Title
Modularity Conserved during Evolution: Algorithms and Analysis

Author
Hodgkinson, Luqman

Publication Date
2013

Peer reviewed|Thesis/dissertation

Modularity Conserved during Evolution: Algorithms and Analysis

by

Luqman Hodgkinson

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science with Designated Emphasis in Computational and Genomic Biology

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Richard M. Karp, Chair
Professor Christos H. Papadimitriou
Professor John P. Huelsenbeck

Spring 2013

Modularity Conserved during Evolution: Algorithms and Analysis

by

Luqman Hodgkinson

Abstract

Modularity Conserved during Evolution: Algorithms and Analysis
by
Luqman Hodgkinson
Doctor of Philosophy in Computer Science with Designated Emphasis in Computational and Genomic Biology
University of California, Berkeley
Professor Richard M. Karp, Chair

Modularity is a defining feature of biological systems. This dissertation presents our work on the development of algorithms to detect modularity in protein interaction networks and techniques of analysis for interpreting the results. A multiprotein module is a collection of proteins exhibiting modularity in their interactions. Multiprotein modules may perform essential functions and be conserved by purifying selection.

A new linear-time algorithm named Produles offers significant algorithmic advantages over previous approaches. An algorithmic framework for evaluation is presented that facilitates evaluation of algorithms for detecting conserved modularity with respect to their algorithmic goals.

Optimization criteria for detecting homologous multiprotein modules are examined, and their effects on biological process enrichment are quantified. Graph theoretic properties that arise from the physical construction of protein interaction networks account for 36 percent of the variance in biological process enrichment. Protein interaction similarities between conserved modules have only minor effects on biological process enrichment. As random modules increase in size, both biological process enrichment and modularity tend to improve, though modularity does not show this trend in small modules. To adjust for this trend, we recommend a size correction based on random sampling of modules when using biological process enrichment to evaluate module boundaries.

Supporting software has been developed useful for designing high quality algorithms for detecting conserved multiprotein modularity. EasyProt is a parallel implementation of scientific workflow software designed for cloud computing that retrieves data from several sources, runs algorithms in parallel, and computes evaluation statistics. VieProt is visualization software for conserved multiprotein modularity that uses a dynamic force-directed layout and displays quality measures and statistical summaries.

With high quality protein interaction data, it may be possible to use modules to improve the prediction of proteins that are orthologous to each other and that have maintained their function. We present statistical methods that may be useful for this purpose. The utility of these models will depend on anticipated improvements in protein interaction data quality.

Acknowledgements

This thesis presents joint work with several collaborators at Berkeley. The work on algorithm design and analysis is joint work with Richard M. Karp, the work on reproducible computation is joint work with Eric Brewer and Javier Rosa, and the work on visualization of modularity is joint work with Nicholas Kong. I thank all of them for their contributions. They have been pleasant collaborators with stimulating ideas and good designs. Our descriptions of the projects in this thesis overlap somewhat with our published descriptions in [Hodgkinson and Karp, 2011a], [Hodgkinson and Karp, 2011b], [Hodgkinson and Karp, 2012], and [Hodgkinson *et al.*, 2012].

With regard to the design of Produles, we thank Manikandan Narayanan for providing a starting point with his Match-and-Split algorithm, graciously providing full source code and ideas. We thank Bonnie Kirkpatrick for her guidance at several stages of the project, and Shuai Cheng Li for providing interesting alternative ideas for various subroutines. We thank John Huelsenbeck and Christos Papadimitriou for listening to ideas and providing constructive feedback. We thank Kimmen Sjölander and Ruchira Datta for their contributions of PHOG data and their encouragement.

We thank Nicholas Kong for contributions to the design of EasyProt and the initial implementation of VieProt. We thank Bonnie Kirkpatrick and Shuai Cheng Li for their feedback with regards to both EasyProt and VieProt. The long hours working on the implementation of EasyProt together with Javier Rosa were some of the highlights of my time at Berkeley, and we thank Eric Brewer for his guidance on this work. We thank Sabry Razick and Ian Donaldson for their help with iRefIndex and Psicquic.

We thank Julie Hopper and Chris DiVittorio for their guidance in statistical analysis techniques for biological data, and Julie Hopper for her guidance with R. We thank Martin Wainwright for his guidance in statistical methods for NetAlign, and Sehand Negahban who provided helpful discussions about the graphical model used in NetAlign during its formative stages. We thank Garvesh Raskutti for valuable advice regarding the analysis and comparison of optimization criteria for detecting conserved multiprotein modularity.

My PhD work has been supported by grants NSF IIS-0803937 and NSF CCF-1052553. I thank LaShana Porlaris and Xuan Quach for their guidance in the Ph.D. program, and Linda Stubbs for her management of funding issues.

I am pleased to warmly thank those who made it possible for me to pursue my academic dreams. Dan and Nancy Hoffman invited me into their home when I was a construction worker and opened the doors of academia to me. Graham Creasey and Laura Holmes provided a family away from home during my young life. Jack Rumbaugh personally drove me to Hiram College to talk to the director of admissions, and secured my admission, shortly before tragically passing away. To these key figures in my life I remain eternally grateful.

I am pleased to warmly thank key academic figures in my life for the critical contributions that they added to my success. My advisors at Hiram College, Ellen Walker and Oberta Slotterbeck (Obie), gave me the initial support I needed in my early education and have been extremely supportive and encouraging ever since. Denny Taylor and

Sigrid Anderson provided supportive academic in-classroom and out-of-classroom instruction with a personal touch. Feodor Dragan at Kent State University gave me important opportunities to advance in my studies. Mihalis Yannakakis at Columbia University provided guidance in computational complexity, algorithms, and mathematics, and maintained an unfailing belief in my abilities, even during difficult times. Itsik Pe'er at Columbia University provided initial instruction in computational biology, provided friendly support, and accepted me as his Ph.D. student during difficult times, though I chose to attend Berkeley. The professors who have taught me at the University of California, Berkeley, provided the education in natural sciences that I desperately needed.

I am especially pleased to warmly thank my advisor, Professor Richard M. Karp. Throughout my studies at the University of California, Professor Karp has guided me towards interesting problems. I have learned much from my interactions with him about the pleasure of solving interesting problems that can be viewed as puzzles. I deeply appreciate that he took me as his student during a difficult time in my life, and for his supportiveness when I was filling large gaps in my education. It was a great pleasure to serve as his teaching assistant twice for CS270: Combinatorial Algorithms and Data Structures.

It is a great pleasure to thank my good friends from various stages in my life:
Joseph Mukhwana Khamala, Charles Mushila Shiveka, T-chindia Mahero, and Lemmy Mahero at Shibuli and Kakamega; Suniil Sadarangani at Mumbai; Gerard Nielsen, Della Alltop, Wolali Dedo, and Ryosuke Kadoi at Hiram College; Tuan-Anh Tran at Kent State University; Sergio Coutinho de Biasi at Columbia University; and Melissa Sandoval and Sihye Kim Edwards at the University of California, Berkeley. These good friends in my life provided constant support, often continuing to the present. It also gives me pleasure to thank the Valenzuela family: Lisa and Ed and their children Sophia and Mia, for welcoming me to their home and providing a supportive and friendly environment during the final stages of my dissertation writing.

From the depths of my heart, it gives me great pleasure to warmly thank my family: Wekoye, Worldpeace, and Lulu, for their constant support. The support system they provided has encouraged me to pursue perpetuation of life and understanding of the universe. I am deeply appreciative to them, and to all that have walked with me on this personal journey from a world of irrationality to the heights of understanding and reason.

To Wekoye, Worldpeace, and Lulu, and to the next generation: Victoria, Lillianne, Angelina, Nicolas, and Katherine, with love.

Contents

Chapter 1

Introduction

1.1 Modularity in Biological Systems

1.1.1 Modularity and Evolution

Life is more than three billion years old [de Duve, 2011]. Since the beginning of life, the principle of natural selection has operated, that is, life has continued to evolve for selective advantage within constraints imposed by chemistry and physics. Organisms that could change a single feature without disrupting the function of other features had a selective advantage, so life evolved to be modular [Wagner, 1996]. In modular biological systems, changes within a single module did not significantly affect the function of other modules. Modularity is ubiquitous in biological systems [Rorick, 2012]. Biological systems tend to be modular, and, thus, evolvable [Callebaut and Rasskin-Gutman, 2005].

1.1.2 Biological Cells and Protein Interactions

Inside biological cells, proteins are molecular machines. There are approximately 22,000 genes in the human genome [International Human Genome Sequencing Consortium, 2004] that code for proteins, and even more proteins than this due to alternative splicing [Nilsen and Graveley, 2010]. A single protein in *Drosophila melanogaster*, Down syndrome cell adhesion molecule (Dscam), can generate 38,016 distinct mRNA isoforms, more than the total number of genes in the species [Nilsen and Graveley, 2010]. Approximately 8% of the coding capacity of a mammalian genome is devoted to the synthesis of proteins that serve as regulators of gene transcription [Alberts *et al.*, 2008, page 450]. Some proteins stay in the cytosol whereas others embed themselves in the cell membrane. Membrane proteins represent 20-30% of all proteins in the Arabidopsis proteome [Schwacke *et al.*, 2003].

Each protein functions in a neighborhood, interacting with other proteins and molecules to perform its tasks. Experimental assays can detect whether two proteins are likely to interact. Testing interactions among all, or a significant fraction, of proteins for a species yields a protein interaction network, or interactome (Fig 1.1), with vertices representing proteins and edges representing interactions [Vidal, 2005]. The interactome is only an

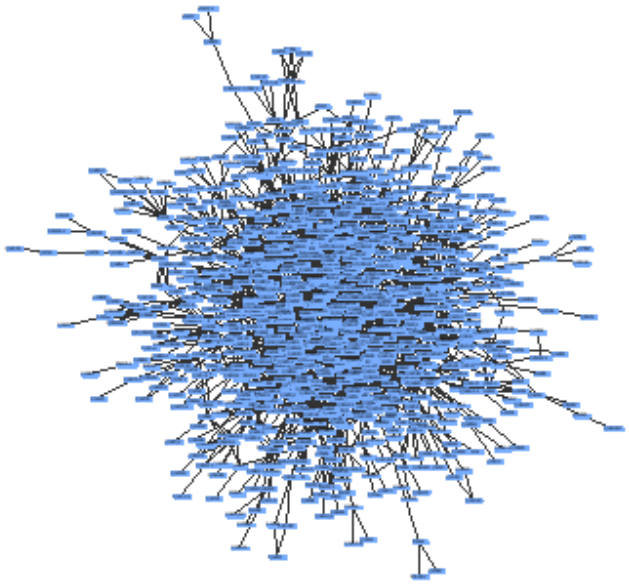


Figure 1.1: The largest connected component of the Drosophila interactome. Not all proteins are visible.

approximation to the organization of the cell as two proteins that can bind to each other may be carefully separated by the cell and may never interact in vivo. Sometimes proteins can directly interact only weakly, but a meaningful interaction is mediated by scaffold proteins. In other cases, weak interactions are actually useless crosstalk between modules [Alberts *et al.*, 2008, page 931]. Moreover, in multi-cellular species, there are many cell types, and only a subset of genes are expressed in each cell type. There are also temporal separations in which various genes are expressed at different times in the cell cycle. Even with these limitations, the interactome is a reasonable starting point for understanding modular organization of protein interactions.

Protein interactions are of several different kinds. There are stable interactions where proteins associate for long periods of time. These protein complexes are best detected using immunoprecipitation-based methods [Lalonde *et al.*, 2008]. Other interactions are short lived, such as proteins functioning in signalling pathways that are best detected using yeast two-hybrid or more advanced techniques such as Förster resonance energy transfer (FRET) [Lalonde *et al.*, 2008]. The human interactome is estimated to contain 650,000 protein interactions [Stumpf *et al.*, 2008].

Protein interactions are often mediated by particular domains [Pawson and Nash, 2003]. Domains can recognize short signal peptides forming domain-peptide interactions or they can can bind to other entire domains forming domain-domain interactions [Pawson and Nash, 2003]. Several methods have been developed for predicting domain interactions from protein interactions [Ta and Holm, 2009].

In an interactome, each meaningful protein interaction places an epistatic constraint on the protein interactants [Schlosser and Wagner, 2008]. Evolution of cells would be severely restricted if interactomes were complete graphs with each interaction being essential for biological viability. Modularity allows each module to evolve with limited dependence on the evolution of other modules.

1.1.3 Purifying Selection and Homology

When genomes were fully sequenced, it was discovered that a large fraction of proteins were conserved across hundreds of millions of years [International Human Genome Sequencing Consortium, 2001; Adams *et al.*, 2000]. Flies and humans separated by at least 500 million years [Futuyma, 2009] have easily detected homology between a large fraction (25-50%) of their proteins [Rubin *et al.*, 2000]. If any of these proteins were broken, the organism either died or was at a disadvantage so that its offspring became extinct. This powerful force for conserving features across millions of years is called purifying selection. Positive selection describes the advantage given to changes that improve fitness in the current environment. Breaking important cell machinery rarely leads to an advantage.

Vestiges of similarity surviving evolution, homology, appear in distantly related taxa at many different levels. Homology means similarity due to common ancestry, especially after millions of years of divergence [Futuyma, 2009]. It is usually applied at the suborganism level, for example, to morphological characters, multiprotein modules, and proteins. Purifying selection maintains homology.

1.1.4 Conservation of Multiprotein Modules

Proteins are highly conserved for hundreds of millions of years despite large phenotypic changes at the organism level, but perhaps multiprotein modular structure is less highly conserved. Perhaps proteins change interaction partners easily and organize into new functional groups. One study estimated that eukaryotic species have rewired their interactomes at a fast rate of approximately 10^{-5} interactions changed per protein pair, per million years of divergence [Beltrao and Serrano, 2007].

There is primary evidence of multiprotein modularity conserved for hundreds of millions of years including fundamental molecular mechanisms underlying cell-to-cell communication such as the Notch pathway [Celis, 2004]. How widespread are these instances of conserved multiprotein modularity? How can we effectively use computational techniques to identify conserved modularity in interactomes with thousands of proteins? It has been considered a difficult task to draw boundaries around modules in large biological data sets, both because the modules themselves have imprecise boundaries in biological systems [Schlosser and Wagner, 2004] and because data sets are incomplete and imprecise [Ali and Deane, 2010].

Early ground-breaking studies searched for conserved pathways in interactomes of *Helicobacter pylori* and *Saccharomyces cerevisiae* [Kelley *et al.*, 2003], and for conserved complexes in interactomes of *S. cerevisiae*, *Caenorhabditis elegans*, and *Drosophila melanogaster*

[Sharan *et al.*, 2005a]. Additional attempts to identify conserved modularity across interactomes were subsequently published [Koyutürk *et al.*, 2006b; Flannick *et al.*, 2006; Hirsh and Sharan, 2006; Narayanan and Karp, 2007; Dutkowski and Tiuryn, 2007; Guo and Hartemink, 2009].

Not all pairs of conserved multiprotein modules need have the same function. During evolution, modules are likely to be co-opted in new functional contexts [Winther, 2005]. This leads to cases where the function is not preserved though the modularity is preserved. After functional divergence, the modules in both lineages are subject to purifying selection even though the important functions the modules perform are somewhat different.

Conservation in phenotypically different species indicates a possibility that the modules are evolutionary modules. An evolutionary module is a module that performs a unitary function with an architecture that allows it to evolve quasi-independently from other features [Brandon, 2005]. An evolutionary module may be compatible with many different global architectures. The purifying selection placed on proteins in an evolutionary multiprotein module may be similar to that placed on the module as a whole. An analogous example from morphology is that hands and feet evolve quasi-independently and can be considered evolutionary modules, whereas left feet and right feet do not evolve quasi-independently and are not evolutionary modules.

Elucidating modular structure of interactomes has practical consequences. For example, certain signal transduction pathways are common to many diseases [Suthram *et al.*, 2010]. Understanding the composition of natural units of the cell can allow several avenues for therapy using different target proteins that affect the same module.

Biological networks, including protein interaction networks, are the subject of the new discipline of systems biology [Palsson, 2006; Klipp *et al.*, 2009; Palsson, 2011]. The goal of systems biology is to understand the structure and functioning of biological systems. One feature of systems biology is the study of networks of components that lead to emergent properties not exhibited by individual components. Static systems biology considers networks without a time dimension whereas dynamic systems biology models behavior of networks over time. Interactomics is a branch of static systems biology. Interactomics does not make spatial or temporal distinctions: e.g. does not consider cell types, cellular compartments, or the cell cycle.

1.2 Biological Data: Interactomics and Homology

1.2.1 Assays to Detect Protein Interactions

Yeast Two-Hybrid (Y2H) is an assay amenable to high-throughput automation that is used to detect protein interactions [Fields and Song, 1989; Fields, 2009]. In Y2H, genes encoding two proteins being tested for interaction are attached to separate halves of a split gene encoding a protein that transcribes a reporter gene [Walhout and Boulton, 2006]. The two hybrid genes are then expressed in a yeast cell. If the two hybrid proteins interact in the yeast cell, the reporter protein is reconstituted and a signal is detected. There are some limitations to the Y2H protocol leading to significant false positive and false negative

rates [Lalonde *et al.*, 2008; Hart *et al.*, 2006]. Proteins are overexpressed in the yeast cell, thus modifying the relative concentrations of potential interaction partners from the in vivo state [Lalonde *et al.*, 2008]. Also, membrane proteins are underrepresented, because they are retained at the membrane and unavailable for reconstitution of a functional transcription factor in the nucleus [Lalonde *et al.*, 2008].

To remedy the inability of yeast two-hybrid to accurately detect protein interactions involving membrane proteins, the mating-based split-ubiquitin system (mbSUS) was developed [Obrdlik *et al.*, 2004]. In this assay, the gene coding for ubiquitin is cleaved into two halves and separate halves are attached to genes encoding the two proteins of interest. One of the proteins must be a membrane protein to which a transcription factor is also attached. If the hybrid proteins bind to each other when the hybrid genes are expressed in yeast cells, the ubiquitin becomes functional and is recognized by ubiquitin-specific proteases that cleave off the transcription factor. Since the ubiquitin is recognized when not in the nucleus, interactions involving membrane proteins can be detected. However, at least one of the proteins must be a membrane protein so that the hybrid with attached transcription factor does not enter the nucleus and generate a signal [Lalonde *et al.*, 2008, Figure 2 caption].

Clusters of interacting proteins are often best detected *in vitro* using biochemical techniques followed by mass spectrometry for identification [Miernyk and Thelen, 2008]. One of the most widely used techniques is coimmunoprecipitation where an antibody is developed for a protein of interest. The antibody is secured to beads, such as protein A/G beads [Miernyk and Thelen, 2008], and the protein of interest along with its interacting proteins become attached to the beads and are then purified. Cross-linking techniques can also be used where cross-linking reagents, such as primary amines and formaldehyde, are added to cell extracts and interacting proteins become covalently linked to each other [Miernyk and Thelen, 2008]. This stabilizes protein interactions allowing protein complexes to be purified and identified [Liu and West, 2002]. Some limitations of these biochemical techniques is that proteins from different compartments are brought together in the cell extracts [Lalonde *et al.*, 2008] and, for the coimmunoprecipitation approach, it can be difficult to generate antibodies specific for each protein of interest [Madeira *et al.*, 2009]. Transient and weak interactions, even when biological important are often missed by these methods [Koh *et al.*, 2012].

Optical methods for detecting protein interactions have also been developed and used [Masi *et al.*, 2010; Madeira *et al.*, 2009; Lalonde *et al.*, 2008], though not yet in high throughput. Förster resonance energy transfer (FRET) is often used to determine distance between two proteins [Masi *et al.*, 2010]. In FRET, a fluorophore accepts a photon of a given wavelength, gets excited, and transfers this energy to a nearby fluorophore that emits a photon of a longer wavelength [Masi *et al.*, 2010]. When the fluorophores are attached to two potentially interacting proteins, either by incorporating them into the genes encoding the proteins or via antibodies [Masi *et al.*, 2010], estimates can be made of how likely the proteins are to interact. Surface plasmon resonance (SPR) is an assay that allows for dynamic studies of protein interactions [Madeira *et al.*, 2009; Lalonde *et al.*, 2008]. In SPR,

a protein of interest is bound to a gold foil, and the foil is placed in a solution containing potentially interacting proteins. Light is passed into the foil and the refractive index is measured. If the protein of interest forms a complex with the proteins in solution, the refractive index changes detectably [Madeira *et al.*, 2009; Lalonde *et al.*, 2008].

1.2.2 Databases Storing Protein Interactions

Biologists detect protein interactions and publish the results. Protein interaction databases consolidate the protein interaction data from published papers. The consolidated protein interaction databases contain a wealth of knowledge from numerous experiments, numerous experimenters, and numerous experimental assays. The first protein interactions stored in databases were interactions based on common metabolic pathways. EcoCyc [Karp *et al.*, 1996; Keseler *et al.*, 2013] stored metabolic interactions for *Escherichia coli* and KEGG [Goto *et al.*, 1997; Kanehisa *et al.*, 2010] stored metabolic interactions more generally. MetaCyc [Karp *et al.*, 2000; Karp *et al.*, 2013] also began to store metabolic interactions more generally. KEGG began to store regulatory interactions [Ogata *et al.*, 1998] and physical protein interactions [Nakao *et al.*, 1999] in addition to metabolic interactions. DIP [Xenarios *et al.*, 2000; Salwinski *et al.*, 2004] began to store physical interactions exclusively. A large number of additional databases were then established that are currently maintained and curated: most notably IntAct [Kerrien *et al.*, 2012], BioGrid [Chatr-aryamontri *et al.*, 2013], HPRD [Prasad *et al.*, 2009], BIND [Isserlin *et al.*, 2011], MINT [Licata *et al.*, 2012], MIPS [Mewes *et al.*, 2011], InnateDB [Breuer *et al.*, 2013], and MatrixDB [Chautard *et al.*, 2011]. STRING [Franceschini *et al.*, 2013] began to store computationally predicted protein interactions of all kinds as well as experimentally detected interactions. Because of the many databases storing often redundant, but sometimes unique, information, iRefIndex [Razick *et al.*, 2008] was established to consolidate data from the various protein interaction databases using a common format and web services interface.

1.2.3 Algorithms to Detect Homology and Orthology of Proteins

When comparing interactomes across species, it is important to know which proteins are homologous in order to compare protein interactions across homologous proteins. In fact, a more stringent requirement on the proteins is sometimes needed, i.e. orthology. Orthologous proteins are proteins in two separate species that derive from the same ancestral protein in the last common ancestor of those two species [Fitch, 2000].

Similarity of amino acid sequences in proteins, or similarity of nucleic acid sequences in the genes encoding proteins, can be used to infer homology. BLAST is a program that does just this [Altschul *et al.*, 1990; Altschul *et al.*, 1997]. When a protein is compared for sequence similarity with a large database of other proteins, BLAST returns an E-value for each similar protein. The E-value is the estimated number of proteins in the database that would be as similar if the proteins in the database were random amino acid sequences [Altschul *et al.*, 1997].

Orthologous proteins have been predicted using phylogenetic trees on amino acid sequences [Storm and Sonnhammer, 2002; Datta *et al.*, 2009], and by analyzing graphs on proteins weighted by sequence similarity [Tatusov *et al.*, 2000; Jensen *et al.*, 2008]. Graph-based methods typically use pairwise BLAST scores as the primary input, and then cluster sequences using these scores [O'Brien *et al.*, 2005; Chen *et al.*, 2006; Tatusov *et al.*, 2003]. Graph-based methods have poor precision relative to tree-based methods for protein orthology prediction [Gabaldón, 2008].

Validation is often performed based on similarity of function [Jensen *et al.*, 2008, page D253], under the assumption that orthologous proteins are most likely to have similar functions.

1.3 Thesis Overview, Contributions, and Organization

1.3.1 Detecting Conserved Multiprotein Modularity

Detecting essential multiprotein modules that change infrequently during evolution is a challenging algorithmic task that is important for understanding the structure, function, and evolution of the biological cell. Conserved proteins are likely to be essential [Peng *et al.*, 2012], but they may be conserved due to their ease of incorporation into a variety of multiprotein modules. In many cases, multiprotein modules may be the evolutionary unit on which purifying selection acts.

1.3.1.1 Produles and Modularity

We define a measure of modularity for interactomes and present a linear-time algorithm, Produles, for detecting multiprotein modularity conserved during evolution that improves on the running time of previous algorithms for related problems and offers desirable theoretical guarantees. Through randomization experiments, we demonstrate that conserved modularity is a defining characteristic of interactomes. Computational experiments on current experimentally derived interactomes for *Homo sapiens* and *Drosophila melanogaster*, combining results across algorithms, show that nearly 10 percent of current interactome proteins participate in multiprotein modules with good evidence in the protein interaction data of being conserved between human and *Drosophila*.

Produles can also be applied with high quality protein orthology data to find orthologous multiprotein modules. Orthologous modules can be examined for evolutionary differences across species. PHOG orthologous proteins and current protein interaction data were used on a proteome-wide scale to detect conserved multiprotein modules in the interactomes for *Homo sapiens* and *Drosophila melanogaster*. We found 29 cohesive and separable modules that seem to be highly conserved. One of these, the TFIID general transcription factor required for eukaryotic transcription, was examined in depth. Evidence exists that the composition of TFIID differs slightly between the two species. This computational pipeline, consisting of high-quality methods for detecting orthologous proteins and orthologous multiprotein modules is generalizable and can be applied to the proteomes and interactomes

for any species, leading to useful insights on the extent and nature of conservation of orthologous multiprotein modules during evolution.

1.3.1.2 Algorithm Evaluation Measures

We present a biologically motivated graph theoretic set of evaluation measures complementary to previous evaluation measures, demonstrate that Produles exhibits good performance by all measures, and describe certain recurrent anomalies in the performance of previous algorithms that are not detected by previous measures. Consideration of the newly defined measures and algorithm performance on these measures leads to useful insights on the nature of interactomics data and the goals of previous and current algorithms.

1.3.1.3 Evaluation of Optimization Criteria

Biological process enrichment is a widely used metric for evaluating the quality of multiprotein modules. We examine possible optimization criteria for detecting homologous multiprotein modules and quantify their effects on biological process enrichment. We find that modularity, linear density, and module size are the most important criteria considered, complementary to each other, and that graph theoretic attributes account for 36 percent of the variance in biological process enrichment. Variations in protein interaction similarity within module pairs have only minor effects on biological process enrichment. As random modules increase in size, both biological process enrichment and modularity tend to improve, though modularity does not show this upward trend in modules with size at most 50 proteins. To adjust for these trends, we recommend a size correction based on random sampling of modules when using biological process enrichment or other attributes to evaluate module boundaries. Characteristics of homologous multiprotein modules optimized for each of the optimization criteria are examined.

1.3.1.4 Supporting Software

Cloud computing opens new possibilities for computational biologists. Given the pay-as-you-go model and the commodity hardware base, new tools for extensive parallelism are needed to make experimentation in the cloud an attractive option. We present EasyProt, a parallel message-passing architecture designed for developing experimental workflows in computational biology while harnessing the power of cloud resources. The system exploits parallelism in two ways: by multithreading modular components on virtual machines while respecting data dependencies and by allowing expansion across multiple virtual machines. Components of the system, called elements, are easily configured for efficient modification and testing of workflows during ever-changing experimentation. Though EasyProt, as an abstract cloud programming model, can be extended beyond computational biology, current development brings cloud computing to experimenters in this important discipline who are facing unprecedented data-processing challenges, with a type system designed for proteomics, interactomics and comparative genomics data, and a suite of elements that perform useful analysis tasks on biological data using cloud resources.

VieProt is a visualization tool implemented in Java for viewing conserved multiprotein modularity and associated statistics with a dynamic force-directed layout.

1.3.2 Protein Orthology Detection Using Protein Interactions

Protein interaction networks, annotated with estimates of the reliability of the experimental data supporting each interaction, provide a rich source of data for inferring functional orthology among proteins from disparate species. Modeling protein interaction network change across species provides a unique perspective on the forces underlying evolution.

1.3.2.1 Graphical Model for Protein Orthology

We design a graphical model for aligning protein interaction networks. As the resulting graphical model is likely to have high treewidth, exact inference using the junction tree algorithm is not likely to be tractable. We survey several methods for approximate inference that can be applied to this graphical model.

1.3.2.2 Structured Learning of Graphical Model Parameters

Netalign is a program we developed that uses protein interaction data to improve the identification of proteins that are functionally orthologous. We show that the detection of functionally orthologous proteins can be cast as a structured learning problem. The resulting optimization problems are intractable. Thus, we survey several methods for finding approximate solutions, focusing on those aspects most relevant to biological network alignment.

1.3.2.3 Whole Genome Probabilistic Model of Evolution

A central goal of computational biology is to use genomic data to detect orthologous proteins, that is, proteins in two separate species that derive from the same ancestral protein in the last common ancestor of those two species. This information can be used to annotate proteins in one species using experiments conducted in another. Orthologous proteins, even between widely divergent species, are often related in function. We present a probabilistic model of genome evolution that models not only changes in gene DNA sequences but also other changes in the genome: protein duplication and loss, changes in gene order on the chromosomes, and the conservation and loss of protein interactions and functional modules. This model is consistent with the known phenomenon of domain shuffling. A protein may be orthologous to numerous proteins that are themselves unrelated. The proposed method, Orthalign, computes an approximate maximum a posteriori estimate using parameters estimated from biological studies of protein families.

Chapter 2

Detecting Conserved Multiprotein Modularity

2.1 Local Network Alignment

A stream of scientific investigation has focused on conservation of modular structure of the cell, such as protein signaling pathways and multiprotein complexes, across species during evolution, with the premise that such structure can be described in terms of graph theoretic properties in the interactomes [Kelley *et al.*, 2003; Sharan *et al.*, 2005b; Koyutürk *et al.*, 2006b; Flannick *et al.*, 2006; Narayanan and Karp, 2007; Hodgkinson and Karp, 2012]. This stream of investigation has led to many successes, discovering conserved modularity across a wide range of evolutionary distances.

The first algorithms for detecting multiprotein modularity did not state this as their explicit aim [Kelley *et al.*, 2003; Sharan *et al.*, 2005b; Koyutürk *et al.*, 2006b], but rather they were called network alignment algorithms. The goal was to find regions of protein interaction networks that were conserved across species. Since the conserved regions of natural interest in protein interaction networks are groups of proteins that work together on particular tasks, the goal became to find conserved functional modules in the protein interaction networks. Optimization criteria for finding conserved functional modules ranged from finding dense regions [Sharan *et al.*, 2005b] to finding isomorphic subgraphs [Koyutürk *et al.*, 2006b].

2.2 Produles Algorithm

Produles is an algorithm designed to detect modular regions conserved during evolution. Produles runs in linear time in the size of the input and is efficient in practice while yielding exceptionally good results.

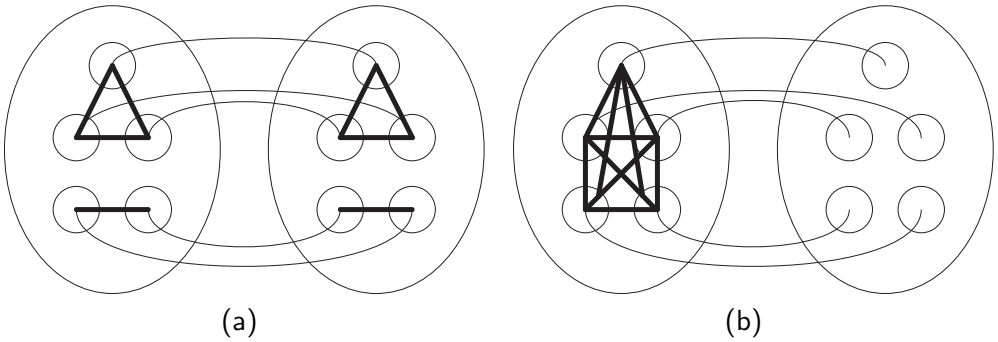


Figure 2.1: Diagrams illustrating difficulties with additivity across data types and species. Species are represented by ovals. Proteins are represented by circles. Protein interactions are represented by thick lines. Proteins with high sequence similarity are connected with thin lines. Algorithms that are additive across the interaction and sequence data may predict module (a) to be conserved due to high sequence similarity. In this case, the module boundaries are not well defined, most likely containing portions of multiple modules that have no relation with each other. Algorithms that are additive in the interactions across species may predict module (b) to be conserved though there is no evidence for module conservation across the species in the protein interaction data.

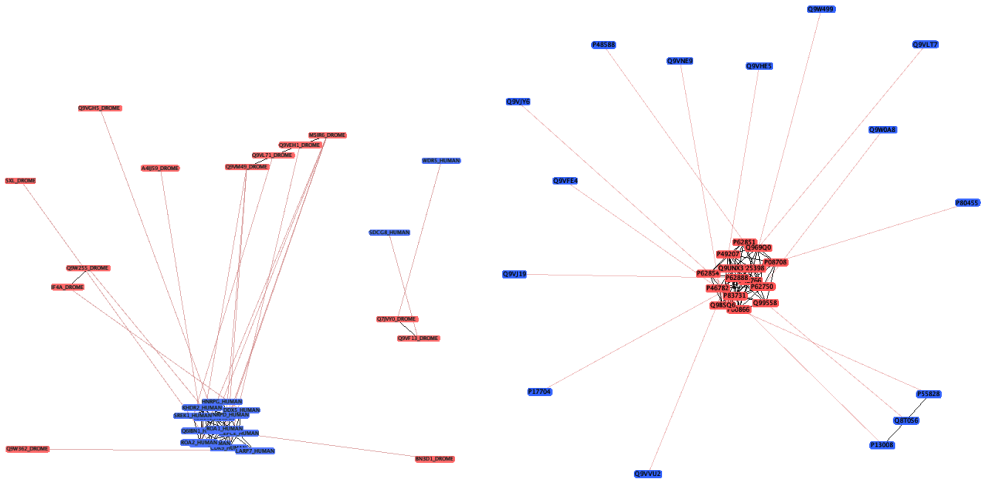
2.2.1 Motivation for Produles

Some previous algorithms for related problems, including NetworkBlast [Sharan *et al.*, 2005b] and Graemlin [Flannick *et al.*, 2006], use a scoring function that is a sum of multiple scores: one score based on protein sequence similarity, and one score from each species based on the density of interactions among the module proteins for that species. These algorithms use a greedy search on this scoring function to find conserved modules. Due to the additivity, module pairs similar to the diagrams in Fig. 2.2 may receive high scores and be reported as conserved. For example, the module pair shown in Fig. **??** was reported as conserved by NetworkBlast-M [Kalaev *et al.*, 2009] when applied to the iRefIndex [Razick *et al.*, 2008] data set in Section 3.3.2.

Good module boundaries are important for the modules that are returned by an algorithm. Fig 2.2 (a) illustrates the situation in which module boundaries may not be well defined as there is no evidence in the protein interaction data that the various components belong in the same module.

Evidence of conservation in the interaction data across species is essential for modules claimed by an algorithm to be conserved during evolution. Homologous proteins may be reorganized during evolution into multiprotein modules that differ both in composition and in function across species [Beltrao and Serrano, 2007]. Due to the additivity of the scoring function for some previous algorithms, including NetworkBlast and Graemlin, in the interaction densities across species, a very dense network in one species can be reported as conserved with homologous proteins in another species that have zero or few interactions among them. In this case, as illustrated in Fig. 2.2 (b), the interaction data does not support a claim of module conservation across the given species.

Produles is an important step to address these issues. Produles runs in linear time,



(a) (b)

Figure 2.2: Module pairs reported as conserved by NetworkBlast-M when applied to iRefIndex protein interaction data for human and Drosophila. Different colors denote proteins from different species. Black edges are protein interactions and pink edges are homology relationships. The module pair on the left contains portions of two modules. In the module pair on the right only two protein interactions exist between the Drosophila proteins.

scaling better than Match-and-Split [Narayanan and Karp, 2007] and MaWISh [Koyutürk *et al.*, 2006b], and does not exhibit the recurrent anomalies that result from the additivity of the scoring function across species and data sources that forms the basis for NetworkBlast and Graemlin.

2.2.1.1 Form of Study Data

An interactome is an undirected graph $G = (V, E)$, where V is a set of proteins and $(v_1, v_2) \in E$ if and only if protein v_1 is found to interact with protein v_2. In this study the input is restricted to a pair of interactomes, $G_i = (V_i, E_i)$, for $i \in \{1, 2\}$, and protein sequence similarity values, $h : V_1 \times V_2 \to \mathbb{R}^+$, defined only for the most sequence similar pairs of proteins appearing in the interactomes. In this study, h is derived from BLAST [Altschul *et al.*, 1990] E-values. As BLAST E-values change when the order of the interactomes is reversed, h is defined with the rule

$$h(v_1, v_2) = h(v_2, v_1) = \frac{E(v_1, v_2) + E(v_2, v_1)}{2}$$

where $E(v_1, v_2)$ is the minimum BLAST E-value for $v_1 \in V_1$, $v_2 \in V_2$ when v_1 is tested for homology against the database formed by V_2. An algorithm using this data as input is general to any pair of interactomes, including those for newly studied species.

2.2.2 Modularity

A modular system consists of parts organized in such a way that strong interactions occur within each group or module, but parts belonging to different modules interact only weakly [Simon, 2005]. Following this, a natural definition of multiprotein modularity recognizes that proteins within a module are more likely to interact with each other than to interact with proteins outside of the module. Let $G = (V, E)$ be an interactome. A multiprotein module is a set of proteins $M \subset V$ such that $|M| \ll |V|$ and M has a large value of

$$\mu(M) = \frac{|E(M)|}{|\texttt{cut}(M, V\backslash M)| + |E(M)|}$$

where $E(M)$ is the set of interactions with both interactants in M, and $\texttt{cut}(M, V\backslash M)$ is the set of interactions spanning M and $V\backslash M$. Of the interactions involving proteins in M, the fraction contained entirely within M is given by $\mu(M)$. This definition of modularity is similar to the recent definition of λ-module [Wang *et al.*, 2011].

The conductance of a set of vertices in a graph is defined as

$$\Phi(M) = \frac{|\texttt{cut}(M, V\backslash M)|}{|\texttt{cut}(M, V\backslash M)| + 2\min(|E(M)|, |E(V\backslash M)|)}.$$

When $|E(M)| \leqslant |E(V\backslash M)|$, as for all applications in this study,

$$\Phi(M) = \frac{|\texttt{cut}(M, V\backslash M)|}{|\texttt{cut}(M, V\backslash M)| + 2|E(M)|} = \frac{1 - \mu(M)}{1 + \mu(M)}.$$

Thus, when searching for relatively small modules in a large interactome, minimizing conductance is equivalent to maximizing modularity. This relationship allows us to modify powerful algorithms from theoretical computer science designed for minimizing conductance [Andersen *et al.*, 2007; Spielman and Teng, 2008]. It has previously been shown that conductance in protein interaction networks is negatively correlated with functional coherence [Voevodski *et al.*, 2009], in agreement with our findings in Section 3.3.

2.2.2.1 Modularity and Degree Bounds

Assuming we are searching for modules of size at most b with modularity at least d, the vertices in any such module have bounded degree. Let $\delta(u)$ be the degree of u in G.

Theorem 2.2.1 (Modularity-maximizing degree bound)**.** If $d > 0$, the objective function

in the optimization problem

$$\begin{array}{ll} \max\limits_{G,M,u} & \delta(u) \\ \text{s.t.} & u \in M \\ & |M| = b \\ & \mu(M) \geqslant d \\ & \mu(M) > \mu(M \backslash \{u\}) \end{array}$$

satisfies the bound $\delta(u) < (b-1)(1+d)/d$.

Proof. Let $M' \triangleq M \backslash \{u\}$. Let $y \triangleq |E(M')|$. Let $x \triangleq |\texttt{cut}(M', \{u\})|$.

$$\mu(M') = \frac{y}{|\texttt{cut}(M', V \backslash M')| + y} < \mu(M)$$

so

$$|\texttt{cut}(M', V \backslash M')| > \frac{y(1-\mu(M))}{\mu(M)}$$

Thus,

$$\begin{array}{rcl} \mu(M) & = & \dfrac{x+y}{[\delta(u)-x] + [|\texttt{cut}(M', V \backslash M')| - x] + [x+y]} \\ & < & \dfrac{x+y}{\delta(u) - x + y + \frac{y(1-\mu(M))}{\mu(M)}} \end{array}$$

which implies

$$\mu(M) < \frac{x}{\delta(u) - x}$$

As $\mu(M) \geqslant d$,

$$\delta(u) < \frac{x(1+d)}{d} \leqslant \frac{(b-1)(1+d)}{d}$$

□

The motivation for the restriction $\mu(M) > \mu(M \backslash \{u\})$ is that when searching for modules with high modularity, there can be proteins with such high degrees that it always improves the modularity to remove them from the module.

Theorem 2.2.2 (Tightness of degree bound)**.** If $d \leqslant \frac{b-2}{b}$, the bound in Theorem 2.2.1 is tight and neither requiring connectivity of M in the underlying graph nor requiring connectivity of $M \backslash \{u\}$ in the underlying graph can allow the bound to be further tightened.

Proof. Consider a module M of size b that induces a clique in an underlying graph G. Of these vertices, only two, u and v, are incident on edges that extend outside of the clique. Let

$$\begin{aligned}\delta(u) &= \frac{(b-1)(1+d)}{d} - \epsilon \\ \delta(v) &= \binom{b-1}{2}(\frac{1-d}{d}) + \epsilon'\end{aligned}$$

for $\epsilon \geqslant \epsilon' > 0$ chosen so that $\delta(u)$ and $\delta(v)$ are integers. To see that $\delta(v) > b-1$, implying that v can indeed be in the clique with at least one edge extending outside of the clique, examine the equivalent claim:

$$\binom{b-1}{2}(\frac{1-d}{d}) \geqslant b-1$$

which is equivalent to $d \leqslant \frac{b-2}{b}$ by expanding and solving for d. Then,

$$\begin{aligned}\mu(M) &= \frac{\binom{b}{2}}{\delta(u)+\delta(v)-2(b-1)+\binom{b}{2}} \\ &= \frac{db}{b+\frac{2d}{b-1}(\epsilon'-\epsilon)} \\ &\geqslant d\end{aligned}$$

and

$$\begin{aligned}\mu(M\backslash\{u\}) &= \frac{\binom{b-1}{2}}{\delta(v)-(b-2)+(b-2)+\binom{b-1}{2}} \\ &= \frac{d(b-1)(b-2)}{(b-1)(b-2)+2d\epsilon'} \\ &< d\end{aligned}$$

where in both cases the second equation follows from the first by multiplying numerator and denominator by $2d$ and simplifying.

It remains to show that ϵ can be taken arbitrarily small while maintaining integrality of $\delta(u)$ and $\delta(v)$. Let

$$\epsilon = \epsilon' = \frac{1}{b}$$

By rearrangement, we have

$$\begin{aligned}\delta(u) &= \frac{b-1}{d} + (b-1) - \frac{1}{b} \\ \delta(v) &= \frac{\binom{b-1}{2}}{d} - \binom{b-1}{2} + \frac{1}{b}\end{aligned}$$

so it suffices to show that

$$\begin{aligned}\delta'(u) &\triangleq \frac{b-1}{d} - \frac{1}{b} \\ \delta'(v) &\triangleq \frac{\binom{b-1}{2}}{d} + \frac{1}{b}\end{aligned}$$

are integers by choosing d appropriately. Let

$$d = \frac{\binom{b-1}{2}}{k - \frac{1}{b}}$$

for k integer and sufficiently large. Then,

$$\begin{aligned}\delta'(u) &= \frac{2k-1}{b-2} \\ \delta'(v) &= k\end{aligned}$$

are both integers for b odd if k is chosen so that $2k-1$ is a multiple of $b-2$. That is,

$$k = \frac{k'(b-2)+1}{2}$$

for b and k' sufficiently large positive integers with b odd and k' even. As $b \to \infty$, $\epsilon = \frac{1}{b} \to 0$. Finally, note that both M and $M \backslash \{u\}$ induce connected subgraphs in the underlying graph G, completing the proof. □

if $d > \frac{b-2}{b}$, which is unlikely to be the case for applications in this paper, a second bound

$$\delta(u) \leqslant \frac{\binom{b}{2}}{d} - \binom{b-1}{2},$$

that holds for all $d > 0$, is tight, and, similarly, cannot be further tightened by requiring connectivity of M or $M \backslash \{u\}$ in the underlying graph.

2.2.2.2 Hierarchy of Modularity

When an algorithm detects conserved multiprotein modules that share proteins, these modules can be combined into a larger composite module. Whenever this union takes place, the

following theorem shows that the modularity of the composite module is at least as large as the minimum modularity of the modules being combined.

Theorem 2.2.3 (Modularity minimum monotonicity)**.** For modules M_1, M_2, it is true that

$$\mu(M_1 \cup M_2) \geqslant \min\{\mu(M_1), \mu(M_2)\}.$$

Proof. Let $e(M) \triangleq |E(M)|$. Let $f(M, M') \triangleq |\texttt{cut}(M, M')|$. Let $g(M) \triangleq f(M, V \backslash M)$. Then

$$\begin{aligned}
\mu(M_1 \cup M_2) &= \frac{e(M_1 \cup M_2)}{g(M_1 \cup M_2) + e(M_1 \cup M_2)} \\
&= \tfrac{e(M_1)+e(M_2)+f(M_1,M_2)}{g(M_1)+g(M_2)-2f(M_1,M_2)+e(M_1)+e(M_2)+f(M_1,M_2)} \\
&= \frac{e(M_1) + e(M_2) + f(M_1, M_2)}{g(M_1) + g(M_2) + e(M_1) + e(M_2) - f(M_1, M_2)} \\
&\geqslant \frac{e(M_1) + e(M_2)}{g(M_1) + e(M_1) + g(M_2) + e(M_2)} \\
&\geqslant \min\{\frac{e(M_1)}{g(M_1) + e(M_1)}, \frac{e(M_2)}{g(M_2) + e(M_2)}\} \\
&= \min\{\mu(M_1), \mu(M_2)\}
\end{aligned}$$

The final inequality follows from the lemma

$$\frac{a+b}{c+d} \geqslant \min\{\frac{a}{c}, \frac{b}{d}\} \text{ for } a, b \geqslant 0 \text{ and } c, d > 0$$

which can be proved by observing that

$$\frac{a+b}{c+d} < \frac{a}{c} \Rightarrow c(a+b) < a(c+d) \Rightarrow bc < ad$$

whereas

$$\frac{a+b}{c+d} < \frac{b}{d} \Rightarrow d(a+b) < b(c+d) \Rightarrow ad < bc$$

which cannot both be true. □

A hierarchy of modularity consists of larger conserved modules composed of smaller conserved modules while maintaining a desired minimum modularity for all modules at all levels of the hierarchy.

2.2.3 Modularity Maximization Algorithms

2.2.3.1 Introduction to Previous Work

Graph conductance is a concept first introduced in 1988 by Jerrum and Sinclair [Jerrum and Sinclair, 1988]. It has been widely studied thereafter. A nice review that places conductance minimization algorithms in the context of graph clustering is [Schaeffer, 2007]. Finding a subgraph with lowest conductance is NP-hard [Sima and Schaeffer, 2005], proved earlier for the weighted case [Shi and Malik, 2000]. In [Kannan *et al.*, 2000], the goal is to find a clustering such that the overall conductance of each cluster is high and there is a minimum of edges spanning clusters. It is not clear why they did not directly minimize conductance of the clusters in the graph. Andersen and Lang [Andersen and Lang, 2008] study the minimum quotient cut that is a generalization of conductance, building on previous work. In particular, Arora, Rao, and Vazirani [Arora *et al.*, 2004] give a $O(\sqrt{\log n})$ approximation algorithm for the minimum quotient cut.

Nibble [Spielman and Teng, 2008] and PageRank-Nibble [Andersen *et al.*, 2006a] are algorithms for finding sets of vertices with low conductance in a graph. They were designed for solving symmetric diagonally dominant linear systems [Kelner *et al.*, 2013]. Reasonable adaptations of the algorithms allow them to run in constant time and to search only for small modules. The project of adapting Nibble and PageRank-Nibble to search only for small modules was initiated in [Voevodski *et al.*, 2009]. The adaptations described in [Voevodski *et al.*, 2009] do not guarantee constant running time for Nibble. A longer version of the original PageRank-Nibble paper for undirected graphs is [Andersen *et al.*, 2006b].

2.2.3.2 Nibble

Nibble [Spielman and Teng, 2008] is an algorithm for finding a set of vertices with low conductance in a graph G with n vertices. Let A be the adjacency matrix for G. Let D be a diagonal matrix with diagonal entries $D_{ii} = d(i)$ where $d(i)$ is the degree of vertex i in G. Let $W = (AD^{-1} + I)/2$ where I is the identity matrix. W is a lazy random walk transition matrix for G that with probability $1/2$ remains at the current vertex and with probability $1/2$ randomly walks to an adjacent vertex. Let q, r be vectors representing distributions on the vertices of G, not necessarily normalized. Define the truncation operator

$$[q]_\epsilon(u) = \begin{cases} q(u) \text{ if } \ q(u) \geqslant d(u)\epsilon \\ 0 \text{ otherwise} \end{cases}$$

Define the distribution that places all mass at vertex v

$$\chi_v(u) = \begin{cases} 1 \text{ if } \ u = v \\ 0 \text{ otherwise} \end{cases}$$

Each iteration of Nibble at time step t generates the vectors

$$q_t = \begin{cases} \chi_v \text{ if } t = 0 \\ W r_{t-1} \text{ otherwise} \end{cases}$$

$$r_t = [q_t]_\epsilon$$

Nibble is run for t_{last} iterations. After each iteration, the vertices are sorted by $q_t(\cdot)/d(\cdot)$. Let $S_j(q_t)$ be a set of j vertices with highest values of $q_t(\cdot)/d(\cdot)$ where ties are broken arbitrarily while maintaining $S_j(q_t) \subset S_{j+1}(q_t)$. These sets $S_j(q_t)$ are called sweep sets and there are always n of them. After each iteration the conductance is computed for at most the first b sweep sets for some constant b, never including any vertex v with $q_t(v) = 0$. No further sweep sets are considered if a vertex with degree greater than $b^2(d+1)/d$ is reached, where d is a constant parameter. This ensures that all vertices in modules returned by the algorithm have degrees bounded by $b^2(d+1)/d$, which is useful for reasons described in Section 2.2.6. The sweep set with minimum conductance over all iterations so far is stored. In original Nibble, b is not a constant but rather a function of the sum of degrees of vertices in the sweep sets, and there is no guarantee that vertices in returned modules have bounded degree.

