this category have the advantage that one can more easily make statements on higher level features as these relate directly to the application area. Unfortunately they generally do not provide detailed information on the particular connections of a single node.

2.2.2 Contextual views

Closest to our proposed approach, contextual views allow the user to pick a point of interest in the graph, and show the immediate context around that point. Both Touchgraph [22] and Palantir [14] offer commercial graph visualization components that allow users to select a focal node and explore the surrounding subgraph. TreePlus [12] allows users to explore graphs using an enhanced tree layout, which users often find more comprehensible than a graph layout. These approaches perform relatively well when the average degree of nodes in a graph is small, but struggle with very high degree nodes.

2.2.3 Computational approaches

By first computing network metrics on the graph, we can compute relevant subgraphs or detect outliers in the graph, without having to visualize it first. SocialAction [15] uses attribute ranking and coordinated statistical views to allow users to use metrics from social network analysis to isolate important nodes, clusters and outliers in graphs. NodeXL [20] also follows this approach by integrating the statistical and visual capabilities of Excel with graph visualizations. Although these type of approaches allow the user to target specific features when exploring, the computational complexity of the metrics involved prohibits application to truly large graphs.

3 Degree-of-Interest Graphs

In this section we show how we can extend the well-known concept of Degree of Interest (DOI) from trees to graphs. In Section 3.1 we will outline our proposed interest function and in Section 3.2 we show how we can use this interest function to both compute interesting subgraphs and control the complexity of the visualization. Finally, in Section 3.3 we explain how we can use these algorithmic pieces to set up a system that lets us browse massive graphs from a remote client.

3.1 Extending DOI to graphs

In his seminal paper [4], Furnas defined a generalized degree of interest function for data items by discerning two components: An a priori interest function that defines the general importance of a data item x irrespective of the user's current interest and a distance function in which the interest of an item depends on the currently selected focus node y. This results in a two part interest function that can be expressed as:

$$\mathrm{DOI}(x \mid y) = \alpha \cdot \mathrm{API}(x) + \beta \cdot \mathrm{D}(x,y)$$

In other words, the total interest of a point x given a focus point y is a linear combination of an a priori interest function API and a distance function D. Furnas showed that for many types of data both functions have natural definitions. As an example, for trees API(x) can be defined in terms of the distance to the root. Distance can then be directly mapped to graph distance in the tree. By effectively eliding nodes below a certain interest threshold from screen we can create comprehensive displays of potentially very large information structures.

There exists a natural way to adapt these basic definitions to general graphs: API(x) can be computed in terms of structural graph properties or node attributes, such as node degree or the value of a particular node attribute. D(x,y) can be directly mapped to minimal graph distance. Note that the distance between two nodes can again depend on multiple factors such as edge weight or other edge attributes. Thus, it is necessary to define a separate disinterest function $EI(e,x,y) > 0$ for each edge, where higher values indicate less interest in following a link e between nodes x and y. This function can be used to define the path length between two arbitrary nodes in the graph. In theory, we can then use this adapted DOI function to extract a maximal interest subgraph from a very large graph given an initial focal node y. In practice, however, two problems remain.

The first is how users actually pick an initial focal node y in a potentially very large graph if users do not know a node's exact name or identifier. Although this process is largely ignored in previous work on degree of interest based visualizations, we emphasize it as the *Search* component of our "Search, Show Context, Expand on Demand" model.

Traditional information visualizations typically rely on opportunistic 'browsing' of high level (visual) overviews of the data to identify potentially interesting starting points for analysis. In the absence of such a full data representation, users must search the entire node set for items of interest that satisfy certain parameters and then select one node from the result as focus. Note that this can be realized in different (visual and textual) ways. In a simple case, the user specifies a textual search term and the system returns a list of nodes matching the term. A search interface that supports faceted search is another option, if structured attribute information is available. Another approach is a visual search interface that presents users with a simplified graphical representation (e.g. a scatterplot) of all items and allows a selection of results graphically.

In fact, the actual search parameters z that users specified give important hints on the interest a user assigns to particular nodes. We therefore propose an inclusion of a third term in the DOI function that captures this fact. The so-called User Interest (UI) function can be expressed as the interest information that is known before the user picks a focal node. Examples of this information include the search score of a node for a particular textual query or the facets used in a faceted search. More explicitly, if users performed a faceted search for items that, for example, are in a particular price range and have a particular feature, the DOI algorithm needs to assign higher relevance to items that match one or more of those criteria. This results in the following function definition:

$$DOI_{naive}(x \mid y,z) = \alpha \cdot API(x) + \beta \cdot UI(x,z) + \gamma \cdot D(x,y)$$

A second problem is that the interest function over graphs has many local maxima. That is, there are many potentially interesting nodes, surrounded by non-interesting nodes. Typically, in previous work on trees, DOI functions are strictly nested, that is, $DOI(x)$ of any node x in a subtree of r will never be larger than $DOI(r)$. This allows for fast recomputation of DOI values [9] and guarantees connected trees when thresholding on DOI. However, there is no such guarantee in graphs. Applying a simple DOI threshold function to the data will then yield a potentially large number of disconnected subgraphs, while a local search algorithm starting from the focal node will not be able to reach the high interest node because it has no global knowledge of the surrounding structure (see Fig. 1).

