Winter 2008

# Biological Networks: Modeling and Structural Analysis

Emad Y. Ramadan
*Old Dominion University*

# BIOLOGICAL NETWORKS: MODELING AND STRUCTURAL ANALYSIS

by

Emad Y. Ramadan
B.S. May 1994, Alexandria University, Egypt
M.S. March 1999, Alexandria University, Egypt

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2008

Approved by:

_____
Alex Pothen (Director)

_____
Kurt Maly (Member)

_____
Mohammad Zubair (Member)

_____
Jessica Crouch (Member)

_____
Christopher Osgood (Member)

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.



ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346

# ABSTRACT

## BIOLOGICAL NETWORKS: MODELING AND STRUCTURAL ANALYSIS

Emad Y. Ramadan
Old Dominion University, 2008
Director: Dr. Alex Pothen

Biological networks are receiving increased attention due to their importance in understanding life at the cellular level. There exist many different kinds of biological networks, and different models have been proposed for them. In this dissertation we focus on suitable network models for representing experimental data on protein interaction networks and protein complex networks (protein complexes are groups of proteins that associate to accomplish some function in the cell), and to design algorithms for exploring such networks. Our goal is to enable biologists to identify the general principles that govern the organization of protein-protein interaction networks and protein complex networks. For protein complex networks, we propose a hypergraph model which more accurately represents the data than earlier models. We define the concept of $k$-cores in hypergraphs, which are highly connected subhypergraphs, and design an algorithm for computing $k$-cores in hypergraphs. A major challenge in computational systems biology is to understand the modular structure of biological networks. We construct computational models for predicting functional modules through the use of graph clustering techniques. The application of earlier graph clustering techniques to proteomic networks does not yield good results due to the high error rates present, and the small-world and power-law properties of these networks. We discuss the various requirements that clusterings of biological networks are required to satisfy, design an algorithm for computing a clustering, and show that our clustering approach is robust and scalable. Moreover, we design a new algorithm to compute overlapping clustering rather than exclusive clustering. Our approach identifies a set of clusters and a set of bridge proteins that form the overlap among the clusters. Finally we assess the quality of our proposed clusterings using different reference sets.

# ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my supervisor Professor Alex Pothen for his encouragement, tireless efforts and invaluable guidance. He spent hours teaching me how to be a real researcher, how to identify research challenges, and how to transform my ideas into crisp research endeavors. He is also a personal friend to whom I have often turned for advice.

My gratitude and appreciation go to my advisory and examining committee members: Professor Kurt Maly, Professor Mohamed Zubair, Professor Jessica Crouch and Professor Christopher Osgood for their interest in my work and their feedback.

I am indebted to Dr. Osgood for sharing his insight and experience in the biological aspects of my work. I am always grateful to Professor Stephen Olariu for advise and thoughtful discussions on both professional and personal levels. Whenever I am in need of advice or stuck in a decision, Professor Olariu has been there for me to share his experience, time and invaluable comments.

Special thanks to Dr. Assefaw Gebremedhin for his support.

My sincere gratitude goes to my wife Dalia for her unceasing patience when I spent way too much time on computer stuff. I am totally indebted to her for her love, care and support. Thanks also to my kids for organizing my life, waking me up early and keeping me awake late at night. They have shown me how to appreciate and make the best use of my time.

My gratitude also goes to my parents. Without their unconditional love, support, encouragement and prayers, I would have never made it this far. Everything I have achieved or will achieve in my life is because of their guidance and support.

Above all, I thank ALLAH. For only through ALLAH's grace and blessings has this pursuit been possible.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER I

# INTRODUCTION

Proteins accomplish their tasks within a cell by forming multi-protein complexes/modules. Recently, the *proteome*–all the proteins in an organism–has been the subject of several large-scale experiments. These experiments have generated large sets of protein-protein interaction and protein-complex association data. These studies form only a part of a wider effort to characterize the *proteome* with a view to understand the proteins' sequences, structures, cellular localizations, functions, and roles in cellular processes.

How should the data from large-scale proteomic studies be modeled in order to facilitate the design of efficient algorithms to query the data? How can the increasing need for computational models that can represent, analyze, visualize and integrate the proteome data be met satisfactorily? This work attempts to answer these and other related questions.

In this work we consider graph and hypergraph models to represent several different proteomic networks data sets, and then identify biologically significant dense subnetworks using the concepts of $k$-cores in graphs and hypergraphs. We also consider clustering the subnetworks into modules, groups of proteins and complexes that share a common function or biological process, or participate in a particular biochemical pathway.

## I.1 MOTIVATION

The principles of cellular organization and function can be understood more clearly by detecting functional modules within a cell's biological network. Such predictions may be used as an inexpensive tool to direct biological experiments. The increasing amount of available proteomic data necessitates an accurate and scalable approach to identify function modules and complexes and super-complexes (high level representation of a complex).

## I.2 OBJECTIVES AND GOALS

The overall goal of this work is to define accurate models for representing, describing and understanding different kind of proteomic networks and to design efficient algorithms for exploring these networks to find biologically significant subnetworks. Specific goals for this work are as follows:

---

This dissertation follows the style of *IEEE/ACM Transactions on Networking*.

- Develop new models for understanding proteomic networks.
- Develop new algorithms for exploring proteomic networks to find biologically significant subnetworks.
- Describe an approach to clustering protein-protein interaction networks to find biologically relevant multi-protein clusters (modules).
- Understand the relationships between the predicted modules and find cross edges or cross nodes (bridges) between modules.

## I.3 CONTRIBUTIONS

1. Unified approach for modeling: Traditionally, protein complex networks are modeled either as a graph of cliques/stars (each clique/star representing a complex) or as bipartite graph representing the association of proteins and complexes. We propose a hypergraph model to represent a protein-complex network. The hypergraph model is a new unified representation that captures dual relationships between proteins and protein-complexes and thus enables us to extract functional information from both sides-proteins and protein complexes.
2. Algorithm for computing $k$-cores in hypergraphs: We also propose a $k$-core algorithm for hypergraphs for the first time. $k$-core algorithms enable us to explore dense subhypergraphs instead of hypercliques.
3. Exclusive clustering for scale-free networks: We introduce a novel clustering approach which is effective for clustering power-law, small-world networks (high degree vertices and short average path lengths). This approach can deal with the fact that there are errors in the data. Furthermore this clustering approach finds clusters with varying topologies: dense clusters (cliques or near-cliques), and sparse clusters (stars). This approach also has low discard ratio.
4. Overlapping clustering approach: We also introduce for the first time a probabilistic approach for overlapping clustering that greatly enhances the identification of function modules in the protein networks. We extend our exclusive clustering approach to overlapping clustering. In addition, we present empirical evidence that the proposed exclusive and overlapping clustering methods work better than the extant methods.

## I.4 OUTLINE

In summary, this thesis deals primarily with the modeling and structural analysis of the biological (proteomic) networks. The thesis consists of the following four papers:

1. E. Ramadan, A. Tarafdar, and A. Pothen. *A hypergraph model for the yeast protein complex network.* Proceedings of the IEEE Workshop on High Performance Computational Biology, 2004 [1].

2. E. Ramadan, C. Osgood, and A. Pothen. *The architecture of a proteomic network in the yeast.* Proceedings of CompLife2005, Lecture Notes in Bioinformatics, vol. 3695, pp. 265–276, 2005 [2].

3. E. Ramadan, C. Osgood, and A. Pothen. *A clustering algorithm for discovering functional modules in proteomic networks* [3].

4. E. Ramadan, M. Ward, X. Guo, S. Durkin, A. Sawyer, M. Vilela, C. Osgood, A. Pothen and O. Semmes. *Development of the HTLV-1 Tax interactome by combined physical and in silico approaches.* Retrovirology, vol. 5, 13 pp., 2008 [4].

The rest of this thesis is organized as follows.

In chapter 2, we provide a brief introduction to graphs and their terminology used for proteomic networks. We will also provide a brief introduction to proteomic networks and their identification methods.

In chapter 3 we provide a brief introduction to cores in graphs and hypergraphs. We provide details of our algorithm for computing $k$-cores in hypergraphs; the computational complexity for this algorithm and some results are also presented.

Methods for exclusive graph clustering are presented in chapter 4. We propose a new hierarchical organization of the proteomic network based on these clustering algorithms. We also report the results of a study on an organism-scale protein-protein interaction network in the yeast with the goal of identifying functional modules.

In chapter 5, we propose an overlapping clustering algorithm and present detailed results on validation approaches. We also compare our clustering methods with other extant methods.

In chapter 6 we provide an application of our approach to discover proteins directly associated with Adult T-cell Leukemia (ATL) disease by studying the protein network of Human T-cell Leukemia Virus which is the causative agent of this disease.

Finally, the conclusions and future works are presented in chapter 7.

# CHAPTER II

# BACKGROUND

The information required to frame this work in context is provided here. Section II.1 discusses relevant graph-theoretic terminology. Section II.2 discusses primarily biology terminology. Section II.3 discusses graph properties. Finally Section II.4 discusses proteomic networks and identification methods.

## II.1 GRAPH-THEORETIC TERMINOLOGY

A graph $G = (V, E)$ is a triple consisting of a *vertex set* $V(G)$, an *edge set* $E(G)$, and a relation that associates with each edge two vertices called its *endpoints*. The *order* of $G$ is denoted by $n = |V|$, and the *size* of $G$ is denoted by $m = |E|$.

A *loop* is an edge whose endpoints are the same; *multiple edges* are edges having the same pair of endpoints. A *simple graph* is a graph with no loops or multiple edges. Two vertices joined by an edge are said to be adjacent. The set of vertices adjacent to a vertex $v$ is denoted by $N(v) = \{w \in V | \{v, w\} \in E\}$. The degree $deg(v)$ of a vertex $v$ is the number of edges incident on $v$, and is equal to $|N(v)|$. The minimum degree of a vertex in a graph $G$ is denoted by $\delta(G)$, the maximum degree is denoted by $\Delta(G)$.

A graph is *complete* if it has an edge between every pair of vertices; it is also called a *clique*. A complete graph on $n$ vertices is commonly denoted by $K_n$. A graph $G$ is bipartite if its vertex set can be partitioned into two sets, say $A$ and $B$, such that every edge of $G$ has one vertex in $A$ and the other in $B$.

A *subgraph* of $G$ is any graph $H$ such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$; we say $G$ contains $H$. If $H$ is a subgraph of $G$ and $V(H) = V(G)$, we say that $H$ is a spanning subgraph of $G$. The subgraph $G_S$ of $G$ induced by $S \subseteq V$ is the graph with vertex set $S$ and edge set consisting of precisely those edges of $G$ whose endpoints are in $S$.

A *walk* in a graph $G$ is a finite alternating $(v_0, e_1, v_1, \cdots, v_\ell)$ sequence of vertices and edges, where the $e_i = (v_{i-1}, v_i)$ are the edges connecting consecutive vertices. A *trail* is a walk in which no edge is repeated, and a *path* is a walk in which no vertex is repeated. Clearly, a path is also a trail. The path length is the number of edges in the path.

We say a graph $G$ is *connected* if there exists a path between every pair of its vertices. A graph is $k-$*connected* if there are at least $k$ vertex-disjoint paths connecting every pair of vertices in the graph. A *component* of a graph is a maximal connected subgraph of

a graph. A *component* is 1−connected; in common terminology, a 2−connected component is called a bicomponent, and a 3−connected component is called a tricomponent. In general, the subgraphs of $G$ defined by various levels of $k$−connectivity are called $k$−components of the graph.

The *distance* $d(v, w)$ between vertices $v$ and $w$ is the length of the shortest path in $G$ connecting $v$ and $w$. If a path connecting $v$ and $w$ does not exist we set $d(v, w) = \infty$. The *diameter*, $diam(G)$, of a connected graph $G$ is $\max_{v,w \in V} d(v, w)$ [5].

The graph *density* is the ratio between the total number of edges in the graph $m$ and the theoretical maximum number of edges possible for the graph, $|E|_{max}$. For a graph on $n$ vertices with no loops, $|E|_{max} = \binom{n}{2}$, i.e., density$(G) = |E|/|E|_{max} = m/\binom{n}{2}$ and is thus a real number ranging from 0.0 to 1.0.

A graph $G$ is *bipartite graph* if $V(G)$ is the union of two disjoint independent sets. The number of edges linking two disjoint subsets $A$ and $B$ is represented by $\delta(A, B)$. We also use $\delta(a, B)$ to denote the number of edges linking a vertex $a$ to a set of vertices $B$.

More details on graph terminology may be found in West [5].

## II.2 BIOLOGICAL TERMINOLOGY

All multicellular organisms are composed of eukaryotic cells. A typical cell has a cell membrane and contains cytoplasm, which contains organelles suspended in a semi-fluid medium called *cytosol*. The cell's genetic material consists of a linear strand of DNA, or deoxyribonucleic acid, organized into chromosomes and located within the nucleus. The basic unit of DNA is the *nucleotide*, which is composed of a deoxyribose sugar molecule bonded to both a phosphate group and a nitrogenous base. There are two types of bases: the double-ringed *purines* and the single-ringed *pyrimidines*. The purines in DNA are adenine (A) and guanine (G), and the pyrimidines are cytosine (C) and thymine (T). Like DNA, RNA, or ribonucleic acid, is a chain of nucleotides. However, it is usually single stranded; the sugar is ribose, and the four bases are adenine (A), guanine (G),cytosine (C) and uracil (U). RNA carries the messages dictated by the DNA to the cytoplasm which directs the incorporation of amino acids to the final protein [6].

The genetic code is a four-letter code made up of the DNA nitrogenous bases A, T, G, and C. Each chromosome is a long DNA molecule with thousands of these bases in a sequence. A *gene* is a region of DNA that controls a discrete hereditary characteristic, usually corresponding to a single protein or RNA. The *Genome* is the totality of the genetic information belonging to a cell or an organism, the DNA that carries this information. Like

the genome, the *proteome* is the entire collection of proteins encoded by the genome of an organism [7].

Each protein consists of one or more *domains*, regions in the protein that fold with a specified geometry. Proteins interact with each other through their domains. A group of proteins interact together to form multi-protein *complexes*, which are molecular machines responsible for cellular function. *Functional modules* are groups of proteins that interact with each other in a biochemical process, but these interactions could take place at different times (as in a cascade) or in different cellular locations. Groups of functional modules give us insight into the architecture of proteomic networks, and help us understand life processes at a higher level than otherwise.

Recently a large-scale experimental study by Cellzome has attempted a systematic characterization of the multi-protein complexes in yeast [8]. This study is a part of a wider effort to characterize the *proteome*, all the proteins in an organism, with a view to understanding their sequence, structure, cellular localization, function, and role in cellular processes. (Identifying the proteome is a challenging task since alternate splicing and post-translational modifications potentially enable many proteins to be made from a gene. The proteome is also more dynamic than the genome, since the set of proteins made by a cell depends on the phase of the cell cycle and the environmental conditions.)

*Homology* is a relationship between two biological features (here we consider genes, or proteins) which have a common ancestor.

## II.3 GRAPH PROPERTIES

We consider three global statistical properties that characterize the structure and behavior of graphs.

First, the *degree distribution* $P(k)$, which is the fraction of vertices in the graph that have degree $k$.

Second, the *clustering coefficient* $C_v$ of a vertex $v$ is the average probability that two neighbors of $v$ are adjacent [9]. More formally, if a vertex $v$ in the graph has $deg(v)$ neighbors, the ratio between the number of edges $|E_v|$ joining the neighbors of $v$, and the largest possible number of edges between them, is called the clustering coefficient of vertex $v$, and is denoted by $C_v$:

$$C_v = |E_v| / \binom{\deg(v)}{2},$$

and is thus a real number ranging from 0.0 to 1.0. The clustering coefficient $C$ of the graph is the average of $C_v$ over all vertices $v$ in the graph.

Third, the *average path length*, $L$, which is the mean distance between vertex pairs in a graph:

$$L = \frac{2}{n(n+1)} \sum_{v \geq w} d(v, w).$$

## II.4 PROTEOMIC NETWORKS

Many of biological networks can be intuitively modeled as a graph. One of these biological networks is the network of mechanistic physical interactions between proteins (as opposed to chemical reactions among metabolites), which is usually referred to as a protein protein interaction network (PPI networks). Protein interaction networks have been studied by a number of authors [10, 11, 12, 13]. Different methods have been used to identify protein interactions, including biochemical as well as computational approaches.

With vast amounts of DNA sequences becoming available in recent years, there is a growing interest in correlating the genome with the proteome to explain biological function and to develop new effective protein targeting drugs. One of the key questions in proteomics today is with which proteins does a certain protein interact. The hope is to exploit this information for therapeutic purposes. Different methods have been used to identify protein interactions, including biochemical as well as computational approaches.

Many laboratories throughout the world are contributing to one of the ultimate goals of the biological sciences "the creation and understanding of a full molecular map of a cell". To contribute to these efforts, we focus our attention on studying currently available proteomic networks.

In this section we give an overview of recent proteome identification methods, currently available data sets and their repositories.

### II.4.1 Identification Methods

As mentioned above, the lists of genes and encoded proteins are becoming available for an increasing number of organisms. Databases such as Ensembl, [14] and GenBank [15] contain publicly available DNA sequences for more than 105,000 organisms, including whole genomes of many organisms in all three domains of life, bacteria, archea, and eukaryota, as well as their protein data. In parallel to the increasing number of genomes becoming

available, high-throughput protein-protein interaction detection methods have produced in the past couple of years a huge amount of interaction data. Such methods include yeast two-hybrid systems (Y2H) to detect protein interaction [16, 17, 10], and tandem affinity purification (TAP) for identifying multi-protein complexes [8, 18]. We outline here the main characteristics of each of these methods.

### II.4.2 Y2H Method

Yeast two-hybrid assay is an in vivo technique involving the fusion of one protein to a DNA-binding domain and the other to a transcriptional activator domain. An interaction between them is detected by the formation of a functional transcription factor (reporter gene) as illustrated in Fig. 1.

The yeast two-hybrid (Y2H) system was developed by Fields and Song [19] on the basis of modular domain structure of the transcription factor GAL4 (reporter gene), comprised of a DNA binding domain and transcription activation domain. In the Y2H system, one of the proteins of one's interest, termed X, is expressed as a hybrid protein with the GAL4 DNA binding domain, whereas the other, termed Y, is expressed with the activation domain. If X and Y interact, the two hybrid proteins, often coined as bait and prey, respectively, are assembled onto GAL4 binding sites in the yeast genome. The assembly functionally reconstitutes the GAL4 transcription factor and induces the expression of reporter genes integrated in the region downstream of the GAL4 binding sites.

The Y2H system enables highly sensitive detection of protein-protein interactions in vivo without handling any protein molecules. It also allows one to screen a library of activation domain fusions or preys for the binding partners of one's favorite protein expressed as a DNA binding domain fusion or bait.

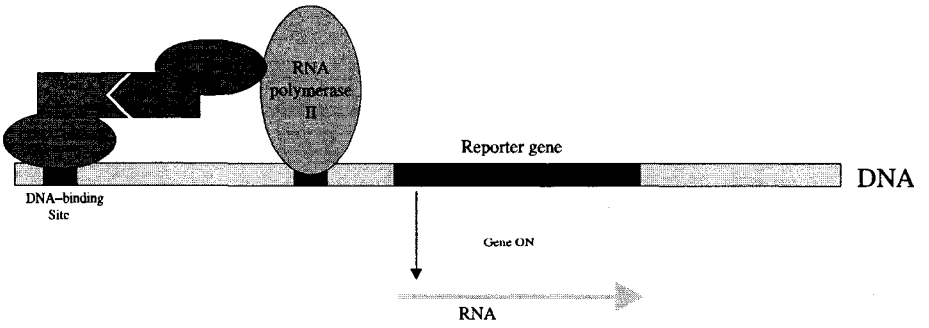



FIG. 1: The Yeast Two-Hybrid Method.

On the other hand, the Y2H system has limitations. First, in principle, it cannot detect interactions requiring three or more proteins and those depending on post-translational modifications. Second, the Y2H system is not suitable for the detection of interactions involving membrane proteins. Finally, the Y2H interaction does not guarantee that the inferred interactions are of physiological relevance. Despite these and other limitations, the Y2H system is now established as a standard technique in molecular biology.

The Y2H system has been successfully used to examine an interaction between two proteins of interest and also to screen for unknown binding partners of a specific protein. It can , in principle, be used in a more comprehensive fashion to examine all possible binary interactions between the proteins encoded by any single genome.

### II.4.3 TAP Method

Gavin et al. [8] and Ho et al. [18] take a different approach called tandem affinity purification (TAP) for identifying multi-protein complexes.

In this approach, protein complexes were purified as follows (Fig. 2 ). First, a set of *bait proteins* in the yeast is selected. The genes corresponding to the bait proteins are tagged with a tandem affinity purification (TAP) tag, and inserted into dividing yeast cells. In the yeast cells the tagged gene is expressed into tagged bait protein. The tagged bait protein and other proteins that are complexed with it are isolated from the lysed cells as follows: The TAP tag is used to purify a protein complex through two affinity column separations before the proteins in it are identified via mass spectrometry, a technique that involves running an electric charge through the complexes on a gel, so that proteins become separated according to mass. Database-search algorithms are then used to identify specific proteins from their mass spectra [8].

The data from the TAP experiment thus consists of protein complexes identified from various bait proteins. Each complex is identified by the proteins that constitute its members.

This approach detects real complexes in their physiological settings and enables a consistency check by tagging several members of a complex at the same time. However its drawbacks are that it might miss some complexes that are not present under the given conditions, tagging can disturb complex formation; and loosely associated components can be washed off during purification.

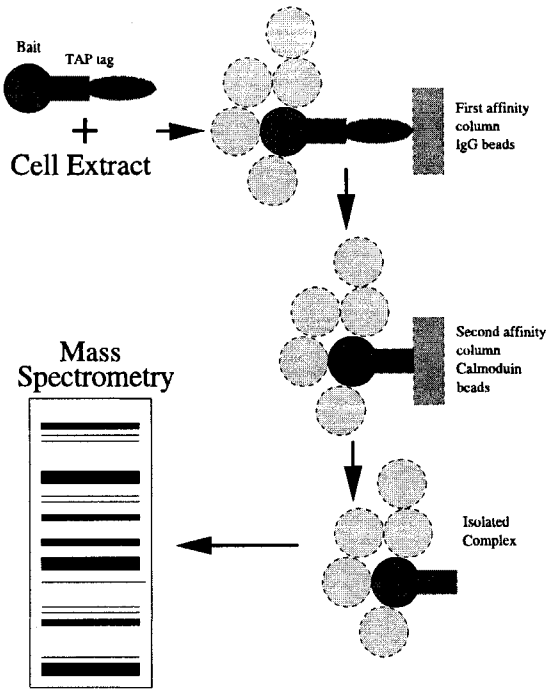



FIG. 2: Tandem Affinity Purification (TAP) Method.

### II.4.4 Public Data Sets

Vast amounts of biological data that are constantly being generated around the world are deposited in numerous databases. There are still no standards for accumulation of proteomic data into databases. Thus, different biological databases contain different kinds of data in different scales: single experiments, high-throughput experiments, and literature sources.

The largest nucleotide sequence databases are EMBL [20] and GenBank [15], which contain sequences from the literature as well as those submitted directly by individual laboratories. These databases store information in a general manner for all organisms. Organism specific databases exist for many organisms. For example, the complete genome of bakers yeast and related yeast strains can be found in *Saccharomyces* Genome Database [21]. FlyBase [22] contains the complete genome of the fruit fly *Drosophila melanogaster*. It is one of the earliest model organism databases. Ensembl [14] contains the draft human genome sequence along with its gene predictions and large scale annotations.

SwissProt [23] and Protein Information Resource (PIR) [24] are two major protein sequence databases. They are both manually curated and contain literature links. They

exhibit a large degree of overlap, but still contain many sequences that can be found in only one of them. SwissProt maintains a high level of annotations for each protein including its function, domain structure, and post-translational modification information.

Some of the main databases containing protein interaction data are the following. The Munich Information Center for Protein Sequences (MIPS) [25] provides high quality curated genome related information, such as protein-protein interactions, protein complexes, pathways etc., spanning over several organisms. The Cellzome company provides information for multi-protein complexes in yeast [8]. Yeast Proteomics Database (YPD) [26] is another curated database. It contains bakers yeast, *S. cerevisiae*, protein information, including their sequence and genetic information, related proteins, PPIs, complexes, literature links, etc. The Database of Interacting Proteins (DIP) [27] is a curated database containing information about experimentally determined PPIs.

# CHAPTER III

# HYPERGRAPH MODEL

## III.1 INTRODUCTION

Proteins accomplish their tasks within a cell by forming multi-protein complexes, which are assemblies of groups of proteins. Recently a large-scale experimental study by Cellzome has attempted a systematic characterization of the multi-protein complexes in yeast [8]. This study is a part of a wider effort to characterize the *proteome*, all the proteins in an organism, with a view to understanding their sequence, structure, cellular localization, function, and role in cellular processes. (Identifying the proteome is a challenging task since alternate splicing and post-translational modifications potentially enable many proteins to be made from a gene. The proteome is also more dynamic than the genome, since the set of proteins made by a cell depends on the phase of the cell cycle and the environmental conditions.)

How should the data from large-scale protein complex studies be modeled in order to facilitate efficient algorithms to query the data? The data consists of the protein membership lists of the protein complexes. The authors of the Cellzome study and others have used graph models. However, graph models that have been proposed to date have represented the data either as protein-protein interaction graphs, or as complex intersection graphs, by projecting out either the complexes or the proteins. Unfortunately, some of these graph models make assumptions about the structure of the protein complexes, for which there is insufficient data; other graph models lose some of the information in the data. In a protein-protein interaction graph, either all the proteins in a complex are considered to form a clique, or a distinguished protein (called the *bait protein* from the nature of the experiments) is considered to be connected to all the others. But both these assumptions are unphysical; in a large complex consisting of nearly a hundred proteins, say, it would be sterically impossible for all the proteins to bind to each other, or for one protein to bind to all others. The other model used, the complex intersection graph, does not represent the proteins at all.

We propose a hypergraph model for the protein complex data. We characterize the properties of the yeast protein complex hypergraph from the Cellzome experimental study. We show that it is both a power-law (scale-free) network and a small-world network. Then we discuss the concepts of the $k$-core of a graph and of a hypergraph, and describe an

algorithm for computing the various cores. Using this algorithm, we compute the maximum core of the yeast protein complex hypergraph, and thereby characterize the *core proteome*. One of the problems with large-scale proteomic techniques when compared to smaller-scale methods is the relatively lower reliability of the experimental results. Hence we consider the problem of selecting a subset of the proteins to be bait proteins (that pull other proteins in complexes with them) in the Cellzome experiment. Minimum weight vertex covers (and multicovers) in hypergraphs are used to choose a set of candidate bait proteins that (1) are provably within a small factor of the optimum cover, and (2) identify the protein complexes with improved reliability.

### III.1.1 The Cellzome Experiment

In the Cellzome approach [8] for identifying multi-protein complexes, a set of *bait proteins* in the yeast is selected. The genes corresponding to the bait proteins are tagged with a tandem-affinity-purification (TAP) tag, and inserted into dividing yeast cells. In the yeast cells the tagged gene is expressed into tagged bait protein. The tagged bait protein and other proteins that are complexed with it are isolated from the lysed cells as described below. The TAP tag is used to purify a protein complex through two affinity column separations before the proteins in it are identified via mass spectrometry.

The TAP tag consists of three components. The first component is a protein that binds to immunoglobulin G beads in an affinity column, and the multi-protein complex is isolated from the rest of the cell lysate through this binding. The second component is a protease, which is used to remove the protein complex from the column by cleaving it from the immunoglobulin beads. The third component is a calmodulin binding peptide, which is used to purify the complex a second time by its binding to a second affinity column containing calmodulin beads. The complex is eluted from this column using a chemical that preferentially binds to the beads, and the proteins in the complex are identified through mass spectrometry.

The data from the TAP experiment thus consists of protein complexes identified from various bait proteins. Each complex is identified by the proteins that constitute its members.

### III.1.2 Graph Representations

Gavin et al [8] visualized the data from their experiments as a complex intersection graph. The vertices of this graph are the complexes, and an edge joins two complexes if they have

one or more proteins in common. Edges in this network reflect common protein memberships in the complexes, and may represent common regulation, localization, turnover, or architecture. Each edge of the complex intersection graph could be weighted to represent the number of proteins two complexes have in common. Unfortunately, information about proteins is not represented in the complex intersection graph.

Another approach to representing the data from the TAP experiment is to use a protein-protein interaction graph, as has been used to represent binary protein interactions from experiments such as the yeast 2-hybrid system. In a binary protein interaction graph, the vertices correspond to proteins, and an edge joins two proteins that bind to each other. Note that the complexes are not represented in the protein interaction graph, and hence it is not powerful enough to represent both proteins and complexes.

There is a more serious difficulty with the use of protein interaction graphs to represent the protein complex data. Which proteins in a complex bind with each other? A range of answers to this problem is possible, and has been employed in graph representations.

All of the proteins in a complex could be considered to interact with each other, and then each complex is represented as a clique defined on the proteins in the complex. This approach leads to unusually high clustering coefficients in the protein interaction graphs [28]. Another viewpoint is to consider the bait protein as interacting with each other protein it pulls down in a complex, and thus each complex is represented as a 'star graph'. Both approaches are unphysical, as we have pointed out earlier.

