Visual Analysis of Regulatory Networks

Ricardo Rúben dos Santos Aires

KDBIO, INESC-Id & Instituto Superior Técnico, Technical University of Lisbon, Portugal.
ricardo.aires@gmail.com

Abstract

The amount of biological data obtained by new high-throughput technologies is growing exponentially, leading to the identification of huge regulatory networks. In this context, the analysis and interpretation of the relationships in these networks is becoming a major bottleneck in computational biology. Although some tools are already available to process and analyze biological networks, several difficulties arise when dealing with large regulatory networks involving thousands of protein interactions. One severe bottleneck is the visual analysis and the interpretation of these regulatory networks. In this context, the aim of this work was to develop a new visualization tool for biological networks which is able to represent and analyze large gene transcription networks, in a visual perspective. In this work, we used as a case study, the regulatory network on *Saccharomyces cerevisiae*, provided by YEASTRACT [1], which contains 6310 genes and more than 42000 interactions between them. Furthermore, we analyzed the same network in a graph mining point of view by studying its structure and by trying to find clusters of transcription factors, i.e. communities of genes that are strongly connected within each other, though sparsely connected among others. The search for communities within graphs representing regulatory networks, became a hot topic in molecular biology, based on the assumption that they may have the same functional roles within the cell. However, in our analysis, it was found that it is impossible to find clusters within the TF subnetwork, due to its complexity and structure. Even though, we observed that it was possible to filter sets of TFs that could provide better results when we try to address their cellular roles.

Keywords: *Visual Analysis; Yeast; Visualization Framework; Graph Mining; Community Finding; Regulatory Networks.*

1 INTRODUCTION

1.1 Context and Motivation

With the rapid development of microarray technology it is now possible to have a complete genome sequence in a single array, which represents an exponential increase in the amount of biological data to analyze and interpret. In this way, computational biology and bioinformatics became areas were investigation needed to be done, in fields like protein structure prediction, DNA sequence alignment, gene finding, gene expression analysis, management of biological data and development of algorithms to find relationships within them, etc. Mainly, the objective on studying the results of microarray experiments is to find sets of genes which exhibit similar expression patterns given two or more experimental conditions, and separate those groups of genes from uninteresting groups. The results of these studies shall also be represented in gene transcription regulatory networks, maps of genes and their relationship between each other, usually represented in a graph structure. Since these networks are growing more and more, visualization of the gene networks is crucial not only to depict the network topology, but also to help the analysis and the data interpretation process.

1.2 Contributions

In this work, we develop a new framework to visualize and depict several properties of regulatory associations in the YEASTRACT regulatory network, based on principles of information visualization and we analyze the network structure using graph mining and clustering techniques.

1.3 Paper Structure

First, we explore some background concepts regarding biological network representation and visualization as well as community finding and graph clustering methods for network structural analysis. Secondly, we describe the development of a new visualization and analysis tool. Then, we describe the results of the referred methods applied to the YEASTRACT regulatory network [1]. Finally, we provide some final remarks about the developed work and discuss some guidelines for future work.

2 BACKGROUND

2.1 Biological Networks

The enormous quantities of gene expression data outputted by microarray experiments allowed to study the causal relations between genes and build regulatory networks. Another way to construct a regulatory network is by searching through existing information about molecular interactions in literature. All the information obtained by these methods allowed for the construction of hundreds of online repositories and databases to store the networks' content. These represent platforms of work for biologists to keep finding more and more about gene and protein interactions, cellular functional modules, signaling pathways, etc. Networks are also a starting point for new experiments by tightening the field of analysis, as well as methods to validate experimental results. In the end, it is a feedback loop of knowledge, where the creation of knowledge improves the creation of further knowledge.

2.2 Network Visualization

Due to the vast quantity of databases comprising hundreds of networks with great levels of information, analyzing these data is far from easy. This way, it is necessary to represent data is visual ways, so it becomes easier for analysts to better understand the information and to extract relevant knowledge. Visualization takes an important role in human perception, since we acquire more information through vision than through all the other senses combined. Furthermore, the eye and the visual cortex of

the brain form a massively parallel processor that provides the highest-bandwidth channel into cognitive locations on the brain, therefore perception and cognition are closely related [2]. Then, it is also straightforward that the human brain has extraordinary visual processing capabilities to analyze patterns and images. This is related to information visualization, known as "the use of interactive visual representations of abstract data to amplify cognition" [2]. This means that information should be visualized in several ways to aid the process of thinking and uncover unknown knowledge. Typically, information visualization explores beyond static pictures and graphics, and thus tend to be computerized and interactive. Ware describes that interactive visualization is a process made of interlocking feedback loops, which fall in three different levels of interaction:

- Data manipulation loop - The first level of interaction where the user search through the visualization, selecting and moving objects using basic skills of eye-hand coordination;
- Exploration and navigation loop - The intermediate level of interaction, where the user searches through the visualization, finding his way in a large visual space;
- Problem-solving loop - Loop at the highest level through which the analyst forms hypotheses about the data in order to unveil relevant information and increase cognition.

It should be clear that a network can be represented as a graph. Most of the time the two concepts are actually referred as the same. Graphs are powerful visualization tools, whose main purpose is to help on the analysis and interpretation of data. Graph visualization can be applied to any type of abstract data which has some kind on internal relationship. Furthermore, graphs are very simple and intuitive structures which can be pictured easily, even if not computationally, thus easy to represent and understand. This is based on the strong human visual perception and the fact that information can be readily perceived without being interpreted and formulated explicitly [3].

2.3 Existing Software for Biological Network Representation

An analysis tool for interaction networks should enable a user to import, store, retrieve and perform analysis on single genes, gene sets and the global structure of the network and interact with them. The framework must be flexible and efficient for all sizes of networks and different types of data originated from as many sources as possible. In the visualization part, it should enable users to construct new graphs through on-demand queries.

One of the most popular among biological communities is the open-source software environment *Cytoscape* [4]. It has the ability to integrate the visualization of different biomolecular networks with expression data, different network states and gene annotations from existing repositories and databases, into a single framework. It also provides highly scalable interactive visualizations for large networks, with different graph layouts and customizable visual styles. Another powerful virtue of *Cytoscape* is the possibility for developers to incorporate plug-ins into the system.

VisANT is an open source, plug-in extendable Java-based web applet for visualization, creation and analysis of mixed large-scale networks [5]. It is designed to perform integrative visual data-mining, offering the possibility to import data from several online public databases. It incorporates different graph layouts and visualization schemes, GO annotations, SVG image export, extensive analysis over networks topology, a batch mode, and so on. An example of a *VisANT* session can be seen in Figure 1.