We can address this problem by slightly modifying the API and UI functions, such that the final interest value for a node x not only depends on its intrinsic interest, but also on the intrinsic interest values of its neighbors $N(x)$:

$$API_{diff}(x) = \max(API(x), \delta \cdot \max(n \in N(x) : 1/EI(e,x,n) \cdot API_{diff}(n)))$$

In other words, the interest of a node depends on the maximum of its own interest values and a fraction of its highest interest neighbour. In a sense, we are diffusing the interest values over the entire graph. The parameter δ ($0 \le \delta < 1$) determines the diffusion factor, where values closer to 1 increase the diffusion. By including the Edge Interest function we can optionally control the direction and amount

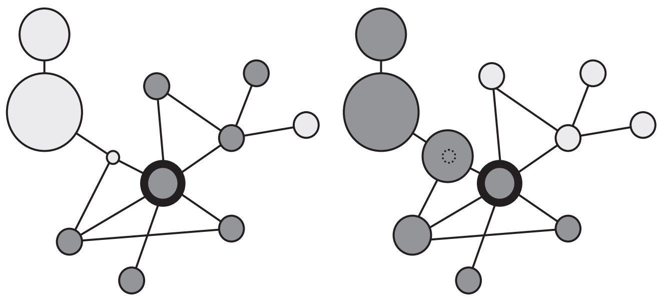


Fig. 1. By diffusing interest values over the network, we can use a greedy local search heuristic to find maximal interest subgraphs (dark grey) when starting from the focal node (circled)

of interest diffusion based on the interestingness of the connecting edge. We can compute this recursive function by recomputing the API for nodes in $N(x)$ only when $API(x)$ changes; the max quantifier limits the number of recomputations needed to a small number in practice, depending on the value of δ. An analogous definition can be used for $UI_{diff}(x)$.

Our final resulting interest function for a node x then captures the a priori interest, the parameters the user has specified in their search z for a focus node y and the distance of x from that focus:

$$DOI(x \mid y,z) = \alpha \cdot API_{diff}(x) + \beta \cdot UI_{diff}(x,z) + \gamma \cdot D(x,y)$$

As defined, API needs to be computed only once, the UI needs to be recomputed whenever the initial search terms z change and D needs to be recomputed whenever the focal node y changes. In the next section, we show how we can use this interest function to allow users to interactively navigate very large graphs using a remote client.

3.2 Extracting contextual subgraphs

The second step of our proposed interaction model is *Show Context*. Given a giant graph G and an initial node of interest y, a challenge still remains on how to compute a subgraph F of G that is small enough to transmit to a client, but also captures as much of the relevant context around y as possible. More formally, we need to efficiently compute a connected subgraph F of size at most S that contains y and has maximal total interest.

It is possible to perform an exhaustive search of the solution space since there are only a limited number of connected subgraphs around y of size S, but such a search is too expensive. Instead, we use a greedy optimization algorithm that provides a good tradeoff between speed and interest optimization. Starting from an empty nodeset F and a list of potential candidates $\{y\}$, we remove the candidate x with the highest DOI from the list, add it to F and then add the immediate neighbors N(x) of that candidate to the list of potential candidates. This process continues until the list of potential candidates is empty or until the size of F is S. The resulting algorithm runs in time $O(S \log S)$, as it is necessary to maintain a sorted heap of potential candidates. The computed subgraph is then transmitted to the client along with graph meta-data such as labels, current interest levels and edge weights. Layout of this subgraph can then either be done client side or server side, depending on the type of client. This flexibility opens up many new possibilities for graph exploration. For example, one can easily imagine users browsing the graph over a mobile phone, a device that generally does not have the processing power needed to compute layouts for larger graphs in interactive time. In these situations, the server is responsible for computing the layout.

3.3 Expanding a subgraph

The final step of our interaction model is *Expand-on-Demand*. Once the local context for a node is available, users can decide to expand this context by bringing in more information. We have implemented a simple mechanism that allows users to click on a node *x* in the current context to bring in more contextual information for that particular node. This allows a user to direct the expansion of the current subgraph based on their current information need.

A naive solution would then bring in all the neighbors of *x* when the user indicates a need for more context. However, the system cannot make any guarantees on the degree of the node in question. This may result in users inadvertently increasing the size of the subgraph with hundreds of nodes resulting in potentially unreadable graphs and putting significant strain on the display and layout subroutines. While indicating that a node has a very high degree (for example, by visually labeling the number of neighbors on each node) is useful, users need a way to limit the number of new items to ensure the visualization does not get too complex.

The interest function defined in the previous section can be reused so the system can make judgment on which nodes in the set of adjacent nodes are most important. We can then limit the number of new nodes added to the current view by only adding the top *N* most interesting nodes. In combination with visual indicators on the size of the context, the amount of new information added to the current context can be controlled. If users wish to see more than *N* connected nodes, they can simple click the node multiple times, with each click bringing in *N* additional nodes.

An important unsolved problem in information visualization is how to direct the users to potentially interesting items in the visualization. Without some sort of guide, visualizations of large datasets can quickly degrade into massive graphical representations that still require the user to click through or drill down on many different items to find something interesting. To address this problem we can reuse our DOI function, which tells us how interesting a particular node is, to guide the user to interesting parts of the graph that are currently hidden. We therefore visually highlight the n most interesting directions for expansion. Typically, *n* should be small ($n<5$) to avoid overloading the user with choices. In most cases this top *n* will be stable as nodes further away from the focus will have lower interest. That is, if the user expands a node that is not in the top n, the interest of the new node will most likely be lower than the interest of the top *n* nodes because its distance from the focal node is greater. Using these visual indicators, users are guided toward interesting parts of the graph, preventing them from having to needlessly expand dozens of items before hitting potentially interesting ones.

3.4 Implementation

We have implemented the above ideas in a client-server system that allows users to browse arbitrary graphs over an Internet connection. The only hardware constraint is on the server, which requires enough physical memory to cache the entire graph data in RAM for performance reasons. If not enough RAM is available we can always use a disk based graph storage, but this will notably impact performance. There are no hardware constraints on the client. In this section we will give a brief description of some of the practical implementation issues. Afterwards, in Section 4, we show how we can apply this general setup to a specific dataset.