Unless we know the set of binary interactions among the proteins in each complex, the approach of representing the data by protein-protein interaction graphs is bound to be inaccurate. It is also expensive in terms of storage, since a clique on $n$ vertices can be represented in $O(n)$ space as a list of vertices, while representing each edge in the clique (as is done in protein interaction graphs) requires $O(n^2)$ space. A similar criticism is also true of complex intersection graphs, since an edge joins two complexes that share a common protein; a protein that belongs to $m$ complexes generates $O(m^2)$ edges in the complex intersection graph, while the complexes can be represented in space proportional to the sum of the numbers of proteins in them.

### III.1.3 Protein Complex Hypergraph

We propose a hypergraph model for the protein complex data. A hypergraph $H = (V, F)$ consists of a set of vertices $V$ and a set of hyperedges $F$; each hyperedge is a subset of vertices. The difference between an edge in a graph and a hyperedge in a hypergraph is

that the former is always a subset of two vertices (or a subset consisting of one vertex in the case of a loop), whereas in a hyperedge, the subset of vertices can be of arbitrary cardinality. In the protein complex data, we represent each protein by a vertex and each complex by a hyperedge.

The *degree* of a vertex is the number of hyperedges it belongs to, and the *degree* of a hyperedge is the number of vertices it contains. We denote the maximum degree of a vertex by $\Delta_V$ and the maximum degree of a hyperedge by $\Delta_F$.

A *path* in the hypergraph consists of an alternating sequence of vertices and hyperedges $v_1, f_1, v_2, f_2, \ldots, v_{i-1}, f_{i-1}, v_i$, (where the $v_j$ are vertices, and the $f_j$ are hyperedges), with the following properties: each hyperedge $f_j$ contains vertices to its left and right in the listing of the path, $v_j$ and $v_{j+1}$; the path begins and ends with vertices, and no hyperedge or vertex can be repeated. The *length* of the path is the number of hyperedges in it. The *distance* between two vertices is the length of a shortest path that joins them. The *diameter* of a hypergraph is the maximum distance between any pair of vertices in it.

For drawing a hypergraph, it is helpful to represent both the vertices and the edges of the hypergraph as vertices in a bipartite graph. This graph $B(H) = (X, Y, E)$ has its vertex set partitioned into two sets $X$ and $Y$, where $X$ corresponds to the vertices of the hypergraph, and $Y$ to the hyperedges of the hypergraph; an edge $(v, f) \in E$ joins a vertex $v \in X$ to vertex $f \in Y$ if the vertex $v$ in the hypergraph belongs to the hyperedge $f$. For the protein complex data, the proteins correspond to one set of vertices, the complexes to the other, and an edge joins a protein to a complex if and only if the protein is a member of the complex.

Fig. 3 depicts a drawing of the yeast protein complex hypergraph from the Cellzome experiment as a bipartite graph. The figure is drawn using the Pajek software (URL: `vlado.fmf.uni-lj.si/pub/networks/pajek`).

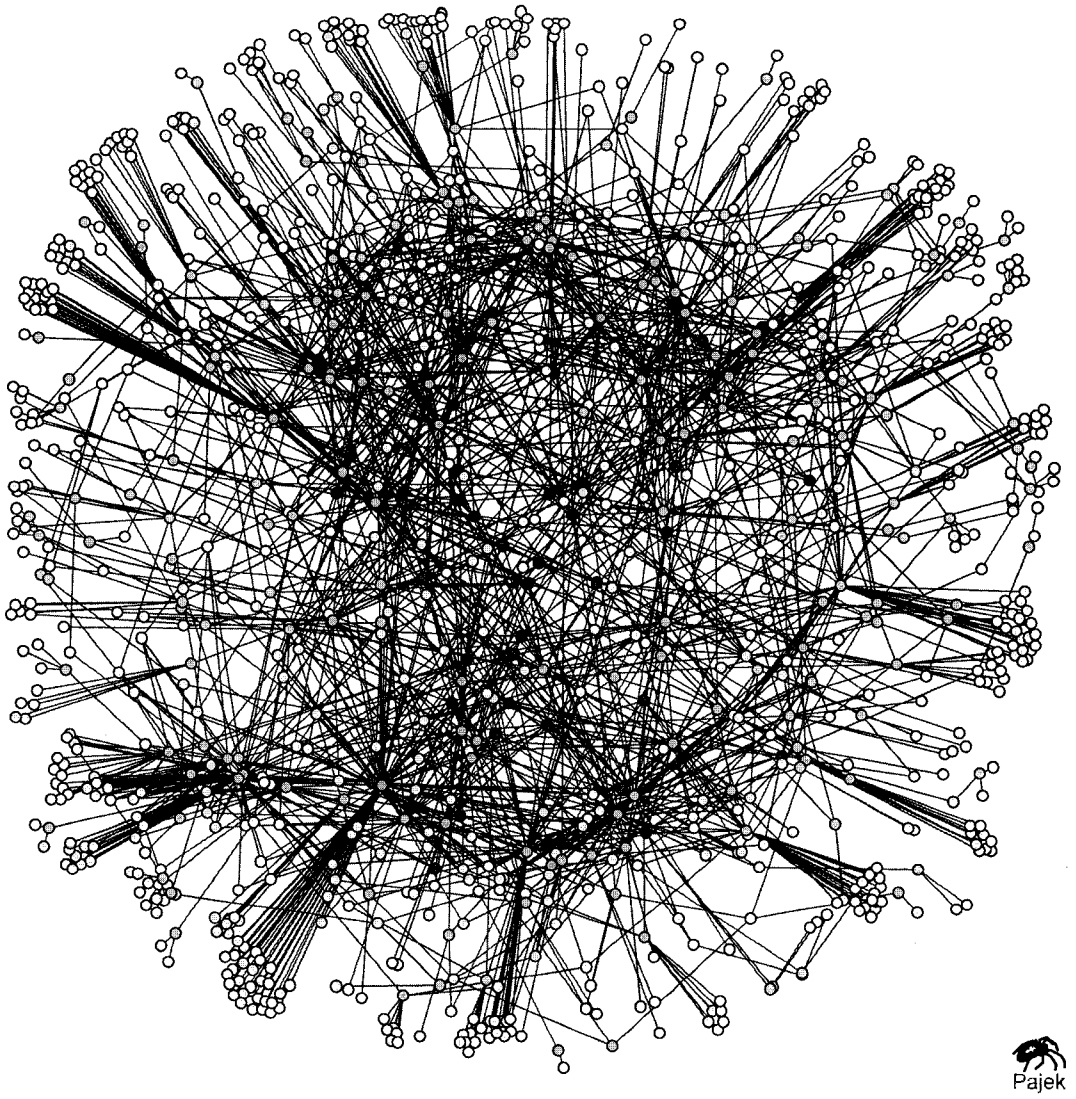



FIG. 3: The yeast protein complex hypergraph and its maximum core.

## III.2 PROPERTIES OF PROTEIN COMPLEX HYPERGRAPH

The protein complex hypergraph from the Cellzome experiments has 33 connected components, the largest of which consists of $1,263$ proteins (vertices) and 99 complexes (hyperedges). There are 846 proteins with degree 1 (belonging to only one complex), and the maximum degree of a protein is 21. This highest degree protein is ADH1, an alcohol dehydrogenase. The diameter of the yeast hypergraph is 6, and the average path length is 2.568, suggesting that the yeast protein complex hypergraph is a small-world network.

The frequency of proteins with a given degree plotted against the degree shows that the yeast hypergraph satisfies a power-law, $P(d) = cd^{-\lambda}$, where $d$ is the protein degree, and $P(d)$ is the number of proteins with degree $d$. From a log-log plot, shown in Fig. 4, we estimate $\log(c) = 3.161$ and the exponent $\lambda = 2.528$; we assess the goodness of the linear fit to be excellent, since the value of $R^2 = 0.963$, where $R^2 = 1 - (r^T r)/(y^T y)$, $r$ is the vector of residuals, and $y$ is the dependent variable measured in deviations from the mean. Unlike proteins, the frequencies of complexes with a given degree do not satisfy a power-law or an exponential distribution.

## III.3 THE CORE OF A HYPERGRAPH

Tong et al. [29] have considered graph models of binary protein interactions of a class of peptide-recognizing proteins. They have studied the $k$-cores of this proteomic network to identify significant interactions common to the computational network and an experimentally observed phage display network. Other authors, e.g., [30] have also studied the cores of protein-protein interaction graphs. Some of this work is done with an eye towards identifying biologically significant interactions, but some try to determine putative protein complexes in this manner. As we have pointed out earlier, the latter endeavor is error-prone since the proteins in a complex might have only few interaction partners. Others have conjectured the existence of a *core proteome*, a set of proteins with sequences, structures, and basic cellular roles that might be common to a set of organisms.

To identify the core proteome from a protein complex hypergraph, we need an algorithm for computing the maximum core of a hypergraph. We begin by considering the concept of a core in a graph, then extend it to a hypergraph, and finally design an algorithm for computing a core in a hypergraph.

The $k$-core of a graph $G$ is a maximal subgraph of $G$ in which every vertex has degree at least $k$ in the subgraph. The $k$-core has been studied earlier in different contexts by several authors; e.g., see Sec. 3.7.2 in [31]. The maximum core of a graph corresponds to the maximum value of $k$ for which the graph has a non-empty $k$-core. The maximum core in the graph shown in Fig. 5 is a 3-core (the subgraph consisting of vertices colored green and the subset of edges both of whose endpoints are green). The entire graph forms the 1-core, the 2-core is the same as the 3-core, and the 4-core is empty. Note that the $k$-core need not be a connected subgraph.

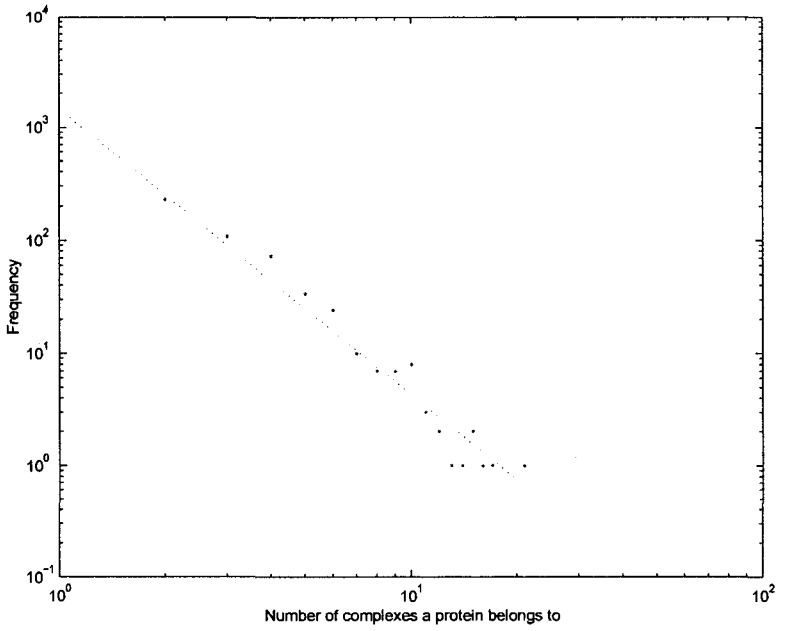



FIG. 4: The frequency of proteins with a given degree plotted against the degree shows that the yeast protein hypergraph satisfies a power-law degree distribution for the proteins.

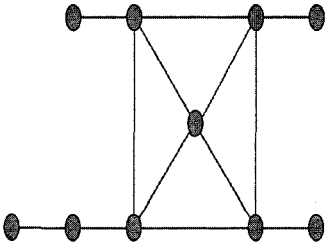


FIG. 5: The $k$-core of a graph.

There is a linear-time algorithm (in the number of edges) for computing all the non-empty cores of a graph. The idea is to repeatedly remove a vertex $v$ of minimum degree in the graph and all edges incident on it, updating the degrees of the neighbors of $v$ in the residual graph as edges are deleted. The algorithm repeats this step until the graph is empty; the highest value of the minimum degree observed during the algorithm corresponds to the value of the maximum core. Note that the minimum degree in the residual graph could increase or decrease during the algorithm.

We now extend the concept of a $k$-core to a hypergraph. A *reduced* hypergraph is one in which every hyperedge is maximal, i.e., no hyperedge is contained in another. The $k$-core of a hypergraph $H$ is a maximal subhypergraph that is reduced, and in which every vertex belongs to at least $k$ hyperedges in the subhypergraph. Thus in a $k$-core, every hyperedge is maximal. When we delete a vertex from the hypergraph, a hyperedge that it belongs to gets deleted when it is no longer maximal. (This includes the special case of a hyperedge becoming empty due to vertex deletions.)

An algorithm for computing the $k$-core of a hypergraph is shown in Algorithm 1. We can detect non-maximal hyperedges by counting overlaps among hyperedges instead of comparing set memberships, and will be discussed later.

Let $|V|$ denote the number of vertices, $|F|$ the number of hyperedges, $\Delta_V$ the maximum vertex degree, and $\Delta_F$ the maximum degree of a hyperedge. Let

$$\sum_{v \in V} d(v) = \sum_{f \in F} d(f) \equiv |E|,$$

denote the sum of the degrees of the vertices (or equivalently, the sum of the degrees of the hyperedges). Note that $|E|$ is the space needed to represent the hypergraph. Define the "degree-2" of a hyperedge $f$, $d_2(f)$, to be the number of hyperedges with which it has a vertex in common. (This is the number of hyperedges that $f$ can reach by a path of length two (edges) in the bipartite graph $B(H)$.) Let $\Delta_{2,F}$ denote the maximum value of the degree-2 of the hyperedges in $H$.

We now establish the time complexity of the $k$-core algorithm. Initially we will consider all the steps in the algorithm excluding the maximality computation for the hyperedges.

A vertex $v$ can be deleted from a hyperedge in $\ln \Delta_F$ time, and a hyperedge can be deleted from the adjacency set of a vertex in $\ln \Delta_V$ time, by maintaining these sets as balanced trees. The key observation to bounding the total number vertex deletions in the algorithm is that each vertex can be deleted from a hyperedge at most once. Hence the number of vertex deletions in the algorithm is $O(|E|)$. Similarly the number of times hyperedges are deleted from the adjacency lists of vertices is also bounded by $O(|E|)$. Hence the time required by vertex and hyperedge deletions in the algorithm can be bounded by

$$O(|E|(\ln \Delta_V + \ln \Delta_F)).$$

**Algorithm 1** An algorithm for computing the $k$-core of a hypergraph.

```
1:  while there are vertices with degree < k do
2:      for each such vertex v do
3:          for each hyperedge f ∋ v do
4:              Delete v from adj(f);
5:              Decrement d(f) by one;
6:              if f is non-maximal then
7:                  for each vertex w ∈ f do
8:                      Delete f from adj(w);
9:                      Decrement d(w) by one;
10:                     if d(w) < k then
11:                         Include w in list of vertices with degree < k;
12:                     end if
13:                 end for
14:             end if
15:         end for
16:     end for
17: end while
```

Now we show that the maximality computations for the hyperedges can be performed without explicitly comparing the vertex lists of the hyperedges during the $k$-core algorithm. Initially we can compute the nonzero pairwise overlaps of the hyperedges by processing the adjacency lists of the vertices in time

$$O(\sum_{v \in V} d(v)^2) \ln \Delta_{2,F} = O(\Delta_V |E|) \ln \Delta_{2,F}.$$

The logarithmic factor comes from inserting into a balanced tree representation of the hyperedges that overlap with a given hyperedge. We need to check for non-maximality only those hyperedges whose degrees are decremented in the algorithm due to the deletion of a member vertex. The key observation is that a hyperedge is contained in another only when its current degree is equal to its current overlap with the latter hyperedge. Hence we can check for maximality without comparing set memberships, but by maintaining the current degrees and overlaps of the hyperedges. This cost can be bounded for the algorithm by

$$\sum_{v \in V} \sum_{f \ni v} d_2(f) \leq \Delta_{2,F} |E|.$$

The time complexity of the $k$-core algorithm is then bounded by

$$O(|E|(\Delta_{2,F} + \Delta_V \ln \Delta_{2,F})).$$

We have implemented this $k$-core algorithm for hypergraphs. On an Intel Xeon 2 GHz processor with 1GB memory, the $k$-core computation of the yeast hypergraph took 0.47 seconds. While the Cellzome study is the largest such current study on protein complexes, it leads to a small hypergraph from a computational perspective. However, larger proteomic studies, e.g., ones that scale to the human proteome, or proteomic studies involving multiple organisms, will require high performance algorithms and software. Meanwhile, we have run the hypergraph core algorithm on larger hypergraphs obtained from scientific computing applications (from the Matrix Market, URL: `math.nist.gov/MatrixMarket`), and the results are included in Table I. The results show that if the numbers of vertices and hyperedges in the core are large, then the run times can be substantial; hence for large hypergraphs, a parallel algorithm will need to be designed. The maximum core in the yeast protein complex hypergraph is a 6-core consisting of 41 proteins and 54 complexes.

It has been conjectured that the core of a protein-protein interaction network or a protein-complex hypergraph might represent the *core proteome*, a set of proteins and complexes performing essential cellular functions and that they might be common to many organisms. We tested this conjecture on the Cellzome yeast protein complex hypergraph.

We found that of the 9 of the 41 proteins in the 6-core are currently unknown or have unknown function; 22 of the 32 proteins that are known or have known function are essential; i.e., deleting the corresponding gene is lethal to yeast. Also, 24 have reported homologs in the Yeast Genome Database (URL: `www.yeastgenome.org`), three of which belong to the subset of proteins that are unknown or have unknown function. The homologous proteins belong to organisms such as the human, mouse, and worm. Since the Comprehensive Yeast Genome Database reports 878 essential and 3,158 non-essential genes, essential proteins constitute a higher fraction of the proteins in the core.

We computed maximum cores in the *protein-protein interaction networks* for yeast and fruitfly obtained from the Database of Interacting Proteins (DIP) circa. Nov 2003 (URL: `dip.doe-mbi.ucla.edu/dip/`). The maximum $k$-core for the yeast had $k = 10$ with 33 proteins, while the drosophila network had $k = 8$ with 577 proteins. The total number of proteins in the yeast network was 4746, while that in the fruitfly was 7065.

TABLE I: Statistics on hypergraphs and their maximum cores from Cellzome and scientific computing applications. Legends for times are h: hours, m: minutes, and s: seconds.

| | $\|V\|$ | $\|F\|$ | $\|E\|$ | $\Delta_V$ | $\Delta_F$ | $\Delta_{2,F}$ | max core | core $\|V\|$ | core $\|F\|$ | core time |
|---|---|---|---|---|---|---|---|---|---|---|
| cellzome | 1361 | 232 | 2678 | 21 | 88 | 84 | 6 | 41 | 54 | 0.47s |
| bfw | 62 | 62 | 450 | 21 | 17 | 47 | 4 | 31 | 23 | 0.06 s |
| fdp1 | 656 | 656 | 10038 | 62 | 62 | 191 | 3 | 276 | 28 | 0.5 s |
| stk13 | 2003 | 2003 | 42943 | 82 | 64 | 287 | 7 | 1159 | 659 | 35.7 s |
| utm | 3060 | 3060 | 42211 | 29 | 20 | 129 | 12 | 160 | 220 | 4.5 m |
| fdp11 | 22294 | 22294 | 623554 | 43 | 43 | 256 | 13 | 20719 | 18578 | 6.8 h |

### III.4 VERTEX COVERS

We now consider the reliability of the Cellzome experiment. Out of the total $1,361$ proteins in the study, $589$ proteins were used as bait proteins. Each bait protein pulls down one or more complexes it belongs to. The large-scale identification of complexes from proteins associated with bait proteins is unfortunately more error-prone than smaller-scale studies (the Cellzome experiments report a reproducibility of $70\%$). Hence we consider the question of choosing the bait proteins from the protein complex hypergraph once it is partially known. We believe this could be useful in two situations: the first is when we wish to repeat the experiments to improve the reliability of the data; the second is when we wish to use one organism as a model to identify the protein complexes in a related organism. The bait selection problem can be formulated as the problem of choosing vertex covers in hypergraphs of minimum size or weight.

#### III.4.1 Vertex Cover for Hypergraphs

Given a hypergraph $H = (V, F)$, and non-negative weights on the vertices, the minimum weight vertex cover problem is to find a subset of vertices $C \subseteq V$ such that $C$ includes at least one vertex from each hyperedge, and the sum of the weights of the vertices in $C$ is minimum. The problem is NP-hard, but approximation algorithms for finding near-optimal vertex covers exist.

The vertex cover problem for a hypergraph can be reduced to the set cover problem for a collection of subsets of a ground set $S$. Given a set $S$ and a collection of subsets of $S$ with weights assigned to each subset, the set cover problem is to choose some of the subsets from the collection so that the union of the chosen subsets is $S$ and the sum of the weights of the subsets chosen is minimum. The vertex cover problem for a hypergraph is equivalent to the set cover problem by viewing each vertex in the hypergraph in terms of its adjacency set.

The greedy algorithm for finding an approximate vertex cover of minimum weight in a hypergraph chooses a vertex with minimum *cost* (to be defined) for each vertex. During the course of the algorithm, the current cost $\alpha(v)$ of a vertex $v$ is obtained by distributing its weight equally among the hyperedges it belongs to that are currently uncovered. Let $F_i$ denote the set of hyperedges not yet covered by a partial vertex cover at the beginning of the $i$th iteration of the algorithm; then

$$\alpha(v) = w(v)/|\mathrm{adj}(v) \cap F_i|.$$

At each step, the algorithm chooses a vertex with minimum cost $\alpha(v)$ to include in the partial cover, and deletes all hyperedges it covers. The algorithm is described in Algorithm 2.

The greedy algorithm for set cover is due to Johnson, Chvatal and Lovasz [32], and is an $H_m = O(\log m)$ approximation algorithm, where $m \equiv |F|$ is the number of hyperedges, and $H_m$ is the $m$th harmonic number: $H_m = 1 + 1/2 + \ldots + 1/m$. We have implemented this algorithm in time

$$O(\sum_{v \in V} d_2(v)) \leq O(\Delta_F |E|).$$

(Here $d_2(v)$ is the "degree-2" of a vertex $v$; i.e., the number of distinct vertices other than $v$ in all the hyperedges that $v$ belongs to. It is equal to the number of vertices that can be reached from $v$ by a path of length two in the bipartite graph $B(H)$.)

A variation on the minimum weight vertex cover problem is the minimum weight multicover problem, where we wish to cover each hyperedge $f$ with $r_f \geq 1$ vertices, where $r_f$ is a given positive integer that depends on $f$. A simple modification to the algorithm given above solves this latter problem with the same approximation ratio. The change is that now when a vertex $v_i$ is included in the cover, only those hyperedges $f \in \mathrm{adj}(v_i)$ whose multicover requirements have been met are deleted.

**Algorithm 2** The greedy algorithm for computing an approximate minimum weight vertex cover of a hypergraph.

```
1: i := 1;                                          ▷ (iteration number)
2: C := ∅;                                          ▷ (cover)
3: F_1 := F;                                        ▷ (hyperedges yet to be covered)
4: while F_i ≠ ∅ do
5:     for v ∈ V \ C do
6:         Choose a vertex v_i with min cost α(v);
7:         Add v_i to the cover C;
8:         F_{i+1} := F_i \ adj(v_i);
9:         i := i + 1;
10:    end for
11: end while
```

Dual and primal-dual algorithms with approximation ratios that depend on the maximum degree of a vertex can also be designed for the weighted vertex cover problem in hypergraphs. For the yeast protein complex hypergraph, the greedy algorithm yields a better approximation bound than these algorithms. However, it is not clear if these algorithms will be practically inferior or superior in quality to the greedy algorithm discussed here. This is the subject of current work.

### III.4.2 Results

For the yeast protein complex hypergraph, the greedy algorithm finds an approximate minimum cardinality vertex cover consisting of 109 proteins, with the average degree of a protein in the cover being approximately 3.7. However, a protein belonging to many complexes might not unambiguously pull down all the complexes it belongs to; hence it is preferable to choose as bait proteins those of low degree. We accomplish this by weighting each protein by the square of their degrees. The greedy algorithm now finds an approximate minimum weight vertex cover with 233 proteins, with the average degree of a protein in the cover reduced to approximately 1.14. In comparison, the Cellzome experiment reports complexes pulled down from 459 bait proteins, with the average degree of a bait protein approximately 1.85.

Since the reproducibility of the Cellzome experiment is low, it would be more reliable to cover each complex more than once. We have also run a multicover algorithm to cover

each hyperedge twice, excluding three complexes that consist of a single protein. The remaining 229 complexes are covered twice each by 558 proteins, with the average degree of a protein in the cover approximately 1.74.

The majority of the bait proteins of the Cellzome study (429) pull down only one complex, with 26 pulling down two complexes, and the remaining 4 pulling down 3 complexes. Of course, factors other than the degree influence the suitability of a bait protein in the TAP experiment. A proteomics expert could set preferences for each protein to be used as a bait, and our algorithms could work with those weights to make a meaningful choice of a candidate set of bait proteins for large-scale proteomics experiments.

## III.5 CONCLUSIONS

We have defined the concept of a $k$-core of a hypergraph, designed an algorithm to compute it, and used it to identify the core proteome of a yeast protein complex hypergraph. The yeast core proteome is rich in both essential and homologous proteins. We have also implemented a greedy approximation algorithm for computing variations of minimum weighted vertex covers in a hypergraph, and used it to suggest candidate bait proteins in the Cellzome experimental methodology for obtaining the protein complex data. We believe this algorithm could be useful for proteomics experts to choose bait proteins to improve the reliability of the large-scale proteomics experiment, and to scale up to experimentation on proteomes of other organisms.

# CHAPTER IV

# EXCLUSIVE CLUSTERING

## IV.1 INTRODUCTION

Systems biology involves the study of complex biological structures and processes by identifying their molecular components and the interactions among them. Looking across the evolutionary landscape, biological subsystems performing discrete functions are capable of being linked together in different ways without lethality to an organism, and often with positive gains in complexity and adaptation. Among the properties that are now recognized in multiple biological systems are: modularity (sets of semi-autonomous molecules that perform specific functions); robustness (the ability of biological systems to tolerate perturbations and noise); and emergence (new properties that emerge from the interaction of functional modules) [33].

One of the challenges in computational systems biology is to create tools that enable biologists to identify functional modules and the interactions among them from large-scale genomic and proteomic data. We report the results of a study on an organism-scale protein-protein interaction network in the yeast with the goal of identifying proteins that form functional modules, (i.e., multiple proteins involved that have identical or related biological function), by clustering techniques.

Methods for clustering proteomic networks have to cope with several features specific to protein interaction data. High-throughput experiments such as the yeast 2-hybrid system and the tagged affinity purification (TAP) [8, 18, 10], have high error rates, nearing 50% in some instances. Proteomic networks are *modified power-law* networks and *small-world* networks [34]. That is, the distribution of the fraction of vertices with a given degree follows a modified power-law; and the average path length between vertices is of the order of $\ln n$ (or smaller), where $n$ is the number of vertices in the network. Hence there is a large number of low degree proteins, and a significant number of high degree proteins. The latter make it harder to discover clusters in the data, while the former increase the computational requirements. Cluster validation is hampered by the fact that there is often little overlap between different experimental studies due to the limited coverage of the interactome [35]. Finally, the predicted clusters must be biologically significant: e.g., functionally homogeneous.

In spite of these difficulties, we believe that we have successfully clustered a yeast proteomic network, with the predicted clusters overlapping well with multi-protein complexes and organelles. Our approach is based on identifying *hub proteins*, proteins that connect to a large number of clusters, and low-shell proteins (defined in the next section), and clustering the residual network. Low-shell proteins can be added to the cluster network at a later stage. We validate the clusters by comparing the clusters against experimental data on multi-protein complexes.

The hub proteins carry interesting information about the architecture of proteomic network, and are organized into a subnetwork of their own. Thus we propose a two level architecture for the yeast proteomic network, consisting of a global subnetwork of hubs, and a local subnetwork of clusters and low-shell proteins. A schematic of this architecture is shown in Fig. 6, where the top level corresponds to the global hub network, and the lower level corresponds to the local cluster network.

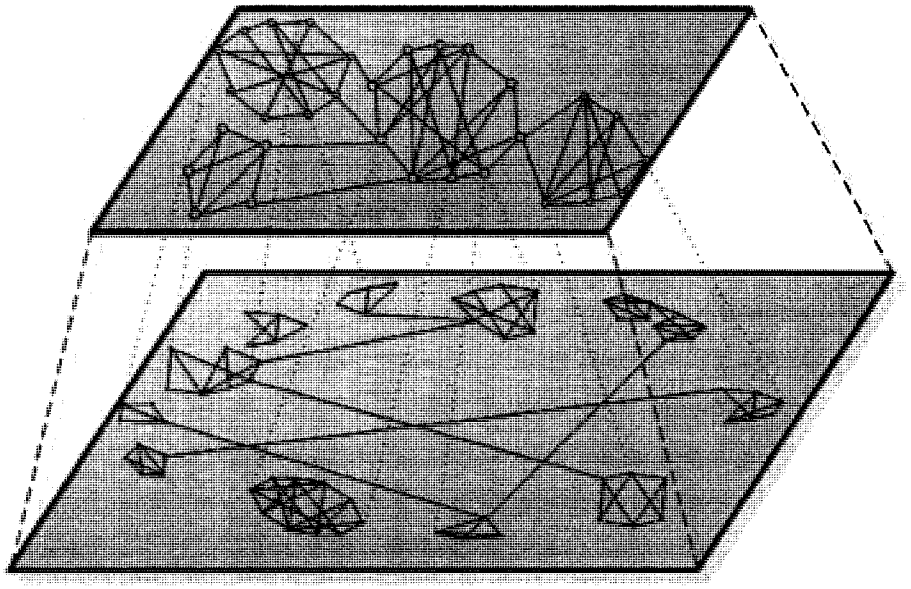


FIG. 6: A schematic representation of the yeast proteomic network as a hub-cluster interaction network. The top level corresponds to a global network of hub proteins, and the bottom level to a local network of cluster proteins.

