

# Supplementary Material for NAViGaTOR: Network Visualization and Graphing At Toronto

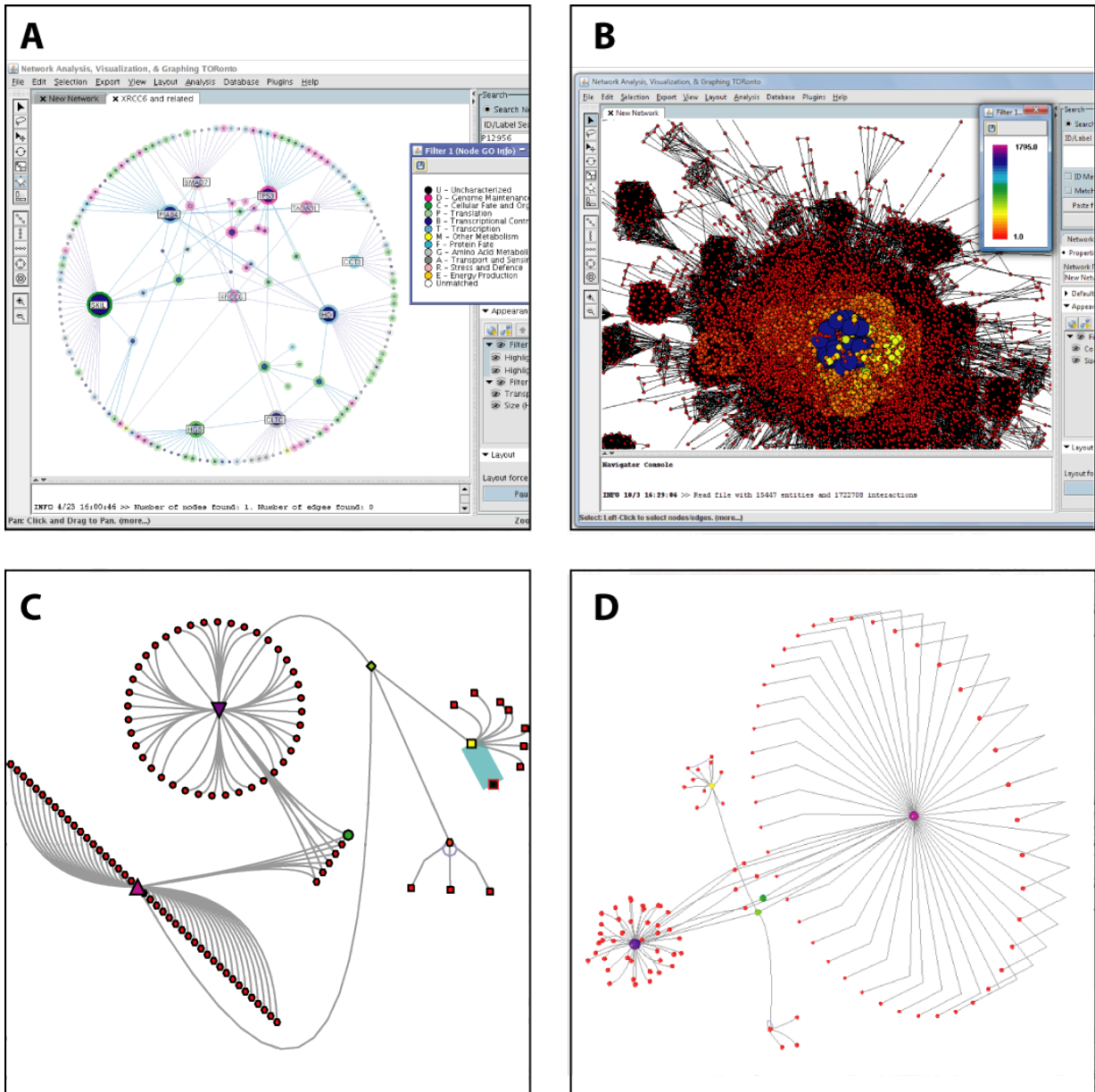
## S1 NAViGaTOR Image Gallery

NAViGaTOR has been designed not only to support interactive visualization and analysis of protein networks, but to allow the end-user to mark up graphs with diverse node and edge features (see Supplementary Figure 1.1). The graphs can be modified automatically, such as using “Filters” to control the node size (Supplementary Figure 1.1A), node color (Supplementary Figure 1.1B), or opacity, or manually, such as manually choosing colors, highlights, or node shapes. Additionally, edges can be annotated using color, thickness, or curved or bent paths. While the current version supports approximate shape control of curved edges, future enhancements to NAViGaTOR will provide the user with precise control over the splines that implement edge curves, and allow users to add arrowheads and line terminators to indicate activating or inhibitory reactions.

## S2 Biological Example Using NAViGaTOR

Since 2006, NAViGaTOR has been freely available to the research community on our web site (<http://ophid.utoronto.ca/navigator/>), and so far has been used to produce visualization of biological networks in 22 publications (Agarwal, et al., 2009; Barrios-Rodiles, et al., 2005; Brierley, et al., 2006; Brown and Jurisica, 2007; Cox, et al., 2007; Cox, et al., 2009; Dong, et al., 2009; Gortzak-Uzan, et al., 2008; Jurisica and Wigle, 2006; Jurisica, et al., 2007; Kislinger and Jurisica, 2006; Lau, et al., 2007; McGuffin and Jurisica, 2009; Mills, et al., 2009; Motamed-Khorasani, et al., 2007; Savas, et al., 2009; Seiden-Long, et al., 2006; Sodek, et al., 2008; Tomasini, et al., 2008; Wigle and Jurisica, 2007; Wu, et al., 2007; Zhu, et al., 2009), and continues to be actively used within the biological community. In addition to these publications and the example networks provided above, below we outline the basic steps taken to perform a typical analysis with NAViGaTOR.

The scenario considers a biologist that has performed a protein-protein interaction (PPI) screen to identify interactors to a small number of baits. They then performed a second screen to confirm the original interactions using a different PPI detection method. The latter method is capable of producing quantitative information about the interactions in addition to qualitative information. The biologist wanted to visualize the results of these two screens, and integrate additional data about known interactions to proteins in their screen.



**Figure 1.1 - Sample Images Produced by NAViGaTOR.**

NAViGaTOR provides many features to enable effective layout and annotation of networks. **A)** Screen-capture showing nodes on a circular layout, with hub nodes automatically scaled according to their degree (connectivity). **B)** The scalability of NAViGaTOR is demonstrated by rendering an extremely large network (1.7 million edges; see 'MouseNet', Table 3.1) in under 20 minutes. Automated filters were used to color nodes by degree. **C)** Circular and linear layouts are shown for a model network, along with both curved and bent edges. We are currently experimenting with curved edges to identify an optimal interface for users to manipulate complex curves in the networks. **D)** A combination of layouts and edge styles is used to illustrate the presence of highly connected hubs in a 3-dimensional rendering of this model network.

A text-based, tab-delimited file was prepared with the screen results, as shown in Table 2.1 (protein SwissProt IDs were randomly generated, as the original data has not yet been published). Each line describes an interacting pair of proteins. The SwissProt IDs (SPID) will be used to annotate each node, and the “Method” column will be used as an edge feature. Additional data captured by the screens is shown in the right-most columns, and will also be loaded as additional edge features.

**Table 2.1 - Example of the tab-delimited file used to load the network into NAViGaTOR**

Target	Target SPID	Bait	Bait SPID	Method	AvgMinus	AvgPlus	AvgPlus-AvgMinus
Protein1	P31946	Bait 1	P60709	Both	3.6	0.5	-3.1
Protein3	Q04917	Bait 1	P60709	Both	3	5.6	2.6
Protein4	P31947	Bait 2	Q562R1	Method1			
Protein5	P27348	Bait 1	P60709	Average	3.5	3.7	0.2
Protein5	P27348	Bait 2	Q562R1	Method1			
Protein6	P13746	Bait 2	Q562R1	Method1			
Protein6	P13746	Bait 3	Q71U36	Method1			
Protein7	P01889	Bait 1	P60709	Average	1.4	3.5	2.1
Protein7	P01889	Bait 2	Q562R1	Method1			
Protein8	P30456	Bait 1	P60709	Method1			
Protein9	P16188	Bait 3	Q71U36	Method1			
Protein10	P05534	Bait 1	P60709	Both	6.5	19.7	13.2
Protein10	P05534	Bait 2	Q562R1	Average	3.3	5.9	2.6
Protein10	P05534	Bait 3	Q71U36	Average	3	3.1	0.1

In NAViGaTOR, the graph is loaded through the “File->Open” menu item, the file type is selected as “.txt – Tab-delimited text”, and the file is selected in the File Chooser. After hitting “Okay”, the file wizard will walk the user through opening the file, and selecting columns as node or edge features. Once the user steps through the wizard, their network will be rendered automatically in NAViGaTOR (see Figure 3.1).

At this point, nodes are labeled by their SwissProt IDs, and edges as SwissProt ID pairs (i.e., P31946:P60709). Node names can be provided in the text file, or they can be retrieved via an Internet connection. To use the Internet connection (which will pull annotations from the Interologous Interaction Database (Brown and Jurisica, 2005; Brown and Jurisica, 2007)), use the “Database” menu and select “Get Ontology/Description for Nodes”. A dialogue will open and ask for the ID type (in this case SWISSPROT\_ID) and Organism (in this case HUMAN). After hitting Okay, NAViGaTOR will retrieve both protein name and Gene Ontology annotations for each node. To apply these annotations to all nodes, right-click on white-space in the graph panel, select the ‘Node->Appearance (All)’ menu item, and then select “Filters...”. Add a new node filter based on the Gene Ontology feature, then click “All”, and “Done”. At this point, all nodes are now

colour-coded according to the GO filter just created. Users can create filters for any node features. For instance, one can create a node size filter based on the degree of the nodes, making more highly-connected nodes larger, or on gene/protein expression (e.g., mapping level of expression to node color, size or shape). A similar filtering can be done on edges analogously, as described further below.

