Linnæus University
School of Computer Science, Physics and Mathematics

Degree Project

ViNCent – Visualization of Network Centralities

Harald Köstinger
2011-02-15
Subject: Computer Science
Level: Master
Course code: 4DV01E

Abstract

In the area of information visualization social or biological networks are visualized in a way so that they can be explored easily and one can get more information about the structure of the network out of it.

The use of network centralities in the field of network analysis plays an important role when it comes to the rating of the relative importance of vertices within the network structure based on the neighborhood of them. Such a single network can be rendered easily by the use of standard graph drawing algorithms. But it is not only the exploration of one centrality which is important. Furthermore, the comparison of two or more of them is important to get some further meaning out of it. When visualizing the comparison of two or more network centralities we are facing new problems of how to visualize them in a way to get out the most meaning of it. We want to be able to track all the changes in the networks between two centralities as well as visualize the single networks as best as possible. In the life sciences centrality measures help scientists to understand the underlying biological processes and have been successfully applied to different biological networks.

The aim of the thesis is it to overcome those problems and to come up with a new solution of how to visualize networks and its centralities. This thesis introduces a new way of rendering networks including their centrality values along a circular view. Researches can then be focused on the exploration of the centrality values including the network structure, without dealing with visual clutter or occlusions of nodes. Furthermore, filtering based in statistical data concerning the datasets and centrality values support this.

Key-words: centralities, network analysis, visualization, social networks, biological networks, graph drawing

Acknowledgments

Thanks to my supervisors Prof. Dr. Andreas Kerren and Ilir Jusufi for supporting me at the University. Also thanks to my parents and my whole family for always supporting me and believing in my ways I am going.

Contents

List of Figures

List of Tables

Terms and Abbreviations

- **FBEB**
  Force-Based Edge Bundling
- **GraphML**
  Graph Markup Language
- **GUI**
  Graphical User Interface
- **HBEB**
  Hierarchical-Based Edge Bundling
- **OpenGL**
  Open Graphics Library
- **PPI**
  Protein-protein-interaction
- **SN**
  Social Network
- **SNA**
  Social Network Analysis
- **TR**
  Transcriptional regulation
- **VANTED**
  Visualization and Analysis of Networks with related Experimental Data
- **ViNCent**
  Visualization of Network Centralities
- **XML**
  eXtensible Markup Language

1 Introduction

In the area of information visualization social or biological networks are visualized in a way so that they can be explored easily and one can get more information about the structure of the network out of it. The analysis itself can be done based on several approaches. When dealing with social networks and the analysis based on a graph representation, usually some important tasks about network analysis are [HFM07]:

- **(T1) identify *communities***
  This means, that the researcher is more interested in groups of actors in the network, who are strongly connected to each other. To perform such an analysis, the tools should be capable of allowing the researcher to group together certain nodes according to attributes. Researchers then can easily study connection patterns in between the single nodes.

  Moreover, labeling of groups according to the common attributes is important in order to answer questions why those actors are grouped together. The possibility of finding *cliques*, groups where every single actor is connected to every other actor, and missing relationships should be given as well. This task is strongly related to attribute based research.
- **(T2) identify *central actors***
  By finding the most connected actors within a network, *central actors* can be identified. But not only the highest connected ones are important. Furthermore actors linking communities together are central actors as well. Those tasks can be performed by having a closer look at the topology of networks, moreover a closer look at centralities which are based on topological tasks such as the betweenness (representing a value which indicates the number of times the node is present in a shortest path between every pair of nodes within the graph).
- **(T3) analyze *roles and positions***
  This tasks analyzes the way, how actors in groups (communities, cliques) and outside of groups are interconnected at all. This task is more a way of interpretation but relies on the previous tasks such as finding common attributes and relationships between actors and groups.

Those tasks allow researcher to find the most important parts and facts in social networks. Typical questions like "Who is the head of a group?" or "Which communities are present in the network and with which topic are they dealing?" can be answered. For doing research based on the above mentioned tasks and features, there are several tools providing good overview and support, to finish those tasks. *NodeTrix* presented by Henry et al. [HFM07] for example uses a hybrid approach to do so.

When it comes to the exploration of a network according to network centralities, there is a lot of research to do, to provide scientists with a good tool to do so. Network centrality analysis measures the relative importance of vertices in a network based on their connectivity within the network structure [DHK+06]. It is commonly used to analyze network-structured data. Especially for analyzing the structure of biological networks the technique of centrality analysis is useful, since it helps to understand underlying processes within networks [KS04]. For biological networks vertices of a network represent transcripts, proteins or metabolites and are interconnected by edges, which represent correlations, interactions or reactions [JKS06b].

Applications in biological networks can be found at the investigation of protein-protein-interaction (PPI) networks and transcriptional regulation (TR) networks [KS04]. Typical tasks for such network analysis by the use of centrality values would be:

- finding nodes with high centrality values, since those are more likely of interest to the researcher
- finding nodes with low centrality values to hide them from the network, since they are of less importance
- finding nodes with high values in several centralities, therefore leading to comparison of more values

The last task as mentioned before for the analysis based on centrality values is the comparison of one or more centrality values within the network at the same time by the use of visualizations. So far, there is a lack of good tools supporting this work in a way, so that visual cluttering is reduced to a minimum and the user is provided a good overview all the time. Tools as presented by Dwyer et al. [DHK+06] like the comparison based on 3D parallel coordinates, orbit-based comparison and hierarchy-based comparison fulfill some of the requirements, but are still facing the problems of visual clutter and limited interaction possibilities. The larger and denser networks are, the harder are they to read and understand.

1.1 Motivation and Aim of the Work

Networks and therefore graphs can be easily rendered by the use of standard graph drawing algorithms and there is a lot of research going on for this part of the information visualization. Jia et al. [JHGJCH08] give a rough overview about existing graph layout algorithms and introduce a new approach for large scale-free networks.

When it comes to the research concerning the visualization of network centralities, the list narrows down as mentioned before. To introduce a new approach for a good way of rendering and comparing network centralities was the actual motivation for this work. When visualizing the comparison of two or more network centralities, new problems arise.

- How visualize data in a way, researchers can get to most meaning out of it?
- How enabling the user to keep track of centrality changes within the network?
- How to minimize occlusions and visual clutter?
- How to build a more flexible solution, to deal with a large number of centrality values at the same time?

In the life sciences centrality measures help scientists to understand the underlying biological processes and have been successfully applied to different biological networks [DHK+06, KS04]. This is a main application scenario. But furthermore, the analysis of networks based on centralities—and therefore the application of this tool—is not only limited to biological networks but can be used for all kinds of networks. This sets up a large background for the work.

The aim of the thesis is it to overcome the problems mentioned before and to come up with a new solution of how to visualize networks and its centralities. This thesis introduces a new way of rendering networks including their centrality values in a circular

view. Researches can then be focused on the exploration of the centrality values including the network structure, without dealing with visual clutter or occlusions of nodes. Furthermore, filtering based in statistical data concerning the datasets and centrality values support this and help keeping the network readable.

To summarize this up, the problem definition can be stated as follows. The tool should be able to visualize networks and their centrality values in a way, so that:

- centrality values are comparable,
- it minimizes visual clutter and occlusions for items in the renderings,
- it maximizes the usability and interaction features to allow the user to interactively filter and explore the network,
- but still allows the user to see the network and the underlying connections.

Goal criteria for the work are the above mentioned points. The comparability of the nodes is the most important goal to fulfill, followed by the minimization of visual clutter and occlusions. Whereas goal one, the comparability of values can easily be measured, goal two, the minimization of occlusions and visual clutter, can just be evaluated in comparison to other tools and existing solutions. The discussion deals with the single points and how they are solved.

1.2 Further Structure of the Thesis

The following sections deal with a rough overview about network analysis and network centralities and should provide a good introduction into the topic of centralities and their importance for (visual) network analysis in terms of social and biological network analysis.

The actual difficulties about existing solutions and the visual analytics of network centralities are summarized up and again named out. The main problems when dealing with the analysis of large scale networks, renderings of them and comparison of n-dimensional data like centrality values on a network are discussed.

To solve those problems and overcome those difficulties, the tool **ViNCent** is introduced. Sections about it bring up again the aim of the system in more details, and are later on dealing with the architecture of it and then providing a detailed description of the methods and approaches used to solve the beforehand described problems. Especially the problems when it comes to the rendering of networks are described in more detail and the used approaches are discussed. For further improvement of the interaction with the tool, features like filtering and hovering are described, therefore pointing out main advantages of the tool.

The discussion in the end figures out the advantages and disadvantages of the tool and furthermore describes the interaction concept of the tool based on a small use case scenario. The conclusions and future work sections deal with possible further improvements of the tool and planed further work.

2 Background on Networks and Centralities

This section deals with the actual background of the work. A brief introduction into graphs is given as well as a definition for them. The field of network analysis is discussed later on in Section 2.2 leading to the actual visual analytics of networks.

Furthermore, this section provides information about network centralities and their usage and outlines some problems when it comes to visualization of network centralities in the end.

2.1 Definition of Graphs

For the actual representation of any kind of networks, usually graphs are chosen, storing information about the single objects and the relationships between those. A graph G is build up with a finite set of nodes V and a finite set of edges E, therefore leading to the definition of $G = (V,E)$. The amount of nodes and edges can be easily calculated as $|V|$ and $|E|$. Edges $e = (u,v)$ in the graph G connect two nodes u and v. The nodes u and v are said to be *incident* with the edge e and *adjacent* to each other. The nodes which are adjacent to u are called the neighborhood of $u : N(u) = \{v : (u,v) \in E\}$. The degree $d(u)$ of a node u is defined as the amount of edges incident to this node u. [KS04, DHK+06]

Furthermore, we can define a *walk* on a graph. Let $(e_1,...,e_k)$ be a sequence of edges in graph $G = (V,E)$. This sequence is a *walk* if there are vertices $v_0,...,v_k$ such that $e_i = (v_{i-1},v_i)$ for $i = 1,...,k$. If the edges e_i are pairwise distinct and the vertices v_i are pairwise distinct the walk is called a *path*. The *length* of a walk or path is given by its number of edges, $k = |(e_1,...,e_k)|$. A *shortest path* between two vertices u,v is a path with minimal length. The *distance* ($dist(u,v)$) between two vertices u,v is the length of a shortest path between them.[KS04, DHK+06]

2.2 Network Analysis

Networks become more and more famous. Online communities for social networks as well as communities for software development and online platforms such as Wikipedia are growing faster and faster. Therefore, network analysis for social, biological, and computer sciences becomes more important as well to get further meanings out of underlying network structures. But it is not just social or technical background information, furthermore, a more connected world leads to faster spreading of potentially pandemic diseases which can be analyzed by the use of networks. The following sections deal with the analysis of such networks and name out important tasks as well as how these networks are usually visualized.

2.2.1 Analytics of Social Networks

Social networks are graphs, where single nodes correspond to actors (*people*) and the edges between those nodes correspond to *relationships*. Moreover, nodes and edges can define more properties and attributes such as:

- Nodes (*people*) can define names, a birth date, a home-town, or the current city people are living in.
- Edges (*relationships*) can define attributes such as when this relationship was set up or at which level this relationship is (friendship, related, married, siblings, etc).

Such graphs vary in size and density. Having graphs from global social networks, there might be many dense local parts in the graph building cliques, but low connected

parts when it comes to the global view. Dealing with local social networks usually means dealing with a dense graph but not having that many nodes involved. When dealing with social networks and the analysis of them, important tasks about network analysis are [HFM07, HF06]:

- "(T1) identify *communities*, i.e. cohesive groups of actors that are strongly connected to each other;
- (T2) identify *central actors*, bridging communities together;
- (T3) analyze *roles and positions*, which are tasks relying on the interpretation of groups of actors (positions) and connection patterns (roles)."

In order to be able to execute those mentioned tasks in a good way, the tools used should provide a basic set of functions. Task **T1** for example focuses on communities. Therefore actions such as grouping single actors together according to certain properties as well as finding relationships and missing relationships are important.

T2 demands a tool which is actually good in representing the topological view of networks to figure out high and low connected nodes. Moreover, nodes connecting communities should be found in an easy way.

T3 is more based on the attributes of a node or relationship since researchers want to find out how this person is integrated into a community or group. Therefore further information about the attributes of relationships and nodes is needed in order to fulfill this task.

For analyzing social networks, Henry et al. state out two big groups of tools, available to do this [HF06, HFM07]:

1. **Menu-based systems**
   Those systems usually provide a lot of functionality but are not that user friendly. Tasks are not that easy to fulfill since intuitive interaction with the tools is missing. Those tools are designed to support fine analysis conducted by an expert [HF06].

2. **Exploration systems**
   Focusing on the interaction with the network, those tools provide a good overview of the network allowing the user to focus down to certain parts of the network as well as investigating properties and relationships of nodes. If such tools get to lightweight, they might loose the ability to support the user in doing the tasks.

Section 2.2.3 deals with a short overview of existing tools for network representation and how they support the user in those above mentioned tasks.

2.2.2 Analytics of Biological Networks

The tasks for the analysis of biological networks might differ from the ones mentioned above. Whereas for social networks graphs are usually dense for local groups, this might not be true for biological networks. Furthermore, paradigms like single actors and communities cannot be applied to those networks directly. More investigations have to be done in order to get further meaning out of biological networks. Junker et al. state the following in their work about *CentiBiN*.

> "Structural analysis of networks can lead to new insights into biological systems and is a helpful method for proposing new hypotheses [JKS06b]."

For structural analysis, several techniques exist: analysis of the *global network structure*, *network motifs* (i.e., small subnetworks which occur more often within the whole network—might be more often the case for biological networks), *network clustering* and *network centralities*.

The latter technique is, what is currently state of the art when it comes to analysis of biological networks, since this technique uses far more information about the network than just the relationships and neighborhood of nodes. In fact, this technique uses the centralities of nodes to rank the elements in the network according to a given importance concept. [JKS06b]

Section 2.3 deals with network centralities in general, how to calculate them and how to visualize them, which is the important task when it comes to the analytics of biological networks.

2.2.3 Visual Analytics of Networks

For doing research based on the before mentioned tasks and features for social networks, there are several tools providing good overview and support, to finish those tasks. Landesberger et al. [vLKS+10] recently came up with a good overview of existing solutions for graph drawing and analysis. For their report they considered three main components of the software, used to analyze the graph:

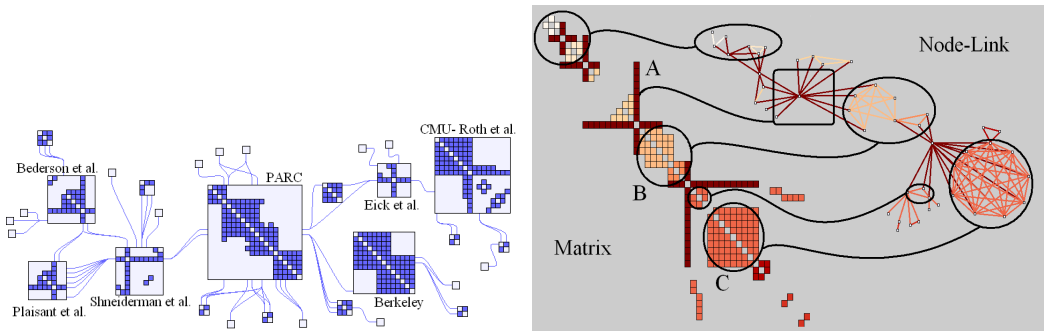
1. visual graph representation,
2. graph algorithmic analysis, and
3. user interaction.

But Landesberger et al. furthermore state in their research:

> "The exploration of large graphs is supported by effective interaction techniques, in particular, in cases when the whole graph is too complex or large to be visualized in one static view. The interaction alone may not be sufficient accomplish certain analytical tasks."