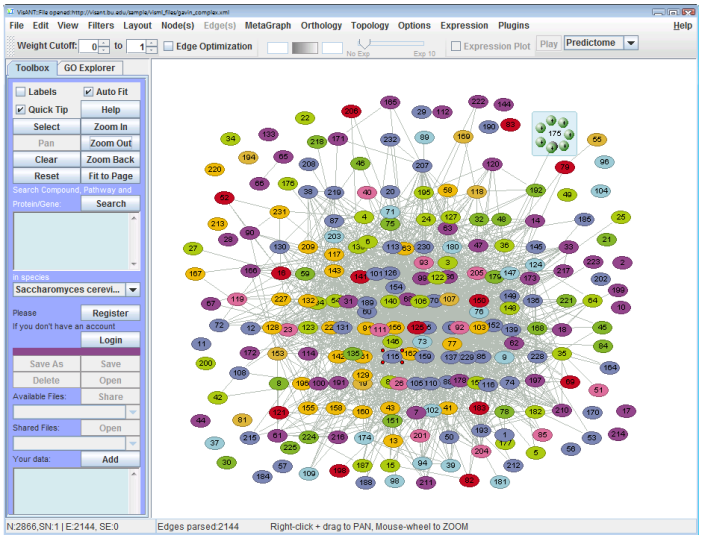


Figure 1: Overview of a VisANT session.

BiologicalNetworks [6] is a freely-distributed Java integrated research environment combining molecular interaction networks, signaling pathways, genomic sequences, 3D protein structure information and many other. It introduces a database-level integration method using an SQL-like query language, able to search through virtually almost every combination of biological properties. Visualization mechanisms include heat maps and graph styled views, which may incorporate different types of nodes, representing several types of molecules, regulations, functional groups, macromolecular complexes and more.

One of the first tools developed for network visualization was *Osprey* [7]. Different graph layouts (although most of them are variations of the circular layout) can be used for dataset representation and genes can be colored based on GO annotations [8]. Output files of generated networks include image printing and SVG. One important feature of *Osprey* is the ability to superimpose several networks and compare similarities.

While most of the mentioned tools develop or adapt algorithms and program specific and incorporated libraries to perform the visualization tasks, other applications make use of already existing toolkits to implement the networks' drawing and interactive visualizations.

The *prefuse* [9] library is one of the most commonly used to build interactive and graphically rich applications. It features mechanisms to simplify the process of representation and data-handling, such as components for layout, color, size and shape, interaction controls, animation, panning and zooming, interactive filtering, integrated text, and a user friendly API to create custom components.

JUNG (Java Universal Network/Graph Framework) is also a open source Java toolkit for graph drawing modeling and analysis, it is focused on graph manipulation, since it supports different graph based classes, such as directed graphs, undirected, hypergraphs, graphs with parallel edges and multi-modal graphs. Furthermore, it includes several graph theory algorithms: clustering,

shortest path finding, random graph generation, statistical and topological analysis and many more.

Related to the original *prefuse* toolkit is the *prefuse flare* toolkit. This library is written in ActionScript [10] and is able to create different visualization styles: graphs with different layouts, charts, maps, pie charts and many others, all with interactive control and animation techniques. Like the previous libraries it has plenty of incorporated classes to deal with data management. Graphs can be imported or exported from several type formats, such as GraphML, JSON or even as a tab delimited text file. Support for topology analysis is also present, although not as much extense as JUNG's.

2.4 Graph Theory Concepts

A graph G is a pair (V, E), being the vertex set of G represented by $V(G)$ and the edge set by $E(G)$, with $V(G) = V$ and $E(G) = E$ such that, $E \subseteq V \times V$. The number of vertices in the graph is denoted by $|V|$ or $|G|$, while the number of edges is given by $|E|$ or $||G||$. Two vertices are *connected* if $(u, v) \in E$ with $u, v \in V$. In the case where all the edges within a graph have no specific orientation, the graph is called *undirected* and for all $u, v \in V, (u, v) \in E$ and $(v, u) \in E$ represent the same edge, being E a set of unordered pairs. If we state that E is a set of ordered pairs the edges $(u, v) \in E$ and $(v, u) \in E$ are different and the graph is *directed*. In this case, edges have an orientation from one vertex to another. A *weighted* graph is a tuple $G = (V, E, \omega)$, wherein each edge of the graph is assigned a weight, by the function $\omega : E \to \mathbb{R}$. This quantifies the strength of the connection. A vertex $u \in V$ is a *neighbour* of another vertex $v \in V$, if $(v, u) \in E$. The set of neighbour vertices of $v \in V$ in G is denoted by $N_G(v)$. The *degree* of a vertex is given by the size of its set of neighbours, $d_G(v) = |N_G(v)|$. In a directed graph, one can distinguish between the *out-degree* of a vertex, $d_G^+(v)$, the number of outgoing edges of that vertex and the *in-degree*, $d_G^-(v)$, the number of incoming edges. A graph G is *connected* if there is a path linking any pair of distinct vertices in G. Given a disconnected graph G, a *connected component* C of G is a maximal set of nodes, $C \subseteq V(G)$, such that exists a path in C connecting any pair of distinct vertices of C. An *adjacency matrix* is a data structure to represent matricially the topology of the graph, i.e. which vertices are connected to each other. Let A_{ij} be an element of the adjacency matrix, A, then $A_{ij} = 1$, if the vertices in the positions i and j are connected, and $A_{ij} = 0$ otherwise. The adjacency matrix is $n \times n$, being n the order of the graph.

2.5 Spectral Clustering

We shall now look into different community finding and graph clustering methods and analyze how they work and how they can be used to extract meaningful information of networks.

Spectral clustering analysis uses the eigenvalue decomposition of the graph adjacency matrix to determine optimal partitioning of the graph. Spectral clustering uses the Laplacian matrix or combinatorial Laplacian L defined as $L = D - A$, where A is the adjacency matrix and D the degree matrix, a $|V| \times |V|$ matrix where $D_{ii} = d_G(i)$. The normalized Laplacian is defined as $L = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, in the case the weight of the each vertex is proportional to its degree. In this problem, we aim to solve the eigenvalue problem on the Laplacian matrix:

$$Lv_i = \lambda_i v_i, \tag{1}$$

resulting in the eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_{n-1}$ and their correspondent eigenvectors, given by $v_0, v_1, ..., v_{n-1}$ being n the order of the graph. Being G an undirected graph, the Laplacian matrix will have real eigenvalues, and the first eigenvalue is always zero, and there will be as many zero eigenvalues as the number of connected components in the graph. Thus, an eigenvalue near zero must belong to an eigenvector with both positive and negative components, which partition the graph into nearly disconnected components. To the first non-zero eigenvalue (also known as the algebraic connectivity) in the set of ordered eigenvalues, is associated the Fiedler vector, named after Miroslav Fiedler [11]. Moreover, the algebraic connectivity is a measurement of how good the division in partitions is, with smaller values representing better divisions [12]. By ordering the rows and columns of the adjacency matrix according to the order of the Fiedler vector, we can obtain and visualize partitions in the graph.

