MULTI-RESOLUTION VISUALIZATION OF LARGE SCALE PROTEIN NETWORKS ENRICHED WITH GENE ONTOLOGY ANNOTATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SEVGİ YAŞAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2009

Approval of the thesis:

MULTI-RESOLUTION VISUALIZATION OF LARGE SCALE PROTEIN NETWORKS ENRICHED WITH GENE ONTOLOGY ANNOTATIONS

submitted by **SEVGİ YAŞAR** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Müslim Bozyiğit _____
Head of Department, **Computer Engineering**

Asst. Prof. Dr. Tolga Can _____
Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Asst. Prof. Dr. Tolga Can _____
Computer Engineering Dept., METU

Prof. Dr. Göktürk Üçoluk _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. Veysi İşler _____
Computer Engineering Dept., METU

Dr. Çağatay Ündeğer _____
Informatics Institute, METU

Dr. Sinan Kalkan _____
Computer Engineering Dept., METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: SEVGİ YAŞAR

Signature :

ABSTRACT

MULTI-RESOLUTION VISUALIZATION OF LARGE SCALE PROTEIN NETWORKS ENRICHED WITH GENE ONTOLOGY ANNOTATIONS

Yaşar, Sevgi
M.S., Department of Computer Engineering
Supervisor : Asst. Prof. Dr. Tolga Can

September 2009, 34 pages

Genome scale protein-protein interactions (PPIs) are interpreted as networks or graphs with thousands of nodes from the perspective of computer science. PPI networks represent various types of possible interactions among proteins or genes of a genome. PPI data is vital in protein function prediction since functions of the cells are performed by groups of proteins interacting with each other and main complexes of the cell are made of proteins interacting with each other.

Recent increase in protein interaction prediction techniques have made great amount of protein-protein interaction data available for genomes. As a consequence, a systematic visualization and analysis technique has become crucial.

To the best of our knowledge, no PPI visualization tool consider multi-resolution viewing of PPI network. In this thesis, we implemented a new approach for PPI network visualization which supports multi-resolution viewing of compound graphs. We construct compound nodes and label them by using gene set enrichment methods based on Gene Ontology annotations. This thesis further suggests new methods for PPI network visualization.

Keywords: Protein-protein interaction network, Compound graphs, Visualization, Gene ontology, Clustering

ÖZ

GEN ONTOLOJİ AÇIKLAMALI BÜYÜK ÖLÇEKLİ PROTEİN AĞLARININ
ÇOK-ÇÖZÜNÜRLÜKLÜ GÖRSELLEŞTİRİLMESİ

Yaşar, Sevgi
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç. Dr. Tolga Can

Eylül 2009, 34 sayfa

Bilgisayar bilimi gözüyle baktığımızda, genom bazındaki protein etkileşim ağları binlerce düğümden oluşan ağ ya da çizgelerdir. Protein-protein etkileşim ağları bir genomun proteinleri ya da genleri arasındaki pek çok etkileşimi ifade etmektedir. Hücresel işlevlerin etkileşen proteinler tarafından yerine getirilmesi ve hücredeki temel yapıların etkileşen proteinler tarafından oluşturulması nedeni ile proteinlerin işlevlerinin öngörülmesinde protein etkileşim verisi hayati önem taşımaktadır.

Protein etkileşim öngörü ve analiz tekniklerindeki yakın dönemdeki artış genomlar için pek çok miktarda protein-protein etkileşim verisinin elde edilmesini sağladı. Bu da protein etkileşim ağları için sistematik bir görselleştirme ve analiz tekniğini kritik düzeye getirdi.

Bildiğimiz kadarıyla, protein-protein etkileşimlerini görselleştiren araçlar, protein etkileşimlerinin çok çözünürlüklü görselleştirilmesini dikkate almamaktadırlar. Tezimizde, protein-protein etkileşim ağlarını bileşik çizgeleri destekleyerek çok çözünürlüklü görselleştiren yeni bir yaklaşım gerçekleştirdik. Yaklaşımımız bileşik düzeneğe sahip çizgelerin gen ontolojisi açıklama analizi ile çok çözünürlüklü gösterimini desteklemektedir. Tezimizde protein-protein etkileşim ağlarının görselleştirilmesine yönelik yeni yöntemleri öneriyoruz.

Anahtar Kelimeler: Protein-Protein Etkileşim Ağı, Birleşik Çizgeler, Görselleştirme, Gen Ontolojisi, Gruplama

To Hayati Çam and Azra Sağlam

ACKNOWLEDGMENTS

I would like to thank my supervisor Asst. Prof. Dr. Tolga Can, for his enlighting ideas and solutions during problems, for his positive and supporting attitude during all master education period.

This thesis is partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) Career Program Grant #106E128.

I would like to thank my dear friend Burçin Sapaz for giving me motivating speech when I was down.

I would like to thank Dr. Sinan Kalkan for providing me a nice place to study, for his valuable supporting feedbacks and suggestions, for his help with latex and for his physcological support.

I would like to thank my boss Muharrem Görkem for his understanding during whole period.

I would like to thank my family for their love and support through all my life.

I would like to thank my brother Dr. Harun Servet Yaşar for waking me up every night and telling me I had to work everyday almost every day and hour.

I would like to thank Bahar Pamuk for the lovely dinners and relaxing chats.

I would like to thank my friends Gökhan Akça for lojistik support and for introduction to java.

I would like to thank Can Erogul for his encouring speeches.

I would like to thank my friends Gökçe Yıldırım, Duygu Saraçoğlu, Esin Saka, Tuba Bayık, Müge Çavdaroğlu, İrem Afşar and Burak Ateş for their endless support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Since main functions in a cell are preformed by proteins interacting with each other, protein-protein interaction data is an essential data for protein function prediction. Protein-protein interaction data can represent different types of interactions such as metabolic events, protein-protein/protein-nucleotide interactions, regulatory relationships or signaling pathways.

Genome scale protein-protein interaction data are large scale graphs/networks having thousands of nodes representing genes or proteins. Still, currently existing protein interaction data is incomplete. As stated by Hart et al. [8], the estimated number of interactions for a full yeast genome network is 37,800-75,500, whereas interactions found so far are less than 20,000. For human genome the estimated number of interactions is 154,000-369,000. Presentation of such a great scale of network is a challenge.

One aim of protein-protein interaction network visualization tools is to provide an insight to researchers to analyze a given genome's protein-protein interaction network for various interpretation purposes. There exist many PPI visualization tools, most of which initially provides a crowded view of a given PPI network.

Another aim is to combine different types of biological data while presenting the network data. So the aim of PPI network visualization tools is not only viewing a given network, but also presenting a view that includes some solid data obtained from domain specific data analysis, too. For example; some tools provide subcellular localization information such as [3], whereas some others use gene expression data [5, 16].

In this thesis, we introduce a new visualization approach providing compound graph architecture, and making use of gene set enrichment analysis. The aim of this thesis to provide a

layered (by making use of Markov Clustering) high performance viewing of protein interaction networks and support making use of Gene Ontology annotations and gene set enrichment during PPI visualization.

1.1 Related Work

1.1.1 Protein-Protein Interaction Visualization Tools

Genome wide protein-protein interaction networks are considered as very large graphs. Tools for visualization of regular graphs lack both performance and biological information for presenting PPI networks. As a consequence PPI visualization requires dedicated tools for presentation.

There exist many tools for PPI network visualization specialized on different purposes as reviewed by Suderman et al. [21]. The main issue to be considered is to present information with combination of analysis of related domain specific information in an explanatory, understandable, interactive manner with relatively high performance. Most widely used PPI visualization tools are as follows: Cytoscape [3, 9, 19, 20], Pathway Studio [16], VisANT [10], PATIKA [5], Arena3D [7]. Suderman et al. [21] in their study review the features possessed by existing visualization tools, and features for next generation visualization tools. Those features can be summarized as follows: network layout profiles, integration of analysis data into visualization, integration with external data resources, user inputs, and integration of third party software.

Among the PPI visualization tools, Cytoscape [20] is a powerful tool, allowing external developers to implement plugins having various features that would still use Cytoscape application on the background to work on [3, 9, 19]. For example, MiSink [19] plugin provides interfaces to well-known databases for retrieval of biological information. Cytoscape also presents a user friendly user interface, various layout formats, with capability to import a variety of interaction files. Cytoscape has several layout formats such as circular, force-directed, etc. Although stated layout formats present nice view for smaller networks such as pathways, it is difficult to interpret the provided view for large scale networks. Figure 1.1 and Figure 1.2 show overall and a zoomed portion view of yeast PPI network of 4932 nodes and 17491 edges on force directed and circular layout in Cytoscape. Orange colored portions indicate proteins,