It remains to show that the algorithm can be implemented to run in constant time for constant t_{last}, ϵ. Since t_{last} is constant, it suffices to show that a single iteration requires constant time. Let $\sigma(\cdot)$ be the support function that returns the set of vertices with positive values in its distribution argument. Define the volume of a set of vertices as the sum of degrees:

$$\text{vol}(S) = \sum_{v \in S} d(v)$$

Rather than computing $q_t = W r_{t-1}$ using matrix multiplication, q_t can be computed by explicitly passing messages to neighbors in the graph. Each vertex $v \in \sigma(r_{t-1})$ keeps half of $r_{t-1}(v)$ and partitions half of $r_{t-1}(v)$ equally among its neighbors. By keeping a linked list of references to vertices with nonzero distribution values, this requires $\text{vol}(\sigma(r_{t-1}))$ messages, leading to $|\sigma(q_t)| \leqslant \text{vol}(\sigma(r_{t-1}))$. The truncated distribution $[q_t]_\epsilon$ can be computed simply by removing references from the linked list for any vertex v such that $q_t(v) < d(v)\epsilon$. Only vertices with nonzero values of $q_t(\cdot)/d(\cdot)$ need be sorted. If the degree of each vertex is stored at the vertex, making degree lookup a constant-time operation, these vertices can be sorted in $\mathcal{O}(\text{vol}(\sigma(r_{t-1})) \log \text{vol}(\sigma(r_{t-1})))$ time. Conductances for the first b sweep sets can be computed in $\mathcal{O}(b^5(d+1) \log b/d)$ time by observing that the conductance of $S_j(q_t)$ can be computed by knowing the sum of degrees of vertices in $S_j(q_t)$ and the number of edges with both endpoints in $S_j(q_t)$. The former can be computed in $\mathcal{O}(|S_j(q_t)|) = \mathcal{O}(b)$ time, and the latter can be computed with at most $b * b^2(d+1)/d$ set inclusion tests in a balanced binary search tree of size at most b, which follows from the bound on the degree of each vertex in $S_j(q_t)$.

It remains to show that $\text{vol}(\sigma(r_{t-1}))$ never exceeds a constant value. For any $v \in \sigma(r_{t-1})$, by the truncation operation, $r_{t-1}(v) \geqslant d(v)\epsilon$. Because the distribution starts with $r_0 = [\chi_v]_\epsilon$

that has total value at most 1 and never increases in total value,

$$1 \geqslant \sum_{v \in \sigma(r_{t-1})} r_{t-1}(v) \geqslant \epsilon \sum_{v \in \sigma(r_{t-1})} d(v)$$

which implies

$$\mathrm{vol}(\sigma(r_{t-1})) \leqslant \frac{1}{\epsilon}$$

2.2.3.3 PageRank-Nibble

PageRank-Nibble [Andersen *et al.*, 2006a] is an algorithm based on PageRank [Page *et al.*, 1999] and Nibble [Spielman and Teng, 2008] for finding a module with low conductance in a graph $G = (V, E)$. Let A be the adjacency matrix for G. Let D be a diagonal matrix with diagonal entries $D_{ii} = \delta(i)$ where $\delta(i)$ is the degree of vertex i in G. Let $W = (AD^{-1} + I)/2$ where I is the identity matrix. W is a lazy random walk transition matrix that with probability 1/2 remains at the current vertex and with probability 1/2 randomly walks to an adjacent vertex. A PageRank vector is a row vector solution $\mathrm{pr}(\alpha, s)$ to the equation

$$\mathrm{pr}(\alpha, s) = \alpha s + (1 - \alpha)\mathrm{pr}(\alpha, s)W^T$$

where $\alpha \in (0, 1]$ is a teleportation constant, and s is a row vector distribution on the vertices of the graph called a preference vector. Define the distribution that places all mass at vertex v

$$\chi_v(u) = \begin{cases} 1 \text{ if } u = v \\ 0 \text{ otherwise} \end{cases}$$

When $s = \chi_v$, a PageRank vector is a weighted sum of the probability distributions obtained by taking a sequence of lazy random walk steps starting from v, where the weight placed on the distribution obtained after t walk steps decreases exponentially in t [Andersen *et al.*, 2007]. There is a unique PageRank vector since

$$\begin{aligned} p &= \alpha s + (1 - \alpha)pW^T \\ p[I - (1 - \alpha)W^T] &= \alpha s \\ p &= \alpha s[I - (1 - \alpha)W^T]^{-1} \end{aligned}$$

which follows as the matrix in brackets is strictly diagonally dominant and, thus, nonsingular.

Let p be a distribution on the vertices of G, and let the vertices be sorted in descending order by $p(v)/\delta(v)$, the frequency of v in distribution p normalized by the stationary distribution of an unrestricted random walk. Let $S_j(p)$ be the set of the first j vertices after sorting. For $j \in \{1, ..., |V|\}$, the set $S_j(p)$ is called a sweep set [Andersen *et al.*, 2007].

PageRank-Nibble consists of computing an approximate Page-Rank vector with $s = \chi_v$, defined as

$$\mathrm{apr}(\alpha, s, r) \triangleq \mathrm{pr}(\alpha, s) - \mathrm{pr}(\alpha, r)$$

where r is a residual vector defined below, and then returning the sweep set $S_j(\text{apr}(\alpha, \chi_v, r))$ with minimum conductance [Andersen *et al.*, 2007].

From the definition, if p is a vector that satisfies $p + \text{pr}(\alpha, r) = \text{pr}(\alpha, \chi_v)$, then $p = \text{apr}(\alpha, \chi_v, r)$. Thus, $0 = \text{apr}(\alpha, \chi_v, \chi_v)$. After initializing $p_1 = 0$, $r_1 = \chi_v$, the approximation $\text{apr}(\alpha, \chi_v, r)$ to $\text{pr}(\alpha, \chi_v)$ is improved iteratively. Each iteration, called a push operation, chooses an arbitrary vertex u such that $r_i(u)/\delta(u) \geqslant \epsilon$. Then $p_{i+1} = p_i$ and $r_{i+1} = r_i$ except for the following changes

1. $p_{i+1}(u) = p_i(u) + \alpha r_i(u)$
2. $r_{i+1}(u) = (1 - \alpha)r_i(u)/2$
3. $r_{i+1}(v) = r_i(v) + (1 - \alpha)r_i(u)/(2\delta(u))$ for each v such that $(u, v) \in E$

in which $\alpha r_i(u)$ probability is sent to $p_{i+1}(u)$, and the remaining $(1 - \alpha)r_i(u)$ probability is redistributed in r_{i+1} using a single lazy random walk step [Andersen *et al.*, 2007].

Each push operation maintains the invariant [Andersen *et al.*, 2006b]

$$p_i + \text{pr}(\alpha, r_i) = \text{pr}(\alpha, \chi_v)$$

When no additional pushes can be performed, the final residual vector r satisfies

$$\max_{u \in V} \frac{r(u)}{\delta(u)} < \epsilon$$

The running time for computing $\text{apr}(\alpha, \chi_v, r)$ is $O(1/(\epsilon\alpha))$ [Andersen *et al.*, 2007]. This follows directly from the claim that if T is the total number of push operations and d_i is the degree of the vertex pushed at the ith iteration, then

$$\sum_{i=1}^{T} d_i \leqslant \frac{1}{\epsilon\alpha}$$

To prove this claim, observe that the vertex u pushed at iteration i satisfies

$$r_i(u) \geqslant \epsilon d_i$$

As $\alpha r_i(u)$ probability is sent to $p_{i+1}(u)$,

$$\begin{aligned} ||r_{i+1}||_1 &= ||r_i||_1 - \alpha r_i(u) \\ &\leqslant ||r_i||_1 - \alpha\epsilon d_i \end{aligned}$$

Because the initial residual vector is $r_1 = \chi_v$ with $||\chi_v||_1 = 1$, the ℓ_1 norm of the residual vector cannot decrease by more than 1 over all iterations, so

$$\alpha\epsilon \sum_{i=1}^{T} d_i \leqslant 1$$

from which the claim follows.

If ϵ and α are set to constants, reasonable given their meanings, and if only the first b sweep sets are considered, the algorithm runs in constant time. Ensuring the degrees of the vertices satisfy the bound in Theorem 2.2.1, we do not consider sweep sets that contain vertices with degree $(b-1)(1+d)/d$ or greater, and we also require connectivity in the underlying graph.

For PageRank-Nibble, it is not necessary to bound the degree explicitly for each vertex in the considered sweep sets in order to guarantee that these degrees are bounded above by a constant. PageRank-Nibble guarantees that

$$\mathrm{vol}(\sigma(\mathrm{apr}(\alpha, \chi_v, r))) \leqslant \frac{2}{(1-\alpha)\epsilon}$$

This follows from the observation that the final push on a vertex $v \in \sigma(\mathrm{apr}(\alpha, \chi_v, r))$ occurred at an iteration i when $r_i(v) \geqslant \epsilon d(v)$ and a fraction $(1-\alpha)/2$ of that probability remained at $r_{i+1}(v)$. Thus, for each $v \in \sigma(\mathrm{apr}(\alpha, \chi_v, r))$,

$$r(v) \geqslant \frac{1-\alpha}{2} \cdot \epsilon d(v)$$

Thus

$$1 \geqslant \sum_{v \in \sigma(\mathrm{apr}(\alpha, \chi_v, r))} r(v) \geqslant \frac{(1-\alpha)\epsilon}{2} \cdot \mathrm{vol}(\sigma(\mathrm{apr}(\alpha, \chi_v, r)))$$

from which the claim follows.

2.2.3.4 Greedy Algorithm

To verify that Nibble and PageRank-Nibble return modules with near-optimal modularity, we use a greedy algorithm that grows a module by adding the neighboring protein that confers greatest improvement to the modularity. By considering only proteins that satisfy the degree bound from Theorem 2.2.1, the algorithm runs in time $O(b^3/d)$. Though this algorithm is slow, comparing its results with faster algorithms increases our confidence in the quality of their results.

2.2.4 Algorithm to Detect Conservation

The algorithm begins by finding a multiprotein module,

$$M \subset V_1$$

with high modularity in G_1 using a modularity maximization algorithm such as those described in Section 2.2.3. Let

$$\mathcal{H}_T(M) = \{v \mid \exists\ u \in M \text{ such that } h(u,v) \leqslant T\}$$

Modules corresponding to the connected components of the subgraph of G_2 induced by $\mathcal{H}_T(M)$ are candidates for conservation with M. Let these modules be $N_1, N_2, ..., N_k$. For $i = 1, ..., k$, let

$$\mathcal{R}_T(M, N_i) = \{u \in M \mid \exists\ v \in N_i \text{ such that } h(u, v) \leqslant T\}$$

If the following are true:

$$\begin{aligned} a &\leqslant |\mathcal{R}_T(M, N_i)| \leqslant b \\ a &\leqslant |N_i| \leqslant b \\ \frac{1}{c}|N_i| &\leqslant |\mathcal{R}_T(M, N_i)| \leqslant c|N_i| \\ \mu(\mathcal{R}_T(M, N_i)) &\geqslant d \\ \mu(N_i) &\geqslant d \end{aligned}$$

where a is a lower bound on size, b is an upper bound on size, c is a size balance parameter, and d is a lower bound on desired modularity, and if $\mathcal{R}_T(M, N_i)$ yields a connected induced subgraph of G_1, then we report the pair $(\mathcal{R}_T(M, N_i), N_i)$ as a conserved multiprotein module.

Each protein is used exactly once as a starting vertex for the modularity maximization algorithm. A counter is maintained for each protein in G_1. When a protein is placed in a module by the modularity maximization algorithm, the counter for the protein is incremented. Each counter has maximum value e for some constant e. The modularity maximization algorithm is restricted to search over proteins with counter value less than e. If a protein in G_1 is reported to be in a conserved module, the counter for the protein is set to $e/2$ in order to reduce module overlap. Furthermore, interactions in the subgraphs induced by the module are marked, preventing these interactions from being used in future searches by the modularity maximization algorithm. When all proteins in G_1 have been used as starting vertices, the roles of G_1 and G_2 are reversed, and the entire process is repeated.

2.2.5 Refinement of Large Connected Components

A module, $M \subset V_1$, may contain proteins that are homologous to a large number of proteins, $S \subseteq V_2$, and S may form a large connected induced subgraph in G_2. In many cases, the size of S cannot reasonably be explained by duplication of module proteins after divergence from the most recent common ancestor. Two reasons explain the majority of this phenomenon. First, proteins may share peripheral domains that cause protein homology detection algorithms to detect proteins only partially homologous, which may interact with module proteins, either genuinely or due to artifacts from experimental assays such as yeast two-hybrid. Second, paralogous modules that may be kept separate by the cell, performing different functions, contain homologous proteins, leading proteins in paralogous modules to be incorrectly detected as interacting. Refinement of large connected components aims to

remove partially homologous proteins and to separate paralogous modules.

2.2.5.1 Biconnected Components

A first approach is to consider biconnected components rather than connected components. If paralogous modules are connected only loosely by bridges, they can be separated using biconnected components.

2.2.5.2 Linear-time Colorful Connected Subgraph Heuristic

As a second approach, we design a heuristic algorithm that requires time linear in the size of the subgraph induced by S. The algorithm proceeds by iterations. At each iteration, each subgraph protein, $u \in S \subseteq V_2$, records the difference between the modularity of the subgraphs induced by S and $S \backslash \{u\}$. Each module protein, $v \in M \subset V_1$, is assigned a distinct color and transfers its color to all homologous proteins in S. Of the proteins in S with the most frequent color, half are removed, precisely those that individually benefit the modularity least. The algorithm iterates for a maximum of b^2 iterations, ensuring that a subgraph of size 2^b can be separated into modules of size at most b. After each iteration, tests are performed for connected components and biconnected components that can reasonably be reported as conserved according to the tests in Section 2.2.4.

2.2.6 Proof of Linear Running Time

Each value of $h(v, \cdot)$ for $v \in V$ is considered only when constructing $\mathcal{H}_T(M)$ for $\{M : v \in M\}$, so each value of $h(v, \cdot)$ is considered at most e times. If v is stored at each vertex in $\mathcal{H}_T(M)$ when constructing $\mathcal{H}_T(M)$, then constructing $\mathcal{R}_T(M, N_i)$ is a union of vertex lists and does not require additional considerations of $h(v, \cdot)$ values. As for all $v \in V_1$,

$$|\{M : v \in M\}| \leqslant e$$

the number of consideration of h values is

$$\begin{aligned} \sum_M \sum_{v \in M} |h(v, \cdot)| &= \sum_v \sum_{M : v \in M} |h(v, \cdot)| \\ &\leqslant e \sum_v |h(v, \cdot)| \\ &= e|h(\cdot, \cdot)| \end{aligned}$$

After finding $\mathcal{H}_T(M)$, it is necessary to compute $N_1, N_2, ..., N_k$. This can be problematic if any of the vertices in $\mathcal{H}_T(M)$ have large degree, which could conceivably be as large as $|V_2| - 1$. However, as we desire N_i such that $\mu(N_i) \geqslant d$ and $|N_i| \leqslant b$, which ideally do not contain any vertex u such that $\mu(N_i \backslash \{u\}) > \mu(N_i)$, we can discard, by Theorem 2.2.1, any vertex $v \in \mathcal{H}_T(M)$ with degree in G_2 of $(b-1)(1+d)/d$ or greater. A modified depth-first search that transitions only among vertices in $\mathcal{H}_T(M)$ is then used to compute

$N_1, N_2, ..., N_k$. This requires time

$$O((\frac{(b-1)(1+d)}{d})|\mathcal{H}_T(M)|) = O(|\mathcal{H}_T(M)|)$$

As

$$|\mathcal{H}_T(M)| \leqslant \sum_{v \in M} |h(v, \cdot)|$$

all of these depth-first searches over the full run of the algorithm require time

$$O(\sum_M |\mathcal{H}_T(M)|) = O(\sum_M \sum_{v \in M} |h(v, \cdot)|) = O(|h(\cdot, \cdot)|)$$

For a given M, constructing all $\mathcal{R}_T(M, N_i)$ by a union of lists stored at the vertices in the N_i requires time $O(\sum_i |N_i| b \log b) = O(|\mathcal{H}_T(M)|)$. Testing for connectivity of a single $\mathcal{R}_T(M, N_i)$ with a modified depth-first search that transitions only among vertices in $\mathcal{R}_T(M, N_i)$ requires constant time as $|\mathcal{R}_T(M, N_i)| \leqslant b$ and as each vertex in M has degree bounded by $(b-1)(1+d)/d$. All of these constructions and depth-first searches over the full run of the algorithm can be completed in time $O(\sum_M |\mathcal{H}_T(M)|) = O(|h(\cdot, \cdot)|)$.

Computing the modularity of module $U \in \{N_i, \mathcal{R}_T(M, N_i)\}$ requires computing the sum of degrees of the vertices in U and the number of edges with both endpoints in U. These can be computed in constant time when $|U| \leqslant b$ as each vertex in U has degree bounded by $(b-1)(1+d)/d$.

The refinement heuristic requires time $O(|\mathcal{H}_T(M)|)$ per iteration maintaining overall linear time. Attaining this time complexity requires using a linear-time median selection algorithm. A fast randomized median selection algorithm yields expected linear time whereas the median-of-medians algorithm [Blum *et al.*, 1973] ensures worst-case linear time. Computing biconnected components maintains the same time complexity as connected components by using a classic algorithm [Hopcroft and Tarjan, 1973].

2.2.7 Colorful Connected Subgraph and Variants

When we are willing to spend more than linear time for the overall algorithm, we can refine the connected components using an approximation algorithm for a variant of the colorful connected subgraph problem [Lacroix *et al.*, 2006]. The colorful connected subgraph problem is as follows: Given a set of colors, and a vertex-colored graph, find a subgraph that includes exactly one vertex of each color in the set [Lacroix *et al.*, 2006]. The colors are assigned to vertices as described in Section 2.2.5.2. The colorful connected subgraph problem is NP-complete [Lacroix *et al.*, 2006]. For an exact solution we could use one of the exponential-time algorithms from Torque [Bruckner *et al.*, 2010]. We present approximation algorithms and heuristics that are faster and work well in practice. Also the nature of the approximations is such that the output from the approximations may be more biologically reasonable than exact solutions.

2.2.7.1 Approximation Algorithm based on Steiner Tree

One approximation algorithm for the colorful connected subgraph problem is based on the Steiner tree problem and its 2-approximation algorithm [Vazirani, 2003]. Initially, we compute all-pairs shortest paths between each pair of proteins in the protein interaction subgraph. This can be done using the Floyd-Warshall algorithm or using a breadth-first search from each protein. The colors are then sorted in nondecreasing order by number of proteins per color to create an ordering $c_1, c_2, ..., c_k$. For each protein with color c_1, we create a solution using the subroutine in the next paragraph and then return the solution with fewest proteins.

Given a protein p_{c1} with color c_1, we add p_{c1} to a set of terminals T. We loop over each color c from c_2 to c_k. For each color c, we loop over each protein p of color c and compute a minimum spanning tree on $T \cup \{p\}$ using shortest path lengths as edge weights. The protein p_c that yields the minimum spanning tree of lowest total weight for color c is added to T before considering proteins of the next color. After all colors are considered, the proteins in T and the proteins in the shortest paths corresponding to each edge in the final minimum spanning tree are returned as a possible solution to the algorithm in the previous paragraph.

If there are n proteins and k colors, the total running time is $O(A + nkM)$ where A is the complexity of the all-pairs shortest paths algorithm and M is the complexity of the minimum spanning tree algorithm. The best implementation of the minimum spanning tree algorithm is Prim's algorithm with Fibonacci heaps, where $M = O(E + V \ln V) = O(\min\{n\delta_{PPI} + n \ln n, n^2\}$, and δ_{PPI} is the maximum degree in the protein interaction network. Using the Floyd-Warshall algorithm, $A = O(V^3) = O(n^3)$. Thus, the total running time is $O(n^3 + \min\{n^2 k\delta_{PPI} + n^2 k \ln n, n^3 k\})$.

This approximation algorithm guarantees that at most $(k-1)L$ proteins will be in the solution, where k is the number of colors and L is the number of proteins in the smallest solution. If there is a colorful connected subgraph, then $L \leqslant k$, but, when there is no colorful connected subgraph, the best solution may have $L > k$ with duplication of colors. To prove this approximation ratio, consider that since every protein of color c_1 is tested, we can assume we are starting from a protein p_{c1} in the smallest solution. The proof is by induction. For the base case, consider that since we assumed without loss of generality to start at a protein, p_1, of color c_1 in a smallest solution, there must be another protein in this same smallest solution, p_2, of color c_2 within distance L of p_1. For the inductive step, assume that the minimum spanning tree, MST_j, on a set of terminals, $p_1, p_2, ..., p_j$ of colors $c_1, c_2, ..., c_j$, respectively, has weight at most $(j-1)L$. Since p_1 is assumed to be in a smallest solution, there is a protein, p_{j+1}, of color c_{j+1} within distance L of p_1. By combining the edge representing the shortest path between p_1 and p_{j+1}, with MST_j, we form a spanning tree on $p_1, p_2, ..., p_j, p_{j+1}$, which is of weight at most $(j-1)L + L = jL$. The minimum spanning tree, MST_{j+1}, therefore, has weight at most jL.

This approximation guarantee can be tightened somewhat by observing that when replacing each weighted edge by the proteins in the shortest path in the proof, except for the first edge so replaced, all other edges need add one fewer protein than the weight as one

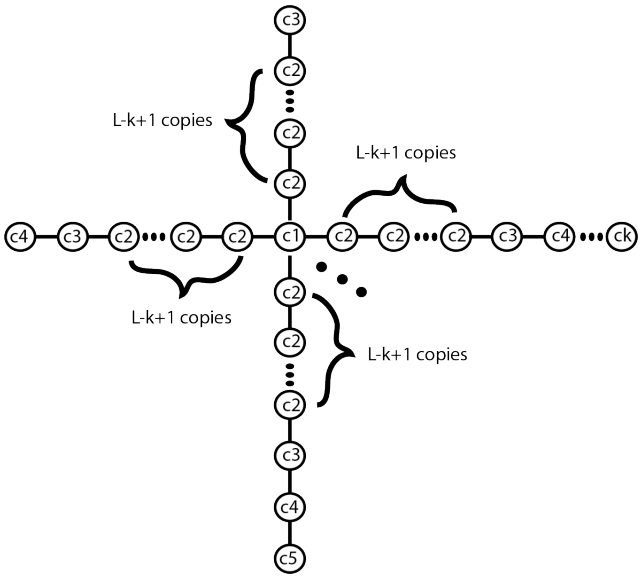


Figure 2.3: Example showing that the bound $(k-2)(L-2)+2$ on the approximate solution is asymptotically tight.

endpoint of the path is already in the solution. The number of proteins in the approximate solution is thus at most $(k-1)(L-1)+1$. Finally observe that in a smallest solution, a protein p_1 of color c_1 must exist that is adjacent to a protein with a different color, so one of the shortest paths in the proof must have length at most 2 rather than L. This yields an asymptotically tight bound on the approximate solution of $(k-2)(L-1)+2$.

To show that this bound is asymptotically tight, consider the example in Figure 2.3. Imagine that the colors are sorted in decreasing order of occurrence $c_1, c_2, ..., c_k$. To ensure that this is so, we can add proteins of colors $c_3, ..., c_{k-1}$ to the single protein with color c_k to ensure the correct relative order of occurrence among these, and a branch of c_1 proteins of arbitrary size to the central c_1 node to ensure that proteins with color c_1 are most abundant. Only the central c_1 protein shown is a viable starting protein. The rightmost branch with L proteins including the central protein is the smallest solution. However the algorithm may unfortunately use $k-2$ branches beginning at the topmost branch and moving counterclockwise. Thus, the algorithm returns a solution with $(k-2)(L-k+1)+1+\sum_3^k(k-2)$. As $L \to \infty$, both the number of nodes in this tight example and the upper bound approach $(k-2)L$, showing that the upper bound is asymptotically tight.

The approximation ratio is on the number of proteins, n, but any spanning tree on the proteins returned has $n-1$ edges, so the approximation ratio also applies to the number of protein interactions. An efficient final step to to construct a spanning tree using a breadth-first search in $O(n^2)$ time.

In our experiments we find that k is rarely more than 10. These are worst-case guarantees but in practice the approximation is generally much better. From experiment, we

found that when the algorithm was applied to 500 connected components containing all colors but too large to be reasonably reported as conserved, the output contained less than $3k$ proteins in all cases and less than $1.1k$ proteins in 95% of the cases.

2.2.7.2 Primal-Dual Approximation Algorithm

The primal-dual method on a linear relaxation of an integer program yields an algorithm with a similar running time of $O(n^3k)$ and a similar approximation ratio of $k-1$, where the approximation ratio is on the number of protein interactions in the solution. Let $G = (V, E)$ be the graph of interest in which it is desired to find a colorful subgraph, where V is the set of proteins and E is the set of protein interactions. Assign variables $x_e \in \{0, 1\}$ to each $e \in E$ where $x_e = 1$ if and only if e is part of the solution returned. For a set $S \subset V$ where $S \neq \varnothing$, let $\delta(S) = \texttt{cut}(S, V \backslash S)$. We restrict the colorful subgraph problem to require that the colorful subgraph includes a designated vertex r. By applying the resulting algorithm n times, once for each choice of r, the general problem can be solved. The following integer program models the colorful subgraph problem when it is constrained to include a designated vertex r.

$$\begin{array}{rll} \text{Min} & \sum_{e \in E} x_e & \\ \text{subject to:} & & \\ & \sum_{e \in \delta(S)} x_e \geqslant 1 & S \ : \ f(S) = 1 \\ & x_e \in \{0, 1\} & e \in E \end{array}$$

where $f(S) = \begin{cases} 1 \text{ if } \ r \in S \text{ and } S \text{ does not contain all colors} \\ 0 \text{ otherwise} \end{cases}$

The LP relaxation is

$$\begin{array}{rll} \text{Min} & \sum_{e \in E} x_e & \\ \text{subject to:} & & \\ & \sum_{e \in \delta(S)} x_e \geqslant 1 & S \ : \ f(S) = 1 \\ & x_e \geqslant 0 & e \in E \end{array}$$

Data: graph $G = (V, E)$, designated vertex $r \in V$
Result: set A of edges approximating colorful subgraph containing r
$A \leftarrow \varnothing$
$\mathcal{C} \leftarrow \{\{v\} : v \in V\}$
for $i \in V$ **do** $d(i) \leftarrow 0$
$\ell \leftarrow 0$
while component C_r containing r does not have all colors **do**
  $\ell \leftarrow \ell + 1$
  Find edge $e_\ell = (i, j)$ with $i \in C_r, j \in C_p \neq C_r$ minimizing $\epsilon = 1 - d(i) - d(j)$
  $A \leftarrow A \cup \{e_\ell\}$
  for $k \in C_r$ **do** $d(k) \leftarrow d(k) + \epsilon$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_r \cup C_p\} - \{C_r\} - \{C_p\}$
end
for $j \leftarrow \ell$ downto 1 **do**
  if all components C of $A - \{e_j\}$ satisfy $f(C) = 0$ **then** $A \leftarrow A - \{e_j\}$
end

Figure 2.4: Primal-dual $k - 1$ approximation algorithm for colorful subgraph containing designated vertex r.

and the dual of the LP relaxation is

$$\begin{array}{rll} \text{Max} & \sum_{S:f(S)=1} y_S & \\ \text{subject to:} & & \\ & \sum_{S:e\in\delta(S)} y_S \leqslant 1 & e \in E \\ & y_S \geqslant 0 & S \ : \ f(S) = 1 \end{array}$$

The function f is a 0-1 function that satisfies the maximality property: if A and B are disjoint, then $f(A) = f(B) = 0$ implies $f(A \cup B) = 0$. Thus we can use the primal-dual algorithm for integer programs of this form with functions satisfying the maximality property as given in [Goemans and Williamson, 1996], adapted to the colorful subgraph problem and displayed in Figure 2.4.

The algorithm has a simple combinatorial interpretation. Starting at r, the algorithm grows a connected component by including all proteins at distance 1, then all proteins at distance 2, and so on, until the connected component contains all colors. Then a shrinking phase begins where edges are removed in reverse order of addition if their removal does not cause the connected component containing r to lose any of its colors. The distance of the final protein in the growing phase is obviously a lower bound on the size of the final solution and this is the value of the objective function in the dual of the LP relaxation. This lower bound is used to prove the approximation ratio.

Essentially, as the algorithm proceeds, the only dual variables, y_S, that are set to nonzero values are the dual variables corresponding to sets S_r^i containing all proteins within distance i of protein r, for those values of i encountered during the growing process. These dual variables, y_S, are set to 1. Since each edge $e \in E$ can be in only of the cuts $\delta(S_r^i)$, the final setting of the dual variables is a dual feasible solution. Since, by LP duality theory, the objective function of the dual LP relaxation is a lower bound on the objective function of the LP relaxation, and since the objective function of the LP relaxation can be optimized to be at least as small as the optimized objective function of the integer program, the sum of the y_S variables, i.e. the distance from r reached in the growing process, is a lower bound on the number of edges in a solution, as was directly obvious.

During the shrinking phase, the edges in the cuts, $\delta(S_r^i)$ are considered for removal in decreasing order of i. S_r^i must contain at least one color, i.e. the color of r. By the pigeonhole principle, if more than $k-1$ edges are in $\delta(S_r^i)$, then at least one edge cannot lead to a connected component with a color needed by the colorful subgraph unreachable by the other edges, so it will be removed. Thus, for each i, at most $k-1$ edges are left in the cut $\delta(S_r^i)$. Since the largest value of i is a lower bound on the size of the optimal solution, the approximation ratio of $k-1$ immediately follows.

The growing and shrinking phases can be implemented in $O(n^2k)$ time using a breadth-first search. As proteins are encountered, global counters for the colors of the proteins are incremented. Pointers to edges are stored in a list in the order in which they are encountered. The shrinking phase decrements the global color counters for each edge removed. The complete algorithm, which loops over every choice of r to choose the best solution, thus, has a time complexity of $O(n^3k)$.

2.2.7.3 The Forcing Heuristic

In this heuristic algorithm for the colorful subgraph problem, the input is a module, M, and a subgraph, $G = (V, E)$, where M and V are sets of proteins and E is a set of protein interactions. Also given is a set of homology relationships between the proteins in M and the proteins in V. The proteins $c \in M$ are interpreted as colors. Initially each color, $c \in M$ is assigned a count set to 1, that says how many proteins homologous to c, i.e. with color c, should be included in the colorful subgraph. Each color $c \in M$ is also given a frequency that is the total number of proteins in V that are homologous to c, i.e. have color c. Each protein $p' \in V$ is assigned a color $c \in M$ that is homologous to p' and that has the lowest frequency of all colors in M homologous to p'. The frequencies of the colors $c \in M$ are then updated to be the number of proteins in V with color c. During this update, one or more colors, $c \in M$, may be assigned a frequency of 0. If this is the case, the count of c is set to 0, a protein $p' \in V$ homologous to c is selected, and the count of the color of p', i.e. the count of a different color in M, is incremented, provided this color's count is not made higher than its frequency. This completes the initialization.

The colors $c \in M$ are sorted from lowest frequency to highest frequency. The idea is that colors with lower frequency are more likely to be required in a colorful subgraph so we should begin by building a basic scaffold for the colorful subgraph from proteins $p' \in V$

that have low frequency colors. Colors with frequency 0 are not included in the sorted list. Starting from $G_0 = (\varnothing, \varnothing)$, for the sorted colors $i = 1, ..., k$, all proteins $p' \in V$ with color i are added to V_{i-1} to form V_i. All edges in E induced by proteins in V_i are added to E_i to form G_i. Connected components of G_i are computed and a statistic,

$$X_i = \sum_{\text{components } C} \frac{(\#\text{ colors in C})^3}{\#\text{ proteins in } C}$$

is evaluated. G_{max} is chosen to be G_i for the i that maximizes X_i.

Let $S_c = |\{p : \text{color}(p) = c\}|$. The components of G_{max} are sorted by size. Starting with the smallest components, components are removed if they do not cause $S_c < \text{count}(c)$ for any $c \in [c_1, ..., c_{max}]$.

The remaining components must be merged and, moreover, proteins of colors $c_{max+1}, ..., c_k$ must be added to the colorful subgraph. First we consider all individual proteins of colors c_i, for $i > max$, whose addition to the colorful subgraph merges two or more components. We add the protein that increases the modularity, μ, the most. This is repeated until all components are merged or until the remaining components cannot be merged by the addition of single proteins.

The next step is to perform a breadth-first search to find a shortest path on proteins with colors $i > max$ from the smallest component to a larger component. The shortest path is added to the colorful subgraph. This is repeated until all components are merged.

At this stage, it may be that some colors $i > max$ are insufficiently represented in the colorful subgraph. We consider each color $c = max + 1, ..., k$. If $S_c < \text{count}(c)$ and if there is at least one protein of color c connected to the subgraph, we select a protein from these that most increases the modularity, μ, and we add this protein to the subgraph. If $S_c < \text{count}(c)$, but there is no protein of color c connected to the subgraph, we add all proteins in a shortest path from the subgraph to a protein of color c. At this stage it is guaranteed that $S_c \geqslant \text{count}(c)$ for all colors.

Now we have a colorful subgraph but it may still be possible to remove some proteins while maintaining connectivity and maintaining $S_c \geqslant \text{count}(c)$ for all c. We perform this step using a heuristic that we call the forcing heuristic, since we identify the proteins that are forced into the colorful subgraph and cannot be removed. All proteins with colors c such that $S_c = \text{count}(c)$ are forced into the colorful subgraph. Also any cut vertex whose removal would split the subgraph into two or more components such that any component would have less than $\text{count}(c)$ of any color c are forced into the colorful subgraph. For every protein forced in, if there are more than $\text{count}(c)$ proteins of its color c, any other protein of color c that is not a cut vertex is removed, until exactly $\text{count}(c)$ proteins remain or until each remaining protein of color c is a cut vertex. This completes the heuristic and the remaining colorful subgraph is returned.

Using this heuristic rather than the heuristic in Section 2.2.5 to refine large connected components in Produles yielded similar results both in terms of quality of output and running time. Using this heuristic is given as an option in the Produles software in case one finds that the default heuristic from Section 2.2.5 is not providing the desired results

for particular inputs.

2.2.8 Produles Discussion and Extensions

2.2.8.1 Modularity Versus Density

Modularity prefers relatively dense regions that are separable from the rest of the graph, whereas density alone does not consider separability of dense regions. Many algorithms for related problems have been based on a search for dense subgraphs. A notable exception is the algorithm from a recent study that uses a definition of modularity similar to ours [Wang *et al.*, 2011].

2.2.8.2 Three or More Interactomes

Detecting multiprotein modularity conserved across three or more interactomes requires only minor modification and maintains running time linear in the size of the input. Note that the size of the input is quadratic in the number of interactomes. Modularity maximization algorithms are applied to each interactome to find natural modules. After computing connected components on homologous proteins in the other interactomes and refining the resulting subgraphs as described in Section 2.2.5, the original module and all refinements are reported if they pass the requirements in Section 2.2.4. The proof of running time in Section 2.2.6 extends without difficulty.

2.2.8.3 Weighted Interactomes

If desiring to focus on modules that preferentially incorporate particular interactions, weights can be assigned to the interactions. This has been used, for example, by NetworkBlast [Sharan *et al.*, 2005b], to focus on interactions among proteins that have been found not only to interact but also to be co-expressed. The definition of modularity in this study is easily extended to weighted interactomes by using weight sums rather than edge counts in the definition of μ. If the weights are bounded from below, which is usually the case due to thresholding, a variant of Theorem 2.2.1 for weighted graphs holds and the proof of linear running time follows.

Weights can also be used to count the number of independent experiments that report a given interaction. This is a form of multiple validation on the same proteins in the same species. A disadvantage of this approach is its implicit down-weighting of interactions in regions of newly-studied proteins that are frequently regions of greatest interest. Produles implicitly enforces multiple validation from independent experiments across species and across the various interactions in the module to increase confidence in a higher level signal of conserved modularity, making Produles ideal for noisy and newly-generated interactomics datasets.

2.2.8.4 Forcing Heuristic Extensions

The forcing heuristic in Section 2.2.7.3 works well in practice but, for some inputs, may return colorful subgraphs with more duplication of colors than necessary. The heuristic can, however, be invoked recursively. To allow a reasonable number of options for proteins of the most frequent colors to join connected components of proteins with less frequent colors, we allow the addition of any protein from the original subgraph when creating paths between connected components.

2.3 Produles on PHOG

We investigate whether orthology among proteins can be used to detect orthology among multiprotein modules. A similar investigation was conducted by Gandhi et al. [Gandhi *et al.*, 2006], using InParanoid [O'Brien *et al.*, 2005] for predictions of orthologous proteins. They used a restrictive method for conservation, requiring every interaction to agree in both interactomes, that may miss conserved modules due to the well-known difficulties of incomplete and imprecise protein interaction data [Hakes *et al.*, 2008]. Another line of research directed toward the detection of multiprotein modules include methods such as PathBlast [Kelley *et al.*, 2003], NetworkBlast [Sharan *et al.*, 2005b], MaWISh [Koyutürk *et al.*, 2006a], and Match-and-Split [Narayanan and Karp, 2007] that use general homology relations between proteins to guide the search.

In the experiments reported here we used the PHOG phylogenomic orthology prediction algorithm [Datta *et al.*, 2009] to identify orthologous proteins between *D. melanogaster* and *H. sapiens*. We applied Produles [Hodgkinson and Karp, 2012] with PHOG protein orthology data on the interactomes for these species, with parameters set for detection of highly conserved multiprotein modules, detecting 29 orthologous modules on more than 300 proteins in each species.

2.3.1 PHOG

PHOG [Datta *et al.*, 2009] uses pre-computed protein family trees in the PhyloFacts (`http://phylogenomics.berkeley.edu/phylofacts/`) [Glanville *et al.*, 2007; Krishnamurthy *et al.*, 2006] resource to predict orthologous proteins. PHOG bases its predictions on individual protein domains so its definition of protein orthology is not transitive. Rather than performing gene-tree species-tree reconciliation to predict orthologous proteins, PHOG can be parameterized for various levels of recall and precision. If high precision is desired, protein orthology predictions are restricted to those proteins that satisfy super-orthology [Zmasek and Eddy, 2002], where each node on the paths between the proteins in the family tree corresponds to a speciation event. If high recall is desired, a tree distance parameter extends protein orthology prediction to more distantly related proteins. PHOG-T(F) offers protein orthology predictions based on a tree-distance threshold that balances recall and precision for protein orthology prediction between *H. sapiens* and *D. melanogaster* [Datta *et al.*, 2009].

Results on a benchmark dataset of 100 non-homologous protein families from the TreeFam-A manually curated protein orthology database [Li *et al.*, 2006] show that PHOG provides a combination of high recall and precision competitive with OrthoMCL-DB [Chen *et al.*, 2006], with a dramatic improvement in precision over InParanoid [O'Brien *et al.*, 2005], and allows users to target different taxonomic distances and precision levels through the use of tree-distance thresholds (a variant of PHOG termed PHOG-T). For instance, OrthoMCL-DB achieved 76% recall and 66% precision on this dataset; at a slightly higher precision (68%) PHOG achieves 10% higher recall (86%). InParanoid achieved 87% recall at 24% precision on this dataset, while a PHOG variant designed for high recall achieves 88% recall at 61% precision, increasing precision by 37% over InParanoid [Datta *et al.*, 2009].

2.3.2 ModuleAlign for Gold Standard Set of Conserved Modules

It has been difficult to evaluate algorithms that detect conserved multiprotein modularity in interactomes due to the absence of a gold-standard data set. Algorithms cannot be expected to find modules that do not exist in the data sets being used, and the interactomics data is presently noisy and incomplete [Hakes *et al.*, 2008]. In this work a new protocol, named ModuleAlign, is designed to evaluate multiprotein module pairs predicted to be conserved by computational methods such as Produles. ModuleAlign can provide a gold standard for computational methods that detect conserved multiprotein modules, such as Produles, given fixed interactomes and protein orthology/homology relations.

The ModuleAlign protocol is as follows. First, a multiprotein module in one species is manually curated. This curation may be a refinement of a module found using a computational method or it may be guided by one of many modules stored in protein interaction databases such as the NetPath [Kandasamy *et al.*, 2010] modules from HPRD [Prasad *et al.*, 2009]. Next, all proteins orthologous to those in the manually curated module are placed into a set S. The set S is refined to a set S' by removing proteins that do not participate in at least one interaction with other proteins in S. Next, the subgraph of the interactome induced by proteins in S' is computed and displayed for interpretation.

Chapter 3

Evaluation of Algorithms

3.1 Biologically Motivated Evaluation Measures Based on Graph Theory

In this section, we present a set of biologically motivated graph theoretic measures that illuminate the characteristics and goals of various algorithms and qualities of the interactomics data. Five goals arising from these measures are presented with comments on when they may not be attained.

3.1.1 Output and Coverage

The algorithms return output that can be expressed as follows:

Algorithm output *Let k pairs of conserved modules returned by an algorithm be*

$$\mathcal{M} = \{(M_1^i, M_2^i) \mid i \in \{1, ..., k\}\}.$$

Let $(M_1, M_2) \in \mathcal{M}$. Let $M \in \{M_1, M_2\}$.

Proteome coverage *Let*

$$\mathcal{C}_i = |\mathcal{U}_i|/|V_i|,$$

where $\mathcal{U}_i$ is the set of proteins from V_i that are part of conserved modules. Let

$$\mathcal{C} = (\mathcal{C}_1 + \mathcal{C}_2)/2.$$

Module size *Let $S(M) = |M|$.*

3.1.2 Overlap

Algorithms differ in the amount and nature of module pair overlap allowed in the algorithm output. Three measures of overlap illuminate these characteristics.

Maximum overlap *Let*

$$\mathcal{O}_k^{ji} = |M_k^j \cap M_k^i| / |M_k^i|.$$

Let

$$\mathcal{O}^i_{max}(M_1^i, M_2^i) = \max_{j \neq i} \min\{\mathcal{O}_1^{ji}, \mathcal{O}_2^{ji}\}.$$

A value of $\mathcal{O}^i_{max} = x$ implies that no module pair $j \neq i$ exists that covers more than fraction x of each module in module pair i.

Sum overlap *Let*

$$\mathcal{O}^i_{sum}(M_1^i, M_2^i) = \sum_{j \neq i} \min\{\mathcal{O}_1^{ji}, \mathcal{O}_2^{ji}\}.$$

Cardinality overlap *Let*

$$\mathcal{O}^i_{card}(M_1^i, M_2^i) = |\{j : j \neq i \ \wedge \ \min\{\mathcal{O}_1^{ji}, \mathcal{O}_2^{ji}\} > 0\}|.$$

Together, $\mathcal{O}^i_{sum}$ and $\mathcal{O}^i_{card}$ measure the extent of overlap in the algorithm output, and $\mathcal{O}^i_{max}$ measures a limiting case. All three measures allow for module duplication during evolution.

Goal 1 (Reasonable coverage and overlap)**.** k and $\mathcal{C}$ should be in reasonable ranges with low average values of $\mathcal{O}^i_{max}$, $\mathcal{O}^i_{sum}$, and $\mathcal{O}^i_{card}$ in a set of conserved modules with reasonable coverage and overlap.

3.1.3 Evidence for Claim of Conservation

These measures address the situation diagrammed in Fig **??** (b).

Filled module *Let*

$$G_{int}(M) = (M, E(M)).$$

Interaction components *Let $C(M)$ be the number of connected components in $G_{int}(M)$.*

Module density *Let*

$$\Delta(M) = |E(M)| / \binom{|M|}{2}.$$

Module average *Let*

$$f_a(M_1, M_2) = (f(M_1) + f(M_2))/2,$$

where $f \in \{\mu, S, \Delta, C\}$.

Module difference *Let*

$$f_d(M_1, M_2) = |f(M_1) - f(M_2)|,$$

where $f \in \{\mu, S, \Delta, C\}$.

Goal 2 (Components and balance)**.** C_d, C_a, and Δ_d should be reasonably low to provide evidence for the claim of conservation across species. This may be problematic for algorithms based on models that are additive in the interaction densities across species.

3.1.4 Ancestral Multiprotein Modules

By grouping homologous proteins, this measure focuses on the number of sequence dissimilar proteins that participate in the module, presumably proteins with diverse origins and functions.

Module homology graph *Let*

$$G_{hom}(M_1, M_2) = (M_1 \cup M_2, H(M)),$$

where, for $p_1 \in M_1$, $p_2 \in M_2$, $(p_1, p_2) \in H(M)$ *iff* $h(p_1, p_2)$ *is defined.*

Ancestral protein *Let*

$$p = (P_1, P_2),$$

where $P_1 \subseteq M_1, P_2 \subseteq M_2$, *and* $G_{hom}(P_1, P_2)$ *is a connected component of* $G_{hom}(M_1, M_2)$.

Ancestral module *Let* $M_a(M_1, M_2)$ *be the set of ancestral proteins for* (M_1, M_2). *The arguments,* M_1, M_2, *may be omitted for brevity when the context is clear.*

Goal 3 (Number of ancestral proteins)**.** $|M_a|$ is a measure of the number of sequence dissimilar proteins and should be reasonably large for multiprotein modules containing proteins with diverse origins and functions.

3.1.5 Interaction Level Model of Evolution

This collection of measures examines agreement of conserved modules with an interaction level evolutionary model that includes interaction formation and divergence, protein duplication and divergence, and protein loss.

Relationship disagreement *Let $p, q \in M_a$, where $p = (P_1, P_2)$, $q = (Q_1, Q_2)$. For $i, j \in \{1, 2\}$, relationship disagreement means there is an interaction in G_i between some $p' \in P_i$ and some $q' \in Q_i$, but no interaction in G_j between any $p'' \in P_j$ with any $q'' \in Q_j$. Let $\mathcal{R}(M_1, M_2)$ be the number of relationship disagreements.*

Relationship evolution *Let*

$$E_r(M_1, M_2) = \mathcal{R}(M_1, M_2) / \binom{|M_a|}{2},$$

the fraction of possible relationship disagreements.

Ancestral module projection *For $i \in \{1, 2\}$, let*

$$\pi_i(M_a) = \{P_i \mid (P_1, P_2) \in M_a \wedge P_i \neq \varnothing\}.$$

Number of protein duplications *Let*

$$\mathcal{D}(M_1, M_2) = |M_1| - |\pi_1(M_a)| + |M_2| - |\pi_2(M_a)|.$$

Protein duplication evolution *Let*

$$E_d(M_1, M_2) = \mathcal{D}(M_1, M_2) / (|M_1| + |M_2| - 2),$$

the fraction of possible protein duplications.

Number of protein losses *Let*

$$\mathcal{L}(M_1, M_2) = 2|M_a| - |\pi_1(M_a)| - |\pi_2(M_a)|.$$

Protein loss evolution *Let*

$$E_\ell(M_1, M_2) = \mathcal{L}(M_1, M_2)/(|M_2| + |M_1|),$$

the fraction of possible protein losses.

Goal 4 (Interaction level evolutionary signal)**.** E_r, E_d, and E_ℓ should be reasonably low to detect an interaction level signal of evolutionary conservation. Unfortunately, conserved natural modules, even those that have been highly studied, may not satisfy this goal in current interactomics datasets due to artifacts in the way the data is collected and stored, leading to large amounts of noise at the individual interaction level. Furthermore, it has been proposed that there is not much selective pressure at the individual interaction level [Beltrao and Serrano, 2007], which, if true, may lead to genuinely high E_r scores in natural conserved modules. The high value of E_d in the conserved modules found by numerous algorithms, as shown in Section 5.1, may reflect characteristics of modules in interactomes, such as a tendency to contain homologous domains that facilitate protein interactions in the modules [Beltrao and Serrano, 2007].

3.1.6 Quality of Module Boundaries

This measure addresses the situation diagrammed in Fig 2.2 (a).

Ancestral protein projection *For ancestral protein $p = (P_1, P_2)$, P_i is the projection of p on M_i for $i \in \{1, 2\}$.*

Ancestral components *Let $C(M_a)$ be the number of connected components in a graph with vertex set M_a, where an edge is defined between two ancestral proteins $p, q \in M_a$ if any protein in the projection of p on M_i interacts with any protein in the projection of q on M_i, for some $i \in \{1, 2\}$.*

Goal 5 (Number of ancestral components)**.** Any value of $C(M_a) > 1$ implies that the module pair is not well defined as there is no evidence that the various connected components belong in the same module.

3.1.7 E_r Scores at Random

Let M_1 and M_2 be modules with densities $\Delta(M_1)$ and $\Delta(M_2)$ generated uniformly at random so that the probability of an edge between any two proteins $p, q \in M_i$ is $\Delta(M_i)$. The average size of an ancestral protein is $(|M_1| + |M_2|)/|M_a|$. Let $x = (|M_1| + |M_2|)/(2|M_a|)$. Suppose for simplicity that all ancestral proteins have size $2x$ with x proteins from each interactome

Table 3.1: E_r Scores Expected at Random

x/Δ	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1	0.00	**0.18**	**0.32**	**0.42**	**0.48**	**0.50**	**0.48**	**0.42**	**0.32**	**0.18**	0.00
2	0.00	**0.45**	**0.48**	**0.36**	**0.23**	**0.12**	**0.05**	**0.02**	0.00	0.00	0.00
3	0.00	**0.47**	**0.23**	**0.08**	**0.02**	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	**0.30**	**0.05**	**0.01**	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	**0.13**	**0.01**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	**0.04**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	**0.01**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

and that $\Delta = \Delta(M_1) = \Delta(M_2)$. Then, for $p, q \in M_a$, the probability of an interactome having no interaction between p and q is $(1-\Delta)^{x^2}$. The probability of neither interactome having an interaction between p and q is $((1-\Delta)^{x^2})^2$. The probability of an interactome having an interaction between p and q is $1-(1-\Delta)^{x^2}$ and the probability of both interactomes having an interaction between p and q is $(1-(1-\Delta)^{x^2})^2$. Thus, the probability of a disagreement in the relationship between p and q is $g(x) = 1-((1-\Delta)^{x^2})^2-(1-(1-\Delta)^{x^2})^2$, which is also equal to the expected value of E_r as $E(E_r) = \frac{E(\mathcal{R})}{\mathcal{R}_t} = \frac{g(x)\mathcal{R}_t}{\mathcal{R}_t} = g(x)$ where $\mathcal{R}_t = \binom{|M_a|}{2}$. Table 3.1 lists $E(E_r) = g(x)$ for various values of x and Δ. Nonzero entries are in bold. For all algorithms in this study, there are no protein losses, so $x \geqslant 1$.

3.2 Evaluation Measures Based on Protein Function

Gene Ontology [Gene Ontology Consortium, 2012] stores a database of protein annotations including annotations for participation in biological processes. Methods for biological process enrichment attempt to determine whether the module proteins have more similarity in biological process annotations than would be expected by chance. The most widely used method for biological process enrichment calculates the probability of obtaining at least as many proteins with the observed annotations if the module proteins were selected at random from a background set of proteins [Boyle *et al.*, 2004].

In the basic model for Gene Ontology enrichment [Boyle *et al.*, 2004], a P-value is computed for each Gene Ontology annotation in a module using a hypergeometric distribution as follows:

$$P = 1 - \sum_{i=0}^{k-1} \frac{\binom{m}{i}\binom{N-M}{n-i}}{\binom{N}{i}}$$

where M is the number of proteins in the background distribution annotated with the annotation being tested, N is the total number of proteins in the background distribution, n is the number of proteins in the module, and k is the number of proteins in the module annotated with the annotation being tested. The P-values for the various annotations are combined with Bonferroni or FDR correction for multiple testing. Two or three proteins with similar annotations in a module can lead to rejection of this null model with high confidence. Protein interactions tend to connect proteins with similar functions so any connected subgraph in a protein interaction network is likely to reject the null model with high confidence.

Several variations on the basic Gene Ontology enrichment algorithms have been introduced, for example to consider the graphical structure of the ontology [Grossmann *et al.*, 2007]. Biological process enrichment algorithms have been implemented in Ontologizer [Bauer *et al.*, 2008].