NodeTrix presented by Henry et al. [HFM07] for example uses a hybrid approach in order to be able to interact with and visualize the graph in a good way. Figure 2.1(a) shows this tool. It uses node-link-diagrams to show the global structure of the network but furthermore provides the possibility to group together nodes to a matrix-representation, therefore reducing the complexity of the network, but still providing all the information. The authors self state that *NodeTrix* is among the most effective and simplest tools to understand the network structures.

MatrixExplorer by Henry and Fekete [HF06] (as shown in Figure 2.1(b)) uses the matrix representation of a graph (the adjacency matrix) to visualize the graph and its attributes. Furthermore, it offers the possibility to display the matrix representation of the network next to the node-link-diagram, synchronized by brushing and linking. The authors argue, that users then can choose the best visualization for their work, however, the system requires two screens to work properly. As well as the space requirements, the cognitive load to the user and the divided attention is named out at problems for this tool. [HFM07, Hen08]



(a) NodeTrix (b) Matrix-Explorer

Figure 2.1: Presented work by Henry et al. [HFM07, HF06]. Figure 2.1(a) showing a hybrid representation of a network. Figure 2.1(b) showing the matrix representation of a network and the similarities between them. ***A*** shows an actor connection several communities, ***B*** a community and ***C*** a clique.

However, *MatrixExplorer* clearly shows the relationships between cliques, communities and single actors in the graph representation and the matrix representation (see Figure 2.1(b)).

Other tools using a *node-link* representation follow the standard approach for graph visualizations. Those tools however lack of certain exploration features and usually deal with visual clutter when graphs are getting dense. Approaches by Holten [Hol06] to reduce visual clutter by edge bundling improves the outcome and provides users with a good overview of graphs, but still are not able to display more information such as additional attributes to the nodes and relationships. Figure 3.6 shows an example of how this approach works.

A problem is still the representation and visualization of both, network structure and network attributes (e.g. centrality values). The following section gives an introduction to network centralities in general as well as the visual analysis of networks using network centralities.

2.3 Network Centralities

Section 2.1 introduced the definitions of a graph and its edges and nodes as well as some other functions based on a graph. Furthermore, definitions for network centralities are needed.

2.3.1 What are Network Centralities?

A centrality is a function assigning a node u out of the set V of a given graph G a value $C(u)$. In order to be able to compare network centralities according to their importance, u is more important than v iff $C(u) > C(v)$. [KS04, DHK+06]

Network centralities are used for understanding underlying processes in networks. If it comes to social networks, centrality measures can help to identify single actors, cliques or communities [HFM07, Hen08]. For life science centrality measures are useful to understand biological processes. Centrality measures are therefore applied to biological networks and then explored. [DHK+06, JKS06b, KS04]

The following list names out some of the centralities used by scientists to get further meaning out of networks [JKS06b, DHK+06, KS04]:

- **Degree**
  Using the degree value of a vertex ($d(u)$), this centrality is one of the simplest ones, ordering the nodes according to their degree value. The centrality measure (C_{deg}, *degree-centrality*) for it is defined as $C_{deg}(v) := d(v)$.

- **Eccentricity**
  This centrality value is calculated by the use of the distance between nodes in the graph. The *eccentricity ecc* of a vertex u is definded as $ecc(e) := max_{u \in V} dist(u,v)$ and the *eccentricity-centrality* (C_{ecc}) as $C_{ecc}(u) := \frac{1}{ecc(u)}$. More central nodes have therefore a higher value of C_{ecc}.

- **Closeness**
  The closeness uses the information about the whole network to calculate its value. It is defined as *closeness-centrality* $C_{clo}(u) := \frac{1}{\sum_{v \in V} dist(u,v)}$.

- **Random Walk Betweenness**
  This centrality models communication paths in networks (information transmission) and therefore matches problems in biological networks. For the *random-walk betweenness centrality* C_r the centrality of a vertex w is equal to the number of times that a random walk from u to v goes through w, averages over all u and v [KS04].

In addition to those four presented values, Junker et al. [JKS06b] use in total 17 centrality values for analysis of biological networks. Those centralities are listed in Table 2.1. For this table, in addition to the definitions in Section 2.1, some more definitions have to be made. $n = |V|$ define the number of vertices in the graph. σ_{st} is the number of shortest paths from s to t and $\sigma_{st}(v)$ is the number of shortest path from s to t crossing vertex v. A is defined as the adjacency matrix of graph G.

2.3.2 Which Centralities to Calculate?

The actual calculation of centralities is not a big deal. Even for large-scale networks this can be done very fast and cached to speed up exploration later on. Koschützki et al. [KS04] did the calculations on a desktop PC (3 GHz, 2 GB Ram, Linux 2.6.x, Java 1.4.2) and it took between less than a second (C_{deg}) and several minutes (C_r). This is due to the complexity of the single centralities, ranging from $O(n)$ to $O((n+m)n^2)$. Event though calculation can be done fast, a big problem stays: it is the question which centralities to calculate on the network.

There are many different centrality measures as for example lined out by Dwyer et al. [DHK+06] or Junker et al. [JKS06b] which can be used to analyze networks. The problem of choosing the right ones differs from network to network. Dwyer et al. [DHK+06] describe this problem as follows:

> "For example, in biological networks correlations between specific centrality measures and functionally important properties have been shown for some networks [JMBO01, WS03]. However, it has also been shown that the degree of a vertex alone is not sufficient to distinguish lethal proteins from viable ones [Wuc02]."

Name	Definition	Remarks
Degree	$C_{deg}(v) := d(v)$	For directed graphs in- and out-degree is used.
Eccentricity	$C_{ecc}(v) := \frac{1}{max\{dist(v,w):w\in V\}}$	
Closeness	$C_{clo}(v) := \frac{1}{\sum_{w\in V} dist(v,w)}$	
Radiality	$C_{rad}(v) := \frac{\sum_{w\in V}(\Delta_G+1-dist(v,w))}{n-1}$	Δ_G is the diameter of the graph G, defined as the maximum distance between any two vertices of G.
Centroid Value	$C_{cen}(v) := min\{f(v,w) : w \in V\setminus\{v\}\}$	Where $f(v,w) := \gamma_v(w) - \gamma_w(v)$ and $\gamma_v(w)$ denotes the number of vertices that are closer to v than to w.
Stress	$C_{str}(v) := \sum_{s\neq v\in V}\sum_{t\neq v\in V}\delta_{st}(v)$	
S.-P. Betweenness	$C_{spb}(v) := \sum_{s\neq v\in V}\sum_{t\neq v\in V}\delta_{st}(v)$	$\delta_{st}(v) := \frac{\sigma_{st}(v)}{\sigma_{st}}$
C.-F. Closeness	$C_{cfc}(v) = \frac{n-1}{\sum_{t\notin v} p_{vt}(v)-p_{vt}(t)}$	Where $p_{vt}(t)$ equals the potential difference in an electrical network.
C.-F. Betweenness	$C_{cfb}(v) = \frac{1}{(n-1)(n-2)}\sum_{s,t\in V}\tau_{st}(v)$	Where $\tau_{st}(v)$ equals the fraction of electrical current running over vertex v in an electrical network.
Katz Status	$C_{katz} := \sum_{k=1}^{\infty}\alpha^k(A^T)^k\vec{1}$	Where α is a positive constant.
Eigenvector	$\lambda C_{eiv} = AC_{eiv}$	The eigenvector to the dominant eigenvalue of A is used.
Hubbell index	$C_{hbl} = \vec{E} + WC_{hbl}$	Where $\vec{E}$ is some exogenous input and W is a weight matrix derived from the adjacency matrix A.
Bargaining	$C_{brg} := \alpha(l-\beta A)^{-1}A\vec{1}$	Where α is a scaling factor and β is the influence parameter.
PageRank	$C_{pr} = dPC_{pr} + (l-d)\vec{1}$	Where P is the transition matrix and d is the damping factor.
HITS-Hubs	$C_{hubs} = AC_{auths}$	Assuming C_{auths} is known.
HITS-Authorities	$C_{auths} = A^T C_{hubs}$	Assuming C_{hubs} is known.
Closeness-vitality	$C_{clv}(v) := WI(G) - WI(G\setminus\{v\})$	Where $WI(G)$ is the Wiener index of the graph G.

Table 2.1: Definition of centrality values (taken from [JKS06b], Table 1).

For the computation of the best centralities, there is usually data missing about the functional properties of networks. This data would allow to choose centrality measures which show the important parts of the network. Therefore, this analysis process is usually done by just visually comparing the centrality values on the networks. Later on, a hypothesis can be build upon the potential important parts found in the network and can be tested. [DHK+06]

2.3.3 Visual Analytics of Network Centralities

The visual analytics of network centralities by the use of visualizations is a quite new approach. Typical methods so far, as stated by Dwyer et al. [DHK+06], are:

- the use of the correlation (Kendall's correlation coefficient τ),
- scatter plots, and
- parallel coordinates.

The problem with this solutions so far is, that they have disadvantages when used for biological networks, since correlations of centralities might occur anyways. The most important thing therefore is not only to show that there are correlations, but to show where those correlations appear within the network. In their work Dwyer et al. [DHK+06] present three new techniques to visualize network centralities:

1. **3D Parallel Coordinates-based Comparison**
   This approach actually uses the technique of parallel coordinates to visualize multi-dimensional data. Standard approaches so far dealt with two dimensions. This approach uses the third dimension, to stack visual representations of a network according to one centrality into the third dimension, meaning, that each two dimensional plane contains the information for a particular centrality.

   Figure 2.2(a) shows such a 3D parallel coordinates-based rendering. Each vertical line represents a centrality measure. On each of this line, several horizontal lines exist, lining up the nodes sharing this centrality value. So called inter-plane edges are then added between the planes and the corresponding nodes. To reduce visual clutter, this approach allows the user to set a certain threshold for the difference in the actual centrality value of the node, until a connecting edge is shown.

   As last step, the ordering of the planes and the ordering of the nodes along one horizontal line are calculated to minimize edge crossings.

   This method gives a good overview of the centrality values within the network and about how many nodes fall into a certain range of centrality values. The time complexity to solve the ordering problems is linear if the centrality values for each centrality and the optimal ordering of the planes are given [DHK+06]. However, this approach does not reveal the actual network structure.
2. **Orbit-based Comparison**
   Arranging nodes in an orbit-based visualization (which is comparable with a circular layout of the network) gives some clear advantage over the 3D parallel coordinates approach. The network structure still can be rendered and the relationships between the nodes can be identified.

   The outcome of this approach is shown in Figure 2.2(b). For each centrality measure a new orbit-based plane is added to the rendering, again stacked into the third dimension. The ordering of the planes is again related to the minimization

problem for the edge crossings and that inter-plane edge lengths are minimized [DHK+06].

Since the single orbits give information about the centrality values (or at least the range the nodes fall into) and the network structure can be seen, this approach is better when it comes to revealing both, structure and centrality values.

Nevertheless, there are some problems though. The visualization has to deal with occlusions in the middle of the orbits and it is hard to keep track of changes within the single centrality measures. The sliding water-layer can be used to highlight one single measure, therefore providing better overview of one layer, but the changes of nodes in the layers behind cannot be followed that easy.

3. **Hierarchy-based Comparison**
   This approach uses the same basics as the 3D parallel coordinates, but divides the nodes according to their centrality values into layers. Those layers are then drawn as horizontal lines, having an ordering on the line as well. This could be for example decreasing centrality value from the left to the right. The top layers in the visualization are considered to be higher in centrality values. In this approach, there might be even connecting edges between nodes on the same layer.

   Again, the problem for ordering the planes and the nodes along the layers stays the same and is NP-hard even if there are just two layers [DHK+06]. Filtering and thresholds are used to reduce visual clutter in between two planes.

   Having a look at Figure 2.2(c) shows, that this approach reveals the hierarchy of a network and clearly brings out the single connections between nodes as well. Adding additional hints like slightly changing size of nodes for higher values in a centrality might enhance the whole outcome.

Junker et al. present a different approach with the tool *CentiBiN* [JKS06b]. Figure 2.2(d) shows the tool. *CentiBiN* uses the visualization of the graph as a normal node-link diagram rendered according to several different approaches for graph drawing algorithms. In addition to the displayed graph, the single centrality values for the measures are displayed next to the visualization in a table.

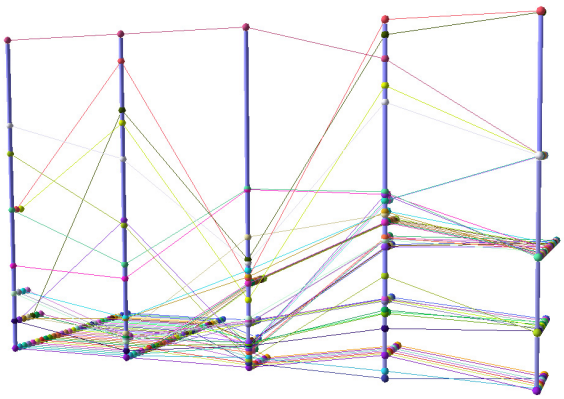
Interactions with the table like marking certain centrality values spread to the visualization of the network as well, therefore allowing the user to locate certain values within the network. Furthermore, a visualization of the centrality values of one measure as histogram is possible. The possibility to open more of them at the same time allows the user to compare data by the use of histograms in a statistical way.

CentiBiN has clear advantages when it comes to the actual computation of centrality values, since it can calculate up to 17 centrality measures for networks, depending on the kind of network. (see Tables 2.1 and 3.2)

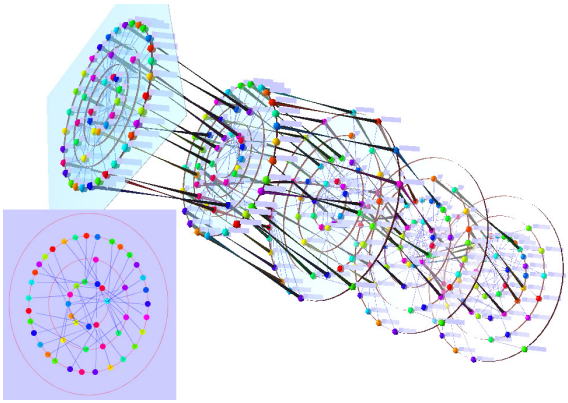
Although those presented solutions are able to visualize and analyze the network in a good way, there is still space for improvements though. The following section deals with some difficulties those presented solutions have and of which one has to be aware of.

2.4 Difficulties

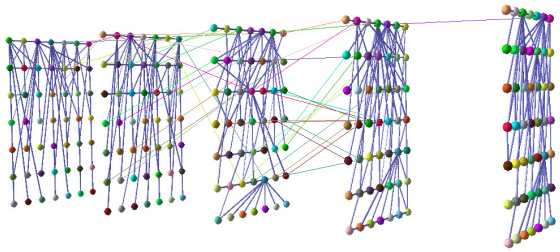
All the before presented tools and approaches solve the problem of network analysis according to centrality values in their own way. Since there is a wide range of different approaches now, the main difficulties should be figured out and summarized here in a short way. Those difficulties should be overcome by the tool **ViNCent**.



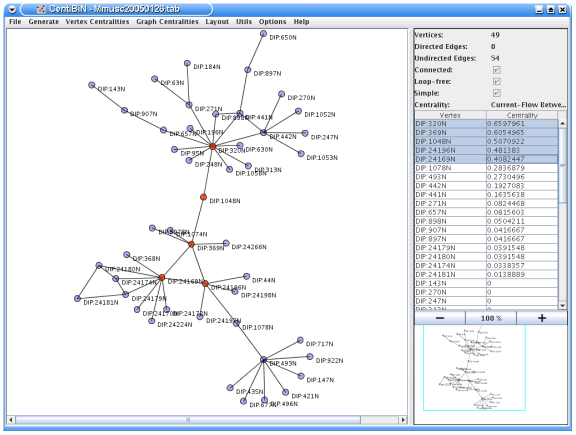
(a) 3D parallel coordinates



(b) Orbit based



(c) Hierarchy based visualization



(d) CentiBiN visualization

Figure 2.2: Overview of existing visualization techniques for biological network centralities all of them showing different approaches to solve the same problem. Figures 2.2(a), 2.2(b) and 2.2(c) taken from [DHK+06], Figure 2.2(d) from [JKS06b].