2.6 Graph Cores

In this section, we focus on the core decomposition method to extract highly connected subgraphs or cores from a graph. This approach is slightly different from spectral clustering, where we try to partition the graph in several parts. In this case, we want to extract only certain cohesive communities, thus their union is not necessarily equivalent to the entire graph. Given a graph $G = (V, E)$, and a subset of vertices $C \subseteq V$, we say that the graph H, induced by C, is a *k-core* or core of order k, with $k \in \mathbb{N}$, if and only if $d_H(v) \geq k$, for all $v \in C$ and H is a maximal subgraph with this property. Cores are not necessarily connected subgraphs, in which case each of the components is called a k-core community. The community with the highest order is also called the main core. The cores are also nested: let C_i and C_j represent cores of order i and j respectively and H_i, H_j the subgraphs they induce; then if $i < j$, we have $H_j \subseteq H_i$ [13]. Figure 2 shows this property of k-core communities.

Recently, Batagelj and Zaveršnik [14] proposed a generalization of the notion of core. Let $G = (V, E)$ be a graph, eventually weighted, and $p : V \times 2^V \to \mathbb{R}$, a vertex property function on G. Then given $\tau \in \mathbb{R}$, a subgraph H of G, induced by $C \subseteq V$, is a τ-core or a p-core at level τ if $p(v, C) \geq \tau$, for all $v \in C$ and H is a maximal subgraph with this property. Furthermore, a vertex property function p is monotone if and only if, it has the property that given $C_1 \subseteq C_2 \subseteq V$, $p(v, C_1) \leq p(v, C_2)$, for all $v \in V$. Examples of monotone property functions are: $p_1(v, C) = d_H(v)$, which leads to the classical definition of k-core proposed and $p_2(v, C) = \sum_{u \in N(v) \cap C} w(u, v)$ for weighted graphs.

To compute the τ-cores, the following procedure is used: given a graph G, a monotone vertex property function p and $\tau \in \mathbb{R}$, we successively delete the vertices with value of p lower than τ:

1. Set $C \leftarrow V$;
2. While exists $v \in C : p(v, C) < \tau$, do $C \leftarrow C \backslash \{v\}$

C is maximal and independent of the order of deletions.

Saito *et* al. [15] proposed a more community restrictive notion of subnetworks, the *k-dense communities* [15]. Let H be a subgraph of G, $H \subseteq G$, we say that H is a k-dense community, if each pair of adjacent nodes in H, has more than $k - 2$ common neighbours in H, that is, for all $\{u, v\} \in E(H)$ with $\{u, v \in V(H), | \cap_{i \in \{u,v\}} N_H(i)| \geq k - 2, k \in \mathbb{N}\}$. This is a much more restrictive condition than that of the k-cores' since the existence of shared neighbour nodes evidences a stronger relation

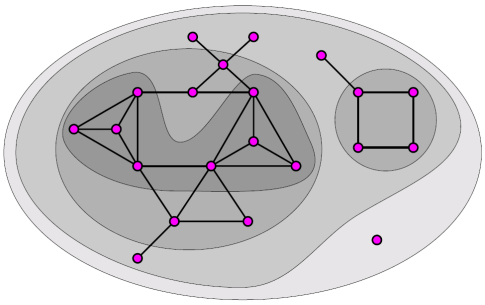


Figure 2: Example of nested k-core communities in a small graph: 0, 1, 2 and 3-core communities, increasing as the shadow darkens. Notice that there are disconnected components within the same core. Reproduced from [14].

between them, rather than being simply connected. To obtain the H, k-dense communities, we can apply the k-dense extraction algorithm proposed by Saito *et* al. [15], wherein we can obtain the k-dense communities by computing first the respective k-cores:

1. Set $H \leftarrow G$;
2. Compute $H \leftarrow kcore(H)$ and $L = \{\{u,v\} \in E(H) : |\cap_{i\in\{u,v\}} N_H(i)| \leq k-2\}$;
3. If $L = \emptyset$, output H, otherwise $E(H) \leftarrow E(H) - L$ and return to 2.

Here, L denotes a set of edges to remove under the k-dense condition. We can extend the notion of k-dense communities to weighted graphs as we did for the k-core communities. In this case, the property function p becomes:

$$p(\{u,v\},H) = \sum_{z\in X\cap V(H)} [w(u,z) + w(v,z)], \tag{2}$$

where H is a τ-dense community, with $\tau \in \mathbb{R}$ and $\{u,v\} \in E(G)$ and $X = N_G(u) \cap N_G(v)$. The procedure to obtain this communities is similar to the one used to obtain the k-dense's.

2.7 Personalized Ranking

Quantitative ranking is related to local clustering and was introduced by Brin and Page [16] when analyzing the network structure of the web. They created PageRank [16], an algorithm used by Google that assigns a weight to each element of a set of web pages in order to determine its relative importance within the set. The approach is based on random walks as a geometric sum. The ranking algorithm is very important for search engines to sort query results for the users. A new notion of Pagerank was introduced by Chung [17] by using the heat kernel of a graph, which is also based on random walks, but with the benefit of satisfying the heat equation and being more accurate. This approach is relevant in the context of regulatory networks, to find regulatory associations given a set of genes. Let a regulatory network be represented as a directed unweighted graph $G = (V,E)$. Let A and D be the adjacency and diagonal matrices of G, respectively, where $D_{ii} = d(v_i)$, is the outdegree of vertex i. Also, let $W = D^{-1}A$, and $L = I - W$. Given a set $T \subseteq V$ of target genes, we want to obtain a rank $R \subseteq V$, where R is the set of regulators of the genes in T. The ranking p_t is given by

$$p_t = \sum_{k=0}^{\infty} \frac{-t^k}{k!} p_0 L^k = p_0 e^{-tL}, \tag{3}$$

being p_0 the preference vector, and t the heat diffusion coefficient, which controls the diffusion. The discrete approximation is given by:

$$p_t = p_0 (I + \frac{-t}{Z} L)^Z, \tag{4}$$

where Z is the number of iterations. Note that the higher the heat coefficient is, the faster heat will diffuse. Since the problem here consists on finding potential regulatory connections, we consider G as the transposed of the graph representing the original network, being the new adjacency matrix $A' = A^T$.

Furthermore, we can measure the quality of a graph partition by computing its conductance. This way, let $\mathcal{P}$ be a partition of graph $G = (V,E)$, we define the conductance of $C \in \mathcal{P}$ as:

$$\phi(C) = \frac{\sum_{u\in C} |N(u)\backslash C|}{\sum_{u\in C} |N(u)|}, \tag{5}$$

It is clear that conductance takes values between 0 and 1 and that it is minimized if C is V [18].

