Deconvolution of PPI Networks: Approximation Algorithms and Optimization Techniques

Dong Hyun Kim

School of Computer Science
McGill University
Montréal

May 2012

A thesis submitted to McGill University in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Contents

Abstract

Understanding the organization of protein-protein interactions (PPIs) as a complex network is one of the main pursuits in proteomics today. With the help of high-throughput experimental techniques, a large amount of PPI data has become available, providing us a rough picture of how proteins interact in biological systems. One of the leading technologies for identifying protein interactions is affinity-purification followed by mass spectrometry (AP-MS). While the AP-MS method provides the ability to detect protein interactions at biologically reasonable expression levels, this technique still suffers from poor accuracy as well as lack of sound approach to interpret the obtained interaction data.

In this thesis, we look for sources of systematic errors and limitations of the data from AP-MS experiments, and propose several approaches for improvements. In particular, we identify various problems that arise within the experimental pipeline, and propose combinatorial algorithms developed using the theory of approximation algorithms and discrete mathematics. The first part of the thesis deals with *quantification of proteins* from MS-based experiments. Existing approaches for protein quantification often use each detected peptide as an indicator for the originating protein. These approaches ignore peptides that belong to more than one protein family. We attack this problem of protein quantification by taking these *shared peptides* into consideration, and propose a framework for estimating protein abundance via linear programming.

In AP-MS data, the identified protein interactions contain a large number of indirect interactions as artifacts from a series of simultaneous physical interactions. The second part of this thesis studies the problem of distinguishing *direct physical interactions* from indirect interactions. To do this, we first propose a probabilistic graph model for the PPI data, and design a combinatorial algorithm suited for graphs with underlying structure that is evident in PPI networks.

While the traditional model for PPI networks is a binary graph representing pairwise interactions, a large number of interactions involve more than two interaction partners. Such collections of proteins interacting concertedly are known as *protein complexes*, and various approaches have been proposed to identify the complexes in the network. When these complexes are overlapping, however, the existing complex detection methods often fail to identify the constituent complexes. Taking one step further on this line of research, the last part of this thesis discusses the problem of modelling PPI networks as hypergraphs by studying the *clique cover* problem on sparse networks.

For each problem discussed throughout the thesis, we obtain either an exact algorithm, an algorithm with provably good guarantee on the output quality, or a heuristic with efficient run-

ning time. Furthermore, each of the proposed algorithms is empirically tested against biological data as well as simulated data, in order to validate both computational efficiency and biological soundness.

Résumé

Comprendre l'organisation des interactions protéine-protéine (IPP) en tant que réseau complexe est un des plus grands problèmes de la protéomique moderne. Avec l'aide de techniques expérimentales à haut débit, une grand quantité de données de IPP est devenue disponible, nous procurant ainsi une image approximative du fonctionnement des interactions entre protéines dans des systèmes biologiques. Une des technologies de fine pointe pour identifier des interactions protéiques est la purification d'affinité suivie de la spectrométrie de masse (PA-SM). Même si la méthode de PA-SM nous permet de détecter des interactions de protéines à des niveaux d'expression raisonnables biologiquement, cette technique souffre encore d'une précision déficiente et d'un manque d'une approche saine pour interpréter les résultats d'interactions obtenus par celle-ci.

Dans cette thèse, nous cherchons des sources d'erreurs systématiques et des limites des données provenant d'expériences PA-SM et nous proposons plusieurs approches amenant des améliorations. En particulier, nous identifions divers problèmes présents dans la procédure expérimentale et proposons des algorithmes combinatoires développés en utilisant la théorie des algorithmes d'approximation et des mathématiques discrètes. La première partie de la thèse étudie la quantification des protéines provenant d'une expérience basée sur la spectrométrie de masse. Les approches existantes pour la quantification de protéines utilisent souvent chaque peptide détecté comme un indicateur de la protéine d'origine. Ces approches ignorent les peptides qui appartiennent à plus d'une protéines. Nous attaquons ce problème de quantification de protéines en prenant ces peptides partagés en considération et proposons un cadre de travail pour estimer l'abondance des protéines via la programmation linéaire.

Les interactions de protéines identifiées dans les données de PA-SM contiennent un grand nombre d'interactions indirectes qui sont des artéfacts d'une série d'interactions physiques simultanées. La seconde partie de cette thèse étudie le problème de distinction des interactions physiques directes de celles qui sont indirectes. Pour ce faire, nous proposons premièrement un modèle d'un graphe probabiliste pour les données de IPPs et concevons un algorithme combinatoire adapté aux graphes ayant des propriétés observées dans de vrais réseaux de IPPs.

Malgré le fait que le modèle traditionnel des réseaux de IPPs est un graphe binaire représentant les interactions par paires, un grand nombre d'interactions impliquent plus de deux partenaires d'interaction. De tels groupes de protéines interagissant de concert se nomment des complexes de protéines. Plusieurs approches ont déjà été proposées afin d'identifier les complexes dans un réseau. Cependant, lorsque ces complexes se chevauchent, les méthodes existantes échouent. La dernière partie de cette thèse discute du problème de modélisation des réseaux de IPPs comme des hypergraphes en étudiant le problème de couverture par cliques sur des réseaux

clairsemés.

Pour chaque problème discuté au cours de cette thèse, nous obtenons un algorithme exact, un algorithme avec de bonnes garanties prouvables sur la qualité de la sortie, ou une heuristique avec un temps d'exécution efficient. De plus, chaque algorithme proposé est testé empiriquement avec des données biologiques et simulées dans le but de valider l'efficience computationnelle et la signification biologique.

Declaration

This thesis contains no material which has been accepted in whole, or in part, for any other degree or diploma. Except for results whose authors are mentioned, materials presented in Chapters 3, 4, and 5 of this thesis is an original contribution to knowledge.

In Section 3.4 of Chapter 3, the biological data for experimental studies was generously provided by Benoit Coulombe's lab at Institut de recherches cliniques de Montréal (IRCM).

In Section 4.3.1 of Chapter 4, the work presented in Theorems 4.1 and 4.3 are done jointly with Ashish Sabharwal and my thesis advisors.

All other parts were done independently under the supervision of my thesis advisors, Adrian Vetta and Mathieu Blanchette.

Acknowledgements

During my first visit to Montréal in February 2005, my old advisor Sue Whitesides told me a number of reasons why McGill is one of the best places to study computer science and discrete maths. Seven years later, I can acknowledge every one of those reasons, and possibly double the list with my own.

First and foremost, I express my deepest gratitude to my thesis advisors, Mathieu Blanchette and Adrian Vetta, for their tireless efforts to guide my research, and making my life as a graduate student ever so enjoyable. Mathieu has introduced me to the field of protein-protein interactions, invited me to his Barbados workshop on the subject, was always ready to suggest the next steps in research, and taught me all about what makes good science. Adrian has introduced me to the field of approximation algorithms, spent countless hours discussing various lines of attacks to problems, and has provided me with seemingly infinite amount of resources in discrete mathematics. Without their guidance, this thesis, or my own scientific becoming, would have been impossible.

I thank my collaborators and colleagues: Ashish Sabharwal's visit to Montreal resulted in the first part of work presented in Chapter 4; Mathieu L.-A., Javier, Mathieu R. and Javad taught me a thing or two about proteomics and machine learning; human PPI data used in Chapter 3 was generously provided by Benoit Coulombe's lab at Institut de recherches cliniques de Montréal (IRCM).

I acknowledge NSERC, CIHR, Walter C. Sumner Fellowship, and my advisors' generous funding for financial support.

Special thanks to the "family" for all the laughters over pints: js, mk, sh, jm, sc, ij, sy, sj, jy, the list goes on. And finally, to my parents, for their love and patience over the years. Thank you.

List of Figures

Chapter 1

Introduction

Upon completion of the Human Genome Project[1], one of the compelling topics in the biological sciences is the large-scale study of proteins encoded by the genes – a field referred to as *proteomics*. Indeed, it is the proteins that execute the functions programmed in the genes, and analyzing the proteome will thus lead us to a better understanding of how cellular processes work in living organisms.

While the genome is considered as a long sequence of DNA, proteome is often regarded as a much more complex system. Because cellular processes are caused by proteins interacting concertedly within the cell, the proteins in the proteome form a large network called *protein-protein interaction* (PPI) network, sometimes referred to as the *interactome*. As a result, both structural and functional analyses of the proteome often require detailed examinations of the PPI network. Efforts to construct the PPI network have begun by developing various experimental techniques to identify the interactions. Using these technologies, *Saccharomyees cerevisiae* (the yeast) was shown to contain approximately 6000 proteins, and over 78,000 interactions have now been identified. To give an example, Figure 1.1 depicts a small portion of the yeast PPI network, using only

[1] An international scientific research program whose goal was to sequence the DNA, and identify the genes of human genome; began in 1990, completed in 2003. (`http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml`)

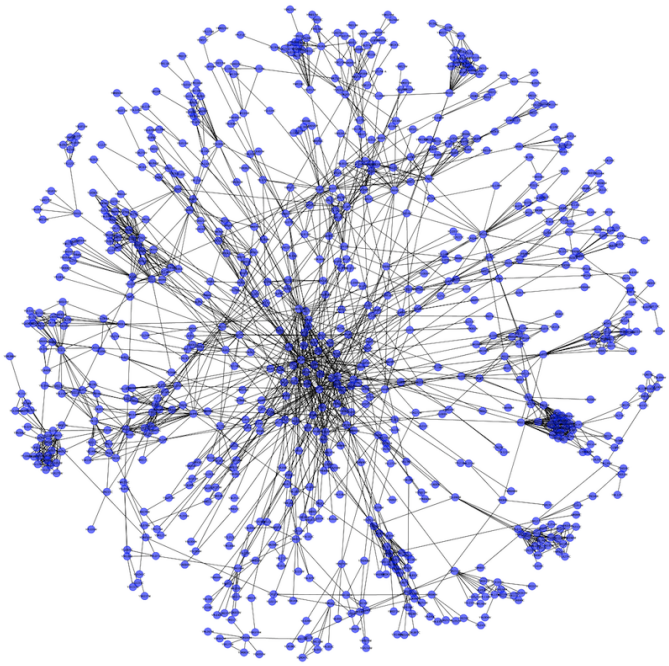


Figure 1.1: The yeast PPI network constructed using Cytoscape software (`http://www.cytoscape.org`) with high confidence PPI data from Krogan et al. [93]

the highest confidence interaction data from Krogan et al. [93]. Humans, more complex organisms as we are, are expected to have approximately 25,000 proteins sharing as many as 650,000 interactions [130]. Such sheer volume of data makes it intractable to analyze manually, and thus computational analyses have become essential to deeper understanding of the proteome.

More recently, it has been revealed that many of these interactions may occur only when three or more specified proteins are all present at the site of an interaction. Such a collection of proteins interacting concertedly is called a *protein complex*, and this theory around protein complexes opened a new set of computational problems, such as the identification and functional annotation of protein complexes. To make things even more interesting, many of these interactions depend on the phase of the cell development cycle during which the interaction takes place, as well as the localization within the cell [80, 97]. Consequently, such temporal and spatial dependence makes the structure of PPI networks much more dynamic than originally expected.

Due to the natural structure of the proteome as a network, graph theory has played a crucial role in computational analyses of PPI networks. As shall be demonstrated later, however, it is often difficult to formulate an optimization problem that correctly captures all the properties of PPI networks, and even when a clean optimization problem is formulated, the problem often turns out to be computationally intractable. In many of these cases, heuristic algorithms targeting local optima have been popular, and very few algorithms with provable guarantee of output quality have been proposed. As such, clean mathematical models and sound algorithms remain in high demand in the field of protein-protein interactions.

This thesis and its contributions can be divided into three parts; each part addresses a particular challenge related to the analysis of PPI networks, and provides a novel approach to reduce errors and noise present in the data. The principal results of this thesis will be outlined in Section 1.4. Prior to that, however, we first give an introduction to the field of protein-protein interactions from the perspective of bioinformatics and discrete mathematics. This introductory chapter consists of three parts: in Section 1.1, we give a brief survey on various experimental techniques for identifying protein interactions. Section 1.2 then discusses computational approaches to analyze the datasets produced by those experimental techniques; one of the leading technologies that is prevalent in the literature is *affinity purification followed by mass spectrometry* (AP-MS). In Section 1.3, we review studies on PPI networks produced by AP-MS experiments, and point out the limitations of existing computational approaches, which will then lead us to the topics of this thesis, namely: (1) protein quantification; (2) predicting direct interaction network; and (3) hypergraph modelling of PPI networks.