3.3 Empirical Evaluation

Through computational experiments on current experimentally derived interactomes for *Homo sapiens* and *Drosophila melanogaster*, we find good evidence that nearly 10 percent of the interactome proteins participate in multiprotein modules that have been conserved across this evolutionary distance, and we demonstrate significance of these results.

3.3.1 Evaluation of Produles Variants

Nibble and PageRank-Nibble were used as subroutines for Produles to compare their performances. Empirical results are given in Tables 3.2 and 3.3.

Both variants were applied to iRefIndex [Razick *et al.*, 2008] interactomes for H. sapiens and D. melanogaster, Release 6.0, filtered to retain binary interactions with UniProtKB [UniProt Consortium, 2012] identifiers. The resulting networks consist of 74,554 interactions on 13,065 proteins for H. sapiens and 40,004 interactions on 10,050 proteins for D. melanogaster. Protein amino acid sequences were obtained from UniProtKB. The blastall program from stand-alone NCBI BLAST [Sayers *et al.*, 2009] was applied in both directions with threshold 10^{-9} on E-values yielding 138,824 pairs of homologous proteins that passed the threshold in both directions. The values that were retained were the averages of the two E-values in both directions. Using this threshold none of the data is expected to be spurious as the total number of comparisons is $2 * (13,065 * 10,050) < 4 * 10^8$. Conserved modules are taken as induced subgraphs. The subroutines based on Nibble and PageRank-Nibble, as described in Section 2.2.3, are modified to consider only connected sweep sets with at most 20 proteins. All experiments were conducted using a MacBook Pro with 2.53 GHz Intel Core i5 processor and 4GB 1067 MHz DDR3 memory running Mac OS X Version 10.6.4. Best entries are in bold.
Parameters are

- All variants: $a = 5, \quad b = 20, \quad c = 1.5, \quad e = 50$

Produles-N	$d = 0.05$	$d = 0.06$	$d = 0.07$	$d = 0.08$	$d = 0.09$	$d = 0.10$
Running time	4m	4m	4m	4m	4m	4m
k	169	151	138	118	101	89
μ_a	(0.13,0.04)	(0.14,0.04)	(0.14,0.04)	(0.15,0.03)	**(0.16,0.03)**	**(0.16,0.03)**
μ_d	(0.08,0.06)	(0.08,0.06)	(0.08,0.06)	(0.08,0.05)	**(0.08,0.05)**	**(0.07,0.05)**
S_a	(7.93,2.86)	(7.93,2.74)	(7.91,2.74)	(7.92,2.90)	(7.90,2.87)	(7.84,2.83)
S_d	(1.56,1.34)	(1.52,1.33)	(1.51,1.28)	(1.58,1.31)	(1.60,1.30)	(1.60,1.23)
Δ_a	(0.32,0.10)	(0.32,0.10)	(0.33,0.10)	(0.33,0.11)	(0.32,0.09)	(0.32,0.09)
Δ_d	(0.07,0.09)	(0.07,0.09)	(0.07,0.09)	(0.08,0.10)	(0.07,0.07)	(0.07,0.07)
C_a	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	0.10	0.09	0.08	0.07	0.06	0.05
$\mathcal{O}$	**(0.28,0.33)**	**(0.28,0.34)**	(0.31,0.35)	(0.29,0.36)	(0.29,0.37)	(0.32,0.37)
$\lvert M_a \rvert$	(5.75,2.43)	(5.87,2.50)	(5.82,2.46)	(5.80,2.61)	**(5.91,2.44)**	(5.81,2.35)
$C(M_a)$	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)
E_r	(0.05,0.13)	(0.05,0.12)	(0.05,0.12)	**(0.04,0.12)**	**(0.03,0.10)**	**(0.04,0.11)**
E_d	(0.30,0.26)	(0.29,0.26)	(0.29,0.26)	(0.30,0.26)	**(0.28,0.24)**	**(0.28,0.25)**
E_ℓ	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)

Table 3.2: Produles-N with varying values of d

Produles-P	$d = 0.05$	$d = 0.06$	$d = 0.07$	$d = 0.08$	$d = 0.09$	$d = 0.10$
Running time	**3m**	**2m**	**2m**	**2m**	**3m**	**2m**
k	**248**	**217**	179	160	129	107
μ_a	(0.13,0.04)	(0.13,0.04)	(0.14,0.04)	(0.14,0.04)	(0.15,0.04)	**(0.16,0.04)**
μ_d	(0.08,0.06)	(0.07,0.06)	(0.07,0.06)	(0.07,0.05)	(0.07,0.06)	**(0.07,0.06)**
S_a	(7.61,2.62)	(7.61,2.68)	(7.70,2.73)	(7.65,2.81)	(7.75,2.87)	(7.88,3.06)
S_d	(1.39,1.27)	(1.40,1.28)	(1.42,1.25)	(1.38,1.19)	(1.49,1.25)	(1.54,1.33)
Δ_a	(0.34,0.10)	(0.34,0.10)	(0.34,0.10)	(0.34,0.10)	(0.33,0.09)	(0.33,0.09)
Δ_d	(0.08,0.10)	(0.08,0.11)	(0.08,0.11)	(0.08,0.11)	(0.08,0.10)	(0.08,0.10)
C_a	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	**0.13**	0.11	0.10	0.09	0.07	0.06
$\mathcal{O}$	(0.34,0.32)	(0.34,0.32)	(0.32,0.33)	(0.33,0.35)	(0.32,0.36)	(0.33,0.37)
$\lvert M_a \rvert$	(5.43,2.19)	(5.48,2.21)	(5.49,2.23)	(5.48,2.34)	(5.67,2.24)	(5.72,2.28)
$C(M_a)$	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)	(1.00,0.00)
E_r	(0.06,0.14)	(0.05,0.14)	(0.05,0.14)	(0.05,0.14)	**(0.04,0.12)**	(0.05,0.13)
E_d	(0.31,0.27)	(0.31,0.27)	(0.31,0.27)	(0.31,0.27)	(0.29,0.25)	**(0.28,0.25)**
E_ℓ	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)

Table 3.3: Produles-P with varying values of d

- P variants: $\alpha = 10^{-3}$, $\epsilon = 10^{-5}$
- N variants: $t_{last} = 10^{3}$, $\epsilon = 10^{-5}$

3.3.2 Comparison of Produles with Previous Algorithms

Produles, NetworkBlast-M [Kalaev *et al.*, 2009], Match-and-Split [Narayanan and Karp, 2007], and MaWISh [Koyutürk *et al.*, 2006b] were applied to iRefIndex [Razick *et al.*, 2008] binary interactions, Release 8.0, for *Homo sapiens* and *Drosophila melanogaster*, filtered to remove computationally predicted interactions and mapped to UniProtKB identifiers to remove isoforms. The interactomes consisted of 69,574 interactions on 12,652 proteins for *H. sapiens* and 38,699 interactions on 9,759 proteins for *D. melanogaster*. All programs were run with varying h threshold, corresponding to varying numbers of homologous protein pairs: $h = 10^{-80}$: 5,730 pairs, $h = 10^{-30}$: 29,598 pairs, $h = 10^{-20}$: 54,012 pairs, and $h = 10^{-9}$: 115,709 pairs. Because of the large number of module pairs returned by NetworkBlast-M, as shown in Fig. 3.1, all of which were assigned NetworkBlast-M quality scores, the set of modules for each h-threshold with highest NetworkBlast-M quality scores of the same size as the set returned by Produles was extracted and included in the comparisons.

The evaluation was performed on the module pairs returned with 7-40 proteins per species. This removes a significant fraction of the output from Match-and-Split and MaWISh that consists of subgraphs with two or three proteins, single edges or triangles, and it removes four large module pairs from MaWISh at threshold $h = 10^{-30}$ with modules of size up to 78 proteins. This has little effect on NetworkBlast-M for which more than 99% of its modules have 7-15 proteins per species. Restricting the analysis to this size range allows meaningful comparison of the algorithms according to the various evaluation measures without the statistics being affected by very large or very small modules.

Graemlin has nineteen network-specific parameters over a wide range of values, and together with the authors of Graemlin, we were unable to find settings that would yield results for the networks in this study. Graemlin 2.0 [Flannick *et al.*, 2009] was designed to address the parameter choice problem faced by Graemlin but requires data outside the scope of this study and has issues with usability.

3.3.3 Detailed Evaluation of Algorithms

In Fig. 3.1, the linear running time of Produles is seen to be very desirable. Match-and-Split could not complete on the dataset with 29,598 homologous protein pairs and MaWISh could not complete on the data set with 54,012 homologous protein pairs after running for more than twelve hours. Fig. 3.2 shows that after restricting the output to modules with 7-40 proteins per species, the average sizes of modules from all algorithms are similar, though NetworkBlast-M has less variance in its size distribution than the other algorithms.

GO biological process enrichment was computed for all modules, separately for each species, using Ontologizer [Bauer *et al.*, 2008] with Bonferroni correction for multiple hypothesis testing at 0.05 significance level. Fig. 3.3 shows that all algorithms perform

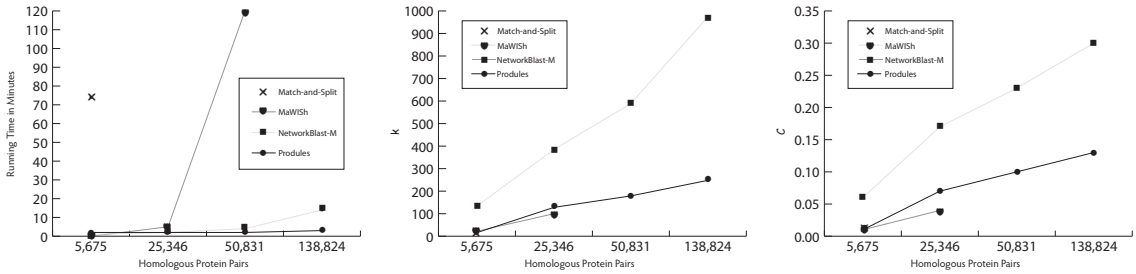


Figure 3.1: Comparison of running time and basic characteristics. The x-axis is the number of homologous protein pairs. The y-axis, from left to right, is the running time, the number of modules k, and the coverage $\mathcal{C}$.

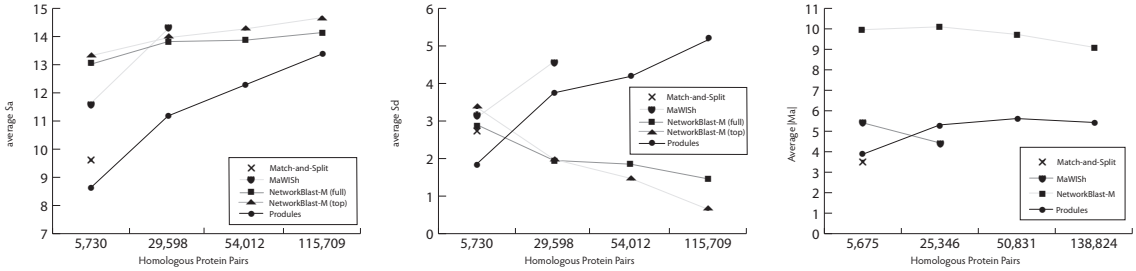


Figure 3.2: Comparison of module sizes. The x-axis is as in Fig. 3.1. The y-axis, from left to right, is average S_a, average S_d, and average $|M_a|$.

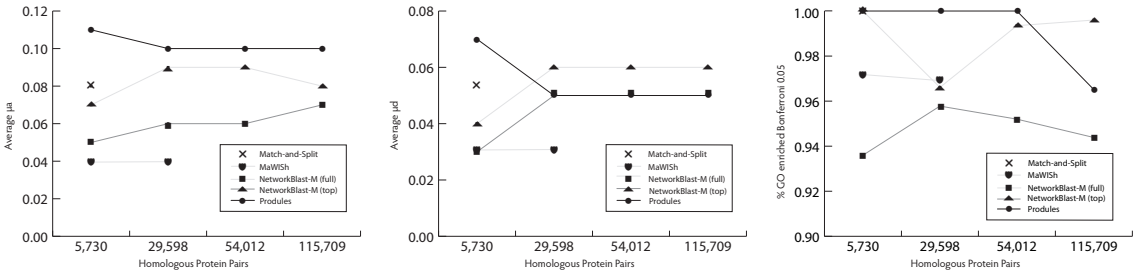


Figure 3.3: Comparison of modularity and GO enrichment showing their correlation. The x-axis is as in Fig. 3.1. The y-axis, from left to right, is the average μ_a, the average μ_d, and the percent of modules enriched at 0.05 significance after Bonferroni correction.

similarly with Produles slightly outperforming MaWISh and NetworkBlast-M when considering full output sets, whereas NetworkBlast-M slightly outperforms Produles at some h-thresholds when considering only its top scoring sets. Fig. 3.1 shows that the top-scoring sets for NetworkBlast-M at the higher h-thresholds have slightly over half the coverage of Produles.

Fig. 3.3 shows that Produles returns modules with highest modularity followed by the top-scoring sets from NetworkBlast-M. This indicates that modularity is correlated with GO enrichment. In Section 3.3.10, we show that several modules returned by Produles without GO enrichment at Bonferroni corrected 0.1 significance are biologically meaningful,

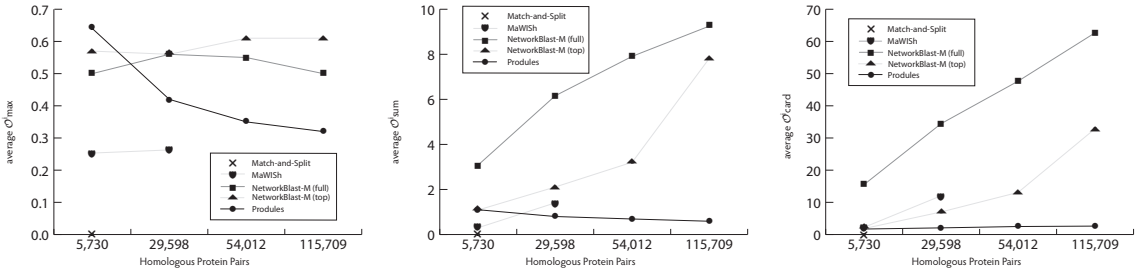


Figure 3.4: Comparison of module overlap. The x-axis is as in Fig. 3.1. The y-axis, from left to right, is the average $\mathcal{O}^i_{max}$, the average $\mathcal{O}^i_{sum}$, and the average $\mathcal{O}^i_{card}$.

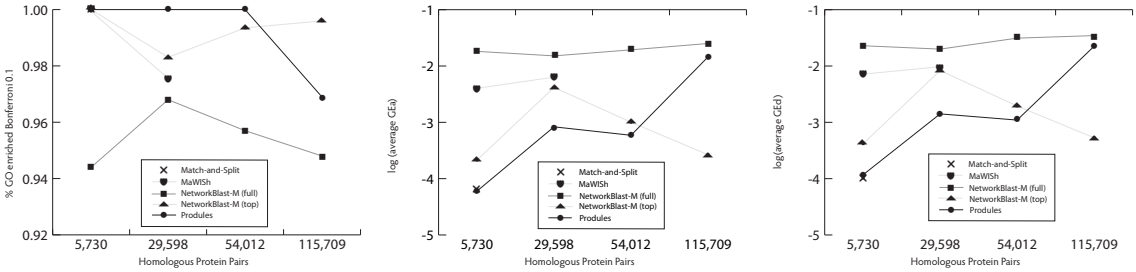


Figure 3.5: Comparison of GO enrichment. The x-axis is as in Fig. 3.1. The y-axis, from left to right, is the percent of modules enriched at 0.1 significance after Bonferroni correction, the log of the average GO enrichment p-value averaged across species, and the log of the average difference in GO enrichment p-values between species.

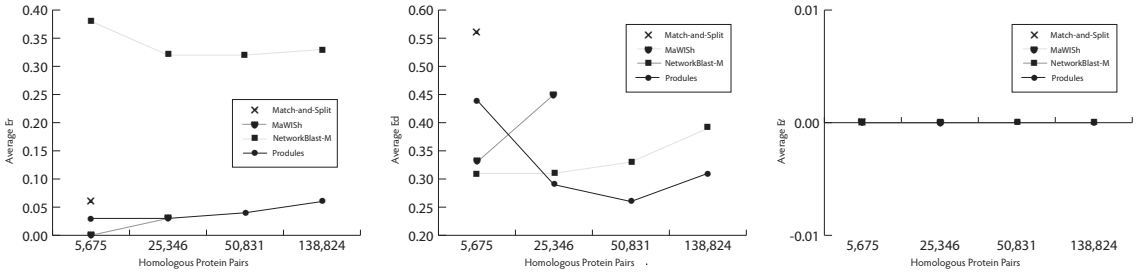


Figure 3.6: Comparison using the interaction level evolutionary model. The x-axis is as in Fig. 3.1. The y-axis, from left to right, is average E_r, average E_d, and average E_ℓ.

demonstrating that conserved modularity is highly correlated with biological function. In Section 3.3.10 we describe how GO enrichment analysis may fail to report enrichment for biologically meaningful modules. More results from GO enrichment are given in Fig. 3.5 including the percent enriched at Bonferroni corrected 0.1 significance. Comparing with the results for 0.05 significance in Fig 3.3 demonstrates that choosing an arbitrary threshold can lead to changes in relative performance, for example the relative performance of MaWISh. In Fig. 3.5, we display average Bonferroni-corrected GO enrichment p-values in a form that is independent of a threshold, calling the measures GE_a and GE_d, extending the definitions of module average and module difference, Definitions 3.1.3 and 3.1.3 in Section 3.1.3, to

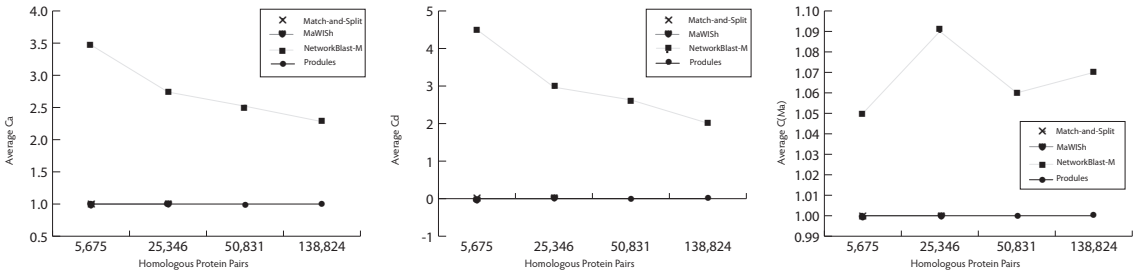


Figure 3.7: Comparison of module boundaries. The x-axis is as in Fig. 3.1. The y-axis, from left to right, is the average C_a, the average C_d, and the average $C(M_a)$.

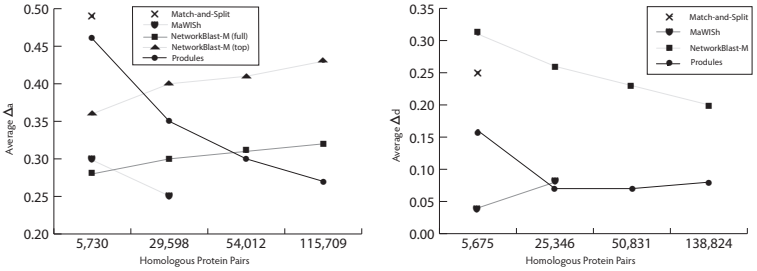


Figure 3.8: Comparison of density and density balance. The x-axis is as in Fig. 3.1. The y-axis, from left to right, is the average Δ_a and average Δ_d.

include GO enrichment p-values.

Fig. 3.4 shows that NetworkBlast-M produces many overlapping module pairs which appear in many cases to be similar to sliding a window across the modules. Produles focuses on optimizing module boundaries by using both the definition of modularity and requiring evidence of module conservation, leading to lower overlap.

Fig. 3.6 shows that MaWISh, uniquely among the algorithms studied, produces modules with low E_r score due to its scoring model that rewards matching graph topologies. Using the random model in Section 3.1.7, estimating $x \approx$ (average S_a/average $|M_a|$) ≈ 2 using Fig. 3.2 and $\Delta \approx \Delta_a \in [0.2, 0.5]$ using Fig. 3.8, Table 3.1 shows that only MaWISh has an E_r score significantly lower than expected by the random model. All algorithms yield modules with large values of E_d, indicating that natural modules may include many proteins with homologous regions. None of the algorithms considered allow protein losses so E_ℓ is zero for all algorithms.

Both Match-and-Split and Produles guarantee that $C_a = C(M_a) = 1$ and $C_d = 0$. Fig. 3.7 shows that MaWISh and NetworkBlast-M have high average values of $C(M_a)$, returning many modules similar to the diagram in Fig. 2.2(a). Figs. 3.7 and 3.8 show that NetworkBlast-M has large values of C_a, C_d, and Δ_d due to additivity of its scoring model in the interaction densities across species. NetworkBlast-M frequently aligns a dense module in one species with a module that has zero or few interactions in the other species. As indicated by Figs. 3.7 and 3.8, for many of these, similar to the diagram in Fig. 2.2(b), the

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	Produles-P	NetworkBlast-M	MaWISh	Match-and-Split
Produles-P		0.09, (0.05,0.11)	0.11, (0.09,0.17)	0.20, (0.18,0.33)
NetworkBlast-M	0.77, (0.67,0.27)		0.70, (0.47,0.22)	0.89, (0.73,0.23)
MaWISh	0.20, (0.22,0.29)	0.15, (0.11,0.12)		0.28, (0.21,0.22)
Match-and-Split	0.21, (0.24,0.41)	0.11, (0.16,0.18)	0.16, (0.10,0.12)	

Table 3.4: $T = 10^{-100}$ with 5,675 homologous protein pairs

interaction data does not support a claim of conservation.

3.3.4 Pairwise Coverage and Overlap of Algorithm Output

Let $\mathcal{U}_i$ be the set of unique proteins in interactome i that an algorithm reports to be part of conserved modules. Let $\mathcal{Y}_i$ be the analogous set reported by another algorithm. Let $\ddot{\mathcal{C}}_i = |\mathcal{U}_i \cap \mathcal{Y}_i|/|\mathcal{Y}_i|$. Let $\ddot{\mathcal{C}} = (\ddot{\mathcal{C}}_1 + \ddot{\mathcal{C}}_2)/2$. Tables 3.4 to 3.7 report $\ddot{\mathcal{C}}$ where the rows correspond to the algorithms used as $\mathcal{U}$ and the columns correspond to the algorithms used as $\mathcal{Y}$. Let $\{(M_1^j, M_2^j) \mid j \in \{1, ..., k_1\}\}$ and $\{(N_1^i, N_2^i) \mid i \in \{1, ..., k_2\}\}$ be the output of two algorithms. Let

$$\ddot{\mathcal{O}}^i = \max_{1 \leqslant j \leqslant k_1} \min\{|M_1^j \cap N_1^i|/|N_1^i|, |M_2^j \cap N_2^i|/|N_2^i|\}$$

Let $\ddot{\mathcal{O}}$ be distributed according to the empirical distribution on $\ddot{\mathcal{O}}^i$. Tables 3.4 to 3.7 also report the pair $(E(\ddot{\mathcal{O}}), \sqrt{\mathbf{var}(\ddot{\mathcal{O}})})$ where the rows correspond to the algorithms used as M and the columns correspond to the algorithms used as N.

Unless otherwise indicated, data sets, programming environment, and parameters were identical to those described in Subsection 3.3.1. For Produles-P, $d = 0.05$. Algorithms were compared after using a filter to remove very large, very small, and unbalanced module pairs. Only pairs of conserved modules with 5-20 proteins per species and with each module having no more than 1.5 times as many proteins as its conserved partner were considered. For each algorithm, the number of module pairs failing the filter and the reasons for the failure are listed in Subsection 3.3.5.

For MaWISh, intra-species BLAST E-values were computed with threshold 10^{-9}, yielding 150,326 pairs of homologous proteins in H. sapiens and 51,956 pairs of homologous proteins in D. melanogaster. For NetworkBlast-M, all interactions are given equal confidence of 1.0. For MaWISh, stringent sequence similarity thresholds are used in place of COGs: inter-species protein pairs are considered orthologous if they have sequence similarity h values at most T_1 and intra-species pairs are considered in-paralogous if they have sequence similarity h values at most T_2 where $T_2 < T_1$. Homology values were converted to MaWISh similiarity scores using the algorithm in their paper [Koyutürk *et al.*, 2006b]. All algorithms were tested with varying sequence similarity threshold T where for MaWISh, $T_1 = T$ and $T_2 = T * 10^{-20}$. The full data set with $T = 10^{-9}$ was used for evolutionary model evaluation.

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	Produles-P	NetworkBlast-M	MaWISh
Produles-P		0.27, (0.17,0.16)	0.29, (0.15,0.22)
NetworkBlast-M	0.72, (0.57,0.28)		0.77, (0.45,0.22)
MaWISh	0.20, (0.11,0.19)	0.20, (0.15,0.12)	

Table 3.5: $T = 10^{-40}$ with 25,346 homologous protein pairs

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	Produles-P	NetworkBlast-M
Produles-P		0.28, (0.14,0.15)
NetworkBlast-M	0.68, (0.51,0.27)	

Table 3.6: $T = 10^{-25}$ with 50,831 homologous protein pairs

3.3.5 Unfiltered Algorithm Output

Let P be the set of module pairs returned by an algorithm. Let $P_f \subseteq P$ be the set of module pairs that fail the filter. Let $P_k \subseteq P_f \backslash (P_1 \cup \cdots \cup P_{k-1})$ be the set of module pairs with at least one module consisting of only k proteins, for $k \in \{1, 2, 3, 4\}$. Let $P_{21+} \subseteq P_f \backslash (P_1 \cup P_2 \cup P_3 \cup P_4)$ be the set of module pairs with at least one module consisting of more than 20 proteins. Let $P_b = P_f \backslash (P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_{21+})$ be the set of module pairs that fail the balance requirement. Let $k = |P|$. Let $k_s = |P_s|$ where s is any subscript.

Filtering has little effect on the results from NetworkBlast-M. Most MaWISh results consist of modules that are conserved single interactions on two proteins which have little significance, and conserved modules on three proteins. Some modules on four proteins may be meaningful and could have been allowed to pass the filter, but they are less significant than larger conserved modules and do not affect the conserved modules found in the range of 5-20 proteins which are the focus of this study. MaWISh returns some modules containing hundreds of proteins with more than 5% of all proteins each, which have $C(M_a) > 1$ and seem to have little significance. Most modules that did not pass the filter were removed by the size filter and relatively few by the balance filter. Most of the conserved module pairs reported by Match-and-Split involve modules with three or four proteins.

3.3.6 MaWISh with reference protein orthology database

In the applications of MaWISh in their paper [Koyutürk *et al.*, 2006b], T_1 is set to a BLAST E-value smaller than 60% of the BLAST E-values between orthologous proteins in COG [Tatusov *et al.*, 2000]. For fixed thresholds, using a database of orthologous proteins as a reference leads to the same number of nonzero $S(\cdot)$ values as our applications, so the

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	Produles-P	NetworkBlast-M
Produles-P		0.28, (0.12,0.14)
NetworkBlast-M	0.68, (0.45,0.26)	

Table 3.7: $T = 10^{-9}$ with 138,824 homologous protein pairs

NetworkBlast-M / T	10^{-100}	10^{-40}	10^{-25}	10^{-9}
k	149	400	614	1021
k_f	15	15	25	47
k_1	0	0	0	0
k_2	1	0	0	0
k_3	0	0	0	0
k_4	0	1	0	1
k_{21+}	0	0	0	0
k_b	14	14	25	46

Table 3.8: Unfiltered NetworkBlast-M results

MaWISh / T	10^{-100}	10^{-40}	PHOG-T(D)	10^{-25}	10^{-9}
k	395	982	976	NA	NA
k_f	373	882	883	NA	NA
k_1	0	0	0	NA	NA
k_2	263	437	476	NA	NA
k_3	72	286	256	NA	NA
k_4	30	104	102	NA	NA
k_{21+}	5	24	20	NA	NA
k_b	3	31	29	NA	NA

Table 3.9: Unfiltered MaWISh results

Match-and-Split / T	10^{-100}	10^{-40}	10^{-25}	10^{-9}
k	63	NA	NA	NA
k_f	50	NA	NA	NA
k_1	0	NA	NA	NA
k_2	0	NA	NA	NA
k_3	28	NA	NA	NA
k_4	21	NA	NA	NA
k_{21+}	0	NA	NA	NA
k_b	1	NA	NA	NA

Table 3.10: Unfiltered Match-and-Split results

MaWISh	$T_1 = 10^{-40}, T_2 = 10^{-60}$	PHOG-T(D), $T_1 = 3 * 10^{-41}$
Running time	**5m**	7m
k	100	93
μ_a	(0.05,0.03)	(0.05,0.02)
μ_d	(0.04,0.05)	(0.04,0.05)
S_a	(7.33,2.58)	(7.75,2.83)
S_d	(1.14,1.07)	(1.03,1.03)
Δ_a	**(0.39,0.13)**	(0.35,0.11)
Δ_d	(0.08,0.09)	(0.07,0.07)
C_a	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	0.04	**0.05**
$\mathcal{O}$	(0.22,0.18)	**(0.20,0.20)**
$\|M_a\|$	(4.43,2.26)	**(5.08,2.81)**
$C(M_a)$	(1.00,0.00)	(1.00,0.00)
E_r	**(0.03,0.09)**	(0.04,0.09)
E_d	(0.45,0.30)	**(0.41,0.27)**
E_ℓ	(0.00,0.00)	(0.00,0.00)

$\ddot{\mathcal{C}}, \ddot{\mathcal{O}}$	MaWISh	MaWISh-PHOG-T(D)
MaWISh		0.69, (0.70,0.37)
MaWISh-PHOG-T(D)	0.75, (0.64,0.37)	

Table 3.11: MaWISh with reference protein orthology database

running time and the sizes of data sets MaWISh can process remain similar. To verify that the omission of a database of orthologous proteins as a reference does not affect MaWISh results significantly, we used the set of all PHOG-T(D) orthologous proteins from PhyloFacts 3.0 [Datta *et al.*, 2009] between the H. sapiens and D. melanogaster proteins as a reference. Any PHOG with more than 30 proteins was removed from consideration. MaWISh was run with the resulting set of PHOGs as a reference and $T_1 = 3 * 10^{-41}$, as 78% of h values between orthologous proteins in this set of PHOGs were above $3 * 10^{-41}$. Detailed results are listed below and in Section 3.3.5.

3.3.7 NetworkBlast-M results minimizing density imbalance

Having discovered an algorithmic flaw in the NetworkBlast model, that density balance is not considered, the module pairs that minimize this flaw, those with lowest Δ_d value, were evaluated. Of all module pairs returned by NetworkBlast-M on the full data set, a summary of the 245 module pairs with lowest Δ_d value is listed below. The evolutionary model results show improvement over the 245 module pairs with highest NetworkBlast-M score.

NetworkBlast-M	full set	highest score	lowest Δ_d
k	1021	248	248
μ_a	(0.07,0.03)	(0.09,0.03)	(0.07,0.03)
μ_d	(0.05,0.04)	(0.06,0.04)	(0.03,0.03)
S_a	(14.12,0.98)	(14.65,0.67)	(14.24,0.87)
S_d	(1.56,1.82)	(0.70,1.34)	(1.16,1.45)
Δ_a	(0.32,0.10)	**(0.45,0.06)**	(0.30,0.11)
Δ_d	(0.21,0.21)	(0.28,0.29)	**(0.02,0.01)**
C_a	(2.28,1.39)	(2.36,1.61)	(1.52,0.62)
C_d	(2.03,2.70)	(2.27,3.21)	(0.53,0.67)
$\mathcal{C}$	0.32	0.09	0.14
$\mathcal{O}$	(0.52,0.21)	(0.65,0.20)	**(0.44,0.27)**
$\lvert M_a \rvert$	(8.95,3.15)	(8.80,3.35)	**(9.88,2.94)**
$C(M_a)$	**(1.07,0.25)**	(1.09,0.28)	(1.10,0.31)
E_r	(0.33,0.23)	(0.36,0.29)	**(0.15,0.16)**
E_d	(0.39,0.23)	(0.43,0.25)	**(0.33,0.22)**
E_ℓ	(0.00,0.00)	(0.00,0.00)	(0.00,0.00)

Table 3.12: NetworkBlast-M minimizing density imbalance

3.3.8 Discussion of Previous Algorithms

3.3.8.1 NetworkBlast

NetworkBlast [Sharan *et al.*, 2005a] is based on a maximum-likelihood scoring model that gives high scores to module pairs inducing dense subgraphs with high sequence-similarity between proteins in the module pair. The model is additive in the densities of the two modules in the pair so the difference in the densities of the modules in a reported conserved pair is often large. A significant fraction of the reported conserved module pairs from NetworkBlast-M [Kalaev *et al.*, 2009], including some of its highest scoring results, consist of module pairs such that one module induces a dense subgraph and the other module induces a subgraph with zero or a small number of interactions. This leads to large values of Δ_d, C_a, C_d, and E_r. Moreover, it is often the case that $C(M_a) > 1$. The search algorithm starts with high-scoring module pairs consisting of one or a few proteins in each interactome and grows them with a greedy algorithm based on the scoring function [Sharan *et al.*, 2005a; Kalaev *et al.*, 2009].

3.3.8.2 MaWISh

MaWISh [Koyutürk *et al.*, 2006b] is based on an evolutionary duplication-divergence scoring model that penalizes protein interaction divergence and infers protein duplications directly from sequence similarity, rewarding recent protein duplications and penalizing ancient protein duplications. The search algorithm is similar to that for NetworkBlast, starting with high-scoring seeds and growing them with a greedy algorithm based on the scoring function. Due to the scoring model described below, MaWISh returns some module pairs with large E_d value and with $C(M_a) > 1$.

The scoring model of MaWISh decomposes into an interaction model and a duplication model. There is a score on each pair of edges from the complete bipartite graph on the proteins in the two modules claimed to be conserved. The score of a module pair is the sum of these scores. High scores are considered good. $S : V_1 \times V_2 \rightarrow [0, 1]$ is a monotonically decreasing function of BLAST E-values that gives a positive value to sequence-similar proteins and a value of 0 to sequence-dissimilar proteins. Let (M_1, M_2) be a pair of conserved modules. Let $u_1, u_2 \in M_1$ and $v_1, v_2 \in M_2$. Two scores are associated with this collection of four proteins: $\mathbf{score}((u_1, v_1), (u_2, v_2))$ and $\mathbf{score}((u_1, v_2), (u_2, v_1))$. Each score is defined symmetrically as

$$\begin{aligned}\mathbf{score}((u_1, v_1), (u_2, v_2)) \quad = \quad & \mathbf{I}(\mathbf{match})S(u_1, v_1)S(u_2, v_2) \\ & -\mathbf{I}(\mathbf{mismatch})S(u_1, v_1)S(u_2, v_2) \\ & +\mathbf{I}(S(u_1, u_2) > 0)(0.1)(S(u_1, u_2) - 0.9) \\ & +\mathbf{I}(S(v_1, v_2) > 0)(0.1)(S(v_1, v_2) - 0.9)\end{aligned}$$

where $\mathbf{I}(\cdot)$ is the indicator function such that $\mathbf{I}(\mathbf{false}) = 0$ and $\mathbf{I}(\mathbf{true}) = 1$. The vast majority of these scores are 0 and are not explicitly represented, as the vast majority of $S(\cdot, \cdot)$ values are 0. The first two terms are the interaction model and the second two terms are the duplication model. **match** and **mismatch** are predicates defined by interactions among the two protein pairs (u_1, u_2) and (v_1, v_2). If both interactions exist, **match** = **true**, **mismatch** = **false** which may lead to a reward. If one interaction exists but not the other, **mismatch** = **true**, **match** = **false** which may lead to a penalty. If neither interaction exists, **match** = **false**, **mismatch** = **false** and the interaction model score is 0.

Considering only $u_1, u_2 \in M_1$, but symmetrically for $v_1, v_2 \in M_2$, the duplication model gives a score of 0 if $S(u_1, u_2) = 0$ or $S(u_1, u_2) = 0.9$. It gives a penalty of $(0.1)(S(u_1, u_2) - 0.9)$ if $0 < S(u_1, u_2) < 0.9$. It gives a reward of $(0.1)(S(u_1, u_2) - 0.9)$ if $S(u_1, u_2) > 0.9$. MaWISh rewards placing highly similar protein pairs in the same module even if they have no interactions between them. As the similarity decreases, the reward gets smaller until it becomes a penalty. As the similarity decreases further, the penalty becomes harsher and harsher until eventually, at a sharp discontinuous cutoff, the penalty vanishes and the duplication model score becomes 0. The reward for placing highly similar proteins in the same module even when they do not involve any interactions leads to a large E_d value.

The MaWISh paper generalizes the scoring model so that the default scoring model given above would be parameterized by $\overline{d} = 0.9, \overline{\delta} = 0.1$. The default value of $\overline{\delta} = 0.1$ places low emphasis on the duplication model relative to the interaction model. However, as MaWISh rewards conserved pairs of proteins with no interactions in either interactome when they are very similar according to $S(\cdot, \cdot)$, even with $\overline{\delta} = 0.1$, a reported conserved module sometimes induces many disconnected subgraphs leading to $C(M_a) > 1$.

3.3.8.3 Match-and-Split

Match-and-Split [Narayanan and Karp, 2007] attempts a symmetric split of both interactomes, recursively matching all pairs of inter-interactome subnetworks that result from the

split, where the two subnetworks in each recursive call are induced subgraphs of the two interactomes. Match-and-Split splits the networks by removing proteins in a network that do not have matching proteins in the other network, and then defining the subnetworks as the resulting connected components. Two proteins are considered matching if they are sequence similar and if both have neighboring proteins in the interactomes that are sequence similar to each other. Unfortunately, this rarely leads to a symmetric split. In most cases each network splits into a large component and many tiny components. The only meaningful recursive comparisons in this case are comparisons involving a large component. The large number of recursive calls in Match-and-Split affects the running time adversely. Several nice features include guarantees that $\mathcal{O} = 0$, $C_a = 0$, and $C(M_a) = 0$.

3.3.9 Modularity in Random Graphs

To examine the extent of conserved modularity in the current interactomes for *H. sapiens* and *D. melanogaster*, both interactomes were randomized while keeping protein sequence similarities fixed. The randomization step consisted of swapping the endpoints of a pair of edges chosen randomly without replacement. More precisely, if $(u_1, u_2), (u_3, u_4)$ are two randomly chosen edges in an interactome, these edges are replaced by the edges $(u_1, u_4), (u_3, u_2)$ unless the four endpoints are not distinct. This randomization maintains the degree of each vertex in the interactome. After all edges in the interactomes were randomized, the various algorithms for conserved module detection were applied to the resulting randomized graphs. Produles with values of d at least 0.05 did not report any conserved module pairs in the random graphs. All other algorithms reported potential conserved modules in the random graphs but none with $\min\{\mu(M_1), \mu(M_2)\} \geqslant 0.05$ and $S_a \geqslant 7$. MaWISh with threshold $h = 10^{-30}$ reported 838 potential conserved modules in the random graphs, but none with $\mu_a \geqslant 0.04$ and $S_a \geqslant 7$. Match-and-Split with threshold $h = 10^{-80}$ reported five potential conserved modules in the random graphs, but none with $S_a \geqslant 7$ or with $\mu_a \geqslant 0.04$. NetworkBlast-M with threshold $h = 10^{-9}$ reported 107 potential conserved module pairs in the random graphs with average $S_a = 13.89$, the same size range as reported on the real interactomes. For these results from NetworkBlast-M, average μ_a was only 0.03. This comparison with random graphs indicates that conserved modularity is a defining characteristic of interactomes.

3.3.10 Multiprotein Modules without GO Enrichment

As shown in Fig. 3.5, when run with $h = 10^{-9}$, approximately 3% of multiprotein modules returned by Produles do not have GO enrichment at Bonferroni corrected 0.1 significance. This corresponds to eight multiprotein modules without significant GO enrichment. Most of these are multiprotein modules reported in the literature with boundaries very well detected by Produles. This section examines four of these conserved modules.

The four conserved modules we examine are from a recent experimental study [Lunardi *et al.*, 2010] that compared the p53 family of tumor suppressor genes in *H. sapiens* with Dmp53, the sole p53-like protein in *Drosophila*. Through independent experiments in both

species, coimmunoprecipitation in human and in vitro pull-down in *Drosophila*, this study found a collection of conserved proteins that interact with the p53 family in human and with Dmp53, indicating that Dmp53 has p53-like function. The h value for Dmp53 with human p73, the member of the p53 family central in the human modules, is $h = 10^{-11}$, so these conserved modules are not detected by algorithms at lower h thresholds. These modules are not detected by NetworkBlast-M presumably because they have density at most $\Delta(M) = 0.18$ despite having modularity as high as $\mu_a = 0.08$. Additional RNAi experiments confirmed function of select module proteins in growth arrest of cancer cells indicating function of these modules in tumor suppression [Lunardi *et al.*, 2010]. It is possible that these annotations have not yet been propagated to GO due to the recency of the study.

These four overlapping modules were combined into a composite module consisting of 22 proteins in *H. sapiens* and 25 proteins in *D. melanogaster*. This composite module has $\mu_a = 0.08$. Six interactions are from other studies and allow us to make predictions to refine the interaction topology of the coimmunoprecipitated complexes. We report here one example. Among the proteins found to coimmunoprecipitate with the p53 family are Asp/ASPM and Sqh/MYL9 [Lunardi *et al.*, 2010]. The proteins Asp/ASPM regulate mitotic spindle formation in *Drosophila* and human respectively. The proteins Sqh/MYL9 are myosin regulatory light chains in *Drosophila* and human, respectively, that have retained their ability to bind calcium. Indeed, Sqh/MYL9 are homologous to calmodulin. A yeast two-hybrid study found that Asp directly binds to calmodulin [Giot *et al.*, 2003]. This indicates that Asp/ASPM may directly bind to Sqh/MYL9, which is why they were coimmunoprecipitated together, and that at most one binds directly to the p53 family.

The remaining four modules are also biologically meaningful. One is the homologous module in *Drosophila* to a deubiquitination complex in human [Sowa *et al.*, 2009]. The human module has a Bonferroni corrected GO enrichment p-value of 2×10^{-5}. The homologous *Drosophila* module has modularity $\mu = 0.13$ and consists of poorly studied proteins that were found to interact by a specialized yeast two-hybrid study. Based on the conserved modularity, we predict that this module is homologous to the deubiquitinating module in human. Another corresponds to a third study. [Lim *et al.*, 2006]. The human module is homologous to a *Drosophila* module with GO $p = 10^{-6}$. The human module has modularity $\mu = 0.1$ and consists of poorly studied proteins with all interactions from the specialized study [Lim *et al.*, 2006]. Another contains interactions from a fourth study [Cannavo *et al.*, 2007] and has GO $p = 0.12$ in the human module; the drosophila module has GO $p = 10^{-3}$. The final one has GO $p = 0.27$ in human but $p = 6 \times 10^{-4}$ in drosophila; the interactions in drosophila are all from high-throughput Y2H. The modularity is $\mu = 0.06$ in drosophila. In human it is known to be a multiprotein module; the proteins are all annotated to a SCF-like ECS (Elongin BC-CUL2/5-SOCS-box protein) E3 ubiquitin-protein ligase complex which mediates the ubiquitination and subsequent proteasomal degradation of target proteins. We predict this to be a conserved multiprotein module across *H. sapiens* and *D. melanogaster*.

Through analyses such as these and detailed follow-up direct interaction experiments, it may be possible to refine results from coimmunoprecipitation experiments to determine

interaction topologies, which may allow an improved interaction level signal of evolutionary conservation in the form of lower E_r scores.

3.3.11 Extent of Conserved Modularity

To examine comprehensively the extent of conserved modularity in the current interactomes for *H. sapiens* and *D. melanogaster*, all modules from NetworkBlast-M with threshold $h = 10^{-9}$ that satisfied $C_a = C(M_a) = 1$ and $\min\{\mu(M_1), \mu(M_2)\} \geqslant 0.05$, a total of 30 module pairs, were examined. The coverage was $\mathcal{C} = 0.03$ with average modularity $\mu_a = 0.08$. As shown in Fig. 3.1, Produles detected coverage of $\mathcal{C} = 0.09$ in conserved modules with $C_a = C(M_a) = 1$ and $\min\{\mu(M_1), \mu(M_2)\} \geqslant 0.05$, with average $\mu_a = 0.10$. When the sets detected by Produles and NetworkBlast-M were combined, the coverage remained $\mathcal{C} = 0.09$ with a small increase in coverage, showing that the conserved modules detected by Produles contained nearly all proteins from the conserved modules detected by NetworkBlast-M. Comparing with Section 3.3.9 on random graphs, this shows that approximately 9% of interactome proteins, 9.6% in human and 8.9% in *Drosophila*, are included in conserved multiprotein modules with good evidence of conservation between the species. This can be compared with the number of proteins that have detectable sequence homology between the species, which at threshold $h = 10^{-9}$, includes 55% of interactome proteins in human and 60% of interactome proteins in *Drosophila*.

3.3.12 Hierarchy of Modularity

To investigate the hierarchy of modularity, all module pairs from Produles with threshold $h = 10^{-9}$ were combined into composite modules. When two conserved modules had overlapping proteins in both interactomes, they were combined. Nineteen non-overlapping composite modules were formed. The largest of these composite modules consists of 611 proteins in *D. melanogaster* and 843 proteins in *H. sapiens* with a modularity of $\mu = 0.19$ in *D. melanogaster* and $\mu = 0.18$ in *H. sapiens*. This conserved modular hierarchy from the largest conserved composite module contains 6.3% of the proteins in *D. melanogaster* and 6.7% of the proteins in *H. sapiens* with $|M_a| = 182$.

When an algorithm detects conserved multiprotein modules that share proteins, these modules can be combined into a larger composite module. Whenever this union takes place, Theorem 2.2.3 shows that the modularity of the composite module is at least as large as the minimum modularity of the modules being combined. This allows inspection of a hierarchy of modularity with larger conserved modules consisting of smaller conserved modules. To investigate these composite modules, all module pairs from Produles-P with $d = 0.05$ were combined into composite modules. When two module pairs had overlapping proteins in both interactomes, they were combined. A summary of the results is listed below. The modularity for these composite modules is similar to that of the original modules. The size increased significantly. Using VieProt for visual inspection, many large reasonable pairs of composite modules are readily seen.

Produles	Produles-P	composite Produles
μ_a	(0.13,0.04)	(0.13,0.04)
μ_d	(0.08,0.06)	(0.07,0.06)
S_a	(7.61,2.62)	(11.17,7.37)
S_d	(1.39,1.27)	(1.76,1.60)
Δ_a	(0.34,0.10)	(0.29,0.14)
Δ_d	(0.08,0.10)	(0.08,0.11)
C_a	(1.00,0.00)	(1.00,0.00)
C_d	(0.00,0.00)	(0.00,0.00)
$\mathcal{C}$	0.13	0.13
$\mathcal{O}$	(0.34,0.32)	**(0.00,0.00)**
$\lvert M_a \rvert$	(5.43,2.19)	**(7.65,6.28)**
$C(M_a)$	(1.00,0.00)	(1.00,0.00)
E_r	**(0.06,0.14)**	(0.07,0.16)
E_d	**(0.31,0.27)**	(0.39,0.29)
E_ℓ	(0.00,0.00)	(0.00,0.00)

Table 3.13: Comparison of Produles-P and composite Produles-P

3.3.13 Discussion of Findings and Extensions

3.3.13.1 Interaction Level Evolutionary Signal

Unfortunately in current interactomics datasets, due to artifacts in the way experiments are performed and recorded, we may not see a strong evolutionary signal of similar ancestral protein relationships or similar graph topologies. Coimmunoprecipitation experiments are often considered more reliable than other assays, but in terms of the evolutionary signal at the individual interaction level, they are among the noisiest. A central protein may not directly interact with all proteins that are coimmunoprecipitated with it. Rather the interactions may be mediated by other proteins in the coimmunoprecipitated cluster. The true graph topology may be a linear path or some other topology rather than the star graph usually recorded in the interaction databases. This causes interaction level signal of evolutionary conservation to be weak. However, as shown in this work, the module-level signal of evolutionary conservation remains strong. As experimental coverage improves, guided by analysis techniques such as those described in Section 3.3.10, the interaction topologies may be improved yielding lower E_r scores in conserved modules.

3.3.13.2 Complementarity of Algorithms

Each algorithm examined in this study has different goals, and they can be considered complementary to each other. MaWISh may be useful when looking for regions with similar graph topologies and low E_r. These regions may be less noisy due to maintenance of an interaction level evolutionary signal, but often do not coincide with natural module boundaries. NetworkBlast-M may be useful when desiring a very large set of potential conserved modules that are frequently very dense in one of the species, may not have evidence of conservation in the protein interaction data, and may have $C(M_a) > 1$. Match-

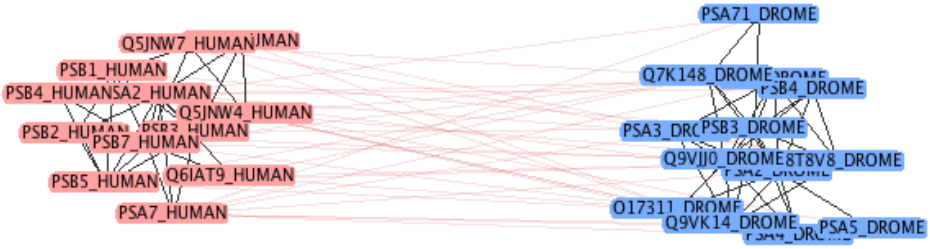


Figure 3.9: The proteosome is detected as conserved by Produles between human and Drosophila with $\mu_a = 0.21$, $\mu_d = 0.13$.

and-Split and Produles may be useful when desiring guarantees such as $C_a = C(M_a) = 1$. Match-and-Split is ideal when desiring a very high quality set of conserved modules with minimal overlap and when it is acceptable to expend large amounts of running time on small datasets. Produles is ideal for detecting conserved module boundaries using evidence from multiple validation in independent experiments across species, for fast running time with good scaling properties, and for examining the extent of conserved modularity in current interactomes. Produles is especially useful for detecting conservation of multiprotein modules that are not particularly dense.

The overlap among the module pairs from the various algorithms tends to be significant as shown by tables in Section 3.3.4, indicating that all algorithms are primarily searching for conserved modules among the same proteins. The difference seems to be largely in how well the boundaries of the conserved modules are discovered.

3.3.13.3 Extent of Conservation

Approximately 10% of proteins in the interactomes for human and fly are found by Produles-P with $d = 0.05$ to be part of conserved modules, a remarkable result, as the ancestors of chordates and arthropods diverged over 500 million years ago [Futuyma, 2009; Hedges, 2002], and as the interactome data remains incomplete [Hakes *et al.*, 2008]. The set of conserved modules includes many well-known protein complexes and pathways, e.g. the proteosome (Fig 3.9).

Given incomplete and sometimes unreliable interactomics data [Hakes *et al.*, 2008], Produles attempts to find regions of the interactomes that are reliable by ignoring those regions that do not exhibit modularity in both interactomes. Less than 15% of the interactomes are found to be part of conserved multiprotein modules even with the most lenient parameter settings. While this may reflect the actual extent of conservation, it is possible that as interactomics data becomes more complete, a larger fraction of the interactomes will be found to be part of conserved multiprotein modules.