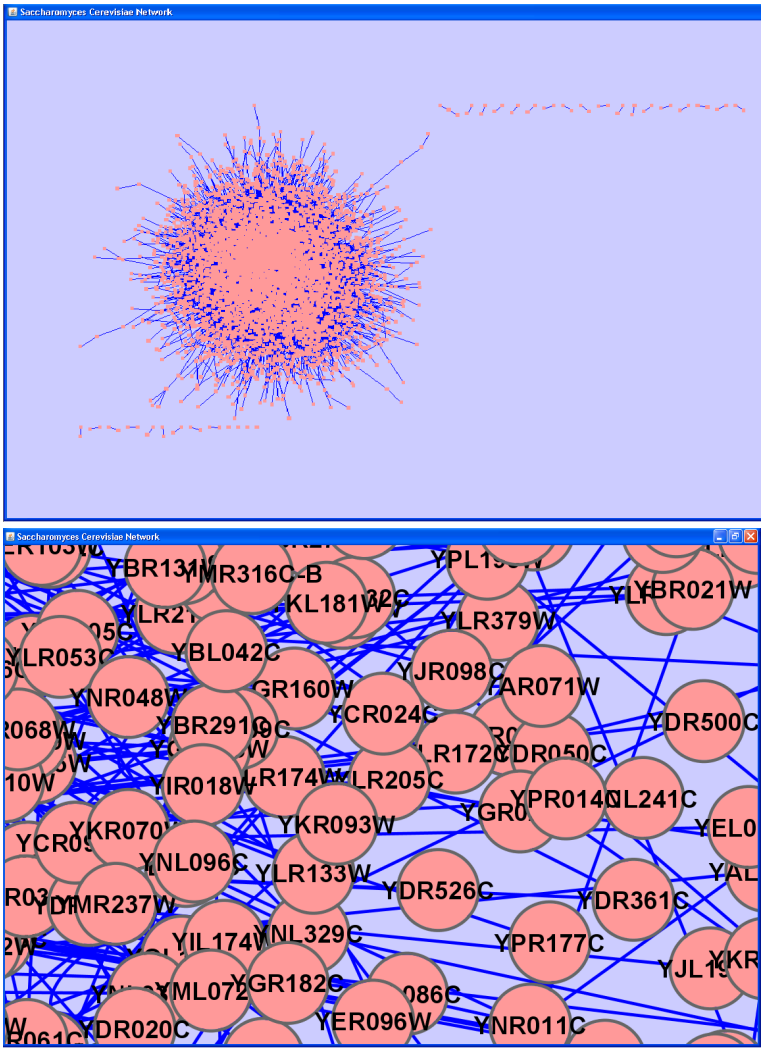

Figure 1.1: Complete and zoomed views of yeast PPI network with Cytoscape Force Directed Layout

whereas dark blue colored regions indicate interactions of the network.

Cytoscape also supports layout profiles of JGraph, however it takes at least 20 minutes to view a protein network of 4932 nodes and 17491 edges with those layouts on a hardware with Intel Core Duo CPU of 1.80 Ghz, and 2GBs of RAM.

Cerabral [3] plugin developed using Cytoscape API for plugin interface, working with base Cytoscape application provides layout of proteins relative to their localization in the cell by making use of subcellurlar localization annotations. However, as mentioned in [3], it takes 4.5 seconds to visualize a graph with 57 nodes and 74 edges, and 204 seconds to visualize 706 nodes and 1269 edges using Cerebral.

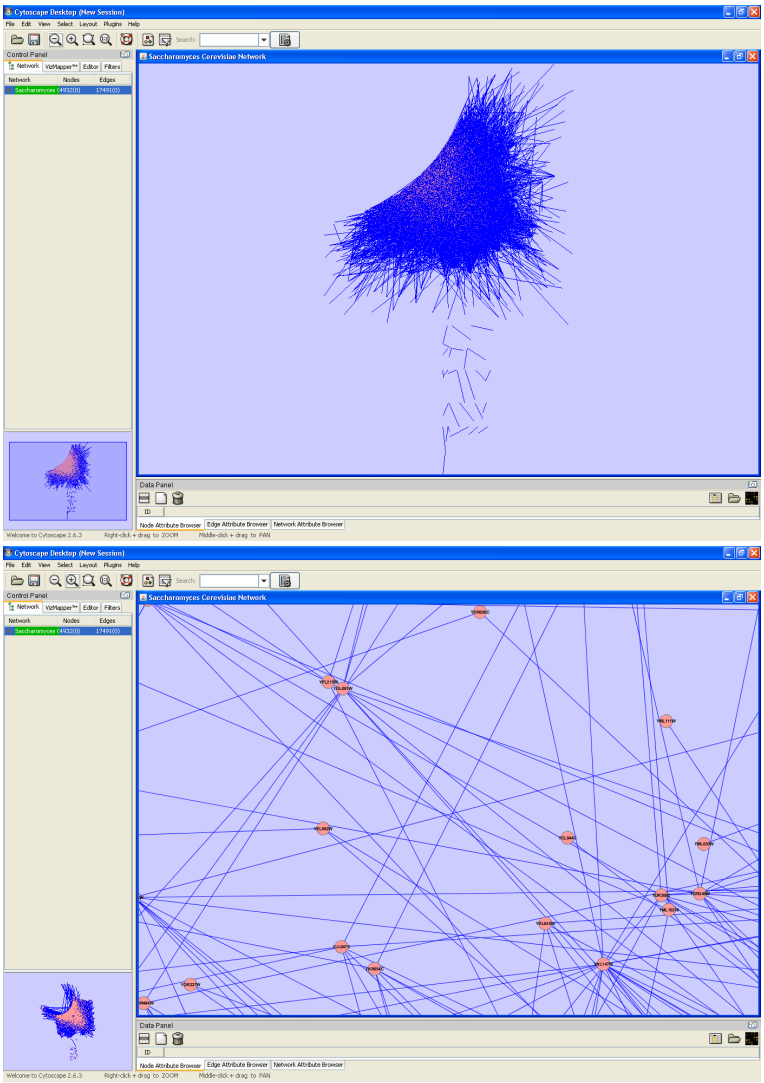

Figure 1.2: Complete and zoomed views of yeast PPI network with Cytoscape Inverted Self-Organizing Map Layout

VisAnt [10], another application for visualization and analysis of biomolecular interactions, can be used in three ways: as an online java applet[1], as a java web application[2], or as an offline client-side local application connected to main VisAnt server for data syncronization. VisANT provides annotations for large range of data sets by being integrated with standard databases such as GenBank, KEGG [16], and SwissProt. Being syncronized with databases, VisANT provides a query interface for variety of poteins and pathways. It provides a user friendly interface with an interactive environment. VisANT introduces the meta-node concept, by which a user can group a set of selected nodes and view it as a single meta-node. Various functionalities such as gene set enrichment with Gene Ontology annotations are provided for meta-data nodes. However since it is left to the user to select the nodes to be grouped, for large scale networks this approach does not seem to be feasible. Figure 1.3 depicts overall and zoomed views of yeast PPI network of 4932 nodes and 17491 edges on circular layout after approximately ten minutes of relaxing operation (a VisAnt functionality).

Pathway Studio [16], a windows desktop application commercially licensed to Ariadne Genomics Inc., is another application for visuzalition of mainly biological networks. It is capable of importing data of well known databases such as DIP [26], BIND [2] and KEGG [17]. It provides force-directed layout as automatic layout profile, as well as enabling user to interact and modify presentation of nodes; both in shape and location. However all these features are provided for biological pathways with gene expression data interpretation, large scale PPI network visualization is not the main concern.

PATIKA [5], standing for "Pathway Analysis Tools for Integration and Knowledge Acquisition", another web application for visualization of biological networks, like Pathway Studio, aims to visualize and analyze molecular pathways using gene expression data.

1.1.2 Using Clustering for Protein Functional Module Detection

Using graph clustering algorithms to detect functional modules of an organism has become a popular interest in bioinformatics. There have been studies on matching clusters obtained by a clustering algorithm to known functional modules. Aim of these studies is to find functional process or module of an unknown gene/protein by using general tendency of the genes in a

[1] http://en.wikipedia.org/wiki/Java_applet, Last accessed 01 Sep 2009
[2] http://java.sun.com/docs/books/tutorial/deployment/webstart/index.html, Last accessed 01 Sep 2009

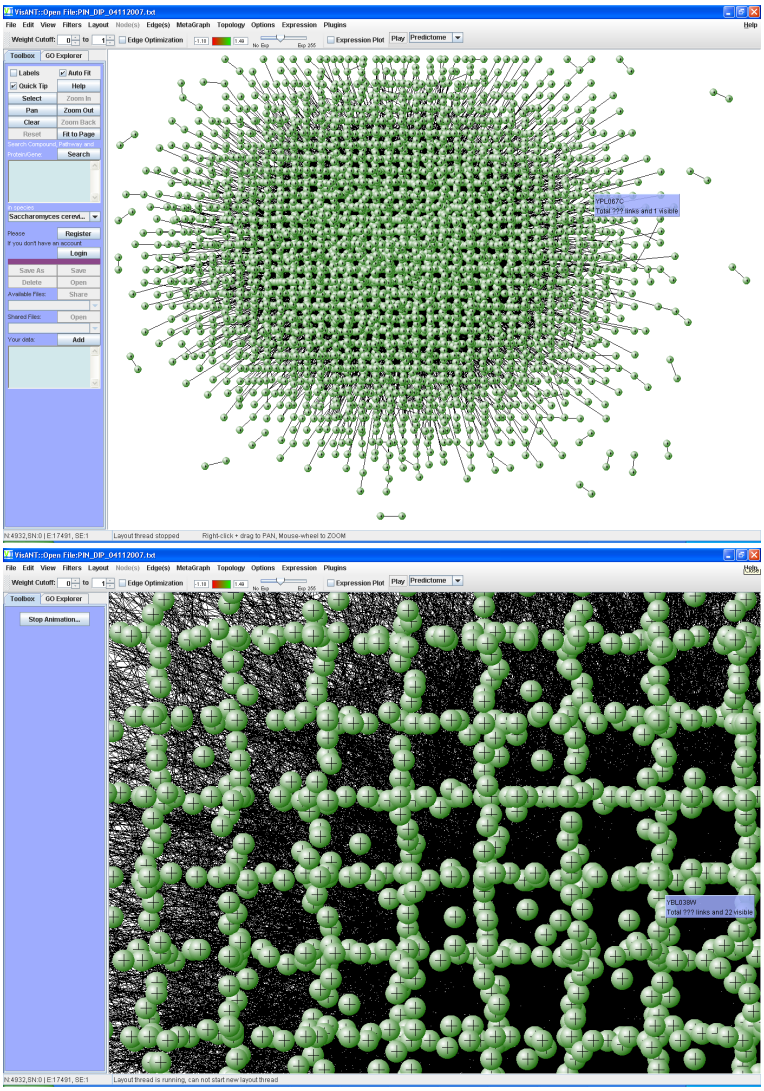

Figure 1.3: Complete and zoomed views of yeast PPI network with VisAnt Circular Layout relaxed approximately for ten minutes

cluster.

