Internet Mathematics Vol. 7, No. 4: 314–347

NAViGaTOR: Large Scalable and Interactive Navigation and Analysis of Large Graphs

Amira Djebbari, Muhammad Ali, David Otasek, Max Kotlyar, Kristen Fortney, Serene Wong, Anthony Hrvojic, and Igor Jurisica

Abstract. Network visualization tools offer features enabling a variety of analyses to satisfy diverse requirements. Considering complexity and diversity of data and tasks, there is no single best layout, no single best file format or visualization tool: one size does not fit all. One way to cope with these dynamics is to support multiple scenarios and workflows. NAViGaTOR (Network Analysis, Visualization & Graphing TORonto) offers a complete system to manage diverse workflows from one application. It allows users to manipulate large graphs interactively using an innovative graphical user interface (GUI) and through fast layout algorithms with a small memory footprint. NAViGaTOR facilitates integrative network analysis by supporting not only visualization but also visual data mining.

1. Introduction

Many real-world systems in diverse fields including physics, biology, and economics can effectively be represented as networks. Although human beings are known to perform image pattern recognition rapidly, nongraphical mathematical

ISSN: 1542-7951 print

packages are typically used to analyze networks, and visualization is often left as an afterthought [Perer and Shneiderman 08].

Exploratory data analysis through network visualization remains a challenge. Overlapping nodes and edges make the interpretation of large networks difficult, so much so that these are sometimes referred to as "hairballs" [Weinberg 10]. However, integrated network analysis and visualization is emerging as a fruitful approach to revealing patterns in data. Such an approach may lead to improved understanding of complex systems [Hidalgo 08]. Due to many different types of users and tasks, "one size does not fit all," and a useful tool needs to be (1) flexible in satisfying diverse user needs through various workflows, (2) intuitive to allow for different displays with a mix of automated and manual improvements, and (3) scalable in terms of memory and speed, which is especially important for large graphs.

NAViGaTOR addresses many of these needs through an integrated approach to network analysis and visualization. We describe a visualization system capable of handling large-scale networks that supports both 2D and 3D views and provides ability to interactively manipulate the layout and visual attributes with a smaller memory footprint than most other graphing applications available [Brown et al. 09]. Considering arbitrary entities as nodes, and relationships between them as edges between the nodes, NAViGaTOR visualizes the corresponding graph using OpenGL acceleration [Shreiner 99] and a dynamic force-directed layout algorithm, optimized for multiple CPU cores.

NAViGaTOR supports multiple analysis functions that can be performed on graph objects from any application domain. It also provides export capabilities (a graph can be exported as a list of edges or as an adjacency matrix) to allow for more customized analyses in tools such as R and Matlab. The software provides (1) the ability to process large data sets representing graphs from different file formats (e.g., PSI-MI XML [Kerrien et al. 07], BioPAX OWL [Demir et al. 10, Antoniou and Harmelen 09], GML [Himsolt 96], MITAB [Kerrien et al. 07], tab-delimited), (2) tools to manipulate, modify, and perform analyses on graphs, and (3) export functionality in various file formats (e.g., PSI-MI XML, GML, tab-delimited) as well as image formats including vector graphics.

In this paper we describe how NAViGaTOR handles complex data-exchange formats and memory usage. We compare the tradeoffs of data-rich XML file formats to store ancillary contextual data for the graph as opposed to tab-delimited file formats that contain only basic link information. NAViGaTOR provides tools to perform data analysis and graph layout in a user-friendly package.

2. Workflows

2.1. Loading Experimental Data

NAViGaTOR depicts a network using a graph structure. A variety of standards are available for data import (such as PSI-MI XML and BioPAX OWL for biological networks, GML for graphs, as well as tab-delimited files), each with a different focus on the ancillary data to be associated with the graph's nodes and edges (also referred to as graph objects). Many existing network data sets are readily available in these formats, and each can be easily loaded into NAViGaTOR using each format's respective file loader (such as tab-delimited and GML). The network format of choice balances the need for generality, speed, and memory requirements, as described below.

2.1.1. Text Formats. Two text formats are the tab-delimited text format and Graph Modeling Language (GML).

Tab-delimited text format: The tab-delimited text format is the most generic way to load a network to visualize within NAViGaTOR. Each edge is represented by a tab-delimited line containing unique identifiers for its incident nodes and ancillary data describing the properties of nodes and edges. The edges can be generated computationally, extracted from spreadsheet files, or entered manually. From within the text loader, the user is prompted by a series of dialog windows to describe the data being entered, indicating identifiers and ancillary data to be associated with nodes or edges. Several common data types are supported, including numbers, text, and XML.

GML: The Graph Modeling Language [Himsolt 96] is a text-based format that stores the layout and visual attributes associated with a graph such as node size and edge thickness, which gives GML a much smaller memory footprint when rendering the network in NAViGaTOR than the XML-based file formats. For example, memory footprint may be reduced from 193 MB to 124 MB when loading a network with 712 nodes and 555 edges in BioPAX format (a reduction of 35%) instead of the same network in GML format. However, since the GML format does not store much ancillary data with the graph, it renders it less versatile than the biology-specific formats. The GML format is specialized for saving the visual appearance of the nodes and edges as well as the layout.

2.1.2. XML Formats. Three XML formats supported by NAViGaTOR are PSI-MI XML, BioPAX, and the NAViGaTOR XML-based file format.

PSI-MI XML format: The PSI-MI XML format [Kerrien et al. 07] is a community standard data-exchange format for representing molecular interactions, maintained by the HUPO Proteomics Standards Initiative [Orchard et al. 03]. Networks are described by interactors and interactions, as well as information regarding nomenclature, supporting literature, and experimental data arranged according to a structured XML schema.[1] Although this format facilitates data exchange between many databases and tools, it also creates storage and data visualization overhead.

BioPAX: The BioPAX format [Demir et al. 10] boasts similar functionality to that of the PSI-MI format, but it focuses on describing biological pathway information rather than the broader area of molecular interactions (http://www.biopax.org/). Within NAViGaTOR, physical entities described by a BioPAX file, as well as some of its logical constructs, are represented as nodes. Nodes are assigned different shapes based on what underlying concept they represent. Like PSI-MI XML, the BioPAX format is very useful for data exchange, but it increases the data storage and data visualization requirements. Currently BioPAX 1 and 2 are supported in NAViGaTOR version 2.2, and we are planning to support BioPAX 3 soon.

NAViGaTOR XML-based file format: This format focuses on storing both graph data and application setting. The biology-specific formats (such as PSI-MI XML and BioPAX) are more concerned with storing annotations of experimental data, interaction sources, and other biologically relevant information than the visual appearance and layout of the network. In addition to basic information about interactions and related proteins, saving a network in the NAViGaTOR XML format retains the state of the network, which includes the graph layout, node and edge annotations, visualization settings, and lists of saved subsets.

Many real problems require loading a large amount of node, edge, and graph annotation data into NAViGaTOR. Storage of this information competes with limited memory resources required for visualization purposes. Although XML formats contain a better structure than tab-delimited text format, storing this ancillary information in the computer's memory while rendering the network and allowing for interactive functionality comes at the cost of considerable overhead.

[1] Available online at http://psidev.sourceforge.net/mi/rel25/src/MIF25.xsd (accessed 17-December-2010).

For instance, the BioPAX file format written in the Ontology Web Language (OWL) contains a strict hierarchy enforced by an underlying ontology.

Importing the entire XML tree allows us to preserve the parent–child relationships stored in the ontology but adds to memory overhead. As an example, consider the network of biological pathways in the worm *C. elegans* available from Reactome, which contains 7353 nodes and 11,706 edges.[2] Opening this worm network in NAViGaTOR BioPAX format takes 65 seconds and requires at least 220 MB in memory resources, whereas opening the same network in tab-delimited format (without extra annotations) takes only 5 seconds, and 95 MB of memory is sufficient.

Nevertheless, the BioPAX format offers the most contextual information about the network by annotating all the nodes and edges with their corresponding XML information and displaying it in a collapsible tree view. The collapsible tree view is flexible, allowing the user to navigate through the network and the desired level of information about nodes and edges.

Combined, each of these file formats is suited for different uses. We recommend loading in a lean file format (not including ancillary data) such as the tab-delimited format to analyze the graph structure. Note that ancillary data may be added to the graph later, by loading a tab-delimited file separately, or directly from web sources (e.g., Gene Ontology [Ashburner et al. 00], I2D [Brown and Jurisica 07], PSICQUIC [Orchard et al. 10], STRING [Jensen et al. 09], KEGG [Kanehisa et al. 10], Reactome [Croft et al. 10], WikiPathways [Pico et al. 08], Pathway Commons [Cerami et al. 11], iHOP [Hoffmann and Valencia 04]).

NAViGaTOR can select graph objects using XQuery [Boag et al. 02] to leverage some of the functionality of XML formats. XQuery goes beyond the simple text and complex regular expression search functionality by defining a query and functional programming language for search. NAViGaTOR takes full advantage of the XML data structure with XQuery's logic and nested expressions. XQuery is defined by the XML Query working group of the W3C, and is implemented in NAViGaTOR with the SAXON library.[3]

Considering complexity and diversity of data and tasks, there is no single best layout, no single best file format or visualization tool: one size does not fit all. One way to cope with these dynamics is to support multiple scenarios/workflows. We continue to add new data formats, analysis algorithms, features, links to external sources, and diverse tasks that require network visualization and analysis.

[2] *Caenorhabditis elegans* events in the BioPAX level 2 format. Available online at http://www.reactome.org/download (accessed 4-January-2011).

[3] Available from Saxonica Limited. "SAXON: The XSLT and XQuery Processor," http://saxon.sourceforge.net (accessed 17-December-2010) .

2.2. Layouts and Visualization of Node and Edge Attributes

While diverse layouts can be used to visualize complex networks, rendering large networks without overwhelming the user presents a challenge. With more and more comprehensive annotation for nodes and edges, interactive visualization is necessary for visual data mining. These challenges can be addressed by combining automated and manual processes that manipulate the network and its layout, showing the data at the right level of detail in a coherent manner using the available screen real estate resourcefully. Since many criteria influence layout choices, knowing the goal of the network visualization is important in enabling effective layout. Subcomponents of graphs with specific properties can usually be placed together to represent functionally similar units. Layouts can be created manually (often through some creative process) or algorithmically (usually using graph-theoretic analysis of network structure to guide suitable layout).

An effective workflow starts with an automated layout, followed by the use of a powerful user interface [McGuffin and Jurisica 09, Viau et al. 10] to manually improve the network layout. Automatically identifying the largest connected component, separating small subgraphs with similar features, locating highly connected components, identifying central nodes and edges, hubs, and other distinctive graph structures play an important role in selecting what should be highlighted and visible. In this way, some subgraphs may be rendered partially transparent to reduce network complexity; edges and nodes can use different colors or shapes based on these features, and so forth. Importantly, NAViGaTOR can store individual subgraphs as sets, and basic set operations can further benefit node and edge selection and visualization, such as complement, intersection, and union.