3.4.1 Graph Server

The graph server is responsible for maintaining an in-memory representation of the entire graph and serving that data out to different clients connecting to the server. Our full database resides in an RDBMS on a separate database server. On startup, our graph server connects to the database and reads the full graph into RAM along with any node attributes needed to compute interest functions. Other attributes are fetched from the database server on demand, to conserve memory. After reading the data, the server computes

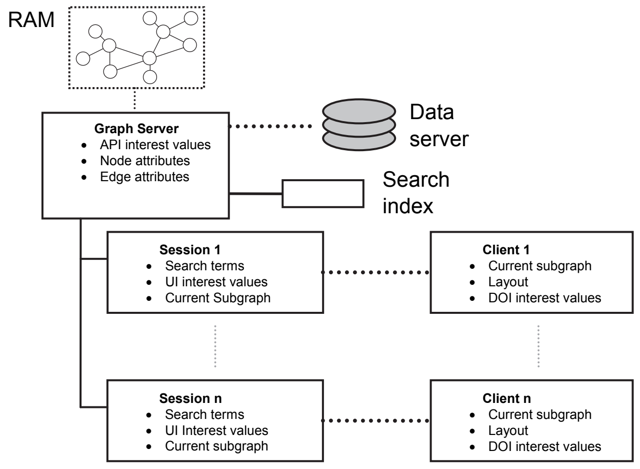


Fig. 2. Basic client-server architecture. The server maintains a copy of the entire network in RAM with the API values. Client interaction is handled from separate session threads, which keep track of session specific information, such as the currently visible subgraph and UI values.

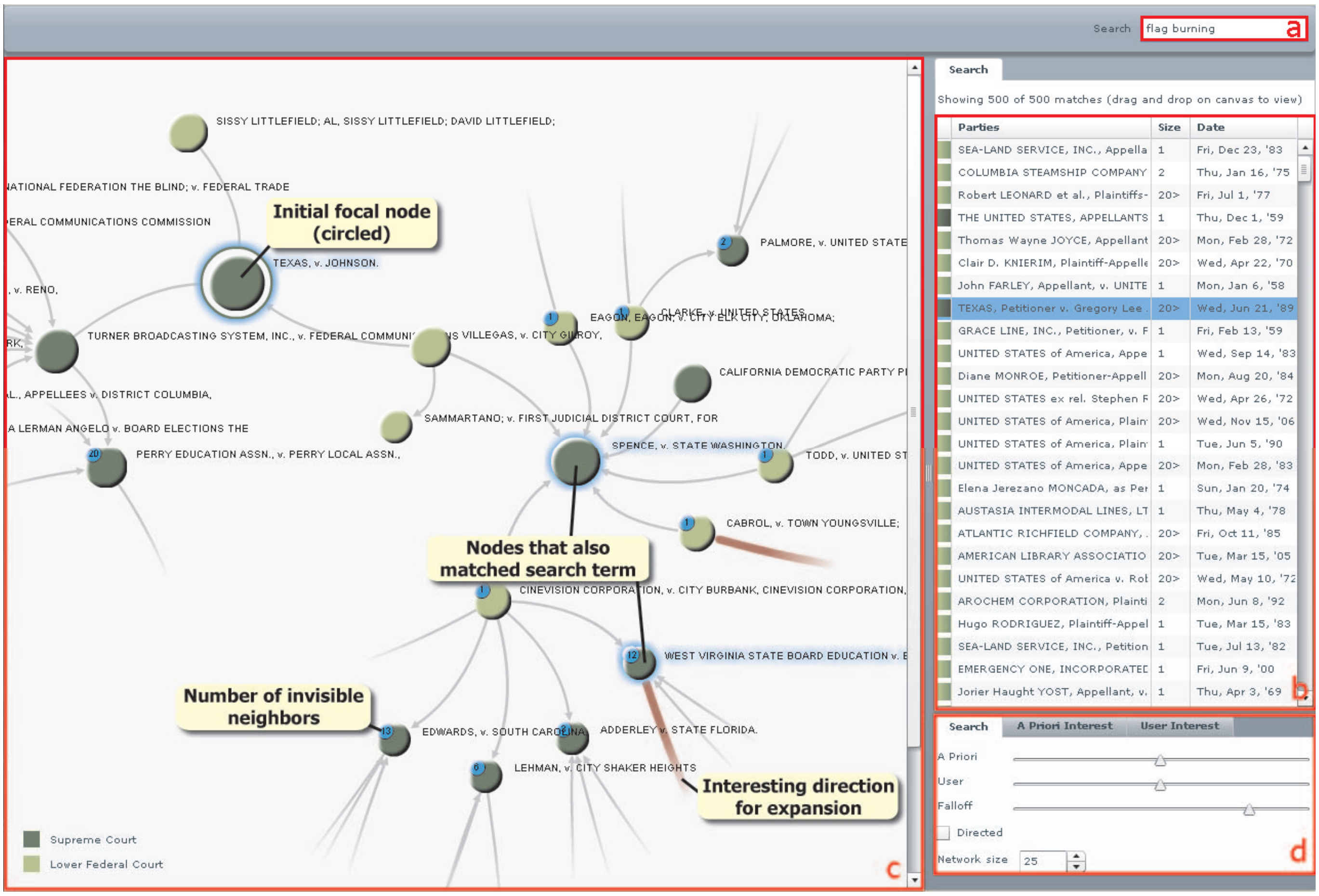
$\mathrm{API}_{\mathrm{diff}}(x)$ for all nodes and stores the result with each node. Finally, it waits for connection requests from any clients. Since the full interest function is dependent on the user's search terms and focal point, we cannot store the results of the DOI computation with each node.

Data that is specific to a user's session is therefore stored with a session object. Every session object runs in a separate thread inside the graph server and is responsible for maintaining information that pertains to a user's current browsing session. The node layout is computed using a force directed algorithm. In past experiments, we generally found the classic gradient descent method unreliable and prone to local minima and oscillations. Instead, we opted to use the stress majorization optimization method [6] that provides far better layout stability and is sufficiently fast for medium sized graphs.