Zhang et al. [28] apply clique percolation algorithm to detect the dense clusters of a given protein-protein interaction network. Their main approach in detecting functional modules is to correlate each detected cluster to a specific well known functional module of an organism. There have been many studies using Markov Clustering (MCL) for PPI networks [6, 25]. Enright et al [6] uses MCL algorithm with modification so as to take preprocessed sequence similarity (based on BLAST) between each proteins in consideration while running Markov Clustering on a given PPI network. Vlasblom et al. [25], however, compares MCL with Affinity Propagation (AP) on PPI networks and they conclude that MCL procedure is significantly more tolerant to noise and behaves more robustly then AP especially for unweighted networks.

1.1.3 Gene Ontology for gene set enrichment on PPI networks

Gene set enrichment with Gene Ontology annotations is a recent study of bioinformatics. The main process is detecting the most significant Gene Ontology term for a given set of genes. There have been different approaches for determination of the most significant GO annotation [1, 14]. During our study we used the topGO method of Alexa et al. [1] for finding most significant Gene Ontology term for a cluster. Using Gene Ontology annotation for clusters of protein interaction networks is a recent study. Marco et al [15] recently focused on correlation of interactome hierarchy with Gene Ontology hierarchy. For their study, they used the UV cluster algorithm to construct clusters as a hierarchically organized tree for Saccharomyces cerevisiae(yeast) interactome. They check whether there would be a cluster such that it includes all proteins of a parent GO term (nine parent GO terms are used for that study). Rate of positive detected enrichments: number proteins having their enriched GO terms as the same as their actual GO term is on the average 80.1% (62-96%). Marco et al [15] additionally state GO coverage (extent a given GO term is detected in the interactome) on the average as 51.2% (34-67%), having the highest coverage at Biological Process Gene Ontolgy. The results studies of Marco et al [15] show that GO and PPI network hierarchy are similar for global structure of PPI networks. We expect usage of GO annotations for clusters to be valuable information for PPI network presentation.

1.2 Contributions

Our contributions in this thesis can be listed as follows:

1. By providing multi-resolution (compound) structure, we aim to provide an organized view to the end user. Unlike the related studies, this approach provides much less crowded view to the end user. Another contribution coming along with this is performance. Most related studies suffer from preparation of a layout for a large-scale network, for not much organized layout profiles, it does not take much time. However even for a little bit sophisticated layout profiles, it takes hours to present a given cluster. By providing an initial organized view with much less nodes and edges we present a layout for a given graph immediately. Afterwards, with the mlti-resolution feature, having a layout for more detailed views is still time efficient.

2. Our approach makes it easier for the end user to make use of provided biological information. Although many related works provide even more biological information to the end user, presenting this information in a crowded view makes it difficult for the end user to interpret the information. With GO term annotations for clusters of each layer, an initial biological foreknowledge is provided for structuring of given PPI network. With this information user can decide on the region of interest to focus on.

3. Our approach provides a test platform for biological network clustering approaches. There has been studies on checking compatibility of PPI network clustering approaches with the nature of PPI networks. Clustering results of related studies can be loaded to our proposed tool and their compatibility with GO annotations can be investigated.

1.3 Thesis Outline

This thesis is organized as follows: In Chapter 2, necessary background information to understand the problem and the problem domain is provided. In Chapter 3, general solution approach is explained by first giving a brief overview about the general architecture of proposed system and consequently describing each component of the system seperately in detail. In Chapter 4 experimental results of the proposed solution and related discussions are given. Finally, in Chapter 5 thesis is concluded with a brief summay and suggested future works.

CHAPTER 2

BACKGROUND

2.1 Proteins

Proteins are one type of the basic organic compounds of the cell which play significant roles in characterization and functionality of the cell. Proteins are basically linear forms of amino acids joined by peptide bonds.

The main reason behind significance of proteins is their constructions going back to DNA. The coding part of a gene on DNA is transcribed to its complementary RNA (mRNA), and using genetic codes on mRNA, Ribosome forms peptide bonds between amino acids at translation. During or after its synthesis, a protein starts to fold into its shape.

2.2 Protein-protein interaction networks

Protein-protein interaction plays significant roles in understanding functionality of a protein. Protein interaction networks are graphs in which nodes represent proteins and edges represent either physical or computationally computed interactions between proteins.

Protein interactions can represent not only physical interaction of proteins to form large complexes but also cell signaling pathways. Protein interactions can be obtained by various methods as also listed in [12, 22].

There are various protein-protein interaction databases at which protein interaction data of the organisms that are most widely used for bioinformatics experiments are readily available. Some of them can be listed as:

- DIP[1], Database of Interacting Proteins [26], combines data from various sources, provides protein interaction data that is both manually (by expert curators) and automatically (by computational approaches) curated.
- MIPS[2], Mammalian Protein-Protein Interaction Database [18], provides collection of manually curated high-quality PPI data collected from the scientific literature by expert curators. This database includes only data from individually performed experiments since they usually provide the most reliable evidence for physical interactions.
- BioGRID[3], The Biological General Repository for Interaction Datasets, provides collections of protein and genetic interactions from major model organism species derived from both high-throughput studies and conventional focused studies.
- BIND[4], Biomolecular Interaction Network Database [2], includes not only interactions among proteins but also interactions betweens proteins and RNA, DNA, and small molecules.

2.3 Clustering

Clustering is one of the fundemantal unsupervised learning problems, dealing with determination of intrinsic groups of given unlabeled data. For a given data composed of objects to be clustered, a cluster is a set of objects which are more similar to each other than the objects in another cluster in some kind of distance measurement sense. This distance measurement may be one of the common measurements such as Euclidean distance (2.1a), Manhattan distance (2.1b), or another user defined distance metric.

$$D_{euc}(x_i, x_j) = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk}\right)^2} \qquad \text{where } x_i = (x_{i1}, x_{i2}, ..., x_{in}) \tag{2.1a}$$

$$D_{mhtn}(x_i, x_j) = \sum_{k=1}^{n} \left|x_{ik} - x_{jk}\right| \qquad \text{where } x_i = (x_{i1}, x_{i2}, ..., x_{in}) \tag{2.1b}$$

Clusters does not necessarily represent different classes of given data that are very distinct from each other, but also can represent large units of given data; a cluster being represen-

[1] http://dip.doe-mbi.ucla.edu, , Last accessed 01 Sep 2009
[2] http://mips.gsf.de/proj/ppi, Last accessed 01 Sep 2009
[3] http://www.thebiogrid.org/, Last accessed 01 Sep 2009
[4] http://www.bind.ca, , Last accessed 01 Sep 2009

tative of other clusters of the given data. Clustering approaches can be classified as follows: hierarhical clustering (such as Single-Link Agglomerative Hierarchical Clustering), and partitional clustering (such as K-means Clustering, Expectation-Maximiation and DBSCAN)[13] .

2.4 Gene Ontology (GO)

Aim of an ontology is to define common concepts and their relations with each other for certain domains. The Gene Ontology Consortium was formed to develop shared, structured vocabularies adequate for the annotation of molecular characteristics across organisms [4]. Gene Ontology defines biological vocabularies to annotate genes and gene products. Gene ontology also defines relationship of terms with each other.

Gene Ontology Consortium defines three ontologies: Molecular function (MF), Biological Process (BP) and Cellular Component (CC) ontologies. "Molecular Function" ontology defines the biochemical responsibility of a gene/gene product, such that some proteins are specialized to be enzymes. For example: GO term for 'carbonate dehydratase' is 'GO:0004089'. "Biological Process" ontology defines the biological process or the biological pathway the gene/gene product involved. For example: GO term for 'signal transduction' is 'GO:0007165'. "Cellular Component" ontology defines the cellular location at which the gene/gene product is active at. It basically defines localization information for the given gene product, example: GO Term for 'Golgi apparatus' is 'GO:0005794'.

Gene Ontologies are structured as directed acyclic graphs (DAGs) having nodes as GO terms; each term can have one or more parent terms. Figure 2.1 displays a sample Cellular Complex ontology graph[5] queried for Golgi apparatus of yeast, *Saccharomyces cerevisiae*.