The second step is often to use the search dialogue to identify the nodes corresponding to the bait proteins. It will be important to spread these out in order to appreciate the overall graph structure. Once the baits are selected, their positions can be fixed, and they can be spread out manually or using automated circular and linear layout tools, along with scaling and rotating the selected nodes. From this point, the user can adjust the position of individual nodes or subsets, using automated tools to spread sets of nodes out, and fixing them in new locations. While the user interacts with the graph, NAViGaTOR continues to apply the force-directed layout to non-fixed nodes. In this way, the user can interactively assist in the untangling of some complicated networks.

In addition to laying out their own network, the user can automatically search the I2D database for additional, known or predicted protein interactions. These interactions can be merged into the original network, filtered, and selectively removed (for example, removing all low-confidence interactions). The goal of such manipulations is to enable the user to add to their network knowledge of interactions that have already been published in the literature and curated in online databases. Alternatively, the user could copy and paste from a second NAViGaTOR network, which easily facilitates the combining of multiple screens.

Once the user has found satisfactory node locations, they can turn their attention to adjusting the edges to convey useful information. Similar to the node filters, edge filters can be created to automate colouring, type or thickness changes according to any arbitrary feature. Examples of such filters are shown in Figure 3.2.

Finally, the user can: export their networks to several standard file formats, such as GML or PSI-MI, that allow loading into other software; export their networks to image formats, such as SVG, that allow the editing of all nodes and edges in programs such as Adobe Illustrator; or they can save their results for later processing in optimized NAViGaTOR XML format.

While this scenario only demonstrates a small example of examining the results of multiple protein-protein interaction screens, NAViGaTOR will facilitate a range of network analysis functions, such as finding hubs and articulation points, network statistics, shortest paths and quasi-cliques, and supports a full range of

enrichment analyses by automatically generating and comparing a given network to random networks, and identifying overlap with other sources, such as KEGG, cancer gene lists, and GO. Users can colour code gene expression data in node filters to show up- and down-regulation, or edge colours to show correlations in expression. Powerful search and node feature tools (shapes and colours) allow users to search for and mark proteins known to be mutated in cancer. The current NAViGaTOR enables visual queries against the I2D (<http://ophid.utoronto.ca/i2d>) and cPath (<http://cbio.mskcc.org/cpath/>) databases, and the plug-in interface will allow users to extend the capabilities, to connect to other PPI, pathway or protein databases, and to extend the analysis capabilities of the software. See the legend for Figure 3.2 for more complete descriptions of the manipulations done in the described scenario.

## **S3 Layout Comparisons**

### ***S3.1 Test Environment***

These tests were performed on a machine with two Pentium(R) 4 CPUs (3.6GHz, dual-core), 2GB of RAM, and running Microsoft Windows XP Professional Version 2002, Service Pack 2. The tests were performed with the following applications:

- NAViGaTOR version 2.0.8 (<http://ophid.utoronto.ca/navigator>)
- Cytoscape version 2.6.0 (<http://www.cytoscape.org>)
- VisANT version 3.29 (<http://VisANT.bu.edu/>)

The sample networks loaded into each application were sample files from the databases in Table 3.1, and were loaded from tab-delimited text files. In cases where the file could be interpreted with several different graph structures, such as BioPax files, the default representation from Cytoscape was exported as a tab-delimited file, which was subsequently loaded into NAViGaTOR and VisANT. Only node and edge names were loaded into each application.

The networks were chosen as representative of large, biologically-based networks that would typically be used by the average researcher.

These tests were used to illustrate the difference between the visual appearances and run-times of the different graph layout algorithms. While difficult to quantify, empirically, more effective layout algorithms will produce a more manageable and intuitive layout in shorter time.

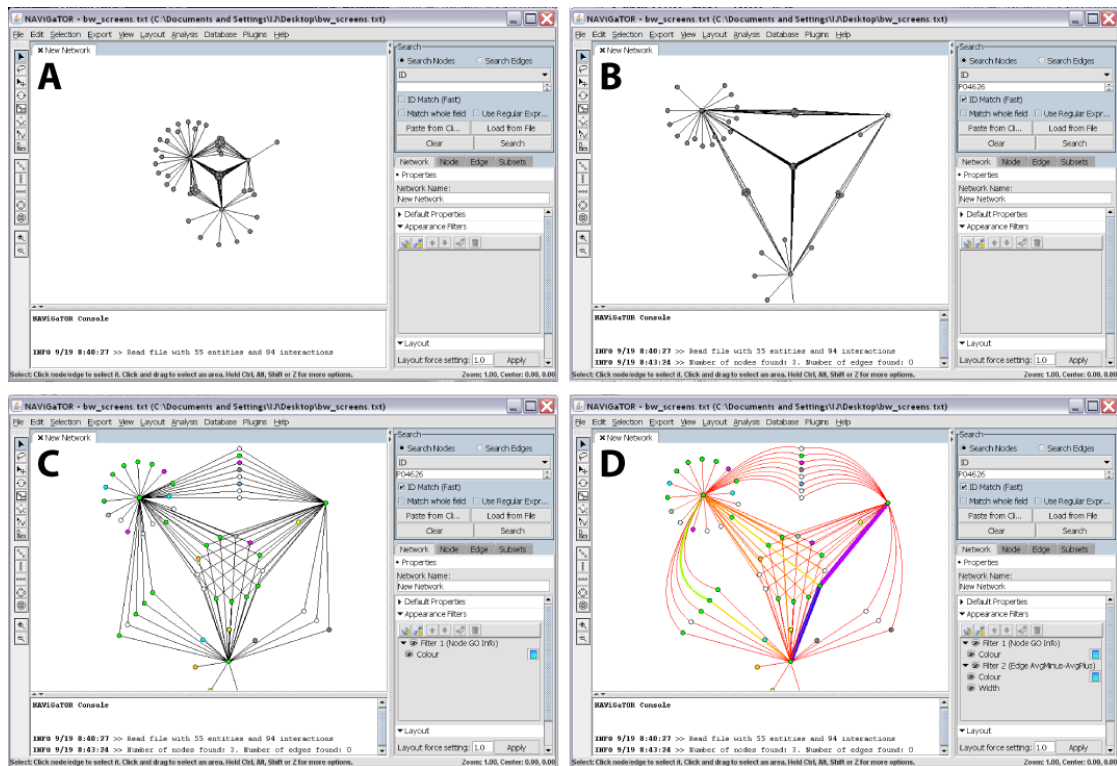
**Table 3.1 - Properties of Networks Used in Testing.**

<b>Database</b>	<b>Nodes</b>	<b>Edges</b>	<b>Undirected Edges</b>
BioPax <i>C. elegans</i> ( <a href="http://www.reactome.org/">http://www.reactome.org/</a> )	7,819	9,111	9,073
BioGRID <i>D. melanogaster</i> ( <a href="http://www.thebiogrid.org/">http://www.thebiogrid.org/</a> )	7,545	28,323	25,464
BIND Human	19,906	31,708	31,345
MouseNet ( <a href="http://www.functionalnet.org/mousenet/">http://www.functionalnet.org/mousenet/</a> )	15,447	1,722,708	1,722,708

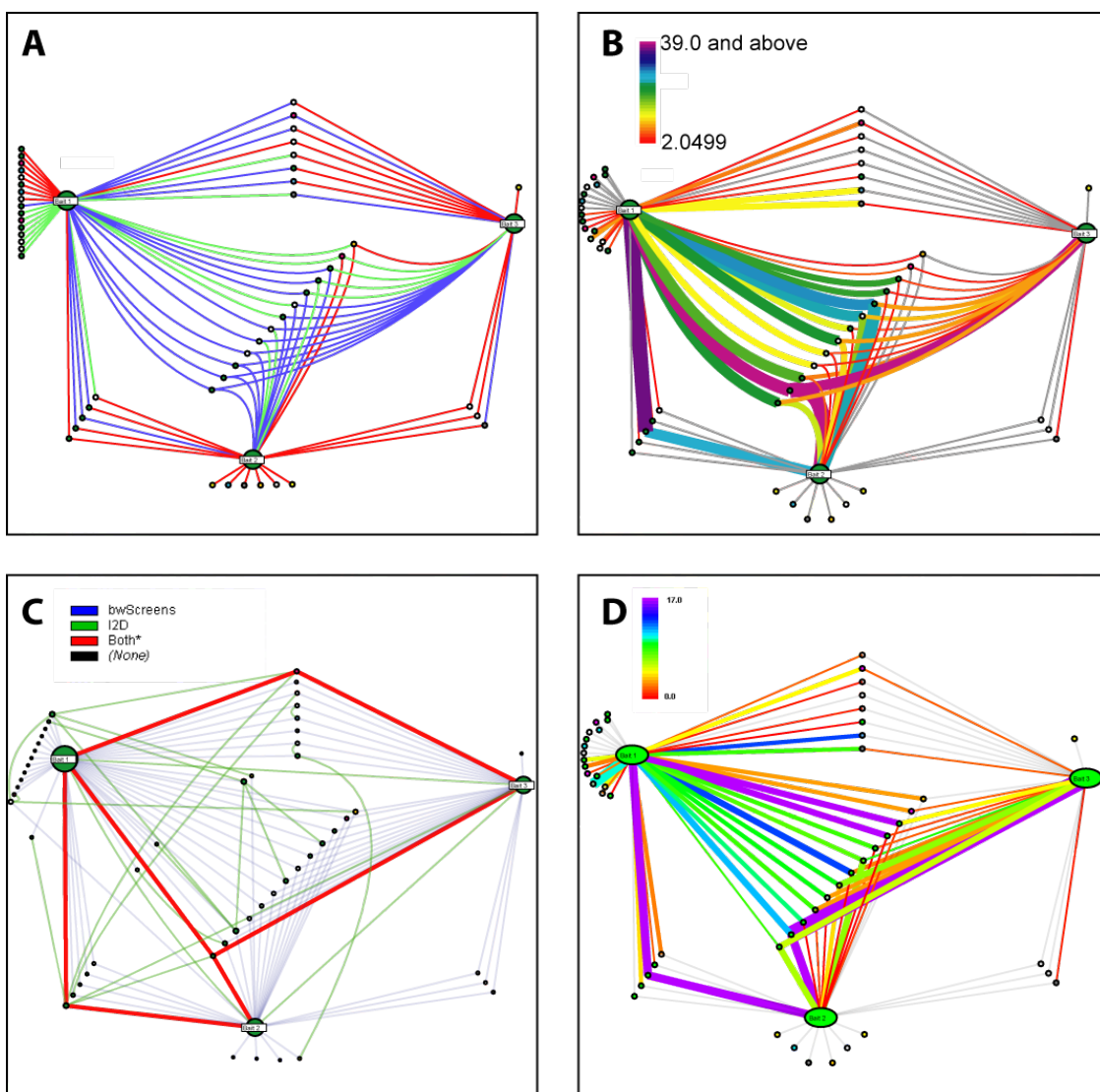
### **S3.2 Method**

Each network from Table 3.1 was loaded into each of NAViGaTOR, Cytoscape and VisANT, and a single layout algorithm was executed with the intent to run until completion. The value recorded in the evaluation indicates the time from initialization of the layout algorithm, to the point at which a stable layout was produced. In the case of dynamic layouts, which continue to optimize indefinitely (such as VisANT's), screen captures of the graph were taken at intervals during the iteration process. Supplementary Figures 3.3 – 3.5 show representative graphs recorded at approximately the same time at which NAViGaTOR's output was produced, as well as at a cutoff of 20 minutes. If no output was produced after 20 minutes, or the algorithm failed to run, this was noted in the results. For VisANT, the program was run in scripting mode, allowing it run with less overhead, and therefore execute faster.