Different layout techniques can be applied to subcomponents of the graph depending on what type of information needs to be highlighted (e.g., listing node neighbors, showing connections between groups of nodes, displaying a hierarchy of data). Uses of these techniques are described in detail below.

2.2.1. Linear Layout. The linear layout enables the user to enumerate all interacting partners for a node. All the interacting nodes can be displayed in a straight line with the labels for the nodes on the side (Figures 1, 2). Note that the linear layout used in Figure 1 is node-centered, focusing on two nodes and their neighbors, whereas the layout in Figure 2 shows relationships between all the nodes in the network. Additionally, it is sometimes helpful to lay out the edges as branching from a single source (see the right side of Figure 1), or to curve them as originating from a single source and diverging out to multiple targets

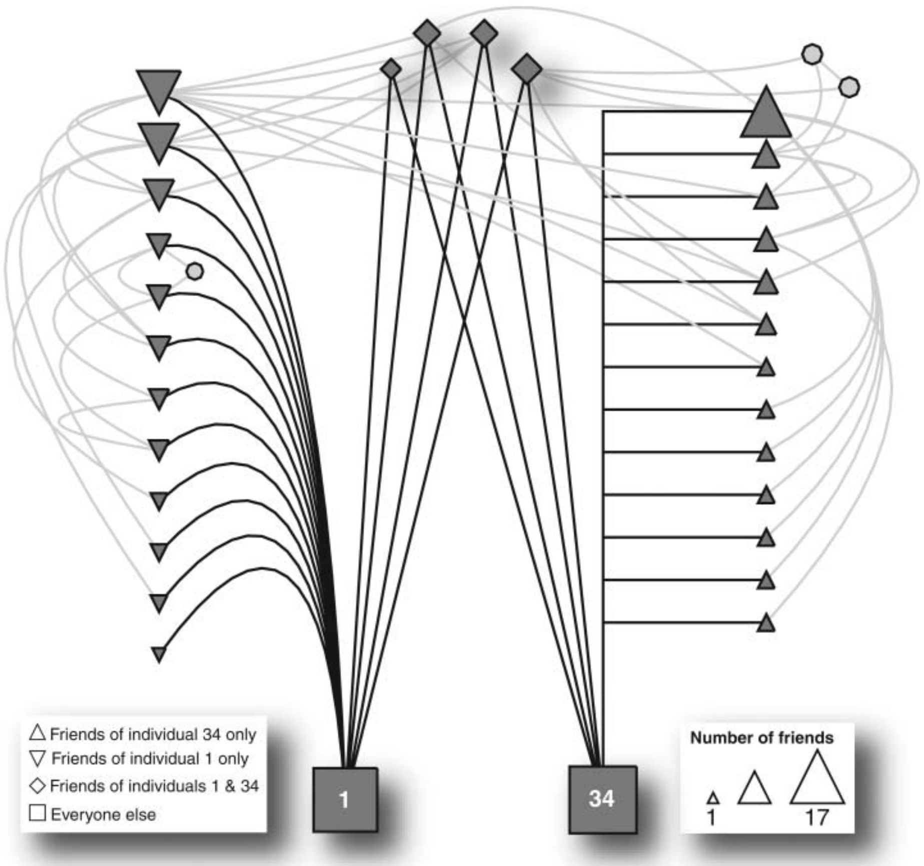
△ Friends of individual 34 only
▽ Friends of individual 1 only
◇ Friends of individuals 1 & 34
□ Everyone else
1
34
Number of friends
1
17

Figure 1. Zachary's karate club network showing friendship among members [Zachary 77]. Friends of members 1 and 34 are highlighted, and all other connections are faded out. Various node shapes are used to describe different sets of friends. The downward-pointing triangles are friends of 1, the upward-pointing triangles are friends of 34, and the diamond-shaped nodes are friends of both 1 and 34. Node size represents the number of friends each member has: the higher the number of friends, the larger the node size. This was done by mapping the node degrees to the node size, as shown in the legend. Additionally, the nodes in the linear layouts for friends of 1 and for friends of 34 are automatically arranged by the number of friends each member has in this social network; members with the most connections are at the top, and those with the least at the bottom.

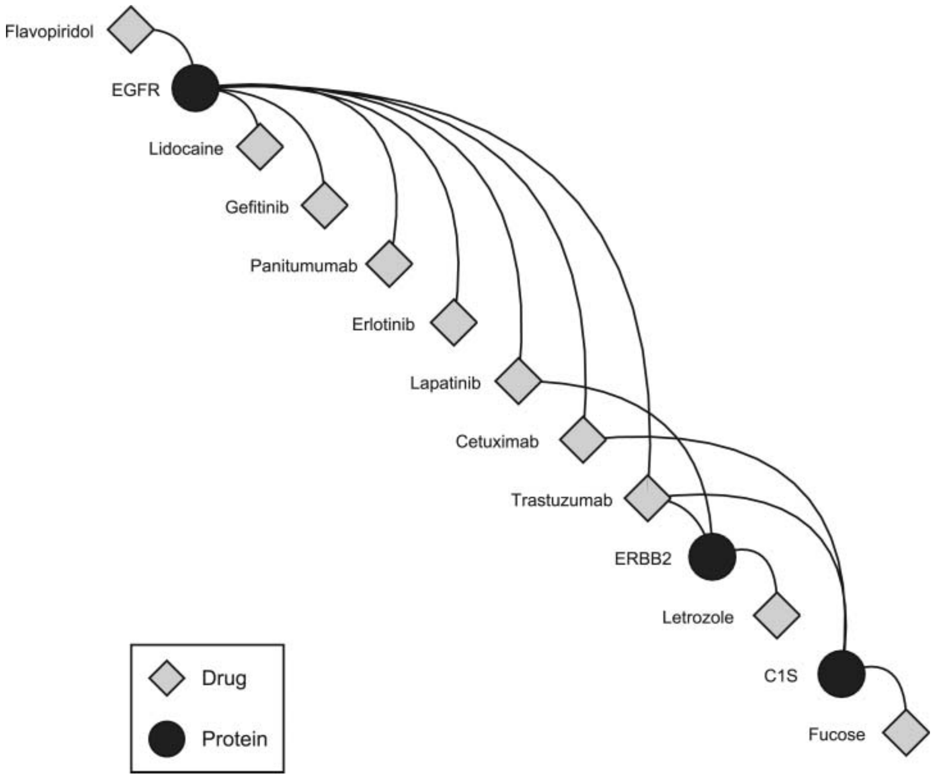

Figure 2. A subnetwork of nodes up to two connections away from the drug Trastuzumab (also known as Herceptin) obtained from the drug-target database DrugBank [Wishart et al. 07, Wishart et al. 06] and chosen to illustrate the linear layout with curved edges. If a drug (diamond) has an effect on a protein (circle), then the two are connected by an edge in the network. The drug Trastuzumab targets the ERBB2 protein (which is overexpressed in a subtype of breast cancer). ERBB2 is also a member of the same receptor family as the EGFR protein, which is targeted by many other drugs such as Gefitinib and Erlotinib in lung cancer.

(see the left side of Figure 1). Importantly, individual nodes can be arranged according to any node attribute such as node name, gene ontology [Ashburner et al. 00], biological function, level of protein abundance, fold change, or other value (e.g., integer, double, text).

A linear layout can also be used to display interconnecting interactions between a group of nodes by using curved edges (splines). If all the nodes are placed on a straight line with straight edges connecting them, then individual edges are overlapping and invisible. However, splines go around the nodes, making individual edges clearly visible (Figure 2). This layout also reduces the number of edge crossings that we would find if the same network were displayed in a circular layout.

2.2.2. Circular Layout. In a circular layout, all the nodes of interest are placed on the circumference of a circle. Although it allows all the connections among relevant nodes to be visible (Figure 3), the circular layout also results in many intersecting edges, and depending on the number and placement of nodes on the circle and its diameter, it may be difficult to properly show all node labels or names. If the nodes are highly connected, or several closely placed nodes are connected in multiple ways, it might be difficult to identify individual connections. The number of edge crossings can be reduced by curving some of the edges outside the circle, or by grouping nodes with a shared neighborhood.

2.2.3. Hierarchical Relationships. Many networks contain either explicit or implicit data structure hierarchy. To visualize these, NAViGaTOR provides the Concentric Circles layout. Starting from a predefined set of "root" nodes, the Concentric Circles layout will place these in the innermost circle. Increasing layers of concentric circles with larger radii display the nodes that are one connection away from the nodes in the previous layer. The nodes in the outermost layer are displayed uniformly on the circumference of the largest circle. Each inner circle has its nodes placed at the center of all its neighbors in the outer circle (Figure 4).

NAViGaTOR can save arbitrary groups of nodes and edges in the graph as subset objects. Subset objects can contain other subsets, thus making it possible to store hierarchical data organization. Individual subsets can be collapsed in the view to show a compact representation of the data. Any connections between the nodes inside the collapsed subset and those outside are represented as a special edge class: "Subset Edges." However, subset edges can use the same diversity of visual attributes as regular edges.

2.2.4. Visualization of Large Networks. Although simple layouts (such as linear, circular and concentric circles) are often not effective on large networks due to screen-size limitations, they may be beneficial on subgraphs of interest. To aid visualization of large networks, NAViGaTOR uses automated layout algorithms that optimize the position of the nodes and take advantage of multiple CPU cores for faster rendering. These algorithms include GRIP (Graph Drawing with Intelligent Placement) [Gajer and Kobourov 01] and the force-directed layout algorithm.

GRIP: NAViGaTOR uses GRIP to perform an initial layout of the graph, which works effectively for small and medium-size networks with hundreds of nodes.

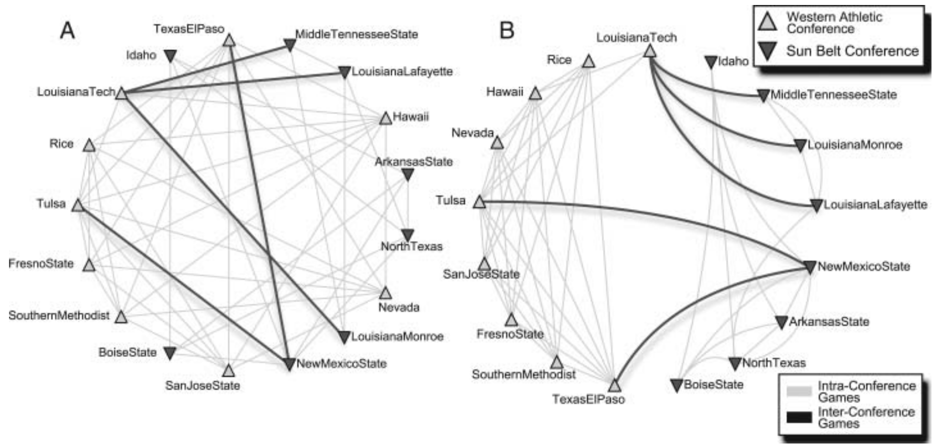

Figure 3. The network of American football games between Division I colleges in the Sun Belt and Western Athletic Conferences during the regular fall 2000 season [Girvan and Newman 02]. Nodes in the graph represent teams (identified by their college names), and edges represent regular-season games between the two teams they connect. Graph A shows a random circular layout with straight edges resulting in many edge crossings. Graph B shows nodes that are in the same conference grouped together into semicircles (using a NAViGaTOR query), which helps to visualize the cross-group games. Curved edges are used to highlight games between the teams (as identified by shortest-path analysis between teams), resulting in a more aesthetically pleasing and stronger visualization. Graph B shows that the Western Athletic Conference has almost twice as many games per team as the Sun Belt colleges, a fact that is difficult to discern from the graph A.