3.3.13.4 Running Time Advantages

Because of the large number of recursive calls, Match-and-Split cannot run on large data sets. Match-and-Split is able to run on the smaller data set with $T = 10^{-100}$ when there are 5,675 homologous pairs and less than 50% of the proteins can possibly be considered. MaWISh can process somewhat larger networks but also encounters difficulties when many homologous proteins are considered. MaWISh creates a graph with a vertex for each pair of homologous proteins and then creates edges among these vertices as described in Section 3.3.8.2.

Only Produles and NetworkBlast-M apply to larger networks that include a large number of homologous protein pairs. The module pairs discovered by Produles at higher values of T are of similar algorithmic quality to those discovered at lower values of T. Some of these module pairs contain short homologous proteins with sequence similarity h values above 10^{-20}. These module pairs may be missed by algorithms that cannot process networks with higher values of T.

3.3.13.5 MaWISh Penalties

In MaWISh, duplications are penalized equally for each duplicate pair so the penalty grows quadratically with the number of duplicates, which, while desirable computationally, is not a reasonable model of evolution. The duplication model in this study has similarities to the linear duplication model described in the MaWISh paper [Koyutürk *et al.*, 2006b]. The interaction model of MaWISh penalizes each duplicate equally for interaction loss or gain. During evolution, if a duplicate loses an interaction and then duplicates again, the subsequent duplicate never participated in the interaction and should not be penalized for interaction loss. Even when a copy of the interaction is truly lost, this is not uncommon as duplication weakens purifying selection on redundant copies [Creighton, 1993]. Development of a truly new interaction, complete loss of an interaction, complete loss of a protein, and, to a lesser extent, duplication of a protein, are primary rare events when multiprotein modularity is conserved by purifying selection [Creighton, 1993].

3.3.13.6 Using the Evolutionary Model in Produles for Optimization

Produles uses the evolutionary model defined in this paper only for evaluation. The evolutionary model could be used directly to improve the evolutionary model score. If $||(E_r, E_d, E_\ell)||_2 > E^*$, for some threshold E^*, this variant of Produles would neither report the modules as conserved nor remove their proteins, other than the starting vertex, from future consideration, as done with modularity. This may not, however, be desirable as natural modules would be carved to remove those regions with lower similarity. As MaWISh explicitly attempts to optimize an evolutionary model score, it may carve regions with better scores from natural modules. With MaWISh and this variant of Produles, the reported conserved modules may be the most-highly conserved subsets of actual conserved modules.

3.3.13.7 Limitations of Topology Matching

Noisy interaction data is the data in which we are most interested as it is here that the networks have not been highly studied, and where new modules can be detected. Some previous algorithms, such as MaWISh and Graemlin, attempt to match graph topologies in the networks which requires high-quality interaction data. Produles requires interactomes constructed from independent experiments in various species, so that it can be viewed as a modular form of multiple validation. If multiple independent experiments in different species yield modular regions on homologous proteins, the boundaries of the conserved regions can be seen, even if particular interactions in these regions are false positives and even if some true interactions in this region were not detected by experimental assays.

3.3.13.8 Protein Losses

Though none of the algorithms in this study allow protein losses, and thus $E_\ell = 0$ in all cases, allowing protein losses may improve results, possibly by lowering C_a or increasing μ_a. Algorithms that allow indirect interactions, such as PathBlast [Kelley *et al.*, 2003] and the variant of MaWISh with $\overline{\Delta} > 1$ [Koyutürk *et al.*, 2006b], do allow protein losses.

3.4 TFIID General Transcription Factor: a Case Study

3.4.1 Experiments

Interactomes were obtained from iRefIndex [Razick *et al.*, 2008], Release 7.0, using their binary interaction set, and from IntAct [Kerrien *et al.*, 2012]. Both sets were retrieved and merged using EasyProt software [Hodgkinson *et al.*, 2012]. An EasyProt filter was applied to retain only interactions on proteins with UniProtKB [UniProt Consortium, 2012] accessions. To reduce redundancy, UniProtKB accessions were mapped to UniProtKB identifiers using EasyProt. Precisely, amino acid sequences were retrieved from UniProt in FASTA format and the mapping was extracted from the headers. The resulting interactomes consisted of 75,120 interactions on 12,838 proteins for *H. sapiens* and 32,593 interactions on 9,451 proteins for *D. melanogaster*. Predicted orthologous protein pairs were retrieved from PHOG-T(F) [Datta *et al.*, 2009], yielding 56,659 pairs.

Produles was applied with parameters $a = 10$, $b = 50$, $c = 2$, $d = 0.05$, and $e = 50$, yielding a set of 29 module pairs. Several corresponded to well-known protein complexes including the SCF ubiquitin ligase complex and the proteosome. Pathways were also found including cell differentiation pathways and developmental pathways containing homeobox proteins. Here we focus in detail on one of the detected module pairs, the TFIID general transcription factor required for transcription in eukaryotes.

We examine the portion of TFIID that has good evidence of conservation between humans and *Drosophila*, using protein interaction data from the public databases and protein orthology relations from PHOG [Datta *et al.*, 2009]. The boundaries found by Produles are then refined manually to find the best conserved core supported by the protein interac-

tion data. Remaining differences in the composition of TFIID between the two species are examined using various bioinformatics tools and resources.

3.4.2 Background of the TFIID General Transcription Factor

In eukaryotes, protein-coding genes are transcribed by RNA polymerase II [Alberts *et al.*, 2008]. For transcription to occur, a preinitiation complex, typically containing more than 85 polypeptides [Holstege *et al.*, 1998], must assemble at the promoter of the gene to be transcribed [Thomas and Chiang, 2006]. A central component of most preinitiation complexes is the TFIID general transcription factor. TFIID is a multiprotein complex containing TBP (TATA-box-binding protein) and several TAFs (TBP-associated factors), of which at least 13 homologous groups, named TAF1-TAF13, have been identified as conserved in various species [Tora, 2002]. Components of TFIID, including TBP, TAF1, TAF2, TAF6, and TAF9, bind to DNA at various core promoter elements: a T/A rich region named the TATA box bound by TBP, the Initiator (Inr) bound by TAF1 and TAF2, and the downstream promoter element (DPE) bound by TAF6 and TAF9 [Alberts *et al.*, 2008; Goodrich and Tjian, 2010; Thomas and Chiang, 2006]. Variants of TFIID without TBP are able to initiate transcription in some cases [Wieczorek *et al.*, 1998; Müller *et al.*, 2010].

Two models of the structure of TFIID were assembled from experiments in *Drosophila* and chordates, respectively. These models differ in their composition. The model based on experiments in chordates [Goodrich and Tjian, 2010], consists of thirteen TAFs: TAF1-TAF13. The model based on experiments in *Drosophila* [Wright *et al.*, 2006], consists of only eight TAFs, omitting five of the first thirteen TAFs: TAF3, TAF7, TAF8, TAF10, and TAF13. How much change has actually occurred in TFIID between humans and *Drosophila*? We decided to examine whether bioinformatics tools and protein interaction data stored in public databases could provide insight into the evolution of TFIID.

In particular, five TAFs are present in Figure 1 of [Goodrich and Tjian, 2010] that are not present in Figure 5 of [Wright *et al.*, 2006]: TAF3, TAF7, TAF8, TAF10, and TAF13. For three of these, TAF3, TAF7, and TAF8, there is little or no evidence in the protein interaction databases for their participation in *Drosophila* TFIID. The *D. melanogaster* protein BIP2 is homologous to the TAF3 protein in human, but seems to be only distantly homologous (36% identity over 146 residues with human TAF3, E-value 3e-21, by NCBI BLAST [Sayers *et al.*, 2012]). Literature search supports a functional association with the *D. melanogaster* TFIID pathway and a physical interaction with *D. melanogaster* TAF10 [Gangloff *et al.*, 2001]. However, this protein interaction is not listed in IntAct [Kerrien *et al.*, 2012] or iRefIndex [Razick *et al.*, 2008], nor are any interactions between BIP2 and any TBF or TAF variants recorded in these databases. TAF7 and TAF8 do appear in *D. melanogaster* according to SwissProt [UniProt Consortium, 2012]. However, IntAct does not contain any interactions between *Drosophila* TAF7 or TAF8 and any TBP or TAF variant. iRefIndex reports a single interaction between *Drosophila* TAF8 and TAF10B. Either the interaction data is incomplete or the composition of TFIID differs between the two species.

TAF10 and TAF13 appear in *Drosophila* [Tora, 2002] and do seem to interact with

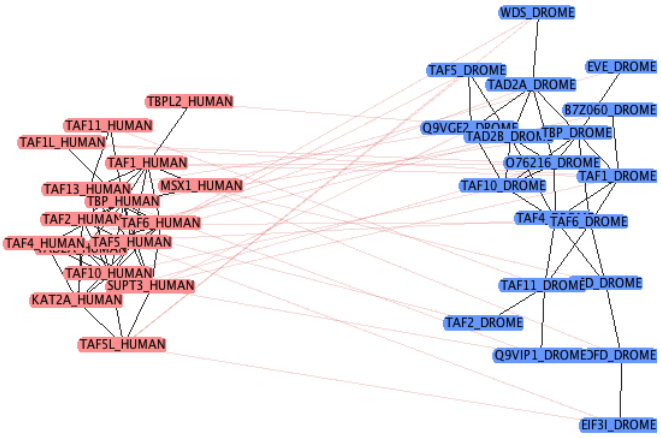


Figure 3.10: TFIID as found by Produles displayed in VieProt: vertices represent proteins with UniProt identifier labels, black edges represent protein interactions, and pink edges represent PHOG-T(F) protein orthology relationships. Proteins in red are from *H. sapiens* and proteins in blue are from *D. melanogaster*.

other proteins involved in TFIID. It would seem unclear why these two proteins were omitted from the experiments and analysis in Wright et al. [Wright *et al.*, 2006] and from the model of TFIID. Though early coimmunoprecipitation experiments in *Drosophila* identified only eight TAFs [Yokomori *et al.*, 1993], later experiments in *Drosophila* showed that TAF10 also coimmunoprecipitates with TFIID [Georgieva *et al.*, 2000]. TAF13 does not have a SwissProt [UniProt Consortium, 2012] entry for *Drosophila*; however the two TrEMBL [UniProt Consortium, 2012] sequences, Q9VIP1 and Q9VGE2, appear in the same PHOG. The sequence Q9VIP1 was found to interact with TAF11 by a high-throughput Y2H experiment [Giot *et al.*, 2003], whereas the sequence Q9VGE2 was found to interact with TAF5 and TAF10 by coimmunoprecipitation [Kusch *et al.*, 2003]. In the coimmunoprecipitation study, Kusch et al. identified Q9VGE2 by homology with Spt3 in yeast. As shown in Table 3.14, SUPT3, the primary homologous protein to Spt3 in human [Brand *et al.*, 1999], is in the same PHOG as TAF13. Brand et al. [Brand *et al.*, 1999] found that Spt3 proteins participate in TBP-free TAFII complexes with multiple TAFs.

3.4.3 TFIID as Found by Produles

Figure 3.10 contains a display from VieProt [Hodgkinson and Kong, 2012] of the TFIID module pair found by Produles on PHOG orthologous proteins. The module pair exhibits high modularity in both interactomes. The projections of the PHOGs on the modules in this module pair are displayed in Table 3.14.

In Table 3.14, two rows are not immediately seen to belong in the same module as the

PHOG	*Homo sapiens*	*Drosophila melanogaster*
PHOG071412087TD	TBP, TBPL2	TBP
PHOG019894133TD	TAF1, TAF1L, KAT2A	TAF1, O76216
PHOG092114207TD	TAF2	TAF2
PHOG071210567TD	TAF4	TAF4, B7Z060
	TAF5, TAF5L	TAF5, WDS, EIF3I
PHOG027708862TD	TAF6	TAF6
PHOG079779542TD	TAF10	TAF10
PHOG071237842TD	TAF11	TAF11
PHOG075676128TD	TAF13, SUPT3	Q9VIP1, Q9VGE2
	TAD2A	TAD2A, TAD2B
	MSX1	DFD, BCD, EVE

Table 3.14: Composition of PHOG projections on the module pair for TFIID as detected by Produles. Proteins are listed by UniProt identifiers without the species suffix. The three rows without PHOG identifiers do not represent single PHOGs. These rows satisfy the property that for any two proteins in a row, one from *H. sapiens* and one from *D. melanogaster*, the pair appear together in a PHOG. The PHOGs for these rows do not claim transitivity of protein orthology. The PHOG identifiers supporting these rows will appear in a future version of the paper.

proteins in TFIID. The proteins TAD2A and TAD2B are called transcriptional adapters or Ada2-like proteins and are found in the PCAF complex that also includes TAF5L, TAF9, TAF10, and TAF12 [Ogryzko *et al.*, 1998]. The proteins MSX1, DFD, BCD, and EVE are homeobox proteins. MSX1 in human, also known as Hox-7, is a transcriptional repressor that is coimmunoprecipitated with TFIID [Zhang *et al.*, 1996]. The function of MSX1 is to interact directly with TBP to repress transcription [Zhang *et al.*, 1996]. The EVE (even-skipped) protein in *Drosophila* performs a similar function in TFIID, repressing transcription by binding directly to TBP and blocking the TFIID-TATA box interaction [Li and Manley, 1998].

The BCD (bicoid) protein interacts directly with TAF6 and TAF4 to activate transcription and this plays a central role in formation of a segmented body plan in *Drosophila* embryos [Sauer *et al.*, 1996]. However, unlike EVE, the protein BCD has a high degree in the interactome, interacting with 72 proteins of which only three are in the module. It does not improve the modularity of the module to include BCD. In future versions of Produles, a modification to remove proteins such as BCD will be added. When BCD is removed, the two proteins DFD and EIF3I become detached and removed from the module, which is good, as they do not play a central role in TFIID.

The proteins TBPL2 and TAF1L substitute for TBP and TAF1, respectively, in the testis and ovary to regulate transcription during germ cell differentiation [Goodrich and Tjian, 2010; Wang and Page, 2002]. KAT2A and O76216, are known as GCN5 proteins. GCN5 proteins are histone acetylases that participate in TBP-free TAF-containing com-

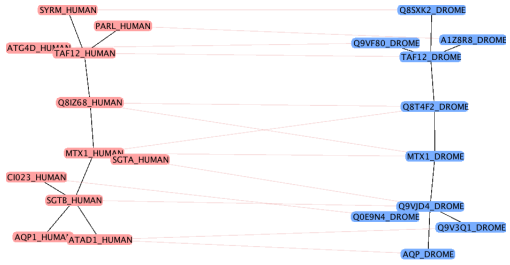


Figure 3.11: The module connecting TAF12 to the mitochondria, as detected by Produles. The pairwise relationships between the PHOGs, whether containing proteins that interact or not, agree exactly in both modules of the pair.

plexes (TFTCs), such as TFTC-HAT which also contains TAF2, TAF5, TAF5L, TAF10, and SUPT3 [Brand *et al.*, 1999]. The sequence B7Z060 is an isoform of *Drosophila* TAF4. The protein WDS, also known as *will die slowly*, is part of the ATAC (Ada two A containing)/Mediator complex that regulates transcription and also includes TAD2A [Krebs *et al.*, 2010]. It is not clear that WDS belongs with the TFIID proteins in the same module. Produles included WDS as it interacts with TAD2A that has a central position in the module and as PHOG predicts orthology between WDS and TAF5. WDS is known to interact with only five other proteins.

The module pair found by Produles does not include TAF9 or TAF12. It is not clear why TAF9 was not found. It may possibly be due to the parameter setting $e = 50$. If TAF9 was internally considered for inclusion in 50 human modules that do not have sufficient evidence for conservation in *Drosophila* or that failed to pass the parameterized quality requirements, then it would have been permanently excluded. This is merely an hypothesis and will be investigated in future work.

The reason TAF12 was missed by Produles is that it had been reported to be in another conserved module pair. Produles removes proteins reported to be in a conserved module pair from consideration for inclusion in other conserved module pairs. Evaluation of various relaxations of this restriction will be left for future work. The module in which TAF12 was found, displayed in Figure 3.11 and Table 3.15, includes TAF12 as a central protein and contains a majority of proteins that are associated with the mitochondria along a path from the nucleus to mitochondria, containing Gene Ontology [Gene Ontology Consortium, 2012] annotations for mitochondrial transport, mitochondrial outer membrane, mitochondrial inner membrane, mitochondrial matrix, and mitochondrial nucleic acid binding. The modules also contain aquaporins that have been shown to play a role in volume regulation of the mitochondrial matrix [Lee and Thévenod, 2006]. The models in *Drosophila* [Wright *et al.*, 2006] and chordates [Goodrich and Tjian, 2010] show TAF12 to be on the periphery of TFIID, possibly receiving or sending signals that coordinate communication with mitochondria. It is known that transcription factors are involved in coordinating communication between the nucleus and mitochondria [Finley and Haigis, 2009].

PHOG	*Homo sapiens*	*Drosophila melanogaster*
PHOG074603165TD	SYRM	Q8SXK2
PHOG064731938TD	PARL	A1Z8R8
PHOG057654444TD	ATG4D	Q9VF80
PHOG079796735TD	TAF12	TAF12
PHOG078530269TD	MTX1, Q81Z68	MTX1, Q8T4F2
PHOG073386299TD	SGTA, SGTB	Q9VJD4
PHOG080986975TD	ATAD1	Q9V3Q1
PHOG014889152TD	CI023	Q0E9N4
PHOG089509716TD	AQP1	AQP

Table 3.15: Composition of PHOG projections on the module pair possibly connecting TAF12 to the mitochondria.

3.4.4 Manual Curation of TFIID

ModuleAlign, as described in Section 2.3.2, was applied to the TFIID module. The results are displayed in Figure 3.12. The module on the left from *H. sapiens* was manually curated by adding to the proteins from the human module in Table 3.14 all TBP and TAF proteins and variants for which protein interaction data existed in the interactomes. This manual curation is reasonable as we have shown that all proteins in the human module found by Produles are coimmunoprecipitated with sizable subsets of the module. The proteins added to those shown in Table 3.14 were TAF3, TAF7, TAF8, TAF9, TAF12, TAF1A, TAF1B, TAF1C, TAF1D, TAF6L, TAF7L, and TAF9B. The largest connected component in *Drosophila* on the right of Figure 3.12 agrees very well with the *Drosophila* module found by Produles in Figure 3.10, differing only in that TAF9 and TAF12 were missed by Produles. It includes all TAFs shown in the model for *Drosophila* [Wright *et al.*, 2006] and also includes TAF10 and TAF13. The largest connected component on the right of Figure 3.12 is not necessarily a gold standard for computational methods that predict conserved modularity from protein interaction data. As observed previously, the module may be improved without inclusion of BCD, DFD, EIF3I, and possibly WDS. An idea for future work is to apply this manual curation protocol in both directions, combining the results from the two directions to form a gold standard module pair for this data set.

3.4.5 Understanding the Conserved Module from Produles

The largest connected component on the right of Figure 3.12 agrees very well with the module on the right of Figure 3.10 found by Produles, differing only in that TAF9 and TAF12 were missed by Produles. The set of TAFs from ModuleAlign includes all those shown in Figure 5 of [Wright *et al.*, 2006] assembled from experiments in *Drosophila* and also includes TAF10 and TAF13. SwissProt does not have a TAF13 entry for *D. melanogaster*, but two TrEMBL [UniProt Consortium, 2012] sequences are orthologous by PHOG-T(F). Both of these TrEMBL proteins participate in the *Drosophila* module found by Produles.

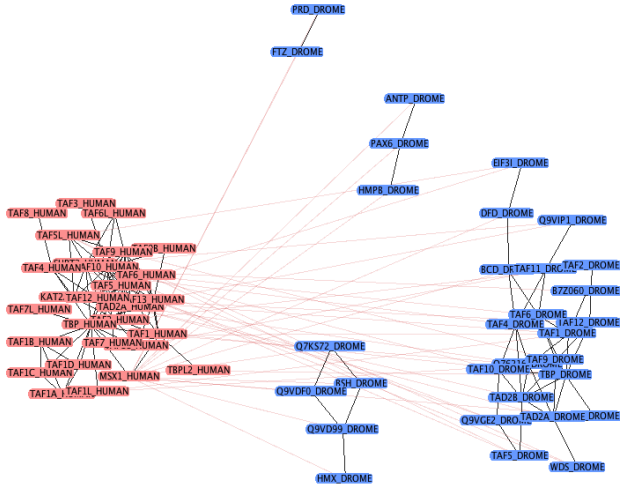


Figure 3.12: TFIID displayed in VieProt after manual curation

These results indicate that the union of PHOG and Produles has enabled detection of the core of TFIID that is known to be conserved across human and *Drosophila*.

We now address the absence of TAF3, TAF7, and TAF8 in the TFIID module for *Drosophila*. There is little or no evidence in the protein interaction databases for their participation in *Drosophila* TFIID. SwissProt does contain entries for *Drosophila* TAF7 and TAF8. However, IntAct does not contain any interactions between *Drosophila* TAF7 or TAF8 and any TBP or TAF variant. iRefIndex reports a single interaction between *Drosophila* TAF8 and TAF10B. This interaction is annotated as coming from a high-throughput Y2H study [Giot *et al.*, 2003]. Additional search revealed that the interaction is also supported by a coimmuniprecipation study [Hernández-Hernández and Ferrús, 2001].

UniProt does not contain an entry for TAF3 in *Drosophila*. The *Drosophila* protein BIP2 is homologous to the TAF3 protein in human [Gangloff *et al.*, 2001], and was shown to coimmunoprecipitate with *Drosophila* TFIID and to interact directly with TAF10 [Gangloff *et al.*, 2001]. However, these interactions are not listed in IntAct [Kerrien *et al.*, 2012] or iRefIndex [Razick *et al.*, 2008], nor are any interactions between BIP2 and any TBF or TAF variants recorded in these databases. The only *Drosophila* BIP2 proteins in UniProt are unreviewed TrEMBL sequences. Using the NCBI entry for BIP2, it appears that BIP2 has 1406 amino acids whereas human TAF3 has 929 amino acids. The alignment from NCBI BLAST [Sayers *et al.*, 2012] with lowest E-value between *Drosophila* BIP2 and human TAF3 aligns over 146 amino acids with 36% identity and E-value 3e-21. For human TAF3, Pfam [Finn *et al.*, 2010] detects a Bromo_TP (Bromodomain associated) domain with envelope from amino acids 4-80 with E-value 2.2e-26, and a PHD-finger (Plant homeodomain) with envelope from amino acids 1342-1389 and E-value 3.7e-11. Pfam detects

identical domains for *Drosophila* BIP2; the Bromo_TP envelope is from amino acids 3-79 with E-value 1.3e-19 and the PHD-finger envelope is from amino acids 867-914 with E-value 1.1e-11.

The TAF7 proteins are shorter, consisting of 479 amino acids in *Drosophila* and 349 amino acids in humans. When scoring *Drosophila* TAF7 against Pfam, the only match is to the family HMM, TAFII55_N, which is the human TAF7 protein conserved region. The envelope is from amino acids 55-214 and the E-value is 4e-53. Human TAF7 was formerly named TAFII55, as it is a protein of approximately 55 kilodaltons, until 2002 when an improved naming convention was introduced [Tora, 2002]. This naming convention established the correspondence for the proteins TAF1-TAF13 between humans and *Drosophila* as well as a mapping to proteins in yeast and *C. elegans*. Pfam naturally finds the same match for human TAF7 with envelope from 12-178 amino acids and E-value 4.2e-55. The proteins align against each other using NCBI BLAST along residues 3-277 in human and 47-315 in *Drosophila* with 41% identity and E-value 1e-44. It is peculiar that any evidence of physical interaction of *Drosophila* TAF7 with TFIID is difficult to find, especially as human TAF7 was coimmunoprecipitated with TFIID and TBP [Wieczorek *et al.*, 1998].

3.4.6 Discussion of Results, Limitations, and Extensions

There are 32 interactions for *Drosophila* from iRefIndex and IntAct in this module, compiled from 8 publications by different authors, most of which were not cited. There are 45 interactions for humans, compiled from 12 publications by different authors, most of which were not cited. This many interactions over so many publications, and the high value of μ exhibited by this pair: $\mu = 0.13$ in human and $\mu = 0.12$ in fly, give us confidence that this module pair is genuinely conserved, despite the fact that some of the interactions may be false positives and other true interactions may not have been detected or reported. In fact, the topological similarity of the two modules in the pair is quite poor. If graphs are created on the rows of Table 3.14 for each of the two species, where an edge is placed between two rows if there is an interaction between a pair of proteins, one from each row, then 44% of these pairwise relationships disagree between the graphs in the pair. Even though we may not be able to see fine-grained details about the truth of particular interactions, the cumulative evidence of the interactions allow us to see the conserved module.

Produles should have better recall for detecting conserved modules in noisy interaction data than the method in Gandhi et al. [Gandhi *et al.*, 2006]. The method in Gandhi et al. [Gandhi *et al.*, 2006] projects orthology groups onto the modules in a possible module pair, removes interactions that do not agree identically in both modules, and then requires that the remaining isomorphic graphs be connected. Produles requires only connectivity, using the modularity of the modules to direct towards regions of the interactomes that are likely to be conserved. Produles should also have better precision as it identifies larger modules for which there is a higher cumulative weight of interaction evidence. A direct comparison would require a gold standard data set of conserved module pairs for which independent interaction evidence exists for more than one species and has been deposited in the publicly available protein interaction databases. Such a gold standard data set to

allow direct comparison has not yet been designed.

The graph topology differs considerably between the two modules in the module pair. Produles is more robust to noisy interaction data than the method in Gandhi et al. [Gandhi *et al.*, 2006]. This is because the method in Gandhi et. al. requires that each interaction be present in both interactomes. More precisely, the method from Gandhi et al. [Gandhi *et al.*, 2006] would project orthology groups onto the modules in the module pair, remove interactions that do not agree identically in both modules, and then require that the remaining isomorphic graphs be connected. Produles requires only connectivity, using the modularity of the modules to direct towards regions of the interactomes that are likely to be conserved.

As demonstrated by our results, the interactomes do not allow one to see temporal or spatial separations. Thus, highly overlapping modules, such as TFIID and the TBP-free TAFII complexes, cannot easily be distinguished except with manual separation.

TFIID is required for eukaryotic transcription of protein coding genes. TFIID is highly conserved between *Homo sapiens* and *Drosophila melanogaster* but there are some noticeable differences in the outcomes of experiments on TFIID in the two species. The portion of the TFIID module known to be most highly conserved between *H. sapiens* and *D. melanogaster* is well detected by Produles. Additional manual curation shows that Produles could have done a bit better, missing TAF9 and TAF12. Produles did not have any knowledge about the boundaries of TFIID and yet detected most of the module. This study validates the use of Produles as an exploratory tool to detect conserved multiprotein modularity in interactomes, but also motivates continued refinement for improved detection of module boundaries.

After our analysis, Produles was modified to allow proteins to be included in multiple modules. When run with the same parameters on the same data set, Produles returned 138 module pairs. Many of these were highly overlapping including some identical module pairs found from different starting positions. Interestingly, the exact two module pairs displayed in Figures 3.10 and 3.11 were found. However, an additional module pair that included most proteins from both of these two modules was also returned. This new module pair had high modularity in both interactomes, $\mu = 0.14$ in human and $\mu = 0.11$ in *Drosophila*, and included 29 proteins in human and 40 proteins in *Drosophila*. For *Drosophila* this module pair included TAF1, TAF2, TAF4, TAF5, TAF6, TAF9, TAF10, TAF11, TAF12, the two TAF13 TrEMBL sequences, BIP2, TAD2A, and TAD2B. Note, in particular, that TAF12, TAF9, and BIP2 were included. In human, the module pair included TAF2, TAF4, TAF5, TAF5L, TAF6, TAF8, TAF9, TAF10, TAF11, TAF12, TAF13, KAT2A, and SUPT3. Several points are worthy of note. First, TBP is not included. It is possible that this is due to the parameter value $e = 50$. Investigation of this will be left for future work. Next, TAF8 was included in the human module. TAF8 does not appear in the *Drosophila* module; rather the human TAF8 is aligned to BIP2, as PHOG-T(F) predicts orthology between BIP2 and TAF8. Recall that BIP2 is putatively orthologous to TAF3 in human. This PHOG-T(F) prediction of orthology is likely due to a shared Bromo_TP domain from Pfam. The bromodomain is composed of four alpha helices and recognizes acetylated lysines on histones [Charlop-Powers *et al.*, 2010]. Next, TAF1 does not appear in human; rather it

is substituted by KAT2A which appears in the same PHOG. Finally, we note that several proteins, especially from *Drosophila*, seem that they could be removed without destroying the connectivity or decreasing the modularity.

Chapter 4

Evaluation of Optimization Criteria

In this chapter, we examine potential optimization criteria for detecting homology between *Homo sapiens* and *Drosophila melanogaster* at the level of multiprotein modules, exploring whether optimizing these attributes contributes significantly to biological process enrichment of the modules. After generating pairs of modules containing proteins homologous between the species, we consider different types of attributes: graph theoretic attributes intrinsic to a single module and homology attributes that consider the relationship between homologous modules. We also consider the appropriateness of biological process enrichment for measuring quality of module boundaries across modules of different sizes.

Modularity, defined as the fraction of interactions within the module among all interactions in which the module proteins participate, has been shown to lead to significant biological process enrichment [Hodgkinson and Karp, 2012]. Graph conductance [Jerrum and Sinclair, 1988], a closely related concept, has been used to detect modularity in protein interaction networks not restricted to homologous modules [Voevodski *et al.*, 2009]. A similar definition of modularity [Newman and Girvan, 2004] defined for a partition of the network into modules has been used with good results [Wang *et al.*, 2007]. Because of these promising results, modularity is a prime candidate for predicting biological process enrichment.

Criticisms of modularity include claims that modules are not sufficiently dense [Sun *et al.*, 2012]. This prompts us to include a measure of density – how similar a module is to a clique – as a possible attribute for predicting biological process enrichment. Density is defined as the fraction of edges in the module over the number of edges if the module were a clique.

Two very obvious attributes to consider are the number of interactions in the module and the module size. If these attributes are important, does biological process enrichment continue to improve without bound as these quantities increase, or are there particular ranges that optimize biological process enrichment? If biological process enrichment continues to improve without bound, this would indicate a limitation in the use of enrichment to evaluate module boundaries.

Modules vary in the diversity of proteins comprising them. It seems plausible *a priori*

that modules with many paralogous proteins may exhibit more significant biological process enrichment than those with fewer paralogous proteins due to inherited similarity of function. However, it may also be possible that modules with more diverse protein composition have more significant biological process enrichment as each protein may perform a different molecular function in the biological process associated with the module.

Protein interaction homology manifests itself through topological similarity in the protein interaction networks across species. Several attempts have been made to use protein interaction homology as a guiding principle to detect homologous modules, the idea being that network regions containing homologous patterns of protein interactions are more likely to represent modules with important functions [Koyutürk *et al.*, 2006b; Narayanan and Karp, 2007]. We define a model of protein interaction homology and study its effects on biological process enrichment.

4.1 Study System

iRefIndex [Razick *et al.*, 2008] is a consolidated database of protein interactions that can be accessed with the PSICQUIC web services protocol [Aranda *et al.*, 2011]. Release 9.0 (Dec. 16, 2011) provides unified protein interaction data through a publicly accessible PSICQUIC web service – data compiled from IntAct [Kerrien *et al.*, 2012], BioGrid [Chatr-aryamontri *et al.*, 2013], DIP [Salwinski *et al.*, 2004], HPRD [Prasad *et al.*, 2009], BIND [Isserlin *et al.*, 2011], MINT [Licata *et al.*, 2012], MIPS [Mewes *et al.*, 2011], InnateDB [Breuer *et al.*, 2013], and MatrixDB [Chautard *et al.*, 2011]. UniProtKB is a publicly available database of protein sequences accessible via web services [UniProt Consortium, 2012]. BLAST is software that detects sequence similarity of protein sequences and can be used to infer homology of proteins across taxa [Sayers *et al.*, 2012]. BLAST is released as a stand alone application to compute pairwise similarities between two large sets of proteins very quickly [Sayers *et al.*, 2012]. Gene Ontology is a database of annotations for protein functions, including biological processes, that can be used for comparing known functions of proteins within modules [Gene Ontology Consortium, 2012].

4.2 Study Species

H. sapiens and *D. melanogaster* were chosen as study species since their protein interaction networks have been the most extensively studied of all metazoans. *H. sapiens* and *D. melanogaster* are separated by over 500 million years of evolution, their most recent common ancestor being an early member of the Bilateria clade [Cartwright and Collins, 2007]. It is estimated that D. melanogaster has approximately 14,000 protein-coding genes whereas *H. sapiens* has between 20,000 and 25,000 protein-coding genes [Pray, 2008]. 67 percent of the proteins in their protein interaction networks have homologous proteins in the other species with BLAST E-value $< 10^{-9}$ when averaged in both directions.

4.3 Optimization Criteria

Let $G = (V, E)$ be an interactome. A multiprotein module is a set of proteins $M \subset V$ such that $|M| \ll |V|$ and such that the proteins in M form an induced connected subgraph of the interactome. Modularity is defined as $\mu(M) = \frac{|E(M)|}{|\texttt{cut}(M, V \backslash M)| + |E(M)|}$ where $E(M)$ is the set of interactions with both interactants in M, and $\texttt{cut}(M, V \backslash M)$ is the set of interactions spanning M and $V \backslash M$. Of the interactions involving module proteins, modularity is the fraction contained entirely within the module. Linear density is defined as $\Delta_L(M) = |E(M)|/|M|$, the number of interactions between proteins in the module divided by the module size. Density is defined as $\Delta(M) = |E(M)|/\binom{|M|}{2}$. Of the total possible number of edges that could be in a module of a given size, density is the fraction of edges actually contained in the module.

To model the extent of protein richness, we placed each pair of module proteins with BLAST E-value threshold 10^{-9} in the same protein homology group including the proteins from both species. Protein richness of a module is the number of protein homology groups that include module proteins. Protein richness can continue to increase as the modules increase in size so we divided by module size for a meaningful measure across modules of different sizes.

To model protein interaction similarity, we tested how well the interactions between protein homology groups were conserved in *H. sapiens* and *D. melanogaster*. For either species, if protein homology group A had a proteins in the species and protein homology group B had b proteins in the species, and the module for the species had density Δ, then if the interactions between A and B were thrown down at random, we would expect $ab\Delta$ interactions between A and B in that species. We examined the actual number for each species to see whether it was above or below expectation. If they were both greater than or equal to expectation or both less than or equal to expectation, we called this an agreement; otherwise it was a disagreement. This model insists that interactions between protein homology groups should be mostly present in both species or mostly absent in both species for the interaction to be considered homologous. If there were n protein homology groups in the module, there were $\binom{n}{2}$ pairs of protein homology groups and this same number of possible agreements. However, a large percentage of the agreements are between protein homology groups that have no interactions between them in either species. We thus ignore agreements of this type and measure only agreements with at least one interaction between the protein homology groups in at least one species, reporting the fraction of the total possible number of these agreements.

4.4 Computational Methods

We obtained protein interaction data from iRefIndex, Release 9.0, for *H. sapiens* and *D. melanogaster*, consisting of 69,651 interactions on 12,692 proteins for *H. sapiens* and 38,731 interactions on 9,796 proteins for *D. melanogaster*. We filtered out any interactions that were derived from computational rather than experimental sources. Protein sequences for

all proteins involved in iRefIndex interactions were retrieved from UniProtKB. We accessed UniProtKB data from March 26, 2012. Pairwise BLAST was run between the *H. sapiens* and *D. melanogaster* proteins to determine protein homologies. We recorded BLAST E-values for each pair. All data were retrieved and processed using the EasyProt software architecture [Hodgkinson *et al.*, 2012].

Beginning at each protein, in turn, we randomly generated two sets of proteins, one set per species. Let A and B be sets of proteins for human and Drosophila respectively, initially containing one protein each. For a set of proteins S, let $I(S)$ be the set of proteins interacting with any protein in S. Let $H(S)$ be the set of proteins homologous to any protein in S in the other species than the proteins in S. Each step of the growing process added a protein p to A randomly selected from $\{p \in I(A) : H(\{p\}) \cap I(B) \neq \varnothing\}$. Next a protein in $H(\{p\}) \cap I(B)$ was chosen at random and added to B unless $H(\{p\}) \cap I(B) \subseteq B$. The roles of A and B were then reversed. This procedure was repeated until both sets were either of a randomly chosen maximum size between 5 and 50 proteins or until it was impossible to grow the sets further. The collection of sets with at least 5 proteins per species was retained for statistical analysis.

When optimizing for biological process enrichment, modularity, linear density, and density we started from randomly chosen proteins and generated a set of conserved modules using the above algorithm, adding only proteins that improved the chosen attribute of the module, allowing up to 50 proteins per module. If the modules reached a size where it was impossible to improve the chosen attribute by adding a pair of homologous proteins, one per species, both interacting with module proteins, then the modules were returned at this size. If the modules were able to grow beyond 50 proteins, they were returned at 50 proteins. When optimizing for density, the homologous module pairs were grown at random up to 5 proteins per species before the optimization criteria were applied due to the difficulty in finding larger cliques. From 100 homologous module pairs, 200 modules were optimized for each of modularity, linear density, and density. Due the long running time, we optimized only 12 modules from 6 homologous module pairs for biological process enrichment. We used the randomly generated set of 5166 modules from 2583 homologous module pairs as a control.

To measure the similarity of known biological processes across proteins within modules, Gene Ontology enrichment values were calculated for each module, separately for each species, using Ontologizer [Bauer *et al.*, 2008]. We used latest releases of the full ontology (March 26, 2012 release) and unfiltered UniProt annotation data (March 6, 2012 release) from the Gene Ontology website. Ontologizer was applied with the Term-For-Term setting with Bonferroni correction and each annotation received a P-value. The lowest P-value for each module was retained for the statistical analysis and transformed using the negative base-10 logarithm.

4.5 Statistical Methods

All statistical analyses were performed using R [R Development Core Team, 2012]. Multiple curvilinear regressions were fitted to predict the negative logarithm of the biological process enrichment P-value using six attributes: modularity, linear density, density, module size, size corrected protein richness, and protein interaction similarity. For protein interaction similarity, each module in the pair was assigned the measurement from the pair. Each regression was fitted using all first and second order terms including interactions between variables. A binary recursive partitioning algorithm [Breiman *et al.*, 1984], was used to determine the most important attributes for prediction of biological process enrichment P-values. Unless indicated otherwise, all statistical results given have P-value < 0.01.

For relative importance, we used the relaimpo R package with the four methods: LMG, Pratt, Last, and First [Grömping, 2006]. LMG assigns each variable the average increase in R^2 when it is added to a regression model containing a subset of other variables [Grömping, 2007]. Pratt measures the product of the regression coefficient and the zero-order correlation for each variable [Thomas *et al.*, 1998]. Starting with the full model, Last assigns the reduction in R^2 when removing a variable from the model as the relative importance of the variable maximizing this quantity; this maximizing variable is then removed from the model and the algorithm recurses on the smaller model. First is similar to Last but adds the variables from an empty model rather than removing them from a complete model, adding the variables in order of relative improvement in R^2. Bootstrapping with 1000 bootstrap replicates was used to generate 95 percent confidence intervals for each method.

The 5166 random conserved modules generated as described in Computational Methods were binned by number of module proteins. For each bin, we computed the mean and standard deviation for each attribute. For each attribute, each optimized module generated as described in Computational Methods was mapped to the number of standard deviations above or below the mean in the random modules, allowing direct comparison across modules of different sizes.

All tests comparing medians of attributes in sets of modules were conducted using a two-sided Wilcoxon rank sum test with continuity correction.

4.6 Experiments to Compare Optimization Criteria

4.6.1 Correlations and Regressions of Attributes

After generating 2583 homologous module pairs at random, 2583 modules per species, we computed six attributes for each module pair: modularity, density, number of interactions, module size, protein richness, and protein interaction similarity. Number of interactions and module size were highly correlated with Pearson's $r^2 = 0.91$ so we divided number of interactions by module size yielding an alternate measure of density that we call *linear density* following [Melancon, 2006], and we did not consider number of interactions directly. Protein richness was also highly correlated with module size with Pearson's $r^2 = 0.84$ so we divided protein richness by module size to create a new attribute that we call *size corrected*

linear density (lind)	0.179
modularity (mod)	0.175
module size (size)	0.112
density (dens)	0.062
size corrected protein richness (rich)	0.031
protein interaction similarity (sim)	0.011
mod, lind, size, dens, rich, sim	0.377
mod, lind, size, dens, rich	0.372
mod, lind, size, dens, sim	0.370
mod, lind, size, dens	0.356
mod, lind, size	0.357
mod, lind	0.313
mod, size	0.291
mod, dens	0.242
sim, rich	0.043

Table 4.1: Adjusted R^2 values for curvilinear regressions on attributes and subsets. Abbreviations in parentheses are used throughout.

protein richness. We then performed curvilinear regression of biological process enrichment on each of the six attributes individually, using first and second order terms. The adjusted R^2 values are listed in Table 4.1. The correlation matrix of the attributes used in the regressions is given in Table 4.2. Multiple curvilinear regression of biological process enrichment on subsets of the six explanatory variables, including interactions between variables and second order terms, yielded the adjusted R^2 values given in Table 4.1.

The protein interaction similarity score described in Section 4.3 ranged from 0 to 1 on the random modules with a mean of 0.29 and a standard deviation of 0.15. The correlations with biological process enrichment were $r = -0.05$ (Pearson's) and $\rho = -0.01$ (Spearman's).

	enrich	mod	lind	size	dens	rich	sim
enrich	1.00000000	0.39940960	0.4125307	0.33289206	-0.20491436	-0.16507973	-0.04770611
mod	0.39940960	1.00000000	0.1483411	0.01513133	0.08709516	0.03372056	0.06667026
lind	0.41253071	0.14834113	1.0000000	0.42383546	-0.13505171	-0.12098508	-0.17010913
size	0.33289206	0.01513133	0.4238355	1.00000000	-0.82738012	-0.19738641	-0.48665568
dens	-0.20491436	0.08709516	-0.1350517	-0.82738012	1.00000000	0.22562763	0.50260948
rich	-0.16507973	0.03372056	-0.1209851	-0.19738641	0.22562763	1.00000000	-0.20942665
sim	-0.04770611	0.06667026	-0.1701091	-0.48665568	0.50260948	-0.20942665	1.00000000

Table 4.2: Correlation matrix of attributes on random conserved modules.

4.6.2 Stratification by Size

Despite the strong correlation between number of proteins and number of interactions in the random modules (Pearson, $r^2 = 0.91$), for modules of a given size there is considerable variation in the number of interactions. For the modules of each size, we performed a linear

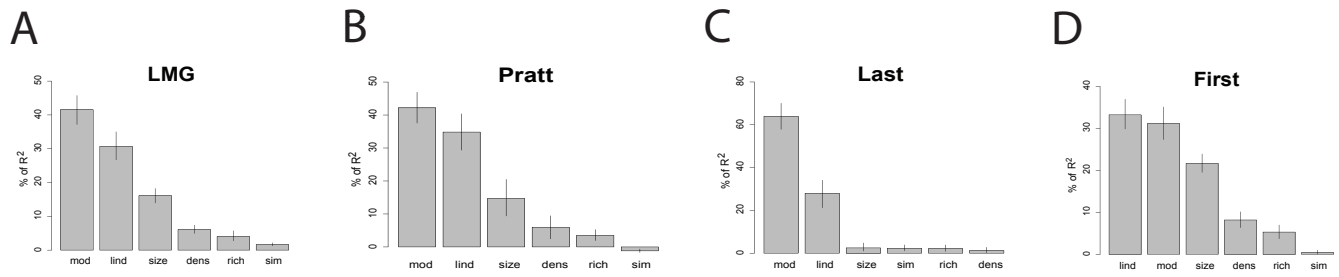


Figure 4.1: Relative importances of optimization criteria for predicting biological process enrichment. Four relative importance methods were used: A) LMG, B) Pratt, C) Last, and D) First. Methods were applied to 2583 randomly generated homologous module pairs. 95 percent confidence intervals were generated from 1000 bootstrap replicates.

regression of enrichment on number of interactions. All 53 least squares best fit lines had positive slope. This shows that for any fixed size, improving the density or linear density tends to improve biological process enrichment. The Pearson partial correlation between number of interactions and biological process enrichment holding number of proteins fixed was 0.28 with P-value $< 10^{-96}$. However, biological process enrichment and density are negatively correlated (Pearson's $r = -0.20$; Spearman $\rho = -0.26$). It is easier to find smaller dense regions but improving density at the cost of size tends to diminish biological process enrichment.

4.6.3 Relative Importances of Attributes

Four methods to determine the relative importance of attributes in curvilinear regressions are described in Section 4.5. Fig. 4.1 shows the results of these methods applied to our regressions. *LMG*, *Pratt*, and *Last* listed modularity as the most important attribute. *First* listed linear density as the most important attribute with modularity as the second most important.

4.6.4 Binary Partition Tree

Curvilinear regression models apply a polynomial function of the attributes to the entire sample space. An alternate nonparametric approach uses a binary partition tree to model the data with different predictors for different discrete ranges of attribute values. A binary recursive partitioning algorithm [Breiman *et al.*, 1984] generated the decision tree in Fig. 4.2 with modularity, linear density, and size being the most important attributes for predicting biological process enrichment.

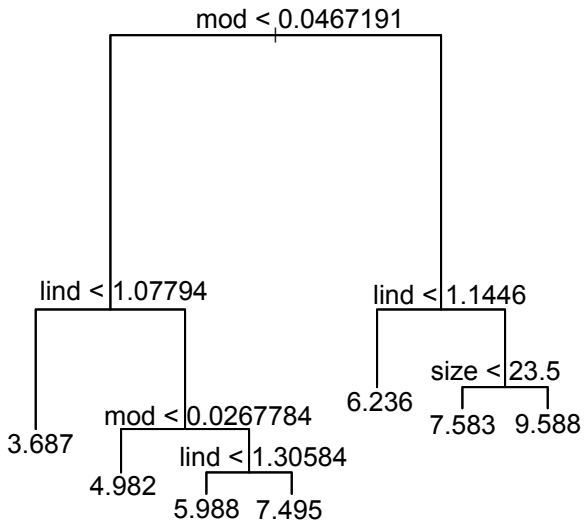


Figure 4.2: Prediction tree for biological process enrichment in random modules.

4.6.5 Optimizations

As described in Section 4.4, the algorithm was modified to select only proteins that improved the biological process enrichment of the modules. Due to the long running time of 8-20 hours per homologous module pair, we generated only 6 homologous module pairs in this way, yielding 12 modules. For each optimization criterion other than biological process enrichment we generated 100 homologous module pairs optimized for that attribute as described in Section 4.4. We optimized for both high protein richness and low protein richness.

Fig. 4.3A compares biological process enrichment of optimized modules with random modules after correcting for size as described in Section 4.5. The average sizes of modules resulting from the optimizations are listed in Table 4.3. The median of biological process enrichment in each optimized group of modules was greater than the median in the random control group (Wilcoxon, $P < 10^{-8}$ in all cases except for the density optimized modules where $P = 0.0017$). The median of enrichment in the group optimized for enrichment was the highest, as expected, greater than the median of enrichment in each group optimized for any other attribute (Wilcoxon, $P < 10^{-6}$ in all cases). The medians of enrichment in the groups optimized for modularity and linear density were greater than the median of enrichment in the group optimized for density (Wilcoxon, $P = 0.0013$ and $P < 10^{-9}$ respectively), but the median of enrichment in the group optimized for modularity was less than the median of enrichment in the group optimized for linear density (Wilcoxon, $P = 0.0015$).

Fig. 4.3B shows a box plot of the modularity in the various groups. The median of

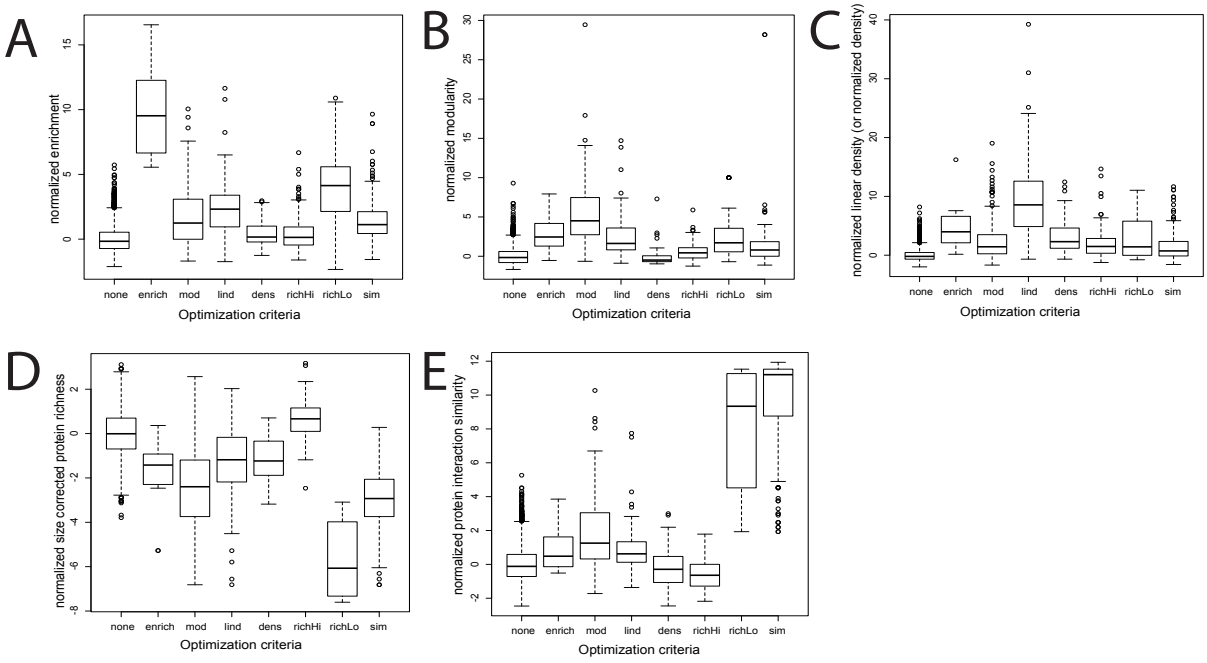


Figure 4.3: Box plots of optimized conserved modules against a random control. A: biological process enrichment, B: modularity, C: linear density and density, D: size corrected protein richness, E: protein interaction similarity. The control consists of 2583 randomly generated conserved module pairs. Each optimized set except for biological process enrichment consists of 100 optimized conserved module pairs. The biological process enrichment optimized set consists of 6 optimized conserved module pairs.

modularity in the group optimized for enrichment was greater than the median of modularity in the random control group (Wilcoxon, $P < 10^{-5}$), but the median of modularity in the group optimized for modularity was greater than the median of modularity in the group optimized for enrichment (Wilcoxon, $P = 0.007$).

Fig. 4.3C shows a box plot of linear density and density in the various groups. That both linear density and density are represented by the same plots follows from the size correction described in Section 4.5 along with the definitions of density and linear density. For modules of any given size, the denominators of both density and linear density are constants, so each module is placed the same number of standard deviations above or below the mean for both density and linear density.

Fig. 4.3D shows a box plot of size corrected protein richness. The enriched set has lower protein richness than the random set and most other optimized sets. This is as expected from the negative correlation between enrichment and protein richness in the random set.