1.1 Experimental Techniques

In this section, we introduce several medium to high throughput experimental methods that have been proposed to identify protein interactions, and discuss the nature

and the quality of the data obtained by each technique. Each technique makes use of different mechanisms to identify the interactions, and bears its own advantages and disadvantages. For example, some techniques identify direct, physical interactions while others are designed to find indirect, functional relationships between proteins. Therefore, understanding the intrinsic differences between the datasets from various experimental techniques will allow us to realistically model the given data, and devise highly customized algorithms for the formulated problems.

1.1.1 Yeast-Two Hybrid (Y2H) Method.

Yeast two hybrid (Y2H) is an *in vivo*[2] technique to detect PPIs by testing physical interactions between two proteins [49, 75, 137]. It was discovered that transcription factors found in eukaryotic organisms have two distinct domains: (1) a binding domain (BD), and (2) an activating domain (AD) [49]. The binding domain is a module responsible for binding to a promoter DNA sequence while the activating domain activates the transcription. The transcription is inactivated while the two domains are far from each other, but it can be restored when a binding domain is in close proximity with an activating domain.

A Y2H experiment starts by fusing the binding domain into a protein X (known as the bait), and the activating domain into a protein Y (known as the prey). If the bait X and the prey Y interact physically, the activating domain is in close proximity to the binding domain. The binding domain then binds to upstream activating sequence (UAS) of a promoter, and thus activates the transcription of the reporter gene[3]. Figure 1.2 shows a schematic representation of a Y2H system. To do this experiment in a high throughput manner, either a matrix of prey clones or a library of random cDNA fragments are

[2] An experiment is said to be *in vivo* (Latin for "within the living") when the subject is a living organism as opposed to *in vitro*, a controlled environment.

[3] In order to determine whether a gene is expressed, a reporter gene (which is easily identifiable when expressed) is attached to a regulatory sequence of the gene of interest. Commonly used reporter genes often induce visually identifiable characteristics, e.g. LacZ, GFP, etc.

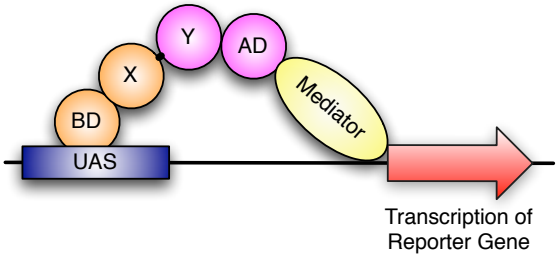


Figure 1.2: A Y2H experiment. Protein *X* (the bait) is fused with a BD that binds to the upstream activating sequence (UAS) of a promoter. The bait interacts with Protein *Y* (the prey) that is fused with an AD, activating the transcription of a reporter gene via the mediator complex.

used [11, 50, 142]: In the matrix approach, a matrix of prey clones is mated with baits, and the interacting prey proteins are identified by the expression of a reporter gene and the position on the matrix. In the library approach, each bait is screened against a library of random cDNA fragments, and the interaction partners are identified by DNA sequencing.

One advantage of a Y2H experiment is that it allows us to identify physical (rather than functional) associations between interaction partners. Furthermore, being an *in vivo* high throughput technique, Y2H produces large scale PPI data that potentially includes even the weaker, transient interactions. As such, the Y2H PPI data provide us a comprehensive snapshot of the proteome. However, because each interaction must occur between the proteins fused with the bait-prey pair, the interactions identified by Y2H experiments are restricted to *binary* interactions. There are other limitations of Y2H systems: proteins that initiate transcription on their own cannot be targeted since they would result in expression of the reporter gene regardless of the presence of bait proteins; the protein fusion may cause changes in the structure of the protein, and the use of such bait proteins may result in incorrect interaction partners. Such a non-native environment causes high false positive ratios for Y2H experiments. To rectify these issues for Y2H experiments, many *in silico*[4] methods have been proposed to post-process Y2H experimental data (see Section 1.2.2). With the help of these computational analy-

[4] Computational analyses are often called *in silico* methods as opposed to *in vivo* or *in vitro*.

sis tools, Y2H experiments have been the most widely used approach to obtain physical interaction data.

Protein-fragment Complementation Assay (PCA). Another similar technique has been proposed by Michnick [54, 103, 134]: Protein-fragment Complementation Assay (PCA) is a technique in which fragments of a reporter protein can be separately expressed with proteins (bait and prey) that are to interact with each other. In this method, the targeted proteins (bait and prey) are fused with incomplete fragments of a third protein (reporter), and are separately expressed *in vivo*. The interaction between the bait and the prey brings the fragments of the reporter in close proximity, allowing the reporter to reform, and then to be identified. While similar to Y2H experiments, PCA experiments can test protein interactions at various subcellular compartments within a pathway of interest in a quantitative manner, making it desirable for drug discovery purposes. Typically, PCA detects many interactions for membrane proteins whereas Y2H identifies many interactions for nuclear proteins [78].

1.1.2 Affinity Purification / Mass Spectrometry (AP-MS)

Affinity Purification followed by Mass Spectrometry (AP-MS) [36, 58, 93, 119], sometimes referred to as *tandem affinity purification* (TAP), is a technique that allows us to determine co-complex membership of protein interactions. This method is a two step process consisting of the purification of protein complexes, and the identification of the interaction partners via mass spectrometry.

Affinity purification. A TAP tag consists of a calmodulin binding peptide (CBP) followed by a tobacco etch virus (TEV) protease cleavage site and the IgG binding domains of *Staphylococus* protein A [118, 119]. The open reading frame of the target protein

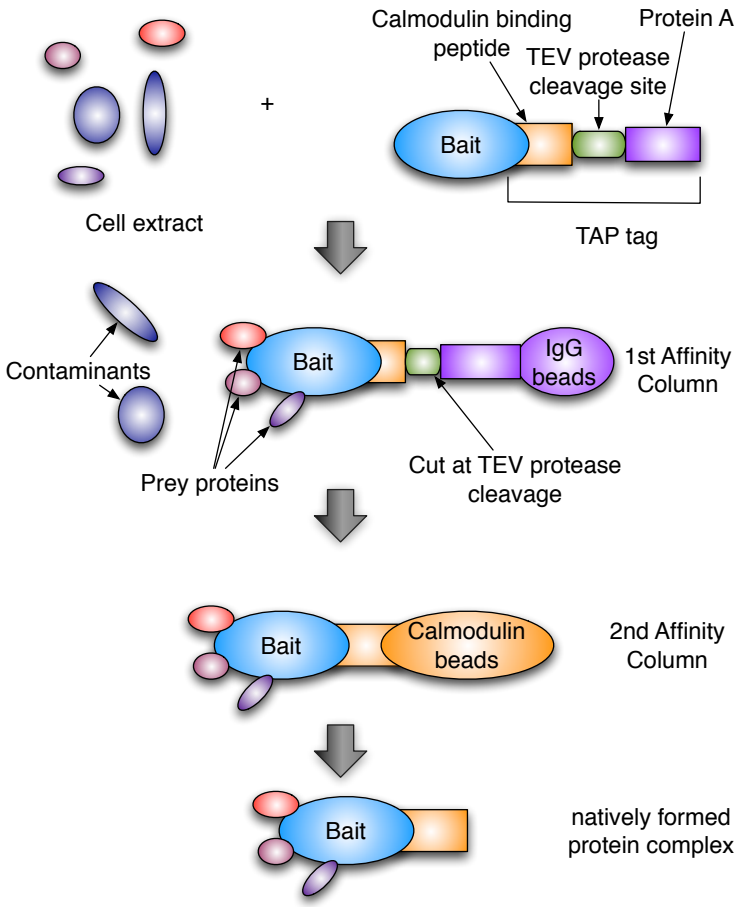


Figure 1.3: Pipeline of TAP experiments. At the first affinity column, the bait protein is cut at the TEV protease cleavage site after washing. At the second affinity column, the bait-prey pairs are pulled down by Calmodulin beads, and the resulting eluate form native complexes which can then be identified using gel electrophoresis and mass spectrometry.

(known as the *bait*) is fused with the TAP tag at the end, and is expressed *in vivo* within the cells of the host organism (e.g. yeast or human). Once the AP-tagged bait protein is expressed, it can interact with other proteins (known as *preys*), and form protein complexes. These protein complexes go through two purification processes: Initially, the protein A is binding tightly to an IgG matrix. After washing out the contaminants, the link between the protein A and IgG matrix is released by the protease. The eluate of this step is then incubated with calmodulin-coated beads, and after washing for the second time, the targeted protein complexes are released. See Figure 1.3 for a schematic representation of the AP process.

Mass spectrometry. The result of the purification processes is the target protein complexes formed by the interaction partners of the bait protein. These protein complexes go through gel electrophoresis, resulting in component peptide fragments that can be identified using mass spectrometers.

Mass spectrometers then read in the peptide fragments, and produce ions with charges based on their masses. Polypeptide sequences are then identified using their mass-to-charge ratios. Various methods have been proposed to convert the peptide molecules into ions in the gas phase using electrospray ionization (ESI) [144] and matrix assisted laser desorption ionionzation (MALDI) [85, 116]. Furthermore, there are several algorithms to analyze the mass spectra produced by MS and either identify the peptides present, or even quantify the peptide abundance [59, 115, 135].

Due to its ability to perform at biologically reasonable expression levels in human cells, as well as the ability to detect protein complexes with fewer false positives [36], AP-MS approaches have become increasingly popular. One caveat of this approach is, however, that a significant number of the co-purified prey proteins are in fact *indirect* interaction partners of the bait protein. This is an artifact of *chains* of binary interactions that occur simultaneously, and AP-MS cannot distinguish such indirect interactions from direct physical interactions. For example, Figure 1.4 shows how the same complex can be perceived differently by different experimental techniques. In Chapter 4 of this thesis, we discuss the problem of identifying direct physical interactions from AP-MS data.

Added to the difficulty of interpretation from indirect interactions, AP-MS carries a high chance of capturing contaminants at the AP level: gel contamination, nonspecific binding to TAP columns, etc. To reduce the false positives from these contaminants, several computational approaches have been proposed [124, 128]. Some of these approaches make use of the network topology to assign a purification score to each hit [35]. On the other hand, Lavallée-Adam et al. [95] recently provided a Bayesian model specifically for contaminants to directly identify false positive interactions.

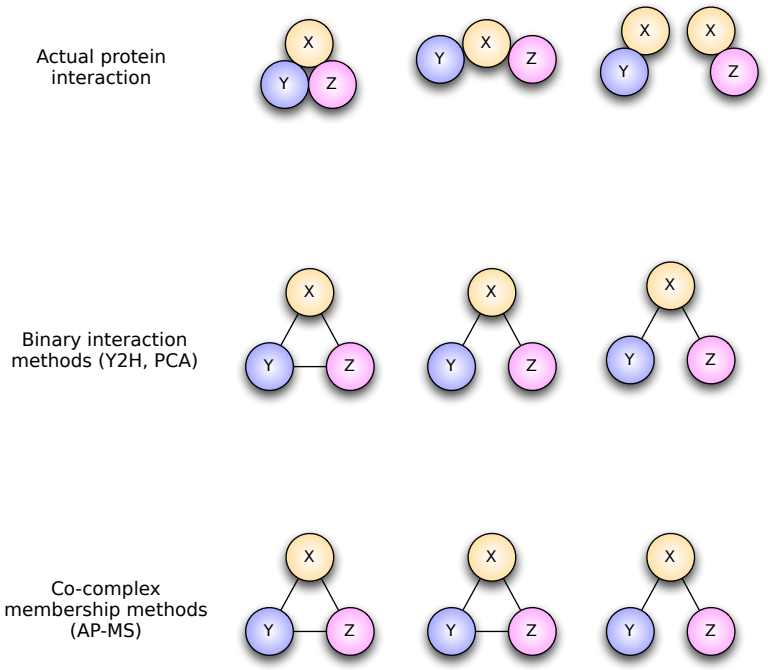


Figure 1.4: Different types of protein interactions among three proteins, and how they are viewed in different experimental techniques (assuming all three proteins have been tagged as a bait). Note that binary testing methods (e.g. Y2H and PCA) cannot distinguish the last two types of interactions, while co-complex membership testing approaches (e.g. AP-MS) identify the first two types of interactions as the same topology.