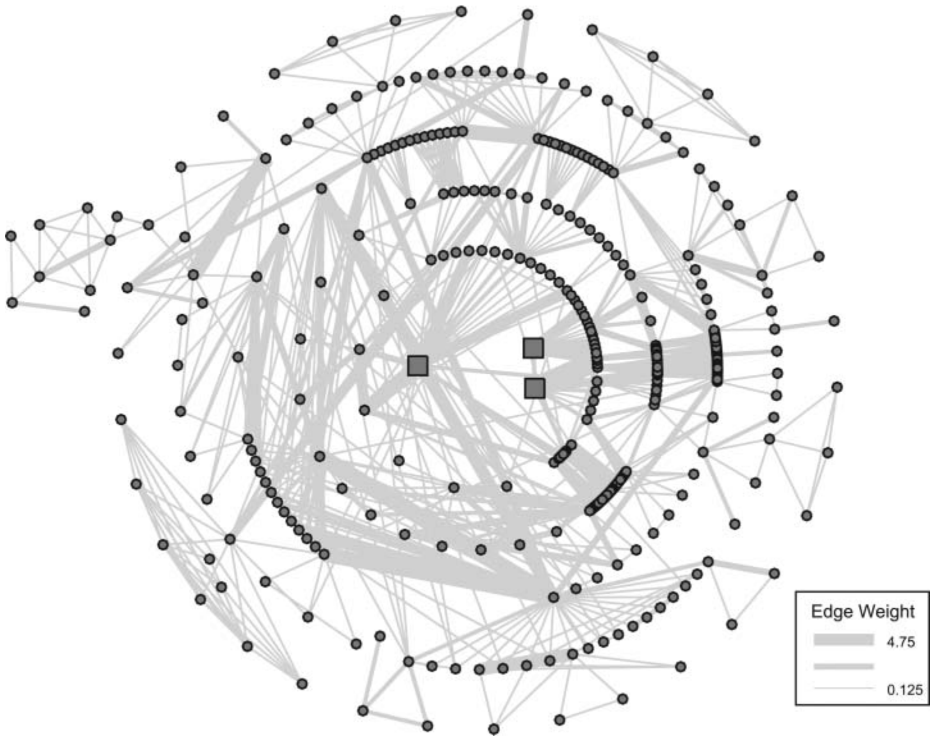

Figure 4. Coauthorship network of scientists working on network theory and experiments [Newman 06]. The three authors with the highest measure for betweenness centrality in the network (displayed as square nodes), were chosen as the "root" nodes for the Concentric Circles layout. Each consecutive circle displays the authors who have collaborated with the authors from the previous layer. Neighbors up to depth 5 were placed on circular layouts. The network is weighted, with weights assigned as described in [Newman 06]. The edge weights were mapped to edge thickness.

Although GRIP organizes larger networks more effectively than a random layout, additional algorithmic or manual arrangement of the nodes may be necessary to "detangle" highly interconnected cores of networks. A useful variant of the GRIP layout separates the largest connected graph from the individual components of the network to reduce overlap (Figure 5).

Force-directed layout: In addition to the GRIP algorithm, which statically computes the positions of the nodes of a given network, NAViGaTOR also provides a dynamic force-directed layout algorithm. The force-directed layout

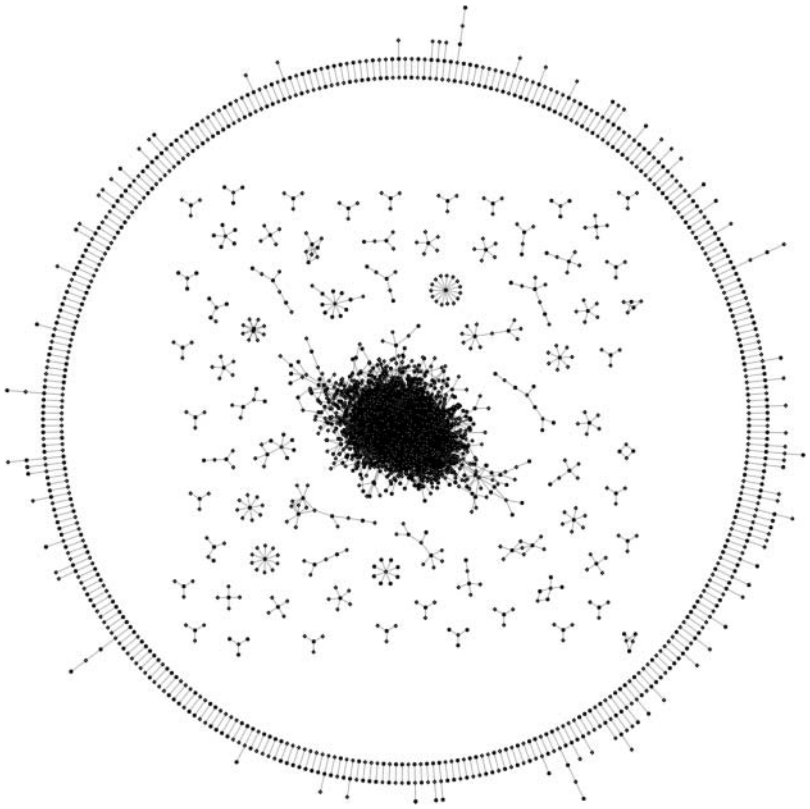

Figure 5. DrugBank network showing 4369 drugs and their 4534 target proteins [Wishart et al. 07, Wishart et al. 06]. If a drug has an effect on a protein, then the two are connected by an edge in the network. There are 11,950 such edges in this network. The layout is performed by the GRIP algorithm with individual components placed separately. As in Figure 2, drugs are rendered as gray diamonds, while proteins are displayed as black circles.

algorithm models a physical system. Nodes are treated as electrons that repel each other, and edges act as springs, moving the nodes at their endpoints closer together. The system iteratively places nodes in positions that bring the system closer to an equilibrium state. Several parameters, such as the spring rest length, can be modified to obtain different displays.

The force-directed layout algorithm is most effective when automated node placement is aided by fixing and organizing nodes of interest. The dynamic algorithm automatically adjusts the positions of the surrounding unfixed nodes, and

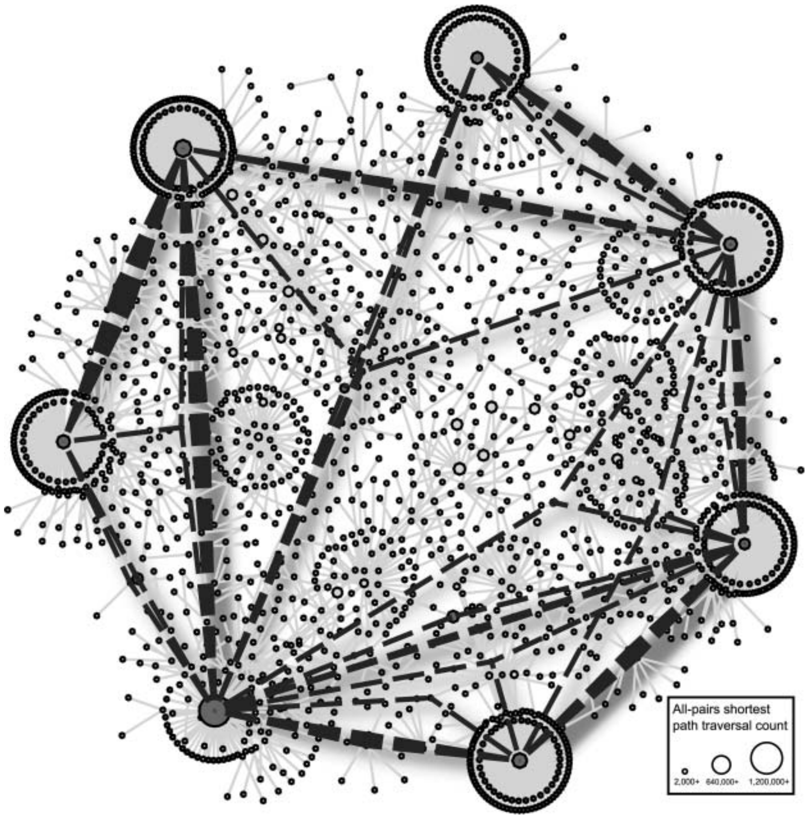

Figure 6. Application of force-directed layout algorithm to a subset of the Drug-Bank network. Seven drugs of interest (NADH, ethylene glycol, nicotinamide-adenine-dinucleotide, alpha-D-mannose, heme, adenosine-5′-diphosphate, acetate ion) were fixed in position and placed in a circular layout. The force-directed layout places their immediate neighboring targets of degree 1 in circles around the nodes. Also notice the straight dashed lines connecting pairs of drugs that have been fixed in position. These are interactions with those drug targets that are interacting with two or more of the drugs of interest. The common neighbors get clustered together in the middle. Node size represents the value of the all-pairs shortest-path traversal count for each node.

this process may be repeated until the visualization is aesthetically satisfying and helps to convey the intended message.

Hence the algorithm helps users by taking some manual layout effort away from them, automatically bringing them closer to a desired layout (Figure 6). As described above, selecting nodes of interest can be done by taking advantage of graph structure or annotation, using graph-theoretic algorithms, simple set operations, query, or XQuery.

Most workflows seem to benefit from the default setting in NAViGaTOR: The GRIP layout is applied first, followed by the dynamic force-directed layout, to continuously render the graph while the user is free to manipulate subgraphs of interest by combining query selection, algorithms, filters, and manual placement.

2.2.5. Visualizing Annotations. In addition to purely graph-structure-based visualization, it is useful to visualize different attributes of nodes and edges. In NAViGaTOR, any annotations associated with nodes and edges can be mapped to diverse visual properties. For example, node degree can be mapped to node size (Figure 1). Other node attributes such as gene-expression values can be mapped to node size, shape, color, highlight, or transparency. Similarly, edge weights or annotation can be mapped to edge thickness, color, style, or transparency (Figure 9). Edge weights can also be mapped to edge lengths. However, this requires the force-directed layout option to be active. The edge lengths are provided as a parameter to the layout algorithm, and it takes that into consideration as it optimally places the nodes. Additionally, the user has the option to perform an inverse mapping, such that higher weights map to shorter edge lengths. This is useful in cases in which a higher edge weight corresponds to a higher connectivity measure. With the inverse mapping option, the edges that have a higher measure of traffic flow will have their neighboring nodes closer together, indicating a higher level of connectivity or “closeness” between the nodes.