3 Developed Framework

In this work, a new client-side tool for gene regulatory network visualization and analysis was developed. The purpose of this application is to provide a mechanism for visual analysis of regulatory networks, based on the YEASTRACT repository [1], which contains thousands of regulatory interactions in *Saccharomyces cerevisae*. This application makes use of the new YEASTRACT Web services [19] to import data necessary to generate a visualization of a subnetwork of interest. Based on principles of information visualization, it has an interactive nature, which makes the visual analysis process easier. The application is currently available at http://nebm.ist.utl.pt/ rsantos/tese/app.swf

3.1 Functionalities

In this application, two different resources are available to generate a network visualization: the *TF-Rank* and the *GroupbyTF* services. The *TF-Rank* resource consists on a visualization where TFs are sorted according to the results of a personalized ranking algorithm [20]. The output for each TF is a real number and the higher it is, the more likely that TF will act as a regulator for the set of input genes. The *GroupbyTF* resource aims at grouping an input list of genes to their documented regulators present in the YEASTRACT database.

To give input to the application, the box on the left shall be used (see Figure 6); it contains fields for: the list of genes/ORFs, the corresponding expression values, a rank minimum threshold and a box for choosing different visualization colors.

3.2 Design

This application is written in ActionScript [10] using the *prefuse flare* library. The input data (genes and weight values) and further parameters are transmitted to the server through a HTTP request and the resource responds to the POST method for *TF-Rank* and to the GET method for *GroupbyTF*. Note that the results of each of the resources provided are computed server-side, while the client side only stores the data and provides a mean for visualization. The server responds to the HTTP request by returning a representation of the data using Java Script Object Notation (JSON). Then, in the client side, the results are parsed and transformed into the appropriate flare data structures

to provide accurate visualization. It is possible to compute several visualizations successively.

3.3 Visualization

The visualization is based on a double semi-circled directed graph where nodes represent genes and TFs; and edges represent regulations between the genes and TFs. Each node is represented by a label corresponding to the gene name or ORF. In the case of *TF-Rank*, the TFs are placed in the top semi-circle, while the genes are placed in the bottom one, as seen in Figure 3(a). For *GroupbyTF*, the genes are placed in the left semi-circle, whereas the TFs are placed in the right one (see Figure 3(b)).

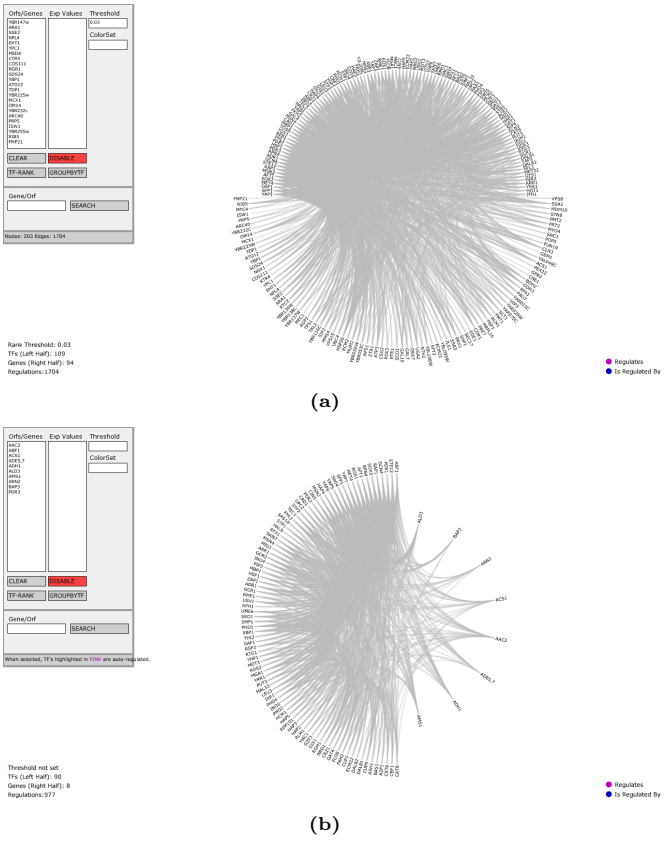


Figure 3: Overview of the developed application, a) after selecting *TF-Rank* for a set of TFs and b) after selecting GroupbyTF, for a set of genes.

When a particular gene or TF is selected or mouse-overed, the edges connecting the regulated genes and regulators of that particular gene are highlighted. Regulator and regulated elements are highlighted in different colors, which may differ based on the chosen color palette. The info regarding the color of the edges is shown in the bottom right legend. When the graph is generated, all the edges are drawn, so all regulations are present. In order to reduce edge cluttering, when a node is selected only the regulations regarding that particular gene or TF are shown (see Figure 4).

Several consecutive selections can be made, which means that after selecting one gene or TF, it is possible to select another element without losing the previously generated subgraph. The set of selected genes is identified by a particular color, different from edge and node colors, so the user is able to keep track of his/her selections. After several interactive selections, the image returns to its original state by clicking in any area outside of the double-radii circle. This case is represented in Figure 5.

If *TF-Rank* was chosen, by mouse-overing one of the TFs, it is possible to see the respective rank value in the left bottom bar (under the search box), along with the gene and ORF names. The latter is also valid for each of the genes in the bottom semi-circle. The TFs are sorted in decreasing order of rank from left to right and the genes are sorted in alphabetical order from right to left. In the case of the *GroupbyTF* service, for each TF, the percentage of genes regulated is shown. The TFs are sorted in decreasing order of percentage of regulated genes from top to bottom, while the genes are sorted in alphabetical order from top to bottom.

Searching for a gene or group of genes is done by entering their names, separated by a semicolon in the search box on the left and by pressing the Search button. The found genes or TFs are highlighted in red, while the other elements are faded out.

The considered layout allows for the visualization of over 300 total vertices, without superimposing the labels, although this value may differ based on screen resolutions.

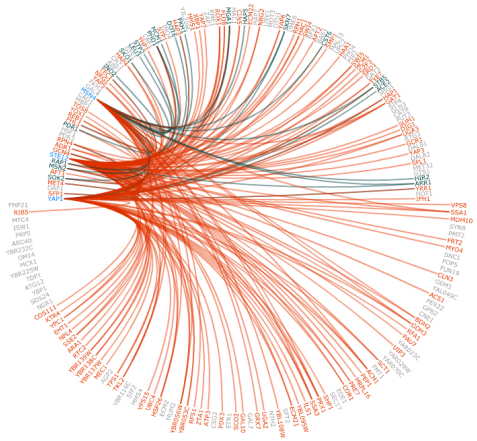


Figure 4: Drawn network for the *TF-Rank* resource, with color set 3, the transcription factor STE12 is selected.

3.4 Discussion

The developed application proved to be efficient for large datasets regarding the construction of the graph and node layout, with most of the time spent when loading the visualization being shared between layout placement and server-side computations and response. Although several graph related libraries were considered, the *prefuse flare* toolkit was used. When compared to JUNG, *flare* is stronger in what concerns to user interaction, while JUNG is much more oriented to graph manipulation and analysis. It is important to note that interaction takes a huge part in information visualization and it is a relevant mechanism for visual analysis.