## IV.2 MATERIALS AND METHODS

### IV.2.1 $k$-Cores and $k$-Shells in Graphs

We begin by describing the concepts of a $k$-core and a $k$-shell in a graph, since our clustering method makes use of these.

Given a natural number $k$, the *$k$-core* of a graph $G$ is the maximal subgraph of $G$ in which every vertex has degree at least $k$ in the subgraph (provided it is not the empty graph). The $k$-cores in a graph are nested: the $(k+1)$-core is contained in the $k$-core, for $k = 0, 1, \ldots, K-1$, where $K$ is the value of the maximum core in the graph. The $k$-core of a graph need not be a connected subgraph even if the original graph is connected. Note that if a graph contains a $k$-vertex connected component or a clique on $k+1$ vertices, then it is contained in a $k$-core; however, the $k$-core need not contain a $k$-connected subgraph or a clique on $k+1$ vertices.

The *$k$-shell* of a graph is the set of vertices that belong to the $k$-core, but not to the $(k+1)$-core. The $k$-shell includes vertices with degree $k$ from the $k$-core, but also other vertices whose degree in the residual graph becomes less than $(k+1)$ when low degree vertices are removed.

There is a well-known linear-time algorithm, in the number of edges, for computing the $k$-core (indeed, for finding all $k$-cores, for $k = 0$ to the maximum core value) of a graph. The idea is to repeatedly remove vertices $v$ of degree less than $k$ from the graph and all edges incident on $v$, updating the degrees of the neighbors of $v$ in the residual graph as edges are deleted. The algorithm repeats this step until all vertices that remain have degree $k$ or higher in the residual subgraph.

We have extended the concept of a $k$-core to a hypergraph in earlier work [1]. $k$-cores have been used earlier for clustering proteomic networks as a way of identifying highly connected subnetworks and for removing proteins belonging to low shell values [36].

We claim that clustering the $k$-core of a network removes noise in the data, in the same spirit as computing a shared nearest neighbor similarity (SNN) network. In an SNN network, two vertices are joined by an edge with weight equal to the number of their common neighbors at a distance less than or equal to $d$, where $d$ is a natural number parameter. The SNN network includes only those edges that have weight higher than a threshold, and clustering algorithms have been designed to work with this network [37]. Unfortunately for large networks of small average path lengths, the computation of the SNN network can be prohibitively expensive. We suggest that the $k$-core is an efficient

way to compute a network that approximates an SNN network. Every vertex in a $k$-core is adjacent to at least $k$ other vertices in the subgraph, each of which is adjacent to $k$ vertices with high core values.

### IV.2.2 Clustering Algorithms

Three major clustering approaches have been employed to identify functional modules in proteomic networks. The first approach searches for subgraphs with specified connectivities, called network motifs, and characterizes these as functional modules or parts of them. A complete subgraph (clique) is one such candidate, but other network motifs on small numbers of vertices have been identified through exhaustive searching or statistical methods [38]. This approach is not scalable for finding larger clusters in large-scale networks. The second approach, recently proposed in this context by Bader and Hogue [36], computes a weight for each vertex (depending on the density of a maximum core in the neighborhood of the vertex); it then grows a cluster around a seed vertex, a vertex with the largest weight in the currently unclustered graph. A vertex in the neighborhood of a cluster is added to it as long as its weight is close (within a threshold) to the weight of the seed vertex. Once a cluster has been identified, the procedure is repeated with a vertex of largest weight that currently does not belong to a cluster as the seed vertex. However, our experience comparing this approach with the spectral algorithms that we describe next shows that this method is less stable than the latter (i.e., the clusters depend on the seed vertices chosen).

We now discuss a spectral algorithm for clustering.

Let $G = (V, E, W)$ denote a weighted graph with vertex set $V$, edge set $E$, and weights on the edges $W$. Consider the problem of partitioning $V$ into two sets $V_1 \cup V_2$. We consider the weights

$$W_{il} \equiv W(V_i, V_l) = \sum_{j \in V_i, k \in V_l, (j,k) \in E} w_{jk},$$

where $i, l = 1, 2$. Minimizing the objective function

$$J(V_1, V_2) = \frac{W_{12}}{W_{11}} + \frac{W_{12}}{W_{22}}$$

minimizes the sum of weights of the edges between distinct clusters, while simultaneously maximizing the sum of the weights of the edges within each cluster. This objective function for clustering has been called the MinMaxCut [39], and it measures a ratio related to the *separability* of a cluster to its *cohesion*. We prefer this function to related objective functions that have been proposed such as Normalized Cut.

Let $Q$ denote the Laplacian matrix of a graph with weights $w_{ij}$ on its edges $(i, j)$; thus $q_{ij} = -w_{ij}$ for $i \neq j$, and each diagonal element $q_{ii}$ is the sum of the weights of the edges incident on the vertex $i$. Let $D$ be a diagonal matrix with its $i$-th component $d_{ii} = \sum_{(i,j)\in E} w_{ij}$; $d_1 = \sum_{i\in V_1} d_{ii}$, and $d_2 = \sum_{i\in V_2} d_{ii}$. Let $p$ be a 'generalized partition vector' with $p_i = \sqrt{d_2/d_1}$ for $i \in V_1$; and $p_i = -\sqrt{d_1/d_2}$ for $i \in V_2$; let $e$ be the $n$-vector of all ones. Then we have $p^T De = 0$, and $p^T Dp = d_1 + d_2$. Ding et al. [40] have shown that

$$\min_{V_1,V_2} J(V_1, V_2)$$

is equivalent to

$$\min_p p^T Qp/p^T Dp, \quad \text{subject to} \quad p^T De = 0.$$

This minimization problem is NP-hard since the generalized partition vector $p$ is restricted to have elements from one of two values. However, we can relax this constraint and let $p$ take values from $[-1, +1]$ to obtain an approximate solution. This problem is solved by the eigenvector $x$ corresponding to the smallest positive eigenvalue of the generalized eigenproblem $Qx = \lambda Dx$.

The partition is obtained by choosing the vertices in one part to consist of vertices with eigenvector components smaller than a threshold value, while the other part has the remaining vertices. The threshold value could be chosen so as to locally minimize the MinMaxCut objective function. For details, see [41, 39].

A clustering method is obtained by recursively applying the spectral partitioning method, by splitting each current cluster into two subclusters. The MinMaxCut objective function can be used to determine if a given cluster should be split further.

### IV.2.3 Algorithm

The yeast protein interaction network under study has 2610 proteins and 6236 interactions; we work with its largest connected component, which has 2406 proteins and 6117 interactions.

In the first step, we separate the high degree proteins, which are candidates for *hub proteins*. A hub is a protein that connects several different clusters in the network together, and these form a subset of the high degree proteins. After some experimentation, we chose candidate hub proteins to be those with degree 15 or higher in the network we study. The residual network has 2241 proteins and 3057 interactions, and consists of 397 connected components. The largest connected component of the residual graph has 1773 proteins

and 2974 interactions (and hence most of the other components have few or no edges). We chose the largest component for further analysis.

In the second step, we compute the 3-core of the residual graph in order to remove the low- shell proteins (the 0-, 1-, and 2-shells) from the network. As discussed earlier, we believe that this step removes some of the noise from the experimental protein interaction data. We have found that this step has two advantages. First, the clustering algorithms generate better clusters of the residual network; the low shell proteins can be assigned to a cluster after it has been identified. Second, this step reduces the graph size substantially since this is a modified power-law network with a large number of low degree proteins.

In the third step, we have applied the spectral clustering recursively to cluster the subgraph and identify the clusters, employing the MinMaxCut objective function. Once the clusters are identified, then the high-degree proteins which were removed as candidate hub proteins can be confirmed as hub proteins if they connect multiple clusters, or can be included among the cluster proteins.

Our spectral clustering code is currently written in Matlab for quick prototyping. The current code takes 65 seconds on a PC with a 1.3 MHz Intel processor and 768 MB memory. The hub and $k$-core computations are faster. Here we have greatly reduced the run times needed by removing the low-shell and hub proteins before clustering.

We have been concerned in this chapter with identifying a methodology that can successfully deliver biologically significant clusters in proteomic networks. Distributed computations will be needed when we consider larger proteomic networks such as the human, and networks consisting of heterogeneous data.

We are also concerned with scalable clustering algorithms. The proposed approach requires $O(|E| \log |V|)$ time, where $|E|$ is number of edges in the network, and $|V|$ is the number of vertices. The $k$-core computation and the eigenvector computation at each clustering step can be performed in time $O(|E|)$; and there are $\log |V|$ partitioning steps needed to cluster. The spectral clustering could be replaced with a multi-level clustering approach that can also be implemented in time $O(|E|)$.

## IV.3 RESULTS

### IV.3.1 Data Source and Analysis

Among the protein interactions produced by high-throughput methods such as the yeast 2-hybrid experiment or tagged affinity purification (TAP) [8, 18, 10], there are many false positives due to experimental limitations as well as biological factors (proteins that are not expressed at the same time or in the same cellular locale) [35]. In order to reduce the interference by false positives, we focused on the protein interaction network from the Database of Interacting Proteins (DIP), circa. April 2004 (URL: `dip.doe-mbi.ucla.edu/dip/`), consisting of the reliable dataset, which includes only data determined by a small-scale experiment, confirmed by independent high-throughput experiments, or scored highly by a probabilistic method that estimates the reliability of an interaction. This dataset has 2610 proteins that involve 6236 interactions considered to be reliable with high confidence.

### IV.3.2 The Cluster and Hub Networks

The local network computed by the clustering algorithm on the yeast protein interaction network, from which high degree proteins (hubs) and low-shell proteins have been removed, is shown in Fig. 7. Colors are used to distinguish the proteins belonging to a cluster, although some colors are reused to color proteins belonging to clusters that are drawn sufficiently far from each other. Thirty-eight clusters are displayed; for clearer presentation, we have omitted the edges joining two clusters when fewer than three edges join a cluster to another. All edges joining proteins within each cluster are shown.

The sum of the numbers of within-cluster edges is 984, while the sum of the between-cluster edges is 239, and the largest number of edges joining one cluster to another is 9. These measures are related to the concepts of *cohesion* and *separation* of the clustering [42], and thus we believe that our method has been able to cluster the residual network well. Each of the clusters is assigned to multi-protein complexes using the Munich Information Center for Protein Sequences (MIPS) database (URL: `mips.gsf.de`), as described in the next subsection. Each low-shell protein can now be easily assigned to a cluster with whose proteins it has the most number of interactions.

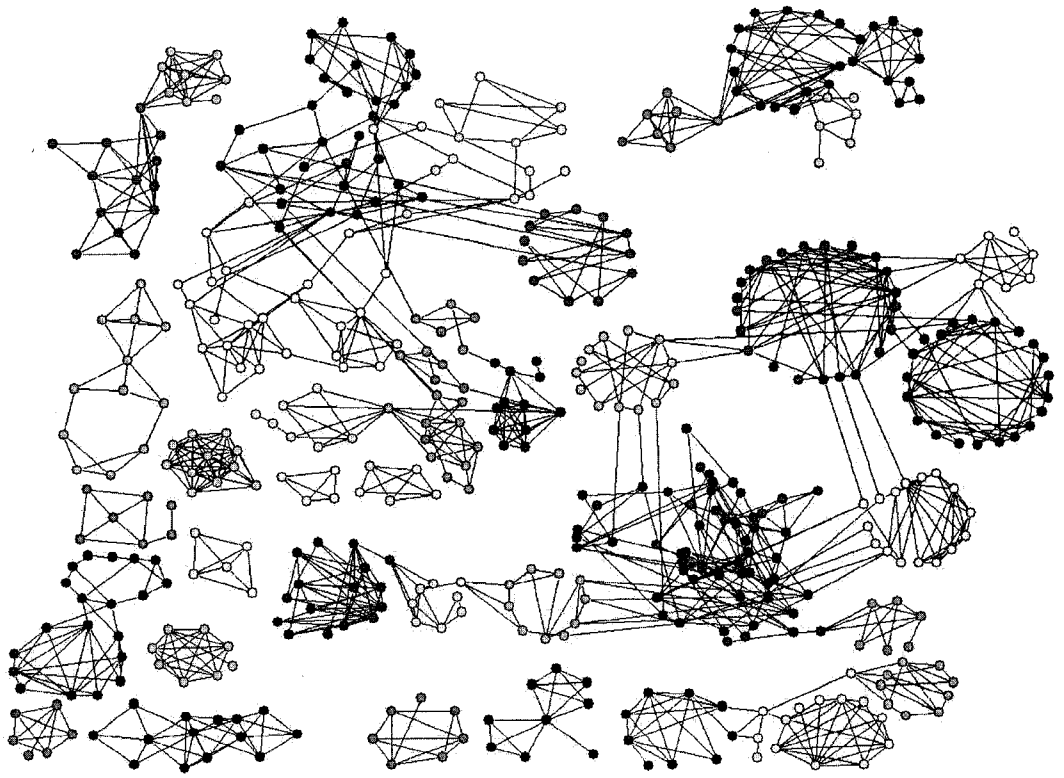


FIG. 7: Clusters in the yeast proteomic network from which hub and low-shell proteins have been removed. When fewer than three edges join a pair of clusters, such edges have not been drawn in this figure, for clarity in presentation.

From a topological point of view, our approach to clustering helps to uncover the hidden topological structure of a proteomic network. We found that there are two major subnetworks within the protein-protein interaction network. In addition to the *cluster network*, we also construct a *hub network*, the subnetwork formed by the hub proteins in the protein interaction network; a subnetwork formed by the 5-core of the hub network is shown in Fig. 8. Four 'super-clusters' are clearly evident in the hub interaction network: from top to bottom, these correspond to the spliceosome, proteins involved in mRNA export and the nuclear pore complex, the regulatory subunit of the proteasome, and proteins that are transcription factors.

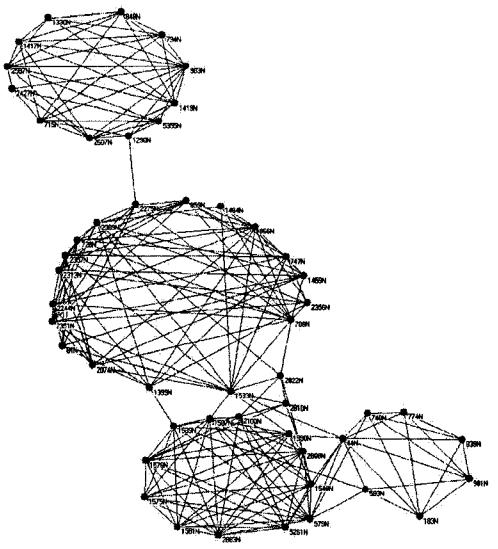


FIG. 8: The 5-core of the global hub network. The four clusters evident in this figure correspond to the spliceosome, mRNA export, the proteasome, and various transcription factors.

We now consider various subnetworks of the yeast protein interaction network to illustrate the differences between the 'global' hub network, and the local 'cluster' network. Table II lists the sizes of these networks, the average path lengths, the diameters, and the cluster coefficients. (The cluster coefficient measures how likely two neighbors of a vertex are to be adjacent to each other in the network, on the average.) The row 'C + S' denotes the 'cluster and shell' subnetwork obtained by removing the hub proteins from the whole network. Note that this subnetwork has the highest diameter and average path length, due to the presence of the large number of low-shell proteins. Once they are removed, the cluster network exhibits the highest clustering coefficient, supporting our premise that this is a local network. The hub network has the lowest diameter and average path length due to the edges joining the hub proteins to each other (cf. Fig. 8). The tight clustering seen in the hub network was surprising to us, but it is clear that hub proteins preferentially interact with cluster proteins and with each other, rather than the low shell proteins. We discuss the hub subnetwork and clusters in it in more detail in the next subsection.

TABLE II: Properties of various subnetworks of the yeast protein interaction network.

| Subnetwork | No. of vertices | No. of edges | Average Path Length ($\ln n$) | Diameter | Cluster Coefficient |
|---|---|---|---|---|---|
| Hub | 165 | 507 | 3.5 (5.1) | 7 | 0.37 |
| Cluster | 495 | 1223 | 6.5 (6.2) | 16 | 0.43 |
| C +S | 1773 | 2974 | 7.6 (7.5) | 19 | 0.15 |
| Whole | 2406 | 6117 | 5.1 (7.8) | 13 | 0.21 |

The average path lengths in these networks are compared against $\ln n$, where $n$ is the number of vertices in each subnetwork. Random power-law networks with exponent $\beta$ satisfying $2 < \beta < 3$ have expected average path lengths of order $\ln \ln n$, while if the exponent $\beta > 3$, it is $\ln n$ [43]. We see that $\ln n$ is a good approximation for the average path length of the cluster and cluster-shell networks; but any network that includes the hub proteins has an even lower average path length.

### IV.3.3 Functional Annotation of Clusters

One way to validate the clusters we discovered is to check how homogeneous the proteins in each cluster are with respect to function or the biological process that they are involved in. Each cluster should consist of one or more multi-protein complexes, which are molecular machines responsible for various cellular functions. We compared 38 clusters that we found with multi-protein complexes listed in the MIPS database. We found that in thirteen of the MIPS protein complexes, every protein in the complex was also identified in a cluster corresponding to it; for nine more complexes, we found more than half the proteins involved in the complex in a corresponding cluster. These results are despite the facts that hub and low-shell proteins are not included in this comparison, and that many proteins in the MIPS database are not included in the DIP protein network under study here. When the hubs and low-shell proteins are included, the coverage will increase further.

We should note that, in general, the clusters that we have discovered contain more proteins than those reported to belong to a corresponding MIPS complex. This suggests possible biological roles and functional assignments for such proteins, many of which are not currently functionally annotated.

### IV.3.4 Interactions between the Hub and the Local Networks

We now consider the hub protein subnetwork and its interaction with the local network in more detail.

One of the complexes in cluster 8, the U4/U6 x U5 tri-snRNP complex, is comprised of a group of proteins involved in spliceosome processing of mRNA. This is the top-most cluster represented in Fig. 8. The spliceosome is required for the ordered and accurate removal of intronic sequences from pre-mRNA and thus plays a key role in alternative splicing, a process of great importance in higher eukaryotes whereby a single gene can generate multiple transcripts (alternatively spliced mRNAs) and thus multiple proteins [44]. The PRP (pre-mRNA processing) and Sm family proteins make up most of the proteins found in cluster 8. Some of the key proteins involved in mRNA processing, including those belonging to the LSM family, are not found in that cluster, but among the hub proteins that interact with multiple clusters.

One of the complexes in cluster 24, the first mRNA cleavage factor complex, includes five proteins involved in mRNA cleavage in preparation for the addition of the eukaryotic signature poly-A tail. Thus, proteins including CLP1 (involved in cleavage of the 3′ end of the mRNA prior to tailing), and RNA 14 and 15 (two proteins that participate with CLP1 in formation of the 3′ end of mRNA), are collectively implicated in alternative selection of the poly-A addition site [45].

Now we focus our attention on the single protein-protein edge which joins the top-most hub cluster in the figure, corresponding to *mRNA splicing*, to a second hub cluster involving mRNA export and nuclear pore formation proteins, corresponding to *mRNA export*, in Fig. 8. The two hub proteins that form the bridge between these clusters are PRP6, a component of the mRNA splicing machinery, and PAB1, the poly-A binding protein involved in the final step in mRNA processing. We note that PRP6 is involved in the later stages of mRNA splicing and is in that sense the penultimate step prior to poly-A tailing. Thus, the overall logic of joining these two complexes by these particular hub proteins is compelling.

We now examine the connections formed by these two hub proteins with the local clusters that we picture as lying below them in the hierarchy of global (hub) and local (cluster) networks (see Fig. 6). PRP6 interacts with a single cluster (cluster 8) through the protein SMD1. SMD1 further interacts with splicing proteins PRP3 and SMD3 in the hub complex that includes PRP6. SMD1 is involved in the early stages of mRNA splicing and is highly conserved, showing greater than 40% amino acid identity between yeast and human [46].

PRP6 interacts with PAB1 in the second hub complex. PAB1 in turn interacts with three clusters, 22, 24 and 34. As noted above, cluster 24 includes RNA14 and RNA15, both involved in mRNA cleavage, and it is these proteins that interact with PAB1. PAB1 also forms connections with cluster 22 (via its interaction with TIF4632 = eIF4F), and with cluster 34 (via PKC1). These latter interactions (eIF4F and PKC1) are at first glance puzzling, but in fact they are entirely consistent with emerging evidence of interactions and regulatory loops that exist between distinct components in the gene expression machinery. The poly-A terminus of mRNA, and the associated PAB1, not only interacts with the 5′ end of the mRNA, ensuring structural integrity of the transcript prior to its participation in protein translation [47] , but the PAB1 terminus also interacts with the translation machinery itself, and specifically with eIF4 initiation factors. Finally, PKC1, a protein kinase crucial to cell signaling pathways, is also implicated in functional interactions with PAB1 and eIFs [48], suggesting global level regulation of protein synthesis from metabolites through mRNA processing.

At the global level of our network model, we find key proteins involved in rate-limiting steps of gene expression, linked in logical order; these are connected to the local network consisting of clusters of proteins involved in execution level functions. Whether this overall pattern is typical of the proteome organizational structure we have identified here, remains to be further investigated.

### IV.3.5 Incorporation of Protein Domain Data

Proteins interact with each other through regions that have a specific sequence and fold, called domains. Here we further validate the protein complexes predicted from our clustering approach using information on the domain structure of proteins.

The study of proteins involved in processing eukaryotic mRNAs indicate that virtually all steps involved in gene expression are coordinated and integrated via protein-protein interactions. The LSM proteins provide an informative example of the integration of cellular protein machinery to couple synthesis and quality control in gene expression [49, 50]. LSM proteins form heptameric complexes that bind to RNA molecules; one such complex is found primarily in the nucleus where it coordinates splicing of mRNAs, while a second, related, complex of LSM proteins assembles in the cytoplasm to monitor mRNA quality control. LSM proteins have been extensively characterized and include two highly conserved protein interaction domains, SM1 and SM2. It is proposed that these conserved domains permit each LSM protein to interact with two other LSM proteins in forming the

heptameric, doughnut shaped ring structure that is implicated in mRNA splicing. LSM proteins comprise a gene family in which successive rounds of gene duplication have increased LSM copy numbers. LSM proteins have also been shown to form stable interactions with other protein types, including the PRP proteins discussed below.

The PRP proteins similarly carry a protein-protein interaction motif, the tetratrico peptide repeat (TPR) [51]. PRP proteins typically contain multiple copies of the 34 amino acid repeat; Prp1 for example contains 19 repeats of the TPR (ibid). Some PRP proteins contain a second conserved site at the C-terminus of the protein that facilitates interactions between them and LSM proteins, thus coupling two complexes with key roles in mRNA splicing. Our hub network predicts the formation of a complex containing LSM proteins 1-5, 7 and 8, as well as PRP proteins 4, 6, 8, and 31. The specific interactions implicated in our subnetwork are, to our knowledge, the first explicit assignments of interactions between these two families of proteins.

We were surprised to find that the hub proteins form a highly interconnected subnetwork. Biological evidence indicates that LSM proteins do indeed form multi-protein complexes in the course of performing their key cellular functions. The fact that each LSM protein has at least two protein-protein interaction domains helps us understand how the complexes are formed. Whether similar binding interactions can account for other closely knit hub networks is under investigation.

## IV.4 CONCLUSIONS

We have proposed a two-level architecture for a yeast protein-protein interaction network. We place a small set of hub proteins, each with at least fifteen interaction partners and involved in gene expression, mRNA export, the proteasome, and transcription factors, into a global subnetwork. A local subnetwork of proteins is organized into clusters that correspond well with multi-protein complexes in the MIPS database. We used the computed clustering to examine the biological significance of some of the interactions observed between the hub and local subnetworks. If the proposed two-level architecture exists in other proteomic networks, then it would be interesting to discover properties that distinguish hub proteins from the proteins in the local network.

In the next chapter, we will consider the computation of an overlapping clustering rather than the exclusive clustering approach considered in this chapter, so that a protein could be included in more than one cluster in the local network. We will also investigate additional clustering approaches and biological networks involving heterogeneous data.

# CHAPTER V

# OVERLAPPING CLUSTERING

## V.1 INTRODUCTION

The complete and systematic analysis of protein-protein interactions networks, which is critical for the understanding of cellular organizations, processes and functions, is a challenging task. The protein interactions provide potentially useful insights into functional associations between proteins [33]. Most current existing clustering techniques of protein interaction networks are concerned with discovering functionally valuable information hidden in the networks by identifying functional modules within the network; a functional module is a group of proteins that perform a specific task. Most of these techniques assume that each node in the graph must be assigned to exactly one cluster which represents coherent functional module. Functional modules within the cell may share proteins. In addition, proteins may participate in multiple cellular components. Therefore, biological networks, especially protein networks, should favor overlapping clustering, wherein some proteins are allowed to be members of one or more discovered clusters. Therefore, when clustering protein networks, it is appropriate to assign proteins to multiple overlapping clusters.

There are several difficulties detecting functional modules from the protein interaction networks using exclusive clustering approach. First, interaction data is not reliable. large-scale experiments to find protein-protein interactions have yielded numerous false positives and negatives. Second, a protein can be included into two or more cellular components.

Several topology-based graph clustering methods have been recently applied to protein interaction networks. However, these approaches have some drawbacks when being applied to these enormous, intricate networks. They are computationally problematic and not effective with scale-free networks since they rely on greedy searching. Also, they are not able to produce complete clusters, which correspond to real functional modules. They focus on detecting densely connected subgraphs as clusters. However, in this definition of clusters, a substantial number of sparsely connected nodes in scale-free networks [52] are not clustered in the appropriate way. These kind of clusters are ignored in the analysis even though they may have biologically important modules. This situation results in a large number of discards. On the other hand, clustering methods should be able to cluster

as many nodes as possible. Finally, the clustering methods should also be capable of fine tuning the parameters such that the resulting clusters vary in size and number of clusters.

In this chapter, we design a new algorithm that overcomes the limitation of the clustering methods introduced in the previous chapter. Moreover, the new algorithm introduces an overlapping clustering rather than an exclusive clustering.

## V.2 BACKGROUND

### V.2.1 $k$-Cores and $k$-Shells in Graphs

The *$k$-core* of a graph $G$ is a maximal subgraph of $G$ in which every vertex has degree at least $k$ in the subgraph. The $k$-core need not be a connected subgraph even if the original graph is connected. The $k$-cores in a graph are nested: the $(k+1)$-core is contained in the $k$-core, for $k = 0, 1, \ldots, K-1$; here $K$ is the maximum core value. The *$k$-shell* of a graph is the set of vertices that belong to the $k$-core, but not to the $(k+1)$-core.

There is a well-known linear-time algorithm (in the number of edges) for computing the $k$-core (indeed, for finding all $k$-cores, for $k = 0$ to the maximum core value) of a graph. The idea is to repeatedly remove vertices $v$ of degree less than $k$ from the graph and all edges incident on $v$, updating the degrees of the neighbors of $v$ in the residual graph as edges are deleted. The algorithm repeats this step until all vertices that remain have degree $k$ or higher in the residual subgraph.

### V.2.2 Spectral Clustering

A clustering of a graph corresponds to a grouping of the vertices into subsets such that vertices in each subset are similar to each other and dissimilar to vertices in other subsets. Often, in a graph a vertex is chosen to belong to a cluster when it has more neighbors inside the cluster than outside it.

Spectral clustering is a powerful tool to reveal high-level substructures within large-scale data that can be represented as graphs or matrices. We apply a spectral clustering method to identify subnetworks and clusters in protein protein interaction networks.

We now consider spectral algorithms for clustering. A spectral clustering algorithm is obtained by recursively applying a spectral method for graph partitioning, which has been studied since the 1970's.

Let $G = (V, E, W)$ denote a weighted graph with vertex set $V$, edge set $E$, and weights on the edges $W$. Consider the problem of partitioning $V$ into two sets $V_1 \cup V_2$. We consider

the weights

$$W_{il} \equiv W(V_i, V_l) = \sum_{j \in V_i, k \in V_l, (j,k) \in E} w_{jk},$$

where $i, l = 1, 2$. Minimizing the objective function

$$J(V_1, V_2) = \frac{W_{12}}{W_{11}} + \frac{W_{12}}{W_{22}}$$

minimizes the sum of weights of the edges between distinct clusters, while simultaneously maximizing the sum of the weights of the edges within each cluster. This objective function for clustering has been called the MinMaxCut [39].