3.4.2 Visualization Client

The client's main responsibility is to visually present the current subgraph to the user and handle all display and interaction tasks. Due to our extensible framework, the actual functionality of the client might depend on the capabilities of the platform on which it runs. For our use case, we implemented a client that runs in a standard web browser using Adobe's Flash framework. Our client consists of four basic sections (see Fig. 3): A searchbox (a), a list of search results (b), the main canvas (c) and a panel (d) that allows users to adjust the DOI settings to the task at hand. The searchbox in the upper right allows the user to perform a textual search on the full *dataset*, as the client makes a *Search* request to the server. The results of that search are then displayed in a tabular search result list. We opted to include basic node properties relevant to the application area, as well as an indication on the size of the surrounding subgraph. The latter immediately gives an indication as to what the size of the display will be when that node is picked as a focal node, and prevents the user from selecting uninteresting singleton nodes. To avoid excessive computational overhead on the server we do not display the exact size if it is over a fixed threshold.

The user can then pick any node *x* from this list of search results and drag it over to the main canvas to view its context. The client makes a request to receive a subgraph around *x* from the server. This subgraph is then displayed on the main canvas with the focal node highlighted. The sizes of nodes correspond to their (normalized) degree of interest to give more visual attention to nodes with high DOI. Node color is used to map a node attribute that is revelant to the application area. Furthermore, nodes that match the users' initial search query are surrounded with a blue halo.


a
c
b
d
SISSY LITTLEFIELD; AL, SISSY LITTLEFIELD; DAVID LITTLEFIELD;
NATIONAL FEDERATION THE BLIND; v. FEDERAL TRADE
ERAL COMMUNICATIONS COMMISSION
Initial focal node (circled)
TEXAS, v. JOHNSON.
PALMORE, v. UNITED STATE
, v. RENO,
EAGON, EAGON; v. CITY ELK CITY, OKLAHOMA;
CLARKE, v. UNITED STATES
TURNER BROADCASTING SYSTEM, INC., v. FEDERAL COMMUNI
S VILLEGAS, v. CITY GILROY,
CALIFORNIA DEMOCRATIC PARTY PI
L., APPELLEES v. DISTRICT COLUMBIA,
SAMMARTANO; v. FIRST JUDICIAL DISTRICT COURT, FOR
A LERMAN ANGELO v. BOARD ELECTIONS THE
PERRY EDUCATION ASSN., v. PERRY LOCAL ASSN.,
SPENCE, v. STATE WASHINGTON
TODD, v. UNITED ST
CABROL, v. TOWN YOUNGSVILLE;
Nodes that also matched search term
CINEVISION CORPORATION, v. CITY BURBANK, CINEVISION CORPORATION,
WEST VIRGINIA STATE BOARD EDUCATION v. E
Number of invisible neighbors
EDWARDS, v. SOUTH CAROLINA;
ADDERLEY v. STATE FLORIDA.
LEHMAN, v. CITY SHAKER HEIGHTS
Interesting direction for expansion
Supreme Court
Lower Federal Court
Search
A Priori Interest
User Interest
A Priori
User
Falloff

Search flag burning

Search

Showing 500 of 500 matches (drag and drop on canvas to view)

Parties	Size	Date
SEA-LAND SERVICE, INC., Appella	1	Fri, Dec 23, '83
COLUMBIA STEAMSHIP COMPANY	2	Thu, Jan 16, '75
Robert LEONARD et al., Plaintiffs-	20>	Fri, Jul 1, '77
THE UNITED STATES, APPELLANTS	1	Thu, Dec 1, '59
Thomas Wayne JOYCE, Appellant	20>	Mon, Feb 28, '72
Clair D. KNIERIM, Plaintiff-Appelle	20>	Wed, Apr 22, '70
John FARLEY, Appellant, v. UNITE	1	Mon, Jan 6, '58
TEXAS, Petitioner v. Gregory Lee	20>	Wed, Jun 21, '89
GRACE LINE, INC., Petitioner, v. F	1	Fri, Feb 13, '59
UNITED STATES of America, Appe	1	Wed, Sep 14, '83
Diane MONROE, Petitioner-Appell	20>	Mon, Aug 20, '84
UNITED STATES ex rel. Stephen F	20>	Wed, Apr 26, '72
UNITED STATES of America, Plain	20>	Wed, Nov 15, '06
UNITED STATES of America, Plain	1	Tue, Jun 5, '90
UNITED STATES of America, Appe	20>	Mon, Feb 28, '83
Elena Jerezano MONCADA, as Per	1	Sun, Jan 20, '74
AUSTASIA INTERMODAL LINES, LT	1	Thu, May 4, '78
ATLANTIC RICHFIELD COMPANY, .	20>	Fri, Oct 11, '85
AMERICAN LIBRARY ASSOCIATIO	20>	Tue, Mar 15, '05
UNITED STATES of America v. Rol	20>	Wed, May 10, '72
AROCHEM CORPORATION, Plainti	2	Mon, Jun 8, '92
Hugo RODRIGUEZ, Plaintiff-Appel	1	Tue, Mar 15, '83
SEA-LAND SERVICE, INC., Petition	1	Tue, Jul 13, '82
EMERGENCY ONE, INCORPORATEL	1	Fri, Jun 9, '00
Jorier Haught YOST, Appellant, v.	1	Thu, Apr 3, '69

☐ Directed

Network size 25

Fig. 3. Basic user interface layout. A user types a query in the searchbox (a) which yields a number of hits presented in tabular form (b). One of these hits can then be dragged to the main screen (c) which shows the subgraph centered on that node. Other nodes that matched the user's search are highlighted in blue. Users can adapt the balance between different components of the DOI function and the size of the subgraph in a separate panel (d).