2.2.6. Fine-Tuning Placement of Graph Subcomponents. In addition to automated layouts, NAViGaTOR’s flexible user interface [McGuffin and Jurisica 09, Viau et al. 10] enables users to manually manipulate nodes and edges. This can help users to fine-tune the network layout by selecting various network components and move, rotate, scale, and annotate them. NAViGaTOR also implements hierarchical network organization by collapsing subgraphs (Figure 7). Users may zoom in and out of the graph view and also bend edges between nodes.

The color, size/length, shape/style, and transparency of both nodes and edges can be modified either manually or automatically using their attributes. Edges may be straight lines or quadratic Bézier curves [Farin and Hansford 00] of various lengths, widths, and dashed styles.

When one is visualizing large networks, it is often useful to focus on specific subcomponents by rendering the remainder of the graph semitransparent. Using transparency is preferred over the deletion of the remainder of the network, since the graph retains the overall network context (and thus supports, for example, querying) while reducing visual complexity.

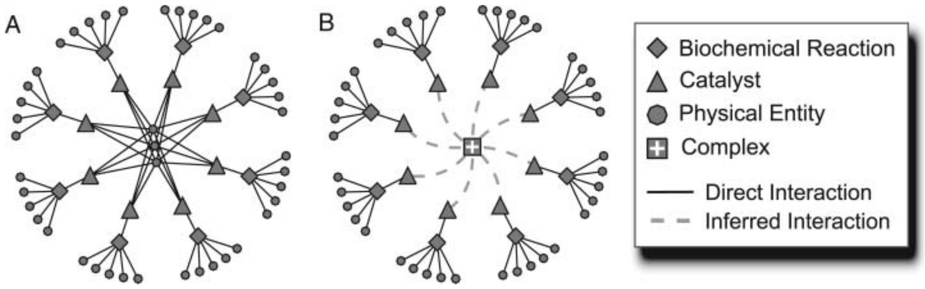

Figure 7. Graph A shows the purine biosynthesis pathway in the *A. thaliana* organism as curated by Reactome [Croft et al. 10]. Diamond nodes represent biochemical reactions, and triangular nodes are catalysts that assist the biochemical reactions. Each biochemical reaction involves several physical entities such as proteins, genes, RNA, DNA, and small molecules, which are represented by circular nodes in these figures. Reactome also provides information describing biological complexes, which are built up from physical entities or other complexes. In Graph B, the biological complex ribonucleotide reductase (as defined by Reactome's curation) has been collapsed into a single square node with a plus sign. The dashed edges are automatically generated by NAViGaTOR and describe interactions from the components of the collapsed complex to their respective interactors. The resulting visualization shows a higher-level organization of the pathway while maintaining the hierarchical structure of the biological complex.

Although there are some recurrent concepts, each network visualization usually needs some level of customization. Thus, trial and error is often involved in finding the most informative and aesthetically pleasing layout for a given network.

2.3. Manipulating Networks

NAViGaTOR's workspace can load multiple networks simultaneously. Users can define and bookmark subgraphs of interest to be recalled later as subset objects. These subsets can be changed by adding or subtracting graph objects. Set operations such as union, intersection, and complement can also be performed on the subsets. The results of set operations are functionally identical to the subsets passed to the operations such that they may in turn be modified with any number or sequence of operations. This feature promotes a versatile environment in which the user can define the most appropriate workflow for the required analysis.

When two graphs share unique node identifiers, these graphs may be combined with cut, copy, and paste functionality to create new graphs. Pasted subgraphs can be book marked as subsets, and are thus recognizable in their new setting and may also be manipulated by set operations. Importing ancillary data associated with graph objects can also enrich individual graphs. For example, NAViGaTOR can be used to visually query other online databases such as I2D [Brown and Jurisica 07], Reactome [Croft et al. 10], and PubMed.

Due to real-world constraints on time and resources, an experiment may concern a only small subgraph on a larger network. For instance, there may be only a subset of genes with associated experimental data (e.g., expression level, mutation). Similarly, only a subgraph of a social network may contain social categories (e.g., gender, age). These graph objects can then be placed in the context of another graph, of which the experimental objects are a subset. In the context of biological networks, data from other experiments can be imported from or integrated with much larger curated databases (IMEx Consortium,[4] Reactome, I2D, etc).

In such cases, NAViGaTOR plug-ins play an important role to import, extend, annotate, and analyze networks. These plug-ins integrate external web services and provide access to the latest data through common workflows that bridge differences in representation and retrieval. Networks can therefore be easily combined and analyzed, enabling users to work in an integrated data space that would not otherwise be available.

For example, interpretation of a list of genes and associated expression levels remains a daunting task. Loaded into NAViGaTOR, the interactions for these genes could be found through the I2D database using the I2D plug-in, providing context for the genes of interest. The network can be further annotated by PubMed references using the iHOP [Hoffmann and Valencia 04] plug-in, or with pathways using KEGG [Kanehisa et al. 10], WikiPathways [Pico et al. 08], cPath [Cerami et al. 06], or Reactome plug-ins, or with other interactions from the PSICQUIC web service [Orchard et al. 10] or gene associations from STRING [Jensen et al. 09]. Alternatively, interaction data could be loaded into the workspace directly from any of the databases listed above (or indirectly from any database that supports export into standard XML, GML, or tab-delimited files), and the experimental genes could be pasted into the existing network.

Plug-ins also assist with the interpretation of networks when there are significant representational differences. For example, in a biology context, some databases such as KEGG emphasize schematic summarization, whereas others, such as Reactome, emphasize biochemical fidelity, resulting in divergent models of the same biological processes. Through the PathSource plug-in, these models can be normalized into a common graph form, more suitable for extension and

[4]Available online at http://www.imexconsortium.org/ (accessed 17-December-2010).

analysis that preserves as much of the variable data as possible in node and edge annotations. These highly complex biomolecular networks can then be easily enhanced using the LitSearch plug-in, which queries selected proteins using more specialized web services (e.g., STRING, iHOP), to retrieve known drug interactions or gene-expression data. This data aggregation can help to elucidate or reinforce relationships between biological entities, strengthen support from the published literature, poll for consensus, or highlight areas for further analysis. Such an integrative approach is particularly important in biology, where individual data sets are often sparse, incomplete, noisy, and distributed across many databases in diverse formats.

Common workflows with online connectivity offer users the immediacy of *live* interaction with the current state of knowledge without having to contend with extraneous technicalities. This underlines the importance of client scalability, since vastly more data can potentially be reached, encouraging the manipulation of larger and more complex networks built from diverse sources. Computationally intensive analyses of these networks can also be outsourced to web services, freeing the user from local resource constraints.

However, network visualization alone is often not sufficient to generate insights from data. Fortunately, in addition to ancillary data import, a number of graph analysis approaches can be brought to bear on this problem.

3. Analyzing Graph Structure with NAViGaTOR

3.1. Introduction

Graph representation offers a simple yet powerful method of modeling complex systems, because the structure of a network is often deeply related to its function. Many real-world systems can effectively be modeled as networks, including routing in the World Wide Web [Aloul and Rawi 06], collaborations of scientists [Palla et al. 05], metabolic networks of genes [Zelezniak et al. 10], neural connections of the worm *C. elegans* [Watts and Strogatz 98], electrical power grids [Watts and Strogatz 98], food webs of predators and prey [Milo et al. 02], human disease transmission [May and Lloyd 01], telephone calls [Wang et al. 09]—the list goes on.

Over the past several decades, many powerful methods for analyzing and characterizing these graphs as mathematical objects have emerged from a wide range of applied and theoretical disciplines. Through a graph-theoretic lens, new connections have been revealed between seemingly disparate natural systems. For instance, graphs of the power grid of the western United States and the neural network of *C. elegans* have been shown to share the "small-world property" [Watts and Strogatz 98]. Briefly, this means that they have a low average shortest path length, and that the neighbors of a vertex tend to be highly interconnected among themselves (they have a high clustering coefficient). Similarly, other graph concepts have been proposed that draw clear distinctions between different kinds of graphs, such as motifs. For instance, the motifs particular to the World Wide Web are very different from those typical of food webs [Milo et al. 02]. The literature on graph analysis is rich, vast, and still growing at a fast pace; basic and advanced concepts have been extensively reviewed elsewhere; see, for example, [Newman 03, Strogatz 01, Barabási and Oltvai 04, Borgatti et al. 09].

3.1.1. Models of Real-World Networks Are Limited by Data Availability. While graph analysis has proven effective in many practical applications, it is important to note that incomplete or noisy data can limit our ability to make final conclusions about the *true* topology of real-world graphs. In fact, putting undue trust in graph models created from limited data that suffer from selection bias is a concern. The most prevalent models for the Internet and for protein–protein interaction (PPI) networks have been scale-free graphs (where the degree distribution follows a power law). Much research has been undertaken in this area [Rzhetsky and Gomez 01, Yook et al. 02, Lukashin et al. 03, Pržulj et al. 04a, Stumpf and Wiuf 05]. The following quotation from [Karagiannis et al. 04] on the changing structure of the Internet highlights the key challenge of modeling empirical graphs: "As the Internet increases in size and the technologies connected to it change, we must constantly monitor and re-evaluate our assumptions to ensure that our conceptual models correctly represent reality."

3.1.2. Basic Definitions. For the purpose of this section, $G = (V, E)$ is a graph with vertices V and undirected edges E [West 01]. Furthermore, $|V|$ denotes the number of vertices in G, and $|E|$ the number of edges, while $V(G)$ denotes the set of vertices in G, and $E(G)$ denotes the set of edges in G. An induced subgraph H of G is a graph such that $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, and H has the same assignment of vertices to edges as in G. All vertex pairs v_i, $v_j \in V$, are associated with an edge $e_{ij} \in E$. Let us assume that $e_{ij} = 1$ if the vertices are connected, and $e_{ij} = 0$ otherwise.

Paths: A path is a sequence of consecutive edges (or vertices) in a graph, and path length is the number of edges in the sequence. There are commonly multiple paths between two vertices, and paths with the minimum length are called shortest paths. The length of a shortest path is called the distance between two nodes.