Let $Q$ denote the Laplacian matrix of a graph with weights $w_{ij}$ on its edges $(i, j)$; thus $q_{ij} = -w_{ij}$ for $i \neq j$, and each diagonal element $q_{ii}$ is the sum of the weights of the edges incident on the vertex $i$. Let $D$ be a diagonal matrix with its $i$-th component $d_{ii} = \sum_{(i,j) \in E} w_{ij}$; $d_1 = \sum_{i \in V_1} d_{ii}$, and $d_2 = \sum_{i \in V_2} d_{ii}$. Let $p$ be a 'generalized partition vector' with $p_i = \sqrt{d_2/d_1}$ for $i \in V_1$; and $p_i = -\sqrt{d_1/d_2}$ for $i \in V_2$; let $e$ be an $n$-vector of all ones. Then we have $p^T De = 0$, and $p^T Dp = d_1 + d_2$. Ding et al. [40] have shown that

$$\min_{V_1, V_2} J(V_1, V_2)$$

is equivalent to

$$\min_p p^T Qp / p^T Dp, \quad \text{subject to} \quad p^T De = 0.$$

This minimization problem is NP-hard since the generalized partition vector $p$ is restricted to have elements from one of two values. However, we can relax this constraint and let $p$ take values from $[-1, +1]$ to obtain an approximate solution. This problem is solved by the eigenvector $x^*$ corresponding to the smallest positive eigenvalue of the generalized eigenproblem

$$Qx = \lambda Dx.$$

The clustering is obtained by applying the $k$-means clustering algorithm on $x^*$ with $k = 2$ to choose the threshold value to split the cluster so that the MinMaxCut objective function value is as small as possible.

The K-means clustering algorithm is a simple way of classifying a group of data objects into a predefined number ($k$) of clusters. The main idea is to select appropriate geometric centers, *centroids*, one for each cluster, and place an object in the cluster with the closest centroid. The distance between each object and each cluster centroid is then calculated. The distance is square root of the componentwise square of the difference between the

object's position and the centroid. Since we are only comparing the results, we can omit the square root. One advantage of this metric is that the distance is a sphere around the centroid. Finally, the position of the centroid is recalculated based on the position of the objects that belong to that cluster. The process is repeated until the centroids no longer move.

### V.2.3 Bridge Identification

We tested the possibility of computationally detecting *bridge* proteins within the protein network. We define proteins that are significantly connected to at least $k$ cluster as *bridge* proteins. Mathematically, let $C = C_1, C_2, ..., C_m$ be the set of conserved clusters, results from the partial exclusive clustering methods. For each vertex $v \in V$, let $N(v)$ be the set of vertices adjacent to $v$ in $G$, and let $N_i(v)$ be the vertices in $C_i$ that are adjacent to $v$ in $G$. For a given vertex $v$ and every cluster $C_i$, we measure the significance of $N_i(v)$ using $p-$ value obtained by hyper-geometric density function defined below:

$$p = 1 - \sum_{x=0}^{|N_i(v)|-1} \frac{\begin{pmatrix} |C_i| \\ x \end{pmatrix} \begin{pmatrix} |V \setminus C_i| \\ |N(v)| - x \end{pmatrix}}{\begin{pmatrix} |V| \\ |N(v)| \end{pmatrix}}.$$

A vertex $v$ is considered to be a bridge if there exists a subset $C' \subseteq C$ such that $|C'| \geq k$ and for every cluster $C_j \in C'$, $|N_j(v)| > l$ and $|N_j(v)|$ is significant, where l is a network specific threshold. For example, in the yeast HC study $k, l = 3$ are used, and a p-value is significant if it is less than 0.05.

### V.2.4 Mutual Information

The mutual information (MI) of two variables is the quantity that measures the mutual dependence of the two variables. Formally, the mutual information of two discrete random variables X and Y can be defined as:

$$\mathrm{MI}(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)},$$

where $p(x,y)$ is the joint probability distribution function of $X$ and $Y$, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of $X$ and $Y$.

Terms and attributes in the GO database should be as independent as possible, because we would like to avoid over weighting attributes that have many ancestors or descendents assigned to highly overlapping gene sets. We measured independence using the uncertainty coefficient $U(X,Y) = \mathrm{MI}(X,Y)/\max_{i,j}(\mathrm{MI}(i,j))$, where MI is the mutual information between two attributes $X$ and $Y$. High $U-$values, possession of either attribute removes a large amount of uncertainty about possession of the other. And for low $U-$values, knowledge of either removes little uncertainty about the other.

If any two attributes are shared by the same collection of genes, one of the attributes should be excluded from the database so as not to consider what is essentially the same attribute twice. To determine which attributes were essentially the same, a threshold of $80\%$ was set for the U-value. If the value was over the threshold, the attributes were considered to be similar enough to each other to exclude one from consideration. If under the threshold, both attributes were distinct enough to be considered separately for the purposes of our analysis.

### V.2.5 Quality Assessment

We consider two different kinds of approaches for quality assessment. All of the measures quantify the similarity between a given collection of conserved clusters and a reference data. As a reference we used known yeast complexes catalogued in the MIPS and attributes (terms) from the gene ontology (GO) databases. The first set of techniques uses measure to obtain the statistically significant matches between conserved clusters and the reference data such as *precision* ($P$), *recall* ($R$) and $F-$*score* ( the harmonic mean of precision and recall). The second group of methods uses measures from classification, such as *sensitivity* (Sn), *positive predictive value* (PPV) and *accuracy* (Acc) (the geometric mean of sensitivity and positive predictive value). The second group methods evaluate the extent to which a cluster contains proteins from a single complex in the reference data.

In the first approach, we measure the level of correspondence between conserved clusters and complexes by computing statistically significant matches between the two collections using overlap scores (or hyper-geometric p-value), and used these matches to evaluate the precision and recall of the suggested clustering solution as follows. Let $\mathcal{C}$ be the initial set of conserved clusters, and let $\hat{\mathcal{C}} \subseteq \mathcal{C}$ be the subset of clusters that had a significant match. The *precision* of the solution is defined as $|\hat{\mathcal{C}}|/|\mathcal{C}|$. Let $\mathcal{M}$ be the set of complexes cataloged in the reference database, and let $\hat{\mathcal{M}} \subseteq \mathcal{M}$ be the subset of complexes with a significant match by a conserved cluster. The *recall* of the solution is defined as $|\hat{\mathcal{M}}|/|\mathcal{M}|$.

In the second approach, we measure the degree to which conserved clusters correspond to complexes in the reference database. Let $T_{ij}$ be the number of proteins of complex $i$ in cluster $j$.

The *complex-wise sensitivity* $\mathrm{Sn}_{co_i}$ of complex $i$ is $\mathrm{Sn}_{co_i} = \max_j \mathrm{T}_{ij}/N_i$, where $N_i$ is the number of proteins belonging to complex $i$. $\mathrm{Sn}_{co_i}$ reflects the coverage of complex $i$ by its best-matching cluster. The overall sensitively Sn of a clustering is weighted average of $\mathrm{Sn}_{co_i}$ over all complexes.

Similarly, we also compute the *cluster-wise positive predicted value* $\mathrm{PPV}_{cl_j}$ of cluster $j$ is $\mathrm{PPV}_{cl_j} = \max_i(\mathrm{T}_{ij}/\sum_i \mathrm{T}_{ij})$. $\mathrm{PPV}_{cl_j}$ reflects the reliability with which cluster $j$ predicates that a protein belongs to its best matching complex.

The overall cluster-wise positive predicted value PPV of a clustering is weighted average of $\mathrm{PPV}_{cl_j}$ over all clusters.

The overall clustering-wise accuracy Acc of a clustering is the geometrical mean of Sn and PPV values.

Finally, we have evaluated the *separation* statistic $\mathrm{Sep}_{ij}$ which indicate weather a cluster $j$ contains only protein of a particular complex $i$ and all proteins of that complex $i$. Also, this statistic deals effectively with multiple assignations. It penalize cases where proteins of a given complex are assigned to multiple clusters. The *separation* statistic is defined by the following equation.

$$\mathrm{Sep}_{ij} = \frac{T_{ij}}{\sum_j T_{ij}} \cdot \frac{T_{ij}}{\sum_i T_{ij}}.$$

The *complex-wise separation* $\mathrm{Sep}_{co_i}$ of a complex $i$ is $\mathrm{Sep}_{co_i} = \sum_j \mathrm{Sep}_{ij}$. Similarly, The *cluster-wise separation* $\mathrm{Sep}_{cl_j}$ of a cluster $j$ is $\mathrm{Sep}_{cl_j} = \sum_i \mathrm{Sep}_{ij}$.

To estimate a clustering results as a whole, clustering-wise $\mathrm{Sep}_{co}$ and $\mathrm{Sep}_{cl}$ values are computed as the averages of $\mathrm{Sep}_{co_i}$ over all complexes, and of $\mathrm{Sep}_{cl_j}$ over all clusters, respectively.

The overall clustering-wise separation Sep of a clustering is the geometrical mean of $\mathrm{Sep}_{co}$ and $\mathrm{Sep}_{cl}$ values.

### V.2.6 Hyper-geometric $p-$value

This is an alternative measure to use instead of the overlapping score. Specifically, for each conserved cluster $C$ we found a complex/attribute $A$ in the reference database with the least $p-$ value based upon the hyper-geometric test:

$$p = 1 - \sum_{i=0}^{k-1} \frac{\binom{N}{i}\binom{T-N}{M-i}}{\binom{T}{M}},$$

where $M, N$ are the total number of proteins in the cluster $C$ and complex/attribute $A$ respectively, $T$ is the total number of proteins in our data set that are spanned by reference database and $k = |C \cap A|$. In this analysis we consider only proteins that appear in both our protein data set and reference database.

### V.2.7 Bridge Importance

To measure the importance of bridge proteins, we computed statistically significant Binomial $p-$values between the bridge protein collections and the essential protein collections. $p-$values measure whether the difference is significant between the test (bridge set) and control (essential set) groups. They are calculated using a cumulative binomial distribution:

$$p = 1 - \sum_{i=0}^{k-1} \binom{N}{i} p^i (1-p)^{N-i},$$

where $N$ is the total number of proteins in the data, $k$ is the number of observed proteins with a specific property (e.g.,essentiality) in the testing group and $p$ is the probability of finding a protein with the same property in the control group. In this manner,we are testing whether proteins with a specific property are over represented compared with the control group.

## V.3 CLUSTERING APPROACHES

### V.3.1 Related Works

Three major clustering approaches have been employed to identify functional modules in proteomic networks. We give hereafter a short conceptual descriptions. More information can be found in the comparison section (see Sec. V.6).

The first approach searches for subgraphs with specified connectivities, called network motifs, and characterizes these as functional modules or parts of them. A complete subgraph (clique) is one such candidate, but other network motifs on small numbers of vertices have been identified through exhaustive searching.

The second approach, the graph growing approach, a cluster grows around a seed vertex using graph search algorithms (greedy algorithms). These are local algorithms that begin with single, or several known nodes, then expand from there. The MCODE algorithm (Bader and Hogue [36]) computes a weight for each vertex (depending on the density of a maximum core in the neighborhood of the vertex); it then grows a cluster around a seed vertex, a vertex with the largest weight in the currently unclustered graph. A vertex in the neighborhood of a cluster is added to it as long as its weight is close (within a threshold) to the weight of the seed vertex (see Sec. V.6).

Similarly, Bader [53] proposed the SEEDY algorithm which progressively adds on proteins to a seed protein to form complexes, based on a particular distance metric. Another software package called Complexpander by Asthana et. al. [54] functions in this way to help identify protein complexes including the seed proteins from a PPI network. However, our experience comparing this approach with the graph (global) clustering approach that we describe next shows that this approach is less stable than the latter (i.e., the clusters depend on the seed vertices chosen).

The third approach, the graph clustering approach, is a broad description though, and algorithms in this category attempt to maximize or minimize certain cluster measures such as connection density, edge cut, or a novel distance metric between nodes in a cluster. In general, these are global algorithms that optimize an objective function for the whole graph. One algorithm by Spirin and Mirny [38] employs super–paramagnetic clustering (SPC), which is a technique based on a principle observed in physics to maximize the cluster density. Another algorithm by Przulji et al. [55] takes the reciprocal approach in which they uses the concept of a minimal cut, which is a division of the nodes of the network into two complementary sets such that the least number of edges cross from one set to the other. In their method, they perform recursive minimal cuts until they end up with densely connected subgraphs. Another method by King et al. [56] called restricted neighborhood search clustering (RNSC) takes a slightly different tact. RNSC begins by randomly assigning nodes to clusters, then reassigns nodes so as to minimize a cost function. Yet another such method by Enright et al. [57] uses a method called Markov clustering (MCL) to simulate the "flow" of the matrix. It does this by calculating increasing powers of the network's adjacency matrix. With the increased powers, the areas of high flow become increasingly separated from those with little to no flow (see Sec. V.6).

The methods described so far are related to exclusive clustering. That is, these algorithms only permit nodes to be members of a single cluster. However, in biological systems many proteins and gene products participate in multiple functions. To address this problem, there also exist overlapping clustering methods. Pereira-Leal et al. [58] used MCL clustering algorithm in order to detect overlapping clusters. Their algorithm (lineG) first turns a network of individual genes, into a network of gene-gene interactions (the line graph of the input graph). Then, the MCL algorithm is used to cluster the network of interactions. Finally, the algorithm re-converts the identified clusters from an interaction-interaction graph back to a protein-protein graph. Since multiple interaction nodes may contain the same gene, thus when the interaction-interaction network clusters are converted back to their contained genes, the same gene can appear in multiple clusters. However, this algorithm is extremely computationally expensive $O(m^3)$, where m is the number of interactions) (see Sec. V.6).

Another overlapping algorithm is the clique percolation algorithm (cFinder) by Adamsek et al. [59]. It looks for clusters by first finding all possible k-cliques (that is, all possible sets of k nodes that are completely interconnected). Upon identifying all of the k-cliques, it defines communities (clusters) by grouping together all k cliques that share k-1 nodes. This gives highly connected regions, but still permits overlapping clusters as nodes in one k-clique may certainly be involved in another cluster. Again, this method is highly computationally intensive (again, $O(n^3)$) (see Sec. V.6).

So, while there are a number of algorithms for searching network structure for important subnetworks, all have trade offs. None fully balance the needs for global and local approaches, as well as permitting overlapping and exclusive clusters.

### V.3.2 Clustering Algorithm

Within this context, we propose a new algorithm to help moderate the different demands and purposes of a cluster analysis algorithm.

The new algorithm, Algorithm 3, begins by identifying the subnetwork with high degree nodes (hubs) as Hub subnetwork. The hub nodes are the nodes among the top 20 percent of the degrees. This Hub subnetwork is separated from the original network and considered separately. The remaining nodes and edges form a residual network that we will call the Local subnetwork. This helps to separate the areas influenced by high-degree hub nodes, which might prevent the detection of other types of clusters (clusters with less densities).

With the hub subnetwork removed, the residual local subnetwork still contains a large number of loosely connected nodes (since the network is a scale free graph). That is, there are still many nodes that have only a few neighbors, and they may again detract from cluster detection in the residual network, which does not contain the highest degree nodes of the overall network, since those were removed in the hub subnetwork. To this end, the algorithm then computes the 3−core of the residual local network. The 3−core is the sub-network in which all remaining nodes have 3 or more neighbors (that is, nodes with fewer than 3 neighbors are iteratively removed until the condition is met). The 3−core sub-network now consists of a far smaller, much denser set of groupings.

The next step is to separately cluster both the hub sub-network and the 3-core of the local sub-network, using spectral clustering method with different parameter sets for each subnetwork.

After identifying the clusters in each the hub sub-network and the 3−core of the local subnetwork, the algorithm merges appropriate clusters. That is, if there are clusters identified in different subnetworks in which they are heavy interconnections (heavy is determined by exceeding a threshold based on the MinMaxCut objective function), then the two clusters actually should represent a single cluster, and are re-attached. This permits the recapture of fine leveled clustering information that comes from a complexly structured cluster (where some is very highly hub-like, but part of it is less densely connected).

As mentioned above, a *bridge* is a node that joins $k$ or more clusters and it is linked to each of these clusters *significantly*. Determining whether a node has a significant number of links to a cluster is based upon the hyper-geometric $p-$value ( see Sec. V.2.3). Moreover, the hyper-geometric $p-$value is based upon the graph size, the cluster size, the number of links from that node to this cluster and the degree of this node.

**Algorithm 3** Clustering Algorithm high-level description.

1: *Identify* subnetwork with high degree proteins ( proteins among the top 20 percent of the degrees) as Hub subnetwork, and the residual subnetwork as the Local subnetwork.
2: *Compute* the 3-core of the Local subnetwork (remove the 0-, 1- and 2-shells).
3: *Cluster both* Hub subnetwork and 3-Core of the Local subnetwork using the exclusive spectral clustering algorithm (use different parameter sets for each subnetwork).
4: *Merge* clusterings from *both* Hub subnetwork and Local subnetwork in which they are heavy interconnections (heavy is determined by exceeding a threshold based on the "MinMaxCut" objective function).
5: *Assign* low-shell nodes to clusters in the order of decreasing core value, ordering with a core by decreasing numbers of already assigned neighbors using Bridge Searching Algorithm.
6: *Output* clusters and bridges found.

Once the clusters have been merged, the bridge searching algorithm outlined in Algorithm 4 examines nodes to be added to the cluster in an order determined by the core value and the number of unknown neighbors. This additional algorithm reintroduces those nodes that had been filtered out previously (i.e., the nodes in the 0, 1, and 2 shells of the local subnetwork). Each of these nodes is added back into the graph with a certain order (core value) and using the hyper-geometric p-value of membership in each of the clusters. Depending on which threshold it exceeds, the node may be added to zero, one, or more than one cluster. If a node takes membership in more than one cluster, it is considered a bridge node. This step of the algorithm not only adds in the previously set-aside nodes, but it also considers the outer ring of nodes in each one of the clusters, and checks to see if they are members of the other clusters. This way, the algorithm can thoroughly check for overlaps in clusters that may have been missed previously. Also, adding the nodes back in now means that their information is not lost in the final analysis, but their removal earlier on allows analysis of the highly connected central structures without the loosely-connected nodes obscuring them.

Since our proposed algorithm includes both global and local approaches, and allows for detection of overlapping clusters, it is thorough and robust. While it is unknown if it provides optimal cluster detection in all situations, the algorithm takes advantage of many of the different detection techniques to help address the complex, and heterogeneous nature of clusters in biological networks.

**Algorithm 4** Bridge Searching Algorithm

1: **for** each low-shell node with a given "order" **do**
2: **if** node is connected to one cluster only **then**
3: Assign this node to that cluster.
4: **else**
5: **for** each cluster connected to that node **do**
6: Compute the node's probability of being a member of that cluster, using the hyper-geometric distribution p-value.
7: **end for**
8: Assign node to clusters based upon the p value. Depending on which threshold it exceeds, the node may be added to one, or more than one cluster. Furthermore, nodes could be left unclustered.
9: **end if**
10: **end for**

### V.3.3 Complexity

The time complexity of our approach is dominated by the clustering step which requires $O(|E|.|V|)$ time, where $|E|$ is number of edges in the graph, and $|V|$ is the number of vertices. The eigenvector computation at each clustering step can be performed in time $O(|E|)$; and there are $|V|$ partitioning steps needed to cluster in the worst case scenario. The hub identification step requires $O(|V|)$ and the $k$-core computation step can be performed in time $O(|E|)$. The clusters merging step requires $O(|Cl|^2)$, where $|Cl| << |V|$ is the number of clusters in the clustering solution. Finally, we iteratively assign low shell nodes to clusters using the Bridge Searching Algorithm (Algorithm 4). The Bridge Searching Algorithm starts by sorting the vertices in the order of decreasing core value, ordering with a core by decreasing numbers of already assigned neighbors, this sorting step require $O(|V|.log(|V|))$ time. Next, the algorithm iteratively assigns low shell nodes to cluster. Each iteration in the assignment step requires $O(\Delta)$ time, where $\Delta$ is the maximum degree. Hence, the total time complexity of the Bridge Searching Algorithm is

$$O(|V|.\max(\Delta, log(|V|)).$$

## V.4 DATA SOURCE AND STRUCTURAL ANALYSIS

### V.4.1 Interaction Data

We study protein interaction networks from three organisms: yeast, human, and worm. For each organism, we use data from multiple sources, in order to demonstrate the broad validity of our approach. The interaction data are produced by high-throughput methods such as the yeast 2-hybrid experiment or tagged affinity purification (TAP) [8, 18, 10], or from small-scale studies reported in the literature. The high throughput studies are known to have false positive and false negative interactions, and hence some of the interaction databases also report a reliable subset of interactions, validated through multiple experiments or statistical methods.

For yeast, we study four different networks. The first source is interaction data from the Database of Interacting Proteins (DIP) [27], circa. April 2006 (URL: `dip.doe-mbi.ucla.edu/dip/`). From DIP, we have also used a smaller, *DIP-core* data set, which is deemed to be more reliable, by including only the data from small-scale experiments, or confirmed by independent high-throughput experiments, or scored highly by a probabilistic method that estimates the reliability of an interaction. The third yeast data set is from the Gerstein lab [60], which is obtained by aggregating interaction data from several published high-throughput and small-scale experiments. The fourth data source for yeast interactions is a high confidence (HC) dataset, which was created by Batada et al. [61] by unioning literature-curated data from five major interaction databases (BIND [62], BioGRID [63], DIP [27], MINT [64], and MIPS [25]), and all published high throughput interaction data [8, 18, 10]. This union generates a large multi-validated, high confidence (HC) network; The minimum criterion for inclusion in the multi-validated dataset is that the relevant interaction was independently reported at least twice. This is the primary yeast data set used in our study.

For human interaction data, we have used two databases. The first dataset was obtained from the Human Protein Reference Database (HPRD [65], where the interactions have been manually extracted from the literature by expert biologists who read, interpret and analyze the published data. The second dataset is from yeast two-hybrid data for the human published by the Vidal lab [66].

For the worm, we have used interaction data available from DIP and from yeast 2-hybrid data from the Vidal Lab [67].

Statistics on the sizes of these networks are shown in Table III. Within each organism, the networks are listed by increasing size. Proteins correspond to vertices in the network, and interactions correspond to edges joining pairs of vertices. The sizes of the largest connected component (the *giant component*) of each network are also tabulated. We study the giant components of the networks in this chapter, since the other components often consist of isolated vertices, or have few vertices. In addition, various network characteristics are tabulated in Table IV.

The degree distributions of the eight networks are tabulated in Table V. The columns list the mean degree, the first, second, and third quartile values, the degree threshold of the hub proteins (the proteins among the top 20 percent of the degrees), and the maximum degree in the network. An interesting observation is that the mean degree, the maximum degree, and the degree threshold for being a hub, all increase with network size. This is in agreement with previous findings about networks as larger numbers of proteins are involved in a proteomic study.

TABLE III: Network sizes. The number of proteins is $|V|$, and the number of interactions is $|E|$. The numbers $|V^*|$ and $|E^*|$ correspond to the largest connected component of the network.

| Organism | Network | $\|V\|$ | $\|E\|$ | $\|V^*\|$ | $\|E^*\|$ |
|---|---|---|---|---|---|
| Yeast | DIP-core | 2,610 | 6,236 | 2,406 | 6,177 |
| | HC | 2,998 | 9,258 | 2,752 | 9,079 |
| | DIP | 4,595 | 16,613 | 4,374 | 15,905 |
| | inter | 4,743 | 23,294 | 4,544 | 22,587 |
| Human | Y2H | 3,134 | 6,149 | 2,784 | 6,007 |
| | HPRD | 18,149 | 22,344 | 7,348 | 18,831 |
| Worm | DIP | 2,638 | 3,970 | 2,386 | 2,825 |
| | Y2H | 3,328 | 5,444 | 2,898 | 5,240 |

TABLE IV: Various network characteristics. The mean cluster coefficient and standard deviation; the network diameter; and the average distance in the network are listed.

| Organism | Network | $\bar{cc}$ | $sd$ | diam | $L$ |
|---|---|---|---|---|---|
| Yeast | DIP-core | 0.22 | 0.31 | 13 | 5.1 |
| | HC | 0.29 | 0.34 | 15 | 4.9 |
| | DIP | 0.14 | 0.26 | 11 | 4.5 |
| | inter | 0.19 | 0.31 | 11 | 4.4 |
| Human | Y2H | 0.07 | 0.19 | 13 | 4.8 |
| | HPRD | 0.09 | 0.21 | 15 | 5.2 |
| Worm | DIP | 0.02 | 0.12 | 14 | 4.8 |
| | Y2H | 0.07 | 0.21 | 13 | 4.9 |

TABLE V: Network degrees. The mean degree $\bar{d}$, quantiles $(Q_1, Q_2, Q_3)$, the degree threshold for hub proteins $d^*$, and the maximum degree $\Delta$ are listed.

| Organism | Network | $\bar{d}$ | $Q_1$ | $Q_2$ | $Q_3$ | $d^*$ | $\Delta$ |
|---|---|---|---|---|---|---|---|
| Yeast | DIP-core | 5.1 | 1 | 3 | 6 | 17 | 61 |
| | HC | 6.6 | 2 | 4 | 8 | 22 | 206 |
| | DIP | 7.3 | 1 | 3 | 7 | 30 | 287 |
| | inter | 9.9 | 1 | 3 | 8 | 56 | 294 |
| Human | Y2H | 4.3 | 1 | 2 | 5 | 14 | 129 |
| | HPRD | 4.9 | 1 | 2 | 5 | 17 | 181 |
| Worm | DIP | 3.2 | 1 | 1 | 3 | 10 | 187 |
| | Y2H | 3.6 | 1 | 2 | 4 | 12 | 187 |

The degree distributions of proteomic networks have been studied earlier, and shown to fit a power-law or modified power-law. Since the HC network has significantly different properties from these earlier networks, we tried to fit its degree distribution with a power-law and a modified power-law. The latter gave a better fit, which is shown in the log-log plot in Fig. 9. The model we used is $f(k) = Ak^{\beta}\exp(\gamma k)$, where $f(k)$ is the number

of vertices with degree $k$, and $\beta$ and $\gamma$ are constants. We fit other networks studied in this chapter to a power-law model (which does not include the exponential function in the distribution. Only three of these networks could be fit well using the power-law model, and the other five networks required the modified power-law, as shown in Table VI. Note that the exponential function is decaying in the high confidence network, while the corresponding terms grow in the other modified power-law networks, indicating a fat-tailed distribution. Again, these results indicate the variability in the current interaction network data.

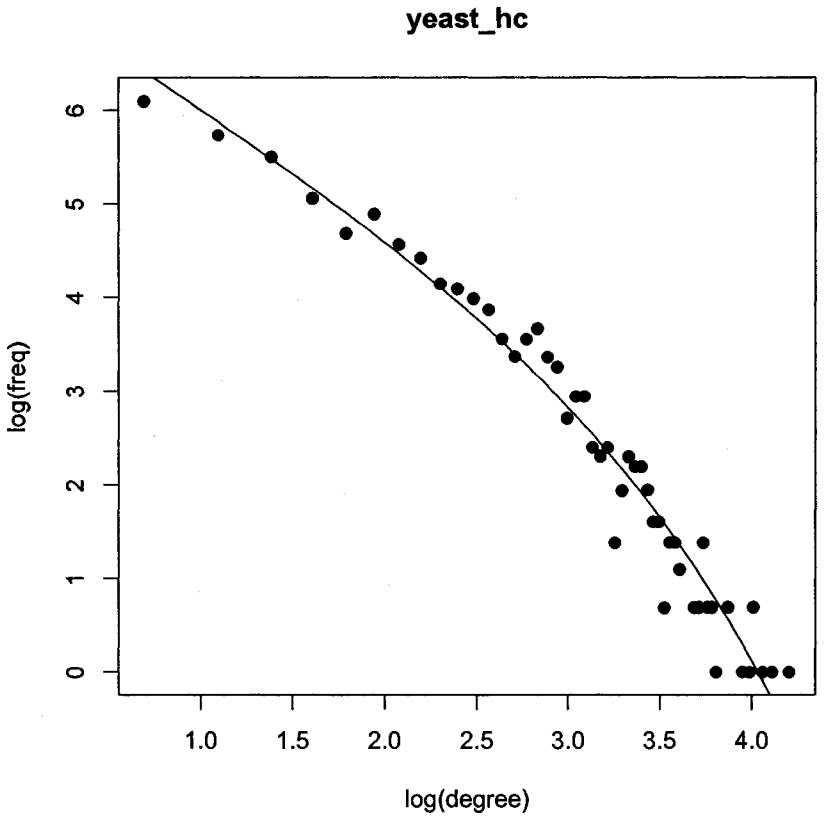



FIG. 9: The degree distribution of the Yeast HC interaction network. The solid line is the curve fitted using a modified power-law.

TABLE VI: Parameters from least-squares fit for power-law and modified power-law networks. Model 1 is $f(k) = Ak^{-\beta}$, and Model 2 is $f(k) = Ak^{-\beta}\exp(\gamma k)$. Here $\alpha = \ln A$.

| Name | M* | $\alpha$ | $\beta$ | $\gamma$ | $R^2$ |
|---|---|---|---|---|---|
| Yeast-HC | 2 | 7.33 | 1.22 | −0.043 | 0.96 |
| Yeast-Inter | 2 | 6.53 | 1.67 | 0.013 | 0.91 |
| Human-Y2H | 2 | 8.92 | 2.54 | 0.026 | 0.96 |
| worm-DIP | 2 | 7.65 | 2.29 | 0.025 | 0.95 |
| worm-Y2H | 2 | 8.21 | 2.35 | 0.023 | 0.96 |
| yeast-DIP-core | 1 | 8.46 | 2.18 | | 0.95 |
| yeast-DIP | 1 | 8.52 | 1.81 | | 0.92 |
| Human-HPRD | 1 | 9.34 | 2.09 | | 0.96 |

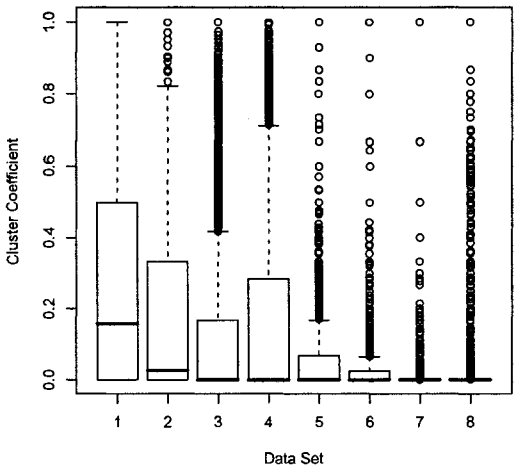



FIG. 10: The box plot distribution of the cluster coefficients of the eight networks. These networks are numbered in the order listed in Table III. The bottom edge of the box is the first quartile ($Q_1$), the horizontal line is the median, and the top edge shows third quartile ($Q_3$). The horizontal line at the top of the dashed line connected to the box shows the smaller of the maximum value or 1.5 times the interquartile range ($Q_3 - Q_1$). The circles denote outliers, the values greater than the multiple of the interquartile range.