2.4.1 Arrangement of Nodes and Planes

To calculate an optimal arrangement of the nodes (and for some approaches as well the planes) is one of the hardest parts. A perfect arrangement is reducing occlusions and visual clutter, as stated in the next section. But those problems are usually NP hard and therefore not that easy to solve in real-time even with good heuristics.

Especially this part makes real-time visualization hard and not that interactive as actually wished. The use of caches for arrangements of nodes and planes however can speed up visualizations.

2.4.2 Occlusions and Visual Clutter

Even if the arrangement of nodes and planes can be calculated exactly and optimized, occlusions and visual clutter may occur due to stacking of planes into three dimensions (see Figure 2.2(b) for example). Thus, arrangement of nodes is not a guarantee for less visual clutter and occlusions.

However, different ways of rendering information and new approaches for the used technique can overcome occlusions of nodes and visual clutter for connecting edges.

2.4.3 Visualizing Structure and Centralities at the same Time

The problem of combining both structure and centrality values into one visualization is overcome by *CentiBiN* by the use of tables displaying the values. However this visualization shows all relevant data, it is not really nice and interactive. Since centrality measures can be multi-dimensional data for nodes, it is really hard to find an optimal solution for this problem.

2.4.4 Interaction and Filtering

Another problem the so far mentioned tools do not solve in a sufficient way is the problem of user interaction and filtering possibilities in order to explore the data set. Interaction with the visualization in real time can speed up the research process thus leading to better results. Moreover, good interaction techniques such as highlighting, brushing and linking supports the user in finding relationships amongst nodes.

Filtering approaches furthermore support the before mentioned points of reducing occlusions and visual clutter. Filtering out data means filtering out nodes and connecting edges, thus leading to less elements which have to be drawn on the display. Good ways of allowing the user to filter data (e.g. based on histograms and therefore the distributions of centrality values) have to be considered.

3 ViNCent - the Tool

The actual outcome of all the research and work done is the tool named **ViNCent** - short for ***Vi**sualization of **N**etwork **C**entralities*. The following sections describe the basic ideas and concepts behind this tool as well as the single features in more detail. Especially Section 3.3 deals with more details on the circular arrangement of the nodes and the edge- and bar-rendering.

3.1 Aim of the System

In order to overcome the difficulties shown in Section 2.4 new approaches have to be considered. Visualizing network centralities might not be hard if it comes to the comparison of two of them. But since the tool should be able to offer the opportunity to compare more centralities at the same time, still providing a good overview and interaction, it gets harder to do this. Many other existing solutions have to deal with occlusions in the visualization or the interaction of the tool might be hard, since it is difficult to navigate nodes and centralities [DHK+06, FHN+07].

The clear aim is it to build a scalable tool which is capable of displaying several different centrality values at the same time without occlusions, still offering the possibility for good interaction and a responding interface.

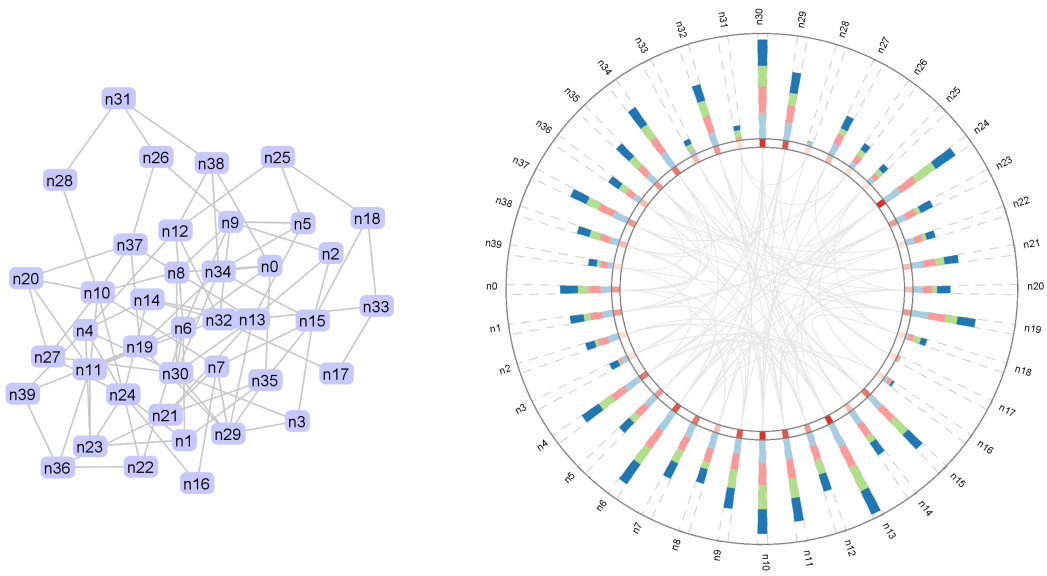
ViNCent solves this problems in a way, that the displayed network (and therefore its single nodes) is not rendered as graph - as usually done - but drawn on a circular view. Every single node in the network becomes a node on this circle having its connections to the others rendered in the inside of this circle. Figures 3.1(a) and 3.1(b) show this.

Even though there are only 40 nodes in the sample graph, occlusions and crossings are present in the rendering of the graph (Figure 3.1(a)). It would not be possible to add other information to this visualization, like displaying values corresponding to the centrality values of single nodes. Furthermore, it is hard to tell which nodes are connected and which of them are higher connected (according to the degree) than others. Depending on the field of research, nodes with higher or lower connection degree might be of more interest than others. Having a closer look at the rendered circular arrangement instead (Figure 3.1(b)), it is more clear how nodes are interconnected, and how many connections a node has. Moreover, features like edge-bundling and degree-marking as described later on in Section 3.3 support the user in finding connections in between nodes and high or low connected nodes.

3.2 Architecture

ViNCent uses the *prefuse* information visualization toolkit [HCL05] to render the visualizations. Mostly relying on the basic workflows and information visualization reference model of *prefuse*, the tool extends several classes in order to be able to render and display items in this special way. Figure 3.3 shows the package guide of prefuse, following the information visualization reference model.

The source data is loaded via the *prefuse.data.io* package into the data tables. Based on this data tables (graph, tree, table, ...), *prefuse.action* classes filter, layout, color, shape and size the elements. The visual abstractions (in *prefuse.Visualization*) are then rendered by the use of the *prefuse.render* package to the acutal *prefuse.Display*, which finally displays the visualization to the user. A feedback-loop for controls such as tool-tips, panning and zooming or queries on data is done to the data tables, to provide the possibility of filtering, zooming and panning and further actions.



(a) Sample, showing the network rendered as graph

(b) Sample, showing the network rendered as circle

Figure 3.1: Comparison of the graph and circle view showing the same network.

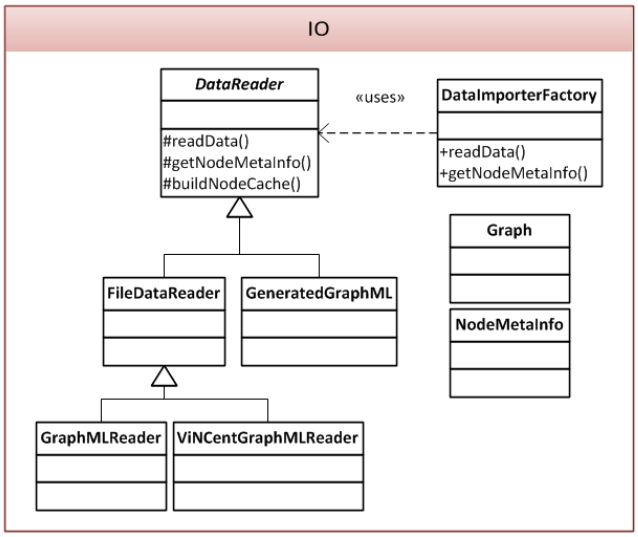


Figure 3.2: ViNCent IO package structure

ViNCent adapts and extends several classes from the *prefuse* packages. The following sections describe in a short overview the main extensions to the toolkit. Figure 3.4 shows the package overview of the tool.

3.2.1 The Data Loader (vincent.io.*)

One important thing is the possibility to keep the import of files and therefore networks to **ViNCent** as flexible as possible. This is ensured by the *vincent.io.** package and classes, responsible for loading all the data to the tool. Figure 3.2 gives an overview of how this small package is organized.

The main work is done by the `DataImporterFactory` class, which, according to settings of the tool or start parameter, loads the correct `DataReader` implementation.

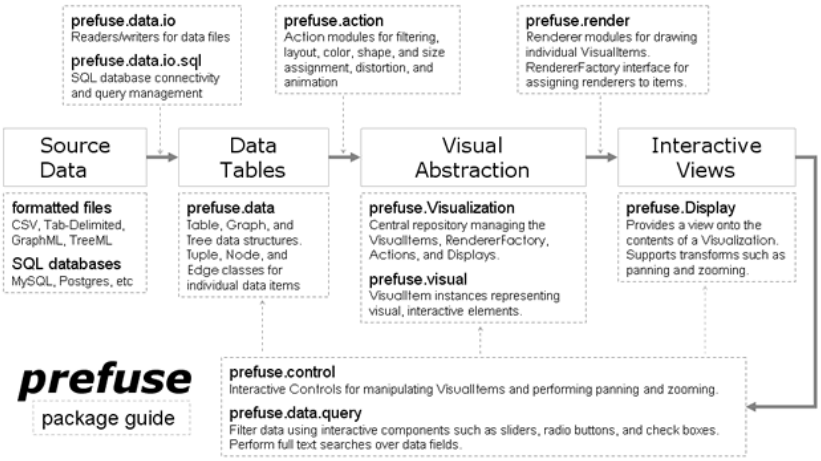


Figure 3.3: prefuse package guide (taken from [HCL05], toolkit structure)

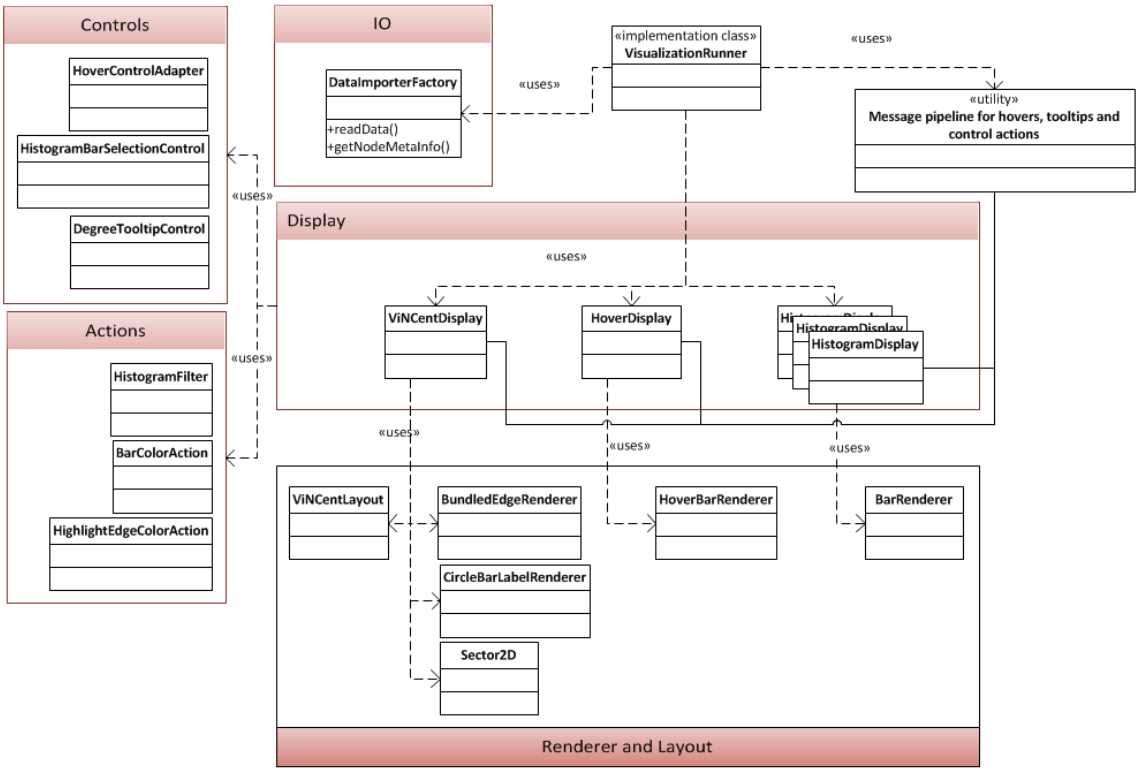


Figure 3.4: ViNCent package structure overview

So far, there exist three implementations:

1. `GraphMLReader`
   This class is responsible for and capable of reading standard *GraphML* files and building up the node cache and node meta information for the loaded graph. Furthermore, this component prompts the user to choose the fields for labels and centralities of nodes. The desired centralities are then calculated.

2. `ViNCentGraphMLReader`
   Is the class to import *ViNCent GraphML* files, having the centralities already calculated and stored as properties of the nodes.

3. `GeneratedGraphML`
   The generator fakes desired centrality values by the use of the *Math.random()* methods. It is capable of generating up to ten faked centralities for a graph having a certain number of nodes and desired density.

More aboout *GraphML*, *ViNCent GraphML* and data loading in general in Section 3.9.

3.2.2 The Display Actions (vincent.action.*, vincent.layout.*)

Those packages (see Figure 3.4) group together all the actions happening on the display. Especially for the circular arrangement of the nodes, special classes are needed for doing the layouting. `ViNCentLayout` (located in *vincent.layout.ViNCentLayout*) arranges the nodes according to the current display settings along the circle. Some important actions within the *vincent.action.** package are:

`HistogramFilter`
This class is applying the filter actions coming from the displays to the underlying data tables.

`BarColorAction`
Responsible for doing the coloring of the bars and stacked bars, this class keeps a cache for the used colors to assign them to the visual items when needed.

`HighlightEdgeColorAction`
When nodes or edges are hovered, they are going to be highlighted in the visualization. This class is capable of doing this and furthermore provides the opportunity to implement the 'Highlight spreading' feature, discussed in Section 5.

3.2.3 The Display Controls (vincent.controls.*)

Containing classes to capture mouse actions, this package supports the most highlighting and hover features of the application as well as all the filter actions spread out from those classes. There are several classes implemented here, whereas the most important ones are (see Figure 3.4):

`HoverControlAdapter`
This class is responsible for filling the message pipeline (see Figure 3.4, utility message pipeline) when hovering nodes with the needed data. Classes observing the message pipeline like the 'Hover window' will then become notified of new actions, therefore changing their displayed data.

`HistogramBarSelectionControl`
Hovering and clicking nodes in the histogram views spreads out a lot of information to the corresponding nodes in the circle view. Again, this class uses the message pipeline utility to notify observers of the actions. The before described `HistogramFilter` action is one observer for this action to filter out the nodes.

`DegreeTooltipControl`
Providing more detailed information for tooltips, this control fires events to the message pipeline to force other controls to display the tooltip text.

3.2.4 The Rendering Pipeline (vincent.render.*, vincent.geometry.*)

For rendering all the different items on the visualizations, **ViNCent** uses its own renderers and geometries in order to be able to fulfill this requirements. The three most important classes in those packages are the classes responsible for rendering the circular arrangement:

`BundledEdgeRenderer`
This renderer supports the edge bundling modes described in Sections 3.3.5 - 3.3.8 and is therefore the single point to touch if new rendering modes for edge bundlings are introduced.