2.5 Gene set enrichment with GO ontology

Microarray experiments provide list of genes which are differently expressed (having statistical score over a threshold) at a cell. In order to interpret the obtained differently expressed genes and understand their functionality, supportive data like biological process or molecu-

[5] http://www.yeastrc.org/pdr/viewGONode.do?acc=GO:0005794, , Last accessed 01 Sep 2009

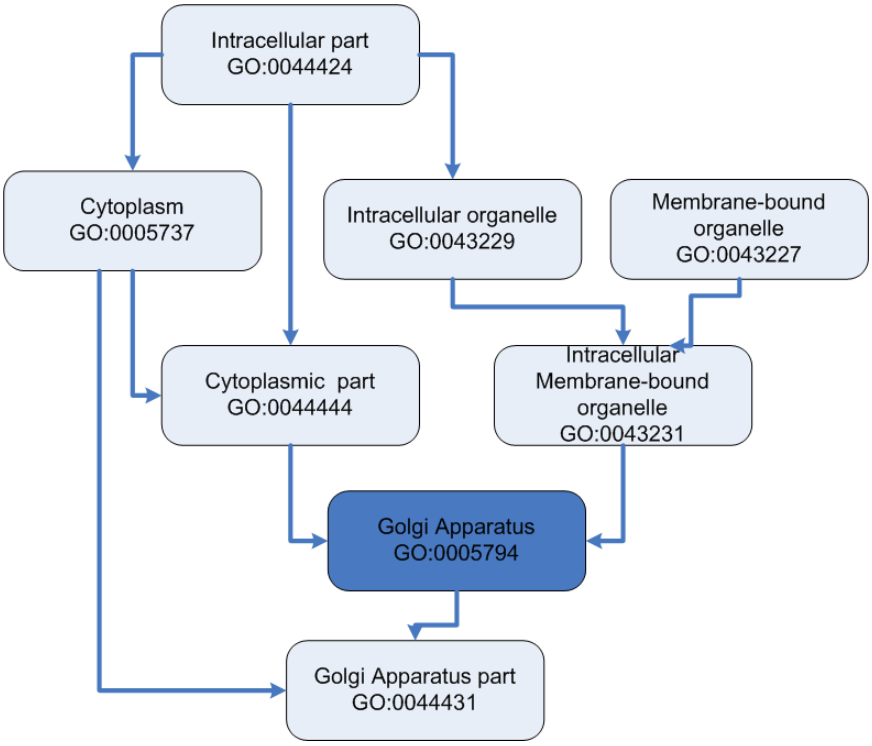

Figure 2.1: Sample GO hierarchy for for Golgi Apparatus

lar function of the set of genes is sufficient. There have been studies on making gene set enrichment wit GO terms [1, 14]. In our study, we decided to use the approach of [1] for determination of significant GO terms.

2.6 Statictical Tests for Significance Calculations

In order to determine the general behavior or general structure of a set of genes, significant properties of the given set shall be determined. Therefore, many gene set enrichment analysis methods are based on statistical test functions for determination of statistically significant properties of a given set. *topGO* package has support three types of statistical tests for significance alaysis: Fisher's exact test, Kolmogorov Smirnov test, and Student-t test. Student's t-test, one of the statistical significance tests, proposes consideration of differences between the means of two groups relative to their scores' variance. KS test is mainly used to estimate a minimum distance for probability distribution of a sample with a reference probability distribution or to estimate the minumum distance for two samples. Fisher's exact test is based on binary relationships between samples. [1] focuses on Fisher's exact test for significance

analysis in their study. Therefore, we also decided to use Fisher's exact test during our tests.

2.6.1 Fischer's Exact Test

Fisher's exact test is a statistical significance test used for analysis of categorized samples in small sizes. Fisher's exact test is based on a contingency table representing the binary relationship between variables. Table 2.1 depict a sample contingeny table for two categories having two samples s.t. A1, A2 representing samples for category A and B1 and B2 representing samples for category B. For example, assume that the collected data is for a set of people, s.t. category A representing people being either male (A1) or female (A2); category B representing people either owning a house (B1), or renting a house (B2).

Table 2.1: Sample contingency table for Fisher's test

	B1	B2	Total
A1	a	b	a+b
A2	c	d	c+d
Total	a+c	b+d	n

Probability values indicated at contingency tables are assumed to be obtained by hypergeometric distribution[6]. Equation 2.2 shows the probability of obtaining a particular arrangement for association of a class with the another one. For example, probability of having people owning a house are distributed evenly among males and females.

$$p = \binom{a+b}{a}\binom{c+d}{c}\Big/\binom{n}{a+c} = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!a!b!c!d!} \tag{2.2}$$

2.7 R

R [11] is a programming language which is widely used for statistical software development and analysis. Development of R is highly influenced by two languages S, which is also a statistical programming language and Scheme. The main idea behind R development is having an interface and syntax like S, but having implementation and semantics of Scheme beneath.

[6] A discrete probability distribution that describes the number of successes in a sequence of n draws from a finite population without replacement

Since bioinformatics requires processing of great amount of data, statistical computing becomes inevitable, and consequently that makes R popular in bioinformatics. R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form[7].

[7] http://www.r-project.org/, Last accessed 01 Sep 2009

CHAPTER 3

ARHITECTURE AND METHODS

3.1 Overview and General Architecture of the System

The main aim of our approach for visualizing a given protein-protein interaction network is to obtain different view resolutions for a given network. In order to obtain different resolutions, we decided to obtain different clusterings for a given data and match them to each layer of resolution one by one. Another aspect of our solution is to annotate GO terms to clusters of genes at each layer of resolution. Our solution for such an environment is depicted in Figure 3.1.

In order to obtain different layers of resolutions, we decided to obtain different clusterings by using the Markov Clustering Algorith (MCL) (see Section 3.4). Gene set enrichment for clusters of a given layer is obtained using the topGO package (see Section 3.3). PPI Visualizer mainly integrates already available components in order to visualize a given protein-protein interaction network.

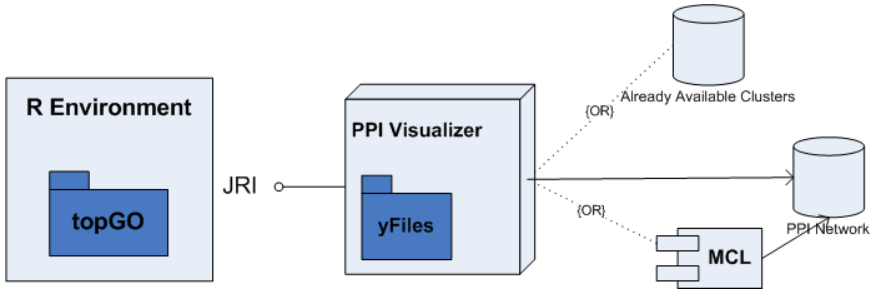

Figure 3.1: General Architecture of the System

3.2 Data Selection

Except for the significant GO term determination phase, our approach does not depend on the type of the organism to be visualized. However, we decided to focus on visualization of yeast, Saccharomyces cerevisiae, PPI network as a proof of concept study. There are four large databases having protein-protein interaction data for Saccharomyces cerevisiae, as also listed in [27]. Among these, we decided to use data from DIP [26] for our tests, since it includes both experimentally and computationally PPIs.

3.3 Gene Set Enrichment with topGO

For gene set enrichment with GO terms we decided to use the package *topGO* developed by Alexia et al. [1]. *topGO* package is available as Free Software [1]. *topGO* package for gene enrichment analysis is developed in R language and needs R environment in order to be used. *topGO* installation is described in Appendix A.2.1. We integrated R engine to java platform using JRI (Java R Interface)(described in Appendix A.1) and ran R scripts for gene set enrichment analysis over JRI interface.

topGO package provides algorithms for finding significant Gene Ontology terms for a given set of genes. As it is explained in Section 2.4, there are three types of ontologies: Molecular function (MF), Biological Process (BP) and Cellular Component (CC). *topGO* package is capable of finding significant GO terms for each category.

In order to find significant genes for a cluster on a specified Gene Ontology category, statistical tests for significance (see section 2.6) to be used for gene set enrichment shall be specified. *topGO* package provides four statistical tests for GO term annotations: Fisher's exact test (Section 2.6.1), Kolmogorov-Smirnov test, t-test, and Goeman's global test.

After generating a *topGOdata* instance for a specified Gene Ontolgy category, and specifying the statistical significance test to be used for annotations, top n GO term annotations for a given cluster can be obtained. We decided to use Biological Process as the Gene Ontology category, and Fischer's exact test as the statistical test for significance during our tests. We labeled the clusters at each resolution with the most significant GO-term obtained for each

[1] http://www.bioconductor.org/packages/2.5/bioc/html/topGO.html, Last accessed 01 Sep 2009

cluster.

3.4 Markov Clustering (MCL)

Markov Clustering [24] is a clustering algorithm aimed to find natural structure of a given graph. The idea beneath Markov Clustering is obtaining clusters such that the probability getting to a node in the same clusters with k-length (or k-step) paths is higher than the probability of getting to a node in a different cluster with k-length path. So, MCL clusters are assumed to be dense regions, expected to satisfy a random walk from a node in an MCL cluster to end in the same cluster.

Markov Clustering paradigm found by Stijn van Dongen [24] is based on simulating flow through a graph, to promote flow where the current is strong, and to demote flow where the current is weak. The underlying clusters will reveal as the current between clusters would disappear by the time. In order to simulate the flow, the given graph is transformed into Markov graph[2] and then to Markov Matrix (column stochastic matrix).

Definition 3.4.1 *Markov Matrix (Column Stochastic Matrix) $M \in \mathbb{R}^{nxn}$, $M \geq 0$, of a given graph of n nodes is:*

$$\mathbf{M_{nxn}} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{bmatrix} \qquad \textit{such that,}$$

$$\forall j \leq n \qquad \sum_{j=1}^{n} \left(x_{1j} + x2j + \ldots + x_{nj} \right) = 1 \qquad \textit{shall be satisfied.}$$

Each column of the stochastic matrix represents probability values for a departing node of the given graph. Each value on the column corresponds to the probability of flow going from departing node to other nodes. In case there is no connection between the nodes, the probability would be zero.