The greedy algorithm used in this study was remarkably effective at optimizing for

size corrected protein richness maximized	48.11
protein interaction similarity optimized	41.65
linear density optimized	38.29
enrichment optimized	36.58
size corrected protein richness minimized	29.83
modularity optimized	28.54
no optimization	27.79
density optimized	6.08

Table 4.3: Average sizes of optimized modules

protein interaction similarity. For every conserved module pair in the optimized set, the protein interaction similarity score was 1.0, indicating perfect topological agreement in each module with the average number of protein homology groups per module being 14.13. Even so, as shown in Fig. 4.3A, these modules did not exhibit greater biological process enrichment than modules optimized only for modularity or linear density which, as shown in Fig. 4.3E, had significantly lower levels of protein interaction similarity.

The enrichment optimized modules tend to have higher values of both modularity and linear density than the random modules, but the modularity optimized modules do not have high linear density (Fig. 4.3C) and the linear density optimized modules do not have extremely high modularity (Fig 4.3B). This is consistent with the regression results in the random modules that found modularity and linear density to be complementary predictors of biological process enrichment.

From Figure 4.3A is is clear that minimizing protein richness improves biological process enrichment. Figure 4.3E shows that the set of conserved modules optimized for low protein richness indeed has very low protein richness relative to the random set. Many of the conserved modules optimized for low protein richness had only one or two protein homology groups which yield perfect protein interaction similarity agreement as indicated in Figure 4.3E. Figure 4.3D shows that optimization for every attribute except high protein richness led to lower protein richness than in the random modules.

4.6.6 Size Effects

Fig. 4.4A shows that biological process enrichment tends to improve as the number of proteins increases up to 50 proteins per module in the randomly generated conserved modules. To test at larger size ranges, 100 homologous module pairs were generated using the algorithm in Section 4.4 choosing random maximum sizes between 5 and 2000 with results displayed in Fig. 4.5A. As shown in Fig. 4.4B, there is no strong correlation between modularity and size in modules up to 50 proteins. The correlation between modularity and size is only $\rho = 0.108$ (Spearman's) and $r = 0.015$ (Pearson's) which is statistically significant for Spearman's ($P < 10^{-14}$) but not for Pearson's ($P = 0.28$). However, as shown in Fig. 4.5B, at larger size ranges, modularity is highly correlated with number of proteins. Linear density tends to increase with number of proteins at all size ranges (Figs. 4.4C and 4.5C). However, density, normalized protein richness, and protein interaction similarity tend to

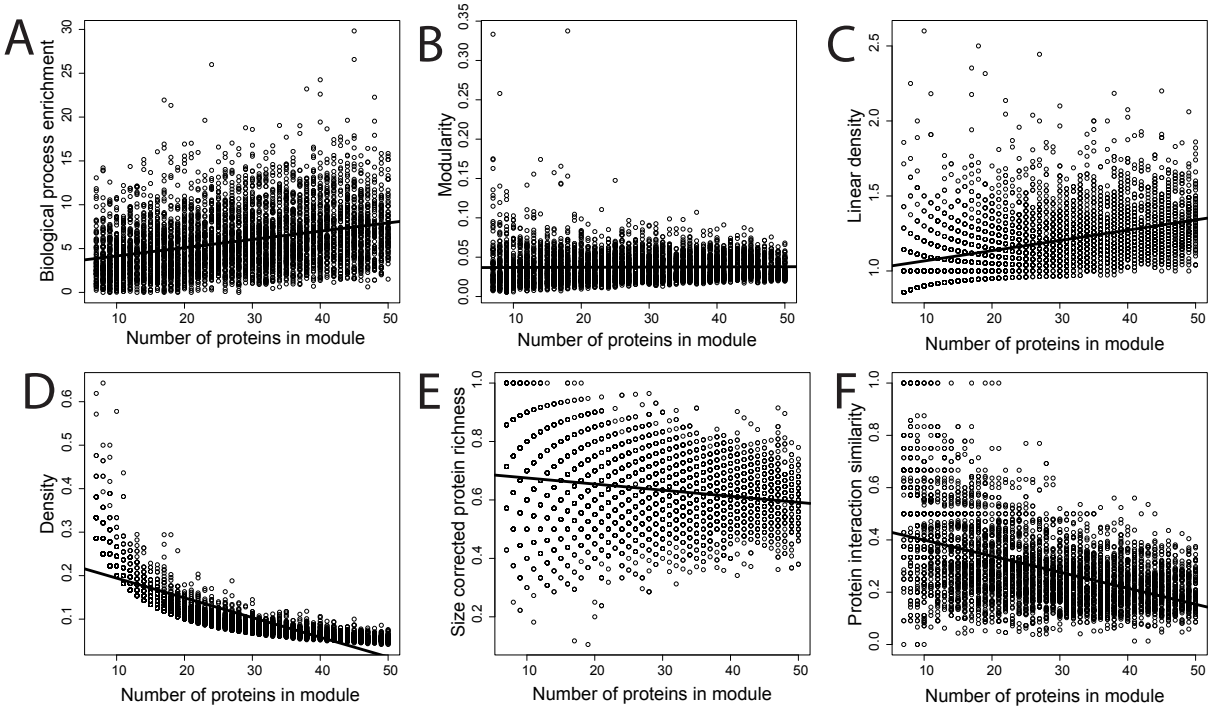


Figure 4.4: Trends by size of 5166 randomly generated conserved modules containing between 5 and 50 proteins each. A: biological process enrichment, B: modularity, C: linear density, D: density, E: size corrected protein richness, F: protein interaction similarity. Least squares best fit lines shown. In each case, the y-axis is the value of the attribute and the x-axis represents module size increasing to the right.

decrease with size (Figs. 4.4D, 4.4E, 4.4F, 4.5D, 4.5E, 4.5F).

4.7 Discussion of Results and Limitations

4.7.1 Importances of the Optimization Criteria

When biological process enrichment is considered a gold standard for module composition, modularity and linear density are the most important optimization criteria and are complementary. Optimizing for protein interaction similarity has only a modest additional effect. The small difference in the adjusted R^2 between the model using only graph theoretic attributes and the full model shows that the attributes based on homology (size corrected protein richness and protein interaction similarity) do not provide significant additional information for predicting biological process enrichment. Three graph theoretic attributes: modularity, linear density, and module size together explain 35.7 percent of the variance in biological process enrichment, whereas the full model explains only an additional 2 percent.

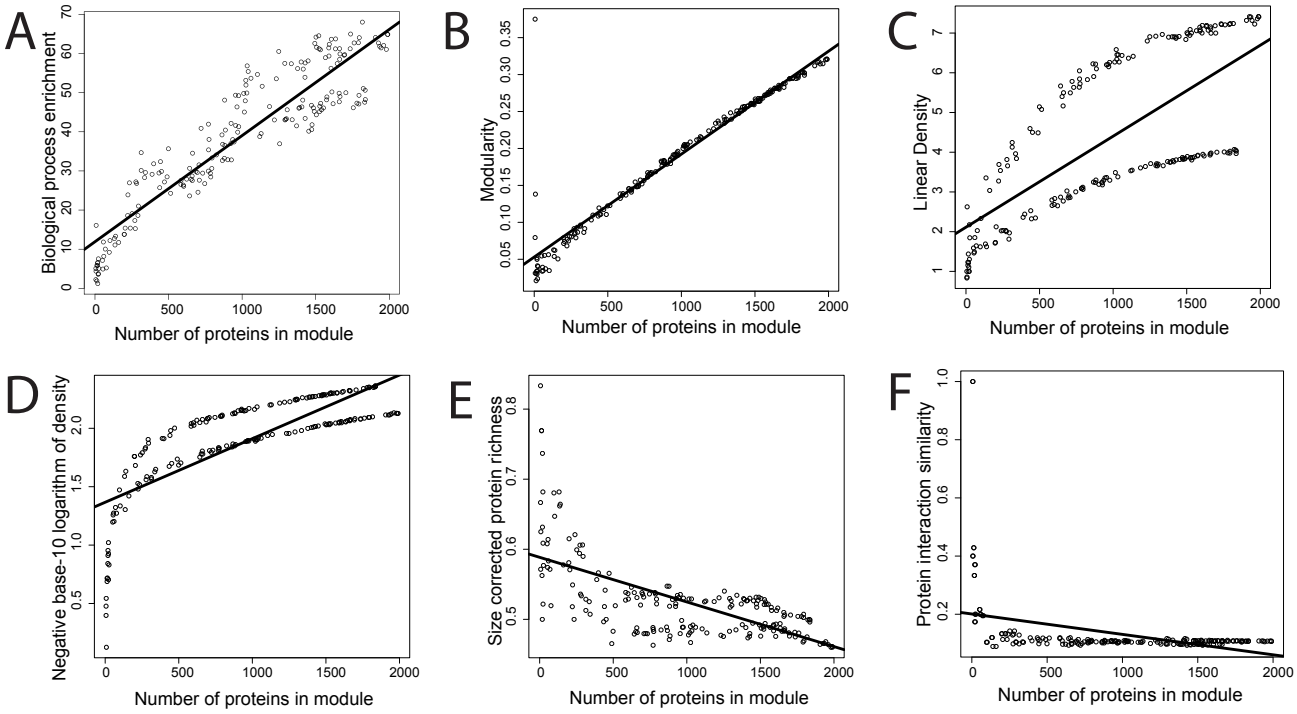


Figure 4.5: Trends by size of 200 randomly generated conserved modules containing between 5 and 2000 proteins each. Least squares best fit lines shown. A: biological process enrichment, B: modularity, C: linear density, D: density, E: size corrected protein richness, F: protein interaction similarity. The modules in A, C, D, and E separate into two curves due to differing characteristics of modules in the human interactome and modules in the Drosophila interactome.

4.7.2 Levels of Homology

That there is extensive homology between proteins in distantly related species is undisputed. Much attention has been given to the search for homology at higher levels such as collections of proteins associated with modules, efforts that have met with considerable success [Kelley *et al.*, 2003; Sharan *et al.*, 2005b; Koyutürk *et al.*, 2006b; Narayanan and Karp, 2007; Hodgkinson and Karp, 2012]. However, at the intermediate level, searching for homology of protein interactions, many studies, including this one, have found extreme limitations. It requires integrating over many protein interactions to obtain any signal of module level homology which makes interaction level similarity a poor predictor of module level homology.

That protein interaction similarity does not significantly predict biological process enrichment poses interesting questions. Is it that there is so little detectable homology at the protein interaction level that it cannot be used reliably to detect a signal of module

homology? Zinman et al. [Zinman *et al.*, 2011] found some evidence that protein interaction similarity was more significant within modules than between modules but their model did not correct for increased density within modules that results from the module search algorithms. Our study shows that evidence of protein interaction similarity leads to modest gains at best if the modules are evaluated in terms of biological process enrichment.

Lack of detectable protein interaction homology may be due either to rewiring of protein interaction networks during evolution [Beltrao and Serrano, 2007] or to artifacts in the protein interaction assays and interaction inference protocols. In the first case, lack of protein interaction homology is an inescapable feature of evolution. In the second, it is a limitation of current technologies. A protein interaction network, as retrieved from public databases, is a mosaic of many experimental assays with many data processing protocols and many often subtle differences in what it means to interact [Koh *et al.*, 2012].

Gandhi et al. [Gandhi *et al.*, 2006] compared protein interaction data in four distantly related species: *Homo sapiens*, *Saccharomyces cerevisiae*, *Caenorhabditis elegans*, and *Drosophila melanogaster*, finding little evidence of protein interaction similarity. Considering the lack of evident similarity among protein interactions across distantly related taxa, Beltrao and Serrano [Beltrao and Serrano, 2007] estimated the rate of rewiring of protein interaction networks to be extremely high with approximately 1000 protein interactions changing in *H. sapiens* every million years of evolution.

Methods that have used higher and lower levels of homology to predict homology of protein interactions have met with much more success. Protein interactions have been predicted with a 30 percent success rate using protein similarity [Matthews *et al.*, 2001], and 40-52 percent success rate [Sharan *et al.*, 2005b] when predictions were restricted to interactions between proteins homologous to proteins in dense complexes.

4.7.3 Module Boundaries and Size Corrections

Number of interactions within a module always increases as the module increases in size so number of interactions is not by itself a useful optimization criterion unless the desired module size is fixed. Modularity, linear density, density, and biological process enrichment seem to provide reasonable criteria for determining optimal module size and composition. As shown in Figs. 4.4 and 4.5, however, biological process enrichment and linear density tend to increase with size in random modules. Modularity tends to increase in very large random modules but is distributed around a constant mean in random modules up to 50 proteins. When the modules are reasonably small relative to the interactomes, modularity provides the ability to compare modules across different sizes in order to choose the best size and composition. Density tends to decrease with size in random modules.

Figs 4.4 and 4.5 provide compelling evidence that for most of the attributes considered, including biological process enrichment, values of the attributes are not directly comparable across modules of different sizes. A set of modules (conserved or otherwise depending on the study) should be generated by a random growth process. The distributions of evaluation metrics for random modules of each size can then be considered a baseline against which more clever algorithms can be compared. This not only corrects for module size but

also provides a realistic null model against which to measure the performance of proposed algorithms. This procedure was used to generate the data in Fig. 4.3.

4.7.4 Implications of Protein Richness Optimizations

That lower protein richness improves biological process enrichment suggests that many biological processes in the cell are implemented by groups of homologous interacting proteins, rather than groups of proteins with diverse origins. Further testing is necessary to determine whether this is a genuine biological discovery or an artifact of the Gene Ontology.

4.7.5 Optimizing for Biological Process Enrichment Directly

When all proteins in the interactomes are sufficiently annotated with biological processes, one way to detect conserved modules is by directly optimizing for biological process enrichment. The conserved modules returned by this algorithm were highly enriched: one, for example, contained 32 of the 34 proteins in the interactome that were annotated with "negative regulation of transcription from RNA Polymerase II promoter", leading to an enrichment P-value of 6×10^{-54}. However, there are proteins that are not annotated or not sufficiently annotated; which modules should these proteins be associated with? Also, what about generalizing to interactomes for which there are few annotations? Furthermore, the running time of optimizing for biological process enrichment may be prohibitive if desiring a full compendium of conserved modules across the interactomes, whereas optimizing for graph theoretic attributes is much faster.

4.7.6 Modularity, Density, and Linear Density

Modularity measures whether the proteins in the module have been associated with each other to a significant extent relative to all protein interaction studies conducted on the proteins. Proteins that are highly studied with interactions to many other proteins require more interactions within the module to associate them with the module to the same extent. Proteins which have few interactions either because they have been poorly studied or because they genuinely have few interactions contribute to high modularity if a significant fraction of their interactions are with other module proteins. Modularity incorporates data from protein interaction studies between the module proteins and proteins in other modules, data that is not used by measures of density.

There has been concern that modularity may not be comparable across modules with different numbers of proteins [Peng, 2012], and Fig. 4.5B provides some justification for this concern. However, as shown in Fig. 4.4B, our results do not indicate any strong correlation between modularity and size in modules up to 50 proteins. The effects described in [Peng, 2012] and demonstrated in Fig. 4.5B are only noticeable at larger size ranges when the number of proteins in the module is a significant fraction of the number of proteins in the interactome.

The high linear correlation between number of interactions and number of proteins shows that the density of modules decreases as their size increases. If density were an important

optimization criterion for predicting biological process enrichment, we should expect the smaller denser modules to be more enriched, but in fact the opposite is observed.

Linear density shows remarkably good performance in predicting biological process enrichment. In the random modules, Fig. 4.2 shows that modularity and linear density together are the best predictors of biological process enrichment. When optimizing for linear density and modularity, Fig. 4.3A shows that the linear density optimized set achieves higher biological process enrichment than the modularity optimized set. Fig. 4.3B shows that the linear density optimized set does not achieve high modularity and Fig. 4.3C shows that the modularity optimized set does not achieve high linear density, confirming that the two measures are complementary for predicting biological process enrichment.

4.7.7 Limitations and Avenues for Further Study

The full model with six attributes explains 37.7 percent of the variance in biological process enrichment, with 35.7 percent of the variance explained by modularity, linear density, and module size alone. The remaining 62.3 percent of the variance is not explained by any of the attributes considered. Other attributes that explain additional variance in biological process enrichment may be discovered from two complementary modeling approaches: models of graph theoretic properties that arise from the physical construction of protein interaction networks, and models of homology that consider how protein interaction networks change during evolution.

Chapter 5

Supporting Software

5.1 EasyProt: Parallel Software Architecture for Experimental Workflows in Computational Biology on Clouds

5.1.1 Overview of EasyProt

EasyProt [Hodgkinson *et al.*, 2012] is a parallel software architecture for acquiring and processing proteomics and interactomics data, and for analyzing results of algorithms that detect conserved multiprotein modules. EasyProt builds on the dataflow concepts from SEDA [Welsh *et al.*, 2001]. In EasyProt, each task, called an element, executes in parallel and passes messages along a DAG in which the elements are vertices. A type system has been developed suitable for proteomics and interactomics data that is used for the messages passed along the DAG. Each message stores a description that identifies the elements through which it and its predecessors have passed, along with any parameters that were used. The user specifies the elements, sets their parameters, and specifies the DAG edges, using a simple graph language that is compiled into Java using ANTLR [Parr and Quong, 1995]. Several message-passing protocols are supported, including broadcast and round-robin.

Elements have been developed for several classes of tasks: obtaining interactomics and proteomics data, managing a cache for intermediate storage, running protein homology detection algorithms, running various algorithms for detecting conserved multiprotein modules and converting to standardized formats, analyzing sets of conserved multiprotein modules, and generating VieprotML for visualization in VieProt as described in Section 5.2.

All steps from data acquisition to final analysis are entirely within the EasyProt framework. This ensures that all algorithms are treated fairly, running on the same data sets and receiving the same analysis. The EasyProt framework allows previously published proof-of-concept implementations for detecting conserved multiprotein modules to be used robustly as practical tools. Currently Produles, NetworkBlast-M [Kalaev *et al.*, 2009], MaWISh [Koyutürk *et al.*, 2006b], and Match-and-Split [Narayanan and Karp, 2007] are fully supported in EasyProt and can be run on any data set with clear algorithmic evaluation

of results.

EasyProt is workflow software for the programmer. EasyProt uses a graph configuration language, modelled after Click [Kohler *et al.*, 2000], that makes it fast and easy to create many experimental workflows on the same elements. Like Conveyor, EasyProt takes advantage of parallelism available in multicore virtual machines by using a multi-threaded approach. EasyProt uses message-passing protocols that can pass arbitrary Java objects, allowing it to scale horizontally, an important feature for cloud computing applications [Khalidi, 2011]. Several message-passing protocols are part of EasyProt, including broadcast and round-robin message-passing, and new protocols can be programmed directly by the element designer. EasyProt supports a data cache for versioning and provenance that is separable from the workflows. In case of failure of a workflow for any reason, intermediate cached results are available that can be used for a modified workflow using data from the cache. The EasyProt system follows modular design, making it easy to modify workflows or to share workflows with other users, and to incorporate new algorithms, new web services, and support for new forms of data as they become available. The focus of EasyProt is on the programmer and the process of development, simplifying adoption of a principled framework for reproducible experimentation.

5.1.2 Previous Systems Research

Systems research in computer science introduced similar modular systems for various applications. Click [Kohler *et al.*, 2000] used a similar architecture to build configurable routers. SEDA [Welsh *et al.*, 2001] extended this architecture to general Internet services with an emphasis on horizontal expansion. Unlike Click, we allow only push semantics which best suits our domain. Given the long running time of the queries and the fast initialization time, it is most reasonable for the queries to be listed in the configuration file and processed immediately using push semantics. We found it unnecessary and difficult to implement pull semantics with our multi-threaded architecture. The requisite stalls would block threads that are needed to receive messages. Elements that are sources in the DAG begin obtaining their data after initialization and the data is pushed through the graph to the sinks. Unlike Click, which was single-threaded, EasyProt uses a multi-threaded model similar to SEDA in which each element waits on a single queue for messages. To maintain the order of the messages received and to avoid conflict among threads attempting to access shared state, each element in EasyProt runs in its own thread rather than using thread pools as in SEDA. Unless an element is performing useful work, it waits for messages from its input queue. There is possibly some additional parallelism that can be exploited by using thread pools which we leave for future work. We use a simple description language to specify the configuration graph, based on the ideas of Click. In contrast to Click, which used a single type of message, we define an extensible type system for proteomics data and allow arbitrary types to flow between elements.

5.1.3 Scientific Workflow Management Systems

The scientific workflow management system [Gil *et al.*, 2007] is a framework that can be usefully adapted to make cloud computing attractive to experimenters in the computational sciences. In a scientific workflow, tasks are connected in a directed graph representing data dependencies, and data flows in parallel along the edges. Scientific workflows generalize ideas from pioneering projects on configurable services [Lord, 1995; Kohler *et al.*, 2000; Welsh *et al.*, 2001]. The strength of the scientific workflow as a cloud programming model is its ability to harness resources as they expand horizontally in the cloud. This computing development model can be easily built on top of Infrastructure as a Service [Armbrust *et al.*, 2009] clouds, providing an attractive development environment for computational biologists.

There have been attempts to extend scientific workflow software into the realm of cloud computing [Taylor *et al.*, 2006], the most notable of which is Pegasus [Deelman *et al.*, 2005; Juve and Deelman, 2010]. Pegasus is designed for large stable scientific applications, and is an excellent choice for this use case. Custom workflows using cloud computing have been designed with other scientific workflow software including Taverna [Hull *et al.*, 2006] and Kepler [Ludäscher *et al.*, 2006]. However there remains a need for a programming framework that allows cost-effective and fast reconfiguration by programmers during scientific experimentation in the cloud. Very recently, a workflow system, Conveyor [Linke *et al.*, 2011], was released that provides a programming model that allows fast configuration during ever-changing experimentation. However, Conveyor does not offer full support for cloud computing, is not released as an AMI, and requires substantial local installation.

5.1.4 Cloud Computing

Cloud computing is set to change the way that bioinformatics research is conducted, improving the cost-effectiveness of massive computations [Stein, 2010]. Cloud computing provides an increasingly attractive alternative for computational biologists as datasets are increasing in size faster than desktop computers are increasing in capacity [Stein, 2010]. With its pay-as-you-go model, new computational tools are needed to make cloud computing an attractive option for scientists conducting computational experiments, to balance speed of applications and development with cost. A primary advantage, and challenge, of cloud computing is its ability for horizontal expansion [Khalidi, 2011]. Single virtual machines typically have several virtual cores while the power of each core remains relatively constant, and multiple virtual machines can be launched on demand.

5.1.5 Abstract Machine Images

Cloud computing opens a new way to share data and workflows, through the abstract machine image or whole system snapshot exchange [Dudley and Butte, 2010]. AMIs are stored in cheap storage, are always available, and can be easily used to develop and maintain multiple versions of software. Sharing of software with AMIs is enormously beneficial as the virtual machines provide complete control over the program execution environment, eliminating issues of portability and dependencies. By releasing EasyProt as an AMI, it

is merely necessary for the user to launch a virtual machine from the AMI to run and modify workflows. Multiple virtual machines can be launched to compute workflows in parallel. Virtual machines can be configured to the needs of the workflows, according to desired specifications for computing capability and memory, and, to a lesser extent, network bandwidth. Releasing workflows as EasyProt AMIs addresses the challenges of reproducible computation [Donoho *et al.*, 2009; Dudley and Butte, 2010], allowing experiments to be readily repeated and verified by others.

5.1.6 Heterogeneity of Biological Data

There have been several attempts to retrieve and process protein interaction data for practical use. Several protocols, including Psicquic [Aranda *et al.*, 2011] and DASMI [Blankenburg *et al.*, 2009] have been developed to make it easy for databases to accept and respond to queries in a standard format. Protein interaction databases that have implemented or are in the process of implementing these protocols include BioGRID [Chatr-aryamontri *et al.*, 2013], DIP [Salwinski *et al.*, 2004], and IntAct [Kerrien *et al.*, 2012]. Psicquic is a protocol developed by the HUPO Proteomics Standards Initiative [Martens *et al.*, 2007] for accessing data from protein interaction databases using a query language MIQL. iRefIndex [Razick *et al.*, 2008] has the goal to remove redundancy in the multiple protein interaction databases and to map all interactions to a consistent format. iRefIndex has implemented the Psicquic protocol so that its data is easily retrieved. Yet, these do not address the issue of allowing arbitrary annotations for the proteins and interactions. DASMI [Blankenburg *et al.*, 2009] does allow protein interaction annotations, but its model does not easily extend to whole-proteome data because of the time cost of retrieving all the various interactions and annotations using its protocols. Also, DASMI allows only static annotations on interactions and proteins. It does not support annotations computed using programs that run dynamically on any data set. For example, in comparative proteomics, required annotations include measures of sequence similarity between all pairs of proteins in any two given proteomes. These annotations are commonly computed using the BLAST program [Altschul *et al.*, 1990].

A standardized data format is necessary for exchanging data between databases. Numerous data exchange formats exist with various goals. BioPAX is a rather bulky format used to share biological pathway data [Demir *et al.*, 2010]. PSI-MITAB and PSI-MI XML are lightweight formats for storing protein interactions and associated annotations in tab-delimited or hierarchical XML formats [Kerrien *et al.*, 2007]. Multiple protein identifier schemas such as SEGUID [Babnigg and Giometti, 2006], NCBI Entrez Gene and Entrez Protein IDs [Wheeler *et al.*, 2006], and UniProtKB IDs [UniProt Consortium, 2012] are used for naming proteins. Most protein interaction databases, including BioGRID, DIP, and IntAct, use a subset of these schemas and do so inconsistently. Data from these databases have been aggregated and somewhat normalized as part of the iRefIndex effort. However, iRefIndex faces the challenge that the original database licenses often prohibit redistribution. Our framework can be used as a shared and easily extensible implementation of a service such as iRefIndex. As the end users themselves would obtain and process the data

using this tool, there is no redistribution or limitations based on redistribution.

5.1.7 Division of Labour

The EasyProt framework can be easily used by researchers in any well-defined research area with clearly defined data collection tasks, algorithm execution tasks, and standard methods for analyzing results. This approach allows for a division of labor between algorithm design and data processing tasks such as data acquisition, data annotation, and analysis of results. Furthermore, it allows for easy sharing of standardized programs to perform these tasks and avoids repeated work by multiple researchers.

5.1.8 EasyProt Software Architecture

EasyProt provides a programming model and development environment for computationally demanding experimental workflows in computational biology. A workflow is represented by instances of elements running in parallel across many virtual cores, connected into a directed graph representing data dependencies using a simple graph configuration language. During development of workflows on predefined elements, the elements are connected using the graph configuration language while ensuring that the types of data flowing along the connections satisfy the element specifications, and processed data is retrieved from the cache. When designing new workflows, computational biologists create new elements by implementing a clean interface using an API that hides the parallelism and details of the dataflow. During the process of developing and modifying workflows, the modified AMIs, including all intermediate data, can be stored in stable cloud storage, with no need for transferring large datasets over the network.

There are three fundamental components to EasyProt. Users of an application using pre-defined Elements must learn only to use the Graph Specification Language and to ensure that the types of data flowing along the connections satisfy the Type System. Programmers create new Elements by implementing a simple clean interface using an API that hides the parallelism and all details of the dataflow.

5.1.8.1 Elements

EasyProt elements each perform a well-defined task. Some of these tasks require accessing external web services to acquire data. Some require running external programs. Others are data processing elements that apply functions to the data before passing it on. Element instances are connected in a directed graph that represents the dependencies among the element instances. Each element instance runs in its own thread, supporting parallelism across multiple virtual cores. The time required to complete the processing is the time of the longest path from a source to a sink in the directed graph, where the length of a path is given by its running time, with the goal being that this time is limited only by the data dependencies.

An element is represented by an abstract Java class with the core functionality hidden behind a clean interface and shared among all elements. Defining a new element for

the user's own needs requires writing two abstract functions in the abstract Java class: an initialization function and a task function. Most of the code necessary for defining a new element is task specific. The element designer is presented with a clean API for retrieving and sending messages through the graph. Instructions for defining new elements are contained on the AMI in well-documented code and examples.

5.1.8.2 Type System

Elements are free to pass custom types along the graph. EasyProt wraps every message in a wrapper type, transparent to the user, before sending it along the graph, providing a uniform interface for messages. A collection of types has been defined, designed for workflows in proteomics, interactomics and comparative genomics. `Proteome` and `Interactome` are generic types that can store proteins and interactions in various formats. A collection of types that can be stored in proteomes and interactomes have been defined, including various protein identifier schemas, annotated proteins, protein amino acid sequences, and protein interactions in various formats. For many elements in the sample workflows described in Section 5.1.8.5, proteomes and interactomes are the units of data passed through the configuration graph; however, some elements combine proteomes or interactomes and release data in a custom output format. Message types have been defined for these custom formats. Designers of new elements are free to use the predefined types or to define new customized types most suitable for their data.

5.1.8.3 Graph Configuration Language

Fig. 5.1 lists a sample graph configuration for a workflow using the elements defined in Section 5.1.8.5. Each configuration file consists of two sections: an initialization section and a connections section. The initialization section defines each instance of the elements that are to be used along with their jobs and initialization strings. The connections section defines the connections among the element instances. For each statement in the connections section the first element instance is connected directly upstream of all other element instances appearing in the statement. Each element defines the types of the elements that can appear directly upstream and directly downstream of it. The elements used in the sample graph configuration are defined in Section 5.1.8.5, and described in more detail, with their jobs and initialization string semantics, on the AMI. The language is implemented as a grammar in ANTLR [Parr and Quong, 1995], a Java-based parser generator, and it can be easily extended to accommodate future enhancements.

5.1.8.4 Cache

EasyProt includes a cache that facilitates reuse of intermediate results. The cache separates the data from the workflows. Each object passed as a message includes a description that records key elements through which the message has travelled. These descriptions are written in string format both as file names with a timestamp and internally. The timestamps allow versioning of the data and the descriptions store the provenance, which

```
INITIALIZATION

Interactome                int1    "iRefIndex\t9606\tall"      ""
Interactome                int2    "iRefIndex\t7227\tall"      ""
MitabToCache               mtc     "2"                         ""
InteractomeFilter          ifil    "2"                         ""
InteractomeToCache         itc     "2"                         ""
ProteinLinearizer          plin1   "2"                         ""
ProteinLinearizer          plin2   "2"                         ""
Sequence                   seq1    "1"                         ""
Sequence                   seq2    "1"                         ""
SequenceToCache            stc     "2"                         ""
Blast                      blas    "1.0E-9:inter"              "/easyprot/workspace/blast/blast1/"
BlastToCache               btc     "1"                         ""
InteractomeDegree          degr    "2"                         ""
Go                         go      "2"                         ""
MergeAnnotations           merg    "2"                         "2"
AnnotationsToCache         atc     "2"                         ""
Produles                   prod    "P\t5\t20\t1.5\t0.05\t50"   "/easyprot/workspace/produles/"
ModuleAlignmentToCache     matc    "1"                         ""
Time                       time    "1"                         ""
Modularity                 modu    "1"                         ""
Size                       size    "1"                         ""
Density                    dens    "1"                         ""
Overlap                    over    "1"                         ""
Evolution                  evol    "1"                         ""
Coverage                   cove    "1"                         ""
Components                 comp    "1"                         ""
Summary                    summ    "1"                         ""
AnalysisTableToCache       attc    "9"                         ""
VieprotML                  vpml    "1"                         "forward"

CONNECTIONS

int1    mtc     ifil    ;
int2    mtc     ifil    ;
ifil    itc     plin1   plin2   degr    prod    modu    cove    ;
plin1   seq1    seq2    ;
plin2   go      ;
seq1    stc     blas    ;
seq2    stc     blas    ;
blas    btc     prod    ;
degr    merg    ;
go      merg    ;
merg    atc     vpml    ;
prod    matc    time    modu    size    dens    over    evol    cove    comp    vpml    ;
time    attc    summ    vpml    ;
modu    attc    summ    vpml    ;
size    attc    summ    vpml    ;
dens    attc    summ    vpml    ;
over    attc    summ    vpml    ;
evol    attc    summ    vpml    ;
cove    attc    summ    vpml    ;
comp    attc    summ    vpml    ;
summ    attc    vpml    ;
```

Figure 5.1: Graph configuration for a sample workflow

allows repeatability. Special elements control reading from the cache and writing to the cache. Caching elements have been defined for all message types currently supported, and it is easy to define new caching elements for custom message types. The cache is organized into subdirectories corresponding to message types. As seen in Fig. 5.1, experimenters decide which data is cached by calling the appropriate caching elements.

5.1.8.5 Sample Workflows

To demonstrate the usefulness of the EasyProt system, we designed a collection of workflows for comparative interactomics [Kiemer and Cesareni, 2007], defining elements for acquiring and annotating protein data and comparing algorithms. A protein interaction network is an undirected graph with nodes that are proteins, and edges that indicate protein interactions between the endpoints. An interactome is a large protein interaction network that ideally includes all protein interactions on a species's proteome [Klipp *et al.*, 2009]. Comparative interactomics is the comparison of interactomes across species. A multiprotein module is a collection of proteins that work together to perform a common task, for example, a protein complex or a signaling pathway [Hartwell *et al.*, 1999; Hodgkinson and Karp, 2012]. Given two interactomes, G_A and G_B, a module alignment is a set of possibly-overlapping multiprotein modules in G_i for $i \in \{A, B\}$; a one-to-one mapping from the multiprotein modules in G_A to the multiprotein modules in G_B; and a many-to-many mapping between the proteins in aligned modules. The sample workflows acquire current high-quality protein data for multiple species, and apply and analyse algorithms to detect multiprotein modules conserved in these species.

Several key elements designed for these workflows are defined below.

1. `Interactome`: acquires interactome datasets from various sources.
2. `InteractomeFilter`: removes extraneous data and simplifies the format for later computation.
3. `ProteinLinearizer`: converts interactomes to proteomes.
4. `Sequence`: obtains amino acid sequences for the proteomes from external web services.
5. `Blast`: performs pairwise protein sequence similarity comparisons between proteomes.
6. `InteractomeDegree`: computes the number of interactions for each protein in an interactome.
7. `Go`: obtains protein function annotations.
8. `Produles`: finds multiprotein modules conserved during evolution using the Produles [Hodgkinson and Karp, 2012] algorithm. Other algorithms are placed in similar elements.
9. `MergeAnnotations`: accepts the output from several elements and produces a unified annotated proteome.

Dataset	EC2 Large	EC2 High-Memory Quadruple Extra Large
H. sapiens vs. *D. melanogaster*	7h 32m	7h 18m
D. melanogaster vs. *S. cerevisiae*	7h 22m	7h 9m
S. cerevisiae vs. *H. sapiens*	7h 21m	7h 14m
Complete	7h 32m	7h 18m

Table 5.1: Time to run sample workflows derived from Fig. 5.1. The workflows run in parallel so the total time required for all datasets is the maximum of the timings. The EC2 Large instances are applied with 6GB of available RAM and the EC2 High-Memory Quadruple Extra Large instances are applied with 65GB of available RAM.

10. `Time`, `Modularity`, `Size`, `Density`, `Overlap`, `Evolution`, `Coverage`, `Components`: compute module alignment evaluation statistics [Hodgkinson and Karp, 2012].

11. `Summary`: compiles statistics on the module alignment for visualization.

12. `VieprotML`: generates output suitable for visualization of module alignments.

Using workflows similar to Fig. 5.1, multiple algorithms to detect multiprotein modularity conserved during evolution were compared and evaluated on current datasets [Hodgkinson and Karp, 2012].

5.1.9 Timings

Versions of the sample workflow in Fig. 5.1 were applied to the species *Homo sapiens*, *Drosophila melanogaster*, and *Saccharomyces cerevisiae* with interactome sizes: *H. sapiens*: 12,952 proteins, 71,035 interactions; *D. melanogaster*: 10,192 proteins, 41,050 interactions; *S. cerevisiae*: 6,083 proteins, 175,113 interactions. Three versions, *H. sapiens vs. D. melanogaster*, *D. melanogaster vs. S. cerevisiae*, and *S. cerevisiae vs. H. sapiens*, were run in parallel on six separate EC2 instances with two configurations of processing capability and memory capacity. Table 5.1 lists the running times. As the workflows run in parallel, the total time required is the maximum of the individual timings.

Interestingly, the time required for the virtual machines with vastly differing amounts of computing capabilities and memory is quite similar. This seems to be due mainly to the large amount of network requests from web services in these workflows that are limited by the speed of the external servers rather than the computing capability, amount of memory, or network bandwidth of the virtual machines.

The more powerful virtual machines are faster for computationally intensive workflows. To demonstrate this, Table 5.2 compares the timings on variants of the sample workflow from Fig. 5.1 that use data from the cache rather than retrieving data through the network. For these computationally demanding workflows, the more powerful virtual machines require less than three quarters of the time required by the less powerful virtual machines.

Dataset	EC2 Large	EC2 High-Memory Quadruple Extra Large
H. sapiens vs. *D. melanogaster*	70m 38s	51m 14s
D. melanogaster vs. *S. cerevisiae*	17m 16s	23m 47s
S. cerevisiae vs. *H. sapiens*	21m 28s	29m 13s
Complete	70m 38s	51m 14s

Table 5.2: Time to run sample workflows using data from the cache with the same virtual machines and settings as Table 5.1.

Species	Proteins	Interactions
Homo sapiens	13,065	74,554
Drosophila melanogaster	10,050	40,004
Saccharomyces cerevisiae	5,965	102,147

Table 5.3: Sizes of the three filtered protein interaction networks used for the results.

The Human vs. Fly data set required more than twice as much time as the other data sets. The most important reason is that the time of this particular graph configuration is dominated by the time for the BLAST element which makes $n_1 \times n_2$ comparisons where n_1 is the number of proteins in species A and n_2 is the number of proteins in species B. The sizes of the three protein interaction datasets are listed in Table 5.3 from which we can see that the Human vs. Fly data set has the largest value for this product. Another possible contributing factor is that the particular EC2 instance used for the Human vs. Fly data set may have been sharing computing resources with other compute-intensive instances. Amazon guarantees a baseline level of performance for its instances but if an instance is not competing for computing resources on the same physical machine, Amazon allows it to benefit from improved performance. This leads to variable performance from instance to instance, though all instances meet the baseline performance.

This is a vast improvement over manual development of scripts to obtain the data. Though defining new elements requires nearly as much time as writing a custom script for the same task, the element is standardized so it can be easily shared with the community, executed in a high-speed parallel environment, and extended by others. With elements for additional common tasks, EasyProt has the potential to become a community standard for data retrieval, simplifying tedious data collection tasks for many researchers.

5.1.10 Perspectives and Discussion

The power of the EasyProt software architecture is its parallel message-passing design that allows the computation to be distributed across multiple virtual cores and multiple virtual machines. In EasyProt, elements have clearly defined roles and pass messages to other

elements allowing for direct horizontal expansion. EasyProt is a programmer-oriented model for reproducible computation in the cloud that supports a parallel element-based approach for using multiple virtual cores and parallel deployment from an AMI for using multiple virtual machines. Enhancements are being implemented for automated partitioning of a single workflow across multiple virtual machines, when desired by the workflow developer, with objectives to minimize message passing across virtual machine boundaries and to group expensive elements with inexpensive elements in order to balance the load.

A clear advantage of the text-based graph configuration language is that it is easy to understand and manipulate directly, allowing for direct modification through a console during the design of ever-changing experiments. For any cost-effective cloud computing programming model, it is essential to minimize transfer of data over the network. The EasyProt model offers immediate availability from any virtual machine running the AMI with no local installation. We are investigating the merits of a hybrid programming environment where flexible console-based access is supplemented with graphical tools that run locally.

MapReduce [Dean and Ghemawat, 2008; Goncalves *et al.*, 2012] is a method of parallelization that may increase efficiency for particular highly repetitive tasks. MapReduce elements can be directly implemented in the EasyProt framework. However, preexisting systems such as Hadoop [Bialecki *et al.*, 2012] are highly optimized for MapReduce. If a programmer desires to use MapReduce within the EasyProt framework, the recommended option is to design an element that passes data to a MapReduce optimized system.

Comparing the cost of development in the cloud with the cost of development on a high-performance locally managed cluster is not straightforward, as there are hidden costs associated with purchasing and maintaining any cluster. A discussion of cost effectiveness with references to detailed studies, and the conclusion that development in the cloud is cost effective, can be found in [Stein, 2010].

We felt it important to leverage the powerful computing resources available through cloud computing. With compute-intensive annotation tasks such as `blastall` and protein interaction confidence scoring, we require computer resources that are costly to maintain. Economies of scale leads to lower cost for computing resources through the cloud which allows execution of many compute-intensive tasks in parallel on high-speed computers at very low cost.

The language we are using to express configurations is limited in some ways. The data type flowing on the edge is determined by the endpoints of the edge. It is conceivable that one element may be able to send two different types of data to a second element. It is possible that the language should be extended to allow this.

Finally, we plan to allow partitioning of the elements among multiple EC2 instances, grouping expensive elements with inexpensive elements, in order to balance the load among multiple machines and speed the computation. The disadvantage of this approach is the increased cost of launching multiple EC2 instances for each data set.

The goal of this work is to increase productivity for researchers in computational biology. While data sources and access protocols are constantly changing, there is no reason for every researcher in the field to be concerned about this. To perform useful work, it is

merely necessary to specify what data is desired and then to retrieve that data for the desired study. If a protocol for data acquisition is modified, only one person must update the corresponding element in EasyProt, rather than every researcher using this data. Even for those who may prefer to write their own scripts, having access to the code for the EasyProt elements may simplify their coding tasks. Hopefully, after seeing the simplicity and advantage of using the EasyProt framework, they will use EasyProt for their data collection and annotation needs and will contribute new elements to the community. We expect this system for protein interaction data acquisition and annotation to be extended beyond protein interaction data into the domain of computational and genomic biology as a whole and to become a useful resource for the community.

Java was chosen as our implementation language because of its built-in support for multi-threaded programming and its ease of use. Our goal was to make the task of defining new elements as simple as possible for the element designer. Java is used by many other cloud applications and frameworks including Apache Hadoop (Bialecki et al. 2005).

5.1.11 Using EasyProt

EasyProt is distributed as a public AMI that runs on Amazon EC2. The AMI is registered with manifest easyprot-ami/easyprot.img.manifest.xml. Upon launching the AMI, one can type from any directory "easyprot X configFile" to launch EasyProt, where configFile is the name of a configuration file stored in the directory /easyprot/configurations, and X is the amount of memory available to EasyProt in the format nT where n is an integer and $T \in \{m, g\}$, where m specifies megabyte and g specifies gigabyte. Thirty sample workflow configuration files are provided including the sample workflow in Fig. 5.1. Instructions are also provided in the welcome message on the AMI. Documentation explaining how to use the elements currently defined is provided in the folder /easyprot/documentation and in the source Javadoc. EasyProt, including all source code, is released under the GNU General Public License, Version 3.

The directory structure on the AMI is as follows:

- easyprot
  - cache: contains the EasyProt cache
  - config: contains 30 sample workflow configuration files including Fig. 5.1
  - documentation: contains documentation for the elements also found in the source Javadoc
  - language: contains the ANTLR grammar and code
  - license: contains open source license
  - project: contains the project with Java source code in subdirectory src
  - scripts: contains the easyprot script that compiles the configuration file, compiles the project, and launches EasyProt
  - workspace: contains temporary workspace and executables for external programs

At the time of this writing, EasyProt supports 61 elements. Documentation for using the elements in the most current version, with allowable connections and specification of jobs and initialization strings, can be found in the file /easyprot/documentation/README on the AMI and in the source Javadoc.

5.2 VieProt: Visualizing Conserved Multiprotein Modularity with a Dynamic Force-directed Layout

VieProt [Hodgkinson and Kong, 2012] is a tool for visualizing conserved multiprotein modules with a dynamic force-directed layout, that accepts data in a custom XML format, VieprotML, generated by EasyProt. VieProt is an enormous improvement on the most current alternate visualization tools such as VANLO [Brasch *et al.*, 2009]. With VieProt, it is easy to evaluate visually new and old algorithmic ideas for detecting conserved multiprotein modularity. Proteins, interactions and interaction sources, protein sequence similarities, GO annotations [Gene Ontology Consortium, 2000], and algorithmic measures of quality are displayed in a visually-appealing format. An image from VieProt appears in Figure 5.2.

The purpose of EasyProt [Hodgkinson *et al.*, 2012] is to allow computational biologists to easily acquire a large number of datasets with associated annotations for the purposes of testing various algorithms involving protein interactions. These algorithms run on networks of tens or hundreds of thousands of interactions, so making sense of the results can be difficult. Visualization is an effective way of facilitating sense making, as it takes advantage of the high bandwidth of the human visual system [Card *et al.*, 1999; Ware, 2004]. We have developed a visualization tool, VieProt[1], to visualize the results of the algorithms in order to help identify their effectiveness or flaws. The dataflow we implemented gathers protein interaction data and runs a protein alignment algorithm, so we designed VieProt to specifically aid the evaluation of the correctness of these algorithms. VieProt is implemented in Java using the `prefuse` visualization library [Heer *et al.*, 2005].

5.2.1 Previous Work

Several visualization tools have been developed for visualizing proteomics data. Cytoscape [Shannon *et al.*, 2003] is a popular node-link diagram visualization tool originally designed for biologists. It has been extended with multiple plug-ins that implement various layout algorithms and interactions. Osprey [Breitkreutz *et al.*, 2003] is a tool that visualizes protein interactions, and interfaces directly with the Gene Ontology (GO) [Gene Ontology Consortium, 2000], a database containing functional descriptions of proteins and other gene products. Visant [Hu *et al.*, 2009] is another protein interaction visualization tool that provides a way to browse networks at different levels of abstraction. However, none of these tools allow visualization of protein interaction network alignments nor do they make user interaction an integral part of their workflow. Since protein interaction networks are very

[1]Vie is the pronunciation of the first letter in "visualization" and the French word for life. Thanks to Javier Rosa for suggesting the name.

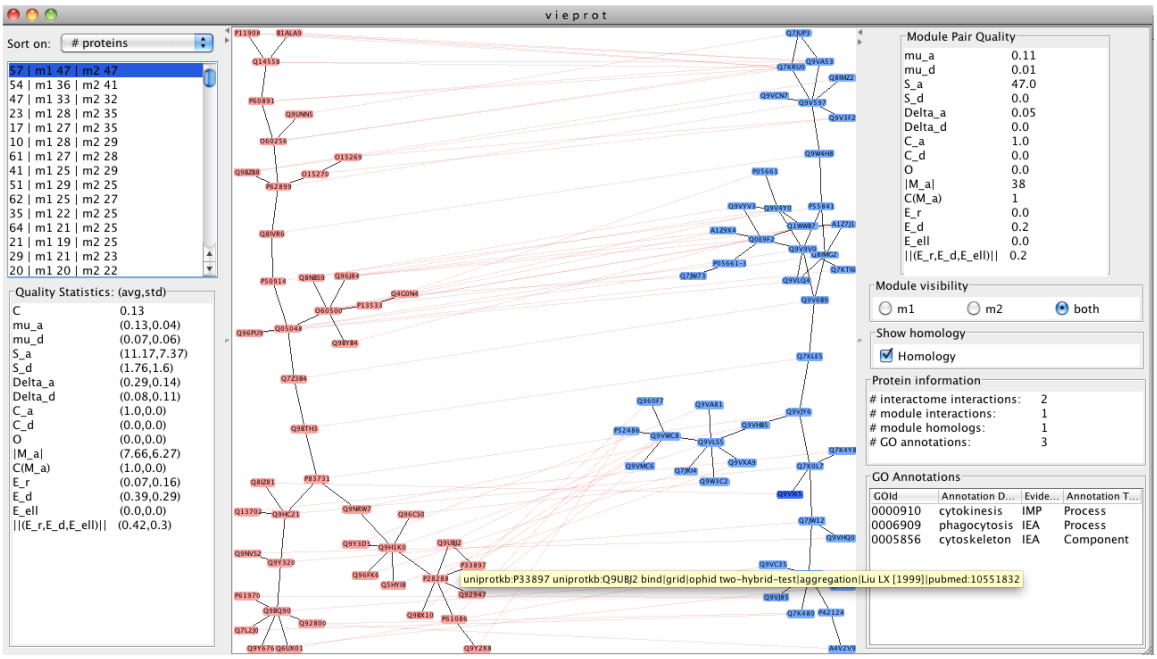


Figure 5.2: VieProt's interface. (a) This panel is the *alignment browser*, where users can view statistics about and sort different alignments. (b) This panel and the rightmost panel comprise the *alignment viewer*. The alignment is laid out using a node-link diagram and the user can select proteins to view their metadata.

large, it is necessary to incorporate filtering and browsing mechanisms to allow users to pick out the information. In addition, it may be the case that other methods of visualization such as line charts or bar charts are more useful to provide various statistics about the data. We thus decided to use the `prefuse` [Heer *et al.*, 2005] information visualization toolkit. It implements many types of layout algorithms for both node-link diagrams and other forms of visualization, and makes it easy to connect interface widgets with the visualization.

Only one tool, VANLO [Brasch *et al.*, 2009], of which we are aware, has been designed for visualizing interactome alignments. Our system is more intuitive to use and provides more information useful for interpreting the meaning of the conserved multiprotein modules.

5.2.2 Visualization Interface and Data Display

VieProt displays the aligned modules and provides access to the protein metadata provided by EasyProt. The visualization interface is shown in Figure 5.2. It has been designed to enable analysts to determine the effectiveness of their interactome alignment algorithms. The goal of these algorithms is to identify protein modules in each species that may descend from a common ancestral protein module. To determine whether an alignment is biologically plausible, the analyst may want to know the following, among other criteria:

- Number of proteins in the modules of each species. A good alignment should pair modules with similar numbers of proteins.
- The homology relationships. A good alignment should align each protein with a number of homologous proteins that is biologically plausible.
- Topology of the modules. A good alignment should align modules with similar topologies.
- Function of the proteins in each module. A good alignment should find and align modules enriched in proteins that perform a well-defined function.
- Percentage of the interactome in each species that is covered by modules.
- Statistical measures of quality described in [Hodgkinson and Karp, 2012].

With the configuration graph in Figure 5.1, EasyProt produces five tab-delimited text files. Two `interactions` files store information about protein interactions within a species. They contain the PubMed IDs of publications describing each interaction (if available) and the database from which the interaction was retrieved. Two `proteins` files each store GO annotations and graph statistics (e.g., degree of a protein in the global protein interaction network) about each protein from a species. Finally, a `produles` file stores conserved module pairs and alignment edges annotated with BLAST scores. A parser takes these five files as input and produces a GraphML [Brandes *et al.*, 2002] file which is parseable by the visualization. Currently the parser is implemented as a Python script that runs on the client machine though in future work we plan to create an EasyProt element to generate the GraphML.

At the left of the visualization (Figure 5.2(a)) is the module browser, with which the user can browse the list of aligned modules. The list displays a brief summary of each module alignment, displaying an identifier, the number of proteins in the modules from each species, and the average alignment degree. The drop-down box at the top of the pane allows the user to sort the module alignments by their identifiers, the total number of proteins in an alignment, and the number of homology relations.

The network diagram and panels to the right comprise the module viewer. This component of the visualization allows users to inspect the topology of alignments and access protein metadata such as GO annotations [Gene Ontology Consortium, 2000]. The alphanumeric string in each node is the database identifier of the protein, and the color of each node corresponds to the species from which the protein derived. The gray links are protein interactions, whereas the red links are homology relations. The user can see the exact homology scores and interaction sources by hovering over an edge and triggering a tooltip.