Originally NAViGaTOR was to be evaluated alongside three other graph visualization packages: VisANT, Cytoscape and Osprey. Osprey failed to load any of the networks in Table 3.1 in under 20 minutes, and thus was dropped from the evaluation. Cytoscape also failed to render the MouseNet network in under 20 minutes. Though VisANT did not produce any output for MouseNet in our comparison, it is reportedly able to render this network in over five hours; its output can be examined at [http://VisANT.bu.edu/vmanual/cmd.htm#to\\_perform\\_time-consuming\\_tasks](http://VisANT.bu.edu/vmanual/cmd.htm#to_perform_time-consuming_tasks).



**Figure 3.1 - Screen-captures outlining the development of a biological graph.** **A)** After importing the data into NAViGaTOR, a simple graph is produced. **B)** The proteins used as baits in the assay are then found through 'Search' dialogue by entering in their SwissProt IDs. They are then fixed in position, and spread apart using the scaling tool. **C)** Subsets of nodes are manually selected, fixed in position, aligned using linear and circular shape tools, and dragged into position. At this point, the protein and GO annotations have been imported from I2D (<http://ophid.utoronto.ca/i2d>) through the 'Database' menu, and a node filter has been applied (lower-right panel) to automate colouring. **D)** The 'Bezier' tool was then used to curve some edges for aesthetic effect (which can sometimes provide easier visual interpretation), and an edge filter was applied (lower-right NAViGaTOR panel) to automatically colour the edges and vary the edge thickness in proportion to an imported edge feature, in this case an interaction binding strength.



**Figure 3.2 – Biological networks showing orthogonal protein interaction screens.**

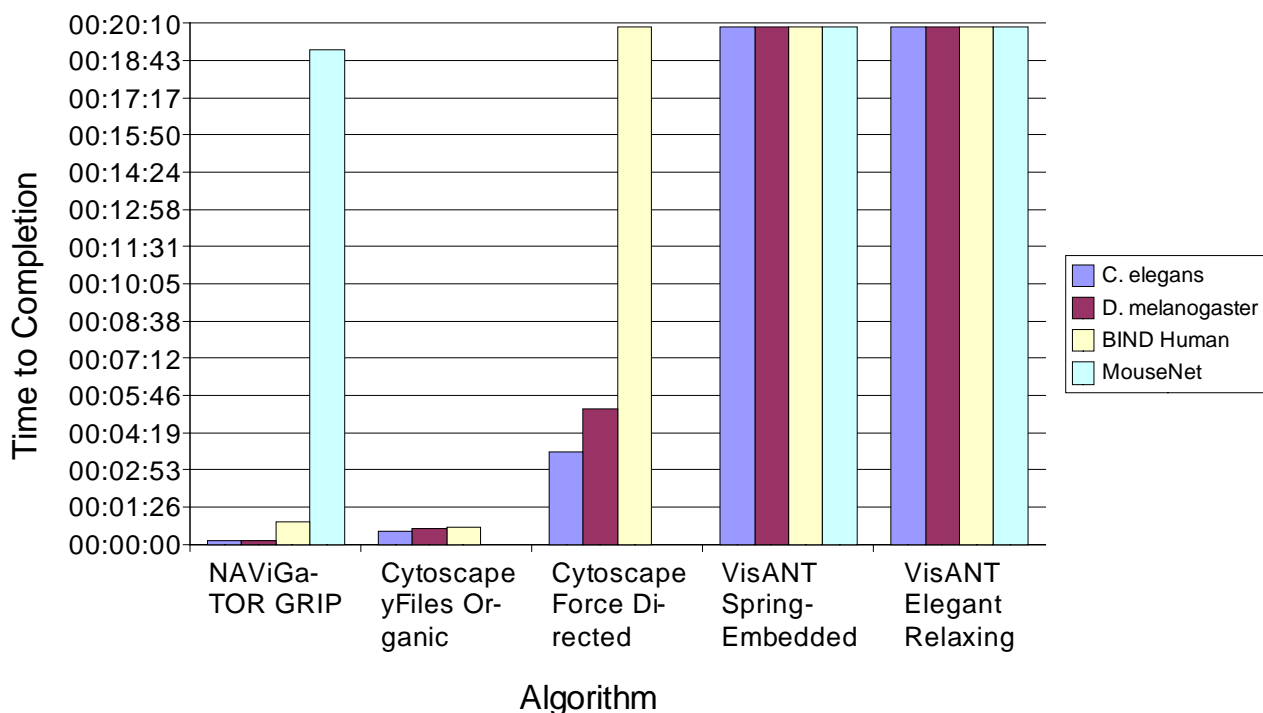
The same graphs from Figure 3.1 were altered using NAViGaTOR filters and database searching against I2D. Some edges were curved for easier viewing, and nodes were coloured by the GO node filter. **A)** Edges were coloured using an edge filter to indicate whether they were detected in one screen (red), both screens (green), or using an alternative scoring for both screens (blue). **B)** Edge filters can also be used to indicate binding strength, a feature that was imported from the text file. This edge filter included both colour and edge thickness. The colour-bar legend was also generated by NAViGaTOR. **C)** I2D was searched to overlay what is already known about this set of proteins. This operation involved selecting all nodes, creating a subset in NAViGaTOR (to make it easier to find the nodes later), searching I2D through the “Database” menu, inverting the selection (to select new nodes that were brought in), and deleting the current node selection. As a result known interactions from the I2D database are added to the network without adding additional nodes. (Screen edges (blue), I2D (green), and both (red).



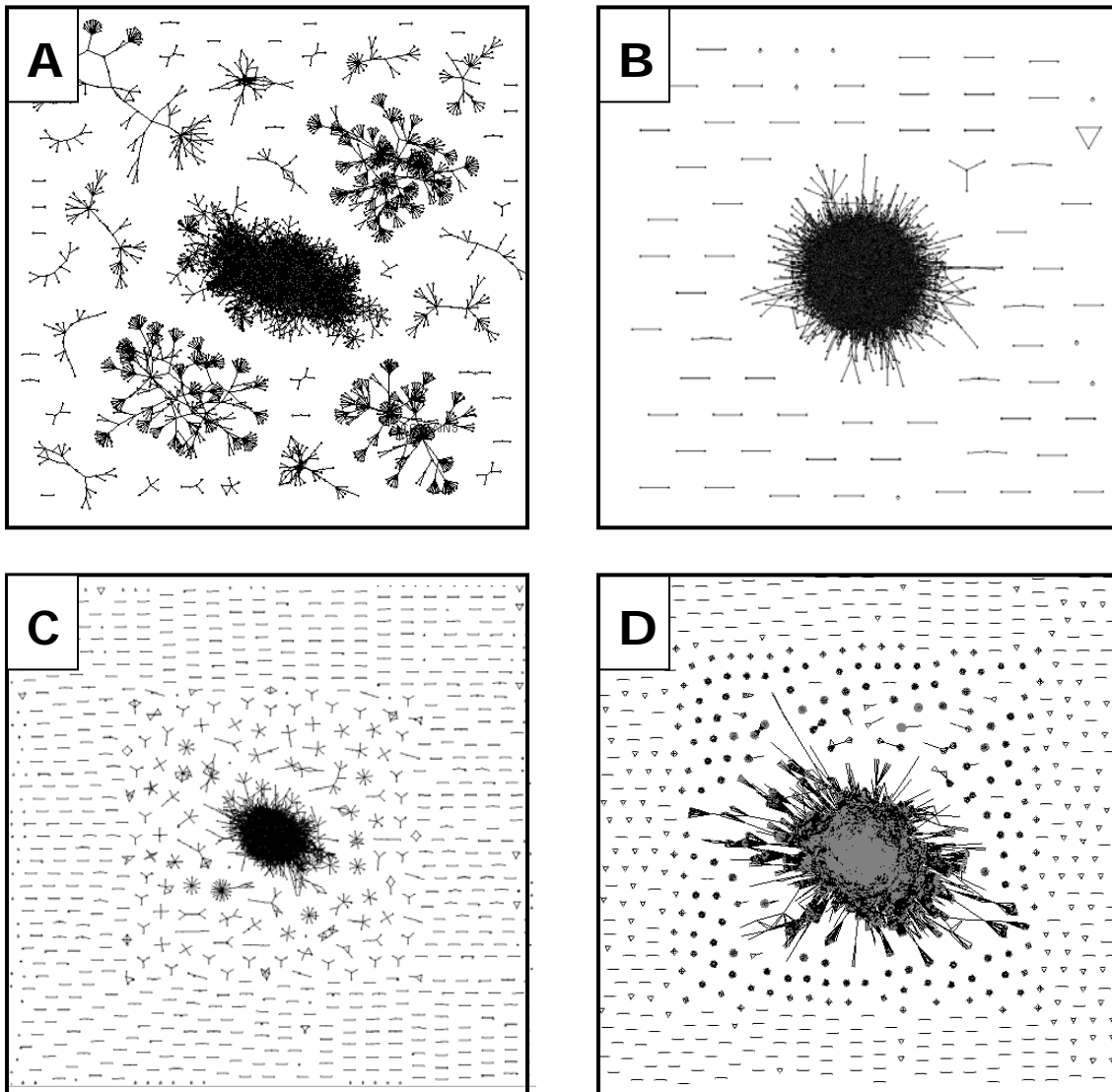
**Figure 3.2 (con't).** In this case, the legend was added after importing the network into Adobe Illustrator. **D)** Finally, an edge filter was created to demonstrate the differences in binding between two network states, for example, in the presence or absence of a ligand. The colour-bar shows that purple, thick edges show a higher difference in binding between the two states, while the red, thin edges show now difference.

