Visualization and Exploration of Complex Networks

Erik van den Bergh

April 6, 2010

Abstract

As the amount of available data increases in biology, there is a big need for tools that are able to visualize this data. Visualization is a good way to quickly and intuitively present a lot of data in a short time. In this thesis, I describe the basics of data visualization as well as the current state of the art, which mainly focuses on static presentation of data. I then describe a tool, called vinet, that was designed to be used as a tool for dynamically exploring data. It focuses on performance and simplicity, while remaining lightweight and flexible. I also present an example use of this tool and the information that can be extracted with it, using a human protein protein interaction database.

Contents

Chapter 1

Introduction

Since the 1990's, the amount of data that has been available to biological research has grown with an enormous rate(Fig. 1.1) The amount of sequences, expression and protein interaction data has never been so large and of such high quality. But while the amount of data available is increasing, tools that are able to analyse this data are lacking.

To cope with this huge amount of data a lot of methods and tools have been developed. Unfortunately, biological data is extremely diverse and it is often that a tool is developed for one dataset specifically and then discarded again when the project is finished. Many of these tools feature a visualization of some sort, because it is an intuitive way to show data.

In this thesis, I first discuss a number of techniques and tools that have been developed to deal with visualizating the data explosion. Then I will describe a new tool that is able to visualize large datasets that aims at maintaining visual clarity and performance, called vinet. Finally I will give an outlook on the future of the project and of visualization tools in general.

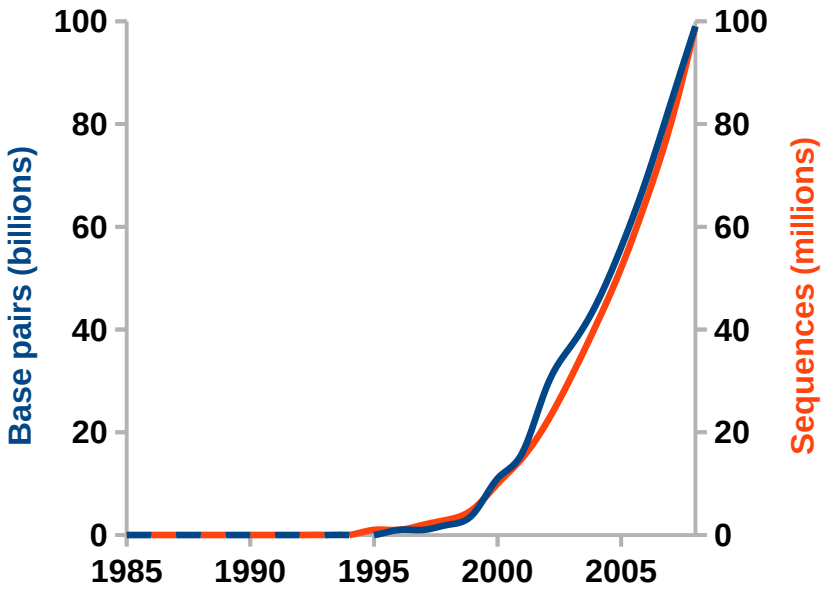


Figure 1.1: Number of base pairs and sequences in the GenBank database [1]. The 'data explosion' from the 1990's and onwards is clearly visible.

Chapter 2

Data visualization

2.1 Definition

Data visualization is displaying data in a schematic manner that makes the information easier to interpret. In order to achieve this there are a number of things that a visualization must do: (adapted from [2])

- Show the data
- Avoid distorting the data
- Present as much of the data as possible in the available space
- Make large datasets coherent
- Make it easy to compare different sets of data
- Reveal the data at several levels of detail

Of course, when we sort data by multiple criteria, similar datapoints will group together automatically. This is what we see in a scatterplot for example, where the datapoints are sorted according to two criteria over the x and y axes in order to achieve groups of datapoints. If there is a grouping of datapoints, we can say that these points correlate to each other in some way. It is of course possible that a 2D plot will not produce any useful grouping of data. Perhaps the wrong units are chosen along the axes or maybe we need a third unit to display. We could show this in a 3D plot. But, what happens when we still see no grouping? We need more criteria, but we have run out of dimensions to display them in. Thus we need different techniques to display the data.

2.2 Biological data

In biology and bioinformatics, we see that most data is about relations. Relations between genes, organisms, proteins and molecules are all part of biological

data. A lot of these relationships together form a network that represents all interactions for that specific organism, pathway or dataset. The most simple and convenient way to display a network is using a mathematical graph. A graph consists of a number of nodes displayed as circles and edges (or vertices) displayed as lines that connect the circles. An edge can have a weight, which represents a property that is part of the relationship that is symbolized by the edge. In biological data, edges usually have multiple weights.

2.3 Visualization techniques

According to D.Merico et al. [3], there are two important factors in the visualization of biological data: Layout and visual features. I will discuss both in this section.

Spatial layout and clustering

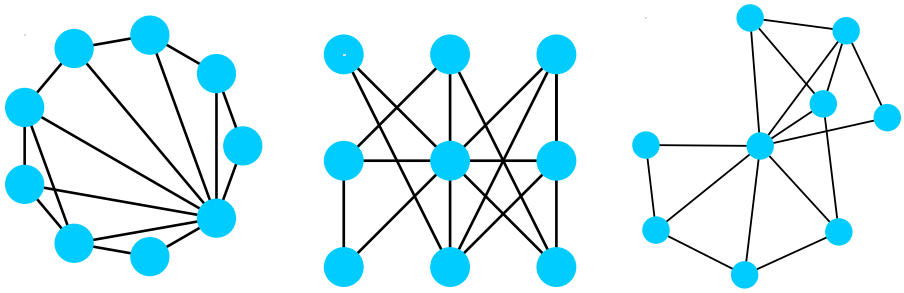
Laying out data points is essential for interpreting a visualization fast. In the previous section I discussed spatial layout in a scatter plot, but it is just as important in a graph. Laying out a graph properly has been a long discussed problem [4] [5] [6] and it is a problem that is very suited for automation . In general, we can state that a functional graph layout must have the following properties

- All relevant nodes must be visible
- Edges should cross each other as little as possible
- The layout must use as little space as possible, without losing information

Of course, this is a very basic list when looking at the complexity of biological data and we will expand it further down.