`CircleBarLabelRenderer`
Having the information about the nodes and the corresponding centrality values, this class renders the stacked bars along the circle by the use of the later described `Sector2D` class according to the relative percentages of the centrality values. More about this rendering process of the stacked bars is described in Section 3.3.2.

`Sector2D`
Since drawing a circle and rendering bars to the outside of the circle leads to not straight shapes, this class deals with the representation of a single sector of one stacked bar in the visualization.

3.2.5 The VisualizationRunner (vincent.gui.*)

The actual executable of the tool is the `VisualizationRunner` class, which keeps together all the information and controls the single displays and windows. Responsible for setting up the visualization in the beginning, this class furthermore provides the message pipeline instances for the displays and controls to use. All settings on the GUI are spread via this class to the actual classes.

The following sections explain certain features of **ViNCent** in more detail. Figure 3.5 shows the complete **ViNCent** tool and how the single panels are arranged in the tool. The central part (marked in red) is the actual rendering of the network in a circular view as described in the following Section 3.3. To the left and to the right of the actual rendering, additional panels for settings and filterings as well as hoverings are present.

The settings panel to the left (marked in blue) allows the user to control how the rendering looks like and how the histogram filtering should work. Additional settings allow the user to generate test data. Section 3.6.1 deals with this panel in more detail.

The panel marked in green to the right of the actual rendering is the histogram panel. Over here, the distributions of each centrality within the network is rendered as histogram allowing the user to filter out data. Section 3.4 deals with this feature.

Below the histogram panel, a panel showing the current hover information is present. (marked in orange) This panel shows the additional information of a hovered node in the circular view. See Section 3.5 for more information.

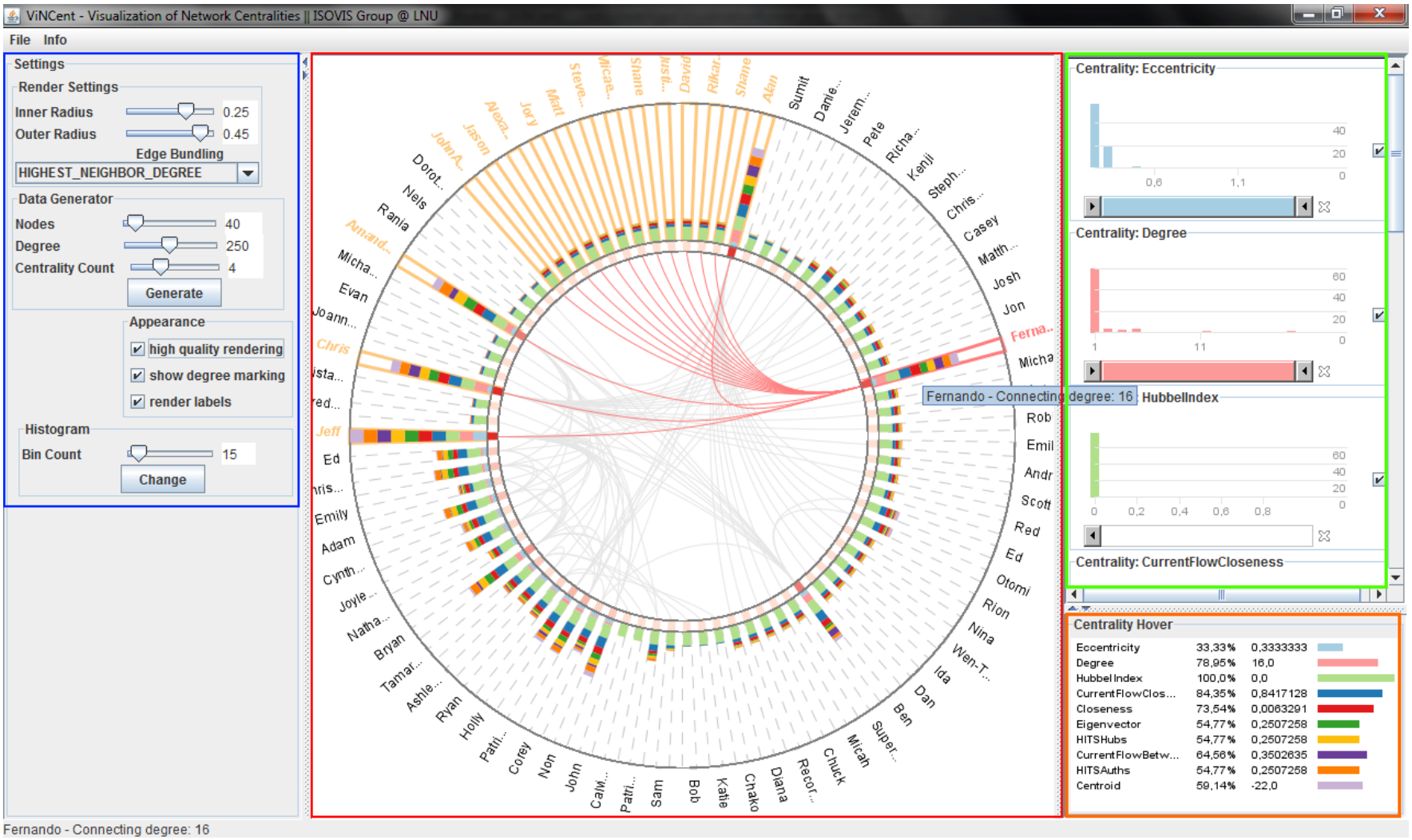


Figure 3.5: Overview of the **ViNCent** tool. The colored boxes show the different features of the tool.

3.3 Rendering Features

The actual advantages of the tool are its scalability concerning the amount of nodes and the capability of displaying several different centrality values at the same time. More or less, those advantages are achieved by the circular arrangement of the nodes and the additional edge-bundling done in the middle of the circle. The latter supports the user in finding connections between nodes and telling if nodes are high- or low-connected. Figure 3.5 shows the actual rendering of a network (red colored part).

3.3.1 Circular Arrangement of Nodes

For the circular arrangement of the nodes, the whole available space in the display is taken for the rendering of the inner and outer circle. Along the inner circle, the single nodes are added, drawing their centrality values to the outside and their connection degree marker on the inside.

The arrangement starts by dividing the whole 360 degree circle into single-angle steps, used to define the positions of the nodes. Let V be the set of nodes in the network, the single-angle α calculates as follows (see Equation 3.1):

$$\alpha = \frac{2 * \pi}{\|V\|} \tag{3.1}$$

Let *innerRadius* be the radius of the inner circle of the circular view, c_x and c_y define the center of the display, and i be the index position of a node n in the dataset V, then the position of a node n on the circle calculates as the follows (see Equation 3.2):

$$\beta = \alpha * i \tag{3.2}$$

$$x = c_x - \cos(\beta) * innerRadius \tag{3.3}$$

$$y = \sin(\beta) * innerRadius + c_y \tag{3.4}$$

Having calculated the x and y coordinate of the node n, it can be placed on the circle.

3.3.2 Centrality Bars

The space from the inner circle to the outer circle defines the available space for rendering the bars corresponding to the nodes centrality values. Since the single centrality values can range from $-\infty$ to $+\infty$, the rendering of the single bars has to be done in a relative way based to the minimum and maximum values of each centrality. This makes the comparison of values drawn in the bars possible.

To calculate the height of a bar, the total amount of stacked-bars (amount of centralities in the visualization) is taken into account, to calculate the maximum height of one bar. Let C be the set of centralities and *outerRadius* be the radius of the outer circle, then the maximum height h_{max} of a bar calculates as:

$$h_{max} = \frac{(outerRadius - innerRadius)}{\|C\|} \tag{3.5}$$

Having the maximum height of one bar, the calculation of the relative size of the bars corresponding to their centrality values can be done.

Let $v_{(i,k)}$ be the value of the node i and centrality k, and max_k and min_k be the maximum and minimum values of the centrality k over all nodes, then the scaling factor s calculates as:

$$s = \frac{(v_{(i,k)} - min_k)}{(max_k - min_k)} \tag{3.6}$$

Using this scaling factor, the height $h_{(i,k)}$ of one stacked bar (corresponding to the centrality value) of one node can be calculated:

$$h_{(i,k)} = h_{max} * s \tag{3.7}$$

In this manner, all the heights of the single stacked bars can be calculated to render a full bar on the outside of one node, representing its centrality values. In order to distinguish between the single centrality values, a color schema is used. **ViNCent** uses the color-schema introduced by the *ColorBrewer* [Bre0x], which means, a centrality gets assigned a certain color out of this set and is always (during one visualization process) rendered with the same color in all the different windows, to allow the user to keep track.

3.3.3 Degree-Marking

Besides the edge-bundling, which is discussed in the next sections, the degree-marking is done to support the user when trying to find the highest connected nodes in the network. Therefore, a color is used to mark this in addition.

ViNCent uses a reddish color schema to distinguish between the connection degrees. Based on the highest connection degree d_{max} of all the nodes n in V and the lowest connection degree d_{min}, a dark red color tone is interpolated to a very bright red tone. This method assigns each connection degree within the range of $d_{max} \rightarrow d_{min}$ a color.

This color is rendered to the inside of the circle, setting up the connection point for all the incoming and outgoing edges as well.

3.3.4 Edge-Bundling: a General Introduction

Graph rendering techniques usually have to deal with occlusions and visual clutter when it comes to more dense graphs. In the area of network analysis especially for social networks, this is usually the case. In order to reduce visual complexity in the renderings, the arrangement of nodes along a circle avoid occlusions of nodes. However, there is still the problem of the visual clutter when it comes to the drawing of the connecting edges in the middle.

Hierarchical edge bundling as a technique to resolve this problem basically follows simple principles: visually bundling adjacency edges together analogous to the way electrical wires and network cables are merged into bundles along their joint path [Hol06]. Danny Holten names out three main advantages about hierarchical edge bundling [Hol06]:

1. It is a flexible and generic method.

2. It reduces visual clutter when dealing with large numbers of adjacency edges.

3. By adapting various parameters of the techniques the outcome of the bundling can be controlled in an intuitive way.

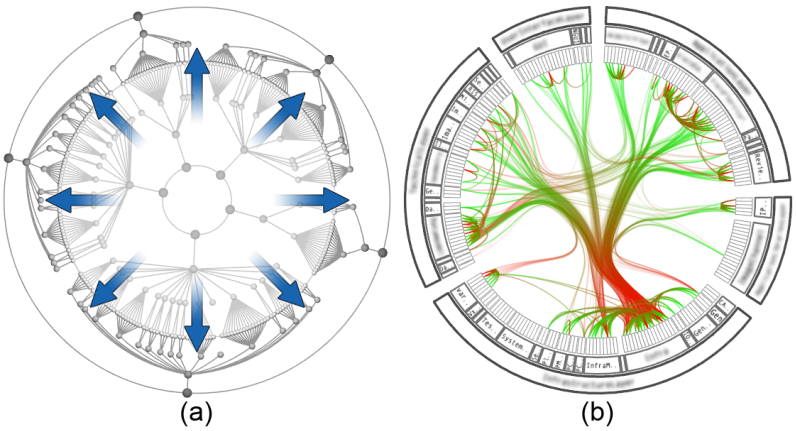


Figure 3.6: Hierarchical based edge bundling as introduced by Danny Holten. It relies on the inner structure to build up the pathways later on. (taken from [Hol06], Figure 12)

This technique works for hierarchical data. **ViNCent** is dealing with graphs as well, but there is a slight difference though. The tool renders all the nodes to the outside of the circle, therefore no hierarchy can be used in the center of the circle, to do edge bundling based on the inner hierarchy. The original hierarchical edge bundling approach presented by Danny Holten relies on an inner hierarchy, usually represented by the e.g. directories or package names, when dealing with source code data. All the nodes on the outside of the circle are then leaves of the spanning tree of the graph, interconnected along the spanning tree which is drawn on the inside. Figure 3.6 shows this in more detail. The initial structure in the middle is constructed via the radial tree layout. The single nodes in inside function as control points for the pathways later on.

The following four Sections (3.3.5 - 3.3.8) show how **ViNCent** implemented the edge-bundling so far. Figure 3.11 shows all the four implemented bundling technologies in comparison. Since the bundling is done at a single point, this can be easily exchanged or extended. A more detailed look on possibilities how to extend the existing edge-bundling modes with other options is discussed in Section 5.

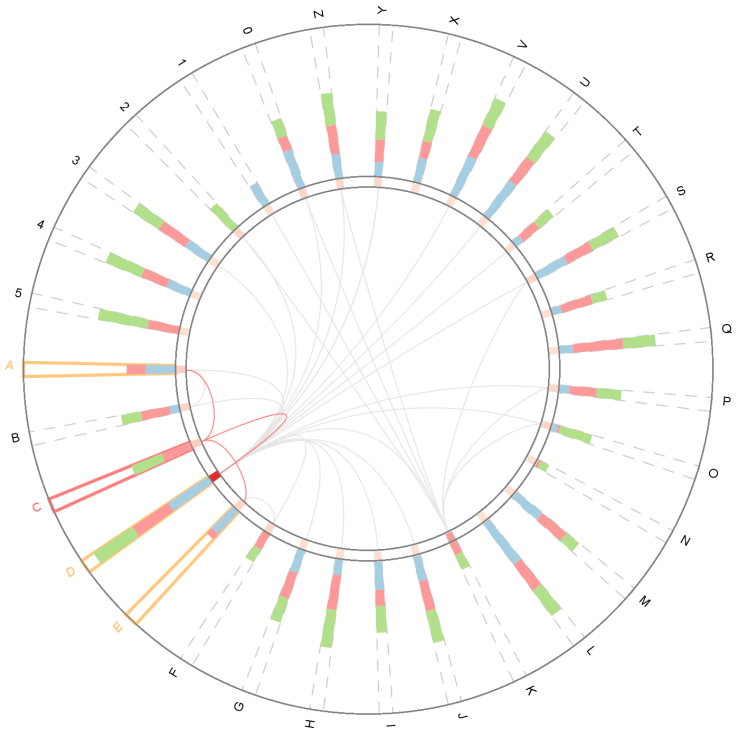
3.3.5 Edge-Bundling: Highest Neighbor

The first and default bundling mode for edges is the so called **Highest Neighbor Degree** mode. As all the other modes as well, this mode influences the control points of the drawn cubic (Bezier) curves by taking into account the connection degrees of the single nodes along the circle.

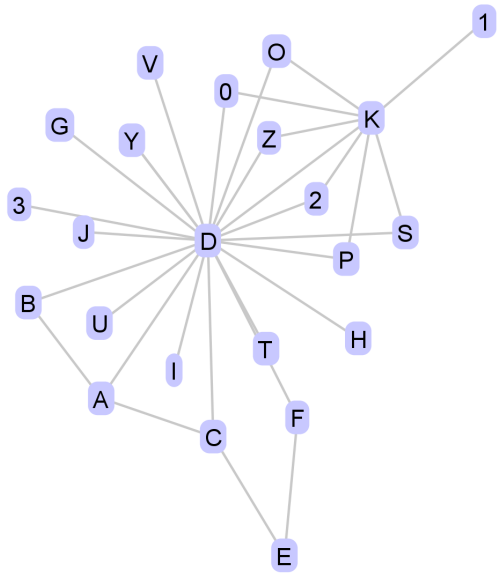
To set up the control points for the curves, this mode uses a build up cache to look up the connection degrees of nodes. It not just uses the connection degrees of the two nodes which are currently connected, furthermore it uses the degrees of the whole neighborhood of the two involved nodes.

To explain the concept in more detail, the sample graph and network as shown in Figure 3.7 is taken. To start drawing the edges, the cache for the neighborhood degrees has to be build. This cache stores the highest connection degree within the neighborhood of one node. Table 3.1 shows the degrees for the first few nodes, whereas the subindex shows, from which node the degree is taken from.