The HC yeast network is qualitatively different from the other three. This is most clearly seen from the distribution of the cluster coefficients in these networks (Fig. 10). The proteins in the HC network have a significantly higher median cluster coefficient, and the distribution is concentrated about the median. There are no outliers in the HC network since the maximum lies within 1.5 times the interquartile range. These structural characteristics are consistent with the "stratus, not altocumulus" view of the interaction network, proposed by the authors of [61]. These authors argue that many proteins belong to multiple modules, and that the modules have significant overlap and share subsets of proteins, whereas the conventional view of the network is as a collection of modules that interact with each other primarily through the hubs.

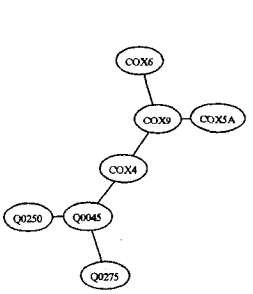
### V.4.2 Expression Data

We used two different datasets to compute expression profile around bridge protein: a cell cycle experiment [68], and the Rosetta compendium expression data set [69]. The two datasets provide a genome-wide expression ratio for 300 states.

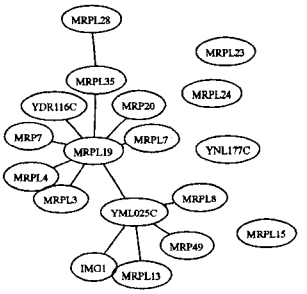
### V.4.3 Reference Data Sets

High quality data collections are needed as gold standards to validate clustering approaches. For this purpose, we have used collections of protein complexes in the yeast that have been culled from the literature and catalogued in the MIPS yeast genome database [70]. We also assess the functional coherence of the conserved clusters based on the Gene Ontology (GO) [71]. The Gene ontology (GO) is a collaborative public database for representing the functional information of genes and gene products (proteins). GO has a large set of controlled vocabularies (biological or biochemical terms), describing gene or gene products (proteins) based upon three distinct ontology hierarchies: biological process, molecular function, and cellular component. Each hierarchy is represented as a directed acyclic graph (DAG), consisting of directed edges (relation) and vertices (terms), such that each vertex may be descended from several others. Annotation of a gene with a descendant attribute implies that the gene has all of its ancestor attributes. We have found the cellular component ontology the most comprehensive among the three, and use it as the primary gold standard to compare the clusters obtained from the interactions data. The components include protein complexes, membranes, and envelopes, which are doubly layered membranes.

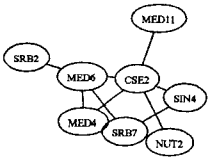
We now consider several of the properties of the GO cellular components. The subgraphs of the yeast-HC protein interaction network induced by some of the GO components are shown in Fig. 11. Among these components are complexes and organelles such as the ribosomal subunit. It is clear from this illustration that protein complexes have topologies that vary from near cliques to much sparser subgraphs with few interactions. Thus any algorithm that relies on identifying dense subgraphs as complexes must necessarily miss identifying many complexes. Furthermore, note that the subgraph induced by the mitochondrial large ribosomal subunit is not a single connected component; there are other such GO components as well. These show that either some interaction data are missing in the current interactome, or that the connected subnetworks of the interactome are not capable of representing every protein complex. These observations should remind us of the limitations of topology based approaches to identifying protein complexes.



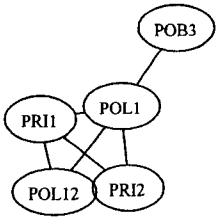

(a) Mitochondrial respiratory chain complex



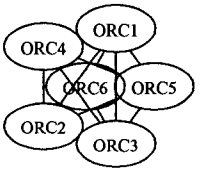

(b) Mitochondrial large ribosomal subunit



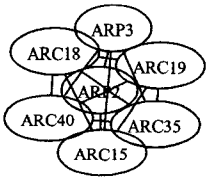

(c) Mediator complex



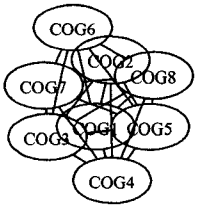

(d) $\alpha$−DNA polymerase-primase complex




(e) Nuclear origin of replication recognition complex



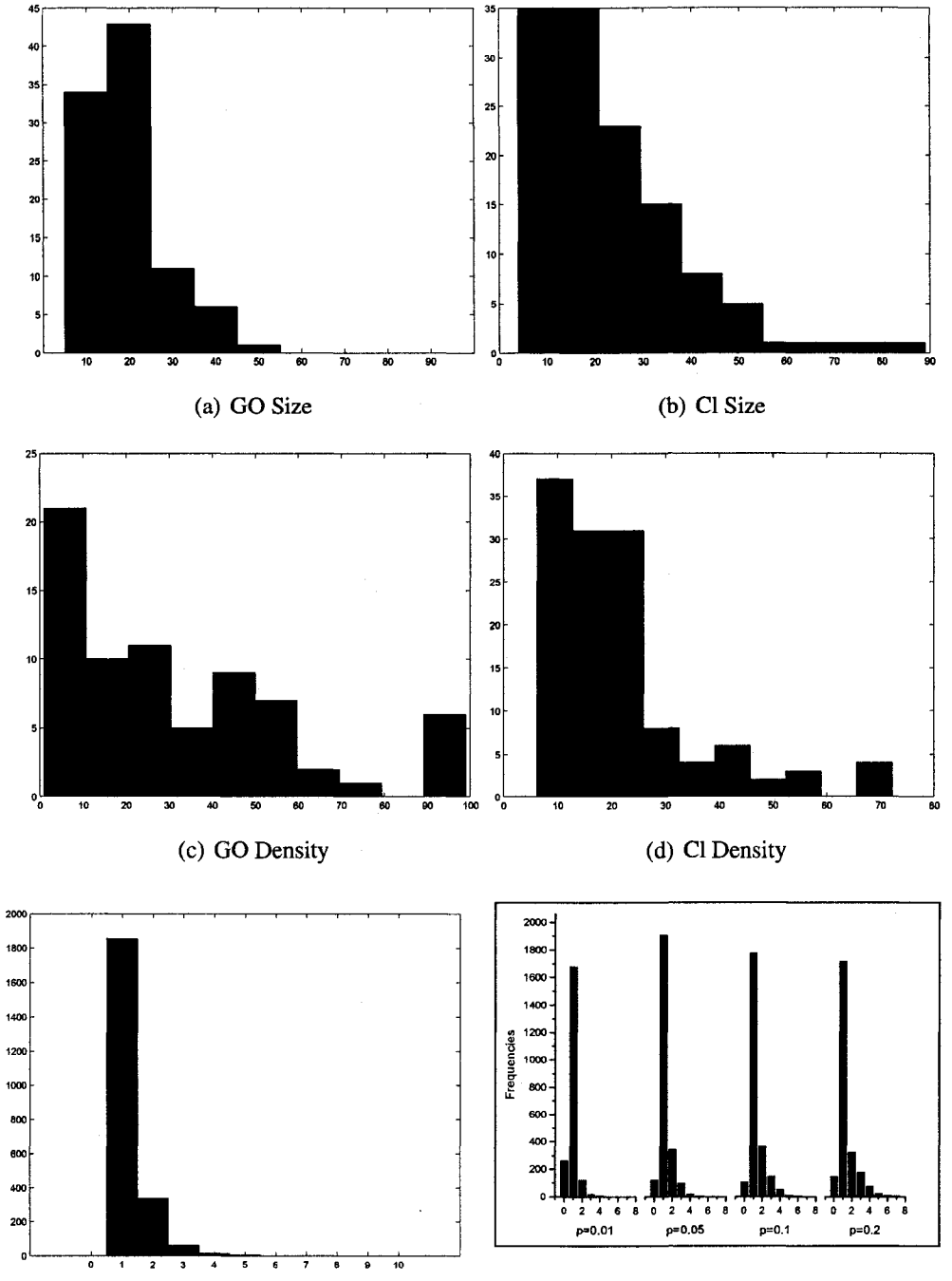

(f) Arp2-3 protein complex




(g) Golgi transport complex

FIG. 11: The subnetworks of the Yeast-HC network induced by some of the GO components. Interactions are represented either by a line joining two proteins, or by the circles representing the two proteins touching each other.

We plot the distributions of the sizes, the densities, and the overlaps among the GO components induced by the yeast-HC network in the left half of Fig. 12. The size of an induced GO component is the number of proteins in it that belong to the yeast-HC network, the density of a cluster is the ratio of the number of interactions among these proteins to the number of interactions if each protein were to interact with all others, and the overlaps count the number of proteins that belong to multiple components. The right halves of these figures show the results we have obtained from our clustering algorithm, and these will be discussed in the next section.




(a) GO Size (b) Cl Size

(c) GO Density (d) Cl Density

(e) GO Overlap (f) Cl Overlap

FIG. 12: The induced GO Components: size, density and overlap distributions. (a) The induced GO components size distribution, (b) The cluster size distribution based on a $p-$value of .05; (c, d) The distribution of the densities of the induced GO component and clusters; (e) The induced GO components overlap histogram; and (f) The cluster overlap histogram based on different threshold $p-$values.

## V.5 RESULTS OF CLUSTERING QUALITY

We applied our methods to search for overlapping modules in the protein interaction networks of yeast, human and the worm. The results of our clustering approach are tabulated in Table VII.

TABLE VII: Clusterings of different networks. Here, $|\mathcal{C}|$ is the number of clusters, $\bar{C}$ is the average cluster size, $|B|$ is the number of bridge proteins joining different clusters, and $D$ is the number of proteins not clustered by the algorithm (expressed as a percentage of the number of proteins in the network).

| Organism | Network | $\lvert\mathcal{C}\rvert$ | $\bar{C}$ | $\lvert B\rvert$ | $D$ |
|---|---|---|---|---|---|
| Yeast | DIP-core | 95 | 25.5 | 69 | 5.5 |
| | HC | 125 | 23.4 | 111 | 4.2 |
| | DIP | 164 | 30.4 | 278 | 4.2 |
| | Inter | 175 | 30.1 | 295 | 3.7 |
| Human | Y2H | 108 | 28.9 | 84 | 5.4 |
| | HPRD | 212 | 42.8 | 500 | 5.1 |
| Worm | DIP | 56 | 37.5 | 10 | 15.0 |
| | Y2H | 107 | 28.6 | 43 | 6.0 |

For each organism, the networks are listed by increasing number of interactions. In yeast, the DIP-core and HC networks are comparable in size, while the DIP and Inter networks are larger by a factor of two. Note that the average cluster sizes are similar for the first pair of networks, and that these sizes are similar for the last two networks also. In the human, the HPRD network has roughly four times as many interactions as the Y2H network. In general, for the yeast and the human networks, increasing number of interactions leads to increases in the cluster size, the number of clusters, and the number of bridges. Note that our algorithm discards only a small fraction of the proteins, except for the worm DIP network. The topologies of these clusters are illustrated in Fig. 13. It should be observed that our clustering approach is capable of identifying densely connected subgraphs as well as sparsely connected subgraphs as clusters.

### V.5.1 Comparing Clusters with the Gene Ontology

We assess the biological significance of the clusters in the yeast-HC network by comparing them with components in the Gene Ontology. Fig. 14 shows a histogram of the overlap scores between clusters of the yeast-HC network and the GO components induced by these proteins. The overlap score of a cluster with a GO component is defined as

$$s = \frac{i}{\sqrt{a * b}},$$

where $i$ is the number of proteins common to both the cluster and the induced GO component, $a$ is the number of proteins in the cluster, and $b$ is the number of proteins in the induced GO component. The GO components have been filtered using the methods of mutual information to remove nearly identical components (if included, they would increase the overlap score further). The figure shows that of the 125 clusters, more than half have overlap scores of 0.5 or larger. Moreover, as Fig. 13 shows, clusters have topologies that range from dense subgraphs (cliques or near-cliques), to sparse subgraphs (stars and star-like topologies),

Table VIII tabulates some of the clusters of the yeast-HC network that have overlap scores higher than 0.6. Each cluster is listed by its ID used in this study and the number of proteins in it is shown. Of the cluster proteins, the number of proteins that belongs to a GO component that has the highest overlap with it, and the number of cluster proteins that do not belong to GO at all. Both these numbers are expressed as percentages. These two percentages add to 100 for most clusters, showing that these clusters in the yeast-HC network overlap well with the corresponding GO components. Proteins in a GO component are not found in the cluster mostly when the proteins are not present in the HC network. This Table shows clearly that topology based methods are capable of identifying functional modules (complexes and organelles) from protein interaction networks.

In Fig. 12, we compare the clusters in the HC network with components in the GO using various distributions. The top figures show histograms of the cluster sizes, the middle figures compare their densities, and the figures at the bottom show the number of proteins that overlap among the clusters and the GO components. The decision whether a protein has a statistically significant number of interactions with a cluster (to decide on the number of clusters a protein could belong to) is made based on a $p$-value. The bottom right figure shows that using a $p$-value of 0.05 in the cluster overlaps leads to the best fit with the overlap data among the GO components.

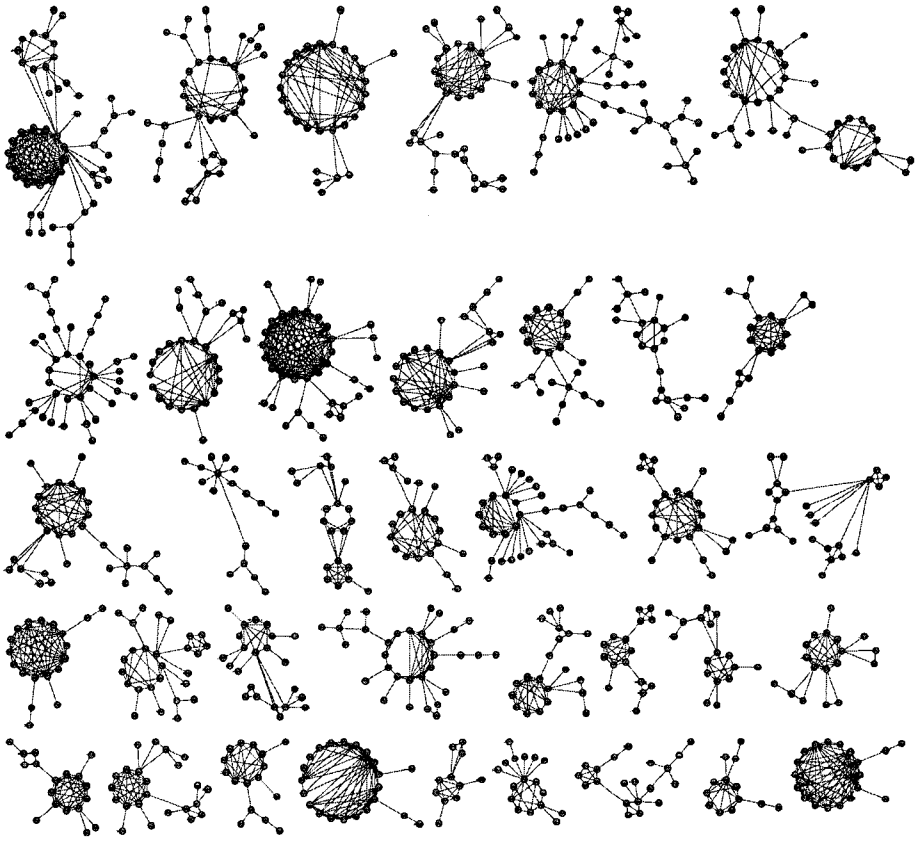


FIG. 13: Some of the clusters in the yeast-HC protein network. Note that the clusters have topologies ranging from nearly completely connected subgraphs (near-cliques) to sparsely connected subgraphs such as stars.

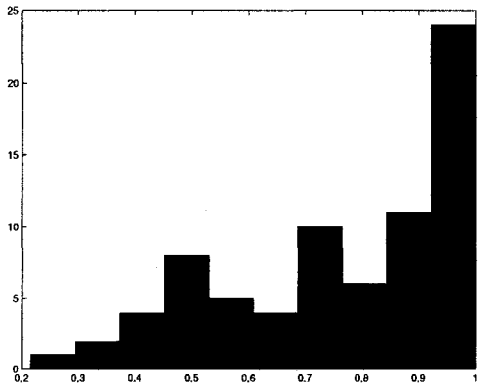



FIG. 14: Histogram of overlap scores between clusters in the yeast-HC network and GO components filtered using mutual information methods.

TABLE VIII: A few of the clusters with the highest overlap scores with GO components. The GO component that has the highest overlap with these clusters is listed, where C. is an abbreviation for complex. The number of proteins in the cluster that overlap with the GO component, and the number of proteins in the cluster not in the GO component are listed as percentages of the number of proteins in the cluster. Overlap scores defined in the text and $p$-values are also shown.

| Cl. Id | Cl. Size | In GO (%) | Not in GO (%) | Overlap score | $-\log$ (p) | GO Id | GO Term |
|---|---|---|---|---|---|---|---|
| 11 | 16 | 87.5 | 12.5 | 0.8 | 21.0 | 0005681 | Spliceosome C. |
| 17 | 12 | 66.7 | 33.3 | 0.8 | 20.4 | 0017119 | Golgi transport C. |
| 23 | 8 | 75.0 | 25.0 | 0.9 | 29.4 | 0000136 | Mannosyltransferase C. |
| 24 | 8 | 50.0 | 50.0 | 0.7 | 12.1 | 0031207 | Sec62/Sec63 C. |
| 28 | 11 | 54.6 | 45.5 | 0.7 | 21.8 | 0030897 | HOPS C. |
| 60 | 10 | 40.0 | 60.0 | 0.6 | 22.6 | 0005946 | $\alpha$,$\alpha$−trehalose-phosphate synthase C. (UDP-forming) |
| 62 | 38 | 68.4 | 29.0 | 0.7 | 28.1 | 0005762 | Mitochondrial large ribosomal subunit |
| 72 | 21 | 38.1 | 61.9 | 0.6 | 25.2 | 0005663 | DNA replication factor C C. |
| 93 | 10 | 40.0 | 60.0 | 0.6 | 23.6 | 0000214 | tRNA-intron endonuclease C. |
| 95 | 27 | 37.0 | 63.0 | 0.6 | 26.3 | 0030008 | TRAPP C. |
| 104 | 9 | 55.6 | 44.4 | 0.8 | 22.4 | 0030904 | Retromer C. |
| 106 | 20 | 50.0 | 50.0 | 0.7 | 21.1 | 0042729 | DASH C. |
| 109 | 44 | 34.1 | 65.9 | 0.6 | 29.0 | 0005847 | mRNA cleavage polyadenylation specificity factor C. |
| 111 | 36 | 44.4 | 55.6 | 0.7 | 23.9 | 0005666 | DNA-directed RNA polymerase III C. |
| 112 | 26 | 46.2 | 53.9 | 0.4 | 22.4 | 0000778 | Condensed nuclear chromosome kinetochore |
| 117 | 20 | 40.0 | 60.0 | 0.6 | 23.1 | 0048188 | COMPASS C. |
| 119 | 20 | 70.0 | 30.0 | 0.8 | 27.4 | 0005680 | Anaphase-promoting C. |
| 120 | 29 | 37.9 | 62.1 | 0.6 | 23.4 | 0043189 | H4/H2A histone acetyltransferase C. |

### V.5.2 Bridges

We analyzed the group of bridge proteins in the Yeast HC network clustering, and have discovered several important properties of bridges as discussed below.

We found that bridges are not correlated well with hubs (unlike bottlenecks, proteins with high betweenness values): the Pearson correlation coefficients are 0.32 between hubs and bridges, and 0.75 between hubs and bottlenecks. Fig. 15 studies how likely it is for bridge proteins to be essential. The histograms show that hubs are likely to be essential, a finding that is well known. But these results show that hubs that are also bridges are even more likely to be essential than hubs that are not bridges. Also, nonhubs that are bridges are more likely to be essential than nonhubs that are not bridges. Thus, being a bridge increases the probability of a protein being essential. Indeed, 36% of the bridge proteins were found to be essential with binomial $p-$value less than .0015, a highly significant percentage given that the percentage of essential genes in the non-hub-non-bridges subnetwork is 21%.

Furthermore, we studied the relation between bridges and proteins participating in several biological process related to cell organization. A biological process is defined in Gene Ontology as "phenomenon marked by changes that lead to a particular result, mediated by one or more gene products" Table IX shows that bridge proteins tend to be involved in organelle organization and biogenesis, physiological regulation, localization, and RNA metabolism. The percentages in a column add to more than 100 since proteins tend to be included in more than one biological process. We observed that many bridges (59% of them) tend to be structural proteins involved in cell organization; there is a high correlation between hub-bridges and cell organization, as many as (75%) of hub-bridges tends to be a member of such process.

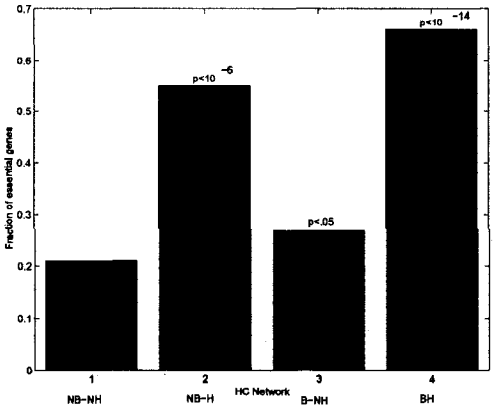


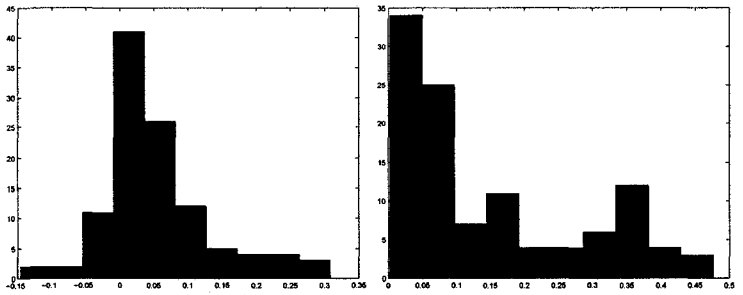

FIG. 15: Essentiality of the four categories of proteins within HC network. HB: Hub-Bridge, HNB: Hub-NonBridge, NHB: nonHub-Bridge, NHNB: nonHub-nonBridge.

TABLE IX: Biological processes that different categories of bridge proteins belong to. Organelle organization is a subcategory of cell organization.

| Process | B-H (%) | B-NH (%) | B (%) |
|---|---|---|---|
| cell organization and biogenesis | 75 | 54 | 58 |
| –organelle organization and biogenesis | 54 | 32 | 37 |
| regulation of physiological process | 33 | 36 | 35 |
| establishment of localization | 30 | 26 | 29 |
| RNA metabolism | 42 | 30 | 26 |

Since bridge proteins interact with proteins belonging to several functional modules, we would expect the expression profile of a bridge protein to differ significantly from its neighbors in a protein interaction network. We tested this hypothesis with normalized micro-array gene expression data [69, 68]. We match each protein with the gene that encodes it, and calculate the average Pearson's correlation of the expression profile for each bridge protein with its neighbors.

Fig. 16 shows comparisons of the homogeneity of the expression profiles of bridges and hubs. The two figures show the correlations of a bridge protein or a hub protein with its neighbors. As expected, the the correlation of protein expression for a bridge with its neighborhood is small, and it is also correspondingly smaller than that of hubs.



(a) Bridges (b) Hubs

FIG. 16: Histogram of the expression profiles of bridges and hubs with their neighbors.

The hubs in the yeast-HC network form a hub subnetwork shown in Fig. 17. The interactions of the hubs with the nonhubs are not shown here. The essential hubs are indicated as solid circles, and non-essential hubs are drawn with open circles. The hubs form six clusters as shown, and the essential hubs are more likely to tend to interact with each other than with non-essential hubs.

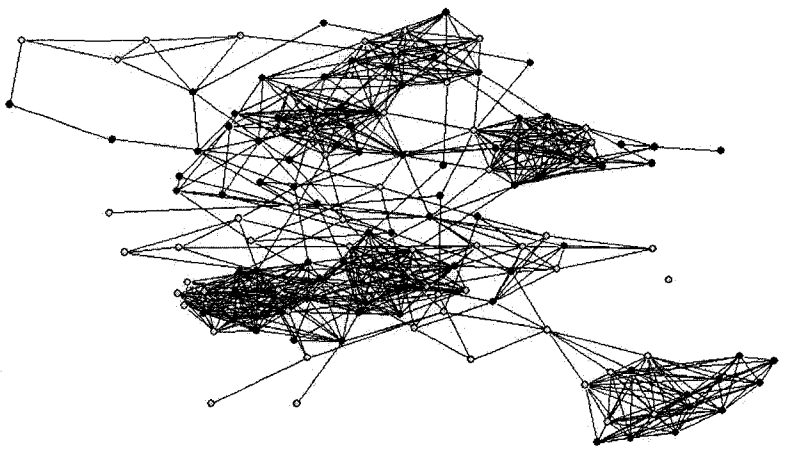


FIG. 17: The essentiality pattern within the hub subnetwork

## V.6 COMPARISONS WITH OTHER APPROACHES AND NETWORKS

In this section, we present a brief review of several methods introduced in Sec. V.3.1, which including two exclusive approaches and two overlapping approaches.

### V.6.1 Graph Growing Algorithm

In the graph growing approach, a cluster grows around a seed vertex using a graph search algorithm. Bader and Hogue [36] used a graph growing technique called *MCODE* in order to detect densely connected regions in large protein-protein interaction networks that may represent molecular complexes. This approach first computes a weight $W_v$ for each vertex $v$ in the graph based on the core value of the vertex and the density of the vertex neighborhood subgraph; then, starting with a seed vertex $s$, it recursively moves outward from the seed vertex, including vertices in the complex whose weight is above a given threshold. If a vertex is included, its neighbors are recursively checked in the same manner to see if they are part of the complex. This process stops when no more vertices can be added to the complex based on the given threshold and is repeated for the next highest unclustered weighted vertex in the network. In this way, the densest regions of the network are identified.

The major drawback of the MCODE algorithm is its sensitivity to the choice of the seed vertices. To overcome this, the algorithm can be executed several times with different seeds, and choose the best clusterings result.

The complexity of MCODE algorithm is $O(n\Delta^2)$, where $n$ is the total number of vertices within the graph. This comes from the vertex-weighting procedure. The computing of the density of the maximum core of the neighborhood of a vertex takes $O(\Delta^2)$ time and there are $n$ such vertices. Finding a k-core in a graph can be done in $O(m)$, where $m$ is the total number of edges within the graph. The complex prediction stage is basically a depth first search in a graph, and hence the complexity is O(m).

### V.6.2 Markov Clustering Algorithm

Van Dong [72] introduced a markov clustering algorithm (MCL) for graph clustering. The MCL algorithm (Algorithm 5) starts by creating the stochastic (markov) matrix $T_1$ of the graph using its adjacency matrix $A$ by normalizing each column of $A$ such that the summation of each column is equal to one. The normalization process starts by making each diagonal element of $A$ equal to the degree of the vertex represented by that column.

Then, computing the sum of each column and scaling each element by the sum of its column. This process yields to the initial column stochastic matrix $T_1$ of the input graph. The stochastic matrix $T$ with $T_{ij} = Pr(j|i)$ represent the probability of moving (transition) from node $i$ to node $j$ in one time step. Since the probability of transition from node $i$ to some other nodes must be one, then the summation of each column should be one. The most characteristic of the stochastic matrix is a probability of transitioning from node $i$ to node $j$ in two steps. This is given by the $(i,j)^{th}$ element of the square of $T$ : $(T^2)_{ij}$. In general the probability transition of going from any node to another node in a finite Markov chain given by the matrix $T$ in $k$ steps is given by $T^k$.