1.1.3 Indirect Approaches

Aside from the methods described above, there are indirect approaches to associate proteins with functional interactions.

Correlated mRNA expression (Synexpression). Genes can be partitioned into distinct groups depending on the mRNA levels measured under a variety of different cellular conditions. Then, the genes in the same partition are enriched to encode proteins that may interact with each other [44]. Though this technique gives a broad coverage[5], the accuracy of the data is poor as it is difficult to infer direct physical (or even functional) interactions solely from gene expression levels.

[5] *Coverage* refers to (correctly identified interactions) / (total number of true interactions). *Accuracy*, on the other hand, refers to (correctly identified interactions)/(number of identified interactions)

Method	Interaction Type
Y2H	physical, binary
PCA	physical, binary
AP-MS	physical, complex
Synexpression	functional
Genetic interactions	functional
Genome analysis	functional

Table 1.1: Different techniques to identify protein-protein interactions, and types of interactions captured by each technique.

Genetic interactions. If two nonessential genes cause lethality when simultaneously knocked out, they are often functionally associated, and thus their encoded proteins may interact [12, 148]. This technique allows pairwise detection of functionally associated interactions.

Genome analysis. Some protein interactions can be detected via computational methods [112]. For example, (1) interacting proteins often show similar phylogenetic profiles,[6] and (2) seemingly unrelated genes are sometimes fused together into one polypeptide chain. Though fast and inexpensive, these computational methods rely heavily on the existing data such as phylogenetic trees, and orthology[7] between proteins which are prone to biases towards well-studied proteins.

Observe that different techniques have different goals in detecting interactions. While Y2H, PCA, and AP-MS methods are designed to discover physical bindings between proteins, the others seek to predict functional associations between entities such as transcriptional regulators and their associated pathways. The different coverage and accuracy of the data causes each technique to produce a unique distribution of interactions [141]. Consequently, when comparing or merging PPI data from different experimental techniques, one must carefully take these intrinsic differences into account. Table 1.1 summarizes different aspects of the methods discussed in this section.

[6] Across a number of species, each gene can be checked for presence, resulting in a binary vector called a *phylogenetic profile*. If two proteins interact in the same biological pathway, the corresponding genes would show similar phylogenetic profiles.

[7] Homologous sequences are *orthologous* if they were separated by a speciation event, as opposed to *paralogous* if caused by a gene duplication event.

1.2 Computational Analyses of PPI Data

The traditional way to model PPI networks is to construct a combinatorial graph where each node represents a protein, and two nodes are joined by an edge if and only if there is an interaction between the corresponding proteins. While this classical model captures the most basic information about the interactome, it does not accommodate various sources of errors that are common in many experimental datasets.

On the other hand, protein complexes are hidden away within the constructed (binary) PPI network. Discovering these complexes would allow us to model the interactome as a more complex system. Therefore, further post-experimental analyses are imperative to obtain a proper view of PPI networks. In this section, we review computational methods for post-processing experimental PPI data, and we discuss how such approaches can help characterize unknown properties of the interactome.

1.2.1 Graph Models for PPI Networks

One approach to the analysis of PPI networks is looking at topological properties of the networks. While studying the topological structure of PPI networks may not immediately lead us to biological discoveries, understanding the characteristics of the networks is typically a crucial step toward modelling real world networks.

Local Structure Models

Network motifs. *Network motifs* are patterns of interconnections (induced subgraphs) that occur in a given biological network much more often than they would in random networks. Shen-Orr et al. [127] have shown that the transcriptional regulatory network of *Escherichia coli* contains three highly frequent motifs, each with a distinct function in gene expression. This suggests that identifying different motifs in a PPI network would help us describe different functional protein groups within the network and, further-

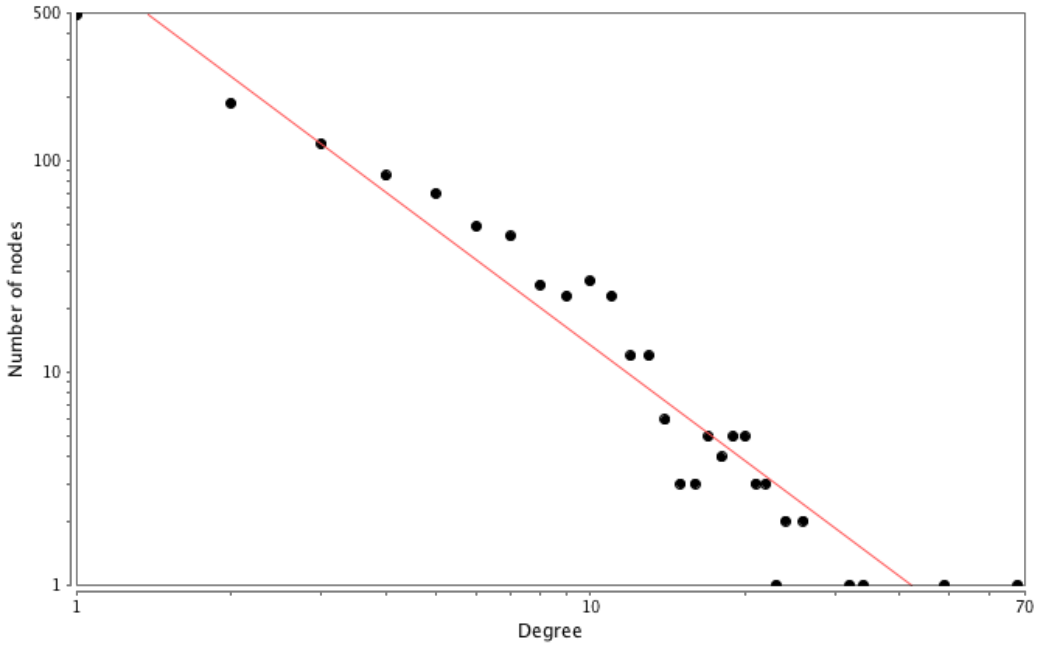


Figure 1.5: Degree distribution of yeast PPI network of 1210 proteins with 2357 interactions [93]. On a log−log plot, the distribution fits a power law distribution with $y = ax^{-\gamma}$ where $a = 874.87$ and $\gamma = 1.809$.

more, determine the function of uncharacterized proteins.

Graphlets. A similar notion of local structure called *graphlets* has been proposed by Przulj et al. [117]. Graphlets are all possible simple graphs over $3-5$ vertices (up to isomorphism). In order to analyze the global structure of PPI networks, Przulj et al. looked at the frequency distribution of each graphlet appearing in the PPI networks, and compared it against other global model networks such as Erdös-Renyi random graphs, scale-free networks, and geometric random networks (see below). While network motifs for a given network capture the most significant local structures within the network, this bottom-up approach of graphlet distribution provides a metric to compare different networks. More specifically, the distribution of each graphlet measured by the relative frequency of graphlets allows us to analyze networks of different sizes. This is particularly useful when PPI networks are compared to various graph models (see next section) whose network sizes may vary.

Global Structure Models

Scale-free networks. In many real world networks, including PPI networks, the degree distribution of the nodes follows the power law $P(k) \approx k^{-\gamma}$; that is, there are many more nodes with a small number of neighbours than nodes with a very large number of neighbours. Networks that exhibit such a degree distribution are called *scale-free* [10], and this property has profound effects on real world networks. For example, scale-free networks are resistant to random failures on the nodes due to the small number of high-degree hubs – there are many more low-degree nodes which are equally likely to fail. This robustness can also be observed in PPI networks, and the connectivity of a node can be correlated to the essentiality of the corresponding protein. For example, studies on the yeast PPI network [79] have shown that:

1. 93% of the yeast proteins have degree at most 5, but only 21% of these caused lethality when deleted;
2. only 0.7% of the proteins had degree at least 15, but 62% of these are essential.

These results suggest that non-essential proteins, whose disruption is non-lethal, tend to have lower degree than essential proteins. Moreover, proteins in the same functional group tend to show similar number of interacting partners. Such structure-function relationships can be exploited for the prediction of protein interactions or complexes that are yet unknown, as we discussed in Section 1.1.3.

Geometric random networks. A geometric random network [113] is a graph model where the nodes correspond to randomly distributed points in a metric space, and nearby nodes are joined by an edge. Przulj et al. [117] used the graphlet distributions to characterize PPI networks as a geometric random network. In particular, upon comparing the yeast and fruitfly PPI networks against various random graph models (including scale-free graphs), the graphlet distributions of these PPI networks were closer to that of geometric random networks than any other model. This result suggests that the geomet-

ric random network may be a better model for PPI networks than the widely accepted scale-free networks.

1.2.2 Identification of Protein-Protein Interactions

The PPI data from high throughput experiments such as Y2H, PCA or AP-MS contain a significant amount of noise, and such high false positive/negative ratios deteriorate the overall confidence level of identified interactions. These high throughput methods have all been used to perform a large-scale study of yeast proteome [58, 70, 93, 134, 149], but comparative assessments of those datasets continuously reveal that a significant portion of identified interactions in one study is unique to that dataset and not observed in others [78, 141]. Therefore, *in silico* approaches to improve the confidence level of PPI data are an attractive topic of research.

Topology-based PPI detection. Inevitably, the PPI data from high throughput experiments is used as the initial input. Since the global view of the currently known interactome may be heavily biased due to incomplete experiments, several authors often resort to examining the local topological structure. For example, Saito et al. [122, 123] developed a measure called interaction generalities (IG1, IG2) which considers the restricted neighbourhood for each protein in the network. The main idea behind this measure is that if a protein interacts with many interaction partners but these interaction partners exhibit no further interactions among themselves then these interactions are more likely to be false positives. This idea works particularly well for PPI data from Y2H experiments, since some "sticky" proteins in the Y2H assays have a tendency to turn on the positive signals by themselves, irrespective of their interaction partners.

More recently, two other measures have been proposed by Chen et al. [27, 26], and Pei and Zhang [111]. The method of Chen et al., called Interaction Reliability by Alternative Path (IRAP) [27], considers alternative paths between two interacting partners. If a pair of proteins exhibit a direct interaction, it is likely that there are alternative se-

quences of interactions that join the two proteins. Hence, IRAP looks for the shortest (measured by distance) such alternative path for each pair of proteins, and the final confidence level is determined by the strength of the discovered alternative path.

On the other hand, Pei and Zhang [111] have shown that, by considering all alternative paths of different lengths, a better result can be achieved. For each pair of vertices, they consider all paths of length $k > 1$. Then, the proposed measure called PathRatio is computed as a weighted sum of all the paths between two vertices. Since enumerating all paths between two vertices is computationally hard, a rough estimate is obtained by only computing for small values of k. Using the PPI data from various sources, the PathRatio method achieved better results in functional homogeneity, localization homogeneity, and gene expression distance when compared against a simulation using IRAP. On the other hand, the iterative hill-climbing version of IRAP, called IRAP* [26], showed similar performance to PathRatio.

While these topology-based methods attempt to assign confidence levels to interactions, the resulting PPI networks still contain high ratios of false positives and false negatives. This may be due to the fact that the models used in these methods do not truly reflect the nature of the experimental PPI data – for example, the input PPI data may contain a large number of indirect interactions from AP-MS experiments, but no known algorithm addresses this issue explicitly (the algorithms discussed in this section are evaluated only using the PPI data from Y2H experiments).

Phylogeny-based PPI detection. The poor accuracy and coverage of topology-based methods may also be due to the lack of information contained in the PPI data itself. To address this, several approaches have been proposed using phylogenetic information. For example, a protein interaction in one species may be inferred from Rosetta Stone proteins (two genes fused into a single one) in another organism (Gene Fusion Method [131]). Or, in the case of bacterial genomes, the conservation of the order of genes may be a good indication of interactions (the Gene Order Conservation Method [37]).