The layout used for visualizing the graph is a variation of the *prefuse flare* CircleLayout class, developed in this work. In our case, there are two half-circumferences with different radii where the nodes are positioned. This proved to be an interesting alternative compared to other layouts, not only because it was absolutely

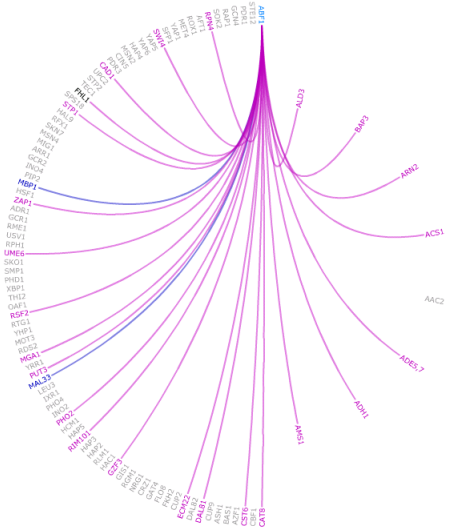


Figure 5: Drawn network for the *GroupbyTF* service; the transcription factor ABF1 is selected.

necessary to visually separate the TFs and the genes (for a better understanding of the network), but also because the TFs needed to be presented in sorted order according to the results of ranking (in the case of *TF-Rank*) and percentage of regulated genes (in the case of *GroupbyTF*) and no other layout was suitable enough to address this problem, since no other one can represent numerical order so clearly.

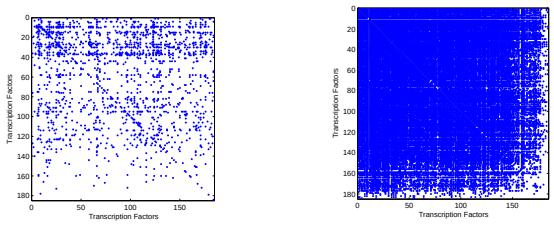
When the network is presented on the screen, it is difficult to depict any information regarding the regulations on large datasets with a great number of interactions, because of edge cluttering. Furthermore, the necessity of drawing the edges is also a factor that may slow down the process of visualization. However, the edges are still drawn, so it is possible to visualize immediately the complexity of the network, even though the number of regulations are also shown in the left bottom legend. To eliminate this situation, when a node is selected only the regulations regarding that particular gene or TF are highlighted (see Figure 5).Another relevant feature is the multiple selection of genes or TFs. Although this feature may increase edge cluttering and slow down the application, it is useful to compare interactions between several nodes, particularly when we are addressing the regulations between two or more TFs. This can be relevant for molecular biologists to understand the behaviour of the network and its components (see Figure 4).

4 Performed Analysis

Let us now focus on the analysis of certain features of the YEASTRACT transcription regulatory network. We will apply the methods referred earlier with the main idea of identifying functional cohesive groups, based on the assumption that genes which are closely related in the network share the same biological function within the cell. The YEASTRACT network composed of 6310 genes and TFs is very sparse (42678 interactions), although if we focus only in the TF subnetwork, that is, if we select only the subset of genes which have an out-degree higher than zero (totaling 184 TF), we see that the resulting subnetwork is not that sparse (1640 regulations), see Figure 6(a). To analyze similarity between vertices, we compute a *vertex similarity function*, such that $\sigma : V^2 \rightarrow [0, 1]$. The value of σ reflects the similarity between two vertices, $u, v \in V$, thus the higher the value $\sigma(u, v)$ is, the more similar the vertices are, i.e. they tend to have the same neighbours. Here we will use the cosine similarity to measure similarity between two vertices. Note that our starting point is the directed unweighted graph representing all the regulations present in the entire YEASTRACT database, and in our case, similarity is given by:

$$\sigma(u, v) = \frac{|\langle u, v \rangle|}{||u||\,||v||}, \tag{6}$$

where each vertex is represented here by his respective row vector in the adjacency matrix of the unweighted directed graph. The resulting matrix from the previous metrics originates a new form of representing the graph. The values computed by the vertex similarity function define a new adjacency matrix of an undirected weighted graph, which will be further represented simply as A. Only for the TF network, the pattern observed in the Figure 6(b) shows that in this case the graph is far from sparse.



(a) **(b)**

Figure 6: a) Sparsity pattern of the TFs in the YEASTRACT network. Each blue dot represents a regulation. b) Adjacency pattern of the undirected weigthed graph defined by the selected vertex similarity function.

In the next section, we analyze only the TF subnetwork, based on the adjacency matrix that we have just defined.

4.1 Spectral Clustering

We will now apply the spectral clustering methodology previously described on the undirected weighted graph defined by adjacency matrix A to find partitions on the TF subnetwork. For this level of similarity, there is only one null eigenvalue and we observe that there is only one community, as seen in Figure 7. To find separate partitions or components using spectral methods, we know that there will be as many disconnected components as null eigenvalues. Therefore, we can only obtain a single community composed of the entire dataset, which we can see when plotting the sparsity pattern of the obtained matrix. See Figure 7.

Since we cannot find explicitly good partitions, for the same level of similarity, we analyze the same values for different versions of the adjacency matrix. We select a given threshold value, and replace all entries of the matrix by 1 if the value is higher than the threshold and by 0 if lower, and then repeat the same procedure. Figure 9 shows the plot of the Fiedler vector for several thresholds. We selected nine different thresholds spaced by 0.03, so it is possible to observe the evolution of the results.

Figures 8 and 9 show respectively, the sorted adjacency matrix for the various thresholds, after the spectral clustering analysis

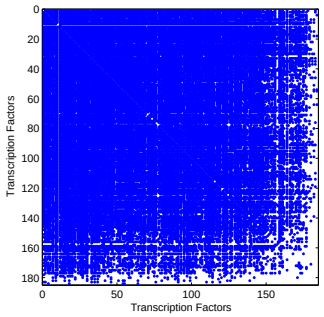
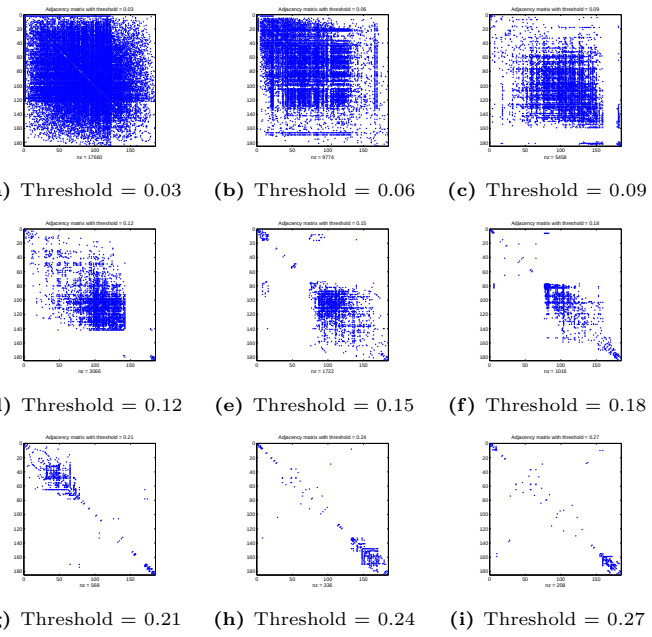


Figure 7: Sparsity pattern of the obtained matrix by spectral clustering.