The MCL algorithm simulates a flow on the graph by calculating successive powers of the associated stochastic matrix $T_1$ of the graph adjacency matrix by the alternation of expansion and inflation operations. Expansion refers to taking the power of a stochastic matrix using the normal matrix product i.e. matrix multiplication. Inflation corresponds to taking the Hadamard power of a matrix (taking powers entry-wise), followed by a scaling step, so that the resulting matrix is also stochastic. The inflation operator transforms a stochastic matrix into another one by raising each element to a positive power $r$ (the inflation factor) and re-normalizing columns to keep the matrix stochastic. The effect of the inflation step will result in larger probabilities in each column are emphasized and smaller ones de-emphasized. The inflation factor $r$ must be greater than one. The greater the inflation factor, the greater the number of clusters. On the other side, the matrix multiplication in the expansion step creates new non-zero elements, i.e., edges. Expansion and inflation have two opposing effects: While expansion flattens the stochastic distributions in the columns and thus causes paths of a random walker to become more evenly spread, inflation contracts them to favored paths. At each round, an inflation step is applied to enhance the contrast between regions of strong or weak flow in the graph. Expansion and inflation are alternated until an equilibrium state is reached. An equilibrium state takes the form of *idempotent matrix*, i.e. a matrix does not change with further expansion step. The process converges towards a a clustering of the graph, with a set of high-flow regions (the clusters) separated by boundaries with no flow. Then, using a threshold criteria, we can obtain our final clustering from the last stochastic matrix on hand.

A single iteration of the MCL algorithm requires $O(n^3)$ time, where $n$ is the number of vertices in the input graph $G$ and $n^3$ factor results from the matrix multiplication operation. When the input graphs are sparse, we can use the sparse matrix multiplication algorithm that requires $O(nm)$ time, where $m$ is the number of edges in the $G$.

**Algorithm 5** MCL Algorithm

**procedure** MCL($G$,$K$,$e$,$r$)
    G: The adjacency matrix.
    K: The maximum number of iterations.
    e: The expansion parameter with default value=2.
    r: The inflation parameter (user defined).
    Normalize $G$ and create the stochastic matrix $T_1$.
    $k = 1$.
    **repeat**
        $T_{2k} = E_e(T_{2k-1})$. ▷ Expansion: $E_e(M) = M^e$
        $T_{2k+1} = \Gamma_r(T_{2k})$. ▷ Inflation: $(\Gamma_r(M))_{pq} = (M_{pq})^r / \sum_{i=1}^{k} (M_{iq})^r$
        $k = k + 1$.
    **until** (($|T_{2k} - T_k| < 10^{-6}$) OR ($k = K$)) ▷ Idempotent Mx
**end procedure**

### V.6.3 Line Graph Algorithm

Pereira-Leal et al. [58] used the MCL clustering algorithm to cluster the line graph $L(G)$ exclusively in order to produce an overlapping clustering of the original graph. The line graph, $L(G)$, is the intersection graph of edges of $G$, where vertices in the $L$ correspond to edges in $G$ and are adjacent if the corresponding edges in $G$ share a vertex as shown by examples in Fig. 18 and Fig. 19 respectively.

The lineG graph algorithm one of the first overlapping clustering approaches used for clustering the protein interaction network. The lineG algorithm starts by creating a network of interactions which is the line graph of the protein interaction graph in which each interaction is condensed into a node that includes the two interacting proteins. Then, these nodes are linked by shared protein content. The MCL algorithm is used to cluster the network of interactions. Finally, the algorithm re-converts the identified clusters from an interaction-interaction graph back to a protein-protein graph for subsequent validation and analysis.

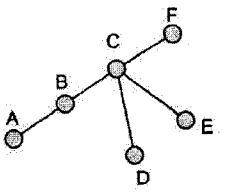



FIG. 18: Schematic representation illustrating a graph representation of protein interactions network.

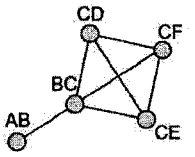



FIG. 19: Schematic representation illustrating the transformation of the protein graph connected by interactions to an interaction graph connected by proteins. Each node represents a binary interaction and edges represent shared proteins.

The complexity of the lineG algorithm is $O(m^3)$ where $m$ is the number of edges in the input graph $G$ and $m^3$ factor is corresponding to the cost of MCL clustering. The line graph clustering algorithm is limited to small networks due to high computational demands.

### V.6.4 Clique Percolation Algorithm

A trivial way of finding an optimal clustering is through an exhaustive search of all possible clusterings. This approach searches for subgraphs with specified connectivity, motifs, and characterizes these motifs as clusters. One such motif is the fully connected subgraph of size $k$ ($k-$clique). Derenyi et. al. [73] introduced a clustering algorithm which interprets as motifs all the $k-$clique percolation clusters in the network. A $k-$clique percolation cluster is a maximal $k-$clique-connected subgraph; i.e., it is the union of all nodes that can be reached via chains of adjacent $k-$cliques.

The authors defined a community, or more precisely, a $k-$clique-community as a union of all $k-$cliques that can be reached from each other through a series of adjacent $k-$cliques (where adjacency means sharing $(k-1)$ nodes). This definition is aimed at representing the fact that it is an essential feature of a community that its members can be reached

through well connected subsets of nodes. In addition, the community definition allows overlaps: (i) a node can be a member of several different communities at the same time, and (ii) communities can overlap with each other by sharing nodes.

The outline of the community finding algorithm:

- The algorithm first extracts all k-clique subgraphs of the network.
- Once the $k-$cliques are located, the clique-clique overlap matrix is prepared. In this symmetric matrix each row (and column) represents a clique and the matrix elements are equal to the number of common nodes between the corresponding two cliques, while each diagonal entry is equal to the size of that clique.
- The $k-$clique-communities for a given value of $k$ are equivalent to such connected clique components in which the neighboring cliques are linked to each other by at least $k-1$ common nodes. These components can be found by erasing every off-diagonal entry smaller than $k-1$ and every diagonal element smaller than $k$ in the matrix, replacing the remaining elements by one, and then carrying out a component analysis of this matrix. The resulting separate components will be equivalent to the different $k-$clique-communities.

In brief, the algorithm systematically lists all possible sets of exactly $k$. For each such set, checks whether all pairs are neighbors in a selected subgraph or not. The basic idea of the algorithm is to use a **k-clique template**, by placing this template onto any k-clique of the original graph, and rolled to an *adjacent $k-$clique*. This can be done by relocating one of its vertices and keeping its other $k-1$ vertices fixed.

By rolling the $k-$clique template through the graph, for each vertex $v$ the algorithm costs $\binom{[N(v)]}{2}$ comparisons to find one clique, where $[N(v)]$ is the set of neighbors of $v$ including $v$. Then the total cost of the algorithm of $O(n\Delta^2)$, where $\Delta$ is the maximum degree in the graph and in the worst case the complexity will be of $O(n^3)$.

This algorithm (could be done faster) if the graph size is small or it is a sparse graph. In general, the exhaustive search approaches are impractical and not scalable for finding larger clusters in large-scale networks.

Adamcsek et al. [59] provided a program, cFinder, for locating and visualizing overlapping, densely interconnected groups of nodes in a given undirected graph.

### V.6.5 Comparison Result

We compare the performance of our exclusive and overlapping clustering algorithms with the methods mentioned in the previous section. We calculate the performance measures which defined in previous section (see Sec. V.2.5). Fig. 20, Fig. 21, and Fig. 22 demonstrate the performance of our algorithms compared with other existing methods. Table X presents a comparison with other existing methods for exclusive and overlapping clustering for the yeast-HC data set, and the cellular components of the gene ontology as the reference data set filtered using mutual information method (see Sec. V.2.4). Also, Table XI demonstrates our approaches outperform the others in term of discard ratio; the discard ratio can be a critical loss of information.

We compare our exclusive algorithm with two algorithms commonly utilized for extracting functional modules from protein interaction networks: MCODE and MCL. A recent study by Brohee and van Helden [74] that compared these algorithms (among others) showed that the MCL algorithm, in particular, was very effective in identifying protein complexes from protein interaction networks. We wish to investigate the benefits of our exclusive clustering when compared to these two algorithms. We used the MCODE and MCL algorithms to extract clusters from the Yeast HC protein network.

In general, MCODE algorithm has a high discard ratio because it searches for very high density modules only. So, it yields a lower number of clusters (62 clusters). Only 28 of them have size greater than 8 proteins (vertices) which cover 20% of the network.

MCODE algorithm yields a high precision value (the computed clusters overlap well with the filtered complexes on GO cellular components); and a low recall value (most filtered complexes formed by the proteins under study does not overlap well with the computed cluster from the proteomic network). One major drawback of this algorithm is that not all the proteins in the network are clustered as illustrated in Table XI. It can be seen that our algorithm outperforms MCODE algorithm by a significant margin in terms of general performances.

In comparison with MCL algorithm, our algorithm has overall similar general performances. Nevertheless, our algorithm exhibits better correspondence with the complexes catalog within the GO cellular components, with higher recall and sensitivity levels than those attained by MCL algorithm.

Due to the overall similarity between the solutions of our exclusive clustering algorithm and MCL algorithm, we conducted a more refined analysis of the differences between the two approaches. Our exclusive clustering algorithm yields a large percentage of clusters

with cluster-size greater than 8 (114 out of the 125 clusters) with an average cluster-size of 23.4 which is comparable to the average complex-size (18.8) within the GO cellular components (Fig. 12 a). On the other hand, MCL produces a large number of clusters and most of the proteins in the clusters are sparsely connected. So, the MCL algorithm generates 223 clusters for the Yeast-HC network. However, on examination, we found that a large percentage of clusters has a cluster-size less than 8 (146 out of the 223) with an average cluster-size of 10.5. Moreover, our exclusive clustering algorithm clustered most of the proteins within the Yeast HC network (Table XI) whereas in the case of MCL, a majority of the proteins (around 51%) were un-clustered.

The key difference between the MCL algorithm and our exclusive clustering algorithm is the way they treat low-shell vertices (vertices not in the 3-core of the network). While MCL algorithm creates separate small clusters for theses vertices which yield large number of clusters with small sizes and consequently a higher discard ratio after all, our exclusive clustering algorithm assigns these vertices to the most relevant cluster based on the majority rule. Thus our exclusive clustering method produces denser clusters with different connectivity patterns which agree with the connectivity patterns within the reference data set.

These results show that MCL algorithm produces many small-sized clusters which are not as homogeneous (with the complexes within the reference data) as the clusters obtained by our exclusive clustering algorithm.

In general, our exclusive clustering algorithm has similar general performance values as MCL, but with a lower discard ratio since the MCL produced numerous small–sized modules, it discard smaller amount of nodes than MCODE (Table XI). Also, our exclusive clustering algorithm shows relatively high recall and sensitivity values which represent a higher complexes coverage (most filtered complexes formed by the proteins under study overlap well with the computed cluster from the proteomic network) than the other exclusive clustering algorithms.

The previous paragraphs discuss the performance measures of the exclusive clustering methods. Here we discuss the performance measures for the overlapping clustering methods. Clique Finder (cFinder) algorithm for overlapping clustering has similar performance measures with MCODE algorithm for exclusive clustering since it search for cliques. Moreover, it has the highest discard ratio (Table XI). Despite of the high discard ratio, cFinder algorithm yields high precision and separation values. Separation measure

is particularly relevant to assessing overlapping clustering algorithms, the higher the separation values the better the overlapping performance within the overlapping clustering. Another overlapping clustering method (lineG) which is applying the MCL method on the line graph of the input graph. This approach may seem hopelessly expensive for large graphs, since the complexity of the lineG algorithm is $O(m^3)$ where $m$ is the number of edges in the input graph. lineG algorithm is characterized by a high sensitivity and a low separation. Our overlapping algorithm has high precision (the computed clusters overlap well with the filtered complexes on GO cellular components); and high recall (most filtered complexes formed by the proteins under study overlap well with some computed cluster from the proteomic network). Also our algorithm has a high separation value with low discard ratio $d$ (i.e., clusters most proteins in the network). Also, our overlapping clustering algorithm outperforms the other algorithm in terms of accuracy and sensitivity.

Finally, our exclusive and overlapping clustering algorithms have much better F-measure values, and outperform in $F(1-d)$, $\mathrm{Acc}(1-d)$, and $\mathrm{Sep}(1-d)$ measures (Table X).

Our clustering algorithms were found to be more accurate and consistent than existing methods. Furthermore, the overlapping algorithm gave superior results to the other overlapping algorithms tested. In addition, our clustering algorithms are scalable; i.e., they are capable of clustering large networks (such as the human protein interaction network), and can be tuned using parameters to obtain clusterings with a desired number and average size of clusters. Finally, our clustering algorithms work for both reliable sub-networks and unreliable networks; and in multiple organisms. Fig. 23 and Fig. 24 demonstrate that our algorithms outperform the others in terms of $F$-measure and $F(1-d)$-measure, where $d$ is the discard ratio of each algorithm. This is demonstrated using four different Yeast data sets, and the cellular components of the gene ontology as the reference data set.

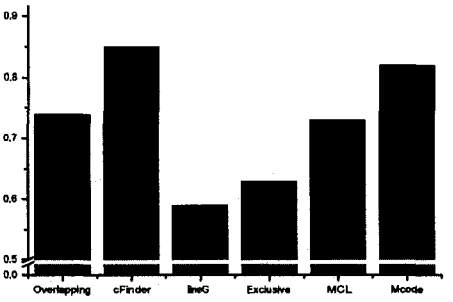
In conclusion, our algorithms outperformed competing approaches and are capable of effectively detecting both dense and sparsely connected biologically relevant functional modules with fewer discards. The incompleteness of clustering is another distinct drawback of existing algorithms, which produce many clusters with small size.

TABLE X: Clustering performances of the algorithms when applied to Yeast HC.

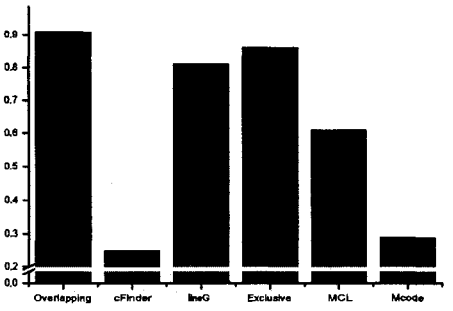
| | Overlapping | cFinder | lineG | Exclusive | MCL | Mcode |
|---|---|---|---|---|---|---|
| $P$ | 0.74 | 0.85 | 0.59 | 0.63 | 0.73 | 0.82 |
| $R$ | 0.91 | 0.25 | 0.81 | 0.86 | 0.61 | 0.29 |
| $F$ | 0.81 | 0.39 | 0.68 | 0.73 | 0.66 | 0.43 |
| $F(1-d)$ | 0.76 | 0.05 | 0.59 | 0.69 | 0.32 | 0.08 |
| PPV | 0.33 | 0.39 | 0.27 | 0.38 | 0.47 | 0.32 |
| Sn | 0.53 | 0.18 | 0.54 | 0.45 | 0.40 | 0.23 |
| Acc | 0.42 | 0.26 | 0.38 | 0.41 | 0.43 | 0.27 |
| $\text{Acc}(1-d)$ | 0.39 | 0.03 | 0.33 | 0.39 | 0.21 | 0.05 |
| $\text{Sep}_{cl}$ | 0.29 | 0.73 | 0.15 | 0.50 | 0.52 | 0.66 |
| $\text{Sep}_{co}$ | 0.19 | 0.11 | 0.15 | 0.21 | 0.23 | 0.11 |
| Sep | 0.23 | 0.29 | 0.15 | 0.32 | 0.35 | 0.27 |
| $\text{Sep}(1-d)$ | 0.22 | 0.04 | 0.13 | 0.31 | 0.17 | 0.05 |

TABLE XI: Discard ratio percentage.

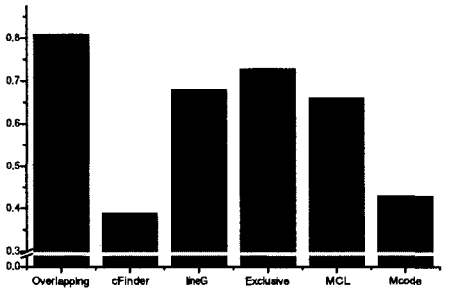
| | Cl-size $>2$ | Cl-size $>4$ | Cl-size $>8$ |
|---|---|---|---|
| Overlapping | 4 | 4 | 6 |
| cFinder | 69 | 78 | 86 |
| lineG | 6 | 8 | 13 |
| Exclusive | 0 | 2 | 4 |
| MCL | 16 | 27 | 51 |
| Mcode | 70 | 72 | 80 |




(a) Precision ($P$)

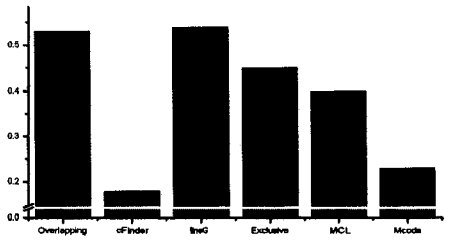



(b) Recall ($R$)

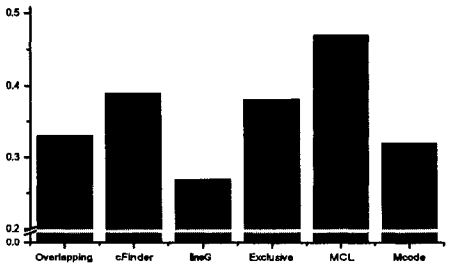



(c) $F-$measure

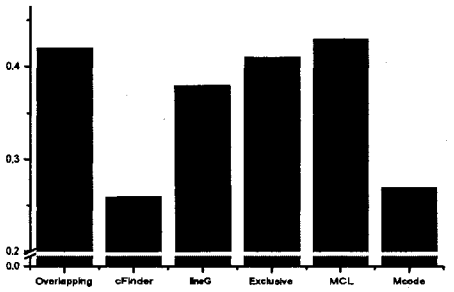
FIG. 20: Clustering performance scores for consensus algorithms which compare clusters in the yeast-HC network to GO components filtered using mutual information methods. Comparisons our overlapping algorithm with cFinder and lineG; and our exclusive algorithm with MCL and MCODE. (a) Precision. (b) Recall. (c) $F-$measure.




(a) Sensitivity (Sn)

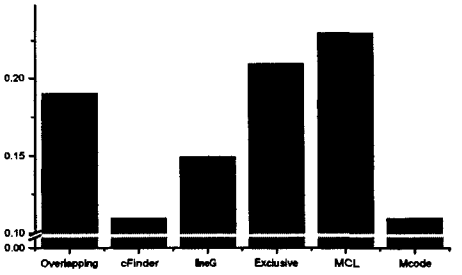



(b) Positive predicted value (PPV)

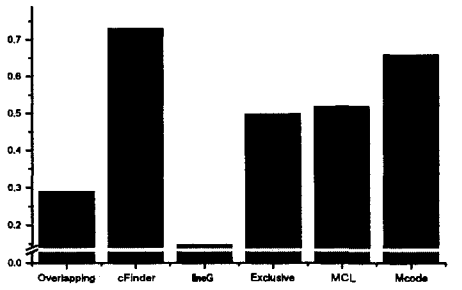



(c) Clustering Accuracy

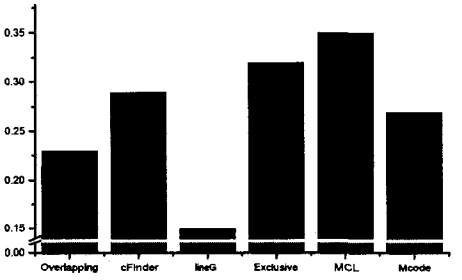
FIG. 21: Clustering accuracy scores for consensus algorithms which compare clusters in the yeast-HC network to GO components filtered using mutual information methods. Comparisons our overlapping algorithm with cFinder and lineG; and our exclusive algorithm with MCL and MCODE. (a) Sensitivity. (b) Positive Predicted Value. (c) Clustering Accuracy.




(a) Complex-wise separation ($Sep_{co}$)




(b) Cluster-wise separation ($Sep_{cl}$)




(c) Clustering Separation

FIG. 22: Clustering separation scores for consensus algorithms which compare clusters in the yeast-HC network to GO components filtered using mutual information methods. Comparisons our overlapping algorithm with cFinder and lineG; and our exclusive algorithm with MCL and MCODE. (a) Complex-wise separation. (b) Cluster-wise separation. (c) Clustering Separation.

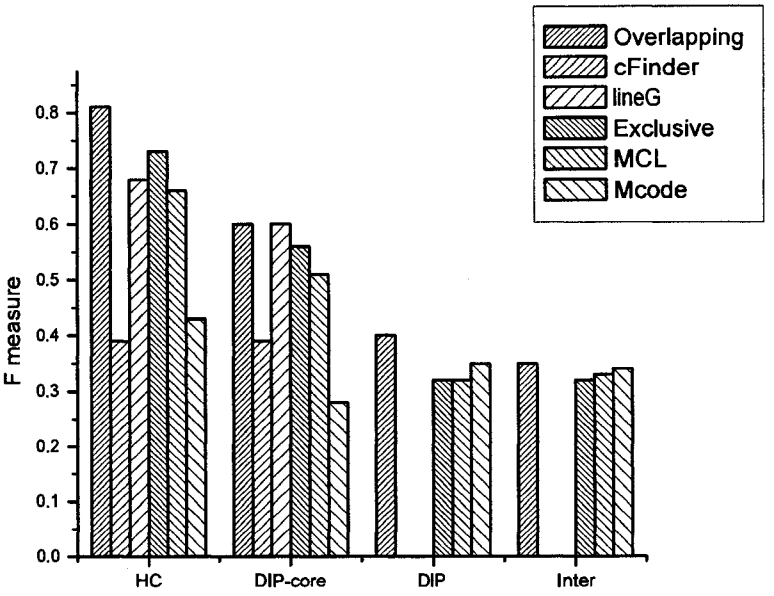



FIG. 23: $F-$measure comparisons for consensus algorithms which compare clusterings from different yeast networks to GO components filtered using mutual information methods.

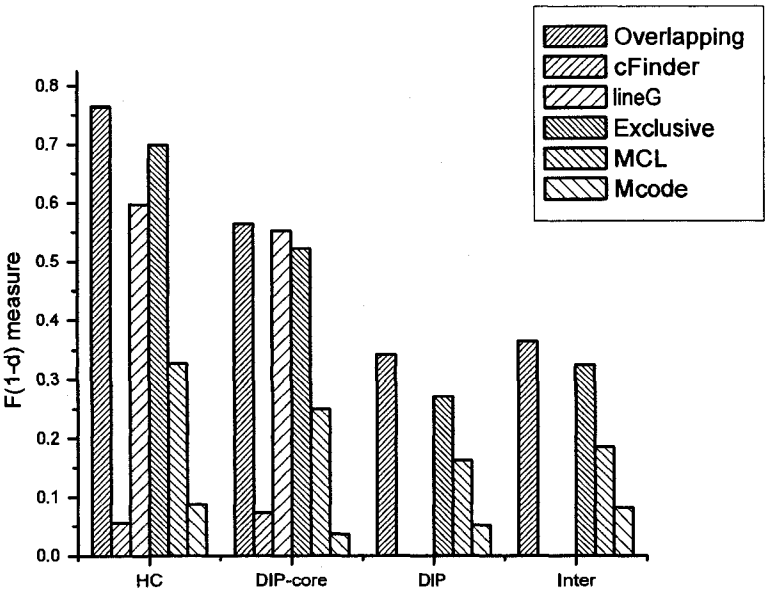



FIG. 24: $F(1-d)$−measure comparison for consensus algorithms which compare clusterings from different yeast networks to GO components filtered using mutual information methods.

## V.7 CONCLUSIONS

Functional modules within the cell may share proteins. This observation indicates that analyzing biological networks, especially protein networks, should favor overlapping clustering approach, wherein some proteins (bridges) are allowed to be members of one or more discovered modules (clusters).

We have proposed a new approach to compute overlapping clustering rather than exclusive clustering approach discussed in the previous chapter. Moreover, we have defined mathematically the concept of a bridge protein (protein connects several modules together), designed a probabilistic algorithm to compute bridges, and used it to identify the organizational set of proteins in the yeast HC protein network.

Our overlapping approach identifies a set of clusters and a set of bridge proteins that form the overlap among the clusters. Furthermore, we use this approach to cluster proteins into specific functional modules as well as to objectively measure each individual protein's value to that functional module.

Our clusterings results show clearly that our clustering methods are capable of identifying functional modules (complexes and organelles) from protein interaction networks that correspond well with multi-protein complexes in the reference datasets. Also, it would be interesting to discover properties that distinguish bridge proteins from the other proteins in the network. So, we analyzed the group of bridge proteins in the Yeast HC network clustering, and have discovered several important properties of bridges a) bridges are not highly correlated with hubs (high degree proteins), b) bridges that are not hubs are more likely to be essential than non-hubs that are not bridges, and c) many bridges tend to be structural proteins involved in organelle organization.

Moreover, we have investigated additional clustering approaches. Our experience comparing the extant approaches with our exclusive and overlapping clustering approaches shows that these methods are less stable with higher discard ratio than ours which optimize a global objective function.

Finally, we have analyzed the suitability of the described overlapping clustering technique for clustering biological networks from different organisms. Our result shows that our clustering algorithms work for both reliable sub networks and unreliable networks; and in multiple organisms.

# CHAPTER VI

# APPLICATIONS

## VI.1 INTRODUCTION

Human T-cell Leukemia Virus type 1(HTLV-1) is the causative agent of Adult T-cell Leukemia (ATL), HTLV-1 Associated Myelopathy/Tropical Spastic Paraparesis (HAM/TSP) as well as other subneoplastic conditions [75, 76, 77, 78, 79]. Although the development of ATL is the culmination of complex events, it appears that the viral oncogene product, Tax, may provide the impetus for the transformation process. This protein has been studied extensively since 1982 when Tax was discovered to be a transactivator of the cognate viral promoter [80]. Since that time many activities and subsequent functions have been assigned to the Tax protein [81, 82, 83]. The critical importance of this protein to human disease makes it a fascinating protein as a research target; however, the result of such focused research efforts has been thousands of articles and a healthy dose of controversy. These qualities also make Tax an ideal candidate for the development of a complete list of interacting proteins as an effort to define potential protein functions.

There have been a number of published accounts of cellular proteins that bind to Tax. For example, Din et al described the binding of Tax to MAD1 as a result of a comprehensive yeast two-hybrid approach [84]. Immunoprecipitation and western analysis has been used to identify specific Tax-protein interactions, for example $IKK_{\gamma}$ [85, 86]. Recently, Kashanchi and co-workers conducted a major effort using 2D gel separation followed by MALDI-MS to identify a 32-member Tax interactome [87]. A combined listing of Tax binding proteins with accompanying literature citations can be found by visiting the publicly accessible Tax website (http://htlv-tax.com).

As data accumulates regarding Tax-protein interactions, a system for analysis and validation of these interactions is needed. This is especially true given the exponential increase in technical ability to identify protein-protein interactions, compounded by the inherent increases in false-positives (protein-protein interactions of no functional consequence). We describe a two-pronged approach for identification and selection of functionally significant Tax-protein interactions. The study begins with the construction of a comprehensive physical interactome using affinity isolation of Tax complexes coupled to MS/MS analysis. Next, we utilized knowledge gained in existing literature that defined a physical interaction

between Tax and a cellular protein, to comprise an *in silico* Tax interactome. This interactome was then restricted to proteins with a putative role in DNA repair response. The final steps expanded the in silico interactions into a nearest neighbor network to identify groups of proteins with greatest functional impact to DNA repair response. Our analysis identified DNA-PK as a top candidate protein for further analysis into the mechanism of action for Tax-induced defects in the cellular DNA damage repair response.

## VI.2 METHODS

### VI.2.1 Cell Culture and Transfection

293T cells were maintained at 37°C in a humidified atmosphere of 5% $CO_2$ in air, in Iscove's modified Dulbecco's medium supplemented with 10% fetal bovine serum and 1% penicillin-streptomycin. Transient transfections were performed by standard calcium phosphate precipitation. Cells were plated in 150−mm plates at $4 \times 10^6$ cells per plate. The following day, 20$\mu$g of plasmid DNA in 2M $CaCl_2$ and 2X HBS were added dropwise to cells in fresh medium. Cells were incubated at 37°C for 5 h and fresh medium was added. The cells were harvested 48 h later

### VI.2.2 Purification of Tax Protein

Tax protein was isolated following a single wash with 1X PBS, in 500$\mu$l M-Per mammalian protein extraction reagent (Pierce, Rockford, IL) supplemented with protease inhibitor cocktail (Roche, Palo Alto, CA) and immediately frozen at −80°C. The cell lysate (2.5 mL) was incubated with 200$\mu$l bed volume of S-protein™ agarose (Novagen, Madison, WI) for 30 min at room temperature as per manufacturer's suggestion. The bound S-Tax protein was then washed 3 times with 1 mL Bind/Wash Buffer (20 mM Tris-HCl pH 7.5, 150 mM NaCl, 0.1% TritonX-100).

### VI.2.3 Isolation of Tax-complexes

Freshly prepared S-Tax-GFP or S-GFP beads were washed 3X in incubation buffer (25 mM HEPES, pH 7.5, 150 mM NaCl, 1% NP-40, 10 mM $MgCl_2$, 41 mM EDTA, 1% glycerol) and placed on ice. A working stock of Jurkat nuclear lysate (Active Motif, Carlsbad CA) was prepared by diluting 25$\mu$g lysate to a total volume of 75$\mu$ L in incubation buffer. The lysate was pre-cleared by adding 30$\mu$L of an S-bead slurry and incubating on ice for 30

minutes with occasional mixing. The pre-clear slurry was spun down at 5000 rpm for 3 minutes and the lysate (70$\mu$L) transferred to a fresh 0.5 ml tube containing 10$\mu$L of the S-Tax protein bound to beads. This slurry was incubated at 4°C for 60 minutes on a shaker. The beads were centrifuged at 5000 rpm for 3 minutes, lysate removed, and beads washed 1X with 250$\mu$L incubation buffer followed by 4 washes with 250$\mu$L ice cold PBS.