A potentially more powerful approach is using a set of reference organisms' genomes. A *phylogenetic profile* of a protein A is a vector whose i-th entry represents the presence or the absence of protein A in organism i. Under the assumption that physically interacting proteins coevolve, the interaction can be inferred from the similarity of two phylogenetic profiles (the Phylogenetic Profile Method [112]). Moreover, the phylogenetic trees for two proteins could also be used in comparative analyses. In this method, a multiple sequence alignment (MSA) is first obtained from a set of reference organisms, and the phylogenetic trees for protein A and B are constructed using the position of orthologs in the MSA. Then the similarity score can be computed by comparing the distance matrices of the two phylogenetic trees. For a more comprehensive exposition of these methods see Jothi and Przytycka [82].

Protein-protein docking. Another approach for detecting protein interactions is done by structural analysis of protein molecules: namely, protein-protein docking. Given a pair, or a small set of proteins, accurate and efficient protein docking algorithms provide a useful tool for understanding both the tendency for protein interactions and the structure of protein complexes. Various docking algorithms have been proposed, and often work in two phases: (1) A population of candidate conformations are generated; and (2) each of the candidates is scored in order to find the most likely structural conformations (see, for example, Azé et al. [6] and Choi [31]). To verify the correctness of these methods, they are often compared against a database of NMR and X-ray structures [77].

1.2.3 Identification of Protein Complexes

While the PPI networks represent binary interactions between two proteins, protein complexes with three or more interacting partners are difficult to identify immediately from these networks. However, the protein complexes are often highly connected subgraphs in PPI networks, and thus graph clustering algorithms are often used to detect these complexes.

Molecular Complex Detection (MCODE) algorithm. Bader and Hogue [7] proposed the three-stage algorithm, MCODE, to identify protein complexes. First, the vertices are given weights according to a local density measure, called the *core-clustering coefficient*. Let $N[v]$ denote the subgraph induced by a vertex v and its neighbours. Then, a *k-core* of v is a subgraph of $N[v]$, induced by vertices with degree at least k. The *highest k-core* refers to a nonempty k-core with the highest k, and the core-clustering coefficient of a vertex v is then calculated as the edge density of the highest k-core of v.

After each vertex is assigned a weight, the second stage of the algorithm recursively starts to expand the clusters. The vertex with the highest weight is initially set to form a cluster, and all of its neighbours are joined to the cluster if (i) the weight of the neighbour is above a given threshold; and (ii) the neighbour has not been previously explored. When no further expansion is possible, the next heaviest unexplored vertex is set to form a singleton cluster, and the process is repeated. Note that the clusters found so far are disjoint at this stage of the algorithm.

Finally, the third phase the the algorithm is a post-processing step, where clusters that are not sufficiently dense are discarded from the set of clusters. Furthermore, any unexplored vertices are joined to nearby clusters. At this stage, these unexplored vertices may be joined to multiple clusters, thereby creating overlapping clusters.

Markov Cluster algorithm. The Markov Cluster (MCL) algorithm, proposed by Van Dongen [139], simulates random walks on the graph by alternating two operations, *expansion* and *inflation*, to transform one set of probabilities into another. First, a weight matrix M of G is normalized into $\tilde{M}$ so that each column of $\tilde{M}$ sums to 1. Then each column of $\tilde{M}$ represents a stochastic process, where each entry $\tilde{M}(i,j)$ corresponds to the probability of going from vertex i to vertex j. The inflation Δ_r of $\tilde{M}$ for a power coefficient r is defined as:

$$\Delta_r(\tilde{M}(p,q)) = \frac{\tilde{M}(p,q)^r}{\sum_{i=1}^{n}(\tilde{M}(i,q))^r}$$

Hence, higher values of r would yield tighter clusters by favouring more probable walks on the graph.

On the other hand, the expansion operation creates longer walks on the graph by computing the e-th power of the associated matrix $\tilde{M}$, where e is the expansion parameter. This operation will try to expand the information flow on the network. The MCL algorithm runs the expansion and the inflation process as alternatively until $\tilde{M}$ stabilizes. When the algorithm reaches its equilibrium, the resulting matrix would correspond to a collection of star-like components, each of which is declared a protein complex.

Min-cut based algorithms. Though not applied directly to the protein complex detection problem, several approaches have been proposed to use minimum cut algorithms when finding clusters in biological networks (e.g. [69, 126]). These approaches first start with a similarity graph as an input, and repeatedly partition the set of vertices into clusters until a partition satisfies a particular stopping criterion. For example, in Hartuv et al. [69], the notion of *highly connected subgraph (HCS)* is used as a stopping criterion. A subgraph H is a HCS if the global minimum $s-t$ cut of H is at least $\frac{|V(H)|}{2}$. The initial PPI network is then recursively partitioned into two clusters by minimum $s-t$ cuts until each connected component is a HCS.

Simultaneous clustering algorithm. More recently, Narayanan et al. [106] proposed a framework, called *JointCluster*, that finds a clustering of multiple networks that integrates large-scale datasets including PPI data and gene expression data. Extending from a previous study on clustering a single graph by Kannan et al. [84], JointCluster finds sparse cuts to generate clusters that allow theoretical approximation guarantees on the quality of the detected clustering relative to the optimal clustering. The biological significance of the resulting clusters preserved across physical and coexpression networks is verified by known protein complexes in the literature or enrichment of functionally coherent pathways.

In a recent study by Brohee and van Helden [20], four clustering algorithms – MCL, Restricted Neighbourhood Search Clustering (RNSC) by King et al. [90], Super Paramagnetic Clustering (SPC) by Blatt et al. [15], and MCODE - were evaluated against PPI networks built from annotated MIPS database. In particular, to test for robustness of the algorithms against false positives and false negatives, edges were randomly added and removed to generate 41 altered graphs. After comparing against the hand-curated protein complexes from the MIPS database, the MCL algorithm clearly outperformed the other methods under most conditions, while RNSC and MCL show similar behaviours in some tests such as altered graphs with edge removal but no edge addition.

Note that these complex detection methods use similarity matrices that are often populated using the number of times each pairwise PPI is observed. However, the similarity between two proteins can be interpreted in various ways, e.g., (1) the confidence level of a direct interaction, (2) the strength of an interaction, or (3) the timing of an interaction to distinguish those that are transient. However, the existing complex detection methods do not reflect on how these similarity matrices are interpreted. In particular, if the PPI data contains many indirect interactions, clusters that highly overlap will be be difficult to separate using the existing PPI data. Furthermore, because the clustering algorithms tend to first create disjoint clusters, and then join the ones that are close to each other, they often do not distinguish overlapping protein complexes well. As such, methods to discover a collection of overlapping clusters are still in demand in order to properly model PPI networks as *hypergraphs*[8].

1.3 AP-MS based PPI Networks

In the previous section, we introduced the types of post-processing and analysis that need to be done computationally, and reviewed different approaches proposed in the literature. In general, these approaches do not take into account the experimental tech-

[8] A hypergraph is a set system where, in the context of PPI networks, each set corresponds to a protein complex.

nique from which the dataset originates, and such generic models often fail to capture the inherent nature of the produced data.

To give an example, the methods introduced in Sections 1.2.1 and 1.2.2 assume that the dataset contains only direct, binary interactions. This is a reasonable assumption when the dataset is from Y2H experiments, but when analyzing AP-MS data, indirect interactions must be filtered out prior to applying these methods. On the other hand, the complex detection methods discussed in Section 1.2.3 are designed to find individual clusters within the network, and as is the case for most clustering algorithms, finding overlapping complexes remains a difficult problem in general.

In this section, we take a brief look at how the analyses are carried out for PPI networks constructed from AP-MS experiments. In particular, we focus on two of the most recent large scale studies on AP-MS based PPI networks from a bioinformatics point of view.

1.3.1 Protein Complexes in Yeast

In Krogan et al. [93], the protein-protein interactions in the yeast were detected by the AP-MS method. In order to improve the accuracy as well as the coverage, two MS purifications were done independently. Furthermore, both interacting partners for each protein-protein interaction were tagged and purified in order to ensure greater data consistency and reproducibility. After 2357 successful purifications of proteins, 4087 yeast proteins were identified as preys with high confidence scores from the mass spectrometry.

After the interactome with confidence scores had been established, protein complexes were detected using the Markov Clustering algorithm (described in Section 1.2.3), where the expansion and inflation operators were appropriately chosen to optimize the overlap against the MIPS database. The Markov Clustering algorithm identified 547 disjoint protein complexes, half of which were not present in the MIPS database. Further-

more, new proteins were identified for most complexes that had been known previously.

For the purpose of the graph theoretic analysis of the protein complexes, a software plug-in (`http://genepro.ccb.sickkids.ca`) for the Cytoscape environment was written to model the interactome in two different ways: (1) the traditional PPI network model, and (2) the protein complex network model, where each protein complex is represented as a node, and two nodes are joined by an edge if and only if the corresponding complexes share common subunits. Using these two models, the authors were able to verify (or re-verify) several hypotheses and findings from the previous literature; for example, (1) proteins in the same complex should have similar function and co-localize to the same subcellular compartment, and (2) highly connected proteins within the network tend to be highly conserved, and conversely, highly conserved proteins tend to be more highly connected and central (measured by betweenness[9]) to the network.

On the other hand, as discussed in Section 1.2.3, the Markov Clustering algorithm does not necessarily separate two or more complexes that share subunits, and the authors point out that the identified clusters may contain multiple complexes. Identifying such overlapping complexes remains to be explored, and better suited complex detection algorithms will provide a finer view of the interactome.

1.3.2 Modularity in Yeast Protein Complexes

Gavin et al. [58] also studied the protein-protein interactions in the yeast via the AP-MS method, and successfully purified 1993 distinct proteins, for which 2760 interacting partners were identified via mass spectrometry. In order to measure the reproducibility, 138 purifications were done repeatedly, and 69% of the repeated purifications were common, giving rough estimates of the false-positive and false-negative ratios.

As acknowledged by the authors, the current graph clustering approaches for iden-

[9] Betweenness is a measure of graph centrality of a node. For a given node v, Betweeness(v) is typically defined as the number of shortest $s-t$ paths that pass through v (sometimes normalized by the total number of $s-t$ paths), where $s \neq v \neq t$.

tifying protein complexes are inappropriate for the PPI data from AP-MS experiments since they do not explicitly capture the nature of the purification output. To avoid this problem, the *socio-affinity index measure* was devised to quantify the propensity of proteins forming an interaction. This index measures the log-odds of the number of times two proteins are observed together, relative to what would be expected from their frequency in the data set. In particular, the socio-affinity index $A(i,j)$ is defined as

$$A(i,j) = S_{i,j|i=bait} + S_{i,j|j=bait} + M_{i,j},$$

where $S_{i,j|i=bait}$ measures the tendency for protein j to be pulled down when protein i is used as a bait (known as the *spoke model*), and $M_{i,j}$ measures the tendency for two proteins to co-purify when other proteins are tagged as a bait (known as the *matrix model*).

The socio-affinity model was the first attempt to quantify physical measurements of the interactions purely from the AP-MS data. These computed values can populate an $n \times n$ matrix representing a PPI network, and a graph clustering algorithm (not described) is then used in order to define protein complexes within this network. However, since each protein can belong to multiple protein complexes, a disjoint clustering from a single run of the clustering algorithm is unlikely to form the appropriate protein complexes. To avoid this pitfall, several runs of the clustering algorithm are carried out using different parameters.

Using this approach, 1784 different sets of complexes were generated, and were compared against hand-curated known complexes. Consequently, the sets of complexes that score highly ($> 70\%$) in coverage and accuracy show similar clusterings. Thus the top scoring set of complexes is merged with other high scoring sets of complexes to form variants of complex formulations called *complex isoforms*. Interestingly, these complex isoforms showed that the proteins within each complex can be partitioned into two types: (1) *core components*, which are the subunits of a complex that are present in most isoforms, and (2) *modules*, which are present in only some of the isoforms, but always together. Furthermore, each module (which can be regarded as a building block

of a complex) appeared to be present in multiple complexes, where the core components differ. The authors verified that this organization of complexes is due to biological phenomena by looking at the co-expression level during the cell cycle, as well as co-localization in the cell. Furthermore, the cores and modules that were detected within the same complex tend to belong to the same functional category, or otherwise, the core–module cross-talk between functional categories show many known connections such as between protein synthesis, transcription, and the cell cycle.