### S3.3 Results

NAViGaTOR's GRIP algorithm generates useable layouts faster than Cytoscape and VisANT in most situations (see Supplementary Figure 3.3). In cases where NAViGaTOR is outperformed in speed, it offers a more intuitive layout than Cytoscape or VisANT (see Supplementary Figure 3.3C versus Supplementary Figure 3.4C for comparison).

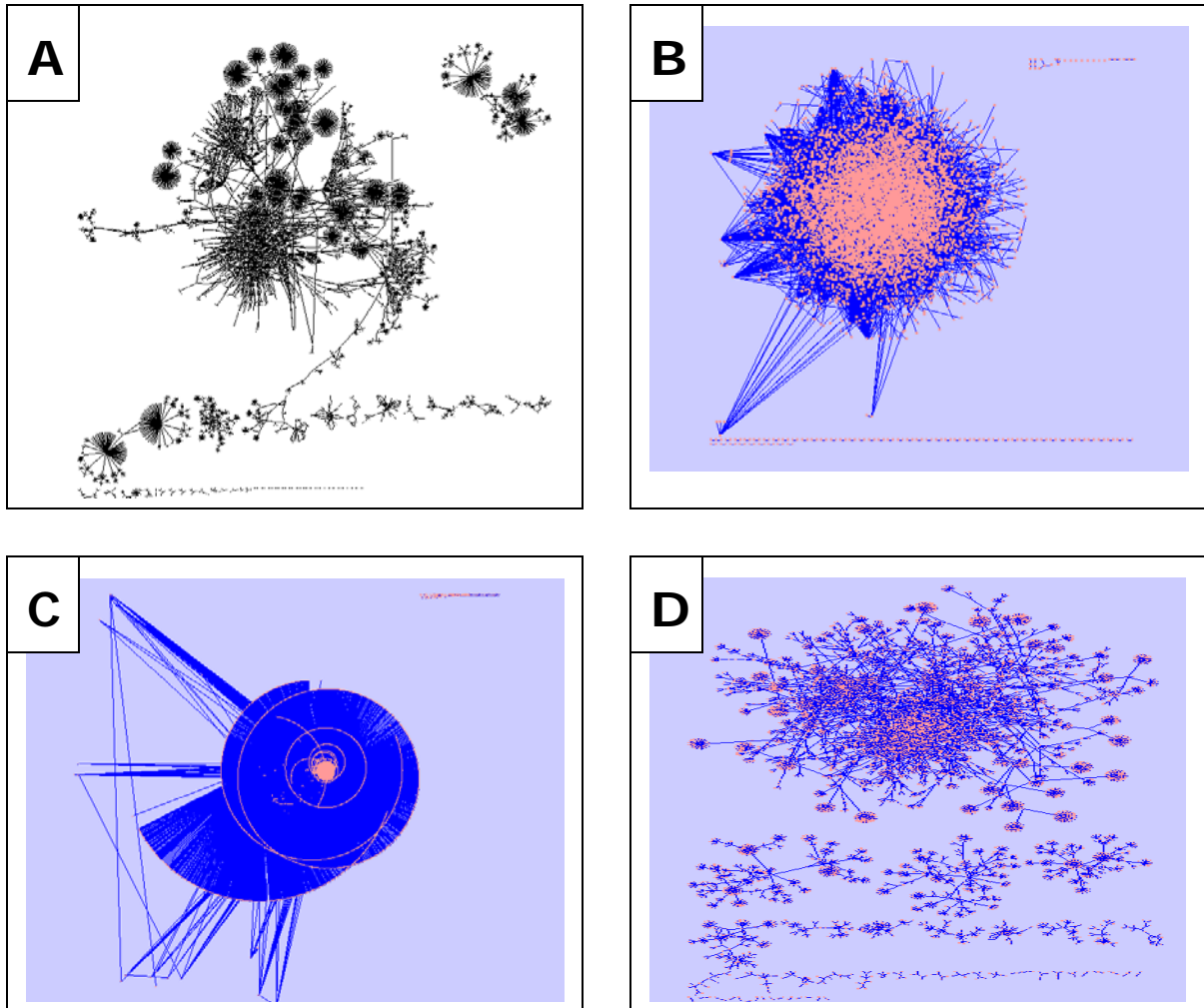


**Figure 3.3 - Measured Performance in Rendering Protein-Protein Interaction Networks.** NAViGaTOR, Cytoscape and VisANT were all used to render worm, fly, human and mouse PPI networks, and the time to generate a rendered network was recorded. Trials were terminated if a network was not produced after 20 minutes. The network sizes were as follows: Worm – 7,819 proteins, 9,073 interactions; Fly – 7,545 proteins, 25,464 interactions; Human – 19,906 proteins, 31,345 interactions; and Mouse – 15,447 proteins and 1,722,708 interactions.



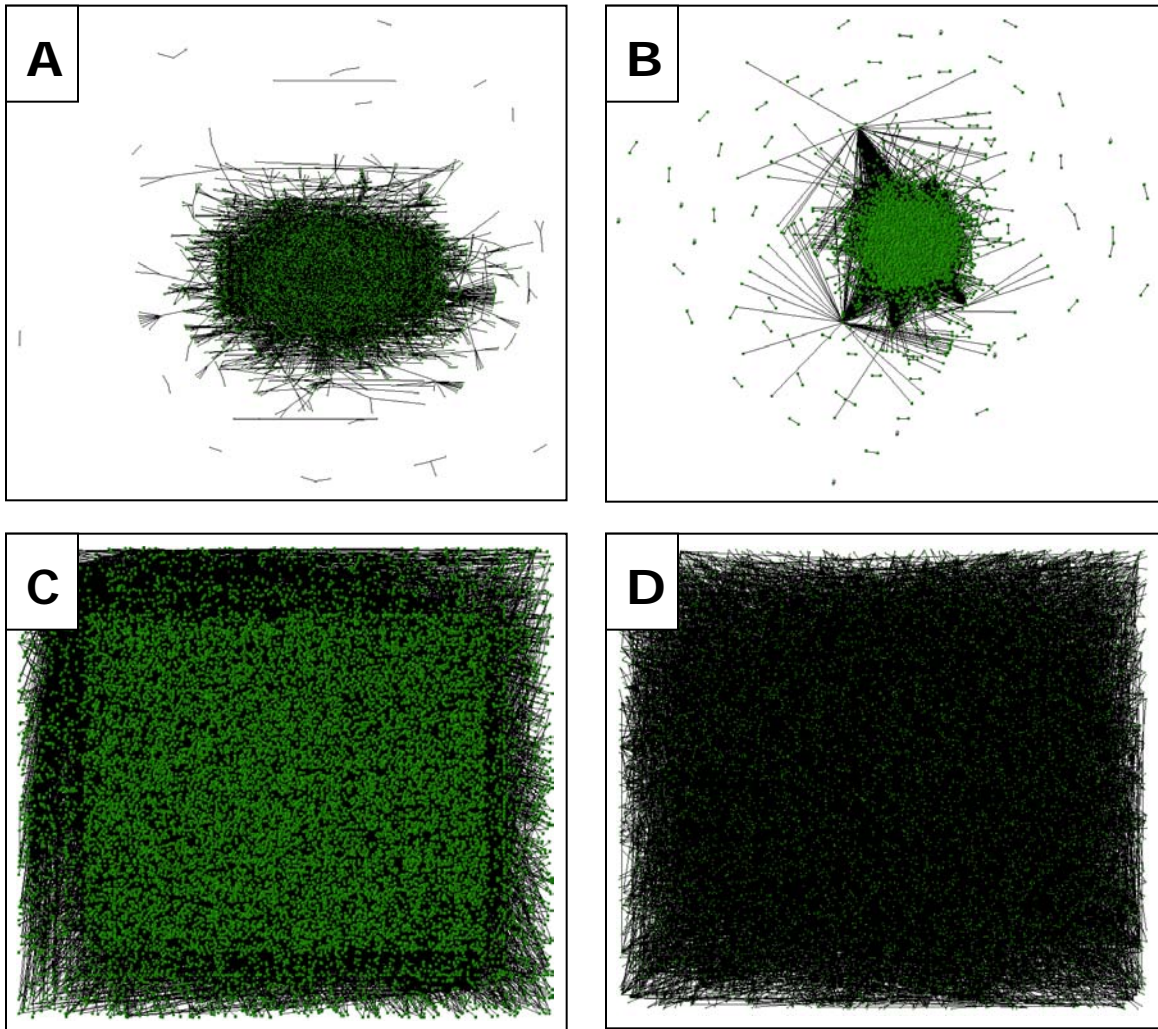
**Figure 3.4 - Networks Rendered Using NAViGaTOR**

The networks from Table 3.1 were rendered in NAViGaTOR using the default GRIP force-directed layout. **A)** The *C. elegans* network, consisting of 7,819 nodes and 9,073 edges, was rendered in seven seconds. This graph includes one large connected component (middle, 5991 nodes, 7204 edges) and 47 disconnected subgraphs. **B)** The *D. melanogaster* network, comprising 7,545 nodes and 25,464 edges in 68 disconnected subgraphs, was rendered in eight seconds. **C)** The BIND Human network was rendered in 52 seconds. The network comprises 19,906 nodes and 31,345 edges in 785 disconnected subgraphs. **D)** The MouseNet network was rendered in NAViGaTOR in 19 minutes and 7 seconds. The network comprises 15,447 nodes and 1,722,708 edges.