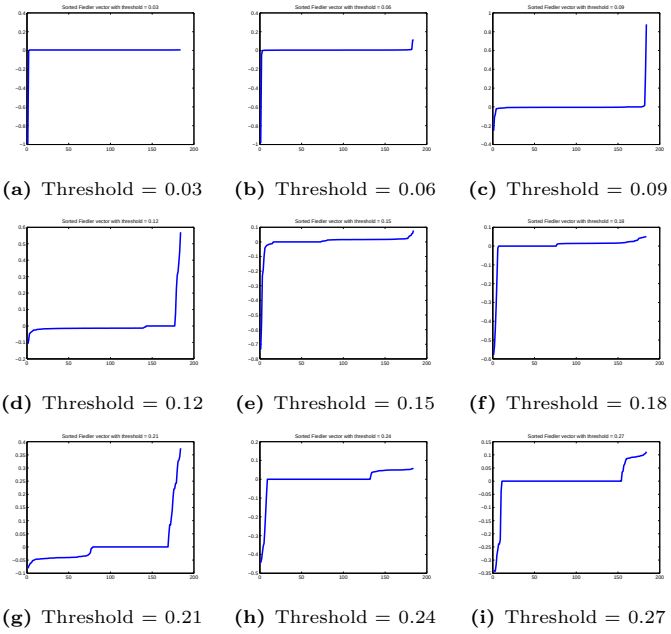
(a) Threshold = 0.03 **(b)** Threshold = 0.06 **(c)** Threshold = 0.09

(d) Threshold = 0.12 **(e)** Threshold = 0.15 **(f)** Threshold = 0.18

(g) Threshold = 0.21 **(h)** Threshold = 0.24 **(i)** Threshold = 0.27

Figure 8: Sorted plot of adjacency matrix after spectral clustering analysis for different thresholds.

and the respective values of the Fiedler vector for the same thresholds. The fact is that even increasing the threshold, we will always obtain a big component of TFs, even though the number of zero eigenvalues keeps increasing, as seen Figure 10. With the number of zero eigenvalues increasing, the number of disconnected components will also increase, although these components are, in most of the cases, composed of a single TF and thus with no relevant component meaning. Furthermore, we see in Figure 11 that the first eigenvalue is smaller for thresholds between 0.2 and 0.3, which represents a better partition when comparing to lower thresholds and we can effectively see a partition by looking at the values of the Fiedler vector for those levels of threshold. However by looking at the correspondent sparsity pattern in Figure 8, we see that the resulting components are very small when compared to the original component where no threshold was imposed, thus they are not very relevant. Remember that the original goal when applying this method was to cluster the entire set of TFs.



(a) Threshold = 0.03 **(b)** Threshold = 0.06 **(c)** Threshold = 0.09

(d) Threshold = 0.12 **(e)** Threshold = 0.15 **(f)** Threshold = 0.18

(g) Threshold = 0.21 **(h)** Threshold = 0.24 **(i)** Threshold = 0.27

Figure 9: Sorted Fiedler vector plots for different thresholds.

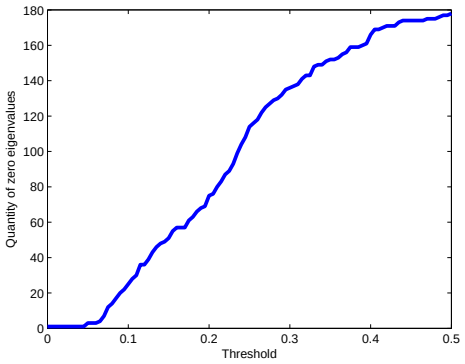


Figure 10: Plot of number of zero eigenvalues for different thresholds.

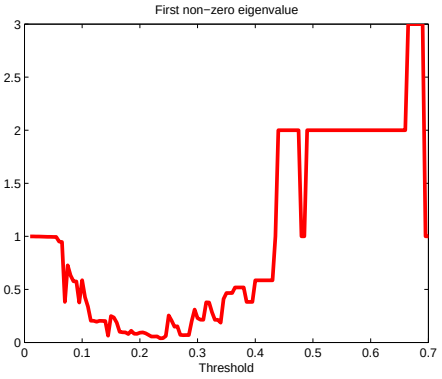


Figure 11: Value of the first non-zero eigenvalue for different thresholds.

4.2 Graph Cores

This method is different from spectral clustering, since in this case we do not want to separate all the graph into several components.

In this case, our goal is to extract groups of TFs that satisfy a certain property function. In our first approach, we use the method proposed by Batagelj and Zaveršnik [14] (previsouly explained) to find τ-cores in our similarity graph and the following property function for weighted graphs:

$$p_2(v, C) = \sum_{u \in N(v) \cap C} w(u, v) \quad (7)$$

For various levels of τ, the sizes of the core communities decreased as shown in Figure 12.

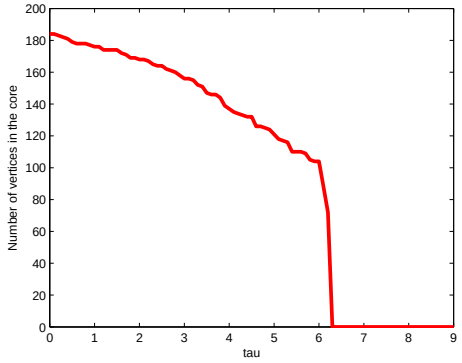


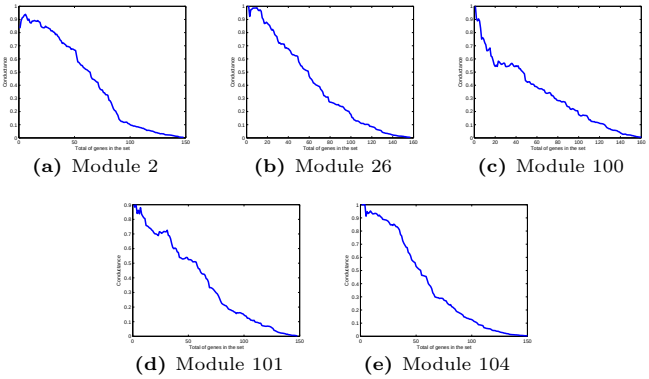
Figure 12: Length plot of τ-core communities.

The size (total number of vertices) of the cores decreases with τ as expected until $\tau \approx 6.26$ when it reduces to zero abruptly, which means that up to some level the TFs are very similar in their behaviour, that is they tend to have the same neighbours. Unsurprisingly, considering the results obtained in the previous section, for all analyzed cores at different sizes, there was only a single connected component. This corroborates the results obtained in spectral clustering, when we saw that the network was so dense that it would be difficult to find partitions.