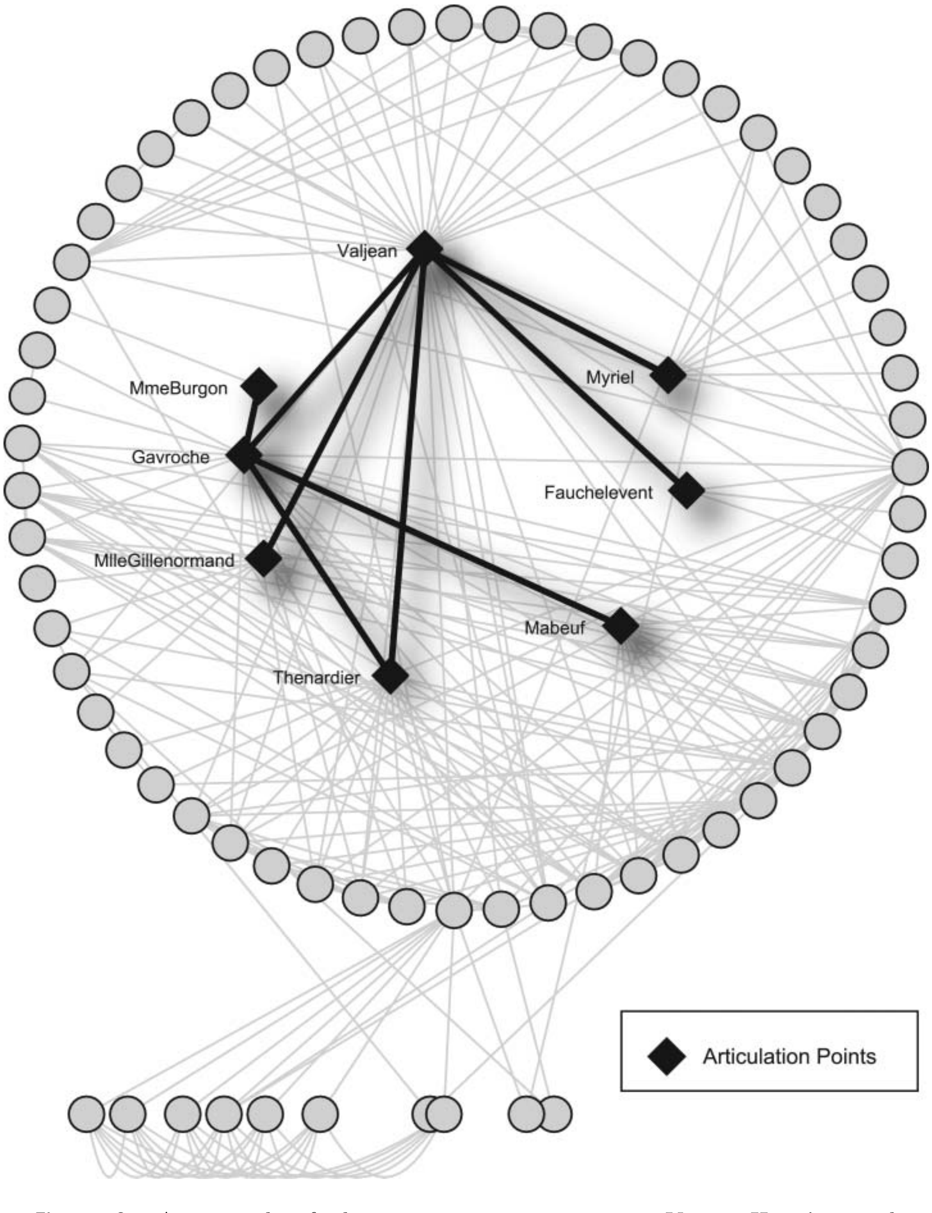

Figure 8. A network of character co-appearances in Victor Hugo's novel *Les Miserables* [Knuth 93]; see SNetwork of coappearances of characters in Victor Hugo's novel *Les Miserables*, available online at http://www-personal.umich.edu/˜mejn/netdata/ (accessed 22-December-2010). Nodes represent characters and are linked together if they appear in the same chapter. Interestingly, some of the eight characters identified as articulation points (diamonds) such as Valjean and Gavroche correspond to major characters in the plot who die by the end of the novel.

Connected components: A connected component is a subgraph in which every pair of vertices are connected by a path. Maximal connected subgraphs are the *components* of a graph [West 01].

3.2. Topological Measures of Vertex and Subgraph Characteristics

3.2.1. Articulation Points. An articulation point is a vertex whose removal disconnects the graph. Articulation points are often critically important in networks (Figure 8). For example, in PPI networks they have been associated with proteins essential for survival [Pržulj et al. 04b]. The Internet can be viewed as a graph whose vertices represent hubs, switches, or routers and edges represent lines of communication [Kim et al. 08]. Identifying articulation points may suggest changes to the network to improve its stability and robustness, such as decreasing their degree.

3.2.2. Measures of Vertex Centrality. Measuring vertex centrality is a crucial analysis step for identifying vertices that play important roles in a graph. There are many ways of defining centrality; two of the most useful are degree and betweenness centrality. For example, in PPI networks, proteins with high degree are much more likely to be essential to cellular function [Jeong et al. 01], and in a scientific collaboration graph (where vertices are authors and edges indicate that the authors have published a paper together), authors with high betweenness centrality play crucial roles: removing a few of them may break the graph apart into disconnected regions [Holme et al. 02].

Degree: The degree Deg of a vertex v is defined as the number of neighbors of a node v: $\mathrm{Deg}_v = \sum_i e_{vi}$, for $i \in V(G)$.

Betweenness centrality: While degree takes into account only the local connectivity of a vertex, betweenness centrality, BC, uses global information: it depends on the connectivity of the whole graph. The betweenness centrality of v is calculated as the number of shortest paths between pairs of nodes in graph G that pass through v:

$$\mathrm{BC}_v = \sum_{i \neq j} \frac{s_{ij}(v)}{s_{ij}},$$

where s_{ij} is the number of shortest paths between vertices i and j and $s_{ij}(v)$ is the number of such paths that pass through v.

3.2.3. Other Topological Measures. Several other topological measures that are functions of graph connectivity have proven important in many applications. For example, the *C. elegans* neural network is a "small-world" graph that is characterized by a high average clustering coefficient and low characteristic path length [Watts and Strogatz 98]. NAViGaTOR can calculate the clustering coefficient of vertices as well as the density and diameter of groups of vertices.

Clustering coefficient: The clustering coefficient CC of a vertex v reflects the extent to which its neighbors are interconnected. It is calculated as the number of existing connections between neighbors of v in the graph $G = (V, E)$ divided by the total possible number of such connections. Thus, if we define the neighbors of v as $N_v = \{w \in V : e_{vw} = 1\}$, then the clustering coefficient of v is given by

$$\mathrm{CC}_v = \sum_{i,j \in N_v} \frac{2e_{ij}}{|N_v|(|N_v| - 1)},$$

where e_{ij} is the number of edges connecting the N_v neighbors of node v to each other.

Edge density: The edge density Den of a group of vertices measures how close that group is to being a clique. For a group p of k vertices in a graph $G = (V, E)$,

$$\mathrm{Den}_p = \sum_{i,j \in p} \frac{2e_{ij}}{k(k-1)}.$$

Diameter: The diameter Dia of a connected component $(C_v, C_e) = C \subset G = (V, E)$ is defined as the longest distance between any pair of vertices in the component:

$$\mathrm{Dia}_C = \max\{d(i,j)\}\, \forall i, j \in C_v,$$

where $d(i, j)$ is the length of the shortest path between vertices i and j. The diameter provides an upper bound on the number of connections it takes to traverse a graph.

3.3. Identifying Graph Clusters

Many real-world graphs have an underlying community structure: They contain groups of vertices that are highly interconnected with one another and only sparsely connected to the rest of the graph. These communities are often meaningful, e.g., in protein networks, graph communities can be used to predict new protein complexes [King et al. 04]. There are many methods for partitioning a graph into clusters; NAViGaTOR supports four different network clustering methods. Three of these (cliques, RNSC, and MCL) are implemented as a plug-in to the NeAT toolbox [Brohee et al. 08], and one (k-clique communities) as a plug-in to CFinder [Adamcsek et al. 06].

Cliques: A clique is the strictest possible definition of a graph cluster. In a k-clique, k vertices are fully connected, i.e., there is an edge between every pair of distinct vertices. Cliques have an edge density of 1.

Restricted neighborhood search clustering (RNSC): Restricted neighborhood search clustering partitions a graph into disjoint sets of densely connected vertices using cost-based clustering [King et al. 11, King et al. 04]. RNSC starts with a random assignment of vertices to clusters, and then at every iteration moves one vertex to a different cluster to lower a cost function that depends on the number of intracluster and intercluster edges.[5]

Markov clustering (MCL): Markov clustering partitions a graph into disjoint sets using flow simulation [Van Dongen 98, Pereira-Leal et al. 04]. It uses stochastic matrices to simulate random walks on the graph and identify densely connected clusters of vertices. Of the four algorithms covered here, MCL is the only one that can use graph edge weights.

k-clique communities: Communities are constructed from chains of cliques that have many vertices in common [Palla et al. 05]. Specifically, a k-clique community is a union of all the k-cliques that are pairwise adjacent, in the sense that every k-clique in the community is reachable from every other one by traveling through a series of k-cliques that each share $k - 1$ vertices. In contrast to many other definitions of graph clusters, two k-communities may overlap, i.e., some vertices may be assigned to more than one k-community.

Overlapping communities can be found in diverse real-world networks: In social networks, individuals may belong to several groups (e.g., clubs, families); in protein networks, a protein may participate in multiple complexes; and in the World Wide Web, a document may be classified under multiple subject headings [Palla et al. 09].

3.4. Motifs and Graphlets

Motifs and graphlets characterize the local structure of a graph. Motifs are small subgraphs that occur significantly more often than at random. They may correspond to the key functional units in a graph [Shen-Orr 02, Milo et al. 02, Yeger-

[5] See (http://www.cs.utoronto.ca/~juris/data/rnsc/).

Lotem et al. 04]. For example, in information-processing networks, motifs may represent small computational circuits [Shen-Orr 02]. Different motifs characterize different types of graphs; for example, the motifs found in a graph of the World Wide Web are different from those found in a graph of a food web (where vertices represent species and edges connect predators to prey) [Milo et al. 02].

Graphlets are subgraphs that differ from motifs in two ways: Their frequencies are not necessarily higher than in random graphs and they must be induced subgraphs (i.e., they must contain all edges that connected their vertices in the full graph) [Pržulj et al. 06]. Graphlets have been used to develop more realistic models of biological networks [Pržulj et al. 04a]. They have also been effective for predicting protein function [Milenkovic and Pržulj 08], identifying cancer genes [Milenkovic 10], and discovering biological pathways [Guerrero et al. 08].

Several techniques are available for locating motifs and other important small subgraphs [Geraci et al. 2011, Wernicke and Rasche 06]. NAViGaTOR supports motif detection through a plug-in to the popular FANMOD program [Wernicke and Rasche 06]. Given a graph G in NAViGaTOR, the FANMOD plug-in converts G into the FANMOD format and generates an ID mapping file used to convert the detected motifs in the FANMOD format back to the notation used by G. An enumeration of motifs is obtained through FANMOD. Each motif m is imported back to NAViGaTOR as a subset of V.