**Figure 3.5 - Networks Rendered Using Cytoscape**

The networks from Table 3.1 were loaded into Cytoscape and rendered, if possible. **A)** The *C. elegans* network was rendered using the yFiles Organic algorithm in 30 seconds. The network is identical to that in Supplementary Figure 3A, comprising 7,819 nodes and 9,073 edges. **B)** The *D. melanogaster* network was rendered using the yFiles Organic algorithm in 35 seconds. The network is identical to that in Supplementary Figure 3B, comprising 7,545 nodes and 25,464 edges. **C)** The BIND Human network was rendered using the yFiles Organic algorithm in 39 seconds. The network is identical to that in Supplementary Figure 3C, comprising 19,906 nodes and 31,345 edges. **D)** The *C. elegans* network from **A)** was rendered again in Cytoscape, this time using the default force-directed algorithm. While more similar to the output from NAViGaTOR's GRIP algorithm (Supplementary Figure 3A), Cytoscape required 3 minutes and 33 seconds to generate its output.



**Figure 3.6 - Networks Rendered Using VisANT.** The networks from Supplementary Table 3.1 were loaded into VisANT and rendered, if possible. **A)** The *C. elegans* network (7,819 nodes and 9,073 edges) was loaded, and a screenshot was captured following 20 minutes of optimization using the Relaxing algorithm. The comparable network rendered by NAViGaTOR is shown in Supplementary Figure 3A. **B)** The *D. melanogaster* network (7,545 nodes, 25,464 edges) was optimized using the Relaxing algorithm, and a screenshot was taken after 20 minutes of rendering. **C)** The Relaxing algorithm performed poorly on the BIND Human network (19,906 nodes, 31,345 edges), showing little improvement after 20 minutes of rendering. **D)** Similarly, the Spring Embedded layout algorithm failed to optimize the *C. elegans* network from Supplementary Figure 3A after 20 minutes of rendering.

## S4 Performance and Function Testing

### S4.1 File Loading Test Environment

To demonstrate NAViGaTOR's robust file handling capability, network files of various types (i.e., PSI-MI XML1.0, PSI-MI XML 2.5, and PSI MI-TAB), various sizes (from 1kB to 378MB), and from diverse databases (i.e., BioGrid (Stark, et al., 2006), DIP (Xenarios, et al., 2000), IntAct (Hermjakob, et al., 2004), MINT (Zanzoni, et al., 2002), NCI Pathways, and Reactome (Joshi-Tope, et al., 2005)) were obtained and tested. The computers used for testing are described in Table 4.1. All Java Virtual Machines (JVMs) were allocated 1,024MB of heap space (memory) to perform the operations, although the 'Tiger' machine and RedHat x86 only had 512MB of physical memory.

**Table 4.1 - Specifications of the computers used in our test suite.**

Computer	CPU	Memory	OS
Leopard	Intel Core 2 duo 2.54Ghz (2 cores)	4GB	Leopard OS X 10.5.8 MacBook Pro 5.5
Tiger	Power PC G4 (1.2) 1.25Ghz (1 core)	512 MB	Mac Mini Power Mac 10.1, Tiger 10.4.11
RedHat32	Intel Celeron 1.80GHz	512 MB	RedHat Linux i686 2.6.9-11.EL
RedHat64	Dual-Core AMD Opteron 3.0GHz (x2)	4GB	x86_64 GNU/Linux 2.6.18-53.el5
Vista64	Intel Xeon 2 CPUs x4 2.66GHz cores	16GB	Vista Business x64 6.0.6001 SP1
XP32	4 CPU Pentium 3.6 Ghz	2GB	Windows XP Professional v.2002 SP2
XP64	Intel Xeon 3Ghz 4 cores	4GB	Windows XP x64 Professional 5.2.3790 SP2

The test files were categorized by file size as 'small', 'medium', and 'large'. 'Small' files were the smallest obtainable from an individual database, 'large' files were the largest obtainable, and 'medium' was the file that was closest to the mean file size provided by a database. In some cases, only a single download file was available, and thus it was assigned to a group that appeared representative of the overall file size. Complete file details are presented in Table 4.2 and Table 4.3. The time required to load each file and file type was recorded, and the results are summarized in Table 4.4.

**Table 4.2 - Details of files used in testing NAViGaTOR file loading.**

Database	File Type	Small			Medium			Large		
		Nodes	Edges	Size (MB)	Nodes	Edges	Size (MB)	Nodes	Edges	Size (MB)
BioGrid	PSI 1.0	2	1	0.005	7,294	25,133	39.6	27,929	167,418	282.4
BioGrid	PSI 2.5	2	1	0.006	1,562	11,570	22.6	5,608	95,315	218.4
BioGrid	TAB	2	1	0.003	8,632	26,969	5.3	27,929	167,418	26.9
DIP	MI-TAB	2	1	0.001	3,077	6,990	2.0	11,643	22,864	5.3
DIP	PSI 2.5	2	1	0.004	2,423	4,916	12.7	7,505	22,884	39.9
IntACT	PSI 1.0	2	3	0.007	2	1	0.0	2,551	77,953	10.2
IntACT	PSI 2.5	2	1	0.015	284	556	2.9	2,551	77,953	41.6
IntACT	MI-TAB	NA	NA	NA	NA	NA	NA	43,420	157,548	378.1
MINT	PSI 2.5	25	13	0.3	515	477	17.6	3,088	13,711	27.4
NCI PW	BioPAX	8,424	15,292	5.6	12,464	34,628	8.6	--	--	--
Reactome	MI-TAB	NA	NA	NA	NA	NA	NA	3,471	68,560	29.3
Reactome	TAB	NA	NA	NA	NA	NA	NA	3,394	61,541	97.4

**Note:** 'NA' indicates that a file was not available from a given source corresponding to the required file size. NCI PW Large failed to load, and thus a node and edge count could not be determined.

**Table 4.3 - Organism and dataset description of files used for load testing.**

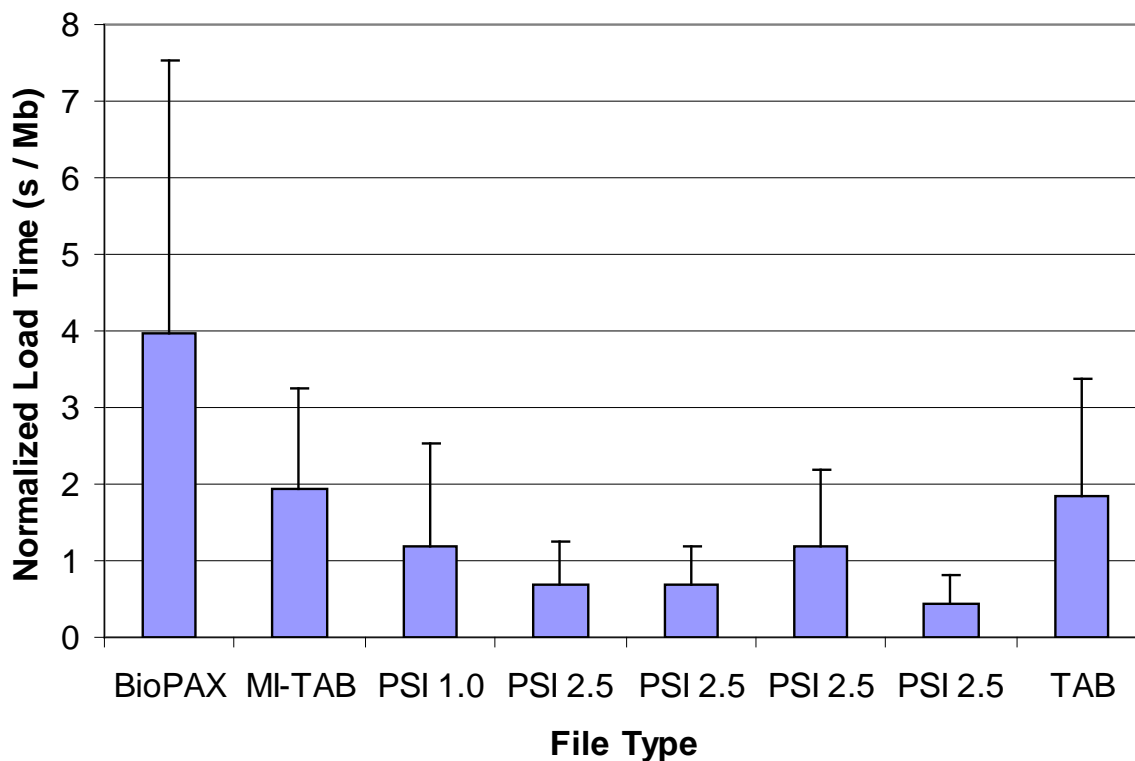
		Small	Medium	Large
Database	File Type	Org/Dataset	Org/Dataset	Org/Dataset
BioGrid	PSI 1.0	Bacillus_subtilis_168-2.0.55	Drosophila_melanogaster-2.0.55	ALL-SINGLEFILE-2.0.55
BioGrid	PSI 2.5	Bacillus_subtilis_168-2.0.55	Schizosaccharomyces_pombe-2.0.55	Saccharomyces_cerevisiae-2.0.55
BioGrid	TAB	Bacillus_subtilis_168-2.0.55	Homo_sapiens-2.0.55	ALL-SINGLEFILE-2.0.55
DIP	MI-TAB	Hpylo20081014CR	Ecoli20081014	Dmela20081014
DIP	PSI 2.5	Hpylo20081014CR	Scere20081014CR	Dmela20081014
IntACT	PSI 1.0	pyrho_small	human_small-32_negative	yeast_small-02
IntACT	PSI 2.5	kunvi_small	arath_small-03	yeast_small-02
IntACT	PSIMI Tab			IntAct
MINT	PSI 2.5	Saccharomyces cerevisiae-45	Saccharomyces cerevisiae-11	full-44
NCI PW	BioPAX	BioCarta	Reactome	
Reactome	MI-TAB			homo_sapiens
Reactome	TAB			homo_sapiens

For small to medium files, ranging from .001MB to 40MB in size, loading took an average of 1.5 to 47.5 seconds. BioPax, with a more detailed and nested XML structure, generally took longer than simpler file types. Overall, these files represent the typical size that an average user would tend to load. However, to demonstrate NAViGaTOR's scalability, much larger files were also tested. On large files, NAViGaTOR took between  $23.5 \pm 23.1$  and  $559 \pm 529$  seconds to load. For the large BioGrid PSI1.0 file (282MB), only the Vista64 machine was able to load the file, while the large NCI PW BioPax file (10.65MB) could not be loaded on any machine due to a deeply nested XML structure.