### VI.2.4 LC-MS/MS of Protein Complexes

S-Tax or S-GFP beads were washed 3X with ice cold 50 mM ammonium bicarbonate, pH 8 and subsequently resuspended in 50 $\mu$L of 50 mM ammonium bicarbonate, 10% acetonitrile containing 3.12 ng/$\mu$L sequencing grade modified trypsin (Promega Corp.). The digest was incubated for 6 hours at 37°C with occasional mixing, transferred to a 0.2 $\mu$m centrifuge tube filter and spun at 5000 rpm for 3 minutes. The flowthrough was recovered and peptides dried in a speed vac. Digests were resuspended in 20$\mu$L Buffer A (5% Acetonitrile, 0.1% Formic Acid, 0.005% heptafluorobutyric acid) and 10$\mu$l were loaded onto a 12 cm x 0.075 mm fused silica capillary column packed with 5$\mu$M diameter C-18 beads (The Nest Group, Southboro, MA) using a N2 pressure vessel at 1100 psi. Peptides were eluted over 300 minutes, by applying a $0 - 80\%$ linear gradient of Buffer B (95% Acetonitrile, 0.1% Formic Acid, 0.005% HFBA) at a flow rate of 150$\mu$l/min with a pre-column flow splitter resulting in a final flow rate of 200 nl/min directly into the source. A LTQ$^{TM}$ Linear Ion Trap (ThermoFinnigan, San Jose, CA ) was run in an automated collection mode with an instrument method composed of a single segment and 5 data-dependent scan events with a full MS scan followed by 4 MS/MS scans of the highest intensity ions. Normalized collision energy was set at 28%, activation Q was 0.250 with minimum full scan signal intensity at $1 \times 105$ with no minimum $MS^2$ intensity specified. Dynamic exclusion was turned on utilizing a three minute repeat count of 2 with the mass width set at 1.0 m/z. Protein searches were performed with MASCOT version 2.2.0$v$ (Matrix Sciences, London GB) using the SwissProt version 51.3 database. Parent ion mass tolerance was set at 1.5 and MS/MS tolerance 0.5 Da.

### VI.2.5 Western Analysis

Total protein concentrations were determined by Protein Assay (Bio-Rad, Hercules, CA). An equal volume of sample loading buffer (Bio-Rad, Hercules, CA) with $\beta$-mercaptoethanol was added to the lysate and boiled for 5 min. Samples were normalized to total protein and separated through a 10% SDS-polyacrylamide gel. The proteins

were transferred onto Immobilon-P (Millipore, Billerica, MA) membrane using a Trans-blot SD semi-dry transfer cell (Bio-Rad, Hercules, CA) at 400 mA for 50 min. Following blocking in 5% non-fat milk in PBS/0.1% Tween-20, blots were incubated in primary antibody overnight, followed by $1h$ incubation in secondary horseradish-peroxidase conjugated anti-mouse or anti-rabbit antibody (Bio-Rad, Hercules, CA). Immunoreactivity was detected via Immunstar enhanced chemiluminescence protein detection (Bio-Rad, Hercules, CA).

### VI.2.6 Sources of Data for in silico Analysis

Interaction data were gathered from three types of information sources: manual extraction from Pubmed, laboratory derived physical interactions, and protein interaction databases. In the first database source, the information was extracted by manually searching the Pubmed literature to obtain a list of known Tax binding proteins. The criterion for acceptance in this group was physical verification of binding in the referenced publication. For the second database source, the physical interactions utilized in this study were all derived from the experimental efforts described elsewhere in this article. For the final database source, we queried a human protein interaction database, The Human Protein Reference Database HPRD [88]. The HPRD (http://www.hprd.org) contains interactions of proteins in the human proteome manually extracted from the literature by expert biologists who read, interpret and analyze the published data.

### VI.2.7 Terms and Definitions for in silico Analysis

For our topological studies of interaction networks, we utilized a novel overlapping clustering approach [3] that exposes the modular structure of the network. We define bridges as proteins that belong to multiple clusters (due to statistically significant overlaps among the clusters). We also employed centrality measures of networks known as betweenness and closeness. To define these measures, first we need to define some network concepts. The distance of a protein $v$ from another protein $w$ is the number of edges in a shortest path between them. The diameter of a network is the maximum distance between any pair of vertices. The average path length of a network is the average distance over all pairs of vertices. The closeness centrality measure for a protein $v$ is the reciprocal of the sum of the distances of $v$ to all other proteins in the network.

The dependence of a protein $s$ on a protein $v$ is the sum over all proteins $t$ in the network of the ratio of the number of distinct shortest paths between proteins $s$ and $t$ that

includes $v$ as an intermediate vertex, and the number of distinct shortest paths between $s$ and $t$. The betweenness value of a protein $v$ is the sum of the dependence values of all proteins $s$ on the protein $v$. This is equivalent to the following equation for betweenness.

$$B(v) = \sum_{\substack{s \in V \\ s \neq v}} \sum_{\substack{t \in V \\ t \neq s, \\ t \neq v}} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

Here $V$ is the set of proteins in the network; the numerator in the fraction shows the number of distinct shortest paths joining $s$ and $t$ on which $v$ is an intermediate vertex; the denominator is the number of distinct shortest paths joining $s$ and $t$. Further details on centrality measures are available in [89].

As in earlier work [52], we define *hubs* as all proteins that are ranked in the top 20% with respect to degree in the network (the number of interactions a protein is involved in). Similarly bottlenecks are all the proteins that are ranked in the top 20% of betweenness values. To calculate betweenness values for proteins, we used an algorithm provided by Yu et al. [60].

In the clustering approach to be described next, we use the concept of a *k-core* of a graph. The $k$-core of a graph is obtained by repeatedly deleting all vertices which are joined to the vertices remaining in the graph by fewer than $k$ edges. This procedure begins by deleting all vertices whose degree is less than $k$. The deletion of such vertices could decrease the degrees of the remaining vertices. If some of these vertices have degrees less than $k$, they would be deleted as well. This process is repeated until the subgraph that remains has every vertex with degree at least $k$; this subgraph is the $k$-core of the graph. All the deleted vertices belong to the $(k-1)$-shell. Computing the $k$-core of a graph helps with denoising the interaction network by removing many false positives, and also reduces the initial size of the network to be clustered. The deleted vertices will be added to the clustering obtained in a subsequent step.

### VI.2.8 Identification of Modules

We now summarize the technique we used for clustering the protein interaction networks [3]. The protein interaction network is represented by a graph $G = (V, E)$, with the proteins constituting a set of proteins $V$, and interactions constituting the set of edges $E$. We obtain clusters in the interaction network by identifying a number of subgraphs of $G$ that have a relatively large number of edges joining vertices in each subgraph and fewer edges to vertices outside the subgraph. We permit these clusters to overlap (have some

vertices in common), since proteins have multiple functions and could be involved in more than one biological process.

The details of the clustering algorithm will be described elsewhere, but here we provide an overview. Clusters are obtained by dividing a subgraph at each step into two subgraphs based on the ratio of the number of edges that join vertices in the subgraph to the total number of edges, a measure called the *cohesion* of the subgraph. Given the initial graph $G$, we recursively split it into subgraphs until the value of cohesion of a subgraph is above a threshold value, or the subgraph has number of vertices fewer than a threshold size. We have used a spectral algorithm that uses the components of an eigenvector of the Laplacian matrix of the graph to divide each subgraph into two. Once the eigenvector is computed (its components correspond to the vertices of the graph), those vertices whose component values are below some specified value are included in one subgraph and the others belong to the second subgraph. The choice of the value where the split should be made is based on computing the cohesion.

We have found that the overall clustering approach described above needed to be adapted to protein interaction networks, which are small-world and modified power-law networks. Initially we decompose the vertices of the network into three sets; hubs or high degree vertices (those in the top 20% of the degrees); low-shell vertices (vertices not in the 3-core of the network); and the residual subnetwork, which forms a 3-core of the network from which the hubs have been removed. We call the last subnetwork as the local network. We have found it advantageous to cluster the local and hub subnetworks separately using the spectral clustering method described above. The clusters from both subnetworks are then merged together if a large number of edges join clusters from the two networks. We check to see if nodes that belong to a cluster are significantly connected to other clusters, and if so, they are included in such clusters as well. The statistical significance of the connections is computed using a $p$-value based on the hypergeometric distribution. Finally, the low-shell nodes are added to clusters; each such node could be added to none, one, or more than one cluster, based on whether it has a statistically significant number of connections to the clusters that have been found. If a node belongs to three or more clusters, we call it a bridge node.

## VI.3 RESULTS

### VI.3.1 Assimilation of an Interaction Database for Tax

We conducted a manual literature search for articles with reference to "Tax Interaction". This list of research articles was then limited to those that could be manually confirmed as containing evidence of Tax binding via physical interaction. The manual filtering resulted in a confirmed list of 67 proteins. As we have alluded to earlier, Tax has many putative functions but for this exercise we have limited our analysis to the DNA damage repair response. Thus, we asked which of these known protein interactions has a known function that would potentially impact the cellular DNA repair response process. Our analysis suggested a starting point of four confirmed Tax-binding proteins; Rad51, TOP1, Chk2, and 53BP1.

### VI.3.2 Construction of a Physical Tax Interactome Map

Our approach to defining the physical Tax interactome began with the selective isolation of Tax-containing multi-protein complexes from mammalian cells. The isolation of multi-protein complexes was facilitated by the use of affinity tagged Tax protein. The S-Tax-GFP vector expresses full length TAX protein fused to amino-terminal His6 and S-tags, and carboxyl-terminal GFP protein. A critical property in such a system is the recapitulation of Tax-associated activity in the fusion protein. We have previously demonstrated that the expressed S-Tax fusion protein is fully functional when compared to wild type Tax protein [90, 91]. The S-Tax-GFP vector was transiently transfected into 293T cells, and the expression of GFP used to assess correct cellular localization and to monitor the transfection efficiency. A series of preliminary experiments were conducted in order to titer the best proportions between lysate concentration and the amount of Tax/beads such that the Tax protein concentration does not either overwhelm the binding partners or disappear from the complex. In an effort to increase the binding specificity of Tax associated proteins, we pre-incubated the nuclear lysate with the s-beads as a "pre-clear" step. This resulted in a significant reduction of nonspecific protein hits such as HSP's and common nuclear structural proteins like tubulin and actin. When each of the three experimental runs was analyzed individually and then compared, we observed that 86% of the proteins were present on all three runs. The control experiments with the S-GFP protein alone resulted in a list of approximately 25 proteins consisting mainly of HSP's, actin and tubulin. Only 10% of these proteins were shared with the S-Tax-GFP experiments.

One approach to assigning value to specific protein-protein interactions is by determining the strength of interaction. A comparable evaluation in mass spectrometry would be measurements that imply the relative amounts of a particular protein. Such a value would be directly influenced by strength of binding. Thus, we combined the data, in which the Tax interactome was analyzed as described above, from three separate experimental runs into one data set. Each of the LC-MS/MS runs contained approximately 23,000 scans. The top 5 protein "hits" as determined via multiple measures of confidence are shown in Table XII. This analysis resulted in the identification of 250 unique Tax-binding proteins that are in the process of being orthogonally confirmed.

### VI.3.3 Defining First Neighbor Interactions of the known Tax-binding Proteins

Our starting group of Tax-binding proteins, Rad51, TOP1, Chk2, and 53BP1, known to play a role in the DNA repair response, was referred to as the set C1. We then created a subnetwork consisting of the first neighbor interactions of these four proteins, which we call G1= 1NN (C1). This subnetwork, G1, consists of a set of 50 proteins involved in 112 interactions as shown in Fig. 25. The $G1$ subnetwork has a diameter of 5, and average path length of 2.7, which are consistent with a small-world network.

Several features in the network G1 and other subnetworks of G1 described below, suggest a significant role for DNA-PKcs (PRKDC). The maximum core of $G1$ is 6, and DNA-PKcs is a member of the 5-core; the 5-core is a highly interacting group of 12 proteins (DNA-PKcs, TOP1, PCNA, RPA1, DDX9, CDK4, CDKN1A (p21), CDK5, ADPRT (PARP), XRCC5 (Ku70), XRCC6 (Ku86), NCOA6 (TRBP)), all of which are related to the DNA-repair process. We also note that active DNA-PK consist of the catalytic subunit (DNA-PKcs) and the two regulatory subunits (Ku70 and Ku86) each of which is a member of this highly interactive core. Furthermore, DNA-PKcs ranks eighth in degree (the number of interactions) and in the top 30% in two centrality measures (betweenness and closeness).

TABLE XII: Confidence measures.

| Protein | Unique Peptides | Protein Score | Coverage | emPAI |
|---|---|---|---|---|
| DNA-dependent Protein Kinase | 25 | 1391 | 9% | 0.27 |
| Vimentin | 21 | 1387 | 44% | 7.54 |
| Gamma Interferon-inducible Protein | 19 | 1116 | 24% | 1.7 |
| Poly[ADP-ribose] polymerase | 15 | 1414 | 34% | 1.78 |
| Core Histone Macro-H2A.1 | 7 | 569 | 30% | 1.25 |

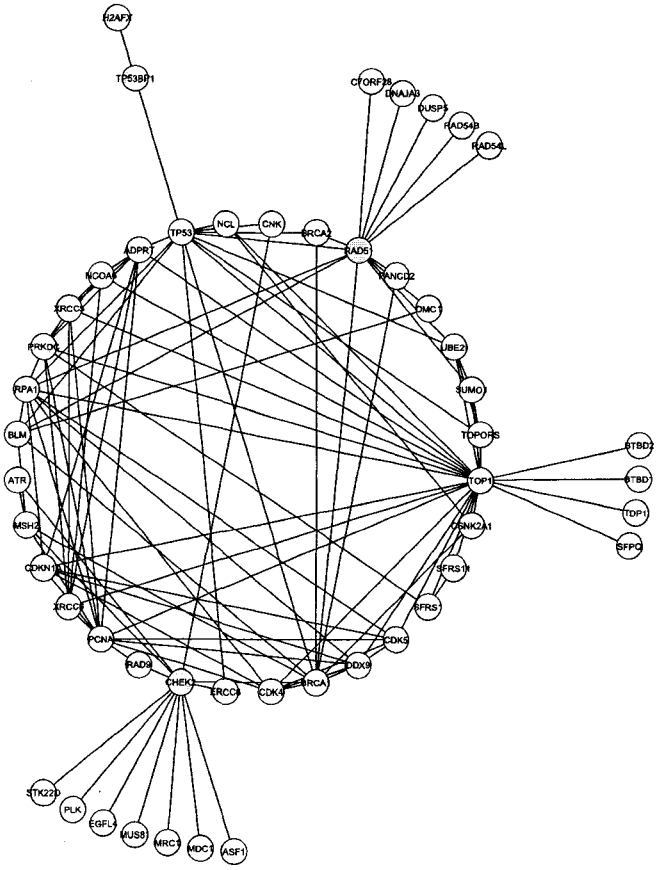



FIG. 25: The first neighborhood network for Rad51, TOP1, Chk2 and 53BP1. The four initial proteins (shaded) were used to generate a network via interrogation of the Human Protein Reference Database. Protein-protein interactions are indicated by lines.

We next considered the structure of the $G1$ subnetwork after the removal of the four initial proteins comprising $C1$. This would allow for an assessment of the degree and centrality of neighbors without interference from the original four proteins. The largest connected component of the resulting network consisted of 29 proteins and 60 interactions as shown in Fig. 26. This network has a diameter of 6 and a small average path length of 2.6. In this subnetwork, DNA-PKcs is among the top six proteins in degree and betweenness centrality.

We then created a subnetwork of G1 restricted to those involved in DNA repair response, referred to as G1*. This network consisted of 26 proteins and 42 interactions as shown in Fig. 27. The G1* network has a diameter of 5 and an average path length of 2.5. In this restricted network, DNA-PKcs ranks fourth in degree and ninth in betweenness centrality. The maximum core of this network is the 4-core, which consists of six proteins of which DNA-PKcs is a member (DNA-PKcs, PCNA, PARP, Ku70, Ku86, and TRBP).

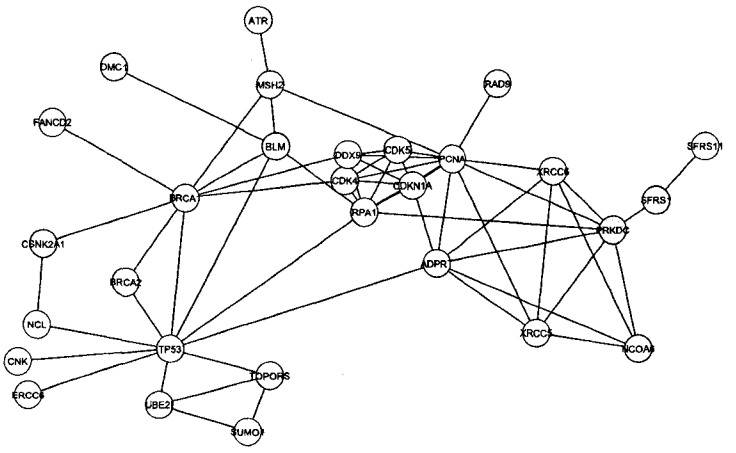



FIG. 26: The largest interacting network remaining after removal of Rad51, TOP1, Chk2 and 53BP1. The components that populated the first neighborhood network were depleted of rad51, top1, chk2 and 53bp1. The remaining components with the highest degree of interaction are shown. DNA-PK is indicated (shaded).

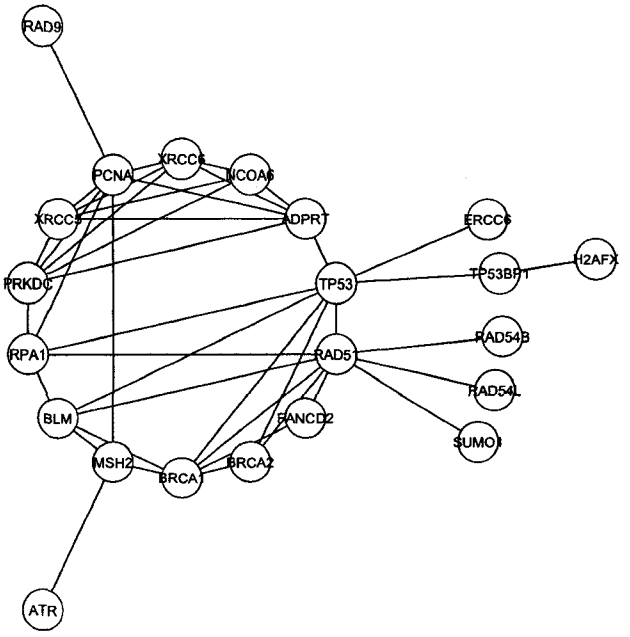



FIG. 27: The first neighborhood network restricted to proteins documented to play a role in the DNA-repair response. The components of the entire first neighborhood network were filtered to remove those not known to have a role in the DNA-repair response. The remaining components are displayed to reveal interactions and a central core.

### VI.3.4 Definition of the Second Neighbors of C1 refined to DNA repair

In our next exercise, we attempt to assign value to the proteins identified in the prior networks by examining their context in the "larger world" of second neighbors. Our assumption was that key proteins from the first neighbor analysis should retain their central role as defined by interactions in the large second neighbor population. Specifically, we considered the first and second neighborhood of the initial set of proteins in $C1$, which we refer to as G2 = 2NN (C1). The G2 network consisted of 667 proteins and 3827 interactions. From the proteins in the G2 network, we created a smaller network by restricting to proteins involved in DNA repair, and refer to this subnetwork as G2*. There were 114 proteins in G2*. We show the 3-core of the G2* network, which consists of 54 proteins, in Fig. 28. All 3-core proteins will have three or more interactions in order to be included in the network. By application of our clustering approach, we expose the structure of this subnetwork. It consists of five clusters of proteins, with the largest cluster having 22 proteins, and the smallest cluster consisting of 3 proteins. Adding proteins of lower degree

clearly generates a larger G2* network, but did not change the integrity of the structure of the network (data not shown). We can also observe from the clustering that three proteins, DNA-PKcs, PCNA, and P53 (TP53) link the various clusters to each other. We call these three proteins "bridges", since they connect the different clusters together. Hence, DNA-PKcs is a bridge protein in this second neighborhood network that links clusters 1, 4, and 5, and is also linked to the bridge protein PCNA. The biological significance of this "cross-talk" between DNA-PK and PCNA is supported by existing literature.

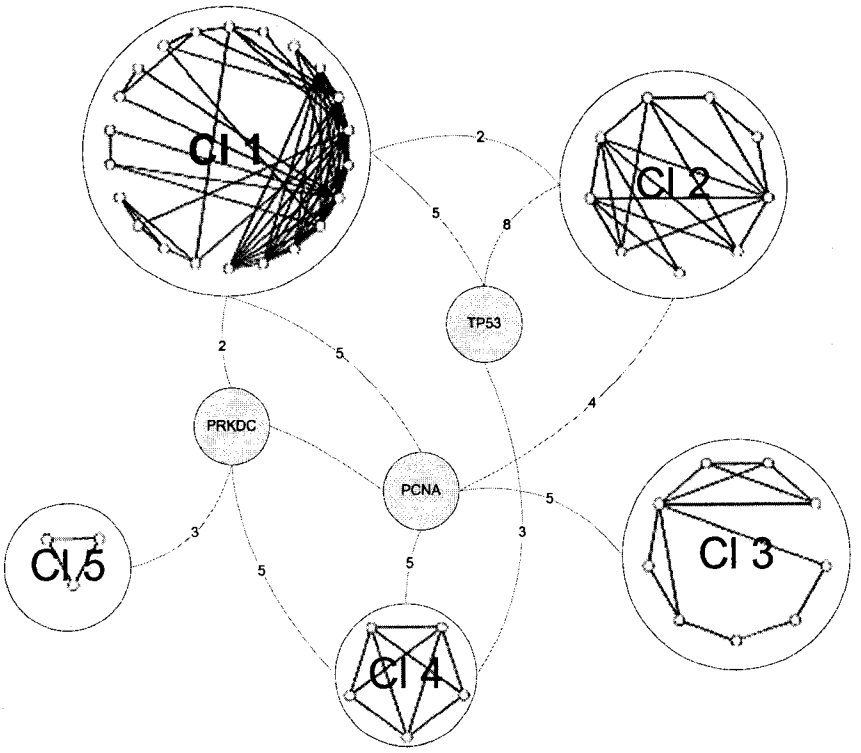



FIG. 28: The 3-core representation of the second neighborhood network restricted to DNA damage repair response. Shown is the result of clustering the components of the second neighborhood network arising from the original four Tax binding proteins known to be involved in the cellular DNA damage response. There are five clusters with three bridge proteins; DNA-PK is one of the bridge proteins. For clarity in drawing the network, we do not show edges from these three proteins to the individual proteins in the clusters. The numbers on the edges from these proteins to the clusters count the number of edges from each protein to proteins in each cluster.

### VI.3.5 Cellular Tax protein-complex contains DNA-PK

As a final verification of the binding between Tax and DNA-PKcs, we performed an affinity pull-down of cellular Tax protein complexes. In this study, we expressed either S-Tax or S-GFP and normalized for S-fusion protein amount. The extracts were then isolated by affinity purification of the S peptide and the complexes separated on SDS-PAGE and subjected to immunoblotting. Endogenous DNA-PKcs specifically associates with the Tax containing protein complex and is detected by staining with anti-DNA-PKcs (Fig. 29). These results confirm the identification of DNA-PKcs as a Tax-binding protein.

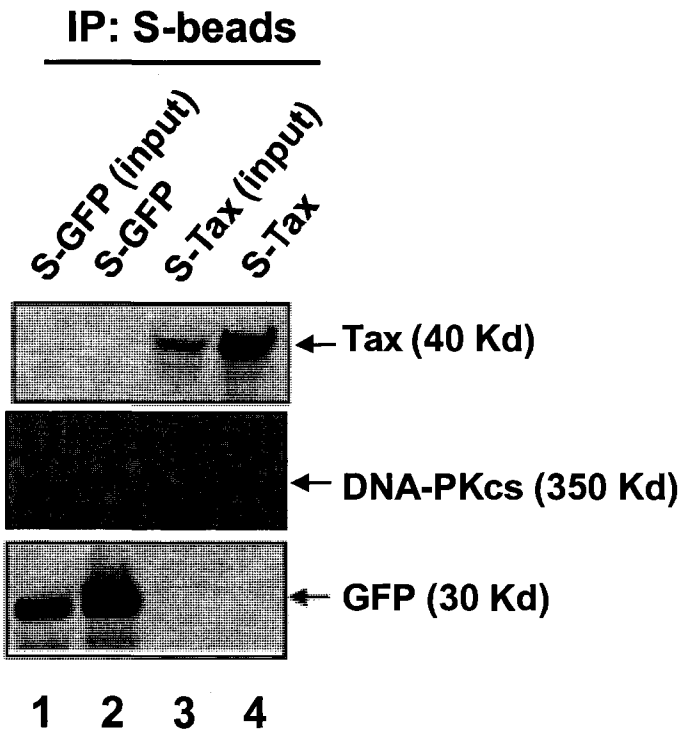



FIG. 29: HTLV-1 Tax binds to DNA-PKcs. The fusions proteins S-Tax and S-GFP were isolated from 293T cells as described and analyzed for co-precipitation with DNA-PKcs. Shown is the pre-isolated extract (input) for S-GFP (lane 1) and S-Tax (lane 3). Also shown is the affinity purified protein complexes for S-GFP (lane 2) and S-Tax (lane 4). Experimental normalization was achieved by using equal amounts of purified protein.

## VI.4 CONCLUSIONS

The HTLV-1 Tax protein has been defined by the proteins with which it interacts [92]. Therefore, it stands to reason that defining the functional properties of this protein will require an understanding of which cellular proteins it interacts with. Clearly, uncovering all potential interactions will include those with functional significance. However, determining which interactions support function and which interactions are of no consequence is an obvious and critical question. We have taken the approach that if we assume that Tax impacts the DNA damage repair process, as many studies support, then those interactions that are critical to the DNA damage repair process will hold greater promise of functional significance. Given this hypothesis, we devised a computational biology approach to help define which physical interactions warrant further study.

One of the challenges in computational systems biology is to create a tool to identify functional modules and the interactions among them from large-scale protein interaction networks. There are three major clustering approaches that have been employed to identify functional modules in proteomic networks. The first approach searches for subgraphs with specified connectivity, called network motifs, and characterizes these as functional modules or parts of them. This approach is not scalable for finding larger clusters in large-scale networks. The second approach, an example of which is work by Bader and Hogue [36], identifies a seed vertex, around which to grow a cluster. The seed vertex is identified by choosing a vertex of largest weight, where the weight of a vertex is a measure of the number of edges that join the neighbors of the vertex, the clustering coefficient. A vertex in the neighborhood of a cluster is added to it as long as its weight is close (within a threshold) to the weight of the seed vertex. Once a cluster has been identified, the procedure is repeated with a vertex of largest weight that currently does not belong to a cluster as the seed vertex. However, our experience comparing this approach with the spectral algorithms we employed in this study indicates that this method is less stable (i.e., the clusters obtained depend strongly on the seed vertices chosen). We used an improved clustering method [3] to reveal proteins that form functional modules, i.e., multiple proteins involved in the same biological function. This approach was used to apply an objective measure to the functional significance of a protein. Specifically we use this to both cluster proteins into specific functional domains as well as to objectively measure each individual protein's value to that functional domain.

When we compared the Tax-binding proteins generated from our physical mapping efforts, DNA-PK was in the top five best represented binding proteins and occupied a top tier ranking via our functional clustering for DNA damage proteins. Clearly, DNA-PK is a critical component in cellular processes that mediate response to damage and thus the fact that our clustering analysis places high value on this protein is as much a validation of the process as it is novel information. However, we began with a network of known Tax-binding proteins and their neighbors and second-neighbors, and DNA-PK was selected, through our functional clustering approach, whereas other equally critical damage response proteins were not. For instance, among the PI3K protein family members ATM and ATR hold positions of prominence in the DNA damage-response arena equal to DNA-PK [93]. In fact, the three proteins are considered redundant in specific pathways and are sometimes able to substitute functionally [94, 95, 96]. However, neither of the other two proteins was reflected in the upper tier interactions when using the Tax-designated protein networks. Furthermore, ATM and ATR were not found among the list of Tax-binding proteins identified in the physical isolation of Tax complexes, again verifying the novelty of the DNA-PK finding.

This is not the first time that DNA-PK has been targeted as a cellular protein through which Tax might mediate genomic instability [97]. In fact DNA-PK is known to mediate many functions associated with reported Tax activities. Specifically, Tax has been shown to cause constitutive activation of Chk2, a downstream target of DNA-PK [91]. DNA-PK can phosphorylate the tumor suppressor p53 at S15 and S37 [98] whereas Tax expression results in phosphorylation at S15 and S392 [99, 100]. In addition, we have recently shown that Tax interaction with DNA-PK results in saturation of the damage response (manuscript submitted). Thus, the Tax-DNA-PK interaction satisfies several previous observations regarding Tax function and provides a unifying model for all of these activities.

Clearly HTLV-1 Tax presents a biological model for an interesting protein with an overwhelming amount of associated published literature. The growth in the Tax knowledge base requires constant surveillance and verification if this body of work is to be useful in understanding how Tax functions. Additionally, as proteomic techniques continue to mature, the data generated in experimental studies is increasing exponentially. We have described a parallel process for combining in silico analysis with experimental proteomic analysis so that information gained in each process facilitates data mining of the other process. Further building of the Tax interactome should reveal other critical proteins that play key roles in mediating the biologically significant Tax functions within the host cell.