3.5. Shortest Paths

The shortest path between two vertices is the shortest sequence of consecutive edges that must be traversed to reach one vertex from the other. More formally, a path can be defined as a set of vertices that can be ordered such that two vertices are adjacent if and only if they are consecutive in the ordering [West 01]. Identifying shortest paths is a key analysis task in many types of networks including the Internet and PPI networks. In PPI networks, identification of shortest paths can help predict disease genes, since such genes are often close together in interaction networks, and consequently, novel disease genes can be prioritized by determining their shortest paths to previously known disease genes [Franke et al. 06].

3.5.1. All Shortest Paths between Vertices. The ability to find all shortest paths between two vertices u and v is useful in different domains. For example, in the domain of Interior Gateway Protocols (IGPs), a highly practical routing option is equal-cost multipath routing, whereby incoming traffic is distributed evenly onto all edges that belong to the shortest paths to the final location [Altin et al. 09].

NAViGaTOR can retrieve all shortest paths between two vertices u and v, i.e., all paths that contain the minimum number of edges. More information about this function can be obtained from NAViGaTOR's user manual.[6]

3.5.2. Single-Source Shortest Path.

Dijkstra's algorithm is often used to compute single-source shortest paths [West 01]. For a graph G with weighted edges, given a source vertex s, the algorithm returns shortest paths from s to all other vertices in V (which includes the shortest path from a given source vertex s to a destination vertex v). This shortest-path algorithm is widely used in many routing problems. A well-known interior gateway protocol, OSPF (Open Shortest Path First) uses this algorithm [Coltun et al. 08].

NAViGaTOR implements Dijkstra's single-source shortest-path algorithm. If there exists more than one shortest path between u and v, the user can specify a weight function to select the preferred shortest path. The selection criteria for the preferred shortest path are as follows.

The length of the shortest path or distance between u and v is denoted by $d(u,v)$, and the weight for edge i is denoted by w_i. Without loss of generality, let s_{uv} denote the number of shortest paths between u and v. The total weight for a given shortest path is defined as

$$T_p = \sum_{i=1}^{d(u,v)} w_i.$$

The selection criterion among all shortest paths is

$$\max(T_{pj}),\ j \in \{1,\ldots,s_{uv}\}, \quad \text{or} \quad \min(T_{pj}),\ j \in \{1,\ldots,s_{uv}\}.$$

3.5.3. All-Pairs Shortest Paths between Subsets of Vertices.

The Floyd–Warshall algorithm computes all-pairs shortest paths (APSP) [Cormen et al. 90]. NAViGaTOR implements this algorithm to retrieve all shortest paths between any two given subsets. Let X and Y be subsets of V, and for vertices $x \in X$ and $y \in Y$, let P_{xy} denote all shortest paths between them. NAViGaTOR will return all shortest paths $\{P_{xy} \mid x \in X \wedge y \in Y\}$. More information about this function can be obtained from NAViGaTOR's user manual.

3.5.4. All-Pairs Shortest-Path-Traversal Counter.

In NAViGaTOR, shortest paths between all pairs of vertices $\{u, v \in V\}$ are calculated using the Floyd–Warshall algorithm [Cormen et al. 90]. If more than one shortest path between u and v exists, ties are broken arbitrarily. Let SP_i denote the set of vertices and SPE_i the set of edges

[6]Available online at http://ophid.utoronto.ca/navigator/.

associated with the shortest path between the ith pair of vertices $u, v \in V$, where $i \in \{1, \dots, \binom{|V|}{2}\}$. We define the traversal information $\mathrm{TN}(u)$ for each vertex $u \in V$, and $\mathrm{TE}(e)$ for each edge $e \in E$ as follows:

$$\mathrm{TN}(u) = \sum_{i=1}^{\binom{|V|}{2}} \begin{cases} 1 & \text{if } u \in \mathrm{SP}_i, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\mathrm{TE}(e) = \sum_{i=1}^{\binom{|V|}{2}} \begin{cases} 1 & \text{if } e \in \mathrm{SPE}_i, \\ 0 & \text{otherwise.} \end{cases}$$

The all-pairs shortest-path-traversal counter can be used in many areas. For example, in routing, an edge e with a high traversal count may get a great deal of traffic; one can consider alternative routes to avoid congestion through e. In a network showing the functional connectivity between brain anatomical regions in macaque visual cortex [Rubinov and Sporns 10, Felleman and Van Essen 91], the APSP traversal count can be used to highlight the importance of visual area V4 (Figure 9).

3.6. Random Subgraph Sampling

It is common to compare a feature associated with a particular subgraph—for example, edge density—to the same feature on a random subgraph. However, there are many ways to generate a random graph, and the method highly influences these comparisons. NAViGaTOR can randomly generate a background set of comparable subgraphs by sampling from the true graph. Given the number n of seed vertices, the subgraph depth d, and the background graph $G = (V, E)$, NAViGaTOR will generate subgraphs $S_i \subset G$ of the background graph by randomly choosing n vertices from V and then extracting them and their neighbors out to depth d.

Additionally, given sets of vertices $U_1, U_2, \dots, U_n$, where $U_i \subset V$, NAViGaTOR provides enrichment analysis by counting the number of vertices in each random subgraph that are from any of the sets U_i. This determines whether a particular subgraph is especially enriched in any particular vertex sets.

Note that random graph sampling is unrelated to a variety of random graph models, including the Erdős–Albert–Rényi (ER) model [Erdős and Rényi 59], the Barabási and Albert network model [Barabási and Albert 99], the geometric random network model [Pržulj et al. 04a], and the hierarchical network model [Ravasz et al. 02].

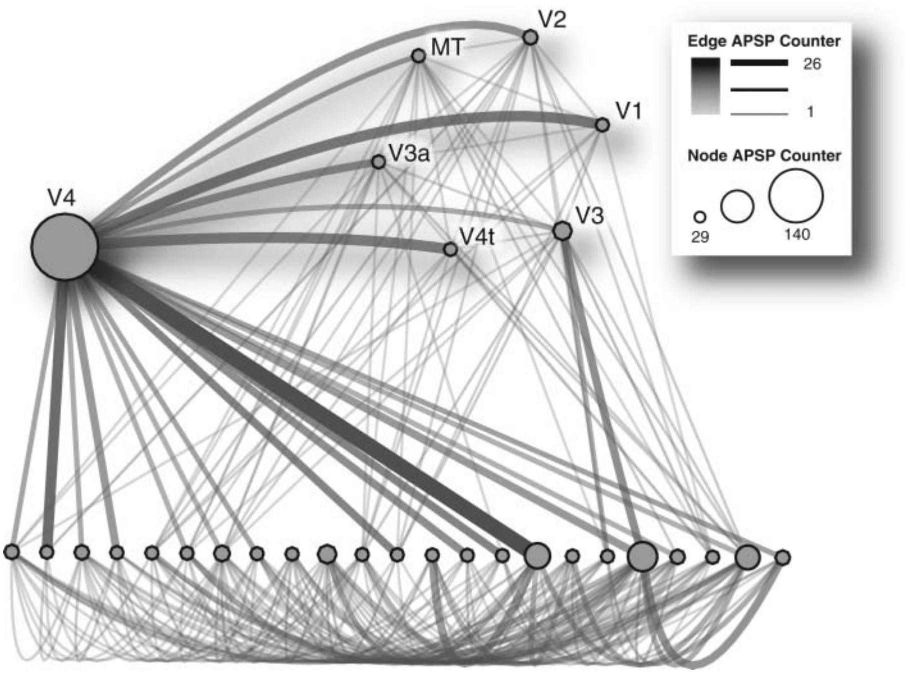

Figure 9. Functional connectivity between anatomical areas of macaque visual cortex. Labels are shown for the nodes corresponding to primary visual cortex (V1) and cortical areas V2, V3, V3a, V4, V4t, and V5/MT. Node size is proportional to node APSP traversal count, and edge shade and width are proportional to edge APSP traversal count.

We describe detailed steps for conducting a number of analyses in NAViGaTOR such as the computation of articulation points, betweenness centrality, clustering coefficient, edge density, diameter, single-source shortest paths, and all-pairs shortest paths in online tutorials.[7] We also describe the format of the results of each analysis and how to visualize them in the network.

4. Network Export

NAViGaTOR supports saving the network in diverse network file formats—GML, PSI-MI XML, BioPAX, Pajek, and tab-delimited—and in multiple image

[7]See http://ophid.utoronto.ca/navigator/sample_data/supplemental_materials/.

file formats—BMP, JPG, TIFF, PDF, and SVG. Among these, the PDF and SVG vector format images scale gracefully, and are ideal for publishing. However, the images produced in SVG for networks with tens of thousands of nodes and edges are increasingly time-consuming and difficult to open in popular vector graphic editing applications such as Adobe Illustrator and Inkscape. Although NAViGaTOR does not optimize output to the SVG format for these applications, it can render large networks with tens of thousands of objects in a few seconds as well as provide interactive functionality, while these applications require minutes to render the same networks statically. By harnessing the power of video cards commonly available in modern computers, NAViGaTOR makes interactive manipulations of large graphs feasible. Contrast this to vector-graphics editing applications that are sometimes not even able to render the same graph statically that NAViGaTOR can render dynamically.

Individual file formats consume storage space that depends not only on the information they represent but also on how they store it. Although XML provides a flexible standard characterized by openness and sharing of features, its level of detail can sometimes hinder scalability. NAViGaTOR is flexible in that it supports a number of file formats for data import and export, and thus it can integrate diverse data and interface with other tools easily.

5. Comparison with Other Network Visualization Software

A number of graph visualization and manipulation applications exist for various platforms. Potential users of these applications will make a decision based on factors such as the application's availability for the user's computing platform, its feature set, and its user interface. Some applications are more suitable for particular domains (social networks, biological network analysis, link analysis, etc.), whereas others are more generic. Here is a brief comparison of some of NAViGaTOR's capabilities with those of Cytoscape (a widely adopted application in the biological community) [Shannon et al. 03] and of Gephi (which advertises itself as "Photoshop for graphs") [Bastian et al. 09]. All three applications allow general-purpose network visualization, are available free of charge, and can run on multiple platforms.

While all three graph-visualization tools provide diverse options for network layouts, they differ in flexibility and speed of how these layouts can be selected and optimized for a given graph. NAViGaTOR's combination of regular, floating, and markup menus [McGuffin and Jurisica 09] provides faster access to relevant tools to select, move, rotate, scale, and lay out subgraphs and curve edges than either Gephi or Cytoscape.

Furthermore, NAViGaTOR has been benchmarked against other graphing applications in the biological domain. In particular, NAViGaTOR consistently loaded networks and rendered graph layouts in significantly shorter time and with a smaller memory footprint than Cytoscape (detailed comparison is available in supplemental material in [Brown et al. 09]). Additional review and comparison of many graph visualization and analysis tools in the biological domain can be found elsewhere [Gehlenborg et al. 10, Suderman and Hallett 07].