Geometric layouts

An intuitive and easy way to layout nodes is using geometric layouts. These layouts use basic geometric shapes such as circles and squares (and their 3D counterparts; spheres and cubes) as a template to place the nodes on. Circular layouts (see fig. 2.1(a)) are commonly used in biological data because they are useful when showing networks that have relatively few nodes and many connections. Square or grid layouts (see fig. 2.1(b)) are useful when the density of nodes and edges is variable throughout the network, because they provide an even spacing and thus a clean overview. This also means that the information they present uses a lot more space than other types of layouts, which is why grid layouts are unsuited for larger networks. A third option is simply spreading all nodes randomly across even distances. This is called an 'organic' layout (see fig. 2.1(c)).



(a) circular layout

(b) grid layout

(c) 'organic' layout (nodes randomly spread at similar distances)

Figure 2.1: Three methods of laying out the same network.

Dynamic layouts

A different approach for laying out nodes is dynamiccaly, through layout algorithms. The first algorithm for laying out graphs that has been published is Eades' spring graph algorithm [4]. This algorithm follows the analogy of a set of balls connected by mechanical springs; the balls are the nodes, the springs are the edges. In a physical sense, if the network is entangled and it is then released, the springs will automatically return to an optimal configuration where the spring lengths are as short as possible. Networks layed out in this manner are visually pleasing because they divide all nodes evenly over the surface. This algorithm was later expanded by Fruchtermann and Reingold [6], introduced the notion of the total energy present in the system. This is useful because Eades' model keeps vibrating indefinitely even with a damping force (the air friction in our analogy). With the total energy measured, we can stop the algorithm when a desirable (low) amount of energy is reached.

A different and much used variation is the algorithm of Kamada and Kawai [5]. They stated that the distance between nodes should not be evenly spread, but that it should resemble the graph theoretic distance (the shortest path between two nodes) so that nodes that are highly connected are closer together. This is already a basic form of grouping nodes according to their properties.

Biologically oriented layouts

The layouts discussed so far are all based on an entirely mathematical vision of graphs. When we look at biological networks, we want to visualize more things. We can state some additional properties for laying out a biological network:

- Nodes should be sorted spatially according to different properties
- The properties of nodes should be visible, preferably in a non-text manner

The second property is discussed extensively in the next section. It discusses different colors, shapes and other means to display as much information as

possible while maintaining clarity. The first property, sorting nodes spatially, can be very straightforward such as sorting by time or by a certain factor. One factor that is useful is a mathematical concept called betweenness, which represents the amount of shortest paths from all nodes to all nodes that pass through a node. So, the higher this value, the more 'connected' a node is. In biology, nodes that have a high betweenness are often important because they play a role in many processes or are a part of many pathways. In a spatial layout, these nodes are usually placed in the center of the network.

But often in biological networks, the properties of the nodes are not easily scaled on an axis. It is often the case that multiple properties are interesting as well the relationships between these properties. In this case, clustering is very useful for these networks. Clustering means dividing data that shares certain properties into subsets. The advantage of clustering methods is that it is able to reduce the complexity of the data while still taking all relevant properties into account. In the chapter describing the vinet tool, I will give a detailed explanation of what algorithm we used in our tool and why.

Visual features

Visual features are very important for data visualization. Examples of visual features are colors, shapes, symbols, etc. Visual features are very useful in visualization because humans are able to interpret them faster than textual data. When using visual features, it is important to have a clear definition of what they stand for, which can be done by means of a legend or they can be based on already established labeling schemes, such as the colors for basic elements in atomic models (hydrogen is white, oxygen is red, carbon is black and so on). The problem with visualizing biological data is that there is no such scheme and that every visualization program defines their own. Standard visual features in networks are edge thickness and color, and node shape, color and size. When used properly in combination with layouts, a lot of information can be shown already without using any text.

2D vs 3D visualization

There has been considerable study regarding the efficiency of 2D and 3D displays [7]. With the advantage of current technology the capabilities for 3D visualization are becoming bigger and bigger. However, in most cases a 3D visualization is limited to a 2D display (a computer monitor). This leads to a simulated third dimension or '2.5D', which affects the accuracy of the visualization. It is hard for the human eye to correctly interpret the 3D information without visual cues such as shadows or rotation of the perspective. Wickens et al. [8] provide us with the Proximity Compatibility Principle (PCP) regarding this matter. The PCP states that tasks that require accurate or quick response are better performed in 2D whereas tasks that require integration of several dimensions are better performed in 3D. In the study by Tory et al. [7], it is shown that answering questions using several dimensions of information is done

considerably better in 3D. In developing vinet, we wanted to create a tool that allows a researcher to get a quick and clear overview of many types of data at the same time. This is why we have chosen for a 3D tool. This allows the researcher to adress many questions at the same time with a single look at the screen and minimal input.

2.4 Current tools

In this section I will describe currently popular tools for displaying and analyzing networks.

Arena3D

Arena3D [9] uses a novel concept that layers multiple 2D visualizations as layers in 3D. The data can be grouped onto different layers and layed out with different layout algorithms. There are also an extensive clustering options, which can be applied per-layer. It uses its own input format, which is a flat text file in a specific syntax, but it can export networks to other formats. Arena3D uses Java3D as its rendering engine, which means that Arena3D's running speed is limited to the capabilities of the Java Virtual Machine, but provides cross-platform compatibility.