Users can click a node to send an expand request to the server to return its most interesting hidden connections. In the client visualization, fading edges emanate from nodes to show the user that more connections are available. Although the endpoints of these edges are not visible, they are included in the layout. This means that different fading edges connecting to the same node will point a single point in space, giving the user a visual clue that they connect to the same node. Not every non-visible connection is drawn because the degree of a single node might be too large to render, so we cap the number of fading edges per node to a maximum. To give users a more quantitative indication on the actual number of adjacent nodes, we also display the precise number on each node. After receiving a new set of connections a new layout is computed based on the initial layout and the whole graph is then smoothly animated to its new position.

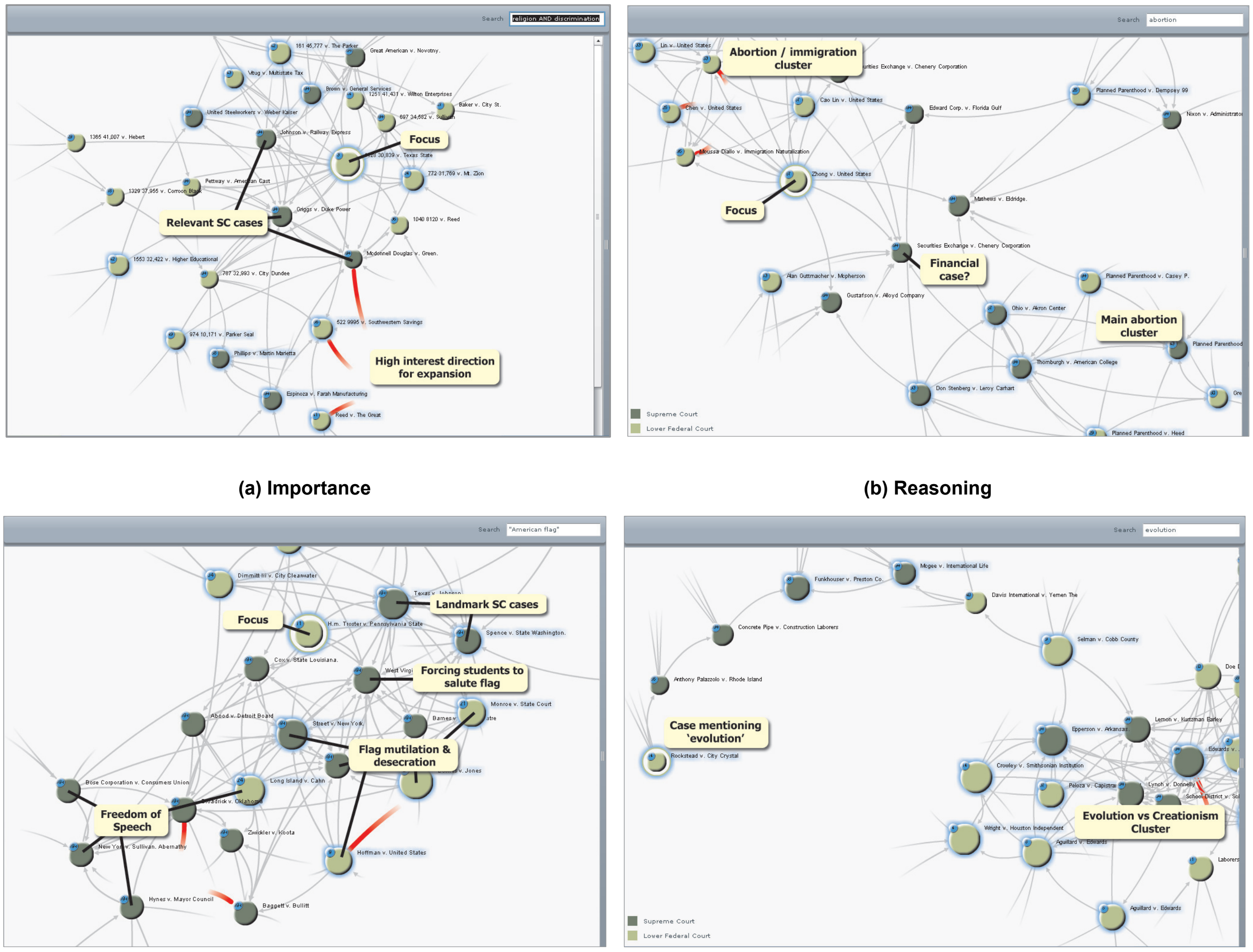
4 Evaluation: Legal Citation Network

To evaluate the initial benefits of our approach we applied it to a massive dataset of legal document citations. Legal practitioners sometimes need to interpret this citation graph of cases in order to understand court decisions. In the United States, laws rely on the concept of precedence, so lawyers must carefully study case citations to understand how relevant issues were ruled in prior cases [24][21]

Typically, searching for legal background information happens through a digital library platform such as LexisNexis or Westlaw [13]. These platforms however, are mainly text-based and focused on providing powerful search, convenient overviews of single documents, as well as quick linking between documents. Obtaining a context of cited documents typically requires manually clicking through a potentially large number of related cases, which are each visible in their own separate windows. This type of interaction model makes it hard to find relevant cases more than one step out, as that could possibly require manual inspection of hundreds of cases.