Having this cache in background, the tool can then start calculating the single control points for the curves.



(a) Sample network



(b) Sample graph

Figure 3.7: Network rendering showing the network which is used for further explanation of the edge bundling concepts in Figure 3.7(a) and the corresponding graph in Figure 3.7(b)

Degree-Type	A	B	C	D	E	F
own-degree	3	2	3	21	2	2
neighborhood-degree	21_D	21_D	21_D	21_D	3_C	21_D

Table 3.1: Neighborhood degrees in cache

Let α be the single-angle for one step along the circle and i and j the corresponding index for the two involved nodes A and B in the set of nodes V, then the angles for the two nodes calculate as:

$$\beta_A = \alpha * i \tag{3.8}$$
$$\beta_B = \alpha * j \tag{3.9}$$

Figure 3.8(a) shows the two angles β_A and β_B. Having those values and the cache for the connection degrees, the position of the control points (marked with a red star in the figure) can be calculated. Let $d(A)$ be the degree of node A and $d(B)$ the degree of node B respectively, and $d_{NBH}(A)$ be the highest degree in the neighborhood of node A and $d_{NBH}(B)$ the highest degree in the neighborhood of node B. Since the edge bundling mode takes into account only the highest degree, the corresponding value for it calculates as:

$$d_{highest} = max(d_{NBH}(A), d_{NBH}(B)) \tag{3.10}$$

Having this value and the radius r of the circle, the distance of the red star along the line from the center of the circle (given as c_x and c_y) to the corresponding node evaluates as:

$$dist_A = r - (r * (d(A)/d_{highest})) \tag{3.11}$$

$$dist_B = r - (r * (d(B)/d_{highest})) \tag{3.12}$$

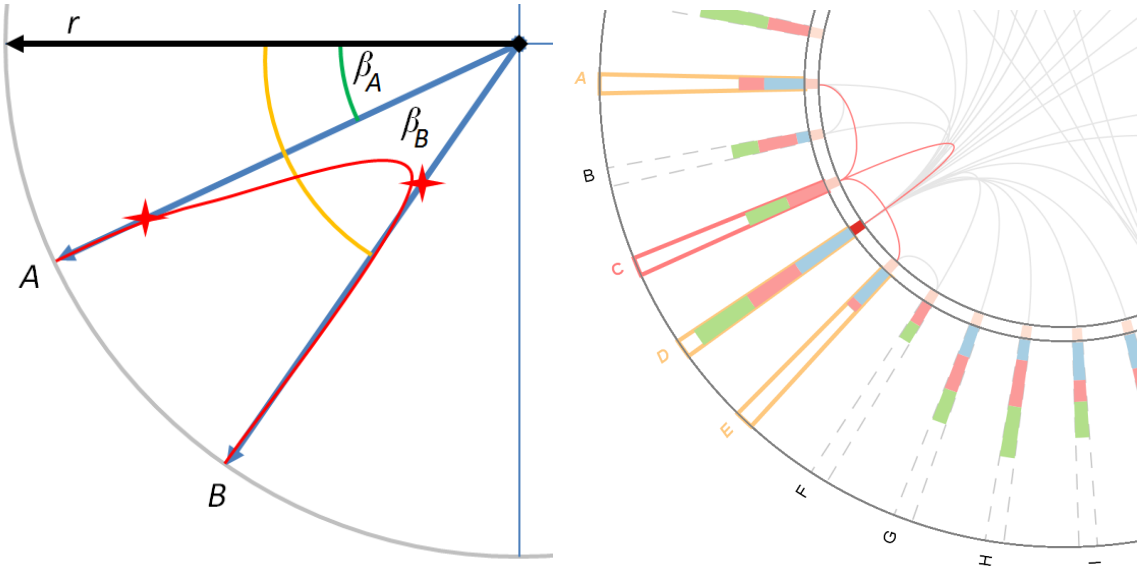
The corresponding control points for the curve are then calculated as:

$$CPA_x = c_x - \cos(\beta_A) * dist_A \tag{3.13}$$
$$CPA_y = \sin(\beta_A) * dist_A + c_y \tag{3.14}$$
$$CPB_x = c_x - \cos(\beta_B) * dist_B \tag{3.15}$$
$$CPB_y = \sin(\beta_B) * dist_B + c_y \tag{3.16}$$

Finally, a *Bezier* curve is drawn from node A to B by the use of the two newly calculated control points *CPA* and *CPB*. Figure 3.8(a) shows a schematic sketch for the whole calculation including the two control points shown in red as well as the resulting curve from A to B in red too.

Figure 3.8(b) shows the result of this edge bundling mode. This mode forces edges between low connected nodes farther to the outside of the circle (as illustrated between node A and C as well as C and E). Nodes having a connection with a high connected node in its neighborhood force the edges farther to the middle of the circle, therefore bundling in front of the higher connected node (see nodes C and D). This all is done by the scaling of the degrees in Formula 3.11 and 3.12. If nodes are low connected but connected to a higher connected one, this term gets bigger, therefore leading to a control point farther to the inside of the circle. If both of the involved nodes are low connected, this leads to control points farther to the outside of the circle, therefore resulting in a curve on the outside of the circle too.

This mode produces so far the best results for bundlings in the circle view of the tool. The following modes are simpler versions of this mode leading to slightly different results, which are useful in certain cases as well.



(a) Highest Neighbor Degree Sketch

(b) Highest Neighbor Degree results

Figure 3.8: Basic principle and results of the 'Highest Neighbor Degree' edge bundling mode

3.3.6 Edge-Bundling: Both Degrees

This mode, in distinction to the previous presented one, takes into account only the degrees of the nodes A and B which are connected with the current edge. The degree values are therefore $d(A)$ and $d(B)$ respectively.

For this mode, the distance of the control points $dist_A$ and $dist_B$ as well as the corresponding coordinates of the points $CPA_{x,y}$ and $CPB_{x,y}$ are calculated in the same manner. The difference is given in the calculation of the $d_{highest}$ value.

$$d_{highest} = max(d(A), d(B)) \tag{3.17}$$

Figure 3.9 shows the results of this edge bundling mode. Since no neighborhood is taken into account, edges from low connected nodes also range farther to the middle of the circle, but are still distinguishable from edges connecting higher degree nodes.

3.3.7 Edge-Bundling: Higher Degree

Again, this mode is basically the same then the previous one, but there is a distinction though. Whereas the mode described in Section 3.3.6 uses both degrees to scale the control points of the splines farther to the inside or outside, this mode uses just one of this degrees: the higher one.

Doing the same calculations for the nodes as listed before in Section 3.3.6, this mode adds another short equation to the code right before calculating the control points $CPA_{x,y}$ and $CPB_{x,y}$ for the points A and B.

$$dist_A = dist_B = min(dist_A, dist_B) \tag{3.18}$$

Equation 3.18 forces the edges to use the higher degree value of both nodes A and B, since a higher degree value leads to a smaller value for $dist_A$ or $dist_B$ when solving

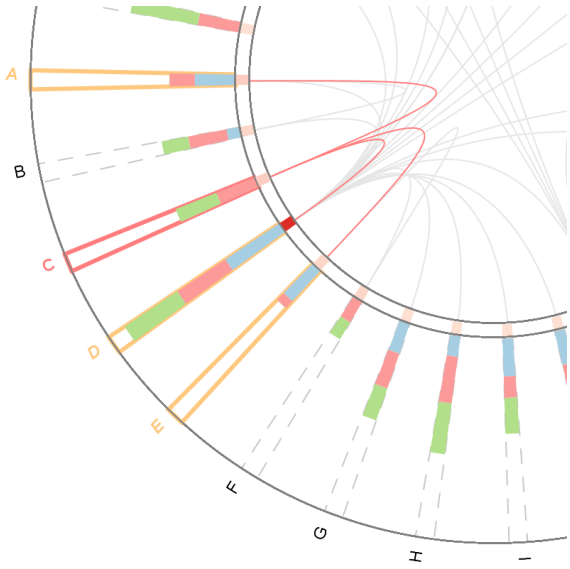


Figure 3.9: Results for the mode 'Both degrees'

the Equations 3.11 and 3.12. Figure 3.10(a) shows the results of this mode. Since there is again no neighborhood information taken into account and only the higher degree of both nodes is used, the edges are drawn far more to the center of the circle. Still, edges connecting a high-degree node bundle, but the bundling is done closer to the center of the circle.

3.3.8 Edge-Bundling: No Rating

The last mode to be described is the mode for 'No rating'. This means, that absolutely no rating concerning the connection degrees, the nodes neighborhood or something else is done. All the edges are drawn in the same style, no matter if the nodes are high or low connected.

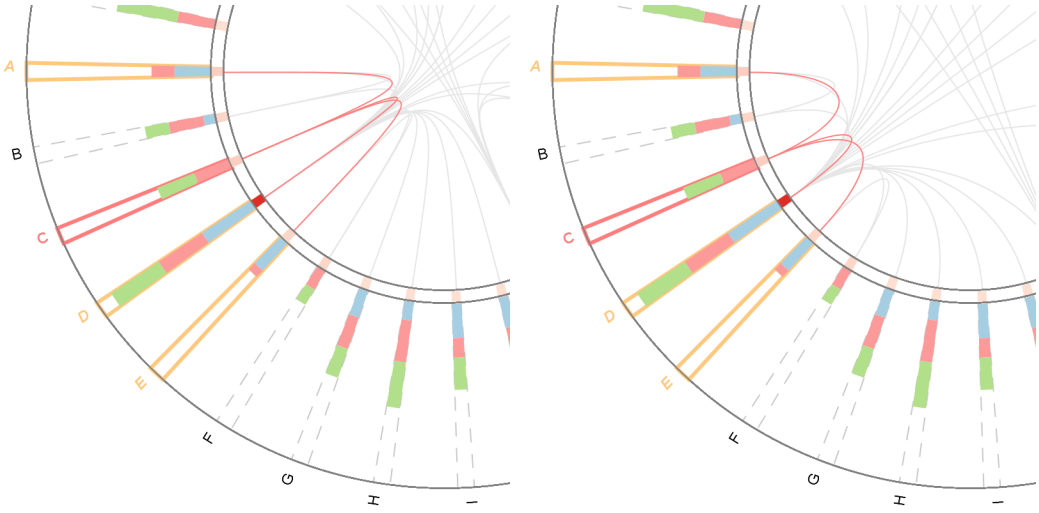
The calculation for this mode is very simple. Using the same equations for calculating the control points, the algorithm just calculates the angles for the nodes A and B as already shown in Equation 3.8, but then uses fixed values for $dist_A$ and $dist_B$.

$$dist_A = r/2 \tag{3.19}$$
$$dist_B = r/2 \tag{3.20}$$

All the control points are set exactly in the middle of the connecting line from the center to the node, therefore leading to well-formed arcs. Figure 3.10(b) shows the outcome of this rendering mode. Connections on the same side of the circle stay on the outside and connections to the opposite side do not bundle that strong. Bundling is therefore done right in front of each node. Figure 3.11 again shows all the bundling modes in comparison, displaying the same network and the same highlighted nodes all the time.

3.4 Filtering Features

Having a closer look at the drawn network and the corresponding centrality values often requires filtering of data. Since the data behind the network is multidimensional, a different approach to filter out certain nodes has to be introduced than usual filtering according to nodes itself. Using the same approach than the *Attribute Explorer* [TSWB94], the tool



(a) Results for the mode 'Higher degree' (b) Results for the mode 'No Rating'

Figure 3.10: Bundling modes 'Higher degree' and 'No Rating'.

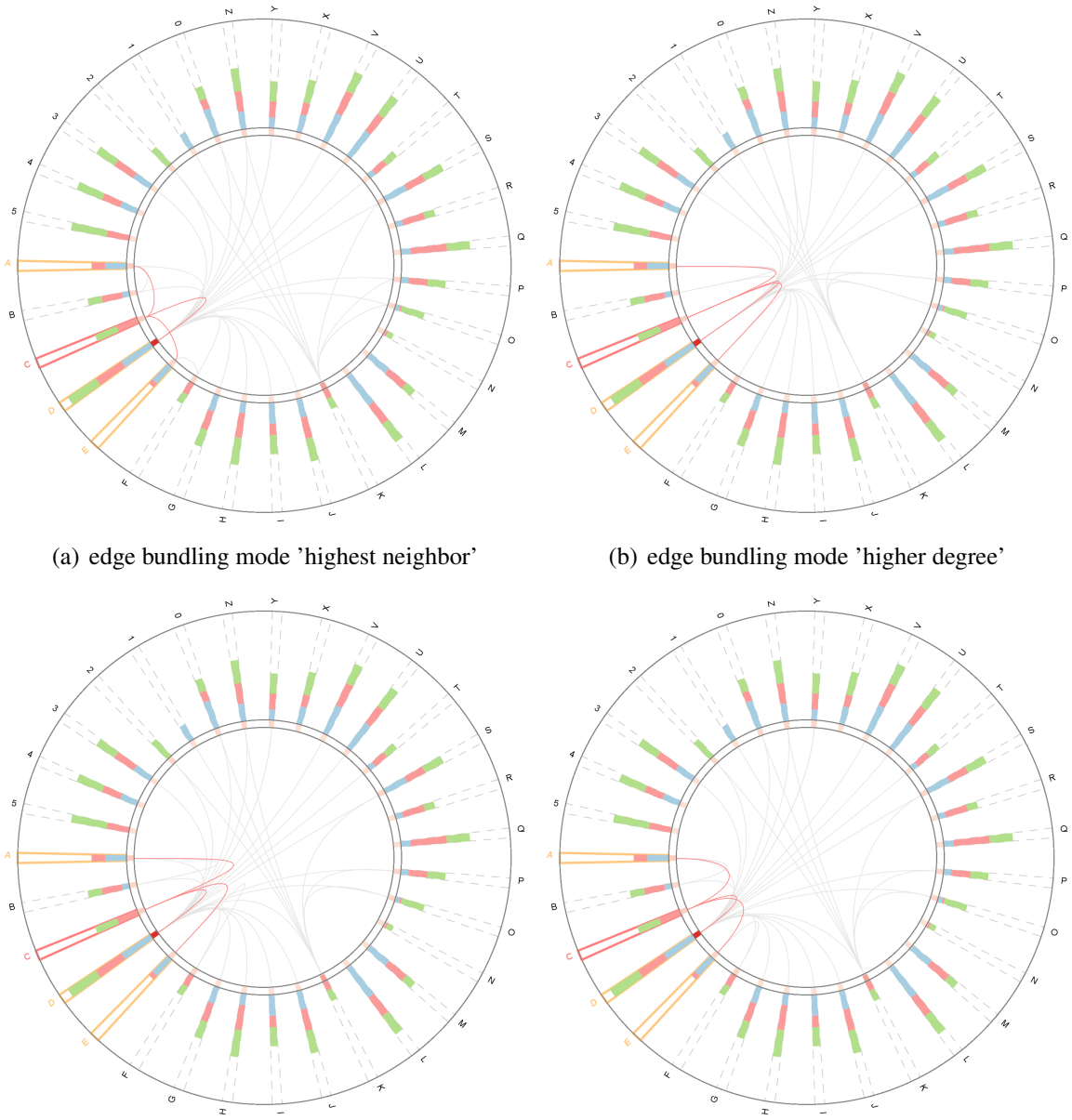
maps each attribute of a node (its centrality values) to a single dimensional representation: a histogram.
Using the representation of a histogram furthermore gives clear advantage when it comes to the exploration of the dataset itself. Shown to the right of the actual rendering (see Figure 3.5, green part), the histogram view provides information like:

- The distribution of the underlying data in the dataset.
- Statistical information such as mean, min- and max-values.
- The skewness of the data can be seen.
- Wholes in the distribution can be seen easily.