Flow simulation is called *Expansion* operation of MCL. [23, 24] empowers the paradigm with another operation called *Inflation*. MCL process application is alternation of *Expansion*

[2] Markov Graph: graph having total weight of outgoing edges of any node equals to one

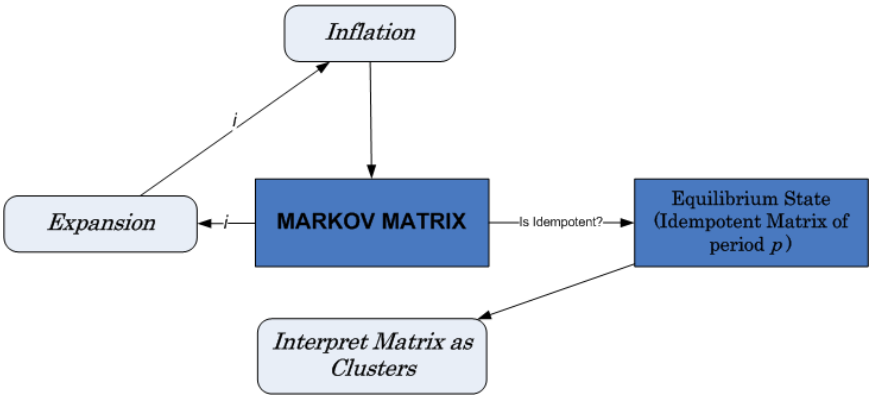

Figure 3.2: MCL Algorithm

and *Inflation* operations on a given graph. *Expansion* operation is mathematically taking square power of the stochastic matrix[3]. Whereas *Inflation* operation is mathematically taking Hadamard-Schur[4](element wise multiplication) power of the stochastic matrix, and then in order to preserve stochastic nature of the matrix, scaling the result using a diagonal matrix.

Definition 3.4.2 *For a given network of n elements and its corresponding stochastic matrix* $M \in \mathbb{R}^{nxn}$, $M \geq 0$ *and inflation parameter* $r \in R$, $r \geq 1$, *inflation on elements of M is defines as follows:*

$$(\Gamma_r M)_{pq} = M_{pq}^r / \sum_{i=0}^{n} \left(M_{iq}\right)^r$$

Since the performed operations are element-wise multiplication and scaling, Γ_r *Inflation* operator does not have any effect on dimensions of the given matrix.

As mathematical structure of the operators also suggest, *Expansion* operation updates the probabilities for each node pair, generating flow from one to the other for all connected nodes of the graph. *Expansion* tries to maximize k value for random walk from any node of the given graph. In some sense, *Expansion* operation resists seperation of clustering. On the contrary, as mathematical nature of *Inflation* operation also suggests, *Inflation* operation by making entrywise multiplication of probabilities, lowers probabilities of random walks among clusters. The vital step of MCL process is iterative application of *Inflation* and *Expansion* operations on a given graph, until an equilibrium state is reached. Figure 3.2 summarizes the MCL algorithm. Iterations reveal underlying natural structuring of the graph.

[3] http://en.wikipedia.org/wiki/Matrix_multiplication, Last accessed 01 Sep 2009

[4] Hadamard-Schur Transform: For $A \in \mathbb{R}^{mxn}, B \in \mathbb{R}^{mxn}$ then $(A.B) \in \mathbb{R}^{mxn}$ where $(A.B)_{i,j} = A_{i,j}.B_{i,j}$

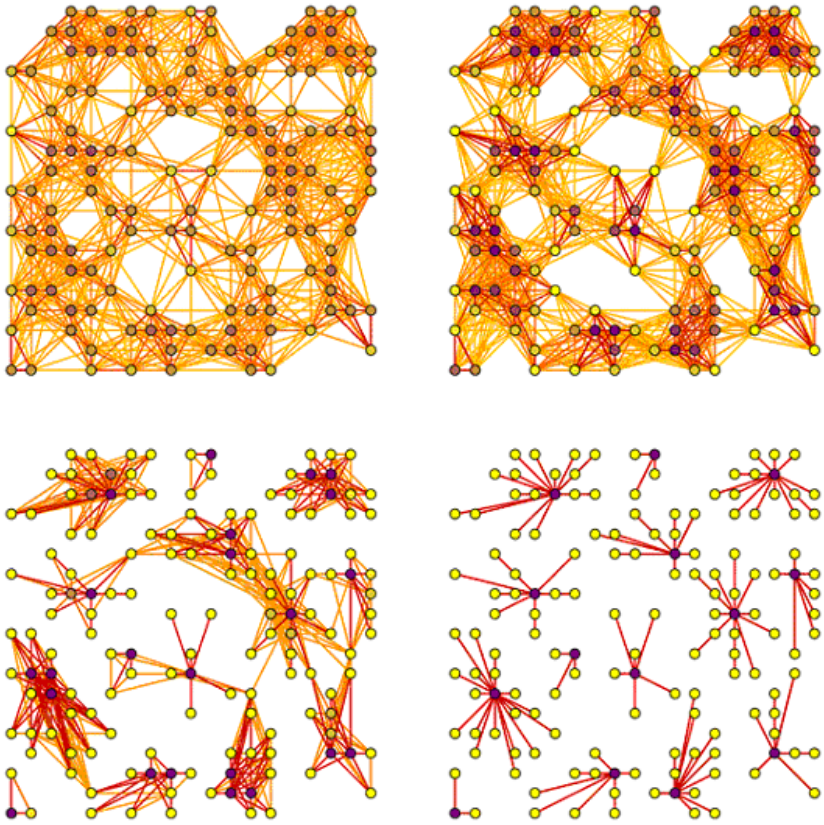

Figure 3.3: Seperation of graph during MCL iterations, from http://www.micans.org/mcl/

During iterations, the k value, for k-length path among nodes decreases, resulting in intra-cluster connections between nodes to dissappear over time. Nodes that are densely connected and the nodes having more inter-connections are expected to preserve their connection longer, leading to star-like appearance of the clusters. Figure 3.3 depicts process of MCL over iterations, it also confirms star-like structuring of the clusters over the time.

3.4.1 Determination of Clusters to be Used for Resolutions

Enright *et al* [6] observes for values of $r > 1$, inflation changes the probabilities of random walks from a particular node to destination nodes as favoring more probable walks over less probable ones. For clustering of yeast, Saccharomyces cerevisiae, we observed similar situation. Figure 3.4 show the cluster count values for r values $0.1 \leq r < 1.0$ and $1.1 \leq r \leq 2.0$. In order to determine the clusterings to use for resolutions, we calculated variance for each clustering. Figure 3.5 shows variances for each clustering obtained for r values of $1.1 \leq r \leq 2.0$.

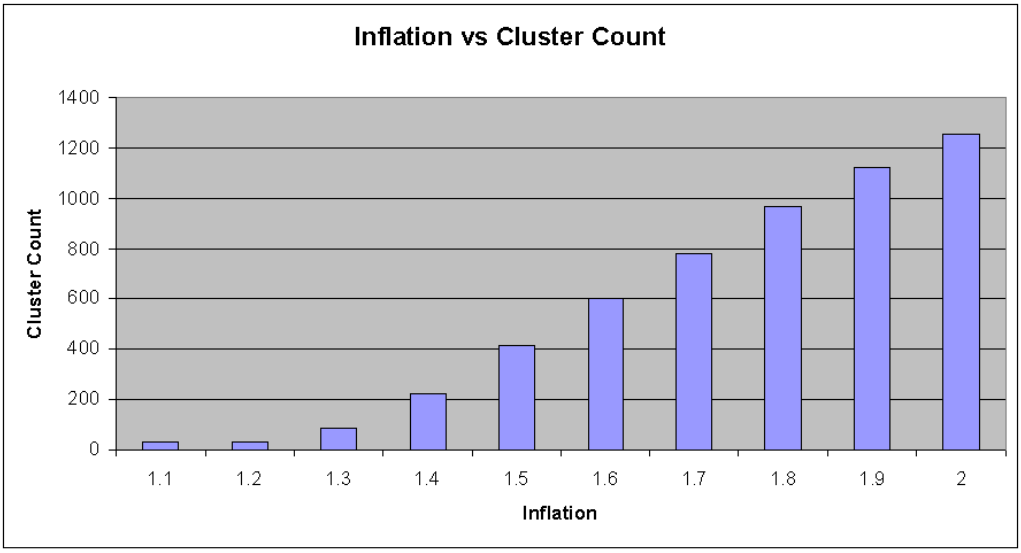

Figure 3.4: Inflation vs cluster counts

j

Based on these observations, we decided to use clusterings for r values of $1.1 \leq r \leq 2.0$ as resolution layers during visualization.

3.5 PPI Visualizer

Core PPI Visualization module is responsible for integration of submodules such as Markov Clustering (MCL), R along with *topGO*, and presentation of the the PPI network with combination of related data from MCL and GO annotations from *topGO*. Application of MCL algorithm on a given network is actually a preprocessing stage for PPI visualization. One of the reason behind the selection of MCL algorithm for clustering is its time performace. Compared to most well-known clustering algorithms, MCL produces clusters in much less time. For *Inflation* $= k$, $1.1 \leq k \leq 2.0$, MCL clusters becomes more granular as k increases. Another reason behind using MCL algorithm for clustering is having clusters of different granularity per inflation parameter.