The following case study is conducted on a set of over 15 gigabytes of legal documents containing every federal court case in US history, but does not include cases tried by state courts. Still, these documents form a graph of over 300k documents and slightly over 3.3 million citations. The average number of citations to and from a node typically lies between 1 and 200 but there are a substantial number of cases with thousands of citations. Fig. 5 shows the detailed degree distribution. We loaded this full dataset into our setup running on a Dual-Core 2.4Ghz laptop with 4GB RAM, which took about 5 minutes of server startup time loading data and computing API values, and ended up using 1.2GB of RAM. Queries from the browser client typically finish in a couple of seconds (about 15 seconds per search and UI recomputation and up to 3 seconds for subgraph computation)

Here, we wanted to see if we could use DOI graphs to extract relevant context around legal cases. We decided that relevant cases would be oft-cited Supreme Court cases that match the initial keyword search and are in close proximity to the focus. The a priori interest (API) function for our legal graph examines node attributes and topology of the graph to assign higher interest to supreme court cases and cases with higher in-degree. The user interest (UI) function considers the score for that node on the search terms used when the legal analyst searched for a focal node. Users may directly manipulate the weights of the different components in the DOI


religion AND discrimination
Focus
Relevant SC cases
High interest direction for expansion
abortion
Abortion / immigration cluster
Focus
Financial case?
Main abortion cluster
Supreme Court
Lower Federal Court
"American flag"
Focus
Landmark SC cases
Forcing students to salute flag
Flag mutilation & desecration
Freedom of Speech
evolution
Case mentioning 'evolution'
Evolution vs Creationism Cluster
Supreme Court
Lower Federal Court

(a) Importance **(b) Reasoning**

(c) Overview **(d) Relevance**

Fig. 4. Sample use cases and tasks. In **(a)** the user is looking for relevant cases on 'religion + discrimination' that do not contain these search terms and might not be cited directly by the focal node. Three relevant supreme court cases are returned **(b)** shows how seemingly unrelated cases might connect two distinct clusters of relevant legal cases on abortion. **(c)** shows an overview of cases related to a search for 'American flag'. Relevant clusters of cases can be identified by examining case details. **(d)** shows how the structure of a network might give clues on the relevance of a case. Here we selected a case containing the word 'evolution', but the actual cluster of cases related to evolution

function by using sliders in the user interface. Finally, as no attribute information for edges exist, the edge interest function *EI* is constant. Even though none of the authors are legal experts, these choices for interestingness seem reasonable as a first test, though further user testing might lead to more refined DOI functions for this particular domain. In the next sections we highlight four different potential usage scenarios.

4.1 Importance: Show me important cases related to my case of interest.

DOI graphs are useful when analysts wish to find important nodes that are related to a specific focal node. In a legal context, this can be particularly useful when analysts are looking for important, oft-cited, Supreme Court cases relevant to their selected case. In this particular scenario, a legal analyst searched for court decisions that matched the keywords "religion AND discrimination". Out of the search results, the analyst selected a decision from 1980, *Miller v. Texas State Board of Barber Examiners*. (annotated 'Focus' in Fig. 4a). This case features a decision about whether or not an employee was fired due to "discrimination on account of race, color, religion, sex, or national origin". As Fig. 4a illustrates, the contextual graph includes many other nodes that also feature the user's searched keywords (nodes highlighted in blue). These user-matched nodes surround three relevant Supreme Court decisions: *Johnson v. Railway Express* (on the limitation period of filing a discrimination suit), *Griggs v. Duke Power* (on indirect discrimination by aptitude tests) and *McDonnell Douglas v. Green*. Especially this last case, an "early substantive ruling" on employee discrimination according to Wikipedia, is remarkable because this case is not directly connected to the original focus case, nor does this case match the user's original keywords. Nonetheless, this case is brought to the users attention due to a priori interest function giving weight to often cited, high court cases in the context of the focal node. Without DOI, the subgraph of all cases two hops out would contain 2345 nodes and 2847 edges, would have a diameter of at most 5, and would be impractical to visualize using a node link diagram.

Additionally, Fig. 4a shows how indicating high interest edges gives us valuable clues as to the direction in which we want to expand the graph. In Fig. 4a three edges are highlighted which all seem to point at a single position. After expanding the marked edges, we find two additional Supreme Court cases on discrimination.

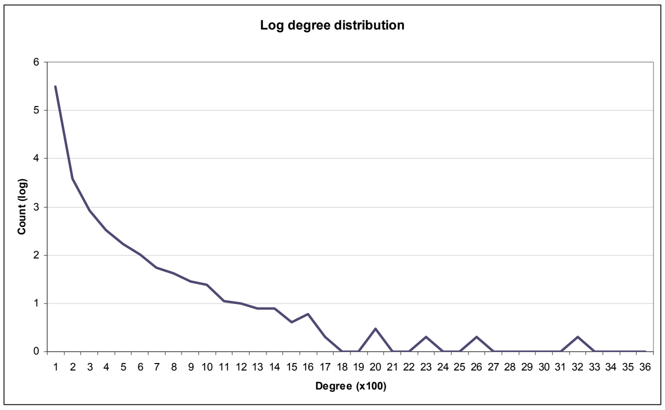


Fig. 5. Log-degree distribution of the case citation graph. Citation count includes both in and out citations.

4.2 Reasoning: Show me *why* this cited case is relevant to my case

One complexity of interpreting the citation graph is that decisions often cite other decisions that may seemingly have nothing to do with the semantic content. Consider this case study example of an analyst searching for decisions involving the keywords, ‘abortion’. He arrives upon a potentially interesting case (*Zhong v. US Dept of Justice*; the focus in Fig. 4b) of a person attempting to receive asylum in the US due to the practice of forced abortion in his home country. The decision curiously cites a case about illegal stock manipulation (*Chenery vs SEC*, annotated ‘Financial Case’), which initially seems to have very little to do with a decision about abortion. This case, which never mentions the word ‘abortion’, is inside the DOI graph since it is the source node’s indirect neighbor and is a heavily cited Supreme Court case, while some of its neighbors matched the initial search term. Upon reading the full text of the document, a feature available in our system, the analyst reads the description of why this particular citation exists ("a judicial judgment cannot be made to do service for an administrative judgment”). However, this cryptic statement does not resolve the analysts concerns and he wants to figure out why this case is cited.