4.3 Personalized Ranking

To finish our set of analysis of the YEASTRACT regulatory network we use the ranking method described in Section 2.7. Since we could not effectively find clusters in the TF subnetwork, we now try to identify subsets of TFs that can provide better results when we try to identify their cellular function. For this, we will use as seed sets the modules selected by Francisco *et* al. [20] (previously identified by Lemmens *et* al. [21]) and reproduce the ranking results in this study to perform our analysis. We analyze the TF modules 2, 26, 100, 101 and 104, involved respectively in regulation of cellular respiration, stress response, ribossome biogenesis, nutrient deprivation and galactose metabolism. These modules were chosen since they were the ones which obtained the best p-value results in the work developed by Lemmens *et* al. [21]. We then use the conductance measure to find those smaller subsets within each module which should be more functionally related according to our reasoning and submit them to the Gene Ontology [8] to quantify their significance when compared to half of the total number of TFs in the module, since the graph is directed.

The ranked lists of regulators for each module were obtained by running 100 iterations of the ranking method with an heat diffusion coefficient of $t = 1.0$. We then normalized the rank values for each TF by its out-degree in the original network. Then, we computed the conductance (Equation 5) for the set as we add the TFs in decreasing order of normalized ranking, that is, we compute the conductance each time a node is added to the set. In Figure 13 presents the evolution of the conductance, for each module.



(a) Module 2 **(b)** Module 26 **(c)** Module 100

(d) Module 101 **(e)** Module 104

Figure 13: Evolution of conductance for the different modules.

Furthermore, we selected the subsets of TFs for which we could find a local minimization of conductance (see Figure 13) and submited those seed sets to the Gene Ontology [8] to find their related functions. A local minimum in the conductance measurement, for a set of TFs, means that after adding more nodes to the set and we will obtain more edges leaving the nodes which define the set. Then, for the set where we identify the relative minimum, there is also a local minimum of crossing edges, which represents a strongly connected subset, when compared to the rest of the still unadded elements.

Table 1 shows, for each module, the number of TFs in the sets for which we could find a local minimum.

Table 1: Number of TFs in the selected sets based on conductance local minimum.

Module	Number of TFs
Module 2 (Respiration)	11, 16, 23
Module 26 (Stress)	17, 24, 31
Module 100 (Ribosome Biogenesis)	6, 7, 12, 20
Module 101 (Nutrient Depravation)	23, 37
Module 104 (Galactose Metabolism)	16, 27

To obtain results from the Gene Ontology [8], the GOToolBox framework was used [22]. This toolbox allows for the functional investigation of groups of genes, based on the Gene Ontology resource. For each group in Table 1, we obtained the GO terms relevant to each module with the smallest p-value, which are shown in Tables 2 to 6. The p-value measure refers to the statistical significance of the terms for the given sets, that is, the lower the p-value, the more significant the terms will be. For each module, we also submitted to the Gene Ontology half of the initial set of regulators according to their normalized ranking order, for comparison. The reason for submitting half of the regulators is based on the fact that since the graph is directed, edges have only one direction and they should be considered only once.

We have then selected, for each module, the most relevant terms according to the modules description provided by Lemmens

Table 2: p-values for the most relevant GO terms for module 2, for different subsets of ranked transcription factors.

	Number of TFs			
GO Terms	**11**	**16**	**23**	**74**
Regulation of carbohydrate metabolic process	4.96e-012	2.21e-010	5.37e-011	8.25e-008
Carbohydrate metabolic process	9.09e-005	3.10e-003	3.94e-006	4.00e-001

Table 3: p-values for the most relevant GO terms for module 26, for different subsets of ranked transcription factors.

	Number of TFs			
GO Terms	**17**	**24**	**31**	**78**
Regulation of transcription in response to stress	1.52e-004	4.81e-004	1.39e-003	4.30e-004
Response to drug	6.60e-002	2.10e-002	7.00e-003	5.80e-002
Heat acclimation	5.00e-003	1.20e-002	2.40e-002	2.67e-001

Table 4: p-values for the most relevant GO terms for module 100, for different subsets of ranked transcription factors.

	Number of TFs			
GO Terms	**7**	**12**	**20**	**80**
Ribosome biogenesis	1.0	1.0	1.0	1.0

Table 5: p-values for the most relevant GO terms for module 101, for different subsets of ranked transcription factors.

	Number of TFs			
GO Terms	**6**	**23**	**37**	**79**
Nitrogen utilization	2.00e-002	8.53e-009	2.73e-007	4.67e-005
Nitrate Assimilation	3.46e-004	1.10e-002	4.10e-002	2.97e-001

Table 6: p-values for the most relevant GO terms for module 104, for different subsets of ranked transcription factors.

	Number of TFs		
GO Terms	**16**	**27**	**75**
Galactose metabolic process	3.23e-001	3.23e-001	1.0

et al. [21]. For some of the modules, we observed an improvement in the p-values as we add TFs to the submitted subsets (in decreasing order of the computed ranking), specially for modules 2, 26 and 101. More specifically, we observed improvements for both selected GO terms for module 2, for the last two terms in module 26 and for both terms in module 101. However, the results are not satisfactory for the other two modules (modules 100 and 104), where there is no improvement in the p-values for smaller subsets and the p-values are not statistically relevant, which may also be related with the generality of the terms. We shall note that those were the modules where the ranking results were worse for the normalized ranking measure in Francisco *et* al. [20], with some well positioned nodes in other rankings being shifted to worse positions.

4.4 Discussion

To analyze the structure of the YEASTRACT regulatory network, we applied different community finding methods on the transcription factor subgraph: a method related to graph partition, another related to community extraction and finally a ranking method for transcription factors.

From the results obtained in the spectral clustering section, we can see that it was difficult to effectively separate the network into different clusters since the subset of transcription factors was very sparse, when compared to the adjacency matrix obtained by computing the cosine similarity. The same conclusion was obtained when computing the $\tau-$core and τ-dense communities and although this method is different from spectral clustering, the subgraphs obtained only contained a single giant component. An interesting method to further analyze would be the submission of different sets of communities to the Gene Ontology [8] to address their differences in cellular function and biological processes. This was not feasible, since we could not find well defined clusters, apart from the giant component. The facts that the network is strongly connected altogether and the clustering coefficient is so low, imply the existence of hubs and a multi-layer hierarchical structure of the network. Then, only the elimination of these hubs would part the network, however this elimination is not obtained by the use of the core finding method, since they are very strongly connected and have high degree. The last method, personalized ranking, is a different approach than the previous ones, since it is based in random walks in a graph. The main idea was to find communities from seed sets of TFs, sorted by personalized ranking. From the several modules, we selected the subsets mentioned in Table 1, since each one represented a minimization of the conductance measure in Equation 5, for its respective module. Then, each subset was submitted to the Gene Ontology, to analyze their respective functions. We could observe that there was an improvement in the p-values for smaller subsets in three of the modules, when compared to the p-values obtained for larger sets. This means that we can effectively select fewer TFs to derive cellular function, for those particular modules, being the main goal to find more about gene function, by filtering TFs with relevant GO terms and identifying their cellular roles further clearly.