In order to construct layers of resolutions, *Inflation* parameter of MCL is adjusted to obtain different clusters from the same graph. Even though the nature of MCL process is close to being nested, MCL clusters obtained with inflation parameter adjustment are not nested. To obtain a compound graph structure, obtained clusters are converted to layers of a nested

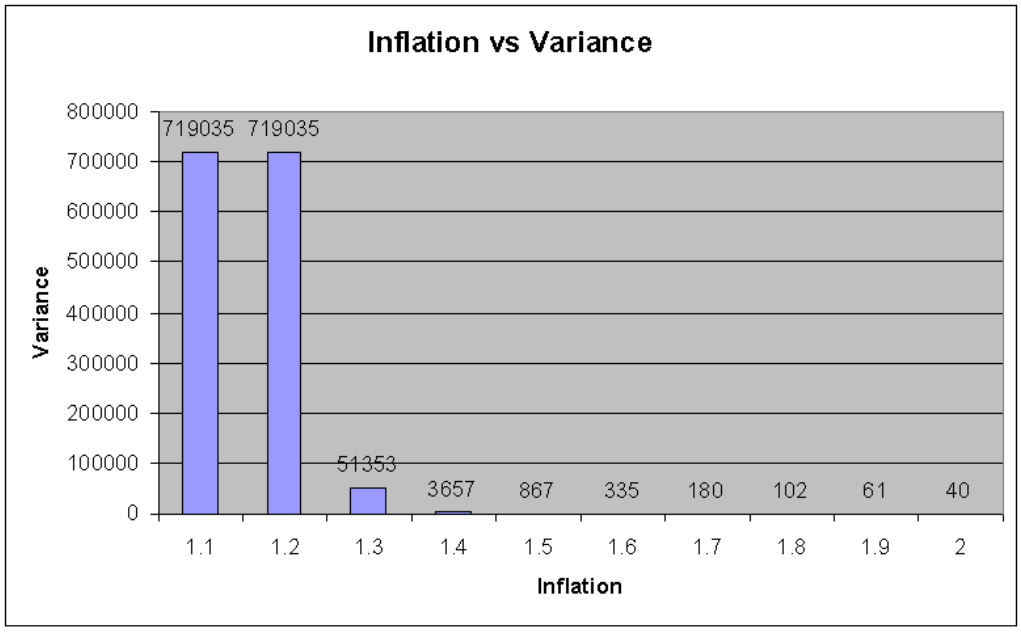

Figure 3.5: Inflation vs Cluster Counts

hierarchy. Given a graph G, a node $n \in G$, resolutions level, $l \in Nandl > 0$, where $l = 0$ indicates the given graph), the cluster n belongs to at resolution level $l + 1$, $C_{l+1}(n)$) is the most common cluster for level $l + 1$obtained for the elements of the previous level $C_l(n)$. For that purpose starting from most granular clusters obtained (k values close to 2.0) to more compact clusters(k values close to 1.0) clusters are organized. By doing this, compound nodes at each resolution forms compound graphs relative to resolution layer.

Depending on offline or online cluster loading feature, Core PPI visualizer is capable of integrating preprocessing stage with visualization stage. For this purpose, PPI Visualizer generates seperate threads for each MCL run. Loading clusterings as different files is supported as well.

After the preprocessing stage, PPI visualizer loads all layers obtained with calls to R-Engine in parallel giving the upmost level the highest priority. For each cluster, most significant GO-term depending on the Gene Ontology category (MF, BP, CC) settings is retrieved with function calls over JRI to R-Engine. As layers are constructed, the view is updated accordingly with Organic layout profile, meanwhile allowing end-user interactions with the graph.

We used yFiles library data structures and API for graph visualization(Section 3.5.1). yFiles'

approach for graph visualization is based on the Model-View-Controller paradigm. For graph layouts, we used layout algorithms of yFiles.

3.5.1 yFiles

yFiles[5] is a 2D graph visualization library which provides a very rich application programming interface (API) that handles most functionalities that would be necessary for graph drawing and diagramming.

It is basically composed of three main sub components:

- ”Basic” which serves as the ”backbone” for the main part of the library including the main structure ”Graph”. It also provides variety of functionalities in addition to various network and graph algorithms such as search algorithms, hashing functions etc.
- ”Viewer” which provides necessary interfaces for the user interaction related events to be handled at the core ”Graph” side. It also provides standard graphical user interface application related API.
- ”Layout” which provides interfaces for well-known basic graph layout algorithms and their customizations. Layout algorithms provided by this component does not only include node placement algorithms but also edge routing algorithms.

Requirements and restrictions for the usage of yFiles can be stated as follows[6]:

- There must exist a valid instance of type ”Graph” to create any graph elements at all.
- There is no way to create a node or an edge ”outside” a graph.
- All created graph elements instantly belong to a distinct graph.

[5] http://www.yworks.com/en/products_yfiles_about.html, Last accessed 01 Sep 2009
[6] http://www.yworks.com/products/yfiles/doc/developers-guide/index.html, Last accessed 01 Sep 2009

CHAPTER 4

RESULTS

4.1 Preprocessing Stage

Although it is possible to run MCL and *topGO* during visualization, in order to make performance dedicated to visualization, we obtained MCL clusters and *topGO* values at the preprocessing phase, and use the results during visualization.

4.1.1 Preparation of Clusters

Table 4.1 lists duration it takes to cluster yeast PPI network data per inflation. As for $r > 1$, MCL starts to favor connections departing from a node. For r values converging to 1, inflation is high, favoring intra-cluster connections, making it difficult for expansion to partition the given graph. That is why for r values close to 1, it takes more iterations to cluster a given network and consequently more time.

Table 4.1: MCL Duration (in sec) per inflation

r	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0	2.1	2.2	2.3	2.4
t	1155	758	452	244	163	112.5	85	67.5	55.5	45.5	38.5	34	29.5	26.5

4.1.2 GO Term Annotation

As we mentioned at Section 3.3, *topGO* package runs significant GO-Term analysis for a given set of genes. Table 4.2 lists the ten most significant genes obtained for the first of the clusters obtained for inflation $r = 1.3$. Obtaining significant terms as in Table 4.2 for a

cluster takes approximately 30 seconds. As Figure 3.4 implies, as the inflation(r) increase the clusters become more granular, leading to an increase in cluster count per MCL run. For $1 < r \leq 2$, average value of 550 clusters per MCL run (including stand alone nodes that do not belong to any cluster). So, it would take approximately 265 hours for *topGO* to complete calculation. To overcome this situation, instead of calculating most significant GO term for all clusters, we decided to discard GO term annotation for small scale clusters. So, we decided to annotate GO-terms to clusters having sizes greater than a threshold. We labeled each annotated compound node with the most significant GO term for that cluster (GO.ID of the first row).

Table 4.2: Top ten significant GO terms for a cluster of 1872 genes

	GO.ID	Term	Annotated	Significant	Expected	pValue
1	GO:0000398	nuclear mRNA splicing, via spliceosome	99	72	30.16	3.0e-18
2	GO:0000377	RNA splicing, via transesterification re...	100	72	30.46	7.6e-18
3	GO:0008380	RNA splicing	134	87	40.82	9.1e-17
4	GO:0000375	RNA splicing, via transesterification re...	105	72	31.99	5.5e-16
5	GO:0051234	establishment of localization	1224	480	372.88	9.4e-14
6	GO:0051179	localization	1293	503	393.90	9.6e-14
7	GO:0006810	transport	1203	468	366.48	1.1e-12
8	GO:0048193	Golgi vesicle transport	183	98	55.75	3.4e-11
9	GO:0006888	ER to Golgi vesicle-mediated transport	88	56	26.81	1.0e-10
10	GO:0016192	vesicle-mediated transport	367	167	111.80	2.6e-10

4.2 Visualization

For visualization of PPI network yeast, we selected clusters for inflation parameters (1.1, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8) to be the resolutions with cluster with $r = 1.1$ being the upper most, and cluster with $r = 1.8$ being the down most resolution layers. Figure 4.1 depicts the most dense clustered view the yeast PPI network, having actual non-connecting regions differentiated as clusters.

Going some level deeper, for $r = 1.4$, we decided to set count limit for clusters such as for clusters having size less than four nodes are not allowed to be a compound node. Figure 4.2 depicts this scenario, whereas Figure 4.3 presents one by one clustering match with the MCL results. Looking at these snapshots, we can say MCL results in many small scale clusters. Even when small sized clusters are unfolded, the network becomes to look more complex because of the number of small sized clusters. So folding all clusters still seems to give better results.