The panels to the far right provide control over the visualization and allow the user access to protein metadata. The network diagram uses an interactive force-based layout and the top three subpanels provide control over the force parameters. The module visibility filter allows the user to view the proteins comprising just one module or both, which may

be useful given large modules. The user can toggle alignment edges on and off; turning off alignment edges also eliminates the forces they exert on nodes. The user can see a protein's metadata by clicking on a protein in the visualization. This populates the *protein information* and *GO Annotations* boxes. The protein information box provides information about the protein, such as the degree of the protein in the species's interaction network and the number of GO annotations. The GO Annotations box is a table containing the GO annotations for the protein which include the GO ID, the annotation description, the evidence code describing the source of the annotation, and the function of the protein.

Chapter 6

Improving Protein Orthology Detection Using Protein Interactions

6.1 Global Network Alignment

It was observed that whereas local network alignment focuses on detecting conserved multiprotein modules, it every protein in a module is forced to align to one other protein in a fashion that minimizes protein interaction disagreement, it may be possible to generate a mapping of orthologous proteins that could be considered functionally orthologous, that is, orthologous proteins that have maintained similar functions [Singh *et al.*, 2007]. This was called global network alignment.

6.2 Graphical Model for Protein Orthology

To gain a better understanding of the evolutionary relationships among disparate species, it is of interest to identify functionally orthologous proteins, that is, proteins across species that are descended from a common ancestral protein with similar function. Early attempts to infer protein functional orthology used only sequence similarity of proteins with the goal to build protein interaction networks [Matthews *et al.*, 2001]. However, experiments revealed that this method had less than a 35% success rate [Matthews *et al.*, 2001]. Some examples where this approach fails are described in Sharan et. al. [Sharan *et al.*, 2005b]. In Section 6.4.1, we develop a graphical model to align protein interaction networks such that finding the best alignment based on experimental data and biologically-reasonable parameters reduces to computing a configuration attaining the mode of the corresponding distribution. The model can be constructed efficiently, scaling linearly with the size of the protein interaction networks and quadratically in the number of species. However, the treewidth of the graphical model is likely to be large for typical instances, making exact mode computation with the junction tree algorithm intractable. In Section 6.3, we survey methods for approximate inference that can be applied to this model.

6.2.1 Input Data

When aligning protein interaction networks, we expect functionally orthologous proteins to have some degree of sequence similarity as, by definition, they have evolved from a common ancestral protein with similar function. Though sequence similarity alone has been shown not to be the best predictor of protein functional orthology [Sharan *et al.*, 2005b], we use sequence similarity as determined by PSI-BLAST [Altschul *et al.*, 1997] to define candidate pairs of functionally orthologous proteins. The PSI-BLAST threshold is chosen in such a way that each protein is possibly functionally orthologous to at most ℓ proteins in any one other species. These relationships are represented by edges connecting proteins that are candidates for cross-species functional orthology. Thus, the input to the network alignment algorithm is a graph $H = (W, F)$ such that $W = \bigcup_{i=1}^{k} W_i$ where W_i is the set of proteins for species i, and $F = F_c \cup F_p$, where F_c is the set of edges defining the candidates for protein functional orthology and $F_p = \bigcup_{i=1}^{k} F_i$ is the set of protein interactions. Let $w_c : F_c \rightarrow (0, 1)$ be the PSI-BLAST sequence similarity score. Weights are also defined on the protein interaction edges corresponding to the estimate that the edge represents a true interaction. Various methods have been proposed for calculating the protein interaction reliability estimates from experimental data [von Mering *et al.*, 2002; Deng *et al.*, 2003; Bader *et al.*, 2004]. In a recent study by Suthram et. al. [Suthram *et al.*, 2006], it was found that the method of Deng et. al. outperformed other methods. The protein interaction reliabilities define a weight function $w_p : F_p \rightarrow (0, 1)$. Let $w = w_c \cup w_p$. As our goal is to infer protein functional orthology by supplementing sequence similarity with information from protein interactions, even a proper subset of all hypothesized protein interactions may be useful for improved inference of protein functional orthology. For reasons that will become apparent in the sequel, it is important to restrict the maximum degree in H. The weight function w_p can be used to remove less reliable interactions until any protein in a protein interaction network interacts with at most δ other proteins. Thus, the degree of any protein in H is at most $\delta + (k-1)\ell$ where k is the number of protein interaction networks.

6.2.2 Model Formulation

The graphical model is defined on a graph $G = (V, E)$ where for each edge $(u, w) \in F$, there is a vertex $v_{uw} \in V$. That is, V contains a vertex corresponding to each edge in the original graph H. Two vertices $u, v \in V$ are connected by an edge in G if their corresponding edges in H share an endpoint. This transformation is illustrated in Figure 6.1. Let V_c be the set of vertices corresponding to cross-species edges and V_p be the set of vertices corresponding to protein interaction edges. A random variable $X_v \in \{0, 1\}$ is associated with each $v \in V$. The setting $X_v = 1$ indicates that v represents either a true protein interaction or a true protein functional orthology, and the setting $X_v = 0$ indicates that v does not represent a true interaction or a true protein functional orthology. A joint distribution over the variables is defined by three classes of potential functions on the graphical model.

1. For each $v \in V$, $\psi_1(x_v) = \begin{cases} w(v) \text{ if } X_v = 1 \\ 1 - w(v) \text{ otherwise} \end{cases}$

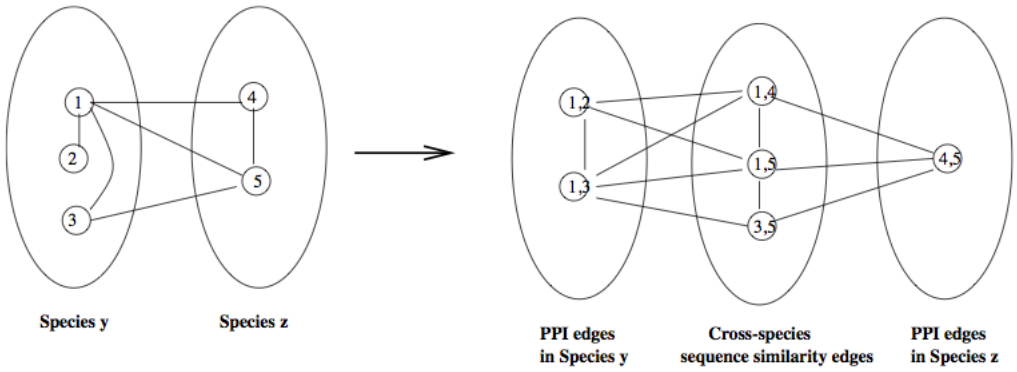


Figure 6.1: Illustration of the graphical model transformation where $k = 2$ (number of species), $\ell = 2$ (maximum degree of a protein w.r.t. edges connecting to another fixed species), and $\delta = 2$ (maximum degree of a protein w.r.t. protein interaction edges)

These potential functions encode the experimental input data, incorporating the protein interaction reliability estimates and the PSI-BLAST sequence similarity scores. The PSI-BLAST sequence similarity scores decrease the probability of proteins being functional orthologous when they have undergone many mutations relative to each other .

2. For each set of vertices $S_{w,z} \subseteq V_c$ consisting of nodes such that, in the original graph H, the corresponding cross-species edges share an endpoint w in species y and each have the property that the other endpoint is a protein in species z, $\psi_2(x_{S_{w,z}}) = r_{y,z,i}$ if $x_{S_{w,z}}$ is a configuration with exactly i 1's.

The values of $r_{y,z,i}$ are parameters of the model, but a reasonable setting would have $r_{y,z,1} > r_{y,z,i}$ for $i \neq 1$, reflecting the biological expectation that in many cases of interest each protein in species y has exactly one functional orthologous protein in species z. The value of $r_{y,z,0}$ should be somewhat small in most cases as this corresponds to the deletion of a protein in species z. The values of $r_{y,z,i}$ for $i > 1$ should be a decreasing sequence in most cases as the value of $r_{y,z,i}$ corresponds to the event that $i - 1$ duplications took place in species z. The values of $r_{y,z,i}$ should be set to reflect the evolutionary divergence between species y and z, and can be thought of as part of an a priori model in a Bayesian setting. Estimation of these parameters from known alignments of small modules is discussed below.

3. For each $t, u \in V_c$ and $v, w \in V_p$ such that the corresponding edges in H form a cycle, where v corresponds to an edge in species y and w corresponds to an edge in species z,

$$\psi_3(x_t, x_u, x_v, x_w, y, z) = \begin{cases} \epsilon_{y,z} \text{ if } x_t = x_u = 1 \\ \quad \text{and } (x_v \text{ xor } x_w) = 1 \\ 1 - \epsilon_{y,z} \text{ otherwise} \end{cases}$$

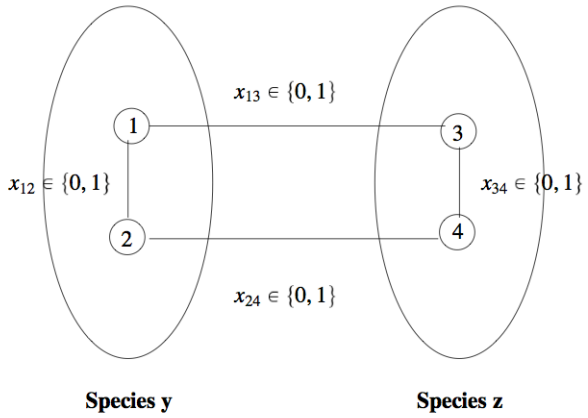


Figure 6.2: The potential function $\psi_3(x_t, x_u, x_v, x_w, y, z)$

This is the most interesting class of potential functions as it combines the information from protein interactions and sequence similarity. Figure 6.2 illustrates the definition of $\psi_3(x_t, x_u, x_v, x_w, y, z)$. The parameters $\epsilon_{y,z}$ should in most cases be somewhat small as $\epsilon_{y,z}$ corresponds to the event of a protein interaction loss in one of the species or a protein interaction gain in the other. As for the $r_{y,z,i}$ parameters, the values of $\epsilon_{y,z}$ are part of an a priori model on protein interaction network alignments in a Bayesian setting and should be chosen according to the evolutionary divergence among the pairs of species being compared. Estimation of these parameters from known alignments of small modules is discussed below in Section 6.2.4. Note that for each $\psi_3(x_t, x_u, x_v, x_w, y, z)$ function, two new edges, (t, u) and (v, w), must be conceptually added to the graphical model. As the actual construction will incrementally create a pairwise Markov random field (MRF) using the method described by Wainwright and Jordan [Wainwright and Jordan, 2008], this is only a conceptual addition.

6.2.3 Complexity of the Construction

A pairwise MRF is constructed incrementally from the original protein interaction networks using a slight modification of the method given by Wainwright and Jordan [Wainwright and Jordan, 2008]. In particular, we begin with the vertex set V and the single-node potentials $\psi_1(x_v)$. Then we add the $\psi_2(x_{S_{w,z}})$ and $\psi_3(x_t, x_u, x_v, x_w, y, z)$ functions incrementally. To add the function $\psi_2(x_{S_{w,z}})$, we create a new node $v_{S_{w,z}}$, connecting it to every $u \in S_{w,z}$. We associate with this new node a random variable that takes values in $\{0, 1\}^{|S_{w,v}|}$. The original $\psi_2(x_{S_{w,v}})$ function is then set as the single-node function for $v_{S_{w,z}}$ and with each edge $(u, v_{S_{w,z}})$, the potential function $\psi(x_u, x_{v_{S_{w,z}}}) = \mathbb{I}(x_u = x'_u)$ where x'_u is the component of $x_{v_{S_{w,z}}}$ corresponding to x_u. The single-node function for $v_{S_{w,z}}$ can be computed from a single copy of $r_{y,z,i}$ stored globally, with the correct value being chosen by examining the number of

1's in the argument. We then perform an analogous procedure to create nodes representing the $\psi_3(x_t, x_u, x_v, x_w, y, z)$ functions, with only one global copy of the $\epsilon_{y,z}$ parameters. Thus, each node can be stored with a minimal amount of space. Globally, only $(\ell + 1)\binom{k}{2}$ numbers need be stored to define all potential functions of the form $\psi_2(S_{w,z})$ and only $\binom{k}{2}$ numbers need be stored to define all potential functions of the form $\psi_3(x_t, x_u, x_v, x_w, y, z)$. The space complexity thus depends primarily on the number of nodes which is given by $|V| + |\{\psi_2(S_{w,v})\}| + |\{\psi_3(x_t, x_u, x_v, x_w, y, z)\}|$ and the number of edges which is upper-bounded by $|V| + \ell|\{\psi_2(S_{w,v})\}| + 4|\{\psi_3(x_t, x_u, x_v, x_w, y, z)\}|$. By the processing of the input, it is guaranteed that $|V| \leqslant \frac{1}{2}|W|(\delta + (k-1)\ell)$ as each protein has a maximum degree of $(\delta + (k-1)\ell)$. There are at most $|W|(k-1)$ sets $S_{w,v}$ as each set is determined by a choice of protein w and species $z \neq y$. There are at most $\lceil\frac{1}{2}|W|(k-1)\ell\rceil\delta\ell$ functions of the form $\psi_3(x_t, x_u, x_v, x_w, y, z)$ as each cross-species edge has at most δ protein interaction edges that share its endpoint in species y and at most ℓ edges that cross to species z. Thus, the number of edges is upper-bounded by $\frac{1}{2}|W|(\delta+(k-1)\ell)+\ell|W|(k-1)+4\lceil\frac{1}{2}|W|(k-1)\ell\rceil\delta\ell < 4|W|(k-1)\ell^2\delta$. A nice feature of this construction is that its size grows only quadratically in k in contrast to some state-of-the-art protein interaction network aligners that create graphs of size exponential in k [Sharan *et al.*, 2005b; Kelley *et al.*, 2003]. This has been one of the major criticisms of NetworkBLAST and related methods (e.g. see Flannick et. al. [Flannick *et al.*, 2006]), as they do not scale easily to multiple network alignment of many species. Some recent attempts have been made to mitigate this exponential dependence on k, such as the $O(2^k)$ algorithm of Kalaev et. al. [Kalaev *et al.*, 2009], which substantially improves on the earlier $O(|W|^k)$.

6.2.4 Parameter Estimation

Flannick et. al. [Flannick *et al.*, 2006] proposed using conserved functional modules in the Kyoto encyclopedia of genes and genomes (KEGG) [Kanehisa and Goto, 2000] as a gold standard for testing protein interaction network alignment algorithms. These conserved functional modules can be used to estimate biologically-reasonable parameters for our graphical model. Recall that the parameters of the graphical model are the values of $r_{y,z,i}$ and $\epsilon_{y,z}$ for $i = 0, 1, 2, ..., \ell$ with y, z ranging over all pairs of species. By considering the smaller protein interaction networks induced by the functional modules in KEGG for which we know the correct alignment, we can estimate the values of $r_{y,z,i}$ and $\epsilon_{y,z}$. The simplest method would be to take counts of duplications, deletions, and protein interaction gains/losses between species y and z in these modules, and then to set the parameters according to these counts. To avoid testing alignments using the same data used to fit the parameters, we can base the tests for our models on the Gene Ontology (GO) hierarchy [Gene Ontology Consortium, 2000]. Alternately, we can use cross-validation on the KEGG data, fitting the parameters from one portion of the data and testing the aligners on the rest of the data, taking the average over different partitions of training and test data.

6.3 Survey of Algorithms for Approximate MAP Estimation

6.3.1 Introduction

A number of combinatorial problems, including maximum weight matching [Bayati *et al.*, 2005; Bayati *et al.*, 2007] and independent set [Sanghavi *et al.*, 2009], have been formulated as graphical models such that the mode of the corresponding distribution is attained by the desired configuration. A wide variety of algorithms, both exact and approximation, have been developed for finding these maximum a posteriori (MAP) configurations. This section contains a survey of some of these algorithms with the goal of finding a good method for protein interaction network alignment. All of these methods apply to pairwise MRFs with discrete random variables taking values in a finite set.

6.3.2 ICM

Perhaps the simplest method for approximately solving the MAP problem is the method of iterated conditional modes (ICM) [Besag, 1986]. ICM iterates through the nodes and for each node, in turn, chooses the value that maximizes the conditional probability of the random variable at that node given the current configuration of its neighbors. Convergence to a local maximum always occurs, in the sense that no change to a single node can increase the probability of the configuration. If a local maximum is not sufficient, then more powerful methods are needed.

6.3.3 Loopy Max-product

Max-product can solve the MAP problem exactly on graphical models that have a tree structure. The model for protein interaction network alignment is likely to have high treewidth, so the junction tree generalization to hypertrees is not likely to be tractable. When applied to graphs with cycles, max-product, if it converges, finds a neighborhood maximum, that is, changing the assignment to variables that form no more than a single cycle cannot increase the probability [Weiss and Freeman, 2001]. However, even for simple graphs consisting of a single loop, max-product may fail to converge [Weiss and Freeman, 2001]. Wainwright et. al. [Wainwright *et al.*, 2004] show that for a strictly positive distribution, max-product has at least one fixed point, but there are no guarantees that max-product will converge to this fixed point. Note that the graphical model for protein interaction network alignment will not define a strictly positive distribution as the conversion to a pairwise MRF, using the method described by Wainwright and Jordan [Wainwright and Jordan, 2008], will introduce configurations with probability equal to zero. As the graphical model for protein interaction alignment will have many short cycles, even if max-product converges, the guarantee of a neighborhood maximum may be only a small improvement over a local maximum in the sense of ICM.

6.3.4 Tree-reweighted Max-product

Along with a better understanding of max-product came an improvement to max-product based on decomposing the original distribution into a convex combination of tree-structured distributions [Wainwright *et al.*, 2007]. Given the canonical overcomplete representation $\phi(x) = \{\mathbb{I}(x_s = j), \mathbb{I}(x_s = j, x_t = k)\}$ and an exponential parameter vector $\overline{\theta}$, a distribution ρ on exponential parameters $\{\theta_i(T_i)\}$ such that $E_\rho[\theta_i(T_i)] = \overline{\theta}$ allows us to apply Jensen's inequality to find that

$$\max_{x \in \mathcal{X}} \left\langle \overline{\theta}, \phi(x) \right\rangle \leqslant E_\rho[\max_{x \in \mathcal{X}} \langle \theta(T_i), \phi(x) \rangle] \tag{6.1}$$

By choosing each element of $\{\theta(T_i)\}$ to have nonzero components only at nodes and edges in a spanning tree T_i of the graph, each of the maximizations $\max_{x \in \mathcal{X}} \langle \theta(T_i), \phi(x) \rangle$ can be performed exactly and efficiently by max-product. The distribution ρ is a positive distribution on the chosen spanning trees T_i. In order for $E_\rho[\theta(T_i)] = \overline{\theta}$ to hold in general, each edge of the graph must be included in at least one of the T_i. Inequality (6.1) is tight if and only if there is a configuration x' such that $\langle \theta(T_i), \phi(x') \rangle = \max_{x \in \mathcal{X}} \langle \theta(T_i), \phi(x) \rangle$ for all T_i. When such a configuration x' exists, strong tree agreement is said to hold, and x' is a MAP configuration [Wainwright *et al.*, 2007]. Given a fixed $\overline{\theta}$ and a distribution ρ on spanning trees $\{T_i\}$, we can consider the problem of choosing the best $\{\theta(T_i)\}$ that satisfies the constraint $E_\rho[\theta(T_i)] = \overline{\theta}$. This leads to the constrained minimization problem

$$\text{minimize} \quad E_\rho[\max_{x \in \mathcal{X}} \{\langle \theta(T_i), \phi(x) \rangle\}] \quad \text{s.t.} \quad E_\rho[\theta(T_i)] = \overline{\theta} \tag{6.2}$$

It is a minimization problem because we are trying to make the bound in (6.1) tight. The Lagrangian dual of (6.2) is

$$\max_{\tau \in \mathbb{L}} \left\langle \tau, \overline{\theta} \right\rangle \tag{6.3}$$

where

$$\mathbb{L}(G) = \{\tau | \sum_{j \in \mathcal{X}_s} \tau_{s;j} = 1, \sum_{k \in \mathcal{X}_t} \tau_{st;jk} = \tau_{s;j} \forall (s,t) \in E, j \in \mathcal{X}_s\}$$

which by strong duality has the same optimum. Since $\max_{x \in \mathcal{X}^n} \left\langle \overline{\theta}, \phi(x) \right\rangle = \max_{\tau \in \mathcal{M}} \left\langle \tau, \overline{\theta} \right\rangle$, where $\mathcal{M}(G) = \{\tau | \exists p(x) \text{ s.t. } E_p[\phi(x)] = \tau\}$, and since $\mathcal{M}(G) \subseteq \mathbb{L}(G)$, we can view (6.3) as an LP-relaxation of the original MAP problem. If a solution to the LP-relaxation occurs at one of the integral vertices, which are exactly the extreme points of $\mathcal{M}$ as $\mathcal{M}$ is the convex hull of $\{\phi(x) | x \in \mathcal{X}\}$, then the solution is MAP-optimal. However, the solution to the LP-relaxation may occur at a fractional vertex. To attempt to solve the LP-relaxation in a manner that exploits the graphical structure, Wainwright et. al. [Wainwright *et al.*, 2007] introduced the tree-reweighted max-product message-passing algorithm. After initializing the messages $M^0 = \{M^0_{st}\}$ with arbitrary positive real numbers, for iterations $n = 0, 1, 2, ...,$

the messages are updated as follows:

$$M_{ts}^{n+1} = \kappa \max_{x'_t \in \mathcal{X}_t} \{\exp\{(\}\frac{1}{\rho_{st}}\overline{\theta}_{st}(x_s, x'_t) + \overline{\theta}_t(x'_t)) \frac{\prod_{v\in N(t)/s}[M_{vt}^n(x'_t)]^{\rho_{vt}}}{[M_{st}^n(x'_t)^{1-\rho_{st}}]}\}$$

where ρ_{st} are the edge-appearance probabilities, that is $\rho_{st} = \Pr_\rho[(s,t) \in T]$, and κ is a normalization constant indicating renormalization after each iteration. Wainwright et. al. [Wainwright *et al.*, 2007] then define pseudo-max-marginals as

$$v_s(x_s) \propto \exp\{(\}\overline{\theta}_s(x_s)) \prod_{v\in N(s)} [M_{vs}(x_s)]^{\rho_{vs}}$$

$$v_{st}(x_s, x_t) \propto \exp\{(\}\frac{1}{\rho_{st}}\overline{\theta}_{st}(x_s, x_t) + \overline{\theta}_s(x_s) + \overline{\theta}_t(x_t)) \frac{\prod_{v\in N(s)/t}[M_{vs}(x_s)]^{\rho_{vs}}}{[M_{ts}(x_s)]^{1-\rho_{ts}}} \frac{\prod_{v\in N(t)/s}[M_{vt}(x_t)]^{\rho_{vt}}}{[M_{st}(x_t)]^{1-\rho_{st}}}$$

Each spanning tree is then associated with the distribution

$$p_{T_i}(x) \propto \exp\{(\} \sum_{s\in V} \log v_s(x_s) + \sum_{(s,t)\in E(T_i)} \log \frac{v_{st}(x_s, x_t)}{v_s(x_s)v_t(x_t)})$$

and the $\{\log v_s(x_s), \log(\frac{v_{st}(x_s,x_t)}{v_s(x_s)v_t(x_t)})\}$ are the desired $\{\theta_i(T_i)\}$ that provide the upper bound in (6.1).

Three concepts, ρ-reparameterization, tree consistency, and the optimum specification (OS) criterion, are important for understanding the behavior of these updates. The first concept, ρ-reparameterization, says that

$$E_\rho[\sum_{s\in V} \log v_s(x_s) + \sum_{(s,t)\in E(T_i)} \log \frac{v_{st}(x_s, x_t)}{v_s(x_s)v_t(x_t)}] = \left\langle \overline{\theta}, \phi(x) \right\rangle$$

This ensures that inequality (6.1) applies. The message-passing algorithm maintains ρ-reparameterization as an invariant, which can be verified by direct calculation using the definitions of v_s and v_{st} [Wainwright *et al.*, 2007]. The second concept, tree consistency, uses the concept of a max-marginal, the analogue of a marginal distribution where summation is replaced by maximization. Tree consistency says that the v_s, v_{st} are exact max marginals for each tree T_i when the v_{st} are restricted to the edges in T_i. This was shown to be equivalent to the edgewise consistency condition that $v_s(x_s) = \kappa \max_{x'_t \in \mathcal{X}_t} v_{st}(x_s, x'_t)$ for every edge $(s,t) \in E$ where κ is a normalization constant that can vary from edge to edge. The message-passing algorithm was shown to satisfy tree-consistency at any fixed point [Wainwright *et al.*, 2007]. The third concept, the optimum specification (OS) criterion, which is equivalent to the earlier-mentioned strong tree agreement, says that there is at least one configuration x^{**}

such that $x^{**} \in \arg\max_{x_s} v_s(x_s) \forall s \in V$ and $(x_s^{**}, x_t^{**}) \in \arg\max_{x_s,x_t} v_{st}(x_s, x_t) \forall (s,t) \in E$. If the v_s, v_{st} were exact max marginals for the graph, then the OS criterion would obviously be satisfied with $x^{**} = x^*$ where x^* is the MAP-optimal configuration. For pseudo-max-marginals, however, the OS criterion may fail to hold [Wainwright *et al.*, 2007]. Wainwright et. al. showed that if the OS criterion holds at a fixed-point of the message-passing algorithm, then any such x^{**} is MAP optimal [Wainwright *et al.*, 2007]. Unfortunately, if the OS criterion does not hold at a fixed point, it is possible that the pseudo-max-marginals do not correspond to an exact solution of the LP relaxation in (6.3) [Kolmogorov, 2006].

One practical consideration in using tree-reweighted (TRW) max-product is constructing spanning trees such that every edge appears in at least one tree. Clearly, there need be at most $|E|$ spanning trees where E is the edge set in the graphical model. By our analysis in Section 6.2.3, this implies that for the graphical model for protein interaction network alignment, we need at most $4|W|(k-1)\ell^2\delta$ spanning trees. It may be possible to develop an algorithm that will do better in many cases at finding a small set of spanning trees that collectively include all edges in the graphical model.

6.3.5 Proximal Methods for Solving the LP-relaxation

In subsequent work, Ravikumar, Agarwal, and Wainwright [Ravikumar *et al.*, 2010] developed message-passing algorithms based on proximal minimization schemes using Bregman divergences that always solve the LP relaxation (6.3), with a superlinear rate of convergence under some conditions. They also developed graph-structured rounding schemes for obtaining integral configurations from the LP solutions. The basic idea of the proximal methods is as follows: instead of trying directly to solve the LP-relaxation $\min_{\mu \in \mathbb{L}(G)} - \langle \theta, \mu \rangle$, a sequence of problems of the form

$$\mu^{n+1} = \arg \min_{\mu \in \mathbb{L}(G)} \{ - \langle \theta, \mu \rangle + \frac{1}{\omega^n} D_f(\mu || \mu^n) \} \tag{6.4}$$

is solved instead, where superscripts denote iteration number, ω^n is a positive weight, and D_f is a generalized distance function known as the proximal function. The proximal function converts the original LP into a strictly convex problem for which coordinate descent schemes are guaranteed to converge to the global optimum. Ravikumar et. al. studied three proximal functions, quadratic, weighted entropic, and tree-reweighted entropic, that each take the form of a Bregman divergence. A Bregman divergence can be formed from any continuously differentiable and strictly convex function f that has bounded level sets as follows: $D_f(\mu' || v) := f(\mu') - f(v) - \langle \nabla f(v), \mu' - v \rangle$. If $f(\mu) = \frac{1}{2} \{ \sum_{s \in V} \sum_{x_s \in \mathcal{X}} \mu_s^2(x_s) + \sum_{(s,t) \in E} \sum_{(x_s,x_t) \in \mathcal{X} \times \mathcal{X}} \mu_{st}^2(x_s, x_t) \}$, then a simple calculation from the definition shows that the induced Bregman divergence is the quadratic divergence: $Q(\mu || v) = \frac{1}{2} \sum_{v \in V} ||\mu_s - v_s||^2 + \frac{1}{2} \sum_{(s,t) \in E} ||\mu_{st} - v_{st}||^2$. If $f(u) = \sum_{s \in V} \alpha_s H_s(\mu_s) + \sum_{(s,t) \in E} \alpha_{st} H_{st}(\mu_{st})$ where $H_s(\mu_s)$ and $H_{st}(\mu_{st})$ are the node-based and edge-based entropies, respectively, and $\alpha > 0$ are positive weights associated with nodes and edges, then the induced Bregman divergence is the weighted entropic divergence, $D_\alpha(\mu || v) = \sum_{s \in V} \alpha_s D(\mu_s || v_s) + \sum_{(s,t) \in E} \alpha_{st} D(\mu_{st} || v_{st})$, where $D(\mu_s || v_s)$ and $D(\mu_{st} || v_{st})$ are the KL divergences between the corresponding marginals. If

$f(\mu) = \sum_{s \in V} H_s(\mu_s) - \sum_{(s,t) \in E} \rho_{st} I_{st}(\mu_{st})$, where $H_s(\mu_s)$ is the node-based entropy, $I_{st}(\mu_{st})$ is the edge-based mutual information, and $\rho_{st} \in (0, 1]$, then the induced Bregman divergence is the tree-reweighted entropic divergence, which takes the form of $D_\alpha(\mu||v)$, except that the node entropy weights α are not always positive. By comparison to Bregman projections, the proximal minimization problem (6.4) becomes $\mu^{n+1} = \prod_f(J_f(\mu^n, \omega^n\theta; \mathbb{L}(G))$ where $\prod_f(\gamma, C) = \arg\min_{\mu \in C} D_f(\mu||\gamma)$ and $J_f(\mu, v) = (\nabla f)^{-1}(\nabla f(\mu) + v)$. This minimization problem can be solved by cyclic projections on a decomposed constraint set: $\mathbb{L}(G) = \bigcap_{(s,t) \in E} \mathbb{L}_{st}(G)$ where $\mathbb{L}_{st}(G) \equiv \sum_{x_t} \mu_{st}(x_s, x_t) = \mu_s(x_s)$ is the edge-marginalization constraint for edge (s, t). Ravikumar et. al. discuss the rate of convergence of these methods, showing that under some conditions, the entire algorithm converges at a superlinear rate, that is $\lim_{n \to +\infty} \frac{|f(\mu^{n+1}) - f^*|}{|f(\mu^n) - f^*|} = 0$, where $f^* = f(\mu^*) = -\langle \theta, \mu^* \rangle$ is the optimal value of the LP-relaxation. For the tree-reweighted entropic divergence, a tree-reweighted sum-product solver can also be used to solve the minimization problem (6.4), in which case, the proximal solver is a direct method for approaching the zero-temperature limit of the tree-reweighted Bethe problem, as described by Wainwright and Jordan [Wainwright and Jordan, 2008].

Ravikumar et. al. also developed rounding schemes for obtaining a MAP configuration from an optimal or near-optimal solution to the LP. Both deterministic and randomized rounding schemes were considered. The deterministic rounding schemes considered are 1) node-based where the decision at a node is based on the single-node pseudomarginal, 2) neighborhood-based where the decision at a node also considers the joint pseudomarginals of a node with its neighbors, and 3) tree-based rounding where all pseudomarginals are considered but inference is based on induced spanning tree distributions using a distribution over a set of spanning trees. These deterministic schemes have easily-computed optimality certificates that can prove a solution is MAP-optimal by checking a form of consistency that depends on the scheme being used. Ravikumar et. al. also considered two variants of a randomized rounding scheme: 1) node-based where the configuration at a node is sampled from the single-node pseudomarginal distribution at that iteration, and 2) tree-based where edges are removed from the graph until the graph becomes a forest and then the configurations for the nodes in each tree are sampled according to the tree-based distribution, $p(x_{V_i}; \mu^n(T_i)) = \prod_{s \in V_i} \mu^n(x_s) \prod_{(s,t) \in E_i} \frac{\mu^n(x_s, x_t)}{\mu^n(x_s)\mu^n(x_t)}$. In conjunction with the proximal methods for solving the LP-relaxation exactly, these rounding schemes give a complete method for MAP estimation that should produce a reasonable approximation even when the MAP problem cannot be solved exactly. This method seems to be the most promising for finding an approximate most probable configuration in the graphical model for protein interaction network alignment.

6.4 Structured Learning of Graphical Model Parameters

One shortcoming of Graemlin 2.0 [Flannick *et al.*, 2009] is that it includes no model for the conservation of functional modules. The protein interaction data is uniformly used to penalize protein interaction loss, that is, to penalize the situation in which a pair of con-

served proteins interact in one species but not in the other. This does not take into account the important difference between protein interaction loss when the interacting proteins are part of a functional module, which should be highly penalized, versus when the interacting proteins are not part of a functional module, which should be less highly penalized. A new protein interaction gain or loss in the connections between modules takes place much more frequently than intramodular protein interaction gain or loss. Several studies have identified biological network motifs, frequently occurring subnetworks that represent interaction patterns of functional modules in a variety of species [Koyutürk *et al.*, 2006a; Gerke *et al.*, 2006; Turanalp and Can, 2008]. Many of these motifs are easy to identify computationally. In this section, we present NetAlign, a program that places factors on some important motifs and learns how heavily to penalize violation of their conservation.

6.4.1 Biological Network Alignment as Structured Learning

We formulate biological network alignment as a binary edge-labeling problem. Given two weighted protein interaction networks, $G_i = (V_i, E_i, w_i), i = 1, 2$, where the weights are estimated protein interaction reliabilities, obtained, for example, by one of the methods surveyed in [Suthram *et al.*, 2006], we use measures of protein similarity, such as all vs. all PSI-BLAST with a reasonable threshold, to find cross-species candidates for protein orthology [Altschul *et al.*, 1997]. The resulting dependency network will look similar to that shown in Figure 6.3 where two protein interaction networks are connected by homology edges. The goal is to assign binary labels to the homology edges to distinguish between edges that connect functionally orthologous proteins and those that do not. The selected edges constitute the alignment.

6.4.2 The Scoring Model

We define types of factors that are likely to contribute to the quality of an alignment. These factors provide the ability to reward or penalize various structures in an alignment. For example, one type of factor would be placed on all pairs of occurrences of a motif, one per species, that are connected by homology edges. This gives Netalign the ability to learn a penalty for failure to conserve the motif in its alignment. After tuning the parameters of our factor functions using training data, we return the alignment that maximizes the overall score or at least receives a high score relative to the other possible alignments.

Using combinatorial algorithms, we search the dependency network for occurrences of the various classes of factors. Each class of factors has a factor function, $F_i(S)$. Here i identifies the class of factors and S is the labelled structure that is to be evaluated. Let N be the total number of classes of factors that we have defined. For $i = 1, ..., N$, suppose that our combinatorial algorithms identify M_i structures, $S_1^i, ..., S_{M_i}^i$, that are to be evaluated. The total score of a labeling is then

$$\sum_{i=1}^{N} w_i \cdot \frac{1}{M_i} \sum_{j=1}^{M_i} F_i(S_j^i) = w \cdot \Phi(x, y)$$

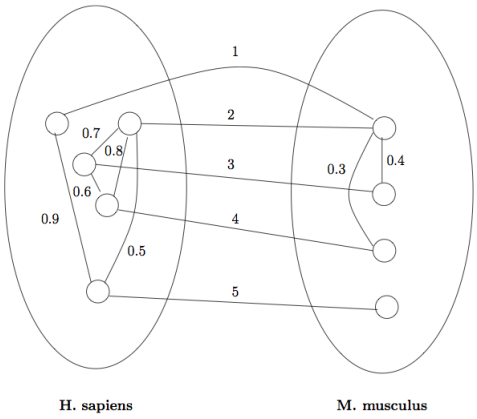


Figure 6.3: The biological network alignment problem. Weights on protein interaction edges reflect reliability of evidence for interaction. Weights on homology edges reflect the amount of protein similarity. The goal is to assign labels in $\{0, 1\}$ to the homology edges. Homology edges that receive a label of 1 are predicted to connect functionally orthologous proteins. In this example, the threshold for inclusion of homology edges is at most 1 and the threshold for inclusion of protein interaction edges is at most 0.3.

where $w = (w_1, ..., w_N)$ are parameters to be learned and

$$\Phi(x, y) = (\frac{1}{M_1} \sum_{j=1}^{M_1} F_1(S_j^1), ..., \frac{1}{M_N} \sum_{j=1}^{M_N} F_N(S_j^N)) \tag{6.5}$$

The notation $\Phi(x, y)$ is chosen to be consistent with the literature on structured learning. The letter x represents the input, that is, the weights on the edges, both protein interaction reliability and protein similarity measures, and the thresholds that together with the weights define the dependency network. Thus, x could represent the network in Figure 6.3. The letter y represents the labeling assigned to the homology edges of the network defined by x. $\Phi(x, y)$ depends on both x and y as our search for structures in the dependency network depends on x and the F_i return values that depend on y, for example, whether labeling y implies the conservation of a motif that we would like to conserve.

Having defined our measure of goodness of an alignment, we have three additional tasks.

1. To find a good alignment y given w and x
2. To decide on a measure of goodness for w
3. To learn a good w from training data

6.4.3 Generalized Viterbi

The first task, which is usually called inference or generalized Viterbi, utilizes methods and software such as those described in [Wainwright and Jordan, 2008; Ravikumar *et al.*, 2010], with extensions to fit the task at hand. The possible solutions to the generalized Viterbi problem are surveyed in Section 6.3. Therefore, we will assume that we have an adequate subroutine for approximate solution of the generalized Viterbi problem.

6.4.4 The Probabilistic Model

Our goal is to model the random process of evolution. If we sample two proteins at random, one from *H. sapiens* and one from *M. musculus*, there is a probability that they descended from the same protein in the MRCA of *H. sapiens* and *M. musculus*, i.e. that they are orthologous. We can measure the similarity of the proteins in various ways, for example by sequence similarity, hydrophobicity patterns, and polarity patterns, which alters the conditional probability that they are orthologous. Using an estimate of this conditional probability, we can try to assign labels, sample by sample, that indicate whether the pair of proteins are functionally orthologous. This procedure has been suggested in the literature [Matthews *et al.*, 2001]. If we had a set of known orthologous proteins we could use it to learn weights specifying how much to reward or penalize the assignment of a label given the similarity measures. We could then use these weights to find the label for a protein-pair sample with highest score, presumably the label with highest conditional probability. In this simplistic model, we have assumed that the protein-pair samples are independent so that the labels can be assigned independently. We thus have a binary classification problem over independent samples. We can imagine using an SVM to learn the weights associated with the various protein similarity measures. We would then obtain a max-margin optimization problem over independent samples.

This is an unrealistic model of evolution where each pair of proteins evolves independently of every other pair, even other pairs that share a protein. If protein MKNK1 in *H. sapiens* has high similarity score with 20+ proteins in *M. musculus*, it is plausible to posit that these proteins are part of the MAP kinase family with highly conserved function. Every once in awhile, over the course of evolution, a protein in the family duplicates but doesn't diverge in sequence. This is more plausible than assuming that all 20+ proteins in *M. musculus* came from the same protein as MKNK1 in the MRCA. That would be a lot of duplications for a single protein on this time scale. With our unrealistic model of evolution, we cannot account for these dependencies between the samples.

Even if two pairs do not share a protein, they may not have evolved independently. Given our understanding of the interactome, we know that proteins group into modules with a given function, such as protein complexes, regulatory networks, metabolic pathways, and signal transduction pathways, and that modules are highly conserved during evolution. Thus, if we have a collection of pairs of proteins to label and only two of the many labelings would result in the conservation of motifs, that is, interaction patterns of common modules, we should probably set the labels in unison to one of these two settings. We can break

the tie by considering protein similarity scores or by comparing to labels of other pairs of proteins. There may be an even higher order motif that can be conserved. Thus, we cannot safely assume that any of our samples are independent. The desired optimization problem for the factor weights is, thus, over entire labelings rather than over independent labels. This setting is called structured learning because the samples have a dependency structure.

Two versions of a maximum-margin optimization problem for structured learning have been proposed [Taskar *et al.*, 2003; Tsochantaridis *et al.*, 2005]. Both versions, for efficiency reasons, assume that the samples can be divided into components such that the samples within a component are dependent only on other samples within the same component. Our probabilistic model of evolution admits no such natural division. We, thus, assume we have a single component. [1] Assuming a single component also simplifies the notation. We shall use this natural simplified notation in the sequel.

6.4.5 The Large Margin Optimization Problem

The goal is to learn weights for factors that separate the true labeling for our training set from all other labelings with a big margin. However, a labeling close to the correct labeling, say one that predicts all of 1000 pairs of proteins correctly but makes one false prediction, is pretty good. It would be unreasonable to try to force a big margin between this labeling and the correct one. On the other hand, a labeling that mispredicts everything should be separated with a huge margin from the correct labeling. In general, a high-loss labeling should be more carefully separated than a low-loss labeling. For biological network alignment, it seems natural to define the loss as the number of mispredictions, that is, the Hamming distance between the labelings, as there seems no reason to give preference to true orthologous proteins predicted nonorthologous over nonorthologous proteins predicted to be orthologous. Thus, we shall exclusively use Hamming distance as our loss function in the sequel. The Hamming distance between labeling y and the correct labeling y^* will be denoted $\Delta(y^*, y)$.

The two forms of the max-margin optimization problem for structured learning, using the terminology from [Sarawagi and Gupta, 2008], are the margin scaling method proposed in [Taskar *et al.*, 2003], and the slack scaling method originally proposed in [Tsochantaridis *et al.*, 2004]. We present both but note that the slack scaling method is not likely to scale well to our problem, though a modification of slack scaling in [Sarawagi and Gupta, 2008] shows some promise.

In margin scaling, incorrect labelings are separated from the true labeling with a margin that grows with the loss. Note that C is a regularization parameter and ξ is a slack variable.

$$\min_{w,\xi} \frac{1}{2}||w||_2^2 + C\xi$$

[1] It would be nice to say that there is no loss of generality as multiple independent components can always be combined into a single component with independent regions, for example, disconnected regions in a factor graph, but it is possible that this may lead to different optimization problems. We leave for future work investigation into whether the optimal parameter vector w may differ in the two formulations.

$$\text{s.t. } \langle w, \Phi(x, y^*)\rangle - \langle w, \Phi(x, y)\rangle \geqslant \Delta(y^*, y) - \xi, \forall y \neq y^*$$

In slack scaling, all incorrect labelings are separated from the true labeling with a fixed margin, but it costs more if a high-loss labeling violates the margin.

$$\min_{w,\xi} \frac{1}{2}||w||_2^2 + C\xi$$

$$\text{s.t. } \langle w, \Phi(x, y^*)\rangle - \langle w, \Phi(x, y)\rangle \geqslant 1 - \frac{\xi}{\Delta(y^*, y)}, \forall y \neq y^*$$

Both of these problems have a constraint for each incorrect labeling. Thus, the number of constraints is exponential in the number of labels. In Section 6.5 we survey several methods that have been proposed for approximately solving these optimization problems.

6.5 Survey of Algorithms for Structured Learning

6.5.1 Primal Descent Algorithm

The simplest method, that used by Graemlin 2.0, is that of Ratliff et. al. for solving the margin scaling problem [Ratliff *et al.*, 2007]. They rewrite the problem in equivalent hinge loss form.

$$\min_{w,\xi} \frac{1}{2}||w||_2^2$$
$$+C \max_{y \in \mathcal{Y} \setminus y^*} (\Delta(y^*, y) - \langle w, \Phi(x, y^*)\rangle + \langle w, \Phi(x, y)\rangle)_+$$

$$= \min_{w,\xi} \frac{1}{2}||w||_2^2$$
$$+C[\max_{y \in \mathcal{Y}}(\langle w, \Phi(x, y)\rangle + \Delta(y^*, y)) - \langle w, \Phi(x, y^*)\rangle]$$

after some rearranging, as $y = y^*$ makes the expression in the hinge loss equal to 0. This is a convex program so they propose to optimize it using subgradient descent [Ratliff *et al.*, 2007]. A subgradient at $\hat{w}$ is

$$\hat{w} + C(\Phi(x, \hat{y}) - \Phi(x, y^*))$$

where

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} [\langle \hat{w}, \Phi(x, y)\rangle + \Delta(y^*, y)] \tag{6.6}$$

In [Flannick *et al.*, 2009] this is simplified further by ignoring the quadratic term in the objective function, it being claimed that there is a low risk of overfitting with only 36 free parameters, thus letting $C \to \infty$.

We can compute $\hat{y}$ using the subroutine for generalized Viterbi, as $\Delta(y^*, y)$ decomposes into a component for each type of factor. That is, we can associate the loss of each label

with a single factor that contains it. We then add all the label losses that correspond to one type of factor to define the component for that type of factor. Then,

$$\Delta(y^*, y) = \left\langle \overline{1}, \Phi_\Delta \right\rangle$$

where

$$\Phi_\Delta = (\Delta_1(y^*, y), ..., \Delta_N(y^*, y))$$

and $\Delta_i(y^*, y)$ is the component for factor type $i, 1 \leqslant i \leqslant N$. Recall that N is the number of factor types. Thus,

$$\begin{aligned} \hat{y} &= \arg\max_{y \in \mathcal{Y}} [\langle \hat{w}, \Phi(x, y) \rangle + \left\langle \overline{1}, \Phi_\Delta \right\rangle] \\ &= \arg\max_{y \in \mathcal{Y}} [\langle \hat{w}, \Phi_{aug} \rangle] \end{aligned}$$

where

$$\Phi_{aug} = (\frac{\Delta_1(y^*, y)}{\hat{w}_1} + \Phi(x, y)_1, ..., \frac{\Delta_N(y^*, y)}{\hat{w}_N} + \Phi(x, y)_N)$$

which we can solve with our subroutine for generalized Viterbi.

If we try to apply the same conversion to the slack scaling problem we get the subproblem

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} [\Delta(y^*, y)(1 - \langle w, \Phi(x, y^*) \rangle) + \Delta(y^*, y) \langle w, \Phi(x, y) \rangle] \tag{6.7}$$

The term on the right couples components corresponding to different types of factors, preventing us from using our algorithm for generalized Viterbi.

6.5.2 Dual Cutting Plane Algorithm

The dual for the margin scaling problem is

$$\max_{\alpha \geqslant 0} C \sum_y \alpha_y \Delta(y^*, y) - \frac{1}{2} C^2 \sum_{y\overline{y}} \alpha_y \alpha_{\overline{y}} \langle \delta\Phi(y), \delta\Phi(\overline{y}) \rangle$$

$$s.t. \sum_y \alpha_y = 1$$

where $\delta\Phi(y) = \Phi(x, y^*) - \Phi(x, y)$. The dual for the slack scaling problem is

$$\max_{\alpha \geqslant 0} \sum_{y \neq y^*} \alpha_y - \frac{1}{2} \sum_{y \neq y^*, \overline{y} \neq y^*} \alpha_y \alpha_{\overline{y}} \langle \delta\Phi(y), \delta\Phi(\overline{y}) \rangle$$

$$s.t. \sum_{y \neq y^*} \frac{\alpha_y}{\Delta(y^*, y)} \leqslant C$$

In both cases, $w^* = \sum_{y \neq y^*} \alpha_y (\Phi(x, y^*) - \Phi(x, y))$. Tsochantaridis et. al. propose a cutting plane approach [Tsochantaridis *et al.*, 2005]. They propose finding the most vio-

lated constraint using (6.7) or (6.6) and then adding the corresponding dual variable to a working set. After each iteration the dual is optimized over the working set and w is updated. They show that after $\frac{8C\bar{\ell}^3R^2}{\epsilon^2}$ iterations, where $\bar{\ell}$ is the number of labels and $R = \max_y ||\Phi(x,y) - \Phi(x,y^*)||_2$, all constraints in the slack scaling problem will be satisfied to within a precision of ϵ. Each iteration involves solving (6.7) and solving a growing quadratic program. As observed in Section 6.5.1, we cannot solve (6.7) using our algorithm for generalized Viterbi as factors of different types are coupled. In [Tsochantaridis *et al.*, 2005], this problem was not adequately solved for any complex model. In [Tsochantaridis *et al.*, 2004], the same authors give a dynamic programming algorithm to solve (6.7) for a hidden Markov model. We leave for future work investigation of whether this algorithm can be adapted for a reasonable approximate solution of (6.7) in a general factor graph. Tsochantaridis et. al. also show that after $\frac{8C\bar{\ell}R^2}{\epsilon^2}$ iterations, all constraints in the margin scaling problem will be satisfied to within a precision of ϵ. We can solve (6.6) to find the most violated constraint for the margin scaling problem with our algorithm for generalized Viterbi.

6.5.3 Marginal Dual Algorithm

Taskar et. al. observed that, as the only constraint in the margin scaling dual is $\sum_y \alpha_y = 1$, we can view the α_y as a probability distribution [Taskar *et al.*, 2003]. This leads to a practical method given in [Bartlett *et al.*, 2004] for approximately solving the margin scaling optimization. Let F be the set of factors. Let $L(f)$ denote the set of possible labelings of factor f. Let

$$I(y,f,\ell) = \begin{cases} 1 & \text{if } y \text{ labels factor } f \text{ with labeling } \ell \\ 0 & \text{otherwise} \end{cases}$$

If we assume that the probability distribution corresponding to the dual variables α_y factors according to the factor graph, we can write it as

$$\alpha_y(\theta) = \frac{1}{Z}\exp\{\sum_{f\in F}\sum_{\ell\in L(f)} I(y,f,\ell)\theta_{f\ell}\} \tag{6.8}$$

where Z is a normalization constant obtained by summing over y. This gives us a parameterization of the distribution with at most $m2^k$ terms, assuming binary labels, where m is the number of factors and k is the largest number of labels in a factor. Thus, θ should be much easier to store than the $2^{\bar{\ell}}$ components of α. The α_y probabilities are defined implicitly by (6.8).

Let $Q(\alpha)$ be the objective function in the margin scaling dual, a quadratic function in α. The exponentiated gradient algorithm is a general algorithm for maximizing a quadratic function. After initializing α^0 to a point in the interior of the feasible region, the iterations consist of updating α as follows:

$$\alpha_y^{t+1} = \frac{1}{Z}\alpha_y^t\exp\{\eta\frac{\partial Q(\alpha^t)}{\partial\alpha_y}\} \tag{6.9}$$

where Z is a normalization constant ensuring that the α_y^{t+1} continue to form a probability distribution, and $\eta > 0$ is a step size.

It would be very difficult to update each α_y individually so we want to update them implicitly using our representation in (6.8). The objective function $Q(\alpha)$ can be equivalently expressed in terms of the factor marginals of the distribution.

Let $\mu_{f\ell}(\alpha) = \sum_y \alpha_y I(y, f, \ell)$. Suppressing the dependence of μ on α, and using that

$$\Phi(x, y) = \sum_{f \in F} \sum_{\ell \in L(f)} I(y, f, \ell)\phi_{f,\ell}$$

where $\phi_{f\ell} = \frac{1}{M_{T(f)}} F_{T(f)}(S_{f\ell})$, $T(f)$ is the type of factor f, and $S_{f\ell}$ are the labelled edges involved in factor f (see (6.5)), and using that the Hamming distance $\Delta(y^*, y) = \sum_{f \in F_s} \sum_{\ell \in L(f)} I(y, f, \ell)(1 - I(y^*, f, \ell))$ where F_s is the set of single-edge factors which here, for simplicity, we assume are all defined, we have that

$$Q(\alpha) = Q_m(\mu(\alpha)) = C \sum_{f \in F_s} \sum_{\ell \in L(f)} \mu_{f\ell}(1 - I(y^*, f, \ell))$$

$$-\frac{1}{2} C^2 \sum_{\mathcal{I}} [I(y^*, f, \ell) - \mu_{f\ell}][I(y^*, f', \ell') - \mu_{f'\ell'}] \left\langle \phi_{f\ell}, \phi_{f'\ell'} \right\rangle$$

where $\mathcal{I} = \{f, f' \in F, \ell \in L(f), \ell' \in L(f')\}$.