This modularity of protein complexes gives a new insight towards the structure of PPI networks in that each protein complex can be further partitioned into functional building blocks – cores and modules, and this structural property of protein complexes may be beneficial to designing new complex detection algorithms. On the other hand, while the socio-affinity index attempts to quantify the probability measure for direct protein interactions, the spoke model and the matrix model described above only consider 1- or 2-hop neighbours of each protein from the raw PPI data. From the bioinformatics standpoint, devising a better suited measure that fully models the stochastic nature of the AP-MS PPI data would certainly improve our understanding of the proteome.

1.4 Contributions and Thesis Outline

Due to its ability to test for protein co-complex memberships, AP-MS has been the method of choice for various recent studies in high-throughput proteomics. However, the AP-MS method has its own drawbacks that require systematic curation as well as reinterpretation of the experimental data. In this thesis, we (1) investigate three such limitations from a mathematical viewpoint, (2) formulate combinatorial optimization problems for each identified limitation, and (3) study algorithmic approaches with the appropriate applications in mind.

Chapter 2. We give an introduction to the mathematical tools that are used extensively

throughout this thesis. This includes techniques from approximation algorithms and combinatorial optimization as well as graph parameters that are useful for designing efficient algorithms for computationally intractable problems.

Chapter 3. Protein Quantification. When testing the existence and the abundance of proteins, mass spectrometry can only look at short fragments of proteins (known as *peptides*), and then identify which protein the peptide belongs to. However, there are a number of "ambiguous" peptides that belong to several different proteins, which makes the identification of constituent proteins a difficult task. We study the problem of quantifying the protein abundance despite the presence of these ambiguous peptides: given a set S of peptide sequences and a set W of protein sequences, together with the peptide abundance σ, find the protein abundance X for each protein. We formulate this problem into various mathematical programs for which we show their hardness, and devise approximation algorithms that can predict absolute abundance for all constituent proteins. This manuscript is to be submitted [89].

- **Ethan Kim**, Adrian Vetta, Mathieu Blanchette, "Protein quantification with shared peptides using multi cover algorithms", *in preparation*.

Chapter 4. Direct PPI network. Once the PPI data from AP-MS experiments are quantified, one needs to be sure that the identified interactions are accurate. However, as discussed in this chapter, a significant fraction of the detected interactions are artifacts of indirect interactions, i.e. detected via chains of simultaneous interactions. We study the problem of separating the direct interactions from indirect ones, in order to minimize the false positives in the AP-MS data. Here, we introduce a probabilistic graph model that captures the direct and indirect interactions within the AP-MS data. Then, under this model, the problem of distinguishing direct interactions from indirect ones is formulated, and we provide an algorithm that is highly customized for the AP-MS data. This work is published in [88].

- **Ethan Kim**, Ashish Sabharwal, Adrian Vetta, Mathieu Blanchette, "Predicting Direct Protein Interactions from Affinity Purification Mass Spectrometry Data", *Algorithms for Molecular Biology*, 5:34, (2010).

Chapter 5. Hypergraph modelling of PPI data. PPI networks are traditionally represented as combinatorial graphs where each interaction is assumed to occur between two proteins. Recent studies, however, revealed interactions involving several proteins working simultaneously. Such discoveries of protein complexes as building blocks of the proteome led to much interest in modelling PPI networks as hypergraphs. We thus study the problem of constructing hypergraphs from binary PPI data using algorithms for the (edge) clique cover problem. This work resulted in various efficient algorithms for both PPI networks and other restricted classes of graphs such as graphs with bounded parameters, and an approximation algorithm in the case of planar graphs, as published in [14].

- Mathieu Blanchette, **Ethan Kim**, Adrian Vetta, "Clique Cover on Sparse Networks", *The 9th SIAM Meeting on Algorithm Engineering & Experiments (ALENEX)* 93-102, Kyoto, Japan (2012)

Chapter 6. We conclude and give a list of open problems.

Chapter 2

Approximation Algorithms and Optimization Techniques

This chapter introduces topics from discrete mathematics that are preliminary to discussions throughout the thesis. In particular, we focus on algorithmic techniques that are frequently used to tackle problems in computational biology. As such, during this pedagogical exposition, we shall discuss problems with applications in computational biology in order to motivate the reader. We shall first define NP-hard optimization problems in Section 2.1. Then, Section 2.2 discusses graph parameters that often allow us to design efficient algorithms for computationally hard problems. Various techniques to design approximation algorithms are discussed in Section 2.3, including α-approximation algorithms, polynomial time approximation schemes, and LP-based methods. Finally we introduce genetic algorithms in Section 2.4.

2.1 NP-hard Optimization Problems

Combinatorial optimization problems ask to find an optimal object from a finite set of objects [125]. Typical problems include shortest path, minimum spanning tree, and

travelling salesman problem, and these problems can often be stated as: given A, find B of size k, where k is the size of the solution for which the problem tries to either minimize or maximize. If k is indeed the minimum (maximum, resp.) possible value admitting a solution, we say k is *optimum*.

From the perspective of computational complexity, we say that a problem Π is in P if there is an algorithm to solve Π efficiently.[1] On the other hand, if a problem Π is such that its solution can be *verified* efficiently, we say Π is in NP. By definition, this implies that problems in P belongs to NP. The hardest problems in NP deserves special attention: if *every* problem in NP can be reduced to a problem Π within polynomial time (without requiring that Π belongs to NP), we say that Π is NP-*hard*. Furthermore, if Π also belongs to NP itself, we say that Π is NP-*complete*. Therefore, if an NP-complete problem (the hardest problem in NP) can be solved efficiently within polynomial time, then every problem in NP can be solved in polynomial time. Therefore, NP-complete problems lie at the core of the class NP.

Consequently, it is of great interest in the theoretical computer science community to decide whether $\mathsf{NP} \setminus \mathsf{P} = \emptyset$, thereby $\mathsf{P} = \mathsf{NP}$. Since it is widely believed that this is not true, algorithm designers face difficulties in finding efficient solutions to these problems. Indeed, as is the case for problems that are discussed in this thesis, optimization problems in computational biology often turn out to be NP-hard. As a result, one must apply various techniques to either find *plausible* solutions to these problems, or restrict ourselves to *special cases* of the problem that allow efficient algorithms. We shall introduce some of these techniques in the following sections.

[1] An algorithm is said to be efficient if its running time can be bounded by a polynomial in the input length. We avoid more rigorous definitions for these classes of problems; precise definitions involving Turing machines and certificates can be found in various complexity theory texts, for example [4, 56].

2.2 Graph Parameters

Throughout this thesis, let $G = (V, E)$ be a finite graph with vertices V and edges E. Unless stated otherwise, we assume G is simple and undirected. The number of vertices and edges are $|V| = n$ and $|E| = m$, respectively.

A *graph parameter* Γ is a function that assigns a non-negative number to every graph G. For a class of graphs $\mathscr{G}$, $\Gamma(\mathscr{G})$ denotes the maximum $\Gamma(G)$ over all graphs $G \in \mathscr{G}$, and we say a graph class $\mathscr{G}$ has *bounded* Γ if $\Gamma(\mathscr{G}) \in O(1)$.

Graph parameters are useful when designing efficient algorithms with parameterized running time. In particular, a problem is *fixed-parameter tractable* (FPT) if it can be solved in $f(k) \cdot |I|^{O(1)}$ time, where f is a computable function depending on some parameter k, independent of the input size $|I|$.

A good example is *treewidth*, which measures how "tree-like" a given graph is. To define treewidth, we need the notion of *tree decompositions*.

Definition 2.1. [120] *A tree decomposition of a graph $G = (V, E)$ is a pair $(X = \{X_i | i \in N\}, T = (N, L))$ where each tree node $i \in N$ is associated with a set of vertices $X_i \subseteq V$, such that*

(1) $\bigcup_{i \in N} X_i = V$.

(2) For each $(v, w) \in E$, there is an $i \in N$ with $v, w \in X_i$.

(3) For each $v \in V$, the set of tree nodes $\{i \in N | v \in X_i\}$ induces a subtree of T.

For clarity, we shall refer to *(tree) nodes* of the tree decomposition T, and *vertices* of the original graph G. The width of a tree decomposition (X, T) is defined as $\max_{i \in N} |X_i| - 1$, and the *treewidth* of a graph G, denoted $\mathrm{tw}(G)$, is the minimum width over all tree decompositions of G. Following this definition, it is easy to see that all trees have treewidth of 1. See Figure 2.1 for an example of a tree decomposition of width 3. In general, it is NP-complete to determine the treewidth of a graph [3]. However, for a fixed k, graphs with

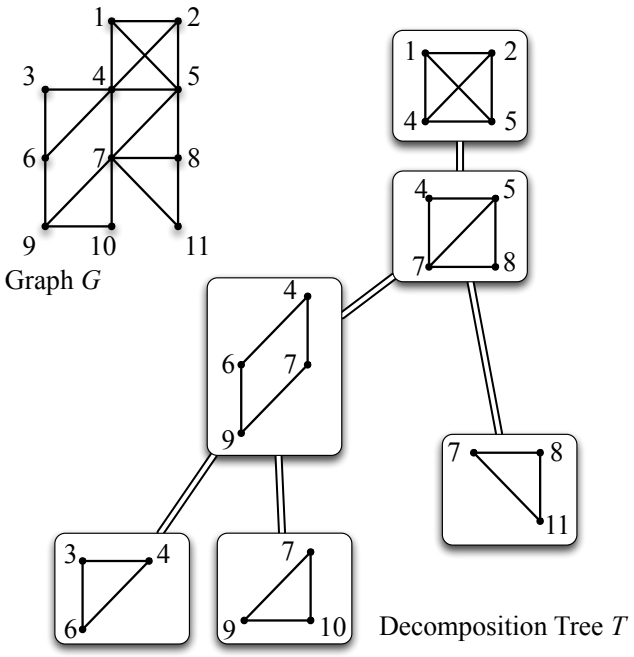


Figure 2.1: A graph G and its tree decomposition T of width 3.

treewidth k can be recognized, and a width k tree decomposition can be constructed in linear time [17].

A tree decomposition of a graph is especially useful for designing divide and conquer algorithms: given a decomposition tree T, one can pick an arbitrary tree node r as root of T. Then, for each tree node X, the problem can be solved for the subgraph induced by the vertices associated with X, and all its descendant nodes. Building up the solution from leaf nodes to r often provides a polynomial time algorithm (parameterized by the treewidth) if solutions to the subproblems can be combined using dynamic programming[2].

Another related parameter for graph sparsity is *branchwidth*.

Definition 2.2. [121] *A branch decomposition* (T,ϕ) *of a graph* G *is characterized by a ternary tree*[3] T*, and a bijection* ϕ *from the leaves of* T *to the edges of* G*.*

[2] Dynamic programming is a method for solving complex problems by subdividing the problem into simpler subproblems. This technique is useful when there are overlapping subproblems, so that one can solve the problem in a bottom-up fashion.

[3] A tree T is a ternary tree if every non-leaf node has degree 3.

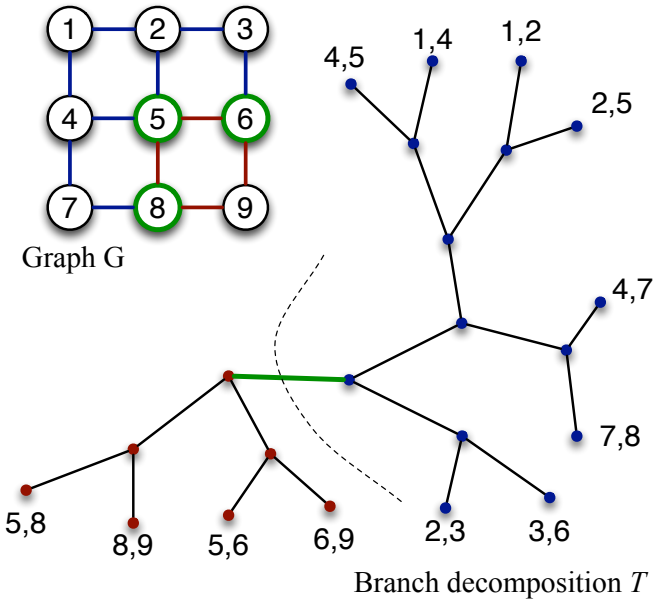


Figure 2.2: A graph G and its branch decomposition T. The e-separation shown on the decomposition tree T corresponds to a middle-set of $\{5,6,8\}$.