5 Final Remarks

This work was clearly separated in two distinct parts: the development of a visualization tool based in principles of information visualization and in the mathematical analysis of the YEASTRACT regulatory network.

The high rate at which new data is being obtained is astonishing and new networks are spawning every day and many more are being updated with new information. Although the background of this work is biology related, it is unarguable that this problem is present in different areas, specially in the social sciences. With all these information, it is becoming harder and harder to analyze the global structure and interpret relationships in these networks. A straightforward way to solve this problem is by building tools that can provide visual analysis of the networks, a relevant area for those who work on real network analysis.

In this context, one of our main contributions is the development of a framework to visualize gene regulations in *Saccharomyces cerevisiae* with an interactive nature, one of the most important topics related to information visualization. Part of the application is already available in the YEASTRACT website (http://yeastract.com) for interactive visualization of sets of genes grouped by TFs. This module is very helpful to molecular biologists, since it is now easy to depict selected groups of regulators and regulated genes as well as take snapshots of multiple selected genes or TFs.

In the second part of this work, we analyzed the structure of the YEASTRACT network (or more specifically the subnetwork

composed only of TFs) from a mathematical point of view. Our main goal was to find closely related groups (clusters) of TFs that could be very well separated from each other. We observed that this was not possible, since the network is very strongly connected, although we found out that it is possible to select smaller sets of TFs, using personalized ranking, that were more functionally related.

6 Future Work

The developed application can be improved by incorporating further network analysis methods, in particular those used in our study of the YEASTRACT network. Besides that, new functionalities regarding the remaining web services provided by YEASTRACT [19] can be implemented, as for example the introduction of protein regulations. In terms of visualization, new graph layouts can be used for different visualization needs; although the circular layout proved itself necessary for representing order, for different applications, other kinds of layouts can be used. There is also room for improvement of the current layout in terms of specificity, by providing more options in order to obtain more information, for example, differentiate documented from potential regulations with different colors, or query the network only for a specific type of interaction. Although the application was only used for the YEASTRACT network on *Saccharomyces cerevisiae*, it would be interesting to extend it for the analysis of networks from different organisms, which would increase its field of applications. Regarding network analysis, further methods could be used to study the structure of this network, as well as studying the biological implications of the obtained results.

References

[1] M. C. Teixeira, P. Monteiro, P. Jain, S. Tenreiro, A. R. Fernandes, N. P. Mira, M. Alenquer, A. T. Freitas, A. L. Oliveira, and I. Sá-Correia, "The YEASTRACT database: a tool for the analysis of transcription regulatory associations in Saccharomyces cerevisiae," *Nucleic Acids Research*, vol. 34, no. suppl 1, pp. D446–D451.

[2] C. Ware, *Information Visualization: Perception for Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[3] W. Cui, "A survey on graph visualization," pqe, Computer Science Department, Hong Kong University of Science and Technology, dec 2007. http://www.cse.ust.hk/ weiwei/PQE/WeiweiPQE.pdf.

[4] C. T. Lopes, M. Franz, F. Kazi, S. L. Donaldson, Q. Morris, and G. D. Bader, "Cytoscape Web: an interactive web-based network browser," *Bioinformatics*, vol. 26, no. 18, pp. 2347–2348, 2010.

[5] Z. Hu, J.-H. Hung, Y. Wang, Y.-C. Chang, C.-L. Huang, M. Huyck, and C. DeLisi, "Visant 3.5: multi-scale network visualization, analysis and inference based on the gene ontology," *Nucl. Acids Res.*, vol. 37, pp. W115–121, July 2009.

[6] M. Baitaluk, M. Sedova, A. Ray, and A. Gupta, "BiologicalNetworks: visualization and analysis tool for systems biology," *Nucleic Acids Research*, vol. 34, no. suppl 2, pp. W466–W471, 1 July 2006.

[7] B.-J. J. Breitkreutz, C. Stark, and M. Tyers, "Osprey: a network visualization system.," *Genome biology*, vol. 4, no. 3, 2003.

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

[9] J. Heer, S. K. Card, and J. A. Landay, "prefuse: a toolkit for interactive information visualization," in *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 421–430, ACM, 2005.

[10] C. Moock, *Essential ActionScript 3.0*. O'Reilly Media, 1 ed., July 2007.

[11] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298–305, 1973.

[12] Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas, "Characterization of complex networks: A survey of measurements," *Advances in Physics*, vol. 56, pp. 167–242, Aug 2007.

[13] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, pp. 269–287, 1983.

[14] V. Batagelj and V. Batagelj, "Generalized cores," tech. rep., 2002.

[15] K. Saito, T. Yamada, and K. Kazama, "Extracting communities from complex networks by the k-dense method," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E91-A, no. 11, pp. 3304–3311, 2008.

[16] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," in *Computer Networks and ISDN Systems*, vol. 30, pp. 107–117, 1998.

[17] F. Chung, "The heat kernel as the pagerank of a graph," *Proceedings of the National Academy of Sciences*, vol. 104, pp. 19735–19740, December 2007.

[18] R. Andersen and K. J. Lang, "Communities from seed sets," in *WWW '06: Proceedings of the 15th international conference on World Wide Web*, (New York, NY, USA), pp. 223–232, ACM, 2006.

[19] D. Abdulrehman, P. T. Monteiro, M. C. Teixeira, N. P. Mira, A. B. Lourenço, S. C. dos Santos, T. R. Cabrito, A. P. Francisco, S. C. Madeira, R. S. Aires, A. L. Oliveira, I. Sá-Correia, and A. T. Freitas, "YEASTRACT: Providing a programmatic access to curated transcriptional regulatory associations in Saccharomyces cerevisiae through a web services interface," *Nucleic Acids Research*, 2010.

[20] A. Francisco, J. Goncalves, S. Madeira, and A. Oliveira, "Using personalized ranking to unravel relevant regulations in the saccharomyces cerevisiae regulatory network," *Proceedings of Jornadas de Bioinformatica*, 2009. 3-6 November 2009, Lisbon.

[21] K. Lemmens, T. Dhollander, T. De Bie, P. Monsieurs, K. Engelen, B. Smets, J. Winderickx, B. De Moor, and K. Marchal, "Inferring transcriptional modules from chip-chip, motif and microarray data," *Genome Biology*, vol. 7, no. 5, p. R37, 2006.

[22] D. Martin, C. Brun, E. Remy, P. Mouren, D. Thieffry, and B. Jacq, "Gotoolbox: functional analysis of gene datasets based on gene ontology.," *Genome Biol*, vol. 5, no. 12, 2004.