We will perform the exponentiated gradient updates on θ using $\bigtriangledown_\mu Q_m(\mu(\alpha(\theta)))$ and then show that the resulting $\alpha(\theta)$ are implicitly updated according to (6.9). The algorithm consists of choosing initial values for θ^0 and then iteratively calculating all marginals $\mu_{f\ell}^t(\alpha(\theta))$ and setting

$$\theta_{f\ell}^{t+1} = \theta_{f\ell}^t + \eta \frac{\partial Q_m(\mu(\alpha(\theta^t)))}{\partial \mu_{f\ell}} \tag{6.10}$$

where, by direct differentiation,

$$\frac{\partial Q_m(\mu(\alpha(\theta^t)))}{\partial \mu_{f\ell}} = C(1 - I(y^*, f, \ell))$$

$$+\frac{1}{2} C^2 \sum_{f' \in F} \sum_{\ell' \in L(f')} [I(y^*, f, \ell) - \mu_{f',\ell'}] \left\langle \phi_{f\ell}, \phi_{f'\ell'} \right\rangle$$

Theorem: $\alpha(\theta^t) = \alpha^t$, where $\alpha(\theta^t)$ is defined in (6.8) and (6.10), and α^t is defined in (6.9).

Proof:

$$\begin{aligned}
\frac{\partial Q(\alpha)}{\partial \alpha_y} &= \frac{\partial Q_m(\mu(\alpha))}{\partial \alpha_y} \\
&= \sum_{f \in F} \sum_{\ell \in L(f)} \frac{\partial Q_m(\mu(\alpha))}{\partial \mu_{f\ell}} \frac{\partial \mu_{f\ell}}{\partial \alpha_y} \\
&= \sum_{f \in F} \sum_{\ell \in L(f)} \frac{\partial Q_m(\mu(\alpha))}{\partial \mu_{f\ell}} I(y, f, \ell) \\
&\triangleq \sum_{f \in F} \sum_{\ell \in L(f)} \delta_{f\ell} I(y, f, \ell)
\end{aligned}$$

Then, from the definitions of the updates, allowing Z to be a normalization constant that changes from line to line,

$$\begin{aligned}
\alpha_y(\theta^{t+1}) &= \frac{1}{Z} \exp\{\sum_{f \in F} \sum_{\ell \in L(f)} I(y, f, \ell) \theta^{t+1}_{f\ell}\} \\
&= \frac{1}{Z} \exp\{\sum_{f \in F} \sum_{\ell \in L(f)} I(y, f, \ell) (\theta^{t}_{f\ell} + \eta \delta^{t}_{f\ell})\} \\
&= \frac{1}{Z} \alpha_y(\theta^t) \exp\{\eta \sum_{f \in F} \sum_{\ell \in L(f)} I(y, f, \ell) \delta^{t}_{f\ell}\} \\
&= \frac{1}{Z} \alpha_y(\theta^t) \exp\{\eta \frac{\partial Q(\alpha)}{\partial \alpha_y}\} \\
&= \alpha_y^{t+1}
\end{aligned}$$

□

6.5.4 Discussion

From a computational perspective, the margin scaling problem seems preferable to the slack scaling problem. Sarawagi et. al. argue that the slack scaling problem may still deserve continued study as margin scaling may give too much importance to increasing the margin for labelings that are already well-separated from the margin [Sarawagi and Gupta, 2008].

Some variants of the methods described in this paper for solving the margin scaling and slack scaling optimization problems have been proposed in the literature. Sarawagi et. al. propose a variational approximation for the slack scaling optimization problem [Sarawagi and Gupta, 2008]. Using this variational approximation they are able to use the algorithm for generalized Viterbi to approximately solve (3.2) . Using a program implementing their methods, which they call PosLearn, they report results on some examples with test error reduced by 25% over margin scaling and by 10% over exact slack scaling. They argue that their variational approximation provides better loss characterization than exact slack

scaling, explaining the latter surprising result.

Rousu et. al. propose a simple conditional gradient algorithm for optimizing the margin scaling dual [Rousu *et al.*, 2007]. They test on simple examples and barely outperform ordinary SVM that assumes sample independence. They did not compare the conditional gradient algorithm to the exponentiated gradient algorithm and it seems unlikely that it would compare well.

Collins et. al. adapt the exponentiated gradient updates for structured learning to the problem of minimizing the regularized negative log likelihood loss rather than the regularized hinge loss [Collins *et al.*, 2008]. The negative log likelihood loss is the negative logarithm of the conditional distribution on the labelings defined by $p(y|x;w) = \frac{1}{Z}\exp\{w \cdot \Phi(x,y)\}$ where Z is a normalization constant. By using the negative log likelihood loss, we can view parameter learning as a regularized maximum likelihood procedure on the conditional distribution defined above. Using the negative log likelihood loss leads to a different dual to which the same exponentiated gradient updates can be applied, which, similarly, under the same assumptions, reduce to computing factor marginals in the factor graph. Interestingly, Collins et. al. obtain a significantly better theoretical rate of convergence for the negative log likelihood loss than for the hinge loss. They show that, using the hinge loss, at most $\mathcal{O}(\frac{1}{\epsilon})$ exponentiated gradient updates are required to converge to within ϵ of the optimal dual objective function value. Using the negative log likelihood loss, they show that only at most $\mathcal{O}(\log \frac{1}{\epsilon})$ updates are required. They acknowledge the possibility that their bound for the hinge loss may not be tight. Though they compare the exponentiated gradient algorithms empirically to other methods using both hinge loss and negative log likelihood loss, they do not empirically compare the exponentiated gradient algorithms using these two loss functions to each other.

All algorithms for the margin scaling and slack scaling problems require as a subroutine either an algorithm for generalized Viterbi on a factor graph or an algorithm for computing factor marginals of a distribution defined on a factor graph. When the factor graph is such that these problems are NP-hard, as is likely to be the case for the factor graphs in biological network alignment, it becomes necessary to study approximate algorithms. Theoretical results, such as the polynomial bound on the number of iterations required for satisfaction of all constraints to within an ϵ precision, rely on exact solutions to the inference problem. Kulesza et. al. and Finley et. al. have studied various approximate methods for inference and show that, in some cases, approximate inference methods do not combine well with the proposed approximate methods for solving the structured learning optimization problems [Kulesza and Pereira, 2007; Finley and Joachims, 2008]. Both studies conclude that the best results are likely to be obtained from inference algorithms based on the LP-relaxation studied by Wainwright et. al. [Wainwright and Jordan, 2008].

6.6 Whole Genome Probabilistic Model of Evolution

Orthologous proteins are proteins in two separate species that derive from the same ancestral protein in the last common ancestor of those two species [Alberts *et al.*, 2008]. Detecting

orthologous proteins is important not only for its intrinsic interest in reconstructing the evolutionary history of proteins, but also because orthologous proteins often maintain similar function [Koonin, 2005]. Experiments on a protein in a model organism often provide information about the function of its orthologous proteins in other species. Proteins are composed of protein domains that are used in constructing proteins of many different functions [Alberts *et al.*, 2008]. A protein with multiple domains may be orthologous to many seemingly unrelated proteins. The best conserved domain across two proteins may not be the domain that determines the primary function of the proteins.

The most popular protein orthology detection methods, as reviewed in [Kuzniar *et al.*, 2008] and [Alexeyenko *et al.*, 2006], are based entirely on neutral models of DNA sequence evolution of genes [Kimura, 1983], though some also consider the physical arrangement of genes on the chromosomes. Many are based on local sequence alignment algorithms that focus only on the best conserved domain, ignoring the rest of the proteins which may include the more informative domains. Others are based on multiple sequence alignment that attempts to align full proteins which is not very good at detecting orthology between multiple-domain proteins that do not have identical domains, a common situation in alignments between vertebrates and invertebrates where protein interaction domains often tag on to preexisting proteins. These methods construct multiple alignments of protein families that are themselves determined using Hidden Markov models [Zmasek and Eddy, 2002; Storm and Sonnhammer, 2002]. cite Song 08 and protein interaction domain papers.

Research in the alignment of protein interaction networks has attempted to model changes in the genome, such as protein loss, protein duplication, and changes in protein interactions [Sharan and Ideker, 2006; Flannick *et al.*, 2009; Koyutürk *et al.*, 2006b]. However, a probabilistic model of genome evolution suitable for protein orthology detection has not yet been defined.

Except for preliminary studies such as [Bandyopadhyay *et al.*, 2006] and [Yosef *et al.*, 2008], and the use of conserved synteny in some methods, the methods currently used to detect protein orthology can be divided into two classes. One class, the phylogeny reconstruction methods, use a multiple alignment of the protein sequences to reconstruct a phylogeny of the proteins from which orthology can be inferred [Storm and Sonnhammer, 2002; Zmasek and Eddy, 2002]. It is difficult to reconstruct an accurate multiple alignment when proteins have diverged significantly in sequence making this step of multiple sequence alignment error-prone [Levasseur *et al.*, 2008]. The other class of methods, the pairwise similarity methods, are based on pairwise measures of sequence similarity, such as Smith-Waterman [Smith and Waterman, 1981] or BLAST [Altschul *et al.*, 1997]. Most pairwise similarity methods are variants of a reciprocal best hits algorithm in which the proteins are predicted to be orthologous if each is the highest scoring match in the genome of the other species [Tatusov *et al.*, 1997; Remm *et al.*, 2001; Kuzniar *et al.*, 2008]. Recent studies have disagreed as to whether phylogeny reconstruction or pairwise similarity methods are more reliable [Hulsen *et al.*, 2006; Chen *et al.*, 2007; Altenhoff and Dessimoz, 2009]. Both classes of methods have difficulty working with protein deletions [Kuzniar *et al.*, 2008]. Neither class of methods considers evidence from protein

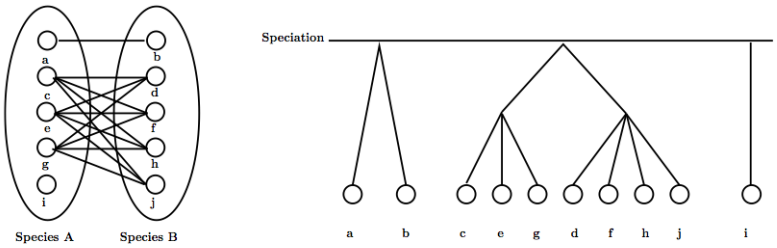


Figure 6.4: An alignment can be represented either as a bipartite graph or as a partial phylogeny.

interaction patterns. Because pairwise similarity methods are computationally more efficient than phylogeny reconstruction methods, databases of putative protein orthology such as HCOP [Eyre *et al.*, 2006], YOGY [Penkett *et al.*, 2006], KEGG [Kanehisa *et al.*, 2006; Aoki and Kanehisa, 2005], and P-POD [Heinicke *et al.*, 2007] are populated primarily with predictions from pairwise similarity methods.

Protein interaction network alignment began with a search for conserved signalling pathways and protein complexes across species [Kelley *et al.*, 2003; Sharan *et al.*, 2004; Koyutürk *et al.*, 2006b; Flannick *et al.*, 2006]. This approach is called local protein interaction network alignment because it does not attempt to align all proteins, rather searching for local highly conserved regions in the network. Global protein interaction network alignment aligns all proteins of interest, aligning proteins based on the alignment of neighboring proteins in the networks [Singh *et al.*, 2007; Flannick *et al.*, 2009]. For orthology detection, if one is interested in finding a few nonspecific highly-supported orthologous proteins between two species, local alignment would be appropriate, but if one is searching for proteins orthologous to particular proteins or conducting a comprehensive study of orthology between two species, global alignment is the better approach.

A basic premise is that there is some sequence conservation between orthologous proteins, but the amount may vary depending on the proteins. Moreover when one is detecting protein orthology, one is primarily interested in functional orthology, those orthologous proteins that have conserved function. If a pair of orthologous proteins is orthologous because of sharing a domain that does not primarily determine the functions of the proteins, those proteins are orthologous but not functionally orthologous.

In this paper we propose a method, Orthalign, for detecting orthology between two animal species by probabilistically modeling whole genome evolution. We test the method using species at varied evolutionary distances: among mammals and between vertebrates and invertebrates. Our method resolves some orthologous proteins correctly that are not resolved correctly by other methods with which we compare.

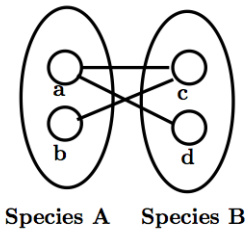


Figure 6.5: Not every bipartite graph is an alignment.

6.6.1 Probabilistic Model

We define an alignment as a partial phylogeny on the proteins in which all coalescence prior to the speciation is ignored, all branching subsequent to the speciation represents protein duplication, and the branching pattern within each lineage is collapsed to a single multifurcation. That is, there is a one-to-many relationship between alignments and phylogenies. Given any phylogeny on the proteins, a unique alignment is implied. Figure 6.4 illustrates an alignment using two representations: a bipartite graph and a partial phylogeny. Figure 6.5 shows that not every bipartite graph is an alignment. All four proteins are connected, implying, by transitivity, that they descended from a single protein in the LCA, but the absence of edge (b,d) implies that b and d descended from distinct proteins in the LCA, a contradiction. An orthologous group (OG) is a maximal complete bipartite subgraph. An alignment is a bipartite graph that can be partitioned into OGs. For example, the alignment in Figure 6.4 can be partitioned into 3 OGs.

Given the genome of the LCA, the random effects of evolution imply a probability distribution on the genomes of the present-day species. As modeled by standard models for sequence evolution, there is a probability distribution on the divergence of two orthologous proteins over the time since the LCA, but there is also a probability in one or both lineages of each protein being lost or duplicated, genes being physically rearranged on the chromosomes, protein interaction patterns being conserved or altered, and similar but nonorthologous proteins forming identical interaction patterns. Many protein interaction changes may be caused by small modifications of coding regions that have a major functional effect [Alberts *et al.*, 2008]. Alternately, many changes in coding regions may have only a small effect on basic function [Fay and Wu, 2003]. Thus, protein interaction information supplements sequence information in the prediction of protein orthology, and while amount of sequence change is correlated with change of function, neither determines the other with certainty (cite Espadaler).

For $i \in \{A, B, LCA\}$, let $\mathcal{G}_i = (V_i, M_i, C_i)$ where V_i is the set of proteins, M_i is the set of modules weighted to reflect any uncertainty in our knowledge, and C_i is the set of chromosomes on which the protein-coding genes are located. Given the genome of the LCA, $\mathcal{G}_{LCA}$, we can model the probability of evolving the genomes $\mathcal{G}_A$ and $\mathcal{G}_B$ as follows. Summing over all alignments, $\mathcal{A}$, that, at this stage, include the ancestral proteins:

$$\begin{aligned}\Pr(\mathcal{G}_A, \mathcal{G}_B \mid \mathcal{G}_{LCA}) &= \sum_{\mathcal{A}} \Pr(\mathcal{G}_A, \mathcal{G}_B, \mathcal{A} \mid \mathcal{G}_{LCA}) \\ &= \sum_{\mathcal{A}} \Pr(\mathcal{G}_A, \mathcal{G}_B \mid \mathcal{A}, \mathcal{G}_{LCA}) \Pr(\mathcal{A} \mid \mathcal{G}_{LCA}).\end{aligned}$$

Since

$$\Pr(\mathcal{A} \mid \mathcal{G}_A, \mathcal{G}_B, \mathcal{G}_{LCA}) = \frac{\Pr(\mathcal{G}_A, \mathcal{G}_B \mid \mathcal{A}, \mathcal{G}_{LCA}) \Pr(\mathcal{A} \mid \mathcal{G}_{LCA})}{\Pr(\mathcal{G}_A, \mathcal{G}_B \mid \mathcal{G}_{LCA})},$$

by Bayes' Rule, each term in the sum is proportional to the posterior probability of the corresponding alignment given the ancestral and present-day genomes. Thus, an alignment that maximizes

$$\Pr(\mathcal{G}_A, \mathcal{G}_B \mid \mathcal{A}, \mathcal{G}_{LCA}) \Pr(\mathcal{A} \mid \mathcal{G}_{LCA})$$

is a maximum a posteriori (MAP) estimate.

We have three types of data representing $\mathcal{G}_A$ and $\mathcal{G}_B$: 1) protein sequence data representing mutations, 2) locations of genes on chromosomes representing synteny, and 3) protein interaction data representing functional modules. The events representing mutations and protein interactions are not independent as high numbers of mutations in a protein are likely to change its interactions. However, it is unclear how the events causing change in synteny, that is, chromosomal inversions and translocations, are correlated, if at all, with mutations and protein interactions, so for these we assume independence. Using these dependency assumptions, we have

$$\begin{aligned}\Pr(\mathcal{G}_A, \mathcal{G}_B \mid \mathcal{A}, \mathcal{G}_{LCA}) &= \Pr(V_A, V_B \mid \mathcal{A}, \mathcal{G}_{LCA}) \\ &\quad \Pr(C_A, C_B \mid \mathcal{A}, \mathcal{G}_{LCA}) \\ &\quad \Pr(M_A, M_B \mid V_A, V_B, \mathcal{A}, \mathcal{G}_{LCA})\end{aligned}$$

Thus,

$$\begin{aligned}\Pr(\mathcal{A} \mid \mathcal{G}_A, \mathcal{G}_B, \mathcal{G}_{LCA}) &\propto \Pr(V_A, V_B \mid \mathcal{A}, \mathcal{G}_{LCA}) \\ &\quad \Pr(C_A, C_B \mid \mathcal{A}, \mathcal{G}_{LCA}) \\ &\quad \Pr(M_A, M_B \mid V_A, V_B, \mathcal{A}, \mathcal{G}_{LCA}) \\ &\quad \Pr(\mathcal{A} \mid \mathcal{G}_{LCA})\end{aligned}$$

and to find the MAP estimate we want to maximize the expression on the right-hand side over alignments.

Our goal is to remove the dependence on $\mathcal{G}_{LCA}$, for which we do not have any data, by the use of reasonable approximations. We desire an approximate distribution over alignments

that, at the end, will not include ancestral proteins, that is, an approximate distribution of the form

$$\Pr(\mathcal{A} \mid \mathcal{G}_A, \mathcal{G}_B) \quad \propto \quad \Pr(V_A, V_B \mid \mathcal{A}) \Pr(C_A, C_B \mid \mathcal{A}) \Pr(M_A, M_B \mid V_A, V_B, \mathcal{A}) \Pr(\mathcal{A})$$

There is a natural prior $\Pr(\mathcal{A} \mid \mathcal{G}_{LCA})$ on alignments. An alignment that requires a large number of duplications of a single protein between closely related species should have low probability whereas alignments that follow reasonable patterns of protein duplication and deletion should have high probability. There are, thus, four factors in the distribution which, in order, represent 1) mutations in coding regions, 2) conservation and changes in synteny, 3) protein interaction conservation, loss, and gain, and 4) protein deletion and duplication. The models and approximations for the factors are described in the four subsections that follow.

6.6.1.1 Protein Deletion and Duplication

First we discuss the prior, that is, the model of protein deletion and duplication. A natural prior is

$$\Pr(\mathcal{A} \mid \mathcal{G}_{LCA}) = \prod_{p \in V_{LCA}} \Pr(d_A(p)) \Pr(d_B(p))$$

where $d_i(p) \triangleq$ number of proteins descended from p in V_i.

Eliminating the dependence on $\mathcal{G}_{LCA}$ requires some approximation. First, an alignment that does not include ancestral proteins does not give any information about duplications in one lineage when the orthologous protein in the other lineage was lost. Second, we must be careful not to count the same duplication multiple times. Figure 6.4 shows an alignment as a partial phylogeny. We can see from the figure that we should count a total of 5 duplications. Figure 6.4 also shows the same alignment as a bipartite graph. Using pairwise comparisons from each node naively, we would count 17 duplications. To count only 5 duplications, as we should, we must use a more clever algorithm, finding each connected component and then counting the number of proteins on each side of the bipartition. Alternately, we can scale the probabilities by the appropriate amount. For example, in Figure 6.4, we can use

$$\Pr(\text{nodeldup})^2 \ (\sqrt[3]{\Pr(3 \text{ dup})})^3 \ (\sqrt[4]{\Pr(2 \text{ dup})})^4 \ \Pr(\text{del}).$$

If we know how many proteins are in-paralogous to a protein in an alignment, we can use this to compute the root value. We can still compute factors locally if we store at each node the number of proteins in-paralogous to each protein relative to the current alignment, and when changing the alignment, automatically update these.

These approximations give the approximate prior distribution

$$\Pr(\mathcal{A}) \approx \prod_{a \in V_A} \sqrt[|IP(a)+1|]{\Pr(|Or(a)|} \prod_{b \in V_B} \sqrt[|IP(b)+1|]{\Pr(|Or(b)|}$$

where $IP(a)$ is the set of proteins in-paralagous to proteins in a and $Or(a)$ is the set of proteins orthologous to proteins in a. We use a distribution with 6 parameters: Pr(0), Pr(1), Pr(2), Pr(3), Pr(4), Pr(5+).

6.6.1.2 Mutations in Coding Regions

Let $Anc(a) \triangleq$ the protein in V_{LCA} ancestral to a. Then,

$$\Pr(V_A, V_B \mid \mathcal{A}, \mathcal{G}_{LCA}) = \prod_{a \in V_A} \Pr(s(a, Anc(a))) \prod_{b \in V_B} \Pr(s(a, Anc(b)))$$

where $s(a, Anc(a))$ is the event of sequence a diverging from the sequence of $Anc(a)$ in the amount that it has.

Not having access to ancestral proteins, a reasonable approximation is to assume the absence of homoplasy and to compare proteins in V_A directly to proteins in V_B. Using this approximation we must omit the probabilities of sequence divergence for proteins that have been deleted in one of the lineages. We must be careful to avoid multiple counting of divergence for duplicated proteins. The divergence of each protein from its putative ancestral protein should be considered only once. A reasonable heuristic is, for each protein, to consider half the average sequence divergence with its orthologous proteins. Using this heuristic for an approximation, we get

$$\begin{aligned}\Pr(V_A, V_B \mid \mathcal{A}) \approx & \prod_{a \in V_A} \Pr\left(\frac{1}{2|Or(a)|} \sum_{b \in Or(a)} s(a,b)\right) \\ & \prod_{b \in V_B} \Pr\left(\frac{1}{2|Or(b)|} \sum_{a \in Or(b)} s(b,a)\right)\end{aligned}$$

where $Or(a)$ is the set of proteins orthologous to proteins in a.

To calculate the probabilities on the right-hand side, we translate sequence-similarity scores to approximate mutation rate per amino acid necessary if the proteins were orthologous. We then use a normal distribution on observing that mutation rate. We learn the mean and variance of the normal distribution on mutation rate from data.

Not all proteins evolve at the same rate even for a constant mutation rate because proteins are subject to varying amounts of purifying selection. Above we are not actually using mutation rate but rather average rate of substitution per nucleotide, averaging over all possible amounts of purifying selection.

6.6.1.3 Conservation and Changes in Synteny

Let $r_i(a)$ be the protein to the right of a on C_i. Let $scons(a, r_i(a))$ be the event of a and $r_i(a)$ having conserved synteny from $\mathcal{G}_{LCA}$ according to $\mathcal{A}$. Then,

$$\Pr(C_A, C_B \mid \mathcal{A}, \mathcal{G}_{LCA}) = \prod_{a \in V_A} \Pr(scons(a, r_A(a))) \prod_{b \in V_B} \Pr(scons(b, r_B(b))).$$

Here, because of inversions, synteny can be in either direction. Two genes have conserved synteny in two genomes if they are adjacent in both genomes regardless of whether they appear in the same order.

It is desirable to eliminate dependence on C_{LCA}. Thus, we assume each pair of physically adjacent genes in C_{LCA} remain physically adjacent in at least one of the present-day genomes. This gives the approximate distribution

$$\Pr(C_A, C_B \mid \mathcal{A}) \approx \prod_{a \in V_A} \Pr(scons'(a, r_A(a))) \prod_{b \in V_B} \Pr(scons'(b, r_B(b)))$$

where $scons'_i(a)$ is the event of proteins $a, r_i(a) \in V_i$ being mapped by $\mathcal{A}$ to adjacent proteins in $C_j, i \neq j$. The proteins a and $r_i(a)$ may be mapped to multiple proteins in C_j, some of which, but not all, may be adjacent. In this case, if any mapped proteins are adjacent, we consider this to be overwhelming evidence of conserved synteny, and the mapped proteins that are not adjacent are inferred to be the result of duplications.

There are just two parameters to learn here: the probability of two adjacent proteins in Species A not being aligned to adjacent proteins in Species B, and the probability of two adjacent proteins in Species B not being aligned to adjacent proteins in Species A.

Conserved synteny only applies to closely related species, such as human and mouse. It is estimated that only 180 chromosome rearrangements took place between human and mouse [Alberts *et al.*, 2008].

6.6.1.4 Protein Interaction Conservation, Gain, and Loss

In line with the neutral theory of molecular evolution, we consider de novo module gain as a rare event.

Let $\texttt{proteins}_i(m)$ be the set of proteins in V_i to which $\mathcal{A}$ maps m. Let $M_i(m)$ be the restriction of M_i to modules of the same type as m consisting of proteins descended from the proteins in m according to $\mathcal{A}$. Let N be the set of modules not in any $M_i(m)$ for any $i \in \{A, B\}$ for any m. Then,

$$\begin{aligned}\Pr(M_A, M_B \mid V_A, V_B, \mathcal{A}, \mathcal{G}_{LCA}) = & \prod_{m \in M_{LCA}} \Pr(M_A(m) \mid \texttt{proteins}_A(m), V_A, V_{LCA}) \\ & \Pr(M_B(m) \mid \texttt{proteins}_B(m), V_B, V_{LCA}) \\ & \prod_{n \in N} \Pr(n)\end{aligned}$$

It is desirable to eliminate the dependence on M_{LCA} and V_{LCA}. Thus, we make the simplifying assumptions that each module remains conserved in at least one of the present-day genomes and that there are no de novo modules, and then we compare modules in the present-day genomes with each other.

Let $OG(m)$ be the set of OGs in $\mathcal{A}$ associated with m. Let $\texttt{modules}(\mathrm{m})$ be the restriction of M_A and M_B to modules of the same type as m consisting of proteins in the same ordered OGs as m. Let $\mathcal{M}$ be the set of module sets: $\{\texttt{modules}(m) : m \in M_A \cup M_B\}$. Note that all

modules appear in $\mathcal{M}$. Then,

$$\Pr(M_A, M_B \mid V_A, V_B, \mathcal{A}) \approx \prod_{\texttt{modules}(m) \in \mathcal{M}} \Pr(\texttt{modules}(m) \mid OG(m), V_A, V_B)$$

Determining these probabilities would involve considering each possible module and, for each, considering how many modules in the corresponding OGs are conserved. For modules consisting of k proteins, there could be $\Omega(n^k)$ sets of OGs to evaluate where n is the number of proteins in the larger species. Even evaluating a single OG could be difficult for an alignment with large OGs.

A more tractable approximation is to model the probability of protein q taking part in x conserved motifs of type i, where the conservation is according to $\mathcal{A}$ on edges each having sequence similarity above threshold t. This probability can be written $p_j(x_q, i, t)$. Then,

$$\Pr(M_A, M_B \mid V_A, V_B, \mathcal{A}) \approx \prod_{q \in V_A} \prod_{i=1}^{m} p_A(x_q, i, t) \prod_{q \in V_B} \prod_{i=1}^{m} p_B(x_q, i, t)$$

where x_q is the number of conserved motifs with protein q of type i in M_A and M_B where conservation is with respect to threshold t using the protein sequences in V_A and V_B. Each $p_j(x_q, i, t)$ can be modeled by a Poisson distribution with mean learned from data.

One nice feature of the above approximation is that it allows us to model the decreased purifying selection on the copies of modules that include duplicated proteins. If a protein is duplicated, each protein not duplicated begins to take part in twice as many modules of the same type. Thus, the probability distributions $p_j(x_q, i, t)$ model the decrease in purifying selection by lowering the probability of a single protein taking part in too many modules of the same type.

Given a fixed threshold t, there are $2 * i$ parameters to learn, the means of the $2 * i$ Poisson distributions.

6.6.2 Computing a MAP Alignment

To compute the MAP estimate we want to maximize $\Pr(\mathcal{A} \mid \mathcal{G}_A, \mathcal{G}_B)$ over $\mathcal{A}$. The alignment $\mathcal{A}$ is completely specified by the OGs. Thus, we implement a hill-climbing algorithm to find the mode of the distribution. We start by assigning each protein to its own OG. Thereafter, we implement the following moves:

1. Merge two OGs
2. Place a protein in its own OG

We iterate until we reach a local maximum.

6.6.3 Parameter Learning

We want to choose the parameters such that the correct alignment on the training set is the MAP estimate and such that our algorithm can find this estimate. There are $10 + 2i$

parameters to learn where i is the number of motifs.

We could try to learn the correct parameters by examining the training set, but, given that the algorithm to find the MAP estimate is only approximate, it may be a better approach to set the parameters iteratively while ensuring that the MAP algorithm can always find the correct alignment on the training set. This ensures that the parameters work together to guide the MAP algorithm to the correct alignment.

If the parameters are poorly set, the MAP algorithm will find an alignment with higher probability than the desired alignment on the training set. There will be some gap between the probabilities of the alignment found and the correct alignment. Our goal is to minimize this gap, ideally to 0. If the correct alignment has a higher probability than the alignment found by the MAP algorithm, this means that the MAP algorithm is unable to find the correct alignment, and so, in this case too, we should adjust the parameters to guide the MAP algorithm to the correct alignment. Thus, we make random changes to the parameters and accept each change with a probability that grows as the gap shrinks.

6.6.4 Discussion

6.6.4.1 Need for Gold Standard

Orthalign requires a sufficiently-large, reliable, and representative gold standard collection of orthologous proteins for a pair of species in order to estimate the parameters of the probabilistic model. With a reasonable model in hand, the detection of additional orthologous proteins between these species can be improved and simplified. Moreover, putative orthologous proteins in the gold standard collection that do not conform well with the model and, thus, may have been incorrectly predicted, can be identified.

6.6.4.2 Graemlin 2.0

The burgeoning research in protein interaction network alignment has used information beyond protein interactions alone to align protein networks across species. One recent study, Graemlin 2.0, uses both phylogeny reconstruction and pairwise BLAST scores to augment information from protein interactions [Flannick *et al.*, 2009]. In principle, the extra evidence of protein orthology given by conserved protein interaction patterns should enable network alignment algorithms to detect orthologous proteins more reliably than extant protein orthology detection methods, but few comparisons have been made [Bandyopadhyay *et al.*, 2006].

Graemlin 2.0 was the first method that attempted to learn species-dependent parameters for protein interaction network alignment. They attempted to learn parameters for a scoring function based on sequence similarity as measured by BLAST, protein deletion and duplication, and single protein interaction gain or loss. They did not attempt to cast their scoring function as a probabilistic model of evolution and while interpreting certain features as probabilities, they add rather than multiply them. Graemlin 2.0 learns parameters from KEGG, a database populated by a pairwise sequence similarity method. Thus, the method and parameters may be optimized to imitate pairwise sequence similarity methods and may

not be fully utilizing the protein interaction data. The large equivalence classes of Graemlin 2.0 cause a loss of protein orthology information since they are paralogous families of proteins that may include duplication events above internal nodes in the phylogenetic tree.

6.6.4.3 CAPPI

One protein interaction alignment method, CAPPI, is based on a probabilistic model of genome evolution, but has severe limitations for the detection of orthologous proteins [Dutkowski and Tiuryn, 2007]. Using BLAST scores, CAPPI divides proteins into clusters which subsequently cannot be modified. It then reconstructs the phylogeny for each cluster with neighbor-joining. Using a Bayesian network based on a model of evolution with parameters fine-tuned by hand, it computes posterior protein interaction probabilities on the supposedly ancestral proteins corresponding to the clusters. Highly probable interactions are then projected down to all proteins in the clusters of present-day species. For protein orthology detection, this method can, at best, be considered as a filter to remove some protein orthology predictions made by pairwise similarity methods, but it is limited in that it must take or reject each cluster as a whole. CAPPI does not claim nor attempt to be a protein orthology detection method; rather it attempts to identify a few regions in each protein interaction network that share conserved interactions. It seems difficult or impossible to extend their framework to a useful protein orthology detection method as the clusters corresponding to proteins descending from a putative single ancestral protein are generated without using protein interaction data and cannot be changed during the course of the algorithm.

6.6.4.4 Method of Berg and Lässig

Another method based on a probabilistic model of genome evolution models protein interaction change with a diffusion process [Berg and Lässig, 2006]. it combines this model of interaction change with a model of sequence change and estimates parameters using a Bayesian analysis. Instead of using the prior on alignments to model deletions and duplications, as described in the sequel, they use a flat prior and do not attempt to model deletions or duplications, though the same authors earlier developed a probabilistic model for duplications [Berg *et al.*, 2004]. They model each interaction identically and do not assume stronger purifying selection on intramodular interactions so they are unable accurately to model modular evolution such as that described in [Hartwell *et al.*, 1999].

6.6.4.5 Genome Wide Applications

For the method to be useful at the genome level, it must be capable of aligning full genomes. Currently this would require significant running time. We note, however, that genome-wide detection of orthologous proteins, even with simple methods, is an expensive and time-consuming process. One genome-wide protein orthology detection method, OMA [Dessimoz *et al.*, 2005], required more than 500,000 CPU hours. When we consider that for the small examples in this paper we use less than 24 CPU hours, it remains likely that Orthalign will

compare favorably with extant methods even at a genome-wide scale. OMA is a symmetric best-hits algorithm based on pairwise Smith-Waterman sequence similarity scores with a consistency check. Computing the Smith-Waterman scores is the most time consuming step. It may be possible to use pre-computed Smith-Waterman scores from a comprehensive effort such as OMA to improve protein orthology detection in a second phase using Orthalign. One difficulty with this latter approach is that different researchers use different sets of proteins, though recently some standardization has been under way [Reference Genome Group of the Gene Ontology Consortium, 2009]. As computing power continues to increase, more reliable methods such as Orthalign may soon become feasible at a genome-wide scale.

It is not necessary to align full genomes with Orthalign, but it is important to use identical complete homologous families in the two species in order to ensure that the interactions that appear to have changed are not missing simply because of incomplete protein families.

6.6.4.6 Nonorthologous Gene Displacement

In some cases nonorthologous gene displacement may cause nonorthologous proteins to assume similar functional roles [Koonin *et al.*, 1996]. However, it is also possible that, due to the different environments encountered by the species, positive selection changes orthologous proteins considerably while they retain their high-level function and interaction patterns within the protein complex or pathway. When interaction patterns disagree with sequence homology, it may be difficult or impossible to distinguish between these two cases. In this work we favor the latter interpretation. A previous study favored the former interpretation, but the same authors, in subsequent work, comparing viral genomes, favored the latter interpretation [Berg and Lässig, 2006; Kolář *et al.*, 2008]. The truth is likely to include a combination of these two scenarios and their relative effects are likely to remain a source of research and debate for some time to come. Resolving this matter is not essential for practical purposes as, whether orthologous proteins have diverged in sequence but retained function or whether nonorthologous proteins have converged to the same function, in either case transfer of functional annotation is more reliable using the extra information of protein interaction patterns than transfer of functional annotation based only on sequence. In addition to its intrinsic interest in elucidating the evolutionary history of protein families, Orthalign can be used to determine which protein in one species is most likely to recover the function of a homologous protein in another species.

6.6.4.7 Orthologous and Paralogous Proteins

The GO consortium currently uses larger paralogous families as determined by PANTHER [Thomas *et al.*, 2003] for their transfer of functional annotation [Reference Genome Group of the Gene Ontology Consortium, 2009]. For functional annotation transfer at a more specific level of detail, it may be useful to transfer annotation only between orthologous proteins.

Traditional phylogenomics approaches based entirely on sequence can have difficulty disambiguating between orthologous proteins and paralogous proteins. Sometimes this is

because of ancient divergence, huge populations, and fast generation time as in the case of some viruses [Kolář *et al.*, 2008]. Sometimes it is because of protein deletions [Kuzniar *et al.*, 2008]. Sometimes it is because of approximations in the multiple alignment algorithms. This study is meant to be an investigation into whether probabilistic modeling of genomic changes other than those considered by multiple alignment programs and traditional phylogenomics can assist in disambiguation of protein orthology detection in these difficult cases and improve the current methods used for functional annotation transfer between orthologous proteins and homologous proteins in general. Our results show that in some cases it may be beneficial to model changes in protein interaction patterns.

6.6.4.8 Neutralism and Selectionism

The neutral theory [Kimura, 1983] makes good sense for DNA sequence evolution. It may be somewhat less applicable for protein interaction evolution and evolution by protein deletions and duplications, as these changes are less likely to be neutral. Here we do not enter into the neutralist-selectionist debate. The probabilistic model makes sense under both interpretations. Even if selection is a dominant force, by choosing a training set similar to the set for which we desire a protein orthology mapping, we can learn frequencies of evolutionary events and attempt to match these in the alignment that we output. Provided our training set is under a selective force of an amount that is similar, on average, to the set for which we desire a protein orthology mapping, it makes no difference to our model whether selection or genetic drift is the driving force in the evolution of protein interactions and protein deletions and duplications.

6.6.4.9 Philosophical Justification for Approximations on a Complex Model

To effectively model evolution, both reasonable models and efficient heuristics for working with the models are needed. A similar conclusion was recently reached for the problem of modeling sequence evolution [Liu *et al.*, 2009] . Performing exact inference on reasonable models is often intractable. Developing approximate models and using algorithms for approximate inference with good heuristics are complementary approaches to make inference tractable.

6.6.4.10 Protein Domains

One topic for future research is to consider protein domains. Many domains evolve independently so it may make more sense to speak of protein domain orthology than protein orthology [Babushok *et al.*, 2007]. If the most highly conserved domain between two proteins is not the domain that most accurately determines function, local sequence similarity methods such as Smith-Waterman and BLAST will not be accurate methods for transferring functional annotation. Taking into account functional interaction patterns should significantly improve functional annotation transfer. Still, it may be useful to know domain orthology. Some databases of orthologous proteins, such as HOPS, are databases of domain-based orthology [Storm and Sonnhammer, 2003] using Pfam domains [Sammut *et*

al., 2008]. The method of Orthalign can be applied to the detection of protein domain orthology by building on recent methods that decompose protein interactions into protein domain interactions [Schelhorn *et al.*, 2008]. We leave this for future work.

6.6.4.11 Protein Duplications

In eukaryotes each gene every million years has one percent probability of duplication [Alberts *et al.*, 2008, page 255]. Mouse and humans diverged 80 million years ago [Alberts *et al.*, 2008, page 249]. Thus as low as $(.99)^80 = 45\%$ of genes may have a one-to-one correspondence even between human and mouse.

6.6.4.12 Limitations

One limitation of the proposed model is that it does not model domain shuffling or reassortment of gene segments. Homology is detected using a local sequence alignment tool which detects homology only between the best matching domains. It may be that the best matching domain is not the domain that determines the primary function of the protein. The Orthalign model for evolution of functional modules should help to mitigate this limitation and is, indeed, one of the primary motivations for the development of Orthalign.

Chapter 7

Outlook

7.1 As New Data Arrives

As cell systems continue to be better understood with increased development of systems biology [Vidal, 2009], this line of research has a bright future beyond the current concept of interactome. Interaction assays are being designed to overcome limitations of the common yeast two-hybrid (Y2H) [Fields, 2009] and affinity-purification mass spectrometry (AP-MS) assays [Lalonde *et al.*, 2008]; these include surface plasmon resonance (SPR) [Lalonde *et al.*, 2008; Madeira *et al.*, 2009], optical microscopic techniques such as FRET [Lalonde *et al.*, 2008; Masi *et al.*, 2010; Alberts *et al.*, 2008], and mating-based split ubiquitin systems (mbSUS) [Lalonde *et al.*, 2008]. As techniques are improved, each interactome is likely to be separated into various sub-interactomes distinguished by cell type and interaction type. A natural next step from the work presented here is to study the relationships between algorithms being used to detect conserved modules, experimental assays being used to construct interactomes, and organization of the cell. This should prove to be fertile ground for designing algorithms tailored to specific experimental assays, with the aim to overcome limitations of each assay and to uncover more details of cellular organization. It was shown the the properties of tissue-specific protein interaction networks differ from those of conglomerate networks [Zhang and Lu, 2010].

Can conserved multiprotein modules be found in gene co-expression networks? A gene co-expression network is a weighted network with genes as vertices and edge weights measuring correlation of expression levels [Stuart *et al.*, 2003; Werner, 2004]. Similar techniques to those used for unweighted networks may be successful, as the original definition of graph conductance applies to weighted networks [Jerrum and Sinclair, 1988]. For future work, we would like to consider using gene expression data [Wang *et al.*, 2009] or combining types of data [Narayanan *et al.*, 2010].

As perhaps 8% of the coding capacity of a mammalian genome is devoted to the synthesis of proteins that serve as regulators of gene transcription [Alberts *et al.*, 2008, page 450], a possibility is to search for conserved multiprotein modules in regulomes: directed graphs with multiple kinds of edges specifying regulatory relationships among genes and proteins.

Early studies analysed modular structure in regulatory networks using small numbers of genes compiled from experimental studies of regulatory relationships [Thieffry and Sánchez, 2004].

Some proteins interact by operating on a common substrate, possibly a molecule other than a protein. These are called functional interactions [Werner, 2004, Figure 5]. Can modules be defined and discovered on functional interactions?

Genetic interactions for yeast have recently been detected using a high-throughput assay [Costanzo *et al.*, 2010]. Are similar networks available for any other species for a comparative analysis? Can the use of modularity assist in the interpretation of these genetic interactomes?

Exciting research on differences in genetic evolutionary modularity between sexual and asexual species has recently been conducted [Livnat *et al.*, 2010]. The finding is that mixable alleles, those that can be matched with a variety of alleles at other genetic loci, have a selective advantage in sexual populations. Each protein interaction in the cell required for organismal fitness imposes an epistatic constraint between the two interactants. Epistasis and co-evolution are related [Schlosser and Wagner, 2008]. Multiprotein modularity implies greater mixability as it implies primarily local interactions, so that proteins in a module are free to evolve independently from proteins in other modules. It may be illuminative to investigate whether there is a greater degree of multiprotein modularity in the interactomes of sexual species relative to those of asexual species. A challenge is to find asexual species that have maintained asexuality for an evolutionarily significant period of time.

Another challenge is to design a reasonable notion of modularity for metabolic networks [Lacroix *et al.*, 2008]. Modularity based on graph conductance is especially suited as it penalizes high-degree nodes: such as the troublesome compounds such as water, etc... that have enormous degree in the metabolic networks. Most people filter these ad hoc before applying algorithms, but a reasonable definition of modularity should allow them to remain in the networks to be processed naturally. The networks could be either directed hypergraphs or directed bipartite graphs where the nodes are compounds and reactions. Enzymes could also play a role [Cottret and Jourdan, 2010]. Building metabolic networks for sequenced genomes and analyzing them comparatively has already led to interesting results [Cottret *et al.*, 2010].

7.2 As New Algorithms Arrive

A new algorithm, EvoNibble [Andersen and Peres, 2009], has been designed for finding sets of vertices with low conductance in a graph, which may be adapted to find small modules in a large interactome. EvoNibble is a randomized algorithm based on a volume-biased evolving set process. Though a randomized algorithm complicates replication of results, it can be run multiple times to create a high-confidence set that may also include results from other module-detection algorithms.

Bibliography

[Adams *et al.*, 2000] Mark D. Adams, Susan E. Celniker, Robert A. Holt, Cheryl A. Evans, Jeannine D. Gocayne, et al. The genome sequence of Drosophila melanogaster. *Science*, 287(5461):2185–2195, 2000.

[Alberts *et al.*, 2008] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell.* Garland Science, Fifth Reference edition, 2008.

[Alexeyenko *et al.*, 2006] Andrey Alexeyenko, Julia Lindberg, Åsa Pérez-Bercoff, and Erik L. L. Sonnhammer. Overview and comparison of ortholog databases. *Drug Discovery Today*, 3(2):137–143, 2006.

[Ali and Deane, 2010] Waqar Ali and Charlotte M. Deane. Evolutionary analysis reveals low coverage as the major challenge for protein interaction network alignment. *Molecular BioSystems*, 6:2296–2304, 2010.

[Altenhoff and Dessimoz, 2009] Adrian M. Altenhoff and Christophe Dessimoz. Phylogenetic and functional assessment of orthologs inference projects and methods. *PLoS Computational Biology*, 5(1):e1000262, 2009.

[Altschul *et al.*, 1990] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene F. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[Altschul *et al.*, 1997] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search algorithms. *Nucleic Acids Research*, 25:3389–3402, 1997.

[Andersen and Lang, 2008] Reid Andersen and Kevin J. Lang. An algorithm for improving graph partitions. *SODA*, 2008.

[Andersen and Peres, 2009] Reid Andersen and Yuval Peres. Finding sparse cuts locally using evolving sets. *STOC 2009*, pages 235–244, 2009.

[Andersen *et al.*, 2006a] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using PageRank vectors. *FOCS 2006*, pages 475–486, 2006.

[Andersen *et al.*, 2006b] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using PageRank vectors (full version), 2006. Available from homepage of Kevin Lang. Accessed January 2011.

[Andersen *et al.*, 2007] Reid Andersen, Fan Chung, and Kevin Lang. Using PageRank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007.

[Aoki and Kanehisa, 2005] Kiyoko F. Aoki and Minoru Kanehisa. Using the KEGG database resource. *Current Protocols in Bioinformatics*, 1(12):1–54, 2005.

[Aranda *et al.*, 2011] Bruno Aranda, Hagen Blankenburg, Samuel Kerrien, Fiona S. L. Brinkman, Arnaud Ceol, et al. PSICQUIC and PSISCORE: accessing and scoring molecular interactions. *Nature Methods*, 8:528–529, 2011.

[Armbrust *et al.*, 2009] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, et al. Above the clouds: a Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[Arora *et al.*, 2004] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *STOC*, 2004.

[Babnigg and Giometti, 2006] György Babnigg and Carol S. Giometti. A database of unique protein sequence identifiers for proteome studies. *Proteomics*, 6(16):4514–4522, 2006.

[Babushok *et al.*, 2007] D. V. Babushok, E. M. Ostertag, and H. H. Kazazian, Jr. Current topics in genome evolution: molecular mechanisms of new gene formation. *Cellular and Molecular Life Sciences*, 64:542–554, 2007.

[Bader *et al.*, 2004] Joel S. Bader, Amitabha Chaudhuri, Jonathan M. Rothberg, and John Chant. Gaining confidence in high-throughput protein interaction networks. *Nature Biotechnology*, 22:78–85, 2004.

[Bandyopadhyay *et al.*, 2006] Sourav Bandyopadhyay, Roded Sharan, and Trey Ideker. Systematic identification of functional orthologs based on protein network comparison. *Genome Research*, 16:428–435, 2006.

[Bartlett *et al.*, 2004] Peter L. Bartlett, Michael Collins, Ben Taskar, and David McAllester. Exponentiated gradient algorithms for large-margin structured classification. *Advances in Neural Information Processing Systems*, 17:113–120, 2004.

[Bauer *et al.*, 2008] Sebastian Bauer, Steffen Grossmann, Matrin Vingron, and Peter N. Robinson. Ontologizer 2.0–a multifunctional tool for go term enrichment analysis and data exploration. *Bioinformatics*, 24(14):1650–1651, 2008.

[Bayati *et al.*, 2005] Mohsen Bayati, Devavrat Shah, and Mayank Sharma. Maximum weight matching via max-product belief propagation. *International Symposium on Information Theory*, 2005.

[Bayati *et al.*, 2007] Mohsen Bayati, Christian Borgs, Jennifer Chayes, and Riccardo Zecchina. Belief propagation for weighted b-matchings on arbitrary graphs and its relation to linear programs with integer solutions. Technical Report 0709.1190, Microsoft Research, arXiv, 2007.

[Beltrao and Serrano, 2007] Pedro Beltrao and Luis Serrano. Specificity and evolvability in eukaryotic protein interaction networks. *PLoS Computational Biology*, 3(2):e25, 2007.

[Berg and Lässig, 2006] Johannes Berg and Michael Lässig. Cross-species analysis of biological networks by Bayesian alignment. *Proceedings of the National Academy of Sciences, U.S.A.*, 103(29):10967–10972, 2006.

[Berg *et al.*, 2004] Johannes Berg, Michael Lässig, and Andreas Wagner. Structure and evolution of protein interaction networks: a statistical model for link dynamics and gene duplications. *BMC Evolutionary Biology*, 4(51), 2004.

[Besag, 1986] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986.

[Bialecki *et al.*, 2012] Andrzej Bialecki, Mike Cafarella, Doug Cutting, and Owen O'Malley. Hadoop: a framework for running applications on large clusters built of commodity hardware, 2012. Wiki at http://lucene. apache. org/hadoop.

[Blankenburg *et al.*, 2009] Hagen Blankenburg, Robert D. Finn, Andreas Prlić, Andrew M. Jenkinson, Fidel Ramírez, Dorothea Emig, Sven-Eric Schelhorn, Joachim Büch, Thomas Lengauer, and Mario Albrecht. DASMI: exchanging, annotating and assessing molecular interaction data. *Bioinformatics*, 25(10):1321–1328, 2009.

[Blum *et al.*, 1973] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

[Boyle *et al.*, 2004] Elizabeth I. Boyle, Shuai Weng, Jeremy Gollub, Heng Jin, David Botstein, J. Michael Cherry, and Gavin Sherlock. GO::TermFinder–open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes. *Bioinformatics*, 20(18):3710–3715, 2004.

[Brand *et al.*, 1999] Marjorie Brand, Ken Yamamoto, Adrien Staub, and Laszlo Tora. Identification of TATA-binding protein-free TAFII-containing complex subunits suggests a role in nucleosome acetylation and signal transduction. *Journal of Biological Chemistry*, 274(26):18285–18289, 1999.

[Brandes *et al.*, 2002] Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M. Scott Marshall. GraphML progress report: structural layer proposal. *Proceedings of the 9th International Symposium on Graph Drawing (GD 2001)*, LNCS 2265:501–512, 2002.

[Brandon, 2005] Robert N. Brandon. Evolutionary modules: conceptual analyses and empirical hypotheses. In *Modularity: Understanding the Development and Evolution of Natural Complex Systems*. MIT Press: Vienna Series in Theoretical Biology, 2005.

[Brasch *et al.*, 2009] Steffen Brasch, Lars Linden, and Georg Fuellen. VANLO - interactive visual exploration of aligned biological networks. *BMC Bioinformatics*, 10(327), 2009.

[Breiman *et al.*, 1984] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Wadsworth, 1984.