Let us denote e to be a tree edge in T. Removing e from T partitions into T_1 and T_2, and this partition induces a partition of edges in G, called an *e-separation*, associated with the leaves of T_1 and T_2. The set of vertices in G that are shared by both G_1 and G_2 is called the *middle-set* of e, and the *width* of this separation is the number of vertices in the middle-set.

Given a branch decomposition (T,ϕ), the width of this branch decomposition is the maximum width over all e-separations in T, and the *branchwidth* of G, denoted $\mathrm{bw}(G)$, is the minimum width over all branch decompositions. Figure 2.2 gives an example of a graph and a branch decomposition. It is well known that the branchwidth is closely related to the treewidth of a graph by the sandwich theorem [121]:

$$\mathrm{bw}(G) \leq \mathrm{tw}(G)+1 \leq \frac{3}{2}\mathrm{bw}(G).$$

These graph parameters often allow us to design efficient algorithms to solve otherwise NP-hard problems, and such problems come in different flavours: The input

graphs sometimes have bounded graph parameters due to the nature of the problem. In some other cases, different techniques can be used when the graph has unbounded parameters, so we can restrict ourselves to the bounded case. Sometimes, the graph decomposition gives rise to efficient algorithms to find near-optimal solutions.

All these techniques will be exploited extensively for modelling PPI networks as hypergraphs in Chapter 5. Indeed, graph decomposition finds applications in a number of different areas within computational biology. The *perfect phylogeny problem*, which will be discussed in the next section, is polynomially equivalent to the problem of triangulating coloured graphs, and it can be solved in polynomial time when the treewidth is bounded [16, 92]. Within the field of proteomics, several studies have shown that protein interaction networks often exhibit low treewidth (see, e.g. Yamaguchi et al. [147] and Cheng et al. [28]). This opens a gate for designing efficient algorithms for many problems on PPI networks. For example, Dost et al. [43] study the problem of querying the existence of a pathway Q within a base network G. While the problem of deciding if a graph is a subgraph of another is an NP-hard problem in general [56], they focus on the restricted class of graphs with bounded treewidth, and provide an efficient algorithm to query such pathways in a database of molecular networks.[4]

2.3 Approximation Algorithms

Many optimization problems in bioinformatics are NP-hard, and often researchers resort to heuristics for finding solutions that are close to optimal. A useful line of attack is the theory of approximation algorithms, which offers compelling techniques for solving NP-hard optimization problems with provable guarantees. For a minimization problem, an algorithm is an α-approximation algorithm if it can find in polynomial time a solution of value at most $\alpha \cdot \mathrm{OPT}$, where OPT is the value of an optimal solution. The

[4] In Dost et al.'s work, similarity scores between vertices are computed using their sequence similarity, which is then used when matching a query network to the base network. The classical problem of subgraph isomorphism on graphs with bounded parameters has been studied by Eppstein [45, 46].

usage of approximation algorithms in bioinformatics includes genome assembly (shortest superstring problem), multiple sequence alignment problem, and phylogenetic tree construction (Steiner tree problem), just to name a few.

Despite the diverse application of approximation algorithms in bioinformatics, the area of PPI networks has seen limited applications. In particular, to the best of our knowledge, no combinatorial algorithm has been proposed so far to identify direct protein interactions (Chapter 4) or protein complexes (Chapter 5) purely from the topological structure of the given PPI network. Such scarcity of applications may be due to the fact that even defining a clean objective function is a nontrivial task. In fact, most existing approaches propose different objective functions, for which heuristic algorithms are used. Moreover, it is often unclear whether these approaches directly capture the experimental processes that produced the PPI data.

In this section, we present a glimpse of approximation algorithms by discussing a few problems as examples. Since the focus of this section is to introduce techniques for devising approximation algorithms, we refrain from discussing the latest results on each problem with the best approximation guarantee. For more extensive surveys, Vazirani [140], and Williams and Shmoys [145] give excellent tutorials on the field.

2.3.1 Lower-bounding the Optimum

When designing an approximation algorithm, one needs to compare the cost of the obtained solution with the cost of an optimal solution in order to establish the approximation guarantee. This is typically done by finding a good lower bound (for minimization problems) on the cost of an optimal solution. For example, consider the vertex cover problem:

CARDINALITY VERTEX COVER. Given an undirected graph $G = (V, E)$, find a set $C \subseteq V$ such that every edge has at least one endpoint incident at C, and $|C|$ is minimized.

A lower bound on the size of an optimal vertex cover can be found using a maximal matching[5] of G. With respect to a maximal matching $\mathscr{M}$, any vertex cover has to pick at least one endpoint of the edges in $\mathscr{M}$. Therefore, the size of $\mathscr{M}$ is at most the size of an optimal vertex cover. This gives rise to the following simple algorithm:

1. Find a maximal matching $\mathscr{M}$ in G.
2. Output *both endpoints* of edges in $\mathscr{M}$.

Observe that the chosen vertices are adjacent to all the edges of G, as otherwise any uncovered edge would have been added to the matching. Furthermore, the size of the cover is exactly $2|\mathscr{M}|$. Since $|\mathscr{M}| \leq \text{OPT}$, the size of the cover is at most $2 \cdot \text{OPT}$, and thus the algorithm is a 2-approximation algorithm.

Application: Phylogenetic Trees [18] Consider a set of m species with n possible characteristics. We can represent this by a $0-1$ matrix M where $M_{i,j} = 1$ if species i has characteristic j. Then, we say M has a *perfect phylogenetic tree* if there exists a tree T such that: (1) there are m leaves, one for each species; (2) Each characteristic label corresponds to one edge in T (there may be edges with blank labels); (3) for each leaf s_i, the path from root to s_i contains exactly those characteristics possessed by s_i. See Figure 2.3 for an example of M and a possible perfect phylogenetic tree.

Let $\mathbf{S}_j$ be the species that have characteristic $c_j = 1$. Then the following characterization holds.

Theorem 2.3. *M has a perfect phylogenetic tree if and only if $\{\mathbf{S}_1, \ldots, \mathbf{S}_n\}$ form a laminar family.*[6] □

Thus, we say that two characteristics c_i and c_j have a *conflict* if $\mathbf{S}_i$ and $\mathbf{S}_j$ intersect, while $\mathbf{S}_i \setminus \mathbf{S}_j \neq \emptyset$ and $\mathbf{S}_j \setminus \mathbf{S}_i \neq \emptyset$. If we remove characteristics so that there are no conflicts, then

[5] Given a graph $G = (V, E)$, a subset of edges $\mathscr{M} \subseteq E$ is a *matching* if no two edges in $\mathscr{M}$ share a vertex. We say a matching is maximal if no more edges of $E \setminus \mathscr{M}$ can be added to $\mathscr{M}$.

[6] A set system is a laminar family if, for each pair of sets $\mathbf{S}_i$ and $\mathbf{S}_j$, either $\mathbf{S}_i \subseteq \mathbf{S}_j$, or $\mathbf{S}_j \subseteq \mathbf{S}_i$, or $\mathbf{S}_i \cap \mathbf{S}_j = \emptyset$.

	c_1	c_2	c_3	c_4	c_5
s_1	1	1	0	0	0
s_2	0	0	1	0	0
s_3	1	1	0	0	1
s_4	0	0	1	1	0
s_5	0	1	0	0	0

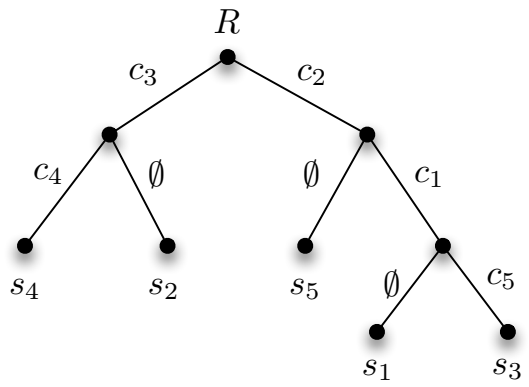


Figure 2.3: Matrix M indicating the characteristics of a set of species (left) and a perfect phylogenetic tree for M (right). Tree edges with $\emptyset$ indicate unlabelled edges.

there is a perfect phylogenetic tree for the remaining characteristics. Thus, we want to find the smallest set of characteristics to remove. We can turn this problem into an instance of vertex cover: define a *conflict graph* whose vertices correspond to characteristics $c_1, \ldots, c_n$, and vertices c_i and c_j share an edge if and only if they conflict. Then, a minimum cardinality vertex cover in the conflict graph consists of a minimum set of conflicting characteristics to remove.

2.3.2 Polynomial Time Approximation Schemes

Some NP-hard optimization problems are approximable to an arbitrary degree defined by the error parameter $\epsilon > 0$. For an NP-hard optimization problem Π with an objective function f, we say that algorithm A is an *approximation scheme* for Π if on input (I, ϵ), A outputs a solution s such that:

- $f(I,s) \leq (1+\epsilon) \cdot$ OPT if Π is a minimization problem.
- $f(I,s) \geq (1-\epsilon) \cdot$ OPT if Π is a maximization problem.

The algorithm A is called a *polynomial time approximation scheme* (PTAS), if for each fixed $\epsilon > 0$, the running time of A is bounded by a polynomial in the size of the instance

I. Note that the running time of a PTAS depends arbitrarily on ϵ. If, in addition, the running time is bounded by a polynomial in both $|I|$ and $1/\epsilon$, we say the algorithm is *fully polynomial time approximation scheme* (FPTAS). Here we give an example of an FPTAS for the knapsack problem.

KNAPSACK PROBLEM. Given a set $S = \{a_1, \ldots, a_n\}$ of objects with $\text{cost}(a_i) \in \mathbb{Z}^+$, $\text{profit}(a_i) \in \mathbb{Z}^+$, and $B \in \mathbb{Z}^+$, find a subset of S whose total cost is bounded by B, and whose total profit is maximized.

The design of a PTAS is often done via first mapping the problem instance to a coarser instance, and the new instance is then solved exactly by dynamic programming approach. Before describing the FPTAS for the knapsack problem, we introduce the notion of *pseudo-polynomial time algorithms*. An algorithm is efficient if its running time is bounded by a polynomial in $|I|$, the size of the instance. However, some problems contain numbers as part of the problem instance. For example, a knapsack problem instance contains cost and profit for each object a_i. Let $|I_u|$ denote the size of a problem instance I, where all the numbers in the instance are written in unary. Then, an algorithm is a pseudo-polynomial time algorithm if its running time is bounded by a polynomial in $|I_u|$. Because the knapsack problem is NP-hard, we do not know whether an efficient algorithm exists; however, there is a pseudo-polynomial time algorithm for this problem.

Let P be the profit of the most profitable object in S. Then nP is an upper bound on the profit of an optimal solution. For every $1 \leq i \leq n$ and $1 \leq p \leq nP$, let $S(i,p)$ denote a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total cost is minimized. Further, define $A(i,p)$ to be the cost of the set $S(i,p)$. If no such set exists, set $A(i,p) = \infty$. Then $A(1,p)$ is known for each value of p, where $1 \leq p \leq nP$, and the following recurrence holds.

$$A(i+1,p) = \begin{cases} \min\{A(i,p), \text{cost}(a_{i+1}) + A(i, p - \text{profit}(a_{i+1}))\}, & \text{if profit}(a_{i+1}) < p \\ A(i,p), & \text{otherwise} \end{cases}$$

A dynamic programming algorithm using this recurrence will compute all values of $A(i,p)$ in $O(n^2P)$ time. Finally, the optimal solution is found in time $O(nP)$ by looking for $\max\{p \mid A(n,p) \leq B\}$. Thus, this is a pseudo-polynomial time algorithm for the knapsack problem.

Observe that, if the profit of each object is bounded by a polynomial in n, the above algorithm would run in polynomial time. This observation leads us to an FPTAS for the knapsack problem. We first scale the profit down to small numbers so that the profit of each object is polynomially bounded by n. Then we can run the above algorithm to compute the optimal solution with respect to the scaled profit function. By carefully scaling with respect to the error parameter ϵ, we shall obtain a solution that is at least $(1-\epsilon)$OPT in time polynomial in $|I|$ and $1/\epsilon$.

Here is the FPTAS for the knapsack problem:

1. Let $K = \epsilon P/n$.
2. For each object a_i, define $\text{profit}'(a_i) = \lfloor \frac{\text{profit}(a_i)}{K} \rfloor$.
3. Run the dynamic programming algorithm with the scaled profit function, $\text{profit}'(a_i)$ to obtain the most profitable set S'.
4. Return S'.