Each application contributes unique features, and may thus be better at different tasks. NAViGaTOR specializes in visual exploration of large-scale graphs (millions of nodes and edges) and integration with high-dimensional data by providing intuitive tools for these purposes. Moreover, all major applications support data exchange using accepted standard file formats. Further integration via sharable plug-ins is planned in the future as well.

6. Conclusion

Tools and systems should not limit users during data analysis and visualization; rather, they should empower them. We have designed NAViGaTOR to empower scientists conducting integrative computational biology analyses. First, network data may be imported in NAViGaTOR from distributed and heterogeneous sources using a variety of file formats. Second, NAViGaTOR is adaptable with an effective user interface, several layouts, and graph-analysis methods. Third, NAViGaTOR's support for multiple export formats and plug-in architecture make it compatible with many other tools and data repositories. Finally, NAViGaTOR is interactive, efficient, and scalable. NAViGaTOR provides the complete workflow required to perform network visualization and analysis. Pooling resources (with core services as well as many plug-ins) supports visual data mining in an intuitive tool. NAViGaTOR also scales well to large real-world data sets with hundreds of thousands of nodes and edges through optimized internal data structures. We are currently exploring better integration with Cytoscape [Shannon et al. 03] (please see [Brown et al. 09] for a detailed comparison between them). We are also continuing to provide user-driven options to facilitate combined network analysis and visualization. The NAViGaTOR software is freely available for academic use at http://ophid.utoronto.ca/navigator.

Acknowledgments. We thank Periklis Andritsos, Wing Xie, and Fiona Broackes-Carter for useful discussion. This research was supported in part by grants from the Natural Sciences and Engineering Research Council (NSERC# 104105), the Canadian Institute for Health Research (CIHR BIO-99745), the Canada Foundation for Innovation (CFI #12301 and #203383), the Ontario Research Fund

(GL2-01-030 and RE-03-020), and the Canada Research Chair Program to IJ. This research was funded in part by the Ontario Ministry of Health and Long Term Care (MOHLTC). The views expressed do not necessarily reflect those of the MOHLTC.

References

[Adamcsek et al. 06] B. Adamcsek, G. Palla, I. J. Farkas, I. Derenyi, and T. Vicsek. “CFinder: Locating Cliques and Overlapping Modules in Biological Networks.” *Bioinformatics* 22:8 (2006), 1021–1023.

[Aloul and Rawi 06] F. A. Aloul and B. A. Rawi. “Identifying the Shortest Path in Large Networks using Boolean Satisfiability”. In *3rd International Conference on Electrical and Electronics Engineering, 2006* 401, pp. 1–4, 2006.

[Altin et al. 09] A. Altin, B. Fortz, M. Thorup, and H. Ümit. “Intra-domain Traffic Engineering with Shortest Path Routing Protocols.” *Quarterly Journal of Operations Research* 7:4 (2009), 301–335.

[Antoniou and Harmelen 09] G. Antoniou and F. Harmelen. “Web Ontology Language: OWL.” In *Handbook on Ontologies*, pp. 91–110, 2009.

[Ashburner et al. 00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, et al. “Gene Ontology: Tool for the Unification of Biology.” *Nature Genetics* 25:1 (2000), 25–29.

[Barabási and Albert 99] A.-L. Barabási and R. Albert. “Emergence of Scaling in Random Networks.” *Science* 286 (1999), 509–512.

[Barabási and Oltvai 04] A. L. Barabási and Z. N. Oltvai. “Network Biology: Understanding the Cell’s Functional Organization.” *Nat. Rev. Genet.* 5:2 (2004), 101–113.

[Bastian et al. 09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. “Gephi: An Open Source Software for Exploring and Manipulating Networks.” In: *International AAAI Conference on Weblogs and Social Media*, pp. 361-362. Menlo Park, CA: AAAI Press, 2009.

[Boag et al. 02] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, J. Siméon, and M. Stefanescu. “XQuery 1.0: An XML Query Language.” *W3C Working Draft* 15, 2002.

[Borgatti et al. 09] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca. “Network Analysis in the Social Sciences.” *Science* 323:5916 (2009), 892–895.

[Brohee et al. 08] S. Brohee, K. Faust, G. Lima-Mendez, O. Sand, R. Janky, G. Vanderstocken, Y. Deville, and J. van Helden. “NeAT: A Toolbox for the Analysis of Biological Networks, Clusters, Classes and Pathways.” *Nucleic Acids Res* 36 (Web Server issue) (2008), W444–451.

[Brown and Jurisica 07] K. Brown and I. Jurisica. “Unequal Evolutionary Conservation of Human Protein Interactions in Interologous Networks.” *Genome Biology* 8:5 (2007), R95.

[Brown et al. 09] K. R. Brown, D. Otasek, M. Ali, M. J. McGuffin, W. Xie, B. Devani, I. L. Toch, and I. Jurisica. “NAViGaTOR: Network Analysis, Visualization and Graphing TORonto.” *Bioinformatics* 25:24 (2009), 3327.

[Cerami et al. 06] E. G. Cerami, G. D. Bader, B. E. Gross, and C. Sander. "cPath: Open Source Software for Collecting, Storing, and Querying Biological Pathways." *BMC bioinformatics* 7:1 (2006), 497.

[Cerami et al. 11] E. G. Cerami, B. E. Gross, E. Demir, I. Rodchenkov, Ö. Babur, N. Anwar, N. Schultz, G. D. Bader, and C. Sander. "Pathway Commons: A Web Resource for Biological Pathway Data." *Nucleic Acids Research* 39 (suppl. 1) (2011), D685.

[Coltun et al. 08] R. Coltun, D. Ferguson, J. Moy, and A. Lindem. "Rfc 5340 - OSPF for ipv6." Technical report, IETF, July 2008.

[Cormen et al. 90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* Boston: MIT Press and McGraw-Hill, 1990.

[Croft et al. 10] D. Croft, G. O'Kelly, G. Wu, R. Haw, M. Gillespie, et al. "Reactome: A Database of Reactions, Pathways and Biological Processes." *Nucleic Acids Research*, November 2010.

[Demir et al. 10] E. Demir, M. P. Cary, S. Paley, K. Fukuda, C. Lemer, et al. "The BioPAX Community Standard for Pathway Data Sharing." *Nature Biotechnology* 28:9 (2010), 935–942.

[Erdős and Rényi 59] P. Erdős and A. Rényi. "On Random Graphs I." *Publicationes Mathematicae* 6 (1959), 290–297.

[Farin and Hansford 00] G. Farin and D. Hansford. *The Essentials of Computer Aided Geometric Design.* Natick: A. K. Peters, 2000.

[Felleman and Van Essen 91] D. J. Felleman and D. C. Van Essen. "Distributed Hierarchical Processing in the Primate Cerebral Cortex." *Cereb. Cortex* 1:1 (1991), 1–47.

[Franke et al. 06] Lude Franke, Harm van Bakel, Like Fokkens, Edwin D. de Jong, Michael Egmont-Petersen, and Cisca Wijmenga. "Reconstruction of a Functional Human Gene Network, with an Application for Prioritizing Positional Candidate Genes." *Am. J. Hum. Genet.* 78:6 (2006), 1011–1025.

[Gajer and Kobourov 01] P. Gajer and S. Kobourov. "GRIP: Graph Drawing with Intelligent Placement." In *Graph Drawing*, pp. 104–109. New York: Springer, 2001.

[Gehlenborg et al. 10] N. Gehlenborg, S. I. O'Donoghue, N. S. Baliga, A. Goesmann, et al. "Visualization of Omics Data for Systems Biology." *Nat. Methods* 7:3 Suppl. (2010), S56–68.

[Geraci et al. 2011] J. Geraci, G. Liu, and I. Jurisica. *Algorithms for Systematic Identification of Small Sub-graphs,* Bacterial Molecular Networks, Series: Methods in Molecular Biology. To appear, 2011.

[Girvan and Newman 02] M. Girvan and M. E. J. Newman. "Community Structure in Social and Biological Networks." *Proceedings of the National Academy of Sciences of the United States of America* 99:12 (2002), 7821.

[Guerrero et al. 08] C. Guerrero, T. Milenkovic, N. Pržulj, P. Kaiser, and L. Huang. "Characterization of the Proteasome Interaction Network Using a QTAX-Based Tag-Team Strategy and Protein Interaction Network Analysis." *Proc. Natl. Acad. Sci. USA* 105:36 (2008), 13333–13338.

[Hidalgo 08] C. A. Hidalgo. "Thinking outside the Cube." *Physics World* 21 (2008), 34–37.

[Himsolt 96] M. Himsolt. "GML: A Portable Graph File Format." Available online (http://www.lkn.ei.tum.de/arbeiten/faq/guidelines/gml-tr.html), 1996.

[Hoffmann and Valencia 04] R. Hoffmann and A. Valencia. "A Gene Network for Navigating the Literature." *Nature Genetics* 36:7 (2004), 664.

[Holme et al. 02] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. "Attack Vulnerability of Complex Networks." *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* 65:5 Pt. 2 (2002), 056109.

[Jensen et al. 09] L. J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, et al. "STRING 8: A Global View on Proteins and Their Functional Interactions in 630 Organisms." *Nucleic Acids Research* 37 (Database issue) (2009), D412.

[Jeong et al. 01] H. Jeong, S. P. Mason, A. L. Barabási, and Z. N. Oltvai. "Lethality and Centrality in Protein Networks." *Nature* 411:6833 (2001), 41–42.

[Kanehisa et al. 10] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and M. Hirakawa. "KEGG for Representation and Analysis of Molecular Networks Involving Diseases and Drugs." *Nucleic Acids Res.* 38 (2010), D355–360.

[Karagiannis et al. 04] T. Karagiannis, M. Molle, and M. Faloutsos. "Long-Range Dependence: Ten Years of Internet Traffic Modeling." *IEEE Internet Computing* 8 (2004), 57–64.

[Kerrien et al. 07] S. Kerrien, S. Orchard, L. Montecchi-Palazzi, B. Aranda, A. F. Quinn, et al. "Broadening the Horizon: Level 2.5 of the HUPO-PSI Format for Molecular Interactions." *BMC biology* 5:1 (2007), 44.

[Kim et al. 08] Y. Kim, Hyuncheol Kim, Jin-wook Chung, Kwang-jong Cho, and Kisung Yu. "Network Stability Assessment Using the Number of Tree Adjacent to an Articulation Node". In *Computational Science and Its Applications-ICCSA 2008*, Lecture Notes in Computer Science 5073, pp. 1229–1241. Berlin Heidelberg: Springer-Verlag, 2008.

[King et al. 04] A. D. King, N. Pržulj, and I. Jurisica. "Protein Complex Prediction via Cost-Based Clustering." *Bioinformatics* 20:17 (2004), 3013–3020.

[King et al. 11] A. D. King, N. Pržulj, and I. Jurisica. *Protein Complex Prediction with RNSC*, Bacterial Molecular Networks, Series: Methods in Molecular Biology. To appear, 2011.