Cytoscape

Cytoscape [10] is currently the most popular network visualization tool in bioinformatics research. It is widely used for analysis and display of networks found in research data. Cytoscape is an open source Java application, running on all JRE-compatible OSes. Its popularity is partly due to its flexible plugin system, which allows users to create their own tools with minimal coding skills. This has allowed many research groups to create their own plugin for their specific data sets, which has led to a very big number of cytoscape users. It is suited for networks up to thousands of nodes and it is as fast as the Java Virtual Machine is on the used platform. It is only suited for 2D visualization but there are developments towards 3D.

BioLayout Express 3D

Biolayout Express 3D [11] is a Java application for 3D visualization of networks. It uses Java's OpenGL library for rendering and it has many optimizations for multi-core processors. It features a Markov Clustering algorithm and it is capable of handling networks with thousands of nodes. It is still relatively new and there is a lack of data that is available for it. Its performance depends largely on the OpenGL capabilities of the machine it is used on, making it very suited for use on visualization workstations.

Osprey

Whereas the previously described programs are more geared towards network analysis, Osprey [12] is a tool that also has powerful network editing capabilities. It is a stand-alone program that is free to use for academic and not-for-profit organizations and it runs on the three major OSes. There is also in-program access to Genome Ontology data that can be compared and layed out as the user wishes. It has a number of standard network analysis features and also a lot of different options for adding, removing and editing edges and node properties. It is capable of handling networks up to hundreds of nodes.

2.5 File format standards

In this section I will describe four popular file formats that are being used in network analysis.

PSI-MI

The PSI-MI format [13] is a file format that describes protein-protein interactions, developed by the Protein Standards Initiative. It is an XML file that is structured as a tree with an entry-set as its root. The set contains entries, which in turn contain information on relations and properties. It is supported by Cytoscape and BioLayout Express 3D and a number of smaller programs.

BioPAX

BioPAX [14] is a file format developed by the BioPAX work group. It began as a format for metabolic interactions (level 1) but has now reached level 3 which describes metabolic pathways, molecular interactions, signaling pathways (including molecular states and generics), gene regulation and genetic interactions. It is based on the OWL standard, an XML based language for ontologies. It also forms the basis of the BioCyc collection of databases which contain several different types of pathways in a wide range of organisms.

SMBL

The Systems Biology Markup Language (SMBL) [15] is based on XML, like the previous two formats. It started at the California Institute of Technology in 2000, making it the oldest standard of the three described here (PSI-MI and BioPAX both started in 2002). It's development is tightly bound to the CellDesigner tool, which is a network editing tool developed specifically for SBML. It is strongly supported by the KEGG database and all its subsidaries, which offer most of their data available for download in this format.

Sparse network format

The sparse network format is a very basic format that is used to describe sparse networks. It consist of two or three columns, with every row being one connection. The first two columns list the two connected nodes, and the optional third lists the weight of this connnection. It is widely used as a preliminary format because it is the easiest form of output.

Chapter 3

vinet

In this chapter I will describe a newly developed program called vinet. I will describe why it was developed and the ideas that are behind it, as well as the basics of its implementation. Finally, I will give a short description of using the program to visualize networks.

3.1 Motivation

The vinet project was created as an alternative to current network visualization programs, such as the ones described in section 2.4. The Bioinformatics department at WUR, where this thesis was written, is currently in posession of a visualization wall. This wall consists of 12 27" LCD screens that are coupled to 3 workstations running on NVidia Quadro FX4700-2 graphics cards, forming one large tiled display. With this setup, current tools were performing bad with large networks (>100 nodes) because the resolution is far higher than the resolution the programs were created for. Therefore we needed a tool that had the flexibility and the options that these tools had, but with increased performance on high resolutions. It should also be a tool that can be used for getting information out of big datasets. The idea behind the visualisation wall is that a user can load his or her dataset into the program and stand in front of the wall to get an overview of their data. Interesting results can be spotted in a split second and they can be further examined using the vinet tool, or the user can load a small part of the dataset into their own favourite tool. In figure 3.1, we can see an example of a small network as rendered by vinet.

3.2 Design

vinet is designed with abstraction in separate layers in mind. Basically, there are 3 layers that the program is concerned with:

- A data layer

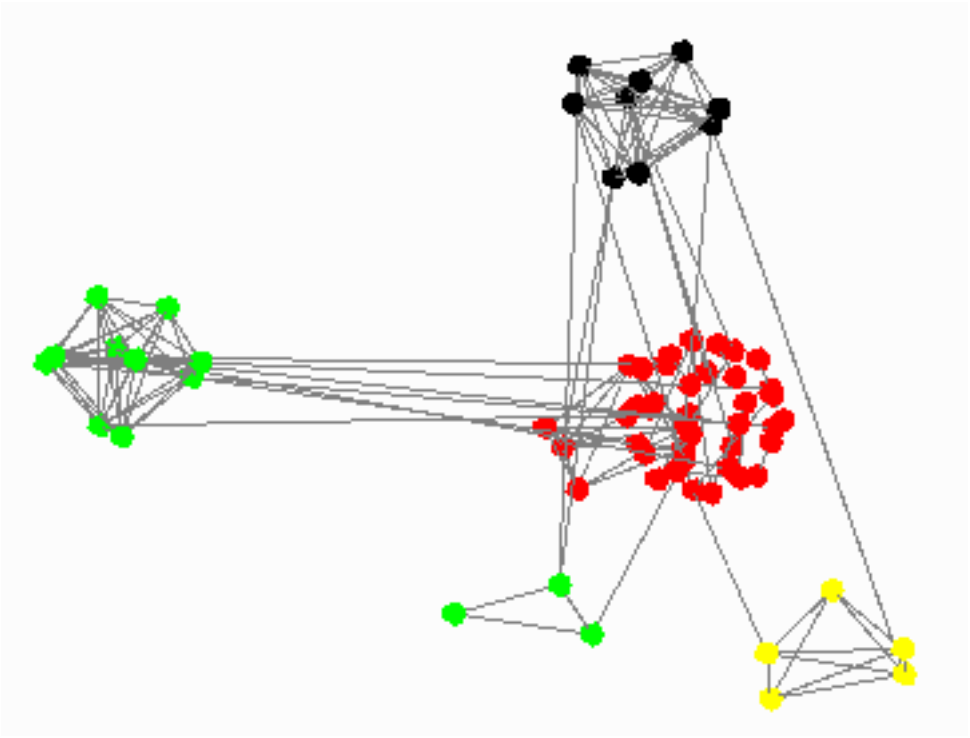


Figure 3.1: A small network rendered by vinet, consisting of 68 nodes.

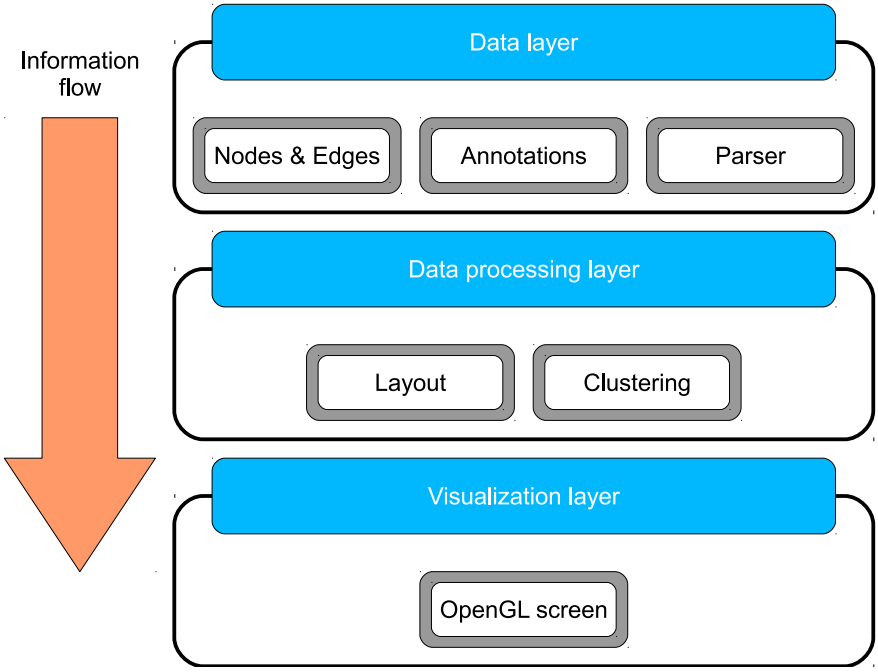


Figure 3.2: An overview of the layers that are part of the design of vinet. The layers are described in detail in their subsections.

- A data-processing layer
- A visualization layer

A detailed overview of these layer can be seen in figure 3.2.

vinet's design focuses on intuitive user interaction. To make the tool more suitable for working on a large tiled display, there is no menu present on the screen. The main menu is instead called with a simple right click anywhere on the screen. The reason for this is that when testing other applications on the tiled display, their menu structures were suddenly located very far from where the user is looking. They are usually located on the top left, which makes it unhandy for the user if he/she is looking at a specific spot in the bottom right. The ability to call up the menu where the user is looking is alot more convenient when navigating through a network. The menu also scales the font size according to the screen size, so that it remains visible regardless of the resolution that is used.

Data layer

The data layer represents the input that the user gives to the program. Practically it is the flat file that contains the network and the additional files that contain data about the nodes. The parsing methods for these files are also part of this layer. The input data is one file containing the network in a sparse format and an optional file containing the annotations of the nodes.

Data processing layer

The data processing or business layer represents the layout methods and the clustering algorithm which are both described below. It is the part that does most of the number-crunching and its details are mostly hidden from the user. The output of this layer is fed to the visualization layer.

Layout

Laying out nodes is split into two parts in vinet. One part is how the individual clusters should be layed out and the other is how the clusters should be layed out amongst each other. In vinet, the former is referred to as cluster layout and the latter as cluster spread. For cluster layout, there are 2 options: spherical and cubic. The cluster spread has only the cube option and a random spread; a spherical spreading of clusters was found to be impractical. There is also a totally random layout, which ignores clustering.

Additionally there is a 2D mode, which has the same options as 3D but ignores the third dimension. It is included because for static pictures of a network a 2D representation is often clearer.

There is also a possibility to 'explode' the cluster layout and spread. This means that the distances between the nodes is increased while the form of the layout is kept intact, resulting in a blown-up picture.

Clustering

As was explained previously, clustering is the grouping of data that is similar in one or multiple aspects. Clustering is an essential part of vinet, because it makes data (and the visualization thereof) much easier to interpret. Clustering was tricky to implement however, because most clustering algorithms require alot of specific parameters before any clustering can be done. Because vinet was meant as an initial way of viewing data, we assume that the user does not know what parameters need to be adjusted. We then chose to use a program called netclust [16] for clustering. This program is a tiny (76kb) program written in C that performs extremely fast clustering with a minimum of parameters. netclust uses the edge weights to generate clusters and it takes an optional cutoff value as a parameter. The weights can represent distance (ranging from 0 to 1) or similarity (ranging from 0 to N).

The speed of netclust makes it possible to cluster models on the fly, which allows the user to quickly browse through different clusterings of the same data. All clustering can be done with a minimum of commands in the main menu.