The following sections describe the filter features of the tool according to the histogram data. To make things more clear, a sample dataset for a social network is used. The original sample data (*socialnet.xml*) is included into the *prefuse* [HCL05] visualization toolkit. For the purpose of this filtering samples some nodes were removed from the set. Figure 3.12 shows the network.

There are several very high connected nodes in the network (e.g. *Jeff, Alan, Chris, Ben, Fernando*). When having a look at the circular rendering, those values are marked with a dark red color for the degree marking and the edge bundling shows, that a lot of edges bundle in front of them, therefore showing high connectivity as well. Looking at the graph view, the same argumentation can be done. *Jeff, Alan, Chris, Ben, and Fernando* have a lot of friends who are not interconnected, therefore marking them self out as actors in the network. The centrality values like 'Eccentricity' will be more likely higher for those nodes than for other nodes.

This fact will now be used to show the filtering processes based on the histograms of the centrality values. In general, filtering out nodes means filtering out the corresponding edges in the circular view as well, to clear up the center of the visualization and therefore, reducing visual clutter.



(a) edge bundling mode 'highest neighbor'

(b) edge bundling mode 'higher degree'

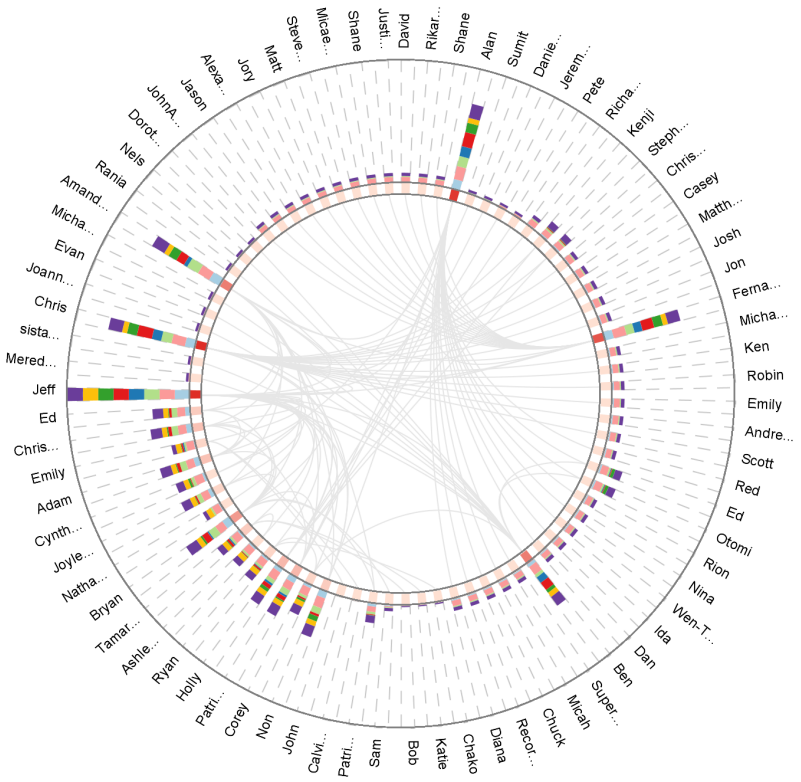
(c) edge bundling mode 'both degrees'

(d) no rating at all for the edges

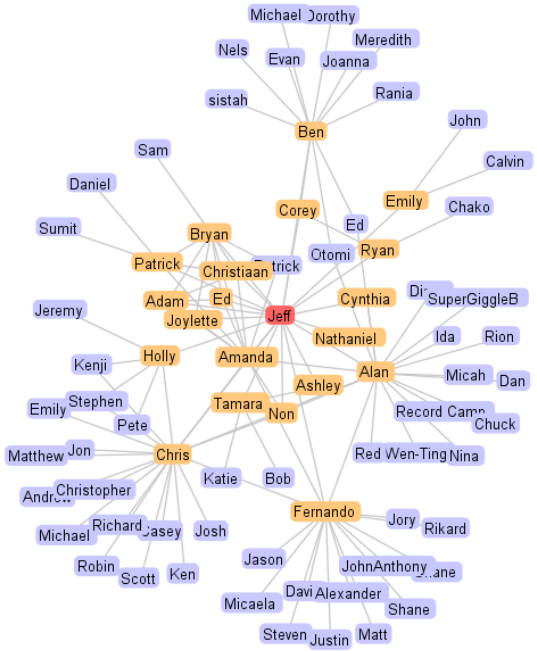
Figure 3.11: Differences in between all the edge-bundling modes offered so far by **ViNCent**. Clearly, Figure 3.11(a) is an advantage when it comes to edge bundling, since edges between low connected nodes stay on the outside, therefore not cluttering the inside of the circle.

3.4.1 Calculation of Histograms

In order to be able to show histograms, the current dataset has to be analyzed. The calculation of the histograms is based on a certain amount of bins, the data values are stored in. Let C be the set of centralities in the current visualization and c_k define the centrality k in the set. $min(c_k)$ is defined as the minimum value of the centrality k and $max(c_k)$ as the maximum value.



(a) the circular rendering of the network



(b) the graph rendering of the network

Figure 3.12: Social Network sample data from the *prefuse* toolkit (reduced).

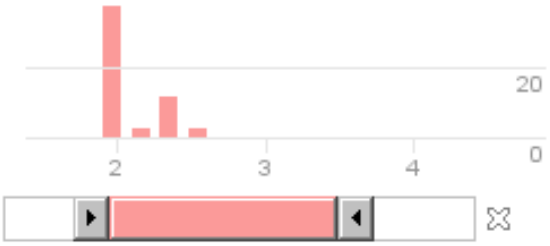


Figure 3.13: 'Radiality' range slider filter

Let B be the set of bins in the system and $||B||$ define the number of bins, then the width of one bin ($binWidth$) calculates as:

$$binWidth = \frac{max(c_k) - min(c_k)}{||B||} \tag{3.21}$$

The dataset is now divided into $||B||$ bins, reaching from $min(c_k)$ to $max(c_k)$ whereas each bin b_l is defined as:

$$b_l = [min(c_k) + l * binWidth, min(c_k) + l * binWidth + binWidth[, l \in [0, ||B||[\tag{3.22}$$

Having defined the set of bins and knowing, which value falls into which bin, the histograms can be rendered and displayed. Each histogram then renders the values of one of the centralities in the network. Therefore, a filter applied on a certain histogram affects just nodes, having values in the corresponding centrality falling into that range. The following sections deal with the filtering process based on the available histograms.

3.4.2 Filtering Based on Histogram Bars

There are several possibilities to filter out data from the dataset, thus hiding the values in the circular rendering for providing better overview. One possibility is it, to filter out elements belonging to one bar in a histogram.

By simply clicking the bar, the corresponding elements in the circular view are then hidden. The bar in the histogram is therefore marked in another color, showing that this bar has been filtered out (see Figure 3.14(c)). Figure 3.15 shows the results of this filtering process.

3.4.3 Filtering Based on Histogram Sliders

Another way of filtering out elements is by using the range sliders below each histogram filter. Figure 3.13 shows an example of the range sliders for the centrality 'Radiality'. By simply sliding from the left or right, the amount of displayed nodes decreases corresponding to the filtered out elements in the dataset. This method is useful to quickly filter out minimum and maximum values of histograms.

3.4.4 Filter Based on Single Nodes

Since the before presented filters act on the whole set, they affect more nodes at once. But to filter out just single nodes from the view, the last filter possibility can be useful. This filter is based on single nodes. By right-clicking nodes in the circular view, the disappear from the view. Right-clicking them again brings them up again.

3.4.5 Hiding Centralities from Circular View

Sometimes, certain centrality values - and therefore the whole centrality - have no further meaning for the exploration of the dataset, since all of the nodes have the same values in that centrality. In this case, the centrality can be hidden from the renderings by simply deselecting the check-box for the visibility of the centrality.

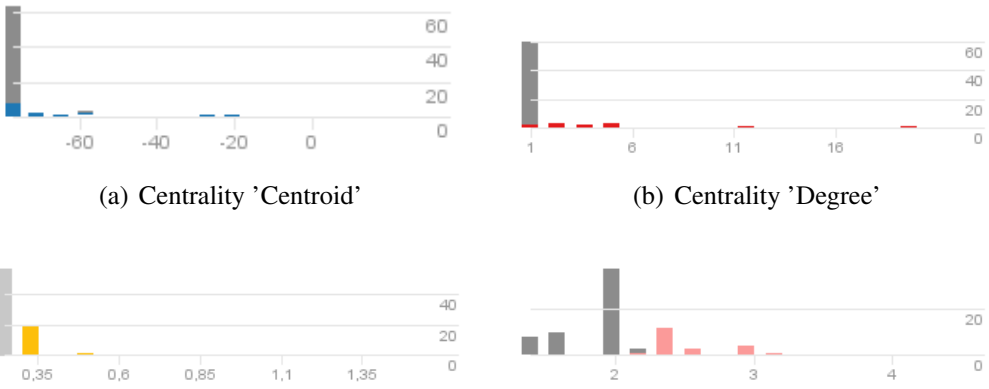
Having active filters on a histogram belonging to a hidden centrality means, that the filters are then deactivated. They become activated again, when the centrality is shown again.

3.4.6 Filter Propagation

Filtering out elements by using the various methods as described before affects directly the circular view. Therefore, the user has an immediate feedback to the actions. But to better keep track of already filtered out elements, they are marked in the histograms as light gray bars, when the filter is applied directly on this histogram, and bars are marked dark gray, when the filter spreads from another active filter on a different histogram.

This filter propagation gives the user better feedback of how certain centrality values are related to each other. The *Attribute Explorer* [TSWB94] already uses this approach.

To better explain this feature, Figure 3.14 shows it in more detail. The user filters out nodes corresponding to the first bar of the 'Eccentricity' centrality histogram (shown in Figure 3.14(c)). Therefore, the desired bar is marked with a light gray color. Since there are now about 55 nodes hidden in the circular view, the filter action has to spread to the other centrality histograms as well. Having a closer look at the 'Radiality' histogram (Figure 3.14(d)) one can see that there are about 55 elements marked in dark gray, meaning, that they are already filtered out by an active filter on another histogram. Thus, this filter propagation provides the user with a better feedback of how centrality values are related to each other. Filtering out low values in the 'Eccentricity' histogram for example filters also out lower values in the 'Radiality' histogram.



(a) Centrality 'Centroid'

(b) Centrality 'Degree'

(c) Centrality 'Eccentricity'

(d) Centrality 'Radiality'

Figure 3.14: Histograms for four different centralities. Figure 3.14(c) has an active filter on the first bar, shown in a light-gray. This active filter spreads to the other histograms shown in dark-gray (see e.g. 3.14(a)).

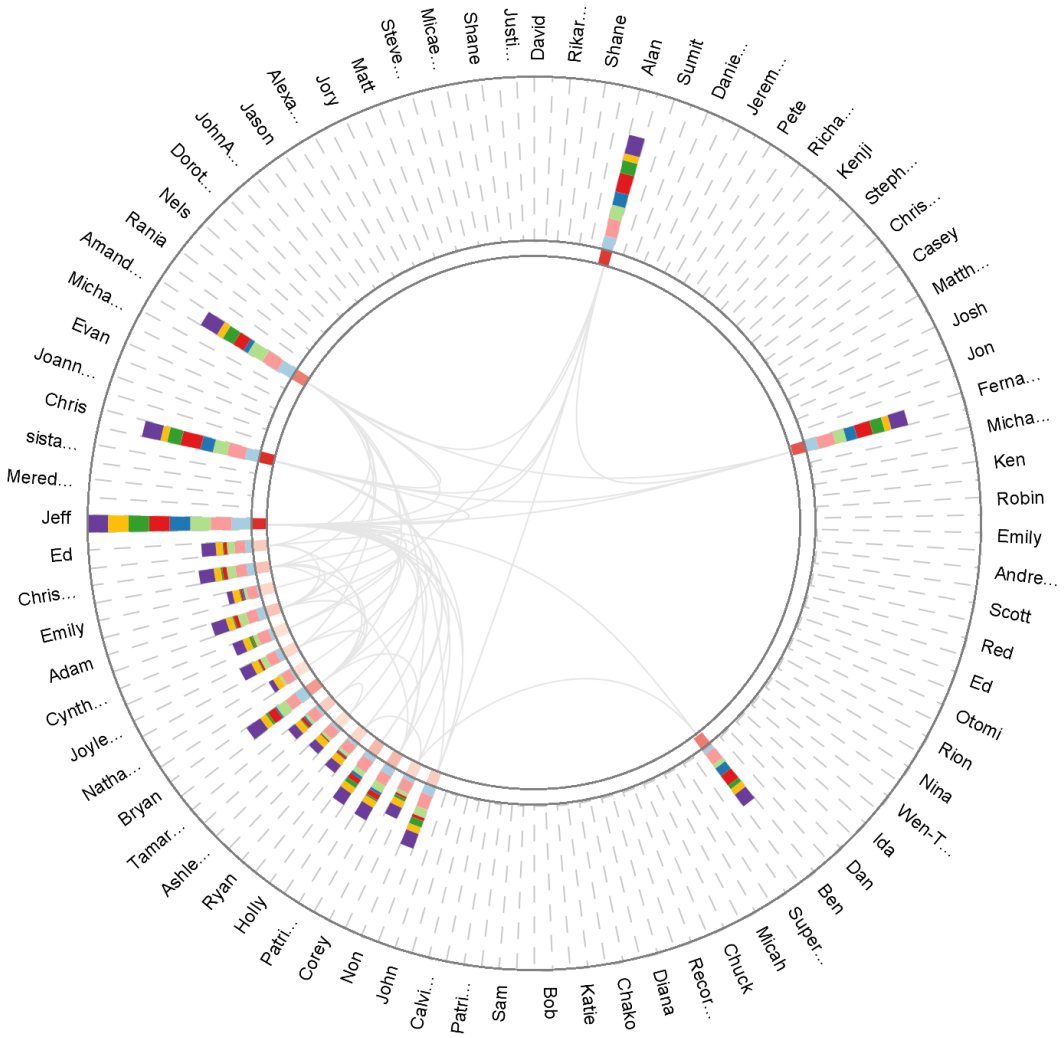


Figure 3.15: Active filter on the network: nodes have been filtered out by using the histograms. The active 'eccentricity' histogram bar filter filtered out all nodes having a low value in this centrality. As Figure 3.14(c) shows, about 55 nodes are affected by this active filter.

3.5 Hovering Features

To provide a good feedback to the user and enhance the interaction capabilities of the tool, hovering features are introduced to highlight elements and show cross-connections. This concept of *Revealing Relationships Amongst Visualizations* was already discussed by Christopher Collins and Sheelagh Carpendale. It helps the user to interactively explore the dataset since techniques like spreading activation or search filters are applied [CC07].

3.5.1 Hovering in the Circle View

The main focus of the user is usually on the circle view in the middle, since there the network including the centrality values is rendered. Hovering nodes in this window leads to an activation of the connected nodes and the connecting edges. Figure 3.11 for example shows this hovering and highlighting process.

The current active and hovered node is highlighted with a red surrounding spreading out the red highlighting via the connecting edges to the direct neighbors of the node. Those are then highlighted with an orange surrounding. In addition, labels are colored in

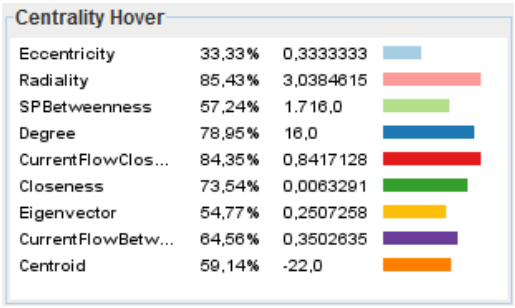


Figure 3.16: Hover window view

the same manner and rendered bold for this process.