# CHAPTER VII

# CONCLUSIONS AND FUTURE WORKS

My work has two interconnected aims. One is to use computational approaches involving biological network modeling and analysis to gain insights into biological complex systems. The second aim is to provide experimentalists with focused targets to further their research goals. This chapter will provide a summary of my results and a discussion of future research.

## VII.1 CONCLUSIONS

I have proposed as an overall goal to study biological networks modeling and analysis.

In chapter 3, we have presented a hypergraph model for the protein complex network obtained from a large-scale experimental study to characterize the proteome of the yeast. Our model views the yeast proteome as a hypergraph, with the proteins corresponding to vertices and the complexes corresponding to hyper-edges. Previous work has modeled the protein complex data as a protein-protein interaction graph or as a complex intersection graph; both models lose information and require more space. Our results show that the yeast protein complex hypergraph is a small-world and power-law hypergraph. Also, we presented an algorithm for computing the $k-$core of a hypergraph, and use it to identify the core proteome, the maximum core of the protein complex hypergraph. We show that the core proteome of the yeast is enriched in essential and homologous proteins. We implement greedy approximation algorithms for variant minimum weight vertex covers of a hypergraph; these algorithms can be used to improve the reliability and efficiency of the experimental method that identifies the protein complex network.

In chapters 4 and 5, we described approaches to clustering protein-protein interaction networks in order to identify functional modules, groups of proteins forming multi-protein complexes accomplishing various functions in the cell. We have developed clustering methods that account for the small-world nature of the network. These methods make use of the concept of $k$-cores in a graph, and employ recursive spectral clustering to compute the functional modules. The computed clusters are annotated using their protein memberships into known multi-protein complexes in the yeast. Finally, we have applied our clustering approach to study a human proteomic network which is associated with a viral protein called Tax in the HTLV-1 virus, and human proteins that interact with Tax.

## VII.2 FUTURE RESEARCH

The research presented in this dissertation touches on a large number of subjects in biological networks modeling and analysis. In this section, I expand on what I consider the most relevant possibilities for future research. I consider two aspects: the improvement of the presented research in order to model multiscale protein data, and the expansion of the presented approaches of clustering to new research subjects such as biochemistry and oncology. *Oncology* is the branch of medicine that studies cancer and seeks to understand its development, diagnosis, treatment, and prevention. My discussion will be organized according to the sequence of chapters in the dissertation.

### VII.2.1 Modeling of Multi-scale Protein Networks

The proteomic universe is very complex. Each protein consists of one or multiple domains, regions in the protein that fold with a specified geometry. Proteins interact with each other through their domains. A group of proteins interact together to form multi-protein complexes, which are molecular machines responsible for cellular function. Functional modules are groups of proteins that interact with each other in a biochemical process. Groups of functional modules give us insight into the architecture of proteomic networks, and help us understand life processes at a higher level than otherwise.

In chapter 3, we have used bipartite graphs and hypergraphs to model the protein complex network. These models more faithfully represent the data and are more space-efficient than other representations computed from this data. Consequently, a question arises regarding the expansion of that model to different scales in the protein universe.

Here we focus on establishing a framework for multi-scale modeling and analysis of proteomic networks from the protein domain scale to the biochemical pathway and pathway regulation scales. The bipartite graph model that we have applied to the protein complex network in chapter 3 can be further extended to a multi-level description of proteomic data, both within and across genomes. This depiction does not imply a hierarchical architecture. Rather, any two levels can be related by a bipartite graph given appropriate experimental data, or can be *inferred* by linking two or more bipartite graphs representing the data. For example, if we have data that links proteins to pathways, and data linking proteins to supercomplexes, then inferred links between pathways and supercomplexes can be discovered through a bridged bipartite graph representation. We refer to this as a computationally derived network.

### VII.2.2 Hypergraph Clustering

In chapters 4 and 5, we have designed efficient algorithms for graph clustering. Future research will focus on how the graph clustering algorithm can be expanded to the level of bipartite or hypergraph clustering. We have applied a modified version of the spectral clustering algorithms presented in chapter 4 to the protein-complex hypergraph represented in chapter 3 to find biologically relevant multiprotein clusters (modules) and multicomplex clusters (we have called these supercomplexes).

A hypergraph can be represented as a bipartite graph which can be clustered using a modified version of our algorithm. A cluster in a bipartite graph will be referred to as a bicluster, since it has vertices from both parts. Here we show four biclusters obtained from spectral biclustering on the maximum core of the yeast protein complex bipartite graph in Fig. 30. Each row represents a complex, and each column represents a protein, in the maximum core of the bipartite graph. The complexes are colored according to their biological process: light-grey denotes a complex involved in transcription, black denotes a complex involved in protein synthesis and turnover, and grey denotes a complex responsible for RNA metabolism. Biological processes that are represented by only a few complexes in the core have been removed for clarity of presentation. It is clear from the biclustering approach that groups complexed by their protein memberships show that the complexes responsible for protein syntheis and turnover (black) interact with both the complexes involved in transcription in RNA metabolism, while the latter two groups do not interact with each other to the same extent.

### VII.2.3 Oncology and Biochemistry Applications

In chapter 6, we have applied our clustering methodology introduced in chapter 5 to study a human proteomic network which is associated with a viral protein called Tax in the HTLV-1 virus (human T-cell leukemia virus, type 1) and human proteins that interact with Tax involved in DNA repair, cell cycle control. This is part of a prototype study to create a Human Virus Interactome Resource (HVIR) [101] that provides interactions between viral proteins and human proteins to enable virologists and proteomics researchers to understand the mechanism of viral infection and transmission.

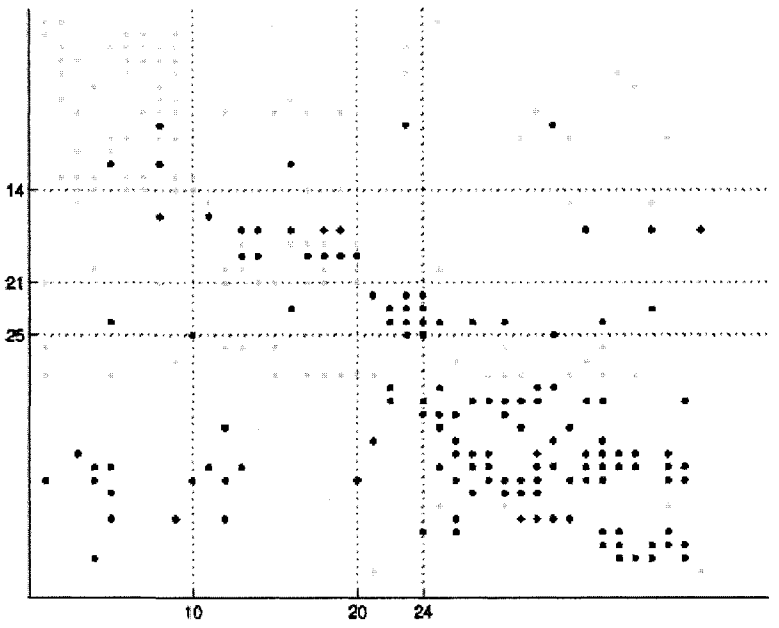



FIG. 30: A biclustering of the proteins and complexes in the maximum core of the yeast protein complex network.

My research in collaboration with Dr. O. John Semmes at Department of Pathology **(Eastern Virginia Medical School)** has helped guide experimentation in choosing significant proteins from the hundreds of human proteins interacting with Tax. It has also helped elucidate Tax's role in disrupting various biological processes. This work can be applied easily to other viruses such as the cytomegalovirus and their interactions with human proteins. In addition to upgrading the existing methods of clustering, we have a plan to extend our research into the other related areas such as biochemistry and oncology.

My current research in collaboration with Dr. Lesley Greene at Chemistry Department **(Old Dominion University)** presents a new technique to represent proteins by creating a two-level representation of the interactions among the protein residues. The first level represents the long range interactions between the residues in the protein. We cluster this network, and from the clustering, obtain a reduced network which represents each cluster by a vertex, and joins two clusters by an edge if they are connected by at least $k$ edges in the first level network. The reduced network provides a smaller representation of the protein, which then could be used to compare families of proteins. Using the second level networks, we will test whether the information captured by the reduced protein networks is conserved among members of the same protein super-family.

One can use a clustering method again to identify modules in the second level network. We refer to modules in the second level network as meta-modules. Meta-modules may reveal a higher order organization among protein structure.

We propose to study methods a) for finding consensus modules across multiple protein structure networks, b) for describing the relationships between consensus modules, and c) for assessing whether the relationship between consensus modules is preserved across different protein families. Thus we hope to contribute to the protein folding problem by using our clustering algorithm as a basic component.

Furthermore, during my postdoctoral research in collaboration with Dr. David Tuck in the Department of Pathology, Division of Pathology Informatics at the **Yale University School of Medicine**, we ultimately plan to incorporate our methodologies in protein functional analysis into a larger pathology- informatics and/or bio-informatics consortium enabling a joint learning from multiple types of genomic data: sequences, structures, microarrays, protein modifications, metabolic pathways, epigenetics, etc. From such a joint exploration system, we hope to acquire a comprehensive knowledge on the functioning of life, and exploit this knowledge in cancer epigenetic modeling.

The goal of the study is to identify and analyze key changes in methylation and chromatin remodeling patterns, specific to cancer (breast cancer). We will use systematic and graph theoretic approaches toward understanding chemical and structural changes throughout the genome that may be associated with the growth and development of breast cancer. Unlike genetic mutations, these complex processes - methylation and chromatin remodeling - can shut down or restrict the activity of key genes and other cellular structures, which normally protect cells from malignant transformation. This project is particularly focused on these processes because they are, unlike genetic mutations, potentially reversible with drug therapy. The project will employ integrated analysis of high throughput data of different types (gene expression, DNA methyaltion, histone modification, titling array, microRNA expression etc) and work on modeling human cancer. The specific goals of the project are: 1) increase the overall understanding of complex epigenetic alterations in neoplasms and 2) utilize high-end information for the improved prognosis, intervention and treatment of human breast cancer. The results of this study will hopefully allow the translational research community to gain insight into the unique problem of epigenetic alteration and its ultimate role in the clinical setting.

# BIBLIOGRAPHY

[1] E. Ramadan, A. Tarafdar, and A. Pothen, "A hypergraph model for the yeast protein complex network," *Proceedings of the IEEE Workshop on High Performance Computational Biology (HICOMB)*, 8 pp., 2004.

[2] E. Ramadan, C. Osgood, and A. Pothen, "The architecture of a proteomic network in the yeast," *Lecture Notes in Bioinformatics*, vol. 3695, pp. 265–276, 2005.

[3] E. Ramadan, C. Osgood, and A. Pothen, "A clustering algorithm for discovering functional modules in proteomic networks," 2008.

[4] E. Ramadan, M. Ward, X. Guo, S. Durkin, A. Sawyer, M. Vilela, C. Osgood, A. Pothen, and O. Semmes, "Development of the HTLV-1 Tax interactome by combined physical and in silico approaches," *Retrovirology*, vol. 5, no. 92, 13 pp., 2008.

[5] D. West, *Introduction to Graph Theory*. Upper Saddle River, NJ: Prentice Hall, 1996.

[6] B. Alberts *et al.*, *Molecular Biology of The Cell*. New York, NY: Garland Publishing, Inc, 1994.

[7] D. Mount, *Bioinformatics, Sequence and Genome Analysis*. New York: Cold Spring Harbor, 2000.

[8] A-C. Gavin *et al.*, "Functional organization of the yeast proteome by systematic analysis of protein complexes," *Nature*, vol. 415, pp. 141–147, 2002.

[9] D. Watts and S. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, pp. 440–442, 1998.

[10] T. Ito *et al.*, "A comprehensive two hybrid analysis to explore the yeast protein interactome," *Procs. Natl. Acad. Sci.*, vol. 98, pp. 4569–4574, 2001.

[11] H. Jeong *et al.*, "Lethality and centrality in protein networks," *Nature*, vol. 411, pp. 41–42, 2001.

[12] S. Maslov and K. Sneppen, "Specificity and stability in topology of protein networks," *Science*, vol. 296, pp. 910–913, 2002.

[13] R. Solé and R. Pastor-Satorras, "Complex networks in genomics and proteomics," in *Handbook of Graphs and Networks*, S. Bornholdt and H. Schuster, Eds. Wiley VCH, 2003, pp. 145–167.

[14] T. Hubbard *et al.*, "The ENSEMBL genome database project," pp. 38–41, 2002, http://www.ensembl.org/.

[15] D. Benson *et al.*, "GenBank," pp. 17–20, 2002, http://www.ncbi.nlm.nih.gov/Genbank/.

[16] P. Utez *et al.*, "A comprehensive analysis of protein-protein interactions in *saccharomyces cerevisiae*," *Nature*, vol. 403, pp. 623–627, 2000.

[17] T. Ito *et al.*, "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," *Procs. Natl. Acad. Sci.*, vol. 97, no. 3, pp. 1143–1147, 2000.

[18] Y. Ho *et al.*, "Systematic identification of protein complexes in *Saccharomyces Cerevisiae* by mass spectrometry," *Nature*, vol. 415, pp. 180–193, 2002.

[19] S. Fields and O. Song, "A novel genetic system to detect protein-protein interactions," *Nature*, vol. 340, pp. 245–246, 1989.

[20] G. Stoesser *et al.*, "The EMBL nucleotide sequence database," pp. 21–26, 2002, http://www.ebi.ac.uk/embl/.

[21] S. Dwight *et al.*, "*Saccharomyces* genome database (SGD) provides secondary gene annotation usingthe gene ontology (GO)," pp. 69–72, 2002, http://genome-www.stanford.edu/Saccharomyces/.

[22] M. Ashburner *et al.*, "FlyBase," pp. 19–20, 1993, http://flybase.bio.indiana.edu/.

[23] A. Bairoch and R. Apweiler, "The SWISS-PROT protein sequence database TrEMBL," pp. 45–48, 2000, http://www.ebi.ac.uk/swissprot/.

[24] P. McGarvey *et al.*, "PIR: a new resource for bioinformatics," pp. 290–291, 2000, http://pir.georgetown.edu/.

[25] H. Mewes *et al.*, "MIPS: a database for genomes and protein sequences," pp. 31–34, 2002, http://mips.gsf.de.

[26] M. Costanzo *et al.*, "YPD, PombelPD and WorkPD: model organism volumes of the bioknowledge library,and integrated resource for protein information," pp. 75–79, 2001, http://www.incyte.com/sequence/proteome/databases/YPD.shtml.

[27] I. Xenarios, "DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions," pp. 303–305, 2002, http://dip.doe-mbi.ucla.edu/.

[28] S. Maslov, K. Sneppen, and U. Alon, "Correlation profiles and motifs in complex networks," in *Handbook of Graphs and Networks*, S. Bornholdt and H. Schuster, Eds. Wiley VCH, pp. 168–198, 2003.

[29] A. Tong *et al.*, "A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules," *Science*, vol. 295, pp. 321–324, 2002.

[30] R. Jansen *et al.*, "A Bayesian network approach for predicting protein-protein interactions from genomic data," *Science*, vol. 302, pp. 449–453, 2003.

[31] D. Hochbaum, Ed., *Approximation Algorithms for NP-hard problems*. PWS Publishing Company, 1997.

[32] V. Vazirani, *Approximation Algorithms*. Springer Verlag, 2001.

[33] L. Hartwell, J. Hopfeld, and A. Murray, "From molecular to modular cell biology," *Nature*, vol. 402, pp. C47–C52, 1999.

[34] S. Bornholdt and H. Schuster, Eds., *Handbook of Graphs and Networks*. Wiley VCH, 2003.

[35] J. Han *et al.*, "Effect of sampling on topology predictions of protein-protein interaction networks," *Nat. Biotec.*, vol. 23, pp. 839–844, 2005.

[36] G. Bader and C. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics*, vol. 4, no. 2, 27 pp., 2003.

[37] A. Jarvis and E. Patrick, "Clustering based on a similarity measure based on shared nearest neighbors," *IEEE Trans. Computers*, vol. C-22, pp. 1025–1034, 1973.

[38] V. Spirin and L. Mirny, "Protein complexes and functional modules in molecular networks," *Procs. Natl. Acad. Sci.*, vol. 100, pp. 12 123–12 128, 2003.

[39] C. Ding *et al.*, "A MinMaxCut spectral method for data clustering and graph partitioning," *Proceedings of IEEE International Conference on Data Mining (ICDM)*, pp. 107–114, 2001.

[40] C. Ding *et al.*, "A unified representation of multi-protein complex data for modeling interaction networks," *Proteins: Structure, Function, and Genetics*, vol. 57, pp. 99–108, 2004.

[41] I. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proceedings of the ACM International Conference on Knowledge Discovery in Data Mining (KDD)*, 2001.

[42] P. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Addison Wesley, 2005.

[43] F. Chung and L. Lu, "The average distances in random graphs with given expected degrees," *Procs. Natl. Acad. Sci.*, vol. 99, no. 25, pp. 15 879–15 882, 2002.

[44] F. Galisson and P. Legrain, "The biochemical defects of prp4-1 and prp6-1 yeast splicing mutants reveal that the prp6 protein is required for the accumulation of the [u4/u6.u5] tri-snrnp," *Nucl. Acids Res.*, vol. 21, pp. 1555–1562, 1993.

[45] C. Noble *et al.*, "Rna14-rna15 assembly mediates the rna-binding capability of *saccharomyces cerevisiae* cleavage factor ia," *Nucl. Acids Res.*, vol. 32, pp. 3364–3375, 2004.

[46] B. Rymond., "Convergent transcripts of the yeast prp38-smd1 locus encode two essential splicing factors, including the d1 core polypeptide of small nuclear ribonucleoprotein particles," *Procs. Natl. Acad. Sci.*, vol. 90, pp. 848–852, 1993.

[47] X. Bi and D. Goss., "Wheat germ poly (a)-binding protein increases atpase and the rna helicase activity of translation initiation factors eIF4A, eIF4B and eif-iso-4f," *J. Biol. Chem.*, vol. 275, pp. 17 740–17 746, 2000.

[48] F. Angenstein *et al.*, "A receptor for activated c kinase is part of messenger ribonucleoprotein complexes associated with polya-mrna in neurons," *J. Neurosci.*, vol. 22, pp. 8827–8837, 2002.

[49] T. Achsel *et al.*, "The sm domain is an ancient rna-binding motif with oligo(u) specificity," *Procs. Natl. Acad. Sci.*, vol. 98, pp. 3685–3689, 2001.

[50] M. Fromont-Racine *et al.*, "Genome-wide protein interaction screens reveal functional networks involving sm-like proteins," *Yeast*, vol. 17, pp. 95–110, 2000.

[51] S. Urushiyama *et al.*, "The prp1+ gene required for pre-mrna splicing in *schizosaccharomyces pombe* encodes a protein that contains tpr motifs and is similar to prp6p of budding yeast," *Genetics*, vol. 147, pp. 101–115, 1997.

[52] A. Barabasi and Z. Oltvai, "Network biology: understanding the cells functional organization," *Nat. Rev. Gen.*, vol. 5, pp. 101–113, 2004.

[53] J. Bader, "Greedily building protein networks with confidence," *Bioinformatics*, vol. 19, no. 15, pp. 1869–1874, 2003.

[54] S. Asthana *et al.*, "Predicting protein complex membership using probabilistic network reliability," *Genome Res.*, vol. 14, no. 6, pp. 1170–1175, 2004.

[55] N. Przulj, D. Wigle, and I. Jurisica, "Functional topology in a network of protein interactions," *Bioinformatics*, vol. 20, no. 3, pp. 340–348, 2004.

[56] A. King, N. Przulj, and I. Jurisica, "Protein complex prediction via cost-based clustering," *Bioinformatics*, vol. 20, no. 17, pp. 3013–3020, 2004.

[57] A. Enright, S. Dongen, and C. Ouzounis, "An efficient algorithm for large-scale detection of protein families," *Nucl. Acids Res.*, vol. 30, no. 7, pp. 1575–1584, 2002.

[58] J. Pereira-Leal, A. Enright, and C. Ouzounis, "Detection of functional modules from protein interaction networks," *Proteins*, vol. 54, no. 49–57, 2004.

[59] B. Adamcsek, G. Palla, I. Farkas, I. Derenyi, and T. Vicsek, "Cfinder: locating cliques and overlapping modules in biological networks," *Bioinformatics*, vol. 22, pp. 1021–1023, 2006.

[60] H. Yu *et al.*, "The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics," *PLoS. Comput. Biol.*, vol. 3, no. 4, pp. 713–720, 2007.

[61] N. Batada *et al.*, "Stratus not altocumulus: A new view of the yeast protein interaction network," *PLoS. Biol.*, vol. 4, no. 10, pp. 1721–1731, 2006.

[62] G. Bader, D. Betel, and C. Hogue, "BIND: the biomolecular interaction network database," pp. 248–250, 2003, http://www.binddb.org/.

[63] T. Reguly *et al.*, "Comprehensive curation and analysis of global interaction networks in *saccharomyces cerevisiae*," *J. Biol.*, pp. 5–11, 2006.

[64] A. Zanzoni *et al.*, "Mint:a molecular interaction database," *FEBS Lett.*, vol. 513, pp. 135-140, 2002.

[65] S. Peri *et al.*, "Development of human protein reference database as an initial platform for approaching systems biology in humans," *Genome Res.*, vol. 13, pp. 2363–2371, 2003.

[66] J. Rual *et al.*, "Towards a proteome-scale map of the human protein-protein interaction network," *Nature*, vol. 437, pp. 1173–1178, 2005.

[67] S. Li *et al.*, "A map of the interactome network of the metazoan *C. Elegans*," *Science*, vol. 303, pp. 540–543, 2004.

[68] R. Cho *et al.*, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Mol. Cell. Biol.*, vol. 2, pp. 65–73, 1998.

[69] T. Huges *et al.*, "Functional discovery via a compendium of expression profiles," *Cell*, vol. 102, pp. 109–126, 2000.

[70] H. Mewes *et al.*, "MIPS: a database for genomes and protein sequences," pp. 31–34, 2002, http://mips.gsf.de.

[71] T. Consortium, "GO: The gene ontology database and information resource." pp. 258–261, 2004, http://www.geneontology.org.

[72] S. Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, University of Utrecht, 2000.

[73] I. Derenyi, G. Palla, and T. Vicsek, "Clique percolation in random networks," *Phys. Rev. Lett.*, vol. 94, no. 16, 4 pp., 2005.

[74] S. Brohee and J. Helden, "Evaluation of clustering algorithms for protein-protein interaction networks," *BMC Bioniformatics*, vol. 7, no. 488, 19 pp., 2006.

[75] A. Gessain *et al.*, "Antibodies to human T-lymphotropic virus type-I in patients with tropical spastic paraparesis," *Lancet*, vol. 2, pp. 407–410, 1985.

[76] M. Osame *et al.*, "HTLV-I associated myelopathy, a new clinical entity," *Lancet*, vol. 1, pp. 1031–1032, 1986.

[77] B. Poiesz *et al.*, "Detection and isolation of type C retrovirus particles from fresh and cultured lymphocytes of a patient with cutaneous T-cell lymphoma," *Procs. Natl. Acad. Sci.*, vol. 77, pp. 7415–7419, 1980.

[78] K. Takatsuki, "Discovery of adult T-cell leukemia," *Retrovirology*, vol. 2, no. 16, 3 pp., 2005.

[79] M. Yoshida *et al.*, "Monoclonal integration of human T-cell leukemia provirus in all primary tumors of adult T-cell leukemia suggests causative role of human T-cell leukemia virus in the disease," *Procs. Natl. Acad. Sci.*, vol. 81, pp. 2434–2537, 1984.

[80] M. Yoshida, I. Miyoshi, and Y. Hinuma, "Isolation and characterization of retrovirus from cell lines of human adult T-cell leukemia and its implication in the disease," *Procs. Natl. Acad. Sci.*, vol. 79, pp. 2031–2035, 1982.

[81] C. Giam and K. Jeang, "HTLV-1 Tax and adult T-cell leukemia," *Front Biosci.*, vol. 12, pp. 1496–1507, 2007.

[82] S. Marriott and O. Semmes, "Impact of HTLV-I Tax on cell cycle progression and the cellular DNA damage repair response," *Oncogene*, vol. 24, pp. 5986–5995, 2005.

[83] J. Peloponese, T. Kinjo, and K. Jeang, "Human T-cell leukemia virus type 1 Tax and cellular transformation," *Int. J. Hematol.*, vol. 86, pp. 101–106, 2007.

[84] D. Jin, F. Spencer, and K. Jeang, "Human T cell leukemia virus type 1 oncoprotein Tax targets the human mitotic checkpoint protein MAD1," *Cell*, vol. 93, pp. 81–91, 1998.

[85] E. Harhaj and S. Sun, "IKKgamma serves as a docking subunit of the IkappaB kinase (IKK) and mediates interaction of IKK with the human T-cell leukemia virus Tax protein," *J. Biol. Chem.*, vol. 274, pp. 22 911–22 914, 1999.

[86] D. Jin *et al.*, "Role of adapter function in oncoprotein-mediated activation of NF-kappaB," *J. Biol. Chem.*, vol. 274, pp. 17 402–17 405, 1999.

[87] K. Wu *et al.*, "Protein profile of tax-associated complexes," *J. Biol. Chem.*, vol. 279, pp. 495–508, 2004.

[88] S. Peri *et al.*, "Human protein reference database as a discovery resource for proteomics," *Nucl. Acids Res.*, vol. 32, pp. 497–501, 2004.

[89] U. Brandes, "A faster algorithm for betweeness centrality," *J. Math. Soc.*, vol. 25, pp. 163–177, 2001.

[90] S. Durkin *et al.*, "Site-specific phosphorylation differentiates active from inactive forms of the human T-cell leukemia virus type 1 Tax oncoprotein," *J. Biol. Chem.*, vol. 281, pp. 31 705–31 712, 2006.

[91] S. Gupta *et al.*, "Human T-cell leukemia virus type 1 Tax oncoprotein prevents DNA damage-induced chromatin egress of hyperphosphorylated Chk2," *J. Biol. Chem.*, vol. 282, pp. 29 431–29 440, 2007.

[92] D. Wycuff and S. Marriott, "The HTLV-I Tax oncoprotein: hyper-tasking at the molecular level," *Front Biosci*, vol. 10, pp. 620–642, 2005.

[93] R. Abraham, "Pi 3-kinase related kinases: 'big' players in stress-induced signaling pathways," *DNA Repair*, vol. 3, pp. 883–887, 2004.

[94] R. Marone *et al.*, "Targeting phosphoinositide 3-kinase: moving towards therapy," *Biochim. Biophys. Acta.*, vol. 1784, pp. 159–185, 2008.

[95] Y. Pommier *et al.*, "Targeting chk2 kinase: molecular interaction maps and therapeutic rationale," *Curr. Pharm. Des.*, vol. 11, pp. 2855–2872, 2005.

[96] J. Yang *et al.*, "ATM, ATR and DNA-PK: initiators of the cellular genotoxic stress responses," *Carcinogenesis*, vol. 24, pp. 1571–1580, 2003.

[97] F. Majone *et al.*, "Ku protein as a potential human T-cell leukemia virus type 1 (HTLV-1) Tax target in clastogenic chromosomal instability of mammalian cells," *Retrovirology*, vol. 2, no. 45, 10 pp., 2005.

[98] S. Lees-Miller *et al.*, "Human DNA-activated protein kinase phosphorylates serines 15 and 37 in the amino-terminal transactivation domain of human p53," *Mol. Cell. Biol.*, vol. 12, pp. 5041–5049, 1992.

[99] C. Pise-Masison *et al.*, "Inactivation of p53 by human T-cell lymphotropic virus type 1 Tax requires activation of the NF-kappaB pathway and is dependent on p53 phosphorylation," *Mol. Cell. Biol.*, vol. 20, pp. 3377–3386, 2000.

[100] C. Pise-Masison *et al.*, "Phosphorylation of p53: a novel pathway for p53 inactivation in human T-cell lymphotropic virus type 1-transformed cells," *J. Virol.*, vol. 72, pp. 6348–6355, 1998.

[101] A. Pothen, M. Zubair, K. Maly, C. Osgood, O. Semmes, E. Ramadan, M. Abu-elela, and P. Namburi, "A Prototype of the Human Virus Interactome Resource (HVIR)," *Proceedings of the Russian Conference on Digital Libraries*, 11 pp., 2006.

# VITA

Emad Y. Ramadan
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

Emad Ramadan earned both his M.Sc. in computer science (1999) and B.Sc. in Computer Science and Mathematical Statistics (1994) at Faculty of Science, Alexandria University, Egypt. Emad was approached by the Computer Science Department at Old Dominion University and offered an assistantship in the Ph.D. program and awarded a GAANN Fellowship (Graduate Assistantships in Areas of National Need) funded by the department of Education. Emad has published several papers on Bioinformatics and Pathology informatics areas. He is a member in the Science Alliance of the New York Academy of Sciences, IEEE, ACM, SIAM, and the National Postdoctoral Association.

Immediately after his dissertation defense in August 2008, Emad joined the Department of Pathology, Division of Pathology Informatics at the Yale University School of Medicine as a postdoctoral associate. He is involved in applied bioinformatics research group with a focus on analysis of genetic and epigenetic regulation in biological networks in stem cells and cancer.

Typeset using LaTeX.