[Breitkreutz *et al.*, 2003] Bobby-Joe Breitkreutz, Chris Stark, and Mike Tyers. Osprey: a network visualization system. *Genome Biology*, 4(3):R22, 2003.

[Breuer *et al.*, 2013] Karin Breuer, Amir K. Foroushani, Matthew R. Laird, Carol Chen, Anastasia Sribnaia, Raymond Lo, Geoffrey L. Winsor, Robert E. W. Hancock, Fiona S. L. Brinkman, and David J. Lynn. InnateDB: systems biology of innate immunity and beyond–recent updates and continuing curation. *Nucleic Acids Research*, 41:D1228–D1233, 2013.

[Bruckner *et al.*, 2010] Sharon Bruckner, Falk Hüffner, Richard M. Karp, Ron Shamir, and Roded Sharan. Topology-free querying of protein interaction networks. *Journal of Computational Biology*, 17(3):237–252, 2010.

[Callebaut and Rasskin-Gutman, 2005] Werner Callebaut and Diego Rasskin-Gutman, editors. *Modularity: Understanding the Development and Evolution of Natural Complex Systems*. MIT Press: Vienna Series in Theoretical Biology, 2005.

[Cannavo *et al.*, 2007] E Cannavo, B Gerrits, G Marra, R Schlapbach, and J Jiricny. Characterization of the Interactome of the Human MutL Homologues MLH1, PMS1, and PMS2. *Journal of Biological Chemistry*, 282(5):2976–2986, 2007.

[Card *et al.*, 1999] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1999.

[Cartwright and Collins, 2007] Paulyn Cartwright and Allen Collins. Fossils and phylogenies: integrating multiple lines of evidence to investigate the origin of early major metazoan lineages. *Integrative and Comparative Biology*, 47(5):744–751, 2007.

[Celis, 2004] José F. De Celis. The Notch signaling module. In *Modularity in Development and Evolution*. University of Chicago Press, 2004.

[Charlop-Powers *et al.*, 2010] Zachary Charlop-Powers, Lei Zeng, Qiang Zhang, and Ming-Ming Zhou. Structural insights into selective histone H3 recognition by the human Polybromo bromodomain 2. *Cell Research*, 20:529–538, 2010.

[Chatr-aryamontri *et al.*, 2013] Andrew Chatr-aryamontri, Bobby-Joe Breitkreutz, Sven Heinicke, Lorrie Boucher, Andrew Winter, et al. The BioGRID interaction database: 2013 update. *Nucleic Acids Research*, 41:D816–D823, 2013.

[Chautard *et al.*, 2011] Emilie Chautard, Marie Fatoux-Ardore, Lionel Ballut, Nicolas Thierry-Mieg, and Sylvie Ricard-Blum. MatrixDB, the extracellular matrix interaction database. *Nucleic Acids Research*, 39:D235–D240, 2011.

[Chen *et al.*, 2006] Feng Chen, Aaron J. Mackey, Christian J. Stoeckert, Jr., and David S. Roos. OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups. *Nucleic Acids Research*, 34:D363–D368, 2006.

[Chen *et al.*, 2007] Feng Chen, Aaron J. Mackey, Jeroen K. Vermunt, and David S. Roos. Assessing performance of orthology detection strategies applied to eukaryotic genomes. *PLoS ONE*, 4:e383, 2007.

[Collins *et al.*, 2008] Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *The Journal of Machine Learning Research*, 9:1775–1822, 2008.

[Costanzo *et al.*, 2010] Michael Costanzo, Anastasia Baryshnikova, Jeremy Bellay, Yungil Kim, Eric D. Spear, et al. The genetic landscape of a cell. *Science*, 327:425–431, 2010.

[Cottret and Jourdan, 2010] Ludovic Cottret and Fabien Jourdan. Graph methods for the investigation of metabolic networks in parasitology. *Parasitology*, 137:1393–1407, 2010.

[Cottret *et al.*, 2010] Ludovic Cottret, Paulo Vieira Milreu, Vicente Acuũna, Alberto Marchetti-Spaccamela, Leen Stougie, Hubert Charles, and Marie-France Sagot. Graph-based analysis of the metabolic exchanges between two co-resident intracellular symbionts baumannia cidadellinicola and sulcia muelleri, with their insect host, homalodisca coagulata. *PLoS Computational Biology*, 6(9):e1000904, 2010.

[Creighton, 1993] Thomas E. Creighton. *Proteins: Structures and Molecular Properties*. Freeman, Second edition, 1993.

[Datta *et al.*, 2009] Ruchira S. Datta, Christopher Meacham, Bushra Samad, Christoph Neyer, and Kimmen Sjölander. Berkeley PHOG: PhyloFacts orthology group prediction web server. *Nucleic Acids Research*, 37:D5–D15, 2009.

[de Duve, 2011] Christian de Duve. Life as a cosmic imperative? *Philosophical Transactions of the Royal Society A*, 369(1936):620–623, 2011.

[Dean and Ghemawat, 2008] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[Deelman *et al.*, 2005] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, et al. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13:219–237, 2005.

[Demir *et al.*, 2010] Emek Demir, Michael P. Cary, Suzanne Paley, Ken Fukuda, Christian Lemer, et al. The BioPAX community standard for pathway data sharing. *Nature Biotechnology*, 28(9):935–942, 2010.

[Deng *et al.*, 2003] Minghua Deng, Fengzhu Sun, and Ting Chen. Assessment of the reliability of protein-protein interactions and protein function prediction. *Pacific Symposium on Biocomputing*, pages 140–151, 2003.

[Dessimoz *et al.*, 2005] Christophe Dessimoz, Gina Cannarozzi, Manuel Gil, Daniel Margadant, Alexander Roth, Adrian Schneider, and Gaston H. Gonnet. OMA, a comprehensive, automated project for the identification of orthologs from complete genome data: introduction and first achievements. *Proceedings of the 9th International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 61–72, 2005.

[Donoho *et al.*, 2009] David L. Donoho, Inam Ur Rahman, and Victoria Stodden. Reproducible research in computational harmonic analysis. *Computing in Science and Engineering*, 11(1):8–18, 2009.

[Dudley and Butte, 2010] Joel T. Dudley and Atul J. Butte. In silico research in the era of cloud computing. *Nature Biotechnology*, 28(11):1181–1185, 2010.

[Dutkowski and Tiuryn, 2007] Janusz Dutkowski and Jerzy Tiuryn. Identification of functional modules from conserved ancestral protein-protein interactions. *Bioinformatics*, 23:i149–i158, 2007.

[Eyre *et al.*, 2006] Tina A. Eyre, Matthew W. Wright, Michael J. Lush, and Elspeth A. Bruford. HCOP: a searchable database of human orthology predictions. *Briefings in Bioinformatics*, 8(1):2–5, 2006.

[Fay and Wu, 2003] Justin C. Fay and Chung-I Wu. Sequence divergence, functional constraint, and selection in protein evolution. *Annual Review of Genomics and Human Genetics*, 4:213–235, 2003.

[Fields and Song, 1989] Stanley Fields and Ok-Kyu Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340:245–246, 1989.

[Fields, 2009] Stanley Fields. Interactive learning: lessons from two hybrids over two decades. *Proteomics*, 9:5209–5213, 2009.

[Finley and Haigis, 2009] Lydia W. S. Finley and Marcia C. Haigis. The coordination of nuclear and mitochondrial communication during aging and calorie restriction. *Ageing Research Reviews*, 8:173–188, 2009.

[Finley and Joachims, 2008] Thomas Finley and Thorsten Joachims. Training structural SVMs when exact inference is intractable. *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 304–311, 2008.

[Finn *et al.*, 2010] Robert D. Finn, Jaina Mistry, John Tate, Penny Coggill, Andreas Heger, et al. The Pfam protein families database. *Nucleic Acids Research*, 38:D211–D222, 2010.

[Fitch, 2000] Walter M. Fitch. Homology: a personal view on some of the problems. *Trends in Genetics*, 16(5):227–231, 2000.

[Flannick *et al.*, 2006] Jason Flannick, Antal Novak, Balaji S. Srinivasan, Harley H. McAdams, and Serafim Batzoglou. Graemlin: general and robust alignment of multiple large interaction networks. *Genome Research*, 16:1169–1181, 2006.

[Flannick *et al.*, 2009] Jason Flannick, Antal Novak, Chuong B. Do, Balaji S. Srinivasan, and Serafim Batzoglou. Automatic parameter learning for multiple local network alignment. *Journal of Computational Biology*, 16(8):1001–1022, 2009.

[Franceschini *et al.*, 2013] Andrea Franceschini, Damian Szklarczyk, Sune Frankild, Michael Kuhn, Milan Simonovic, et al. STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Research*, 41:D808–D815, 2013.

[Futuyma, 2009] Douglas J. Futuyma. *Evolution*. Sinauer, Second edition, 2009.

[Gabaldón, 2008] Toni Gabaldón. Large-scale assignment of orthology: back to phylogenetics? *Genome Biology*, 9(10):235, 2008.

[Gandhi *et al.*, 2006] T. K. B. Gandhi, Jun Zhong, Suresh Mathivanan, L. Karthik, K. N. Chandrika, et al. Analysis of the human protein interactome and comparison with yeast, worm and fly interaction datasets. *Nature Genetics*, 38:285–293, 2006.

[Gangloff *et al.*, 2001] Yann-Gaël Gangloff, Jean-Christophe Pointud, Sylvie Thuault, Lucie Carré, Christophe Romier, Selen Muratoglu, Marjorie Brand, Laszlo Tora, Jean-Louis Couderc, and Irwin Davidson. The TFIID components Human TAFII140 and *Drosophila* BIP2 TAFII155 are novel metazoan homologues of yeast TAFII47 containing a histone fold and a PHD finger. *Molecular and Cellular Biology*, 21(15):5109–5121, 2001.

[Gene Ontology Consortium, 2000] Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[Gene Ontology Consortium, 2012] Gene Ontology Consortium. The Gene Ontology: enhancements for 2011. *Nucleic Acids Research*, 40:D559–D564, 2012.

[Georgieva *et al.*, 2000] Sofia Georgieva, Doris B. Kirschner, Tereza Jagla, Elena Nabirochkina, Susanne Hanke, et al. Two novel *Drosophila* TAFIIs have homology with Human TAFII30 and are differentially regulated during development. *Molecular and Cellular Biology*, 20(5):1639–1648, 2000.

[Gerke *et al.*, 2006] Mirco Gerke, Erich Bornberg-Bauer, Xiaoyi Jiang, and Georg Fuellen. Finding common protein interaction patterns across organisms. *Evolutionary Bioinformatics Online*, 2:45–52, 2006.

[Gil *et al.*, 2007] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the challenges of scientific workflows. *Computer*, 40(12):24–32, 2007.

[Giot *et al.*, 2003] L. Giot, J. S. Bader, C. Brouwer, A. Chaudhuri, B. Kuang, et al. A protein interaction map of *Drosophila melanogaster*. *Science*, 302:1727–1736, 2003.

[Glanville *et al.*, 2007] Jake Gunn Glanville, Dan Kirshner, Nandini Krishnamurthy, and Kimmen Sjölander. Berkeley Phylogenomics Group web servers: resources for structural phylogenomic analysis. *Nucleic Acids Research*, 35:W27–W32, 2007.

[Goemans and Williamson, 1996] Michel X. Goemans and David P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In *Approximation Algorithms for NP-hard Problems*, pages 144–191. Course Technology, 1996.

[Goncalves *et al.*, 2012] Carlos Goncalves, Luis Assuncao, and Jose C. Cunha. Data analytics in the cloud with flexible MapReduce workflows. *4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012)*, 2012.

[Goodrich and Tjian, 2010] James A. Goodrich and Robert Tjian. Unexpected roles for core promoter recognition factors in cell-type-specific transcription and gene regulation. *Nature Reviews Genetics*, 11:549–558, 2010.

[Goto *et al.*, 1997] S. Goto, H. Bono, H. Ogata, W. Fujibuchi, T. Nishioka, K. Sato, and M. Kanehisa. Organizing and computing metabolic pathway data in terms of binary relations. *Pacific Symposium on Biocomputing*, pages 175–186, 1997.

[Grömping, 2006] Ulrike Grömping. Relative Importance for Linear Regression in R: The Package relaimpo. *Journal of Statistical Software*, 17(1), 2006.

[Grömping, 2007] Ulrike Grömping. Estimators of relative importance in linear regression based on variance decomposition. *The American Statistician*, 61(2):139–147, 2007.

[Grossmann *et al.*, 2007] Steffen Grossmann, Sebastian Bauer, Peter N. Robinson, and Martin Vingron. Improved detection of overrepresentation of gene-ontology annotations with parent-child analysis. *Bioinformatics*, 23(22):3024–3031, 2007.

[Guo and Hartemink, 2009] Xin Guo and Alexander J. Hartemink. Domain-oriented edge-based alignment of protein interaction networks. *Bioinformatics*, 25:i240–i246, 2009.

[Hakes *et al.*, 2008] Luke Hakes, John W Pinney, David L Robertson, and Simon C Lovell. Protein-protein interaction networks and biology–what's the connection? *Nature Biotechnology*, 26(1):69–72, 2008.

[Hart *et al.*, 2006] G. Traver Hart, Arun K. Ramani, and Edward M. Marcotte. How complete are current yeast and human protein-interaction networks? *Genome Biology*, 7(120), 2006.

[Hartwell *et al.*, 1999] Leland H. Hartwell, John J. Hopfield, Stanislas Leibler, and Andrew W. Murray. From molecular to modular cell biology. *Nature*, 402:C47–C52, 1999.

[Hedges, 2002] S. Blair Hedges. The origin and evolution of model organisms. *Nature Reviews*, 3:838–849, 2002.

[Heer *et al.*, 2005] Jeffrey Heer, Stuart K. Card, and James A. Landay. `prefuse`: a toolkit for interactive information visualization. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 421–430, 2005.

[Heinicke *et al.*, 2007] Sven Heinicke, Michael S. Livstone, Charles Lu, Rose Oughtred, Fan Kang, Samuel V. Angiuoli, Owen White, David Botstein, and Kara Dolinski. The Princeton Protein Orthology Database (P-POD): a comparative genomics analysis tool for biologists. *PLoS ONE*, 8:e766, 2007.

[Hernández-Hernández and Ferrús, 2001] Angel Hernández-Hernández and Alberto Ferrús. Prodos is a conserved transcriptional regulator that interacts with dTAFII16 in *Drosophila melanogaster*. *Molecular and Cellular Biology*, 21(2):614–623, 2001.

[Hirsh and Sharan, 2006] Eitan Hirsh and Roded Sharan. Identification of conserved protein complexes based on a model of protein network evolution. *Bioinformatics*, 23:e170–e176, 2006.

[Hodgkinson and Karp, 2011a] Luqman Hodgkinson and Richard M. Karp. Algorithms to detect multi-protein modularity conserved during evolution. Technical Report UCB/EECS-2011-7, EECS Department, University of California, Berkeley, 2011.

[Hodgkinson and Karp, 2011b] Luqman Hodgkinson and Richard M. Karp. Algorithms to detect multiprotein modularity conserved during evolution. *Proceedings of the International Symposium of Bioinformatics Research and Applications (ISBRA 2011)*, pages 111–122, 2011.

[Hodgkinson and Karp, 2012] Luqman Hodgkinson and Richard M. Karp. Algorithms to detect multiprotein modularity conserved during evolution. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):1046–1058, 2012.

[Hodgkinson and Kong, 2012] Luqman Hodgkinson and Nicholas Kong. Vieprot: visualizing conserved multiprotein modularity with a dynamic force-directed layout. *Manuscript in preparation*, 2012.

[Hodgkinson *et al.*, 2012] Luqman Hodgkinson, Javier Rosa, and Eric A. Brewer. Parallel software architecture for experimental workflows in computational biology on clouds. *PPAM 2011*, LNCS 2704:281–291, 2012.

[Holstege *et al.*, 1998] Frank C. P. Holstege, Ezra G. Jennings, John J. Wyrick, Tong Ihn Lee, Christoph J. Hengartner, Michael R. Green, Todd R. Golub, Eric S. Lander, and Richard A. Young. Dissecting the regulatory circuitry of a eukaryotic genome. *Cell*, 95:717–728, 1998.

[Hopcroft and Tarjan, 1973] John Hopcroft and Robert Tarjan. Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[Hu *et al.*, 2009] Zhenjun Hu, Jui-Hung Hung, Yan Wang, Yi-Chien Chang, Chia-Ling Huang, Matt Huyck, and Charles DeLisi. VisANT 3.5: multi-scale network visualization, analysis and inference based on the gene ontology. *Nucleic Acids Research*, 37:W115–W121, 2009.

[Hull *et al.*, 2006] Duncan Hull, Katy Wolstencroft, Robert Stevens, Carole Goble, Matthew R. Pocock, Peter Li, and Tom Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34:W729–W732, 2006.

[Hulsen *et al.*, 2006] Tim Hulsen, Martijn A. Huynen, Jacob de Vlieg, and Peter M. A. Groenen. Benchmarking ortholog identification methods using functional genomics data. *Genome Biology*, 7:R31, 2006.

[International Human Genome Sequencing Consortium, 2001] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001.

[International Human Genome Sequencing Consortium, 2004] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431:931–945, 2004.

[Isserlin *et al.*, 2011] Ruth Isserlin, Rashad A. El-Badrawi, and Gary D. Bader. The Biomolecular Interaction Network Database in PSI-MI 2.5. *Database (Oxford)*, page baq037, 2011.

[Jensen *et al.*, 2008] Lars Juhl Jensen, Philippe Julien, Michael Kuhn, Christian von Mering, Jean Muller, Tobias Doerks, and Peer Bork. eggNOG: automated construction and annotation of orthologous groups of genes. *Nucleic Acids Research*, 36:D250–D254, 2008.

[Jerrum and Sinclair, 1988] Mark Jerrum and Alistair Sinclair. Conductance and the rapid mixing property for Markov chains: the approximation of the permanent resolved. *STOC 1988*, pages 235–244, 1988.

[Juve and Deelman, 2010] Gideon Juve and Ewa Deelman. Scientific workflows in the cloud. In *Grids, Clouds and Virtualization (Computer Communications and Networks)*. Springer, 2010.

[Kalaev *et al.*, 2009] Maxim Kalaev, Vineet Bafna, and Roded Sharan. Fast and accurate alignment of multiple protein networks. *Journal of Computational Biology*, 16(8):989–999, 2009.

[Kandasamy *et al.*, 2010] Kumaran Kandasamy, S. Sujatha Mohan, Rajesh Raju, Shivakumar Keerthikumar, Ghantasala S. S. Kumar, et al. NetPath: a public resource of curated signal transduction pathways. *Genome Biology*, 11:R3, 2010.

[Kanehisa and Goto, 2000] Minoru Kanehisa and Susumu Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

[Kanehisa *et al.*, 2006] Minoru Kanehisa, Susumu Goto, Masahiro Hattori, Kiyoko F. Aoki-Kinoshita, Masumi Itoh, Shuichi Kawashima, Toshiaki Katayama, Michihiro Araki, and Mika Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Research*, 34:D354–D357, 2006.

[Kanehisa *et al.*, 2010] Minoru Kanehisa, Susumu Goto, Miho Furumichi, Mao Tanabe, and Mika Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Research*, 38:D355–D360, 2010.

[Kannan *et al.*, 2000] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings – good, bad and spectral. *FOCS*, 2000.

[Karp *et al.*, 1996] Peter D. Karp, Monica Riley, Suzanne M. Paley, and Alida Pelligrini-Toole. EcoCyc: an encyclopedia of Escherichia coli genes and metabolism. *Nucleic Acids Research*, 24(1):32–39, 1996.

[Karp *et al.*, 2000] Peter D. Karp, Monica Riley, Milton Saier, Ian T. Paulsen, Suzanne M. Paley, and Alida Pellegrini-Toole. The EcoCyc and MetaCyc databases. *Nucleic Acids Research*, 28(1):56–59, 2000.

[Karp *et al.*, 2013] Peter D. Karp, Suzanne Paley, and Tomer Altman. Data mining in the MetaCyc family of pathway databases. *Methods in Molecular Biology*, 939:183–200, 2013.

[Kelley *et al.*, 2003] Brian P. Kelley, Roded Sharan, Richard M. Karp, Taylor Sittler, David E. Root, Brent R. Stockwell, and Trey Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proceedings of the National Academy of Sciences of the United States of America*, 100(20):11394–11399, 2003.

[Kelner *et al.*, 2013] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. *Arxiv*, 1301:6628v1, 2013.

[Kerrien *et al.*, 2007] Samuel Kerrien, Sandra Orchard, Luisa Montecchi-Palazzi, Bruno Aranda, Antony F. Quinn, et al. Broadening the horizon–level 2.5 of the HUPO-PSI format for molecular interactions. *BMC Biology*, 5(44), 2007.

[Kerrien *et al.*, 2012] Samuel Kerrien, Bruno Aranda, Lionel Breuza, Alan Bridge, Fiona Broackes-Carter, et al. The IntAct molecular interaction database in 2012. *Nucleic Acids Research*, 40(D1):D841–D846, 2012.

[Keseler *et al.*, 2013] Ingrid M. Keseler, Amanda Mackie, Martin Peralta-Gil, Alberto Santos-Zavaleta, Socorro Gama-Castro, et al. EcoCyc: fusing model organism databases with systems biology. *Nucleic Acids Research*, 41:D605–D612, 2013.

[Khalidi, 2011] Yousef A. Khalidi. Building a cloud computing platform for new possibilities. *Computer*, 44(3):29–34, 2011.

[Kiemer and Cesareni, 2007] Lars Kiemer and Gianni Cesareni. Comparative interactomics: comparing apples and pears? *Trends in Biotechnology*, 25(10):448–454, 2007.

[Kimura, 1983] Motoo Kimura. *The Neutral Theory of Molecular Evolution*. Cambridge University Press, 1983.

[Klipp *et al.*, 2009] Edda Klipp, Wolfram Liebermeister, Christoph Wierling, Axel Kowald, Hans Lehrach, and Ralf Herwig. *Systems Biology: A Textbook*. Wiley-VCH, 2009.

[Koh *et al.*, 2012] Gavin C. K. W. Koh, Pablo Porras, Bruno Aranda, Henning Hermjakob, and Sandra E. Orchard. Analyzing protein-protein interaction networks. *Journal of Proteome Research*, 11(4):2014–2031, 2012.

[Kohler *et al.*, 2000] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.

[Kolář *et al.*, 2008] Michal Kolář, Michael Lässig, and Johannes Berg. From protein interactions to functional annotation: graph alignment in Herpes. *BMC Systems Biology*, 2(90), 2008.

[Kolmogorov, 2006] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[Koonin *et al.*, 1996] Eugene V. Koonin, Arcady R. Mushegian, and Peer Bork. Non-orthologous gene displacement. *Trends in Genetics*, 12(9):334–336, 1996.

[Koonin, 2005] Eugene V. Koonin. Orthologs, paralogs, and evolutionary genomics. *The Annual Review of Genetics*, 11(22):309–338, 2005.

[Koyutürk *et al.*, 2006a] Mehmet Koyutürk, Yohan Kim, Shankar Subramaniam, Wojciech Szpankowski, and Ananth Grama. Detecting conserved interaction patterns in biological networks. *Journal of Computational Biology*, 13:1299–1322, 2006.

[Koyutürk *et al.*, 2006b] Mehmet Koyutürk, Yohan Kim, Umut Topkara, Shankar Subramaniam, Wojciech Szpankowski, and Ananth Grama. Pairwise alignment of protein interaction networks. *Journal of Computational Biology*, 13(2):182–199, 2006.

[Krebs *et al.*, 2010] Arnaud R. Krebs, Jeroen Demmers, Krishanpal Karmodiya, Nan-Chi Chang, Alice Chien Chang, and László Tora. ATAC and Mediator coactivators form a stable complex and regulate a set of non-coding RNA genes. *EMBO Reports*, 11(7):541–547, 2010.

[Krishnamurthy *et al.*, 2006] Nandini Krishnamurthy, Duncan P. Brown, Dan Kirshner, and Kimmen Sjölander. PhyloFacts: an online structural phylogenomic encyclopedia for protein functional and structural classification. *Genome Biology*, 7(9):R83, 2006.

[Kulesza and Pereira, 2007] Alex Kulesza and Fernando Pereira. Structured learning with approximate inference. *Advances in Neural Information Processing Systems*, 20:1–8, 2007.

[Kusch *et al.*, 2003] Thomas Kusch, Sebastián Guelman, Susan M. Abmayr, and Jerry L. Workman. Two *Drosophila* Ada2 homologues function in different multiprotein complexes. *Molecular and Cellular Biology*, 23(9):3305–3319, 2003.

[Kuzniar *et al.*, 2008] Arnold Kuzniar, Roeland C. H. J. van Ham, Sándor Pongor, and Jack A. M. Leunissen. The quest for orthologs: finding the corresponding gene across genomes. *Trends in Genetics*, 24(11):539–551, 2008.

[Lacroix *et al.*, 2006] Vincent Lacroix, Cristina G. Fernandes, and Marie-France Sagot. Motif search in graphs: application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):360–368, 2006.

[Lacroix *et al.*, 2008] Vincent Lacroix, Ludovic Cottret, Patricia Thébault, and Marie-France Sagot. An introduction to metabolic networks and their structural analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(4):594–617, 2008.

[Lalonde *et al.*, 2008] Sylvie Lalonde, David W. Ehrhardt, Dominique Loqué, Jin Chen, Seung Y. Rhee, and Wolf B. Frommer. Molecular and cellular approaches for the detection of protein-protein interactions: latest techniques and current limitations. *The Plant Journal*, 53(4):610–635, 2008.

[Lee and Thévenod, 2006] Wing-Kee Lee and Frank Thévenod. A role for mitochondrial aquaporins in cellular life-and-death decisions? *American Journal of Physiology-Cell Physiology*, 291:C195–C202, 2006.

[Levasseur *et al.*, 2008] Anthony Levasseur, Pierre Pontarotti, Olivier Poch, and Julie D. Thompson. Strategies for reliable exploitation of evolutionary concepts in high throughput biology. *Evolutionary Bioinformatics*, 4:121–137, 2008.

[Li and Manley, 1998] Chi Li and James L. Manley. Even-skipped represses transcription by binding TATA binding protein and blocking the TFIID-TATA box interaction. *Molecular and Cellular Biology*, 18(7):3771–3781, 1998.

[Li *et al.*, 2006] Heng Li, Avril Coghlan, Jue Ruan, Lachlan James Coin, Jean-Karim Hériché, Lara Osmotherly, Ruiqiang Li, Tao Liu, Zhang Zhang, Lars Bolund, Gane Ka-Shu Wong, Weimou Zheng, Paramvir Dehal, Jun Wang, and Richard Durbin. TreeFam: a curated database of phylogenetic trees of animal gene families. *Nucleic Acids Research*, 34:D572–D580, 2006.

[Licata *et al.*, 2012] Luana Licata, Leonardo Briganti, Daniele Peluso, Livia Perfetto, Marta Iannuccelli, et al. MINT, the molecular interaction database: 2012 update. *Nucleic Acids Research*, 40:D857–D861, 2012.

[Lim *et al.*, 2006] Janghoo Lim, Tong Hao, Chad Shaw, Akash J. Patel, Gábor Szabó, et al. A Protein-Protein Interaction Network for Human Inherited Ataxias and Disorders of Purkinje Cell Degeneration. *Cell*, 125(4):801–814, 2006.

[Linke *et al.*, 2011] Burkhard Linke, Robert Giegerich, and Alexander Goesmann. Conveyor: a workflow engine for bioinformatic analyses. *Bioinformatics*, 27(7):903–911, 2011.

[Liu and West, 2002] Yilun Liu and Stephen C. West. Distinct functions of BRCA1 and BRCA2 in double-strand break repair. *Breast Cancer Research*, 4:9–13, 2002.

[Liu *et al.*, 2009] Kevin Liu, Sindhu Raghavan, Serita Nelesen, C. Randal Linder, and Tandy Warnow. Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science*, 324:1561–1564, 2009.

[Livnat *et al.*, 2010] Adi Livnat, Christos Papadimitriou, Nicholas Pippenger, and Marcus W. Feldman. Sex, mixability, and modularity. *Proceedings of the National Academy of Sciences of the United States of America*, 107(4):1452–1457, 2010.

[Lord, 1995] Hambleton D. Lord. Improving the application development process with modular visualization environments. *Computer Graphics*, 29(2):10–12, 1995.

[Ludäscher *et al.*, 2006] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18:1039–1065, 2006.

[Lunardi *et al.*, 2010] Andrea Lunardi, Giulio Di Minin, Paolo Provero, Marco Dal Ferro, Marcello Carotti, Giannino Del Sal, and Licio Collavin. A genome-scale protein interaction profile of *drosophila* p53 uncovers additional nodes of the human p53 network. *Proceedings of the National Academy of Sciences U.S.A.*, 107(14):6322–6327, 2010.

[Madeira *et al.*, 2009] Alexandra Madeira, Elisabet Öhman, Anna Nilsson, Benita Sjögren, Per E. Andrén, and Per Svenningsson. Coupling surface plasmon resonance to mass spectrometry to discover novel protein-protein interactions. *Nature Protocols*, 4(7):1023–1037, 2009.

[Martens *et al.*, 2007] Lennart Martens, Sandra Orchard, Rolf Apweiler, and Henning Hermjakob. Human Proteome Organization Proteomics Standards Initiative: data standardization, a view on developments and policy. *Molecular and Cellular Proteomics*, 6(9):1666–1667, 2007.

[Masi *et al.*, 2010] Alessio Masi, Riccardo Cicchi, Adolfo Carloni, Francesco Saverio Pavone, and Annarosa Arcangeli. Optical methods in the study of protein-protein interactions. In *Integrins and Ion Channels: Molecular Complexes and Signaling*. Springer Science: Advances in Experimental Medicine and Biology, 2010.

[Matthews *et al.*, 2001] Lisa R. Matthews, Philippe Vaglio, Jérôme Reboul, Hui Ge, Brian P. Davis, James Garrels, Sylvie Vincent, and Marc Vidal. Identification of potential interaction networks using sequence-based searches for conserved protein-protein interactions or "interologs. *Genome Research*, 11:2120–2126, 2001.

[Melancon, 2006] Guy Melancon. Just how dense are dense graphs in the real world?: a methodological note. *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization (BELIV 2006)*, pages 1–7, 2006.

[Mewes *et al.*, 2011] H. Werner Mewes, Andreas Ruepp, Fabian Theis, Thomas Rattei, Mathias Walter, et al. MIPS: curated databases and comprehensive secondary data resources in 2010. *Nucleic Acids Research*, 39:D220–D224, 2011.

[Miernyk and Thelen, 2008] Jan A. Miernyk and Jay J. Thelen. Biochemical approaches for discovering protein-protein interactions. *The Plant Journal*, 53:597–609, 2008.

[Müller *et al.*, 2010] Ferenc Müller, Andreas Zaucker, and Làszlò Tora. Developmental regulation of transcription initiation: more than just changing the actors. *Current Opinion in Genetics and Development*, 20:533–540, 2010.

[Nakao *et al.*, 1999] Mitseru Nakao, Hidemasa Bono, Shuichi Kawashima, Tomomi Kamiya, Kazushige Sato, Susumu Goto, and Minoru Kanehisa. Genome-scale gene expression analysis and pathway reconstruction in KEGG. *Genome Informatics Series*, 10:94–103, 1999.

[Narayanan and Karp, 2007] Manikandan Narayanan and Richard M. Karp. Comparing protein interaction networks via a graph match-and-split algorithm. *Journal of Computational Biology*, 14(7):892–907, 2007.

[Narayanan *et al.*, 2010] Manikandan Narayanan, Adrian Vetta, Eric E. Schadt, and Jun Zhu. Simultaneous clustering of multiple gene expression and physical interaction datasets. *PLoS Computational Biology*, 6(4):e1000742, 2010.

[Newman and Girvan, 2004] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004.

[Nilsen and Graveley, 2010] Timothy W. Nilsen and Brenton R. Graveley. Expansion of the eukaryotic proteome by alternative splicing. *Nature*, 463:457–463, 2010.

[Obrdlik *et al.*, 2004] Petr Obrdlik, Mohamed El-Bakkoury, Tanja Hamacher, Corinna Cappellaro, Cristina Vilarino, et al. K^+ channel interactions detected by a genetic system optimized for systematic studies of membrane protein interactions. *Proceedings of the National Academy of Sciences, U.S.A.*, 101(33):12242–12247, 2004.

[O'Brien *et al.*, 2005] Kevin P. O'Brien, Maido Remm, and Erik L. L. Sonnhammer. Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucleic Acids Research*, 33:D476–D480, 2005.

[Ogata *et al.*, 1998] Hiroyuki Ogata, Susumu Goto, Wataru Fujibuchi, and Minoru Kanehisa. Computation with the KEGG pathway database. *Biosystems*, 47(1-2):119–128, 1998.

[Ogryzko *et al.*, 1998] Vasily V. Ogryzko, Tomohiro Kotani, Xiaolong Zhang, R. Louis Schiltz, Tazuko Howard, Xiang-Jiao Yang, Bruce H. Howard, Jun Qin, and Yoshihiro Nakatani. Histone-like TAFs within the PCAF histone acetylase complex. *Cell*, 94:35–44, 1998.

[Page *et al.*, 1999] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the Web. *Stanford Digital Library Technologies Project*, Technical Report, 1999.

[Palsson, 2006] Bernhard Ø. Palsson. *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, 2006.

[Palsson, 2011] Bernhard Ø. Palsson. *Systems Biology: Simulation of Dynamic Network States*. Cambridge University Press, 2011.

[Parr and Quong, 1995] T. J. Parr and R. W. Quong. ANTLR: a predicated-LL(k) parser generator. *Software-Practice and Experience*, 25(7):789–810, 1995.

[Pawson and Nash, 2003] Tony Pawson and Piers Nash. Assembly of cell regulatory systems through protein interaction domains. *Science*, 300:445–452, 2003.

[Peng *et al.*, 2012] Wei Peng, Jianxin Wang, Weiping Wang, Qing Liu, Fang-Xiang Wu, and Yi Pan. Iteration method for predicting essential proteins based on orthology and protein-protein interaction networks. *BMC Systems Biology*, 6:87, 2012.

[Peng, 2012] Pan Peng. The small community phenomenon in networks: models, algorithms and applications. *TAMC 2012. LNCS*, 7287:40–49, 2012.

[Penkett *et al.*, 2006] Christopher J. Penkett, James A. Morris, Valerie Wood, and Jürg Bähler. YOGY: a web-based, integrated database to retrieve protein orthologs and associated Gene Ontology terms. *Nucleic Acids Research*, 34:W330–W334, 2006.

[Prasad *et al.*, 2009] T.S. Keshava Prasad, Renu Goel, Kumaran Kandasamy, Shivakumar Keerthikumar, Sameer Kumar, et al. Human Protein Reference Database–2009 update. *Nucleic Acids Research*, 37:D767–D772, 2009.

[Pray, 2008] Leslie A. Pray. Eukaryotic genome complexity. *Nature Education*, 1, 2008.

[R Development Core Team, 2012] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012.

[Ratliff *et al.*, 2007] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. (Online) subgradient methods for structured prediction. *Robotics Institute*, 55, 2007.

[Ravikumar *et al.*, 2010] Pradeep Ravikumar, Alekh Agarwal, and Martin J. Wainwright. Message-passing for graph-structured linear programs: proximal projections, convergence and rounding schemes. *The Journal of Machine Learning Research*, 11:1043–1080, 2010.

[Razick *et al.*, 2008] Sabry Razick, George Magklaras, and Ian M. Donaldson. iRefIndex: a consolidated protein interaction database with provenance. *BMC Bioinformatics*, 9(405), 2008.

[Reference Genome Group of the Gene Ontology Consortium, 2009] Reference Genome Group of the Gene Ontology Consortium. The Gene Ontology's reference genome project: a unified framework for functional annotation across species. *PLoS Computational Biology*, 5(7):e1000431, 2009.

[Remm *et al.*, 2001] Maido Remm, Christian E. V. Storm, and Erik L. L. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *Journal of Molecular Biology*, 314(5):1041–1052, 2001.

[Rorick, 2012] Mary Rorick. Quantifying protein modularity and evolvability: a comparison of different techniques. *BioSystems*, 2012.

[Rousu *et al.*, 2007] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Efficient algorithms for max-margin structured classification. In *Predicting Structured Data*, pages 105–128. MIT Press, 2007.

[Rubin *et al.*, 2000] Gerald M. Rubin, Mark D. Yandell, Jennifer R. Wortman, George L. Gabor Miklos, Catherine R. Nelson, et al. Comparative genomics of the Eukaryotes. *Science*, 287(5461):2204–2215, 2000.

[Salwinski *et al.*, 2004] Lucasz Salwinski, Christopher S. Miller, Adam J. Smith, Frank K. Pettit, James U. Bowie, and David Eisenberg. The database of interacting proteins: 2004 update. *Nucleic Acids Research*, 32:D449–D451, 2004.

[Sammut *et al.*, 2008] Stephen John Sammut, Robert D. Finn, and Alex Bateman. Pfam 10 years on: 10,000 families and still growing. *Briefings in Bioinformatics*, 9(3):210–219, 2008.

[Sanghavi *et al.*, 2009] Sujay Sanghavi, Devavrat Shah, and Alan S. Willsky. Message-passing for maximum weight independent set. *IEEE Transactions on Information Theory*, 55(11):4822–4834, 2009.

[Sarawagi and Gupta, 2008] Sunita Sarawagi and Rahul Gupta. Accurate max-margin training for structured output spaces. *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 888–895, 2008.

[Sauer *et al.*, 1996] Frank Sauer, David A. Wassarman, Gerald M. Rubin, and Robert Tjian. TAFIIs mediate activation of transcription in the Drosophila embryo. *Cell*, 87:1271–1284, 1996.

[Sayers *et al.*, 2009] Eric W. Sayers, Tanya Barrett, Dennis A. Benson, Stephen H. Bryant, Kathi Canese, et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 37:D5–D15, 2009.

[Sayers *et al.*, 2012] Eric W. Sayers, Tanya Barrett, Dennis A. Benson, Evan Bolton, Stephen H. Bryant, et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 40:D13–D25, 2012.

[Schaeffer, 2007] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1:27–64, 2007.

[Schelhorn *et al.*, 2008] Sven-Eric Schelhorn, Thomas Lengauer, and Mario Albrecht. An integrative approach for predicting interactions of protein regions. *Bioinformatics*, 24:i35–i41, 2008.

[Schlosser and Wagner, 2004] Gerhard Schlosser and Günter P. Wagner, editors. *Modularity in Development and Evolution*. University of Chicago Press, 2004.

[Schlosser and Wagner, 2008] Gerhard Schlosser and Günter P. Wagner. A simple model of co-evolutionary dynamics caused by epistatic selection. *Journal of Theoretical Biology*, 250:48–65, 2008.

[Schwacke *et al.*, 2003] Rainer Schwacke, Anja Schneider, Eric van der Graaff, Karsten Fischer, Elisabetta Catoni, Marcelo Desimone, Wolf B. Frommer, Ulf-Ingo Flügge, and Reinhard Kunze. ARAMEMNON, a novel database for Arabidopsis integral membrane proteins. *Plant Physiology*, 131(1):16–26, 2003.

[Shannon *et al.*, 2003] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[Sharan and Ideker, 2006] Roded Sharan and Trey Ideker. Modeling cellular machinery through biological network comparison. *Nature Biotechnology*, 24(4):427–433, 2006.

[Sharan *et al.*, 2004] Roded Sharan, Trey Ideker, Brian P. Kelley, Ron Shamir, and Richard M. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *Proceedings of the 8th International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 282–289, 2004.

[Sharan *et al.*, 2005a] Roded Sharan, Trey Ideker, Brian Kelley, Ron Shamir, and Richard M. Karp. Identification of protein complexes by comparative analysis of yeast and bacterial protein interaction data. *Journal of Computational Biology*, 12(6):835–846, 2005.

[Sharan *et al.*, 2005b] Roded Sharan, Silpa Suthram, Ryan M. Kelley, Tanja Kuhn, Scott McCuine, Peter Uetz, Taylor Sittler, Richard M. Karp, and Trey Ideker. Conserved patterns of protein interaction in multiple species. *Proceedings of the National Academy of Sciences of the United States of America*, 102(6):1974–1979, 2005.

[Shi and Malik, 2000] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[Sima and Schaeffer, 2005] Jiri Sima and Satu Elisa Schaeffer. On the NP-completeness of some graph cluster problems. *Arxiv*, 2005.

[Simon, 2005] Herbert A. Simon. The structure of complexity in an evolving world: the role of near decomposability. In *Modularity: Understanding the Development and Evolution of Natural Complex Systems*. MIT Press: Vienna Series in Theoretical Biology, 2005.

[Singh *et al.*, 2007] Rohit Singh, Jinbo Xu, and Bonnie Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. *Proceedings of the 11th International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 16–31, 2007.

[Smith and Waterman, 1981] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[Sowa *et al.*, 2009] M E Sowa, E J Bennett, S P Gygi, and J W Harper. Defining the Human Deubiquitinating Enzyme Interaction Landscape. *Cell*, 138(2):389–403, 2009.

[Spielman and Teng, 2008] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *CoRR abs/0809.3232*, 2008.

[Stein, 2010] Lincoln D. Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11(5):207, 2010.

[Storm and Sonnhammer, 2002] Christian E. V. Storm and Erik L. L. Sonnhammer. Automated ortholog inference from phylogenetic trees and calculation of ortholog reliability. *Bioinformatics*, 18:92–99, 2002.

[Storm and Sonnhammer, 2003] Christian E. V. Storm and Erik L. L. Sonnhammer. Comprehensive analysis of orthologous protein domains using the HOPS database. *Genome Research*, 13:2353–2362, 2003.

[Stuart *et al.*, 2003] Joshua M. Stuart, Eran Segal, Daphne Koller, and Stuart K. Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302:249–255, 2003.

[Stumpf *et al.*, 2008] Michael P. H. Stumpf, Thomas Thorne, Eric de Silva, Ronald Stewart, Hyeong Jun An, Michael Lappe, and Carsten Wiuf. Estimating the size of the human interactome. *Proceedings of the National Academy of Sciences, U.S.A.*, 105(19):6959–6964, 2008.

[Sun *et al.*, 2012] Siqi Sun, Xinran Dong, Yao Fu, and Weidong Tian. An iterative network partition algorithm for accurate identification of dense network modules. *Nucleic Acids Research*, 40(3):e18, 2012.

[Suthram *et al.*, 2006] Silpa Suthram, Tomer Shlomi, Eytan Ruppin, Roded Sharan, and Trey Ideker. A direct comparison of protein interaction confidence alignment schemes. *BMC Bioinformatics*, 7:360, 2006.

[Suthram *et al.*, 2010] Silpa Suthram, Joel T. Dudley, Annie P. Chiang, Rong Chen, Trevor J. Hastie, and Atul J. Butte. Network-based elucidation of human disease similarities reveals common functional modules enriched for pluripotent drug targets. *PLoS Computational Biology*, 6(2):e1000662, 2010.

[Ta and Holm, 2009] Hung Xuan Ta and Liisa Holm. Evaluation of different domain-based methods in protein interaction prediction. *Biochemical and Biophysical Research Communications*, 390:357–362, 2009.

[Taskar *et al.*, 2003] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin Markov networks. *Advances in Neural Information Processing Systems*, 16:25–32, 2003.

[Tatusov *et al.*, 1997] Roman L. Tatusov, Eugene V. Koonin, and David J. Lipman. A genomic perspective on protein families. *Science*, 278:631–637, 1997.

[Tatusov *et al.*, 2000] Roman L. Tatusov, Michael Y. Galperin, Darren A. Natale, and Eugene V. Koonin. The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Research*, 28(1):33–36, 2000.

[Tatusov *et al.*, 2003] Roman L. Tatusov, Natalie D. Fedorova, John D. Jackson, Aviva R. Jacobs, Boris Kiryutin, Eugene V. Koonin, et al. The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4:41, 2003.

[Taylor *et al.*, 2006] Ian J. Taylor, Ewa Deelman, and Dennis B. Gannon, editors. *Workflows for e-Science: Scientific Workflows for Grids*. Springer, Heidelberg, 2006.

[Thieffry and Sánchez, 2004] Denis Thieffry and Lucas Sánchez. Qualitative analysis of gene networks: toward the delineation of cross-regulatory modules. In *Modularity in Development and Evolution*. University of Chicago Press, 2004.

[Thomas and Chiang, 2006] Mary C. Thomas and Cheng-Ming Chiang. The general transcription machinery and general cofactors. *Critical Reviews in Biochemistry and Molecular Biology*, 41:105–178, 2006.

[Thomas *et al.*, 1998] D. Roland Thomas, Edward Hughes, and Bruno D. Zumbo. On variable importance in linear regression. *Social Indicators Research*, 45:253–275, 1998.

[Thomas *et al.*, 2003] Paul D. Thomas, Michael J. Campbell, Anish Kejariwal, Huaiyu Mi, Brian Karlak, Robin Daverman, Karen Diemer, Anushya Muruganujan, and Apurva Narechania. PANTHER: a library of protein families and subfamilies indexed by function. *Genome Research*, 13:2129–2141, 2003.

[Tora, 2002] László Tora. A unified nomenclature for TATA box binding protein (TBP)-associated factors (TAFs) involved in RNA polymerase II transcription. *Genes and Development*, 16:673–675, 2002.

[Tsochantaridis *et al.*, 2004] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, page 104, 2004.

[Tsochantaridis *et al.*, 2005] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output spaces. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[Turanalp and Can, 2008] Mehmet E. Turanalp and Tolga Can. Discovering functional interaction patterns in protein-protein interaction networks. *BMC Bioinformatics*, 9:276, 2008.

[UniProt Consortium, 2012] UniProt Consortium. Reorganizing the protein space at the University Protein Resource (UniProt). *Nucleic Acids Research*, 40:D71–D75, 2012.

[Vazirani, 2003] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2003.

[Vidal, 2005] Marc Vidal. Interactome modeling. *FEBS Letters*, 579:1834–1838, 2005.

[Vidal, 2009] Marc Vidal. A unifying view of 21st century systems biology. *FEBS Letters*, 583(24):3891–3894, 2009.

[Voevodski *et al.*, 2009] Konstantin Voevodski, Shang-Hua Teng, and Yu Xia. Finding local communities in protein networks. *BMC Bioinformatics*, 10(297), 2009.

[von Mering *et al.*, 2002] Christian von Mering, Roland Krause, Berend Snel, Michael Cornell, Stephen G. Oliver, Stanley Fields, and Peer Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.

[Wagner, 1996] Günter P. Wagner. Homologues, natural kinds and the evolution of modularity. *American Zoologist*, 36(1):36–43, 1996.

[Wainwright and Jordan, 2008] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[Wainwright *et al.*, 2004] Martin Wainwright, Tommi Jaakkola, and Alan Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14(2):143–166, 2004.

[Wainwright *et al.*, 2007] Martin Wainwright, Tommi Jaakkola, and Alan Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2007.

[Walhout and Boulton, 2006] Albertha J. M. Walhout and Simon J. Boulton. Biochemistry and molecular biology. In *WormBook*. The C. elegans Research Community, 2006.

[Wang and Page, 2002] P. Jeremy Wang and David C. Page. Functional substitution for TAFII250 by a retroposed homolog that is expressed in human spermatogenesis. *Human Molecular Genetics*, 11(19):2341–2346, 2002.

[Wang *et al.*, 2007] Chunlin Wang, Chris Ding, Qiaofeng Yang, and Stephen R. Holbrook. Consistent dissection of the protein interaction network by combining global and local metrics. *Genome Biology*, 8:R271, 2007.

[Wang *et al.*, 2009] Kai Wang, Manikandan Narayanan, Hua Zhong, Martin Tompa, Eric E. Schadt, and Jun Zhu. Meta-analysis of inter-species liver co-expression networks elucidates traits associated with common human diseases. *PLoS Computational Biology*, 5(12):e1000616, 2009.

[Wang *et al.*, 2011] Jianxin Wang, Min Li, Jianer Chen, and Yi Pan. A fast hierarchical clustering algorithm for functional modules discovery in protein interaction networks. *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, 8(3):607–620, 2011.

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

[Weiss and Freeman, 2001] Yair Weiss and William T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2):736–744, 2001.

[Welsh *et al.*, 2001] Matt Welsh, David Culler, and Eric Brewer. SEDA: an architecture for well-conditioned, scalable internet services. *Proceedings of the 18th Symposium on Operating Systems Principles (SOSP), ACM*, pages 230–243, 2001.

[Werner, 2004] Thomas Werner. Proteomics and regulomics: the yin and yang of functional genomics. *Mass Spectrometry Reviews*, 23(1):25–33, 2004.

[Wheeler *et al.*, 2006] David L. Wheeler, Tanya Barrett, Dennis A. Benson, Stephen H. Bryant, Kathi Canese, et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 34:D173–D180, 2006.

[Wieczorek *et al.*, 1998] Elzbieta Wieczorek, Marjorie Brand, Xavier Jacq, and László Tora. Function of TAFII-containing complex without TBP in transcription by RNA polymerase II. *Nature*, 393(6681):187–191, 1998.

[Winther, 2005] Rasmus G. Winther. Evolutionary developmental biology meets levels of selection: modular integration or competition, or both? In *Modularity: Understanding the Development and Evolution of Natural Complex Systems*. MIT Press: Vienna Series in Theoretical Biology, 2005.

[Wright *et al.*, 2006] Kevin J. Wright, Michael T. Marr, II, and Robert Tijan. TAF4 nucleates a core subcomplex of TFIID and mediates activated transcription from a TATA-less promoter. *Proceedings of the National Academy of Sciences of the United States of America*, 103(33):12347–12352, 2006.

[Xenarios *et al.*, 2000] Ioannis Xenarios, Danny W. Rice, Lukasz Salwinski, Marisa K. Baron, Edward M. Marcotte, and David Eisenberg. DIP: the database of interacting proteins. *Nucleic Acids Research*, 28(1):289–291, 2000.

[Yokomori *et al.*, 1993] Kyoko Yokomori, Jin-Long Chen, Arie Admon, Sharleen Zhou, and Robert Tjian. Molecular cloning and characterization of dTAFII30α and dTAFII30β: two small subunits of *Drosophila* TFIID. *Genes and Development*, 7:2587–2597, 1993.

[Yosef *et al.*, 2008] Nir Yosef, Roded Sharan, and William Stafford Noble. Improved network-based identification of protein orthologs. *Bioinformatics*, 24:i200–i206, 2008.

[Zhang and Lu, 2010] Minlu Zhang and Long J. Lu. Investigating the validity of current network analysis on static conglomerate networks by protein network stratification. *BMC Bioinformatics*, 11:466, 2010.

[Zhang *et al.*, 1996] Hailan Zhang, Katrina M. Catron, and Cory Abate-Shen. A role for the Msx-1 homeodomain in transcriptional regulation: residues in the N-terminal arm mediate TATA binding protein interaction and transcriptional repression. *PNAS*, 93:1764–1769, 1996.

[Zinman *et al.*, 2011] Guy E. Zinman, Shan Zhong, and Ziv Bar-Joseph. Biological interaction networks are conserved at the module level. *BMC Systems Biology*, 5:134, 2011.

[Zmasek and Eddy, 2002] Christian M. Zmasek and Sean R. Eddy. RIO: analyzing proteomes by automated phylogenomics using resampled inference of orthologs. *BMC Bioinformatics*, 3:14, 2002.