We now show that $\text{profit}(S') \geq (1-\epsilon)$ OPT. Let S^* denote the optimal set with respect to the original profit function. For each object a_i, due to the rounding down at Step 2,

$K \cdot \text{profit}'(a_i)$ is smaller than $\text{profit}(a_i)$ by a difference at most K. Therefore,

$$\text{profit}(S^*) - K \cdot \text{profit}'(S^*) \leq nK.$$

Furthermore, S' must be at least as good as S^* under the scaled profits. By scaling the profits back up by K we have:

$$\text{profit}(S') \geq K \cdot \text{profit}'(S^*) \geq \text{profit}(S^*) - nK = \text{OPT} - \epsilon P \geq (1-\epsilon) \cdot \text{OPT},$$

since $\text{OPT} \geq P$.

The running time of the algorithm is $O(n^2 \lfloor \frac{P}{K} \rfloor) = O(n^3 \cdot \frac{1}{\epsilon})$, which is polynomial in both n and $1/\epsilon$.

Application: Target Protein Set. Due to limited resources, structural biologists often face the problem of which protein to study. Because of diverse nature of protein structures, certain protein molecules are more difficult to crystallize than others. It would be beneficial, therefore, if the chosen set of proteins are not too expensive to investigate. Furthermore, certain proteins are more *important* than others due to biological relevance, homology against existing results on similar proteins, etc. As such, researchers want to find the set of target proteins that maximizes the importance staying under their budget.

We can model this simple problem of task selection as follows: for each protein a_i, define the cost of investigating it to be $\text{cost}(a_i)$, and the importance of the particular protein to be $\text{profit}(a_i)$. If we let B to be the total resources that we have at our disposal, an optimum solution to this knapsack problem gives us the most profitable set of proteins we can study.

2.3.3 LP-based Algorithms

Linear programming gives a profound framework in which approximation algorithms can be designed. Many NP-hard optimization problems can be stated as integer programs. While solving an integer program itself is NP-hard in general, solving a (fractional) linear program is not, and thus the linear relaxation of the integer program provides a natural way of bounding the cost of the optimal solution.

A widely used approach for designing LP-based approximation algorithms is *LP-rounding*. In this approach, a fractional solution to the LP-relaxation is first obtained, and it is then converted to an integral solution by a rounding scheme. The approximation guarantee is established by comparing the cost of the fractional and integral solutions. For more detailed discussion on the this technique, see [140]. Here we present a simple example of the LP-rounding technique via a weighted version of the set cover problem.

WEIGHTED SET COVER. Given a universe U of n elements, a collection of subsets of U, $\mathscr{S} = \{S_1, \ldots, S_k\}$, and a cost function $c : \mathscr{S} \to \mathbb{Z}^+$, find a minimum cost subcollection of $\mathscr{S}$ that covers all elements of U.

We can formulate this problem as an integer program by assigning a binary variable x_S for each set $S \in \mathscr{S}$, whose value is set to 1 if S is picked in the set cover, and 0 otherwise. Furthermore, for each element $e \in U$, at least one of the sets containing e must be picked. So we have:

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathscr{S}} c(S) \cdot x_S & \\ \text{subject to} \quad & \sum_{S : e \in S} x_S \geq 1 & \forall e \in U \\ & x_S \in \{0,1\} & \forall S \in \mathscr{S} \end{aligned}$$

Let OPT denote the optimum solution of this integer program. The LP-relaxation of this integer program is obtained by replacing the domain of variables x_S with $0 \leq x_S \leq 1$ for all $S \in \mathscr{S}$. The derived linear program can then be solved optimally to obtain OPT_f. By construction, $\text{OPT}_f \leq \text{OPT}$ since the optimum integral solution is feasible for the LP.

Then, we can apply the rounding scheme as follows: pick all sets S for which $x_S \geq 1/\alpha$ in the OPT_f, where α is the frequency of the most frequent element (i.e. the maximum number of sets containing an element). Let C denote the resulting subcollection of sets chosen by the rounding scheme. To see C is a valid set cover, consider an arbitrary element e. Since e is in at most α sets, one of these must have been assigned at least $1/\alpha$ by OPT_f. Thus, e is covered by C.

To check the approximation ratio, the rounding scheme increases x_S by a factor of at most α. Therefore, the cost of C is at most $\alpha \cdot \text{OPT}_f$.

Application: Representative Protein Set. In Section 2.3.2, we introduced the problem of choosing a set of target proteins to investigate. Here we modify this problem slightly. Instead of maximizing the *profit* we gain by studying a set of proteins, we consider the notion of *representative protein set*: from a large collection of protein molecules, we would like to study the ones that somehow *represent* the entire collection of proteins. Such a representative profile would provide us a succinct, yet global picture of the given protein sample.

We can model this problem as follows: for each protein x, define the cost function $w(x) \geq 0$ measured by the difficulty of crystallizing x. Then, for each pair (x, y) of proteins, define a distance function $d(x, y) \geq 0$ measured by the sequence alignment of the two proteins. We can then construct a graph $G = (V, E)$ where each vertex corresponds to a protein, and two vertices are joined by an edge if and only if the corresponding proteins are within some prescribed distance Δ. We say a subset $V' \subseteq V$ of vertices is *representative*, if every vertex in V is adjacent to at least one vertex in V'. Then the problem of finding a minimum cost representative set is precisely the *dominating set* problem.

WEIGHTED DOMINATING SET. Given a graph $G = (V, E)$ and a cost function $c : V \to \mathbb{Z}^+$, find a minimum cost subset $D \subseteq V$ such that every vertex not in D is joined to at least one vertex in D.

It is well known that dominating set is equivalent to set cover [83]. To reduce dominating set to set cover, construct a set cover instance as follows: the universe U is V. For each vertex $v \in V(G)$, create a set S_v containing the vertex v and all vertices adjacent to v. For each set S_v, define its cost to be $c(v)$. Now, if D is a dominating set for G, then $C = \{S_v : v \in D\}$ is a feasible set cover with the same cost. Conversely, if $C = \{S_v : v \in D\}$ is a set cover, then D is a dominating set for G with the same cost.

Therefore, we can solve the representative protein set problem using the LP-rounding scheme for set cover.

2.4 Genetic Algorithms

In order to solve optimization problems, a search heuristic called a *genetic algorithm* (GA) often offers useful solutions. Inspired by natural evolution, genetic algorithms are especially useful when a large number of candidate solutions can be efficiently generated, and pairs of candidates can be somehow "merged" to obtain a better offspring of the two parent solutions. Each candidate solution is typically encoded as a string of characters. A genetic algorithm can then be designed via the following four main phases:

I. *Initialization*: Many individual solutions are randomly generated to form an initial population that spans the entire search space.

II. *Fitness function*: Each candidate solution is evaluated by the fitness function that gives a score based on the optimization criteria.

III. *Crossover*: Given two parent solutions, the crossover operation generates an offspring based on its parents. Various methods have been proposed for the crossover

operation, with one point "cut-and-swap" being the simplest form, while it can be fully customized in a problem-specific way.

IV. *Mutation*: In order to maintain genetic diversity from one generation to the next, a prescribed portion of the population goes through a mutation operation. The mutation operation is typically done by randomly switching characters in the candidate string, which can be fine-tuned depending on the application.

Using these operators, a large pool of candidate solutions are maintained from one generation to another. Since a genetic algorithms is a search heuristic, one typically cannot provide a theoretical bound on the output quality. Further, the fitness operator only tells us whether one candidate solution is better than other existing candidates, and thus it is never clear when to stop the algorithm. However, when a large initial population can be easily generated, and the fitness function can be evaluated quickly, carefully designed GAs provide an efficient approach to find good solutions. In this thesis, we shall present a highly customized genetic algorithm to find direct physical PPI network in Chapter 4. For more detailed discussions on this optimization technique, see [64, 94].

Chapter 3

Protein Quantification with Shared Peptides

With the advance of mass spectrometry (MS) based techniques, biologists in proteomics research are not only interested in determining the presence of proteins within a mixture, but also the *quantitative* nature of the constituent proteins, i.e. their abundance. MS-based shotgun proteomics approaches this problem by first shattering the (unknown) protein molecules into shorter fragments known as *peptides*, and then reading the peptides present in the mixture using a mass spectrometer.

Mass spectrometers typically work by first producing ions with charges based on the masses of the peptides, and then separating the ions by their mass-to-charge ratios. Various methods have been proposed to convert the peptide molecules into ions in the gas phase using electrospray ionization (ESI) [144] and matrix assisted laser desorption ionization (MALDI) [85, 116]. The produced ions can then be detected and processed into mass spectra. There are several algorithms to analyze the mass spectra and either identify the peptides present or quantify the peptide abundance [59, 115, 135].

Among these quantification methods, approaches known as *label-free* methods have become popular due to the low cost and simplicity of the experiments. These methods

work under the premise that more abundant peptides would produce a larger number of spectra. By counting the number of spectra, we can obtain measures such as spectral counting [30, 109], and peptide counting [87, 143]. Peptide count is the measure of how well a given protein is represented in a sample by counting the number of unique peptides. On the other hand, spectral counting simply counts the number of spectra for each peptide. As such, while peptide counting measures the coverage of the parent protein sequence, spectral counting provides a concrete measure of peptide abundance that can later be translated into protein abundance.

This chapter deals with the next step in the protein quantification problem: translating the peptide abundance into protein abundance. In order to identify (and quantify) constituent proteins in a mixture, the detected peptides often serve as indicators for the parent proteins. For example, one of the most widely used approaches for MS protein identification is the Mascot score[1] [114]. Given a set of peptide mass spectra, the Mascot score measures the probability that the match between the mass spectra and a particular protein sequence is a random event (thus a lower probability yielding a higher score), and is calculated for each protein in the protein sequence database. To compute this measure, each protein sequence in the database is digested and fragmented *in silico*, and the resulting "theoretical spectrum" is compared against the experimental spectrum observed by mass spectrometers. Consequently, the computation of Mascot scores involves considering all possible digested peptides for each protein in the database.

However, such an approach creates a problem when there is a peptide that belongs to more than one parent protein; in other words, the mapping of peptides to proteins is no longer one-to-one. These ambiguous peptides are called *shared peptides*, and Nesvizhskii and Aebersold [108] recently highlighted the complications in protein identification with the presence of shared peptides. Unfortunately, shared peptides are prevalent in several mass spectrometry datasets. For example, proteins in a large protein family have similar sequences that may create identical peptides when trypsinized, and al-

[1] Mascot score is the *in silico* method to identify proteins from a sequence database, used by the Mascot software (`http://www.matrixscience.com`).

ternative splicing may also create nearly identical protein isoforms. However, in most studies of protein identification and quantification, shared peptides are often simply ignored [100, 151].

In this chapter, we study the problem of identifying and quantifying the constituent proteins from MS spectra, without having to exclude the existence of shared peptides. Depending on the mixture of proteins under investigation, the shared peptides can constitute as much as 50% of the peptides in the mixture [42, 81]. Consequently, incorporating the shared peptides into analyses would certainly lead to a more accurate quantification of constituent proteins.

Recently, shared peptides have been a useful tool for protein quantification in several studies. For example, Jin et al. [81] proposed a statistical model that finds groups of proteins sharing significant number of peptides, namely peptide-sharing closure groups (PSCG). In their work, proteins grouped together in the same PSCG were shown to be biologically related, and when applied to data from differential expression analysis, all the peptides within each group showed consistent abundance relative to controls, justifying the inclusion of shared peptides in protein quantification studies.

On the other hand, Dost et al. [42] recently proposed a clean combinatorial model for quantifying relative protein abundance. Their model is built around peptide-protein relationships and, using the relative peptide abundance data from multiple samples, they set up a linear program to obtain relative protein abundance. As their input data is relative (e.g. peptide s_i is expressed 5 times more in sample x than in sample y, say), their output is also in relative ratios across multiple samples. As such, solving the linear program optimally (which gives a fractional value to the abundance of each protein) sufficed to give a viable solution.