Trends were also observed for different file types. After removing test sets with missing data for the medium file size, and ranking the tests based on the medium file size loading time, 32-bit machines always performed the slowest, with the exception of 32-bit OSX 'Leopard'. In most tests, OSX 'Tiger' and RedHat x86 had the longest load times.

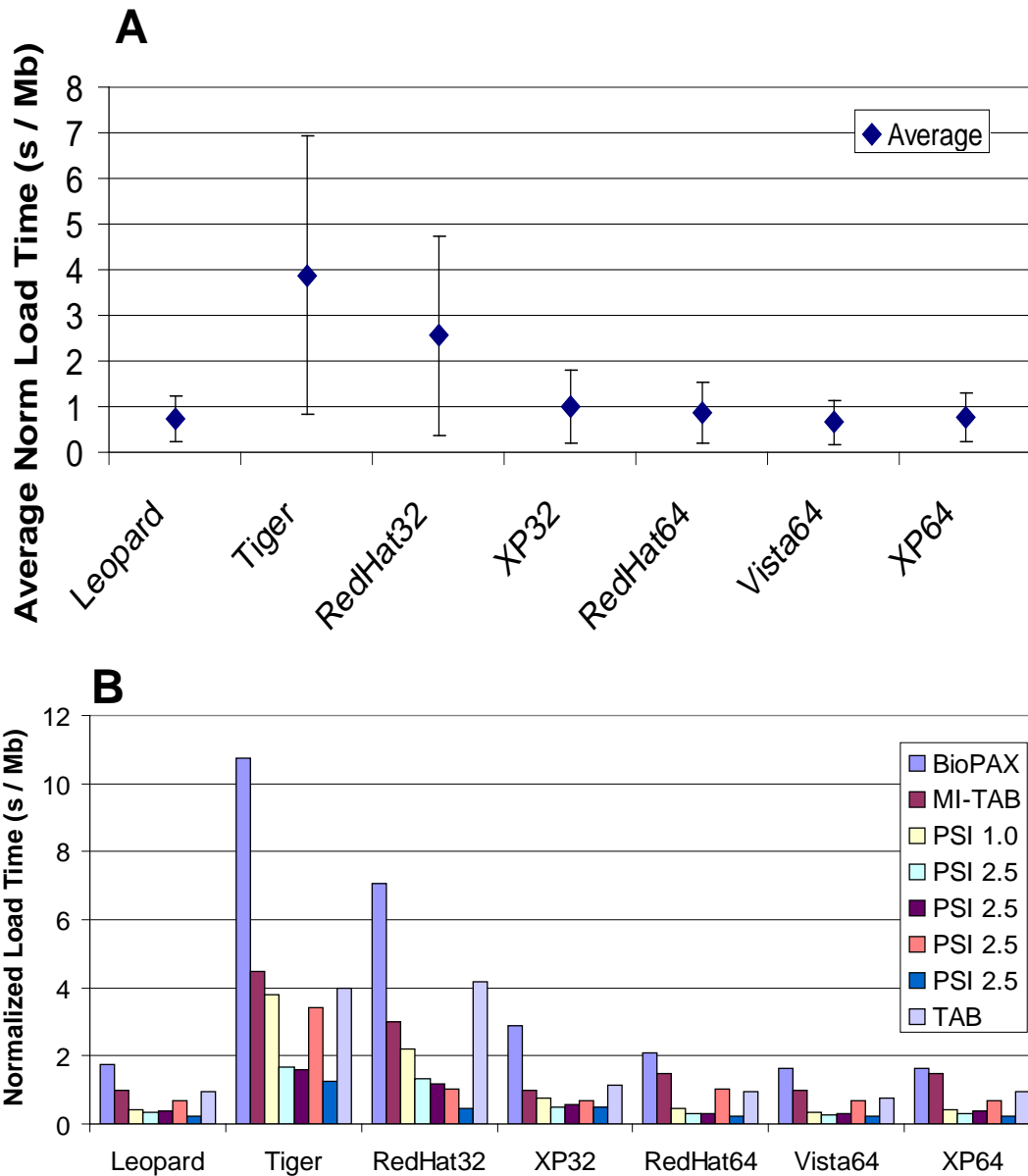
In order to assess whether different file types load in different amounts of time, the loading time had to be normalized against the file size (see Figure 4.1). On a per megabyte basis, BioPax files load the slowest, while the newer PSI 2.5 files load the fastest.

Similarly, we wanted to measure the relationship between machine architectures and load times for different files (see Figure 4.2). At first glance, it would appear that OSX 'Tiger' and Redhat x86 had significantly slower performance. However, these test machines only had 512MB of physical memory, as compared to 4GB or more for each of the other test machines. The 32-bit OSX 'Leopard' and XP32 performed similarly to each of the 64-bit test machines.



**Figure 4.1 - Load time for various file types.** In order to compare the loading time across files of varying size, the time was normalized by the size of the file. This shows that, on a per megabyte basis, the PSI2.5 files are actually more efficient to load. However, there is a significant memory overhead to loading these larger files, as noted below.





**Figure 4.2 - Loading time by system architecture.**

File loading performance was assessed for each system architecture in our test environment and for each medium sized file type. This is plotted as the mean normalized loading time (A), and as the individual normalized loading time for each file type (B). While OSX 'Tiger' and RedHat x86 appear to perform the worst, this is likely due to the fact that these test machines only had 512MB of physical memory, and thus memory swapping overhead was a significant factor.

**Table 4.4 - File loading times for small, medium and large files.**

Database	File Type	Small	Medium	Large
BioGrid	PSI 1.0	1.5 (0.7)	47.5 (52.4)	379 ( )
BioGrid	PSI 2.5	2.4 (1.8)	15.2 (13.0)	559 (529)
BioGrid	TAB	3.4 (3.4)	9.7 (8.0)	106 (207)
DIP	MI-TAB	2.2 (1.7)	3.8 (2.6)	62.1 (145)
DIP	PSI 2.5	2.1 (1.4)	8.5 (6.3)	29.4 (29.3)
IntACT	PSI 1.0	2.2 (1.3)	1.8 (1.5)	23.5 (23.1)
IntACT	PSI 2.5	2.1 (1.3)	3.4 (2.9)	32.4 (34.9)
IntACT	PSIMI Tab			201 (309)
MINT	PSI 2.5	2.8 (1.5)	7.8 (6.5)	73.4 (172)
NCI PW	BioPAX	20.4 (17.0)	34.2 (30.8)	StkOFlow
Reactome	MI-TAB			39.5 (48.8)
Reactome	TAB			150 (234)

Notes: All times shown are (mean  $\pm$  s.d.). BioGrid PSI1.0 only completed on Vista64; BioGrid PSI2.5 only loaded on Leopard, Vista64, and XP32. NCI PW BioPax Large failed to load on any test machine due to stack overflow errors caused by deeply nested XML.

## S5 Memory Analysis

### S5.1 Scalability Test Environment

These tests were performed on a desktop PC with an AMD Athlon 64 Processor 3400+ (2400 MHz, 1 core) with 2,046 MB of available physical memory and running the Windows Vista 64 operating system (version 6.0.6000 build 6000). The tests were performed in conjunction with YourKit Java Profiler version 7.0.11 (<http://www.yourkit.com/>). Although run on a 64-bit machine, all tests were conducted using the 32-bit Java Runtime Environment (JRE) version 1.6.0\_04. The tests were performed with the following applications:

- NAViGaTOR version 2.0.8 (<http://ophid.utoronto.ca/navigator>)
- Cytoscape version 2.6.0 (<http://www.cytoscape.org>)
- VisANT version 3.29 (<http://VisANT.bu.edu/>)

The sample networks loaded into each application were fully connected synthetic graphs (i.e., each node was connected to every other node in the graph) with 100, 200, 300, 400, 500 and 600 nodes. The graphs were labeled k100 to k600 respectively. These networks are unlikely to model any real-life biological system. Instead they serve as benchmarks for scalability testing purposes.

**Table 5.1 - Graph Characteristics for the Memory Footprint Comparison**

<b>Graph</b>	<b>nodes</b>	<b>edges</b>
k100	100	4950
k200	200	19900
k300	300	44850
k400	400	79800
k500	500	124750
k600	600	179700

## ***S5.2 Critical Memory***

These tests were designed to measure the minimum memory required to load a network file. This is the critical amount of memory, without which the application fails to load a given network. An application with lower memory consumption can load larger networks than one with higher memory requirements.

### **S5.2.1 Procedure**

For each test, the minimum and maximum memory allocated to the JRE was initially set to the same starting values. Each application was given an excess amount of memory, and an attempt was then made to load a network. If the network loaded successfully, the application was restarted with a lower memory allocation and tested again. If the network failed to load (resulting in error messages, or an application crash), the application was restarted with a higher memory allocation and tested again. These operations were performed iteratively until a memory allocation was obtained such that using any value below that resulted in failure to load the network. This procedure was carried out for each graph for each graph visualization program.

### S5.2.2 Results

VisANT consistently had the lowest minimal memory requirements, followed by NAViGaTOR (see Table 5.2 and Supplementary Figure 5.1). Both of these applications had considerable savings in memory consumption compared to Cytoscape, especially for larger networks. Cytoscape required about twice as much memory as NAViGaTOR for larger networks.