In addition to the highlighting done in the circle view, the hover window (located in the lower right corner of the tool, see Figure 3.5, orange part) becomes filled as well. Figure 3.16 shows this hover window displaying data of the current hovered node. There, the single stacked bars from the centrality bar are rendered again, but showing more detailed information about this node's centrality values.

Each centrality value is rendered in there with its name and the relative percentage, which means the relative percentage within this centrality over all nodes. After the percentage, the absolute value is given followed by a bar indicating the relative percentage again as bar. This provides again a quick overview of the centralities of one node and furthermore allows the user to have a closer look at the absolute values too.

3.5.2 Hovering in the Histogram View

To also provide feedback in the histogram views, hovering bars in the histograms spreads to the corresponding nodes in the circle view as well, thus enabling the user to check in more detail, which nodes are within this bin of the histogram and would be affected by filtering them out.

3.6 Various Features

Despite the before described features of **ViNCent** the tool offers several others. The following short sections give an overview.

3.6.1 Settings

Besides the automatically applied settings for the arrangement of the nodes and the bar renderings, the user can control several other settings for the rendering by the use of the settings panel, which is located to the left of the actual rendering (see Figure 3.5, blue part). This panel offers the following possibilities:

- **Inner Radius**
  The user can control the inner radius of the rendering in percent of the actual visualization size. Default setting is set to 25%. The range for the setting is from 5% to 35%.

- **Outer Radius**
  Same as for the inner circle, the user can control the radius of the outer circle as well. Default setting is set to 45%. The range for the setting is from 35.1% to 47%.

- **Edge Bundling**
  Here the user can select the different edge bundling modes as described in Sections 3.3.5 - 3.3.8.

- **High quality rendering**
  Forces the display to use anti-aliasing for the renderings. The visualization looks much nicer and smoother then, but is much slower than normal.

- **Show degree marking**
  Indicates, if the inner circle with the degree marking is shown or not. Hiding this saves same rendering space.

- **Render labels**
  By default in checked state. Indicates if labels are rendered on the visualization or not.

- **Bin count for histograms**
  By the use of this slider, the user can control how many bins exist for the histograms. The default setting is set to 15 whereas it can range from 2 up to 100.

3.6.2 Exporting Screenshots

To take screenshots, the tool offers the opportunity to export the current displayed visualization including all the highlights to a file. The export then can even be scaled, therefore providing the possibility of a high resolution screenshot not relying on the actual screen resolution.

For the main visualization (the circle view), this export can always be done by either clicking the menu button *'File - Export display ...'* or pressing the shortcut *Ctrl + E*. The user can then scale and export the screen to one of the following formats: `PNG`, `JPG`, `BMP`, and `GIF`. To export the histogram views, the user has to right click the view and select *'Export display ...'* from the context menu in order to export the histograms as well.

3.7 Interaction-Concepts in ViNCent

So far the description of the single features of the tool already named out some interaction concept **ViNCent** uses in order to provide a good interaction and usability for the user. This section deals with the interaction concepts in more detail and discusses, how they help the user to keep track of the displayed data.

3.7.1 Multiple Views

In order to explore the displayed dataset, **ViNCent** provides various views onto the same amount of data. Those views are the following tow:

1. The circular arrangement of the nodes shows the nodes, its centrality values and the network itself, therefore providing a big overview of the complexity of the network allowing the user to get the main actors at a first glance.

2. The single histogram views again provide a view to the same amount of data, but in a different representation. Not rendering as networks but as distributions of centrality values, those views help the user to understand the underlying data in a more statistical way.

Not necessarily needed for the structural analysis of networks, the histogram views are needed when it comes to effective exploration of centrality values. Furthermore, those views provide the user with the possibility of doing further interactions such as hovering bars and filtering out data based on the distribution of centrality values.

3.7.2 Linking and Brushing

ViNCent makes extensive use of *linking and brushing* features to link together certain amount of data in the visualization just by highlighting it in different visualizations. The following interactions in the tool use this feature to highlight data for the user:

- **Hovering in the circle**
  This leads to highlighting of the neighborhood of the current hovered node including the connecting edges. Moreover, the hover window itself shows the hovered node data in more detail.

- **Hovering bars in the histograms**
  To show the user which nodes are falling into which certain range of data values according to one centrality measure, hovering a bar in the histogram views highlights the corresponding nodes in the circular view as well.

- **Filter spreading across histograms**
  Filtering out bars in the histograms means filtering out nodes in the view. Based on this applied filter, the underlying dataset changes, since certain nodes are not present anymore. This is reflected in the other histogram views by marking already filtered out nodes as gray in the corresponding bars.

Those described features are the most important ones in order to be able to analyze the network structure as well as being able to keep track which nodes are filtered out by which action based on histograms.

3.7.3 Filterings and Dynamic Queries

Exploration of big datasets is not possible without filtering of data and dynamic applied queries. **ViNCent** uses several approaches to allow the user to filter out data, therefore shrinking the dataset to a certain amount of nodes, to explore it in a better way.

- Nodes can be filtered out in the circular view directly by right-clicking them. This leads to filter spreading across the histogram views as well.

- Nodes can be filtered out by the use of the histogram view bars. Clicking bars filters out all the corresponding nodes falling into that range.

- Dynamic queries can be applied by the use of the range sliders below each histogram view, therefore making it easy for the user to filter out top and bottom values of distributions.

Used to limit the amount of displayed data, filters support the user in fulfilling his tasks more efficiently. A basic principle when it comes to filtering and dynamic queries is the response of the interface. Actions should be visualized within a short amount of time (e.g. one second) in order to be able to associate user interface actions and the corresponding filtered out nodes.

3.7.4 Colors

Human beings are good in distinguishing colors. Sahler [IS05] defines color codings as the following:

> "The typical human eye can distinguish over 10,000 different colors. We use color everyday to identify, classify and determine the state of objects around us." [IS05]

Therefore, **ViNCent** uses a color scheme introduced by the *ColorBrewer* [Bre0x] to code the different centrality values with different color representations. This allows the user to easily separate different centrality values from each other. Using the same color scheme in all the different views (circular arrangement, histograms and hovering window) is important to speed up the human perception.

3.7.5 Details on Demand: Hovering

To not overload the interface and to not provide the user with too much data at once, only the necessary amount of data is shown in order to be able to navigate the network fast. On demand, the user can get more (and detailed) information about certain nodes by hovering them. More relevant data is then shown in the hover window to provide the user with the exact values of centralities too.

3.8 Plug-in: CentiBiN

ViNCent is actually not limited to a certain field of application when dealing with centrality values. It can handle every single network, as long as it is represented in a *GraphML* format and the centrality values in the file are in a numerical representation.

To support more fields of applications - especially the biological network analysis - **ViNCent** uses the **CentiBiN** plug-in developed by Junker et al. [JKS06b] to calculate network centralities for biological networks on graphs, loaded as plain *GraphML*.

Opening such a file, the user is then asked to provide information about which attribute in the *GraphML* file to use for the node labels and which of the possible centrality values to calculate for the given graph. So far, the **CentiBiN** plug-in and therefore **ViNCent** can calculate the centralities on directed and undirected graphs, shown in Table 3.2.

Not every centrality measure can be applied to every graph since there are preconditions a graph has to fulfill in order to calculate the values. These can be simplicity, connectedness, and loop-freeness [JKS06b].

Once **ViNCent** has calculated the values for the graph, the network is displayed. Those calculated values can also be saved by exporting the current graph including the values to a *GraphML* file. More about this is described in the following section.

3.9 Import and Export Data

ViNCent can deal with any kind of network represented as *GraphML* file, as mentioned above. The *GraphML* standard is an open standard defining how a graph is represented as XML structure. The *GraphML* [BEK$^+$0x] format furthermore defines which attributes to add to nodes and edges in the graph. This feature is used by the tool to load and save calculated centrality values, therefore allowing every network with centrality values to be loaded, as long as it is in a *GraphML* format.

A saved file by **ViNCent** may look like the following example. The *GraphML* file starts by introducing *key* elements, which are used to define attributes for nodes and edges. In

Centrality	Type	Directed graphs	Undirected graphs
Degree	Neighborhood based	X	X
Eccentricity	Distance based	X	X
Closeness	Distance based	X	X
Radiality	Distance based	X	X
Centroid	Distance based	X	X
Stress	Shortest-Path based	X	X
S.-P. Betweenness	Shortest-Path based	X	X
C.-F. Betweenness	Current-flow based		X
C.-F. Closeness	Current-flow based		X
Katz Status	Feedback based	X	X
Eigenvector	Feedback based	X	X
Hubbell index	Feedback based	X	X
Bargaining	Feedback based	X	X
PageRank	Feedback based	X	X
HITS-Hubs	Feedback based	X	X
HITS-Auths	Feedback based	X	X
Closeness-vitality	Vitality based	X	X

Table 3.2: Centrality measures implemented in CentiBiN. (taken from [JKS06b], Table 2)

this sample, nodes are named using the data-key *name*. Centrality values for the 'Centroid' centrality are stored in the data-key *Centroid*.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
         http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">

  <!-- defining attributes for nodes -->
  <key id="name" for="node"
       attr.name="name" attr.type="string"/>
  <key id="Centroid" for="node"
       attr.name="Centroid" attr.type="double">
    <default>0.0</default>
  </key>

  <graph edgedefault="undirected">
    <!-- nodes -->
    <node id="0">
      <data key="name">n0</data>
      <data key="Centroid">-15.0</data>
    </node>
    <node id="1">
      <data key="name">n1</data>
      <data key="Centroid">-23.0</data>
    </node>
     <!-- edges -->
```