Visualization layer

The visualization layer is the user frontend of the program. It is responsible for everything that can be seen on the monitor. The layout has been described in a large part in the previous layer. Without getting into too much detail on OpenGL code, we can say that this is where the gpu does all its work and that in this layer the most performance gains were had. These gains are described in detail in the section on performance further on.

3.3 Implementation

This section gives a general overview of the vinet program. For the nitty-gritty details there is html documentation available that comes with the program.

The 3D visualization of vinet is implemented in the Open Graphics Library, commonly known as OpenGL. It is a graphics API that has implementations on different platforms such as Windows, Linux and MacOS. It uses a system of primitives called vertices to draw images in 2D and 3D. OpenGL has been widely used in scientific simulations and visualization and is considered a standard in the area of graphics libraries. One of the reasons of its popularity is its availability on multiple platforms and programming languages. Java, Fortran and Python all have implementations of OpenGL functions and it is the industry standard in game development. This is also a reason for us to use OpenGL: It is a widely supported cross-platform API that is well documented.

vinet is written in C++. This language was chosen because of several reasons. OpenGL is a language that is usually implemented in C. However, the C language has limited capabilities for abstraction and no object orientation, both of which are very useful in a complex project like this. C++ does have these capabilities and it is for a large part backwards compatible with C, which is why it was used for vinet.

Data Structures

vinet uses a number of datatypes, which are listed in table 3.1. These are all separate classes. As the data layer, there is a Parser class which reads the inputfiles and generates everything necessary for data-processing. The business layer has a Layout class and it also implements the netclust program for clustering. Finally, there is a Displayer class which contains the OpenGL code for the actual visualization.

Type name	Description
Network	The network to display
Node	Represents nodes in the network
Edge	Represents edges in the network
Cluster	Provides functions related to layout and clustering.

Table 3.1: Data types used in vinet

Performance

A considerable part of developing vinet went into ensuring good performance. As mentioned earlier, performance with large networks was not very good with existing tools. When used in the setup currently available at the WUR, this resulted in unworkable program loading and navigating speeds.

We have explored a big number of options for increasing performance. Some of those are very specific to 3D rendering such as backface culling and polygon smoothing and describing them here would be too detailed. One interesting speed up that is built in OpenGL are display lists. These are lists that contain calls to OpenGL stored in a format that is optimized for the rendering pipeline. Multiple display lists are used in vinet : One list draws one node, one uses this list to draw the entire network and one more list draws all the edge connecting the nodes. This method of drawing is much faster than drawing from the Network datastructure described above.

An important performance tradoff is the level of detail vs. drawing speed. Obviously, if the level of detail is higher, drawing speed also decreases. In games and visual simulations so called 'fog' can be used to obscure parts that are drawn farther away from the viewers perspective and thus increase drawing speed. However, when representing data we aim for completeness and accuracy which means fog is not an option. Therefore we needed to find a balance between the level of detail and maintaining workable performance. Because of the way 3D images are rendered (using straight lines), spheres are computationally very costly to draw. We chose to draw the nodes as low detail spheres, which means that it is visible that they are actually regular polygons when the user zooms in. This does not decrease the factual accuracy of the representation in any way and we feel that this level of detail is a good balance between aesthetic and performance.

Another advantage of the minimalist design of vinet is the small memory footprint of the program. It also means that the total binary size is no more than 190kb. This small size means that vinet is easy to install, and that it is capable of running on all types of hardware that has OpenGL capabilities. While vinet was designed and tested on Linux, it has been successfully compiled on Microsoft Windows, requiring no major adjustments.

The Network datastructure that stores all information about the network connects all Node datastructures togehter as well as the Edge datastructures.

Controls	
Left mouse click	Display information about the clicked node(s)
Left mouse drag	Rotate model (when in 3D mode)
Right mouse click	Layout menu
Middle mouse drag	2D transformation of model
Scroll up/down	Zoom in/out

Table 3.2: vinet in-program controls

From a software designing and maintaining point of view this is good design, but when the nodes are drawn, the only information that is needed are the coordinates of all polygons. That is why vinet stores the 'bare' network in a simple array of floating point numbers, increasing the OpenGL drawing speed.

3.4 Using vinet

The tool takes a minimum of 1 commandline argument and a maximum of 4. Syntax is as follows:

```
network -cf cluster_f [-af annotation_f] [-geom wxh] [-td]
```

As can be seen, the input file must be a space or tab delimited file. The program assumes by default that the input file is space delimited, but this can be changed to tab-delimited with the -td flag. Each line in the input file must have the following columns:

```
nodelabel1 nodelabel2 edgeweight
```

The edgeweight value can be an integer or a fraction. Additionally, an annotation file can be specified which contains additional information about the nodes. It must be in the following format

```
name info1_name info2_name info3_name ... //first line
nodelabel1 info1 info2 info3 ...   //all other lines
```

As we can see in table 3.2, vinet has all basic options for navigating through 3D space. When a node is left clicked, the user recieves information on that node in a small extra window. The level of detail of this information depends on the level of zoom. When the user is zoomed out, general information about the cluster that the node belongs to is given. When the user is zoomed in, he/she gets all annotations that belong to that node.

The layout menu is the main control menu available in vinet. It lists a number of options regarding the spread of clusters, the type of clustering layout,

the annotation to use when clustering, switching from 3D to 2D mode and vice versa and resetting the view for when a user is lost in navigation. See figure 3.3 for a detailed explanation.