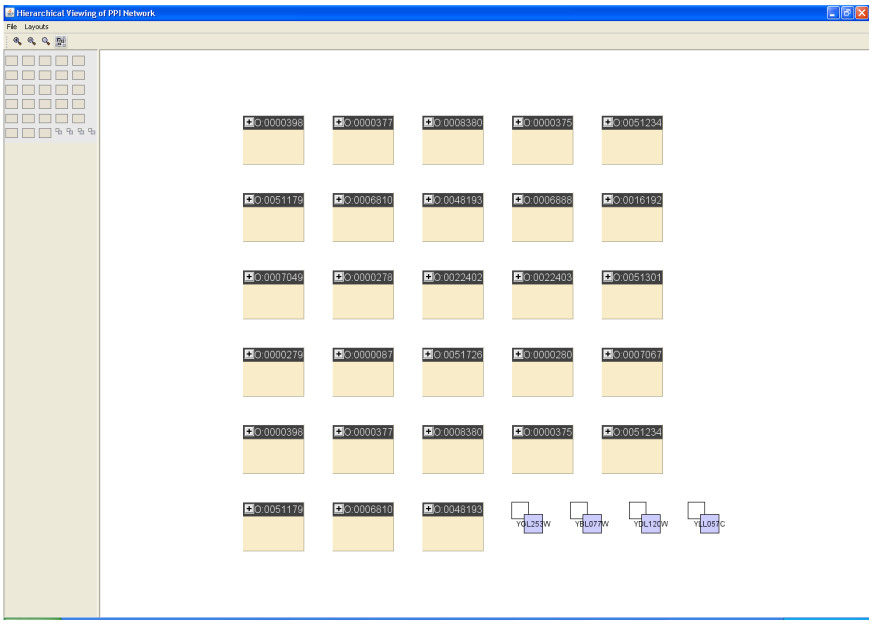

Figure 4.1: Yeast PPI network for $r = 1.1$

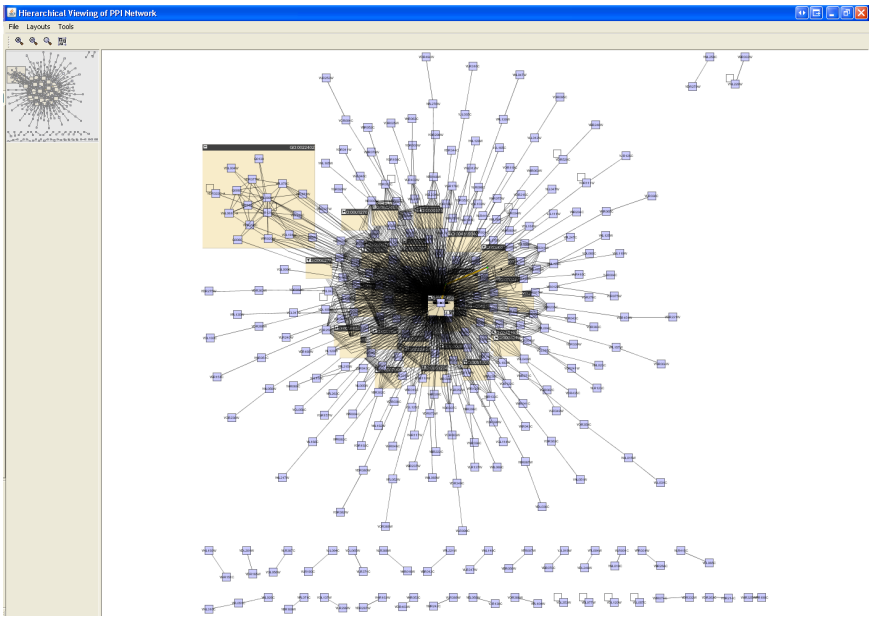

Figure 4.2: Yeast PPI network for $r = 1.4$ with compound nodes having at least 4 nodes

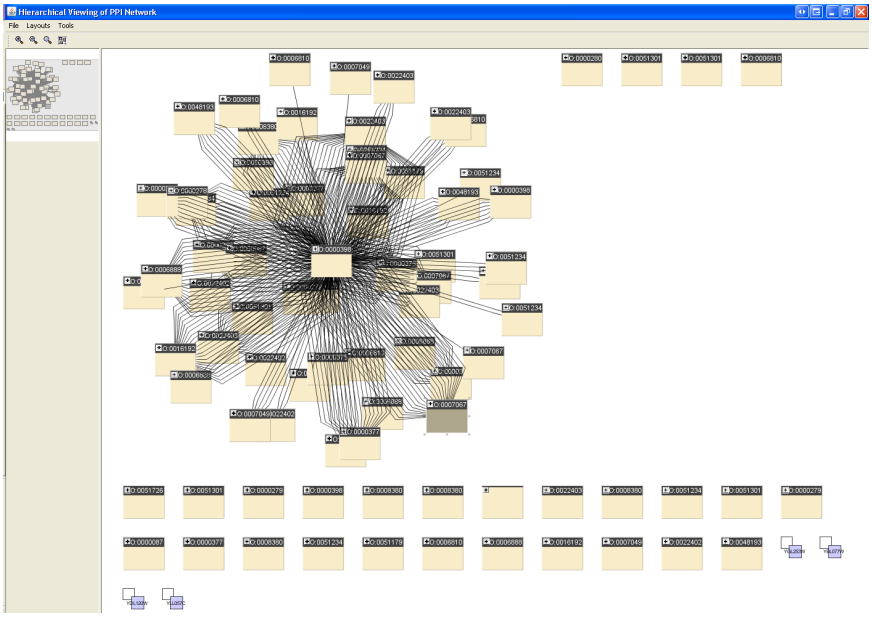

Figure 4.3: Yeast PPI network for $r = 1.4$

Since the compound nodes include subgraph in the inner side, for interesting GO Terms, navigating to inner graph gives better view of the compound node (See Figure 4.4). In order to observe a compound node's sub nodes' interactions with complete graph, unfolding a compound node within the complete is another sufficient functionality for PPI visualization (See 4.5).

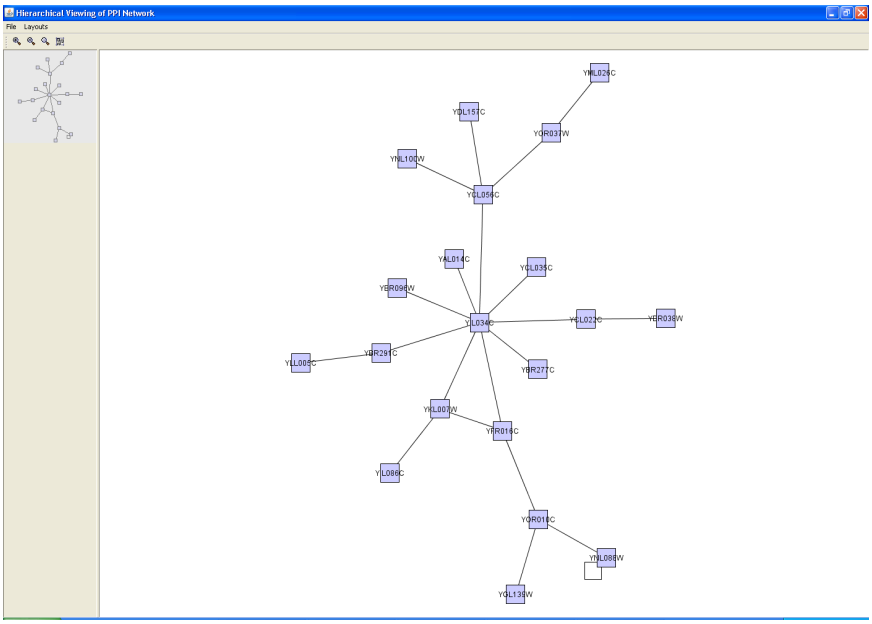

Figure 4.4: Yeast PPI network for $r = 1.4$ navigation to inner graph

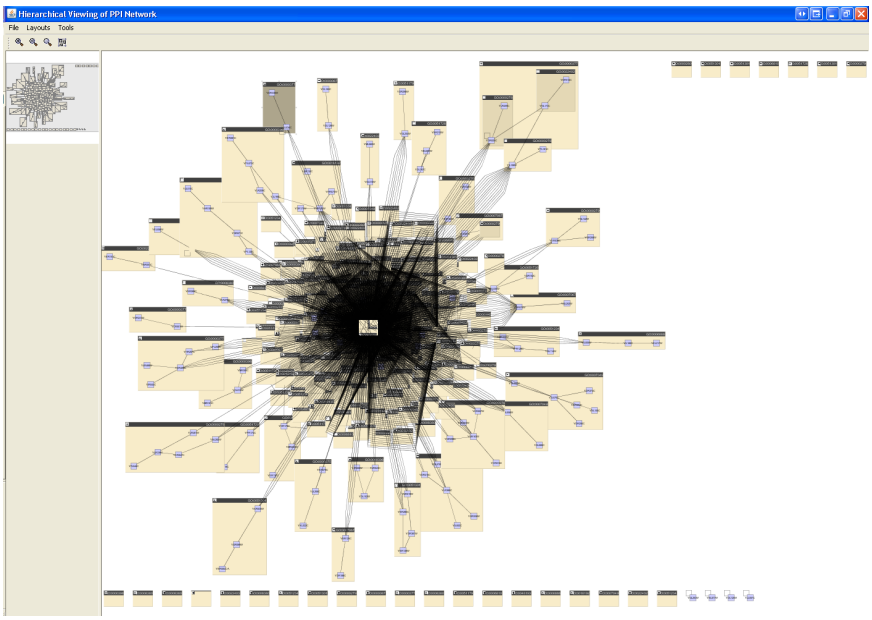

Figure 4.5: Yeast PPI network for $r = 1.6$ some compound nodes zoomed

CHAPTER 5

CONCLUSION

In this thesis, we aimed to visualize large scale protein-protein interaction networks. As a result of the size of the data, any process on a large-scale PPI network suffers from performance. We proposed a multi-resolution visualization approach, using MCL clustering, to accomplish an organized view for a given PPI network. This way, instead of dealing with the whole data, we processed clusters as units of network.

In this study, we concentrated on presenting a top-most view of the network initially, which gives the user an insight about the organization of the network via GO Terms. This way, unlike most visualization approaches, we present an initial view to the user, instead of making the user wait through all process. While the upper layers of the network are presented, presentation of lower layers are prepared in the backgound, allowing the user to interact with the network.