### S5.3 Memory Consumption Following Network Loading

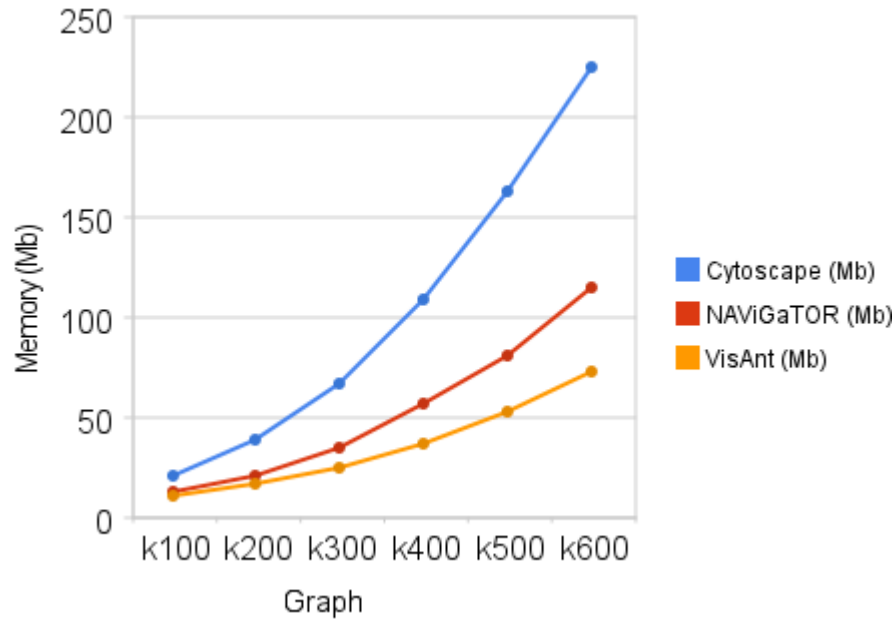
These tests measured the memory consumed by each application after a network had been loaded. Lower memory consumption will allow smoother work flow and interactive operation when working with network objects. If available memory is limited, then more CPU time is spent swapping memory locations and clearing unused memory slots. This results in lag when working with the application.

#### S5.3.1 Procedure

For these tests, the minimum and maximum memory heap sizes available to each application were set to 500 MB. To run the tests, each application was started with modified arguments so that they could be profiled using YourKit. Once started, a network was loaded into each application. After each application

Table 5.2 - Minimal memory required for loading a network.

Graph	NAViGaTOR (MB)	Cytoscape (MB)	VisANT (MB)
k100	13	21	11
k200	21	39	17
k300	35	67	25
k400	57	109	37
k500	81	163	53
k600	115	225	73



**Figure 5.1 - Minimal memory required for loading a network.**

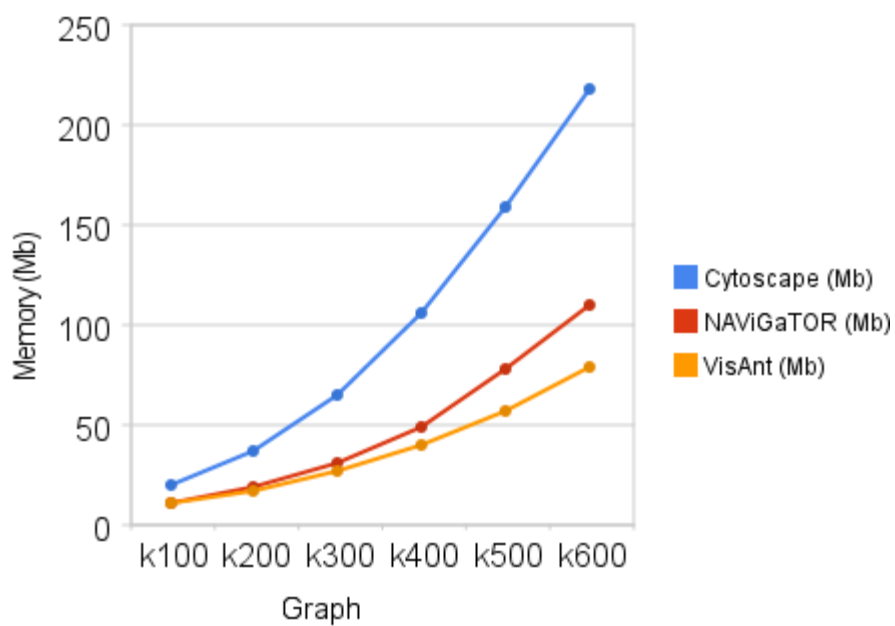
had fully loaded and displayed the network, the 'Force Garbage Collection' command was used in YourKit to force the Java garbage collection routine to run. This ensured that any excess unused memory allocations were removed. The memory usage at this point was recorded, and is shown in Table 5.3. Each application was restarted before loading in a new network.

### **S5.3.2 Results**

NAViGaTOR and VisANT's run-time memory requirements were similar to each other (Supplementary Table 5.3 and Supplementary Figure 5.2), with VisANT performing better, especially on larger networks. Cytoscape, on average, required about twice as much memory as NAViGaTOR.

**Table 5.3 – Run-time memory requirement.**

<b>Graph</b>	<b>NAViGaTOR (MB)</b>	<b>Cytoscape (MB)</b>	<b>VisANT (MB)</b>
k100	11	20	11
k200	19	37	17
k300	31	65	27
k400	49	106	40
k500	78	159	57
k600	110	218	79



**Figure 5.2 – Run-time memory requirement.**

## S6 User Interface Features

As the number of options and functions in a graphical user interface grows, the user is often confronted with increasingly packed panels, long and deep menus, as well as a growing number of hotkeys. Panels consume screen space, panels and pull-down menus both require the user to frequently travel between controls and the data they want to view, and hotkeys need to be memorized before they can be used. To help alleviate these problems, NAViGaTOR's user interface incorporates novel semi-transparent popup widgets that appear over the data, in context. These widgets consume screen space only when needed, reduce occlusion of data by virtue of their transparency, eliminate the need to travel between data and controls, and encourage a faster, more gestural type of interaction. These widgets are described briefly below; more details are given in (McGuffin and Jurisica, 2009).

The user may select nodes by dragging out a rectangle or lasso. To avoid having separate modes for these two functions, we reused a previous technique (Saund, et al., 2003) where the ink stroke dragged out by the cursor is dynamically interpreted to either mean a rectangle or a lasso, depending on its shape. As the user starts to drag, initially the ink trail is interpreted as a rectangle. However, if the user starts to drag back toward their starting position, it is interpreted as a lasso (Figure 6.1).

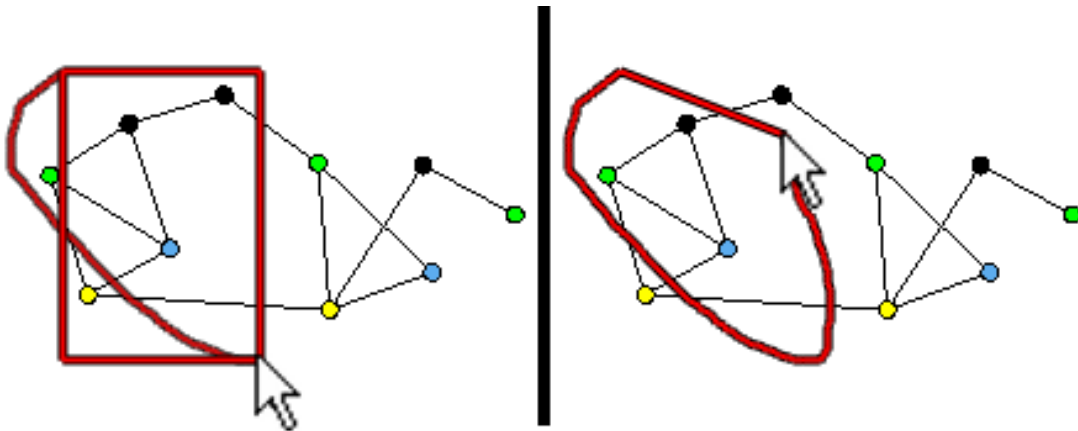
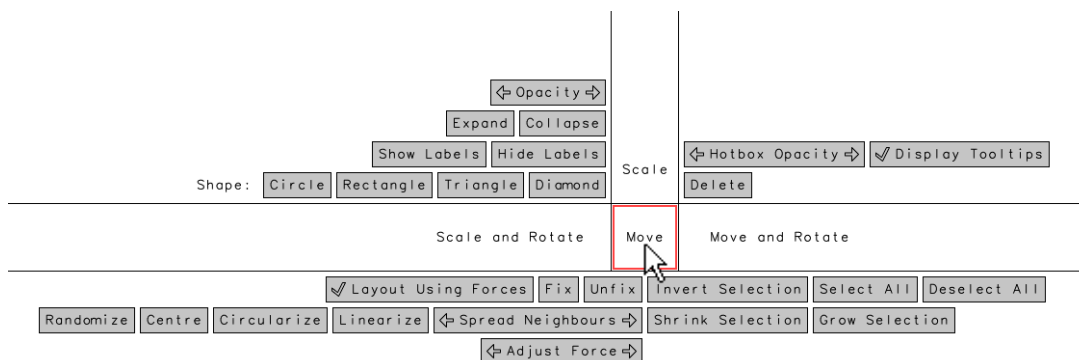


Figure 6.1 - NAViGaTOR selection tool.

Once a subgraph has been selected, the user may invoke operations on the subgraph by popping up a semi-transparent 2D menu of interactive controls called the "hotbox" (Figure 6.2). The hotbox contains pushbuttons, checkbox items, and specialized regions analogous to 1D or 2D sliders. Clicking down on these sliders and then dragging allows the user to fluidly specify arguments at the same time that they invoke a command. For example, clicking down on the "Move" 2D slider (in the center of the hotbox), and then dragging, causes the selected subgraph to be moved by the distance dragged out. During such dragging, the hotbox is faded out so the user can see the network responding to the drag (Figure 6.3). Once the user has completed the drag, the hotbox fades back in, and the user may invoke additional subsequent hotbox commands before dismissing the hotbox. There are 1D and 2D sliders in the hotbox for performing rotations, scaling, changes of opacity, and changes to node-weights used in force-directed layout.