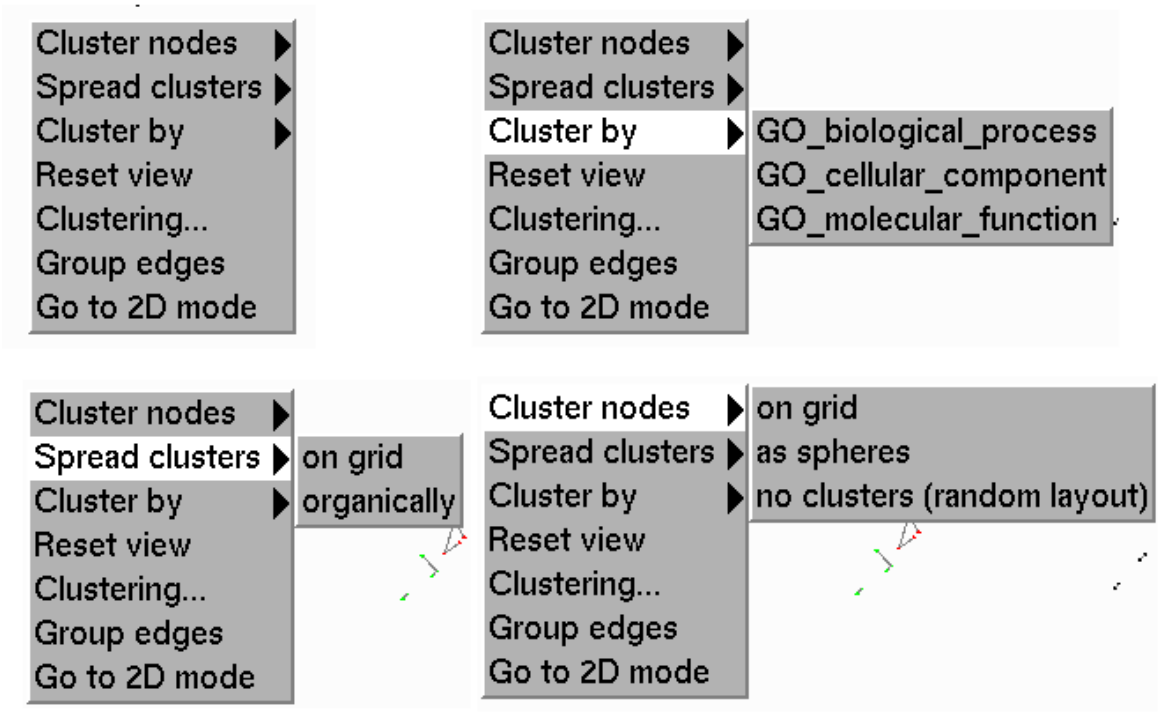


Figure 3.3: An overview of the menu options in vinet. **Top-left**: The main menu as it looks collapsed. The 'reset view' option restores the view to the initial setting, for when the user gets lost while navigating the model. The 'Clustering...' option provides a means to adjust the cutoff for the netclust program and reclusters the model. The group edges option reduces the number of between-cluster edges to 1 per cluster pair to reduce on screen clutter. See the next chapter for a detailed example of this (fig. 4.2). The bottom option allows switching between 2D and 3D mode. **Top-right**: The 'Cluster by' menu allows for clustering by the annotations that this network has. In this case, it shows the options for the database described in the next chapter and its GO term annotations. **Bottom-left**: The 'Spread clusters' menu gives the user the option to spread the clusters on a grid, or organically. **Bottom-right**: The 'Cluster nodes' menu gives options for the shapes or layouts of the clusters: spherical or on a grid. The last options ignores all clustering information and spreads the nodes randomly over the viewing space.

Chapter 4

An example: The human protein reference database.

The Human Protein Reference Database (HPRD) [17] is a database that contains around 10000 proteins and data on the interactions of these proteins. It is curated through manual extraction of interactions from published articles. The data that is used are all derived from published protein-protein interaction experiments. The proteins are annotated with their corresponding GO term [18] and connected with a binary connection. The connection is also annotated with the source of its annotation and in what type of experiment the connection was found.

This database is a perfect example to show the capabilities of vinet, because it is of considerable size (around 40000 connections in over 10000 proteins) and because it offers an option to download all data in the database as a tab-delimited flat-text file. This file can easily be manipulated to generate a connection and an annotation file that are compatible with vinet. As an example, we have chosen to use the GO term annotation to cluster the nodes. We have used all three the GO terms that are available: the molecular function, the cellular component and the biological process. Figure 4 shows a screenshot of the HPRD clustered by the cellular component terms.

We see that there is a great variety of GO terms, as can be expected. See table 4.1 for an overview of the number of clusters. We see that in all three the GO terms there are a number of clusters that contain an extremely large number of nodes and very densely connected. These are usually general GO terms, such as "enzyme activity" (GO:0003824) in the molecular functions ontology or "metabolism" (GO:0008152) in the biological activity ontology.

In figure 4.2, we can also see the advantage of between-cluster edge grouping. The GO clusters are often so densely interconnected that the edges clutter the data model. The edge grouping option remedies this problem by displaying all the edges that go from one cluster to another as one single line. This reduces the screen clutter considerably and gives a better overview of the between-cluster