Fortunately, DOI graphs are able to utilize this seemingly unrelated hub to search outward for additional abortion cases. In fact, two different decisions that match the user’s search of ‘abortion’ also cite the stock manipulation case (in the group labelled ‘main abortion cluster in Fig. 4b). These new abortion cases are inside the DOI graph because they both matched the analyst’s keywords and have a highly cited Supreme Court case (the financial case) as neighbour. By examining the rulings of these new abortion cases in detail the analyst is able to conclude that the context for this citation is the extent to which government entities can interpret law and retroactively set rules. As an added bonus the analyst has discovered a separate cluster of rulings on abortion cases that are not directly connected to the main cluster.

4.3 Overview: Show me an overview of this subfield

DOI graphs can also be used to provide an overview of the context surrounding a node of interest. In this example, the legal analyst searched for all decisions involving the key words “American Flag”. Since the user is interested in seeing a range of cases involving American flags, the interest sliders are adjusted to give more preference to matched keywords and less interest to a priori attributes like in-degree and level of court. From the search results, the analyst selects a focus case (*Troster v. Pennsylvania*) about an employee who refused to wear a flag on his uniform. The resulting DOI graph (Fig. 4c), is a dense graph but by examining the cases in detail we can still identify a number of conceptual clusters. First, it clearly shows the source node’s connections to important Supreme Court decisions about American flags, including *Texas v. Johnson* (constitutionality of flag burning) and *Spence v. State of Washington* (constitutionality of displaying flags with superimposed images). Second, the DOI graph highlights a variety of other lower court flag-cases not necessarily connected to the source, which deal with decisions that force students to salute flags, destroying flags for commercial purposes, and other cases of flag mutilation. However, on the other side of the graph, there are a variety of cases that have nothing to do with flags, but do revolve around the theme of the freedom to express oneself. So even though these decisions were not explicitly about flags, their arguments are critical to the flag cases. Even though we told the server to return a small 25 node context we can clearly group cases into relevant clusters.

4.4 Relevance: Show me if this case is what I think it is

This last scenario is an example where DOI graphs can help you understand the relevance, or lack thereof, of your selected search node. A legal analyst conducted a search using the term ‘evolution’ to discover decisions dealing with this biological theory. Similar to the previous scenario, the analyst adjusted the interest sliders to give more preference to matched keywords (UI), as the user was interested in cases about evolution. At the same time, less weight was given to the distance component of the DOI function. The analyst sorted the search results by date, and selects a recent case that is in the result list (*Rockstead v. Crystal Lake*). The resulting graph is quite peculiar, as the DOI graph brings in a cluster of “evolution”-matched decisions (labeled ‘Evolution vs Creationism Cluster’ in Fig. 4d), but they are 7 degrees away. Upon closer inspection, it seems the original source node has nothing to with the biological theory of Evolution, but instead the “evolution of common law”. The resulting picture made the analyst realize that the selected case had no “evolution”-based context in the legal graph, and that perhaps the selected case was not what the analyst was seeking after all.

We have presented several stories of how DOI graphs allow analysts to dig through very large and dense citation graphs. Although this anecdotal evidence should obviously be confirmed through more rigorous evaluation techniques such as long-term field studies (e.g. [16]) with a broad field of actual experts, these initial explorations were encouraging. Without the use of interest functions to make these otherwise dense graphs legible, or using only the currently available textural search tools it is doubtful the analyst would have made some of the observations above.

5 Discussion & Future Work

We have demonstrated how degree of interest can be used to effectively navigate large and highly connected graphs. However, there are still many outstanding challenges to support the analytical needs of users. Although we have chosen legal citations as data domain, there are many other data domains where this approach might be useful. In social network visualization, ego-centric views of the network are common and this approach fits well with that paradigm. Internet browsing could be enhanced by providing a user a local map around the visited page, using the user’s current information needs as input to an interest function.

We assumed that the user selects a single node as the focus, but we can very well generalize these ideas to multiple foci. That is, if a user selects multiple nodes in the graph, can we show a fixed size interesting (and connected) subgraph containing these nodes? Running our localized greedy algorithm no longer guarantees connectedness, but approaches like [3] and [11] use the graph topology to generate network ‘summaries’ that capture as much as much as possible of the basic proximity structure of the graph using a fixed node budget. We can directly integrate these techniques in our setup, or enhance them by taking node interest into account.

In this paper, we have defined a general framework for interest functions but have been deliberately vague about their precise implementation. The actual choice of properties or metrics to use for DOI computation should depend on the task and data domain for which the visualization is intended and it is unlikely that there is an optimal function that performs well for all domains and tasks. In our sample scenario we have used simple measures like the values of inherent attributes and local topology metrics for API and a keyword match percentage for UI. We could also choose an API function that uses global topological measurements relevant to users' tasks and domain. For instance, social network analysis metrics such as betweenness centrality would allow users to find bridge nodes more easily. PageRank-like algorithms would allow users to find hub nodes more easily. Although these metrics are generally expensive to compute, they only have to be computed once, at startup. In a similar manner, UI can be enhanced by reusing the history of interactions a user has performed on the current subgraph. Keeping track of common parameters of nodes selected as foci and nodes selected for expansion might help us make better decisions on the intent of the user, resulting in an adaptive DOI function and more relevant graphs.

Finally, in our visualization we are currently only showing the immediate context of a node at the lowest level. This makes it hard for users to orient themselves in the full information space. One possible solution to explore is to annotate edges on the periphery of the visible graph with distances to landmark cases in the graph, much like road signs in real life.