Our approach brings two preprocessing stages along with: MCL, and topGO annotations. Although MCL does not suffer much from time performance (see 4.1), topGO does. Since MCL clusters have sparse distributions for any scale of clusters, they include too many clusters having quite few nodes in addition to large scale clusters. In order to prevent a lack of time performance because of these clusters, we decided to discard significant GO Term calculation for small scale clusters. However, still an additional mechanism might improve performance by getting rid of the long time period preprocessing stage.

The initial results show that even the top most view of a given graph is well organized compared to straightforward visualization approaches discussed in section 1.1. MCL results in too many small scale clusters which still makes network complex than it ought to be, making it ambigious to differentiate actual large structures inside the graph from small scale ones. In

order to overcome this problem, we decided to unfold small scale compound nodes, however this led to a more complex view. We used Organic Layout algorithm provided by yFiles as default layout profile during tests. Although, most of the time, Organic Layout provides well organized layout structure, some of the results were still not very well organized.

To conclude, we think that presenting large-scale PPI networks in multi-resolutions not only improves visualization performance but also clarifies interpretation of a PPI network.

5.1 Future Work

This work can be used in reseach visualizing clusters of PPI networks, as it provides offline loading of clusters, as well. With the topGO annotations, cluster compatibility with biological process, or molecular functions can be examined.

This work further can be improved by a mechanism to optimize the number of resolution layers, and the clusterings to be used at each resolution. Another addition would be allowing end-user to add or remove resolution layers, so as to integrate additional comments on visualization.

Providing modified layout algorithms which take compound nodes in consideration, during layout organization may also improve the quality of presentation.

Since this tool is integrated with the R interpreter, additional functionalities can be added via R scripts, or using additional packages related to Bioinformatics coming along with R.

REFERENCES

[1] A. Alexa, J. Rahnenführer, and T. Lengauer. Improved scoring of functional groups from gene expression data by decorrelating go graph structure. *Bioinformatics*, 22:1600–1607, 2006.

[2] G. D. Bader, D. Betel, and C. W. Hogue. Bind: the biomolecular interaction network database. *Nucleic Acids Res.*, 31:248–250, 2003.

[3] A. Barsky, J. L. Gardy, R. E. W. Hancock, and T. Munzner. Cerebral: a cytoscape plugin for layout of and interaction with biological networks using subcellular localization annotation. *Bioinformatics*, 23:1040–1042, 2007.

[4] The Gene Ontology Consortium. Creating the gene ontology resource design and implementation. *Genome Res.*, 11:1425–1433, 2001.

[5] E. Demir, O. Babur, U. Dogrusoz, A. Gursoy, G. Nisanci, R. Cetin-Atalay, and M. Ozturk. Patika: An integrated visual environment for collaborative construction and analysis of cellular pathways. *Bioinformatics*, 18:996–1003, 2002.

[6] A. J. Enright, S. van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Res.*, 30:1575–1584, 2002.

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

[8] G. T. Hart, A. K. Ramani, and E. M. Marcotte. How complete are current yeast and human protein-interaction networks? *Genome Biol.*, 7, 2006.

[9] H. Hernandez-Toro, C. Prieto, and J. De Las Rivas. Apid2net: unified interactome graphic analyzer. *Bioinformatics*, 23:2495–2497, 2007.

[10] Z. Hu, J. Mellor, J. Wu, and C. Delisi. Visant: an online visualization and analysis tool for biological interaction data. *BMC Bioinformatics*, 5, 2004.

[11] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

[12] T. Ito, K. S. Muta, R. Ozawa, and T. Chiba. Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins. *PNAS*, 97:1143–1147, 2000.

[13] A. K. Jain, M.N. Murthy, and P.J. Flynn. Data clustering: A review. *ACM Computing Reviews*, 1999.

[14] A. Lewin and I. C. Grieve. Grouping gene ontology terms to improve the assessment of gene set enrichment in microarray data. *BMC Bioinformatics*, 7, 2006.

[15] A. Marco and I. Marin. Interactome and gene ontology provide congruent yet subtly different views ofaeukaryotic cell. *BMC Bioinformatics*, 69, 2009.

[16] A. Nikitin, S. Egorov, E. Daraselia, and I. Mazo. Pathway studio - the analysis and navigation of molecular networks. *Bioinformatics*, 19:2155–2157, 2003.

[17] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 27:29–34, 1999.

[18] P. Pagel, S. Kovac, M. Oesterheld, B. Brauner, I. Dunger-Kaltenbach, and G. Frishman. The mips mammalian protein-protein interaction database. *Bioinformatics*, 21:832–834, 2005.

[19] L. Salwinski and D. Eisenberg. The misink plugin: Cytoscape as a graphical interface to the database of interacting protein. *Bioinformatics*, 23:2193–2195, 2007.

[20] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, 13:2498–2504, 2003.

[21] M. Suderman and M. Hallett. Tools for visually exploring biological networks. *Bioinformatics*, 23:2651–2659, 2007.

[22] P. Uetz, L. Giot, G. Cagney, and T. A. Mansfield. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature*, 403:623–627, 2000.

[23] S. van Dongen. A cluster algorithm for graphs. *Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science, Netherlands*, 2000.

[24] S. van Dongen. Graph clustering by flow simulation. *Ph.D. thesis in University of Utrecht*, May 2000.

[25] J. Vlasblom and S. J. Wodak. Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. *BMC Bioinformatics*, 99, 2009.

[26] I. Xenarios, L. Salwinski, X. J. Duan, P. Higney, S. Kim, and D. Eisenberg. Dip, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30:303–305, 2002.

[27] S. H. Yook, Z. N. Oltvai, and A. L. Barabasi. Functional and topological characterization of protein interaction networks. *Proteomics*, 4:928–942, 2004.

[28] S. Zhang, X. Ning, and X. S. Zhang. Identification of functional modules in a ppi network by clique percolation clustering. *Computational Biology and Chemistry*, 30:445–451, 2006.

APPENDIX A

TOPGO INTEGRATION

A.1 Java R Interface Integration

In order to run R commands over Java, Java R Interface (JRI) needs to be installed and related necessary configurations should be done. JRI basically load R dynamic library and provides a Java API to make necessary calls to R engine from java. Having R engine installed on the system is a prerequisite. R engine installation is available from R project site[1]. JRI installation related information is also available at JRI site[2].

JRI package can be installed in two ways: either as a subcomponent of rJava package, which mainly has the complete opposite functionality or as seperate jar files. rJava package provides interface to R to make Java calls. In case of JRI is meant to be installed with rJava installation: firstly, rJava package needs to be be downloaded from rJava site [3]. It shall be installed to REngine directory. JRI related files are by default comes to path library/rJava/jri in R Working Directory. Otherwise, in case only standalone JRI is meant to be installed, JRIEngine.jar, REngine.jar and JRI.jar needs to be downloaded from JRI project site[4] and copied under R working directory.

In order to run JRI three jar files are necessary JRIEngine.jar, REngine.jar and JRI.jar. First these jar files shall be added as external library to java project aiming to use Java API calls to R.

Environment variables of the system shall be updated as follows:

[1] http://www.r-project.org/
[2] http://www.rforge.net/JRI/files/
[3] http://www.rforge.net/rJava/files/
[4] http://www.rforge.net/JRI/files/

1. *R_HOME* must be set to directory of R engine.

2. *PATH* must include path of R.dll

Check whether JRI is listed in *java.library.path* if it is not then add it as parameter to JVM as *-Djava.library.path = JRI directory*

After completing the necessary steps mentioned above, in order to use JRI interface within java, importing related packages of JRI will be sufficient.

A.2 "topGO"

A.2.1 "topGO" installation to R environment

In order to use *topGO* package, having R engine installed on the system is prerequisite. *topGO* package can directly be installed from project site via R console using the following commands:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("topGO")
```

topGO package can also be installed in java over Java R Interface as follows:

```
> import org.rosuda.JRI.REXP;
> import org.rosuda.JRI.Rengine;
> Rengine re=new Rengine(args, false, null);
> REXP expr;
> expr = re.eval("source("http://bioconductor.org/biocLite.R")",false);
> expr = re.eval("biocLite("topGO")",false);
```

A.2.2 "topGO" functionality

In order to make GO annotations for clusters, R scripts for *topGO* shall be implemented. If the given PPI network data does not include orf names of the given interactome, first orf names shall be loaded via r script:

```
> library(topGO)
> #read affy ORF dictionary to convert ORF names to affy names
> orf_affy_table <- read.csv(orfFile, header=T, as.is=T)
> affyList <- orf_affy_table$ID
> names(affyList) <- orf_affy_table$ORF
```

Then, library for the given interactome shall be installed and then loaded for GO annotations. For yeast it can be done as follows

```
> library(topGO)
> # load the affy yeast2.db library
> library(package = "yeast2.db", character.only = TRUE)}
```

After loading the required library *topGOData* object instance needs to be created and GO analysis shall be performed.

```
> # we can use this list to construct the topGOdata
> GOyeast2 <- new("topGOdata", ontology = "BP", allGenes = geneList,
  description = "Cluster 2",\newline
  nodeSize = 5, annot = annFUN.db, affyLib = "yeast2.db")\newline
> # perform the GO analysis
> result <- runTest(GOyeast2, "classic", "fisher")
> tableFis <- GenTable(GOyeast2, pValue = result, topNodes = 1, numChar = 40)
```