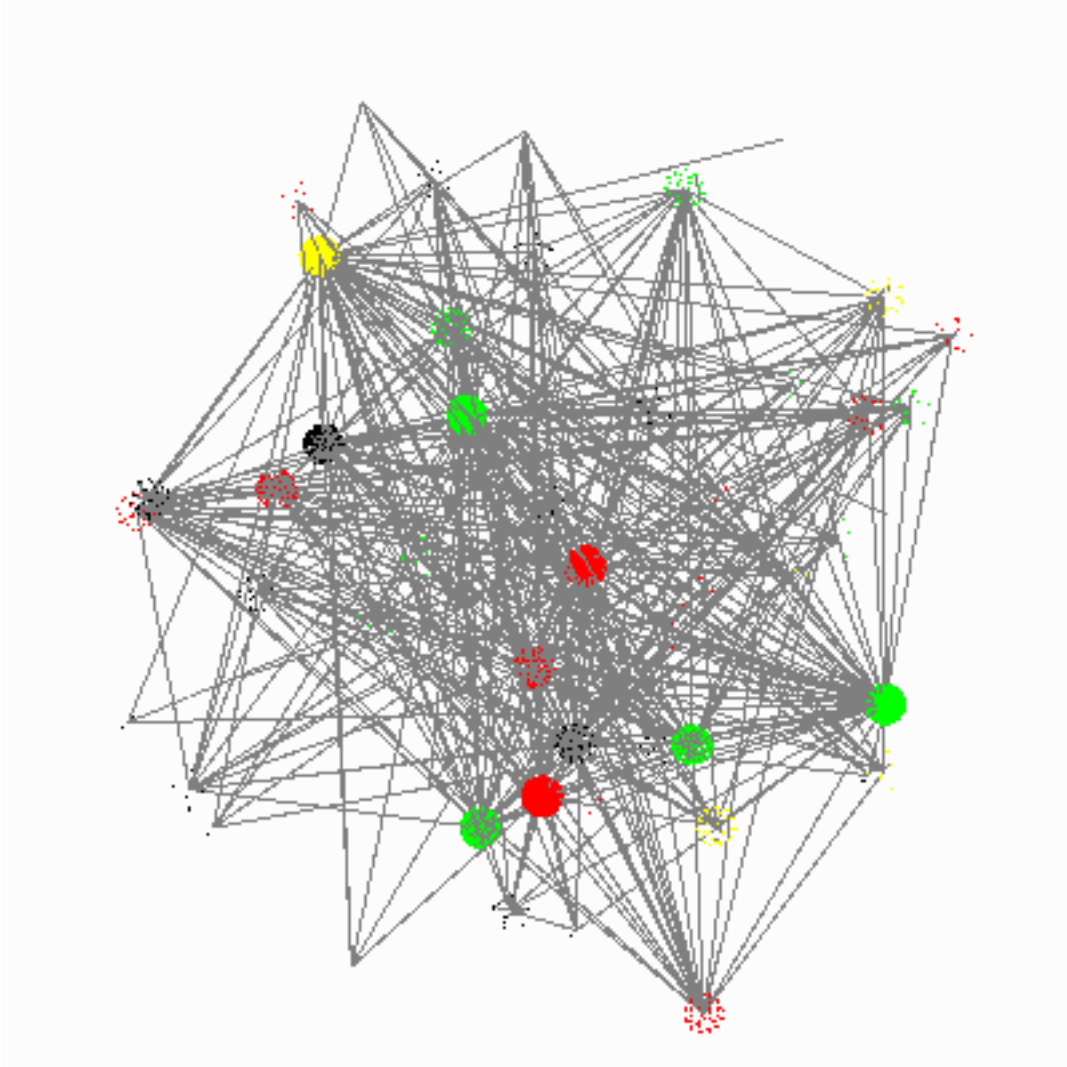
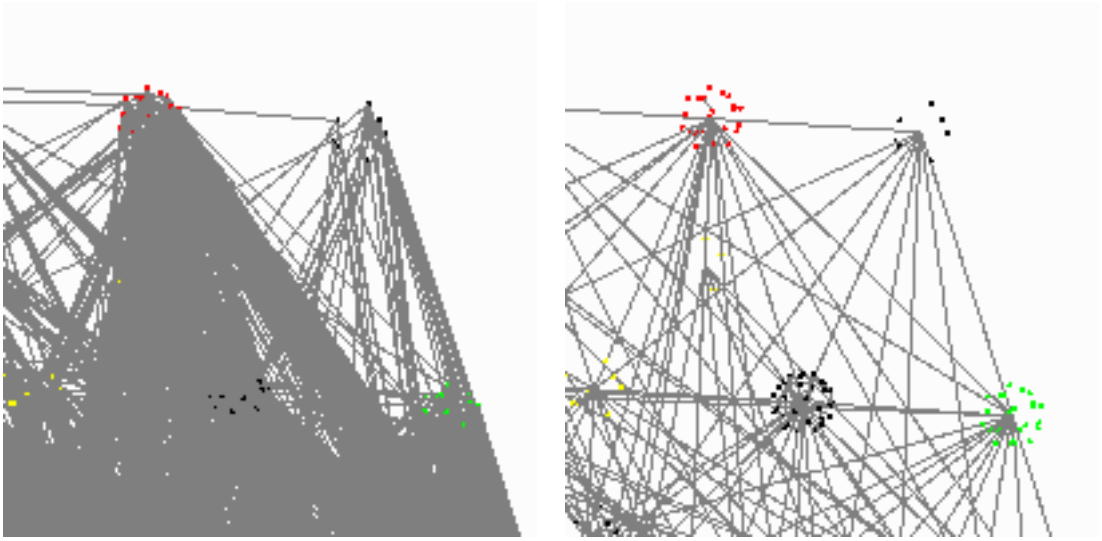


Figure 4.1: The entire HPRD network (10000 nodes, 40000 edges) rendered in vinet. It is clustered by the GO cellular component term; layout-wise, the clusters are spread organically and they are layed out spherically.

Type of clustering	Number of clusters
with netclust	108
by GO biological process	122
by GO molecular function	196
by GO cellular component	64

Table 4.1: The number of clusters generated using different clustering methods on the HPRD database.



(a) ungrouped edges

(b) grouped edges

Figure 4.2: A comparison between non grouped edges(a) and grouped edges(b). It is clearly visible that grouping the edges improves visibility and reduces screen clutter.

relationships.