On the downside of this approach, the decisions made by the DOI based algorithms might seem abstract to the user at times. Looking for a specific citation between two cases can be problematic, as cases are brought in on a 'most-interesting' basis. We can address this by including some of the textual approaches used in common legal analysis tools, and allow the user to also search for and expand connections to particular cases.

Furthermore, with the most cited cases in the database having over 3600 citations, some of the heavily cited procedural cases are often included in the extracted DOI graphs, even though they might not be that interesting from a legal perspective. Fine tuning of the API function might be helpful in that regard. For example based on an examination of the degree distribution (Fig. 5) we can state that citation counts over 1600 do not add much to the information value and we cap the API interest level for those cases. Finally, our approach helps ameliorate some of the issues with rendering node link diagrams for dense graphs, but does not solve them completely, as is witnessed by Fig. 4c. Individual edges are still hard to follow in dense clusters and node link representations simply do not do a very good job in those cases. Possible solutions may include edge bundling [10] or including constraints on the maximum number of edges returned in the most interesting subgraph.

6 Conclusion

In this paper, we have shown that an adaptation of degree-of-interest functions to graphs helps manage the complexity of large and dense graphs. We advocate an additional term in the classic degree of interest function that captures the search for a focal node. We also introduced a method to diffuse interest over the entire graph to mitigate local minima. Concretely, we can use degree of interest to extract subgraphs from data residing on a centralized server and subsequently browse them on remote clients. On the interaction side we can use these same DOI functions to reduce the potential complexity introduced by very high degree nodes in the graph and simultaneously use it to visually indicate directions in which browsing might yield interesting results. In conclusion, we believe that an extension of the basic concept of degree of interest from tree to graphs offers promising ways to tackle the complexity of very large real world graphs and foresee a rich area for follow-up work.

Acknowledgments

The authors wish to thank Chris Collins for his efforts in processing and formatting the raw data.

References

[1] S.K. Card and D. Nation. Degree-of-Interest Trees: A Component of an Attention-Reactive User Interface. Proceedings Advanced Visual Interfaces AVI 02, 2002.

[2] T. d'Entremont and M.-A. Storey, Using a Degree-of-Interest Model for Adaptive Visualizations in Protégé, in *Proceedings of the 9th International Protégé Conference* (extended abstract), 2006.

[3] C. Faloutsos, K.S. McCurley, and A. Tomkins. Fast discovery of connection subgraphs. In Proc. 10th ACM SIGKDD conference, pp. 118–127, 2004.

[4] G.W. Furnas. Generalized fisheye views, In *Proceedings Human Factors in Computing Systems CHI 86*, pp. 16-23 ,1986

[5] G.W. Furnas. A fisheye follow-up: further reflections on focus + context, In *Proceedings Human Factors in Computing Systems CHI 06*, pp. 999-1008, 2006.

[6] E. Gansner, Y. Koren, and S. North, Graph Drawing by Stress Majorization, *Proc. 12th Int.Symp. Graph Drawing (GD'04), LNCS 3383*, pp. 239–250, 2004.

[7] E. Gansner, Y. Koren, and S. North, Topological Fisheye Views for Visualizing Large Graphs, *IEEE Transactions on Visualization and Computer Graphics.* Vol. 11 No.4 pp. 457-468*, 2005*.

[8] F. van Ham and J.J. van Wijk, Interactive Visualization of Small World Graphs. In *Proceedings of the IEEE Symposium on Information Visualization,* pp. 199 – 206, 2004.a

[9] J. Heer and S.K. Card. DOITrees revisited: Scalable, Space-constrained Visualization of Hierarchical Data. *Proceedings of the Working Conference on Advanced Visual Iinterfaces AVI'04*, pp. 421-424, 2004.

[10] D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, No. 5, Pages 741 - 748, 2006.

[11] Y. Koren, et al. Measuring and extracting proximity in networks. In *Proc 12th ACM SIGKDD conference,* pages 245-255*,* 2006.

[12] B. Lee, et al. TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts. *IEEE Transactions on Visualization and Computer Graphics*, vol. 12 (6), pp. 1414-1426, 2006.

[13] S. Makri. et al. Studying Law Students' Information Seeking Behaviour to Inform the Design of Digital Law Libraries, *European Conference on Research and Advanced Technology for Digital Libraries*, 2006.

[14] Palantir, http://www.palantirtech.com, Accessed May 2009.

[15] A. Perer and B. Shneiderman, Balancing systematic and flexible exploration of social networks. *IEEE Transactions on Visualization and Computer Graphics*, vol. 12 (5) pp. 693-700, 2007.

[16] A. Perer and B. Shneiderman, Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *Proceedings Human Factors in Computing Systems CHI 08*, pp. 265-274, 2008.

[17] M. Sarkar and M.H. Brown. Graphical fisheye views of graphs. In *Proceedings Human Factors in Computing Systems CHI 92* pp. 83-91, 1992.

[18] B. Shneiderman, The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pp. 336-343, 1996.

[19] B. Shneiderman and Aris, A. Network Visualization by Semantic Substrates. *IEEE Transactions. on Visualization and Computer Graphics*, Vol. 12, No. 5, pp. 733-740, 2006.

[20] M.A. Smith et al. Analyzing (Social Media) Networks with NodeXL. *Community & Technologies*, 2009.

[21] S.A. Sutton. The role of attorney mental models of law in case relevance determinations: An exploratory analysis. *Journal of the American Society for Information Science vol. 45 (3)*, 1994.

[22] Touchgraph LLC.. http://www.touchgraph.com. Accessed March, 2009.

[23] M. Wattenberg, Visual exploration of multivariate graphs. In *Proceedings Human Factors in Computing Systems CHI 06,* pp. 811-819, 2006.

[24] P. Zhang and L. Koppaka. Semantics-based legal citation network. *Proceedings of the 11th international conference on Artificial intelligence and law*, pp. 123-130, 2007.