**Figure 6.2 - 'Hotbox' popup menu in NAViGaTOR.**



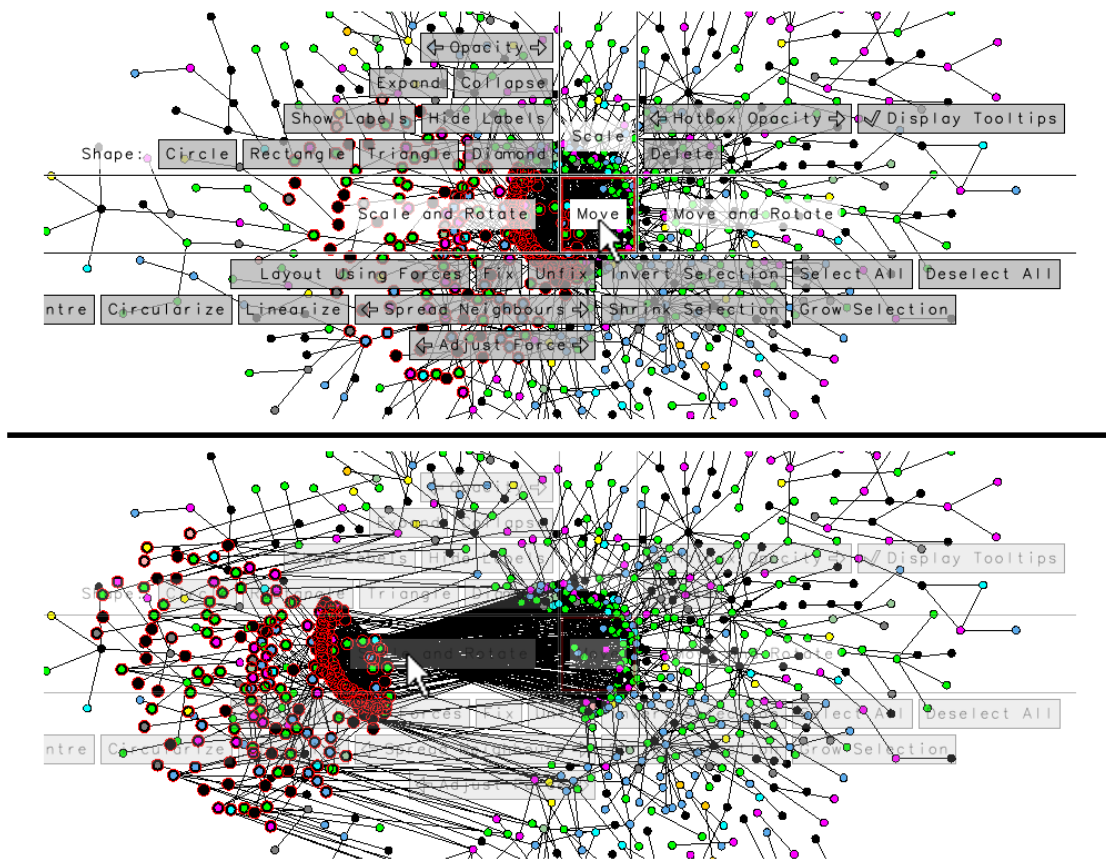


Figure 6.3 - Sample translation using the 'hotbox' tool.

The user may also change the layout of nodes with commands in the hotbox that "Linearize" or "Circularize" the positions of selected nodes, projecting them onto straight lines or circles that can then be moved, rotated or scaled. Finally, various pushbuttons and checkbox items in the hotbox allow the user to change the shapes of nodes, reveal or hide labels, collapse or expand meta-nodes, and activate or deactivate force-directed layout of nodes. Future versions of this interface could even nest popup menus within the hotbox, enabling it to scale up to hundreds of functions accessible via a single unifying interaction technique.

Figure 6.4 shows a sample layout created with NAViGaTOR's interface. The user has identified 6 hub nodes and arranged them in a hexagon, and made them appear as large diamonds. Nodes interacting with the same pairs or triples of hubs are collected into linear arrangements on the right, allowing their labels and edges to be read easily. The set of currently selected nodes (highlighted in red, on the left) is all nodes at least 6 edges away from any of the 6 hubs (this set was found with the interface simply by selecting all nodes in the graph and then deselecting the neighborhoods of radius 5 centered on each

hub). Concentric circles of nodes were arranged around the selected set with a single drag.

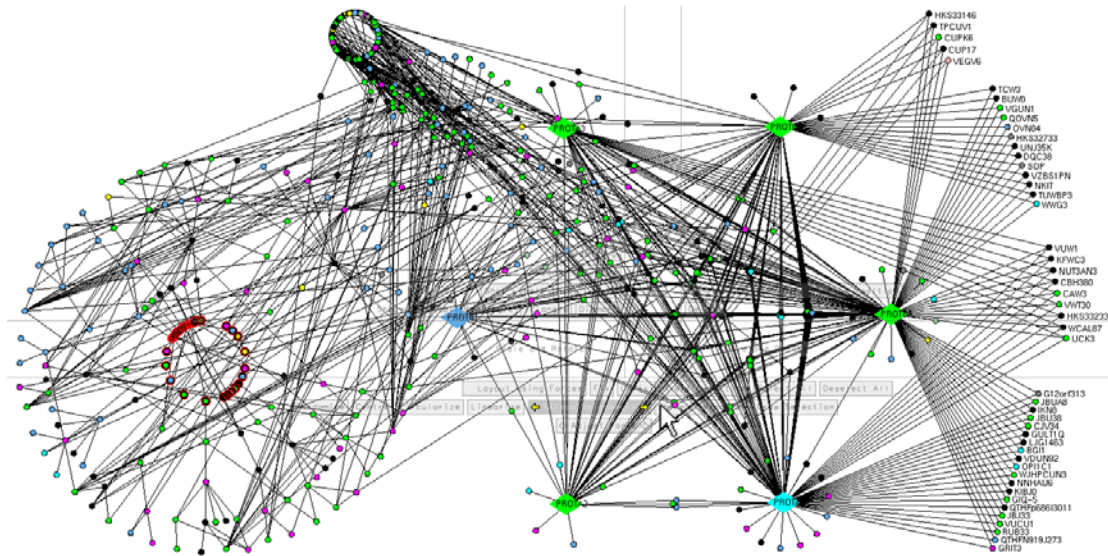


Figure 6.4 - Network layouts accelerated using the 'hotbox' tool.

## S7 References

- Agarwal, R., et al. (2009) The Emerging Role of the RAB25 Small GTPase in Cancer, *Traffic*, **In press**.
- Barrios-Rodiles, M., et al. (2005) High-throughput mapping of a dynamic signaling network in mammalian cells, *Science*, **307**, 1621-1625.
- Brierley, M.M., et al. (2006) Identification of GAS-dependent interferon-sensitive target genes whose transcription is STAT2-dependent but ISGF3-independent, *Febs J*, **273**, 1569-1581.
- Brown, K.R. and Jurisica, I. (2005) Online predicted human interaction database, *Bioinformatics*, **21**, 2076-2082.
- Brown, K.R. and Jurisica, I. (2007) Unequal evolutionary conservation of human protein interactions in interologous networks, *Genome Biol*, **8**, R95.
- Cox, B., et al. (2007) Integrated proteomic and transcriptomic profiling of mouse lung development and Nmyc target genes, *Mol Syst Biol*, **3**, 109.
- Cox, B., et al. (2009) Comparative systems biology of human and mouse as a tool to guide the modeling of human placental pathology, *Mol Syst Biol*, **5**, 279.
- Dong, J., et al. (2009) Lung cancer: developmental networks gone awry?, *Cancer Biol Ther*, **8**, 312-318.
- Gortzak-Uzan, L., et al. (2008) A proteome resource of ovarian cancer ascites: integrated proteomic and bioinformatic analyses to identify putative biomarkers, *J Proteome Res*, **7**, 339-351.
- Hermjakob, H., et al. (2004) IntAct: an open source molecular interaction database, *Nucleic Acids Res*, **32**, D452-455.
- Joshi-Tope, G., et al. (2005) Reactome: a knowledgebase of biological pathways, *Nucleic Acids Res*, **33**, D428-432.
- Jurisica, I. and Wigle, D.A. (2006) *Knowledge Discovery in Proteomics*. Mathematical Biology and Medicine Series, **8**. Chapman and Hall/CRC Press.
- Jurisica, I., et al. (2007) *Cancer informatics in the post genomic era. Toward information-based medicine*. Cancer Treat Res, **137**. Springer Verlag.
- Kislinger, T. and Jurisica, I. (2006) Proteomics and Bioinformatics in Biomedical Research, *Cancer Genomics-Proteomics*, **3**, 11.
- Lau, S.K., et al. (2007) Three-gene prognostic classifier for early-stage non small-cell lung cancer, *J Clin Oncol*, **25**, 5562-5569.
- McGuffin, M. and Jurisica, I. (2009) Interaction techniques for selecting and manipulating subgraphs in network visualizations, *IEEE Trans. Visual Comput. Graphics*, **In Press**.
- Mills, G.B., et al. (2009) Genomic amplicons target vesicle recycling in breast cancer, *J Clin Invest*, **119**, 2123-2127.
- Motamed-Khorasani, A., et al. (2007) Differentially androgen-modulated genes in ovarian epithelial cells from BRCA mutation carriers and control patients predict ovarian cancer survival and disease progression, *Oncogene*, **26**, 198-214.
- Saund, E., et al. (2003) Perceptually-supported image editing of text and graphics. ACM New York, NY, USA, 183-192.

- Savas, S., et al. (2009) A comprehensive catalogue of functional genetic variations in the EGFR pathway: protein-protein interaction analysis reveals novel genes and polymorphisms important for cancer research, *Int J Cancer*, **125**, 1257-1265.
- Seiden-Long, I.M., et al. (2006) Transcriptional targets of hepatocyte growth factor signaling and Ki-ras oncogene activation in colorectal cancer, *Oncogene*, **25**, 91-102.
- Sodek, K.L., et al. (2008) Identification of pathways associated with invasive behavior by ovarian cancer cells using multidimensional protein identification technology (MudPIT), *Mol Biosyst*, **4**, 762-773.
- Stark, C., et al. (2006) BioGRID: a general repository for interaction datasets, *Nucleic Acids Res*, **34**, D535-539.
- Tomasini, R., et al. (2008) TAp73 knockout shows genomic instability with infertility and tumor suppressor functions, *Genes Dev*, **22**, 2677-2691.
- Wigle, D. and Jurisica, I. (2007) Cancer as a system failure, *Cancer Inform*, **5**, 10-18.
- Wu, C., et al. (2007) Systematic identification of SH3 domain-mediated human protein-protein interactions by peptide array target screening, *Proteomics*, **7**, 1775-1785.
- Xenarios, I., et al. (2000) DIP: the database of interacting proteins, *Nucleic Acids Res*, **28**, 289-291.
- Zanzoni, A., et al. (2002) MINT: a Molecular INTERaction database, *FEBS Lett*, **513**, 135-140.
- Zhu, C.Q., et al. (2009) Understanding Prognostic Gene Expression Signatures in Lung Cancer, *Clin Lung Cancer*, **10**, In press.