```
    <edge id="0" source="0" target="1">
  </graph>
</graphml>
```

3.9.1 Importing Data

When importing data, **ViNCent** offers two possibilities to the user:

- **Loading a standard *GraphML* file**
  This option allows the user to load a standard *GraphML* file into the application. The tool later on asks for a field in the *GraphML*, which should be used as label for the nodes. Furthermore, the tool lists the possible centrality values it can calculated based on the *CentiBiN* plug-in. After selecting the desired values, the tool calculates those values and renders the network.

- **Loading a *ViNCent GraphML* file**
  When loading a *ViNCent GraphML* file, the tool assumes that the centrality values are already calculated. Therefore, the user is asked for the label field and then the desired centrality fields, already present in the file, to display.

3.9.2 Exporting Data

ViNCent can export all graphs shown including the calculated centrality values to *GraphML* files, thus allowing the user to share networks with other users. The written files are in the format as beforehand shown.

4 Discussion

The field of information visualization concerning network centralities has become more and more important over the last years, since scientists can gather a lot of information out of networks by the use of centrality measures. The introduced tool **ViNCent** combines different approaches to overcome the mentioned difficulties when visualizing network centralities and is making them comparable.

It comes up with a complete new way of visualizing centrality values within a network by the use of the circular arrangement, therefore minimizing visual clutter and occlusions of nodes. Nodes and their centrality values can always be read by the user. Furthermore, the usage of stacked bars to the outside of the nodes give the user the possibility to see important nodes—therefore having high centrality values—at a first glance, since those nodes lead to big stacked bars. Moreover, the additional visual hints for the degree marking in the middle of the circle support this approach.

By displaying the nodes and bars to the outside, therefore providing the user all the time the same view onto a network, the cognitive load is reduced. The changing parts in the middle—the actual structure of the network represented as connecting edges between nodes—are arranged in a way, that they are not disturbing to the allover view but furthermore enhance the visual appearance of the rendering by supporting the user to find high connected nodes within the network.

A feature of **ViNCent** which is improving the concept of exploration is the way of filtering out data. By the use of histograms and therefore the underlying distributions of network centrality values, the user can get an overview real quick and filter out unimportant data with a few clicks. The concept of supporting bar filtering, range-sliders and single node filtering allows the user to filter out any kind of combination of nodes, therefore leading to less visual clutter in the actual rendering.

The interaction with the tool especially the brushing and linking features for visual linking parts of the renderings together supports the exploration of a network and provides the user with a good feedback of which nodes belong together, which are interconnected and which belong to a certain range when it comes to a histogram bin. The possibility to show the actual graph which is shown as circular arrangement gives even more feedback about how the network looks like.

Before looking on the benefits and drawbacks of **ViNCent**, a short use case scenario in the following section should show, how the tool can be used to analyze a social network.

4.1 Use Case Scenario

By the use an example social network the application of **ViNCent** should be shown now. Therefore the dataset of the "Padgett's Florentine families" is used [WF94]. It shows the marital relations of 16 families in the 15th century in Florence, whereas each edge corresponds to a marriage between the families.

Aim of this short use case scenario is now to find out which families were the most important ones in Florence at that time. This task is achieved by the use of **ViNCent** and the corresponding centrality values of the network.

As figured out already before, centrality values such as *degree* and *betweenness* are usually used to find out the importance of nodes in a social network. The *degree* value indicates connections which is related to the marriages in this case. The higher the degree value, the more important the family is. The *betweenness* centrality however is based on the shortest path, therefore relying on the overall network structure and topology. High values in this centrality indicate *central actors* in the network linking together whole

groups (communities). In this current example, a high value in the *betweenness* would mean that this family has a lot of influence by marriages across several communities, therefore linking together families to big groups.

Figure 4.1 shows the example dataset and the corresponding results for the task. Figure 4.1(a) presents the total dataset of the marriages in Florence in the 15th century. Just by having a quick glance one can see, that the family of the 'Medici' plays an important role, since it has the highest values in all the centrality values.

To now figure out the top three families, one can take the *betweenness* centralities and *degree* centralities into account. First of all, nodes which are not connected at all are filtered out by the use of the *centroid* centrality, since this one is a distance based measure. In this first step, the node of the 'Pucci' family falls out, since it is not connected at all.

The next step uses the *betweenness* (in this case, the *SPBetweenness*) range slider filter to filter out the nodes having a small value. Sliding the bar from the left to the right until only three families remain shows too the correlation between the *SPBetweenness* and *CurrentFlowBetweenness*, since both filter out items from the left (see the *CurrentFlowBetweenness* histogram in Figure 4.1(b); the first few bars are marked dark-gray, indicating that those node are already filtered out by another applied filter).

The now remaining three nodes in the view are the families having the most influence in Florence at that time. Clearly the 'Medici' are first. Rating the second and third position according to the *SPBetweenness* centrality, the 'Guadagni' family places second and the family of 'Albizzi' third.

This small sample shows an application of centrality measures in social networks and how **ViNCent** can solve this problems by visualizing the networks.

4.2 Benefits of ViNCent

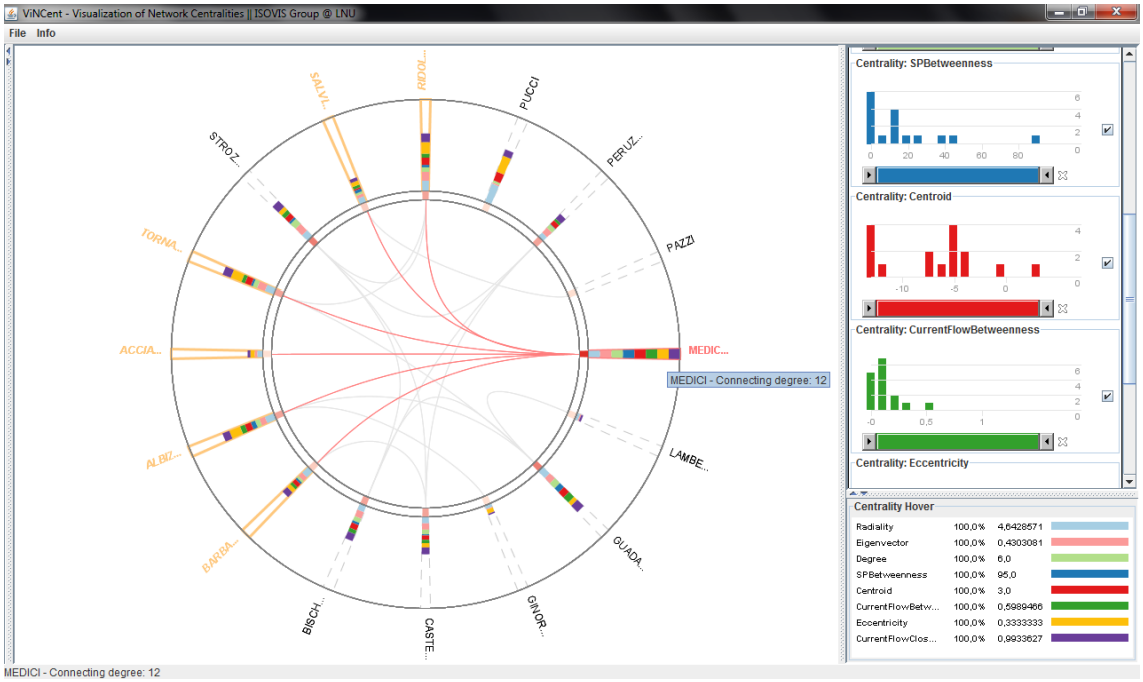
Advantages about **ViNCent** are the circular arrangement of nodes which leads to less visual clutter and allows the visualization of many nodes up to a few hundred (limited by the actual screen-size). Other advantages are the filter possibilities, which speed up the exploration of networks and support the user in filtering out less important data at a first glance. The previous example showed how easy filters can be applied and how powerful they are when it comes to shrink down the amount of data.

4.3 Drawbacks of ViNCent

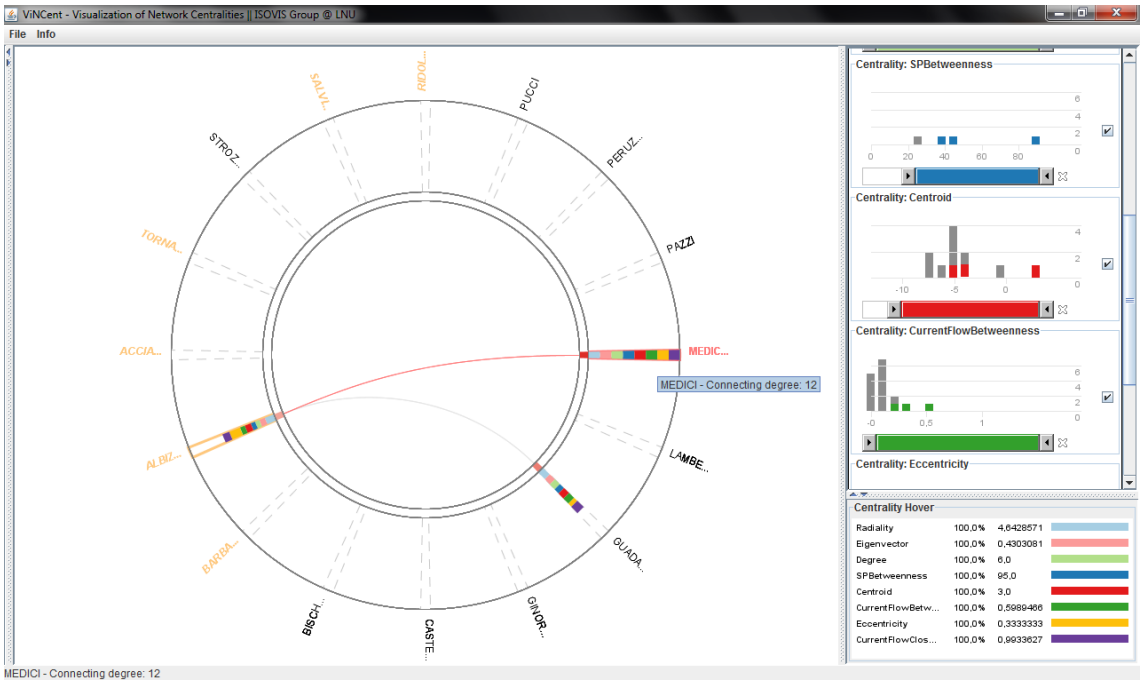
Current drawbacks in **ViNCent** are the lack of linking and brushing between histograms when hovering single bars (just already filtered out data is linked and brushed) and the missing visual links from the circular view to the histogram windows (so far, just hovering in histograms hovers nodes in the circle).

Especially the last feature (hovering nodes in the circle shows the location of the node in the histogram) would enhance the filter possibilities even more.

Another big drawback is the performance. When dealing with a lot of data, the performance of the tool is not sufficient to keep up the full interaction possibilities. Linking and brushing features are then lacking of immediate feedback, therefore leading to a bad user experience and bad results for filtering tasks.



(a) Sample, showing the social network of the Padgett dataset



(b) Solved task using the **ViNCent** tool. The top three families in Florence are found.

Figure 4.1: Use-case scenario of **ViNCent**. Filtering out the top three families in Florence in the 15th century.

5 Conclusion and Future Work

ViNCent represents a tool which can be used for any kind of network exploration and any kind of centrality visualization. It was actually build as a tool for biological network centralities, but can be used to explore any kind of networks, as long as nodes have attributes represented as numerical values. Thus, **ViNCent** is a powerful tool for gaining more knowledge about the inner structure of networks.

Compared to other tools, **ViNCent** is better capable of visualizing centrality values for nodes, since it provides a direct visual feedback by the use of stacked bars. This is a clear advantage when it comes to fast exploration of networks. It still reveals the network structure and makes it therefore possible to follow pathways in a network too.

The tool scales for a few nodes up to a few hundred and even more nodes, depending on the screen-size, which is available, since the size of the inner circle limits the number of nodes. There are are few features though, which are not yet implemented, but would enhance the visualization even more. The following paragraphs discuss the possible future work on the tool as well as name out some further improvements.

Edge-Bundling Modes

The most important possible future features are new edge-bundling modes, since they would lead to less visual clutter in the center of the circles, thus providing the user with a better overview of the network structure and pathways. There is a lot of ongoing work for edge bundling in graphs, however, those approaches cannot be directly applied to the tool because of the lack of certain preconditions. Danny Holten introduces two edge bundling modes, which reduce visual clutter in a good way: *Hierarchical-based edge bundling* and *force-based edge bundling* are described in his works [Hol06, HvW09].

Hierarchical-Based-Edge-Bundling (HBEB)

Based on the inner hierarchy of a network, this bundling mode draws together connections between nodes, following the same inner structure (pathways). Furthermore, frequently followed pathways are highlighted with thicker and different colored lines.

However, this feature cannot be applied directly to the **ViNCent** tool, since the arrangement of all the nodes of a network along the circle remove the inner structure (hierarchy) from the center. Also a mirroring of the structure to the inside or the use of spanning trees is not possible.

To introduce this feature, more research has to be done in order to figure out, how a hierarchy can be build up to support the bundling along common pathways.

Force-Based-Edge-Bundling (FBEB)

Another approach presented by Danny Holten is the approach of simulating forces between edges, therefore either sticking together or pushing each other farther away. It is closely related to the approach of force-directed graph layouts.

Because of the fact this approach does not rely on any further structure in the middle of the circle, it would be applicable to solve the problem of edge bundling for **ViNCent**. Since the computation of bundling edges so far is done on one single point, a further bundling mode like this could be easily introduced to the system.

Arrangement of Nodes on the Circle

Closely related to the problem of edge bundling is the arrangement of nodes on the circle itself. Depending on the positions of nodes, edge bundling (as already introduced in the

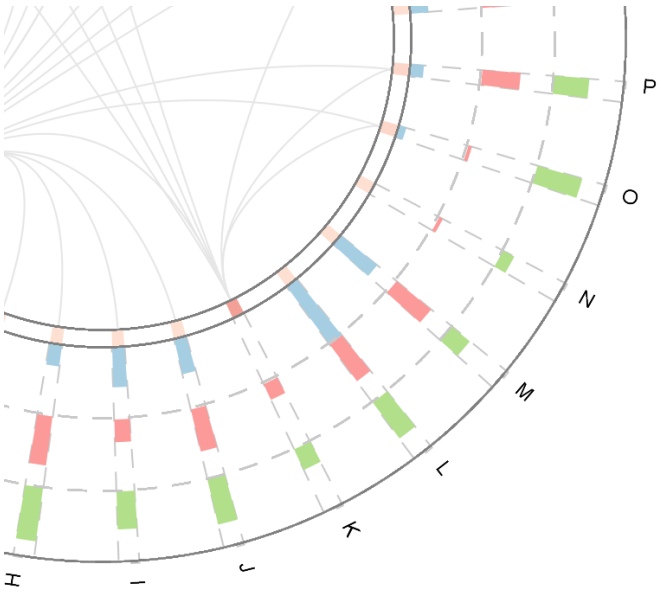


Figure 5.1: Arrangement of Bars in the Circle: improvements

tool or force-based approaches as well) may produce better results.

For this problem, one could take into account the information out of the network analysis, like information concerning cliques or communities, single actors and density. This information is useful to decide how nodes should be arranged along the circle, since out of this, the possible amount of edges crossing each other could be calculated. However, this is so far still in research and not implemented yet.

Arrangement of Bars in the Circle

After a first review of the tool by *Falk Schreiber* (see [DHK+06]), he suggested a new way of arranging the bars on the circle. Rather than just to stack the bars of a single node immediately together, he would suggest to stack bars always onto the maximum value of a certain centrality. This leads to better comparability of values. Figure 5.1 shows a schematic drawing of this feature. Node *L* has the maximum value of the blue centrality. Therefore, this node defines the border (starting point for the new red centrality) for all the nodes. The red centrality value is then stacked onto this virtual border for all other nodes.

This feature makes it easier to compare centrality values in a visual form in the rendering itself. A first glance at the circles provide the user with a rough overview of the relative values of the nodes.

Highlight Spreading Based on Setting

In case of highlighting actions in the tool **ViNCent** so far highlights the current hovered node and the immediate neighborhood. This gives information about the direct connections. The possibility of a setting for a highlight spreading could be useful though. Configurable, this highlight spreading could spread for a certain amount of levels, getting weaker and weaker on each level, therefore too showing the level of neighborhood.

Plug-in System for Further Data Sets

As shown in Figure 3.2, **ViNCent** uses a factory design to choose the correct class for loading up data. To ensure flexibility, a plug-in system for loading further data sets could be introduced in order to allow researchers to use their own data format.

Since there is already a base-class structure for doing this, only the *DataImporterFactory* has to be touched to enable dynamic class loading from *jar* files.

Plug-in System for VANTED

VANTED presented by Junker et al. [JKS06a] is a system for advanced data analysis and visualization in the context of biological networks. Since they offer a plug-in system for various other software parts, **ViNCent** could implement this plug-in in order to be used in *VANTED*, therefore providing the possibility to directly use it the context of biological network analysis without exporting and importing data.

Performance Issues

So far, **ViNCent** uses the *prefuse* toolkit to render the visualizations. Since this toolkit relies on *Java2D* instead of faster *Java OpenGL* implementations, the tool has some performance issues. Especially when rendering many separate objects and using high-quality rendering mode, the performance goes down.

A possible restructuring of the code might help in the first row, to avoid unnecessary round trips in code and renderings, but still, changing to a faster visualization toolkit may lead to better results.

Test with a Biological Network

Based on the so far executed tests while developing the tool, it has clear benefits when it comes to the analysis of networks according to their centrality values. However, the tool should be tested with a biological network and a real use-case scenario as well to clearly figure out problems in interaction and visualization techniques.

References

[BEK+0x] Ulrik Brandes, Markus Eiglsperger, Michael Kaufmann, Jürgen Lerner, and Christian Pich. *GraphML Specification*. http://graphml.graphdrawing.org, last accessed: 2010-12-04, 200x.

[Bre0x] Cynthia A Brewer. *ColorBrewer*. http://www.ColorBrewer.org, last accessed: 2010-12-02, 2.0 edition, 200x.

[CC07] Christopher Collins and Sheelagh Carpendale. Vislink: Revealing relationships amongst visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1192 –1199, nov. 2007.

[DHK+06] Tim Dwyer, Seok-Hee Hong, Dirk Koschützki, Falk Schreiber, and Kai Xu. Visual analysis of network centralities. In Proceedings of the 2006 Asia-Pacific Symposium on information Visualisation, editor, *Kazuo Misue, Kozo Sugiyama, and Jiro Tanaka*, pages 189–197, Darlinghurst, Australia, 2006. Australian Computer Society, ACM International Conference Proceeding Series, vol. 164.

[FHN+07] Xiaoyan Fu, Seok-Hee Hong, Nikola S. Nikolov, Xiaobin Shen, Yingxin Wu, and Kai Xu. Visualization and analysis of email networks. *6th International Asia-Pacific Symposium on Visualization, APVIS '07*, pages 1 –8, feb. 2007.

[HCL05] Jeffrey Heer, Stuart K. Card, and James A. Landay. prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 421–430, New York, NY, USA, 2005. ACM.

[Hen08] Nathalie Henry. *Exploring Social Networks with Matrix-based Representations*. PhD thesis, UNIVERSITY OF SYDNEY, AUSTRALIA, July 3 2008.

[HF06] Nathalie Henry and Jean-Daniel Fekete. Matrixexplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12:677–684, 2006.

[HFM07] Nathalie Henry, Jean-Daniel Fekete, and Michael J. Mcguffin. Nodetrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics (IEEE Visualization Conference and IEEE Conference on Information Visualization) Proceedings*, 13:1302–1309, 2007.

[Hol06] Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 12(5), September/October 2006.

[HvW09] Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. *IEEE-VGTC Symposium on Visualization 2009*, 28(3), 2009.

[IS05] Sahler Office Interiors and Information Systems. *Color Coding*. http://www.sahler.com/record-keeping/color-coding, last accessed: 2011-01-04, 2005.

[JHGJCH08] Yuntao Jia, Jared Hoberock, Michael Garland, and IEEE-CS John C. Hart, Member. On the visualization of social and other scale-free networks. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 14(6):1285–1292, November/December 2008.

[JKS06a] Björn H Junker, Christian Klukas, and Falk Schreiber. Vanted: A system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics*, 7(109), March 2006.

[JKS06b] Björn Junker, Dirk Koschutzki, and Falk Schreiber. Exploration of biological network centralities with centibin. *BMC Bioinformatics*, 7(1):219, 2006.

[JMBO01] Hawoong Jeong, S.P. Mason, Albert-László Barabási, and Zoltan N. Oltvai. Lethality and centrality in protein networks. *Nature 411, 41-42*, 2001.

[KS04] Dirk Koschützki and Falk Schreiber. Comparison of centralities for biological networks. In J. Stoye R. Giegerich, editor, *Proc. German Conf. Bioinformatics (GCBŠ04)*, pages 199–206, 2004.

[TSWB94] Lisa Tweedie, Bob Spence, David Williams, and Ravinder Bhogal. The attribute explorer. *CHI'94 - Celebrating Interdependence*, pages 435–436, April 24-28 1994.

[vLKS+10] Tatiana von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J. van Wijk, Jean-Daniel Fekete, and Dieter W. Fellner. Visual analysis of large graphs. *EUROGRAPHICS 2010, 3-7 May, Norrköping, Sweden*, 2010.

[WF94] Stanley Wasserman and Katherine Faust. Social network analysis: Methods and applications. *Cambridge Univ. Press.*, 1994.

[WS03] Stefan Wuchty and Peter F. Stadler. Centers of complex networks. *Journal of Theoretical Biology 223, 45-53*, 2003.

[Wuc02] Stefan Wuchty. Interaction and domain networks of yeast. *Proteomics*, 2:1715 – 1723, 2002.

Linnæus University
School of Computer Science, Physics and Mathematics

SE-351 95 Växjö / SE-391 82 Kalmar
Tel +46-772-28 80 00
dfm@lnu.se
Lnu.se