[Knuth 93] D. E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing.* Reading, MA: Addison-Wesley, 1993.

[Lukashin et al. 03] A. V. Lukashin, M. E. Lukashev, and R. Fuchs. "Topology of Gene Expression Networks as Revealed by Data Mining and Modeling." *Bioinformatics* 19:15 (2003), 1909–1916.

[May and Lloyd 01] R. M. May and A. L. Lloyd. "Infection Dynamics on Scale-Free Networks." *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* 64:6 Pt 2 (2001), 066112.

[McGuffin and Jurisica 09] M. J. McGuffin and I. Jurisica. "Interaction Techniques for Selecting and Manipulating Subgraphs in Network Visualizations." *IEEE Transactions on Visualization and Computer Graphics* 15:6 (2009), 937–944.

[Milenkovic 10] T. Milenkovic, V. Memisevic, A. K. Ganesan, and N. Pržulj. "Systems-Level Cancer Gene Identification from Protein Interaction Network Topology Applied to Melanogenesis-Related Functional Genomics Data." *J. R. Soc. Interface* 7:44 (2010), 423–437.

[Milenkovic and Pržulj 08] T. Milenkovic and N. Pržulj. "Uncovering Biological Network Function via Graphlet Degree Signatures." *Cancer Inform.* 6 (2008), 257–273.

[Milo et al. 02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. "Network Motifs: Simple Building Blocks of Complex Networks." *Science* 298:5594 (2002), 824–827.

[Newman 03] M. E. J. Newman. "The Structure and Function of Complex Networks". *SIAM Review* 45:2 (2003), 167–256.

[Newman 06] M. E. J. Newman. "Finding Community Structure in Networks Using the Eigenvectors of Matrices." *Physical Review E* 74:3 (2006), 36104.

[Orchard et al. 03] S. Orchard, H. Hermjakob, and R. Apweiler. "The Proteomics Standards Initiative." *Proteomics* 3:7 (2003), 1374–1376.

[Orchard et al. 10] S. Orchard, J. P. Albar, E. W. Deutsch, M. Eisenacher, P. A. Binz, and H. Hermjakob. "Implementing Data Standards: A Report on the HUPOPSI Workshop September 2009, Toronto, Canada." *Proteomics* 10:10 (2010), 1895–1898.

[Palla et al. 05] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. "Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society." *Nature* 435:7043 (2005), 814–818.

[Palla et al. 09] G. Palla, P. Pollner, A. L. Barabási, and T. Vicsek. "Social Group Dynamics in Networks." In *Adaptive Networks: Theory, Models and Applications (Understanding Complex Systems)*, edited by T. Gross and H. Sayama. New York: Springer, 2009.

[Pereira-Leal et al. 04] J. B. Pereira-Leal, A. J. Enright, and C. A. Ouzounis. "Detection of Functional Modules from Protein Interaction Networks." *Proteins* 54:1 (2004), 49–57.

[Perer and Shneiderman 08] A. Perer and B. Shneiderman. "Integrating Statistics and Visualization: Case Studies of Gaining Clarity during Exploratory Data Analysis." In *Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems, CHI '08*, pp. 265–274. New York: ACM, 2008.

[Pico et al. 08] A. R. Pico, T. Kelder, M. P. Van Iersel, K. Hanspers, B. R. Conklin, and C. Evelo. "WikiPathways: Pathway Editing for the People." *PLoS biology* 6:7 (2008), e184.

[Pržulj et al. 04a] N. Pržulj, D. G. Corneil, and I. Jurisica. "Modeling Interactome: Scale-Free or Geometric?" *Bioinformatics* 20:18 (2004), 3508–3515 .

[Pržulj et al. 04b] N. Pržulj, D. A. Wigle, and I. Jurisica. "Functional Topology in a Network of Protein Interactions." *Bioinformatics* 20:3 (2004), 340–348.

[Pržulj et al. 06] N. Pržulj, D. G. Corneil, and I. Jurisica. "Efficient Estimation of Graphlet Frequency Distributions in Protein–Protein Interaction Networks." *Bioinformatics* 22:8 (2006), 974–980.

[Ravasz et al. 02] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási. "Hierarchical Organization of Modularity in Metabolic Networks." *Science* 297 (2002), 1551–1555.

[Rubinov and Sporns 10] M. Rubinov and O. Sporns. “Complex Network Measures of Brain Connectivity: Uses and Interpretations.” *Neuroimage* 3 (2010), 1059–1069.

[Rzhetsky and Gomez 01] A. Rzhetsky and S. M. Gomez. “Birth of Scale-Free Molecular Networks and the Number of Distinct DNA and Protein Domains per Genome.” *Bioinformatics* 17:10 (2001), 988–996.

[Shannon et al. 03] P. Shannon, A. Markiel, O Ozier, N. S. Baliga, J. T. Wang, et al. “Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks.” *Genome Research* 13:11 (2003), 2498–2504.

[Shen-Orr 02] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. “Network Motifs in the Transcriptional Regulation Network of *Escherichia coli*.” *Nat Genet* 31:1 (2002), 64–68.

[Shreiner 99] D. Shreiner. *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.2*. Boston: Addison-Wesley, 1999.

[Strogatz 01] S. H. Strogatz. “Exploring Complex Networks.” *Nature* 410:6825 (2001), 268–276.

[Stumpf and Wiuf 05] M. P. Stumpf and C. Wiuf. “Sampling Properties of Random Graphs: The Degree Distribution.” *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* 72:3 Pt. 2 (2005), 036118.

[Suderman and Hallett 07] M. Suderman and M. Hallett. “Tools for Visually Exploring Biological Networks.” *Bioinformatics* 23:20 (2007), 2651.

[Van Dongen 98] S. Van Dongen. “A New Cluster Algorithm for Graphs.” Technical report. Amsterdam: Centre for Mathematics and Computer Science, CWI, Information Systems, 1998.

[Viau et al. 10] C. Viau, M. J. McGuffin, Y. Chiricota, and I. Jurisica. “The FlowVizMenu and Parallel Scatterplot Matrix: Hybrid Multidimensional Visualizations for Network Exploration.” *IEEE Transactions on Visualization and Computer Graphics* 16:6 (2010), 1100–1108.

[Wang et al. 09] P. Wang, M. C. Gonzalez, C. A. Hidalgo, and A. L. Barábasi. “Understanding the Spreading Patterns of Mobile Phone Viruses.” *Science* 324:5930 (2009), 1071–1076.

[Watts and Strogatz 98] D. J. Watts and S. H. Strogatz. “Collective Dynamics of “Small-World” Networks.” *Nature* 393:6684 (1998), 440–442.

[Weinberg 10] R. Weinberg. “Point: Hypotheses First.” *Nature* 464:7289 (2010), 678.

[Wernicke and Rasche 06] Sebastian Wernicke and Florian Rasche. “FANMOD: A Tool for Fast Network Motif Detection.” *Bioinformatics* 22:9 (2006), 1152–1153.

[West 01] D. B. West. *Introduction to Graph Theory*, 2nd edition. Upper Saddle River, NJ: Prentice hall, 2001.

[Wishart et al. 06] D. S. Wishart, C. Knox, A. C. Guo, S. Shrivastava, M. Hassanali, P. Stothard, Z. Chang, and J. Woolsey. “DrugBank: A Comprehensive Resource for in Silico Drug Discovery and Exploration.” *Nucleic Acids Research* 34 (Database Issue) (2006), D668. Available online (http://drugbank.ca); accessed 12-October-2010.

[Wishart et al. 07] D. S. Wishart, C. Knox, A. C. Guo, D. Cheng, S. Shrivastava, D. Tzur, B. Gautam, and M. Hassanali. “DrugBank: A Knowledgebase for Drugs, Drug Actions and Drug Targets.” *Nucleic Acids Research* 2007. Available online (http://drugbank.ca); accessed 12-October-2010.

[Yeger-Lotem et al. 04] E. Yeger-Lotem, S. Sattath, N. Kashtan, S. Itzkovitz, R. Milo, R. Y. Pinter, U. Alon, and H. Margalit. "Network Motifs in Integrated Cellular Networks of Transcription-Regulation and Protein–Protein Interaction." *Proc. Natl. Acad. Sci. USA* 101:16 (2004), 5934–5939.

[Yook et al. 02] S. H. Yook, H. Jeong, and A. L. Barabási. "Modeling the Internet's Large-Scale Topology." *Proc. Natl. Acad. Sci. USA* 99:21 (2002), 13382–13386.

[Zachary 77] W. W. Zachary. "An Information Flow Model for Conflict and Fission in Small Groups." *Journal of Anthropological Research* 33:4 (1977), 452–473.

[Zelezniak et al. 10] A. Zelezniak, T. H. Pers, S. Soares, M. E. Patti, and K. R. Patil. "Metabolic Network Topology Reveals Transcriptional Regulatory Signatures of Type 2 Diabetes." *PLoS Comput. Biol.* 6:4 (2010), e1000729.

Amira Djebbari, Ontario Cancer Institute, Princess Margaret Hospital/UHN, and the Campbell Family Institute for Cancer Research, Toronto, ON M5G 1L7, Canada. (amirad@gmail.com)

Muhammad Ali, Ontario Cancer Institute, Princess Margaret Hospital/UHN, and the Campbell Family Institute for Cancer Research, Toronto, ON M5G 1L7, Canada. (muhammad.ali.uhn@gmail.com)

David Otasek, Ontario Cancer Institute, Princess Margaret Hospital/UHN, and the Campbell Family Institute for Cancer Research, Toronto, ON M5G 1L7, Canada. (dotasek.deu@gmail.com)

Max Kotlyar, Ontario Cancer Institute, Princess Margaret Hospital/UHN, and the Campbell Family Institute for Cancer Research, Toronto, ON M5G 1L7, Canada (mkotlyar@uhnres.utoronto.ca)

Kristen Fortney, Department of Medical Biophysics, University of Toronto, the Ontario Cancer Institute, Princess Margaret Hospital/UHN, and the Campbell Family Institute for Cancer Research, Toronto, ON M5G 1L7, Canada (k.fortney@utoronto.ca)

Serene Wong, Department of Computer Science, York University, the Ontario Cancer Institute, Princess Margaret Hospital/UHN, and the Campbell Family Institute for Cancer Research, Toronto, ON M5G 1L7, Canada (swong@cse.yorku.ca)

Anthony Hrvojic, Ontario Cancer Institute, Princess Margaret Hospital/UHN, and the Campbell Family Institute for Cancer Research, Toronto, ON M5G 1L7, Canada (ahrvojic@uhnresearch.ca)

Igor Jurisica, Ontario Cancer Institute, Princess Margaret Hospital/UHN, and the Campbell Family Institute for Cancer Research, IBM Life Sciences Discovery Centre, TMDT Room 9-305, 101 College Street, Toronto, ON M5G 1L7, Canada (juris@ai.utoronto.ca)

Received January 18, 2011; accepted June 14, 2011.