Chapter 5

Conclusions and outlook

Currently, visualization is a very hot topic in bioinformatics. It is hailed as a useful way to understand complex data and it opens up whole new parts of research. To make this branch of bioinformatics successful, good tools are needed, some of which are already there. However, the current state of the art focuses on static data presentation, rather than data exploration.

Here, I have presented a tool that is a step towards the dynamic exploration of data. It handles large networks while maintaining a steady performance and it is lightweight, both in size and in working memory footprint. It has been developed for linux and it has been succesfully compiled on Windows, covering the 2 biggest research platforms. Its flexibility lies in the fact that it can be used for all types of networks that can be described in a simple network format, and its controls are intuitive. All in all, it can be said that vinet provides a good basis for the first steps into 3D visualization and that it provides plenty of room for expansion.

Outlook

vinet is only a part of an entire research pipeline. It can and must be adapted to the specific wishes of the researcher. Because it is built in a layered way, it is easy to integrate different tools into it such as statistics modules or new layout options. For vinet to have a future in bioinformatics research, it needs these kinds of expansions. This expanding will have to be research driven, similar to plugin development in the Cytoscape community.

The file formats mentioned in chapter 2 are the current standards in systems biology. For vinet to be more interoperable, it should implement some or all of these standards. However, there is currently very little support from the research community in using these standards because they are often complex, and there are no tools available that allow conversion between them. Incorporating these standards into vinet is therefore not very useful at this moment, but with time one standard may emerge as 'the' standard to be used.

vinet will have a future career as the first 3D rendering engine that is built specifically for tiled displays and the Bioinformatics group has expressed enthousiasm in using it as their basis to build further on. This is a promising outlook, as much more research will be available there in the future which will put the visualisation wall, and vinet to good use.

Chapter 6

Acknowledgements

Many thanks to Harm Nijveen of the Bioinformatics department at the WUR, for testing and borrowing all of his books. Many thanks to Arnold Kuzniar for explaining his netclust program and letting me use it in vinet. Thanks to Aalt-Jan van Dijk for providing us with useful test data in the first stages of the project. Also thanks to Jack Leunissen for coming up with the initial idea of hacking molecular viewers to display networks, which started it all. I want to thank everyone else at the Bioinformatics for their support and the fruitful discussions. And a big warm thanks to my girlfriend Rosanna, for getting me back to work when I didn't feel like it anymore.

This thesis was created using LaTeX.

References

[1] http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html, March 2010.

[2] Edward Tufte. *The visual display of quantative information.* Graphics Press, 1983.

[3] D. Merico, D. Gfeller, and G. D. Bader. How to visually interpret biological data using networks. *Nature Biotechnology*, 27(10):921 – 924, 2009.

[4] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[5] T. Kamada and S. Kawai. An algorithm for drawing general unidirected graphs. *Information Processing Letters*, 31:7–15, 1989.

[6] T. Fruchterman and E. Reingold. Graph draawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

[7] M. Tory, A. E. Kirkpatrick, M. S. Atkins, and T. Moller. Visualization task performance with 2d, 3d, and combination displays. *IEEE Transactions on Visualization and Computer Graphics*, 12:2–13, 2006.

[8] C. D. Wickens, D. H. Merwin, and E. L. Lin. Implications of graphics enhancements for the visualization of scientific data: Dimensional integrality, stereopsis, motion, and mesh. *Human Factors*, 36(1):44–61, 1994.

[9] G. A. Pavlopoulos, O'Donoghue S. I., V. P. Satagopam, T. G. Soldatos, E. Pafilis, and Reinhard Schneider. Arena3d: visualization of biological networks in 3d. *BMC Systems Biology*, 2(1):104, 2008.

[10] P. Shannon. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[11] A. Theocharidis, S. van Dongen, A. J. Enright, and T. C. Freeman. Network visualization and analysis of gene expression data using BioLayout Express3D. *Nature Protocols*, 4(10):1535–1550, 2009.

[12] B. J. Breitkreutz, C. Stark, and M. Tyers. Osprey: a network visualization system. *Genome Biology*, 4(3):R22, 2003. PMID: 12620107.

[13] L. Perrone. *Proceedings of the 2006 Winter Simulation Conference Monterey, CA, 3-6 December 2006*. IEEE Service Center, 2006.

[14] BioPAX working group. BioPAX biological pathways exchange language. level 1, version 1.0 documentation, 2004.

[15] M. Hucka. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[16] A. Kuzniar. *Graph-based methods for large-scale protein classification and orthology inference*. PhD thesis, Wageningen University, 2009.

[17] S. Peri, J.D. Navarro, R. Amanchy, T.Z. Kristiansen, C.K. Jonnalagadda, V. Surendranath, V. Niranjan, B. Muthusamy, T.K. Gandhi, M. Gronborg, N. Ibarrola, N. Deshpande, K. Shanker, H.N. Shivashankar, B.P. Rashmi, M.A. Ramya, Z. Zhao, K.N. Chandrika, N. Padma, H.C. Harsha, A.J. Yatish, M.P. Kavitha, M. Menezes, D.R. Choudhury, S. Suresh, N. Ghosh, R. Saravana, S. Chandran, S. Krishna, M. Joy, S.K. Anand, V. Madavan, A. Joseph, G.W. Wong, W.P. Schiemann, S.N. Constantinescu, L. Huang, R. Khosravi-Far, H. Steen, M. Tewari, S. Ghaffari, G.C. Blobe, C.V. Dang, J.G. Garcia, J. Pevsner, O.N. Jensen, P. Roepstorff, K.S. Deshpande, A.M. Chinnaiyan, A. Hamosh, A. Chakravarti, and A. Pandey. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Research*, 13:2363–2371, 2003.

[18] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25:25–29, 2000.