As described, existing studies of protein quantification using shared peptides focus primarily on *relative* abundance measures across different samples. This may be because the quality of peptide abundance measures is relatively poor, and thus most studies only consider the differential profiling aspect of protein abundance. However, vari-

ous methods for absolute peptide quantification have been recently proposed [9, 61, 91], and the quality of peptide abundance measures is improving rapidly. In this chapter, therefore, we consider the problem of *absolute* protein quantification under the assumption that absolute peptide abundance is available. In Section 3.1, we modify the model of Dost et al. to suit our needs for absolute protein quantification, and formulate various optimization problems. Then, in Section 3.2, we establish the computational hardness of these problems, and design approximation algorithms in Section 3.3. In Section 3.4, we evaluate the performance of our approaches using both simulated and real MS data. Finally, Section 3.5 discusses how our approaches can be modified to handle degenerate cases, and incorporate additional biological data such as peptide detectability.

3.1 Problem Formulation for Protein Quantification

In this section, we formulate optimization problems for protein quantification. Let $W = \{w_1, \ldots, w_m\}$ denote the set of all known protein sequences, and let $S = \{s_1, \ldots, s_n\}$ denote the set of peptides we observe from a given mass spectrometry experiment[2]. We assume that we are given the set of protein sequences and peptide sequences, together with the abundance of each observed peptide sequence. Then we want to find the protein abundance that can be constructed from the given peptide abundance. For simplicity, we make the following assumptions.

1. For each protein w_j, a peptide s_i may only appear in w_j at most once.
2. The absolute abundance for each observed peptide is given as a positive integer, and thus protein abundance should be a nonnegative integer.

Lifting the first assumption can be done easily as shall be discussed in Section 3.5. The absolute peptide abundance can be estimated by the spectral counting method, as

[2] We use w for words and s for syllables as an analogy to proteins and peptides.

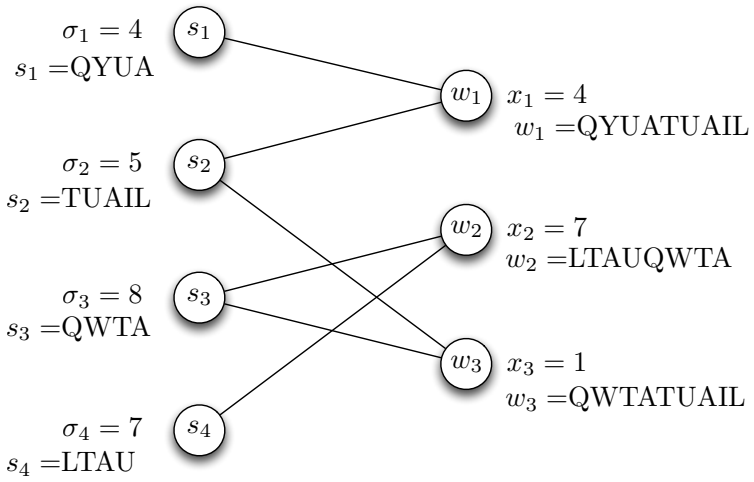


Figure 3.1: An example of the bipartite graph from peptides $(s_1,\ldots,s_4)$ and the parent proteins $(w_1,\ldots,w_3)$. Peptides s_2 and s_3 are shared peptides. The abundance of the peptides are indicated by σ_i while x_j's denote the protein abundance.

it measures the number of corresponding mass spectra for each peptide.

Let us construct a bipartite graph $G=(S,W;E)$ with $|S|=n$, $|W|=m$, where $(i,j)\in E$ if and only if peptide s_i is a substring of protein w_j. Let A be an $n\times m$ matrix where $A_{i,j}=1$ if $(i,j)\in E$, and 0 otherwise. Let σ_i be a positive integer that denotes the multiplicity of a peptide s_i, for $1\leq i\leq n$. Then, let x_j denote the unknown variable for multiplicity of protein w_j. Figure 3.1 gives an example of the formulation. Then, we can encode our problem into a system of linear equations as follows:

Problem 3.1.

$$\sum_{j=1}^{m} A_{i,j}\cdot x_j = \sigma_i \qquad i=1\ldots n$$

$$x_j\in\mathbb{Z}^+\cup\{0\} \qquad j=1\ldots m$$

This model captures the dependencies of peptides on proteins, and our problem reduces to solving a system of linear equations. However, there remain a few practical issues. The multiplicity of peptides σ_i from MS contains a significant amount of noise,

which would lead to inaccurate solutions or non-satisfiable system of equations. Further, even after assuming the peptide multiplicities are accurate, the system may not contain a unique, integral solution. We thus need a relaxed set of constraints and an optimization criterion to select the most plausible solution. We propose to modify this model in four different ways to progressively capture the data being analyzed in an increasingly realistic manner.

MULTICOVER. We can relax the equality constraints: rather than requiring x_j's sum up to precisely σ_i for each peptide W_i, we turn them into covering inequalities, and minimize the sum of x_j's.

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} x_j & \\ \text{subject to:} \quad & \sum_{j=1}^{m} A_{i,j} x_j \geq \sigma_i & i = 1 \ldots n \\ & x_j \in \mathbb{Z}^+ \cup \{0\} & j = 1 \ldots m \end{aligned}$$

Observe that this problem is closely related to the *set cover* problem. In fact, our formulation above is a generalization of set cover known as *multicover*: instead of requiring each element (peptide, in our case) to be covered at least once, each element needs to be covered at least the prescribed number of times. If each set (protein, in our case) can be picked at most once, the problem is called *constrained* multicover, while if a set can be covered arbitrarily many times, the problem is called *unconstrained* multicover, which is precisely the formulation above.

MINIMUMPROTEINTYPES. One further refinement from above is to look for a solution with the smallest number of different proteins, that is, minimize the number of x_j's with

positive values.

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} \chi(x_j) & \\ \text{subject to:} \quad & \sum_{j=1}^{m} A_{i,j} x_j \geq \sigma_i & i = 1 \ldots n \\ & x_j \in \mathbb{Z}^+ \cup \{0\} & j = 1 \ldots m \end{aligned}$$

where $\chi(x_j) = 1$ if $x_j > 0$, and $\chi(x_j) = 0$ otherwise.

In this problem, we want to find a solution that uses a smallest possible number of *distinct protein types*. Since the objective function is nonlinear, the resulting problem is no longer an integer linear program. While nonlinear integer programs are much harder class of problems in general, as we shall see later, we can still solve the above program, via a reduction to set cover.

MINIMUMUNIFORMERROR. Instead of simply looking for covers on σ_i's, we can allow an error term ϵ such that the x_j's sum to a value within $\sigma_i \pm \epsilon$.

$$\begin{aligned} \text{minimize} \quad & \epsilon & \\ \text{subject to:} \quad & \sigma_i - \epsilon \leq \sum_{j=1}^{m} A_{i,j} x_j \leq \sigma_i + \epsilon & i = 1 \ldots n \\ & x_j \in \mathbb{Z}^+ \cup \{0\} & j = 1 \ldots m \end{aligned}$$

This formulation looks for a solution such that, for each peptide s_i, the proposed multiplicities for proteins containing s_i sum up to a value within ϵ of σ_i. This introduction of additive error term helps us incorporate noise in the MS data while adhering to the peptide abundance σ_i by minimizing the error term. One issue with this approach is that some peptides are significantly harder to detect than others, and thus the error

for those peptide multiplicities are much larger than peptides that are easier to identify. We can resolve this issue of non-uniform errors using the following formulation.

MINIMUMERRORSUM. While MINIMUMUNIFORMERROR looks for a solution where the uniform error term ϵ is minimized, here we introduce an individual error term ϵ_i for each σ_i, and minimize the sum $\sum_i \epsilon_i$.

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{n} \epsilon_i && \\ &\text{subject to:} && \sigma_i - \epsilon_i \leq \sum_{j=1}^{m} A_{i,j} x_j \leq \sigma_i + \epsilon_i && i = 1 \ldots n \\ & && x_j \in \mathbb{Z}^+ \cup \{0\} && j = 1 \ldots m \end{aligned}$$

Using these error terms allow different error for each peptide. We note that error terms ϵ_i can be weighted by "detectability" that would penalize differently any discrepancies for different peptides. We discuss such extensions in Section 3.5.

Each of these four problems captures a different aspect of protein quantification, and requires a different algorithmic technique. Although some problems clearly model the mass spectrometry data more realistically than others, the theoretical results build up progressively, and thus we shall discuss each problem in this chapter.

3.2 Hardness of Protein Quantification Problems

In this section, we study the computational hardness of the problems posed in Section 3.1. To show the hardness of a given problem, we need to find a problem that is known to be hard, and reduce its instances to our problem of interest. For that purpose,

we consider the following two NP-hard problems[3].

SETCOVER. [56, 86] Given a set S of elements and a collection of sets W over the ground set S, choose a minimum cardinality sub-collection of W that covers all elements in S.

EXACTCOVER. [56, 86] Given a collection W of sets over elements S, find a subset $W' \subseteq W$ such that each element in S belongs to exactly one set in W'.

Before showing the hardness of our four problems, we first consider the original formulation; Problem 3.1 is simply a system of linear equations, but we can define an integer program by adding an objective function as in MULTICOVER.

Problem 3.2.

$$\begin{aligned} & \textit{minimize} && \sum_{j=1}^{m} x_j && \\ \textit{subject to:} \quad & \sum_{j=1}^{m} A_{i,j} \cdot x_j = \sigma_i && && i = 1 \dots n \\ & x_j \geq 0 && && j = 1 \dots m \\ & x_j \in \mathbb{Z} && && \end{aligned}$$

The hardness of this problem is evident from EXACTCOVER. Due to unavoidable errors in MS measurements, this problem is of less practical interest, and we refrain from further discussing algorithmic approaches.

Theorem 3.3. *Problem 3.2 is* NP*-hard.*

Proof. Take an instance Π of EXACTCOVER, and transform into an instance $\phi(\Pi)$ of Problem 3.2 by assigning $\sigma_i = 1$ for all $1 \leq i \leq n$. If $\phi(\Pi)$ finds a solution of any size, then Π has a solution for EXACTCOVER. Otherwise, Π has no solution for EXACTCOVER. □

[3] See Section 3.6 for inapproximability results on these problems.

Next, the hardness of the multicover problem is well known [140].

Theorem 3.4. [140] MULTICOVER *is* NP*-hard.*

Proof. Consider the decision problem of MULTICOVER(K). Take an instance Π of SETCOVER(K), and cast into an instance $\phi(\Pi)$ of MULTICOVER(K), where $\sigma_i = 1$ for all i. If Π contains a solution of size k, then the corresponding chosen sets form a solution for $\phi(\Pi)$ of size k. Conversely, if $\phi(\Pi)$ has a multicover of size k, then there are at most k x_j's with positive values. These x_j's correspond to a set cover of size k for Π. □

The hardness of MINIMUMPROTEINTYPES comes from a reduction from SETCOVER.

Theorem 3.5. MINIMUMPROTEINTYPES *is* NP*-hard.*

Proof. Take an instance of set cover Π, and cast into an instance $\phi(\Pi)$ of MINIMUMPROTEINTYPES by assigning the coverage requirement $\sigma_i = 1$ for each element. If Π admits a set cover of size k, then the corresponding chosen proteins form a solution for $\phi(\Pi)$ of size k. Conversely, if $\phi(\Pi)$ admits a solution of size k, the proteins chosen with positive multiplicity correspond to a set cover of size k in Π. □

The error minimization problems can be shown to be hard by reductions from EXACTCOVER.

Theorem 3.6. MINIMUMUNIFORMERROR *is* NP*-hard.*

Proof. Take an instance Π of EXACTCOVER. Then, cast into an instance $\phi(\Pi)$ of MINIMUMUNIFORMERROR by assigning $\sigma_i = 1$ for $1 \leq i \leq n$. If $\phi(\Pi)$ contains a solution with $\epsilon = 0$, then Π contains an exact cover. Otherwise, Π contains no solution. □

A similar proof yields hardness for MINIMUMERRORSUM.

Theorem 3.7. MINIMUMERRORSUM *is* NP*-hard.* □

3.3 Approximation Algorithms for Protein Quantification

In this section, we design approximation algorithms for the formulated problems. While MULTICOVER and MINIMUMPROTEINTYPES are both covering problems, MINIMUMUNIFORMERROR and MINIMUMERRORSUM share a common theme of minimizing the error terms. For the covering problems, we obtain $O(\log n)$ approximation guarantees, and we design randomized algorithms for the error minimization problems.

3.3.1 An Algorithm for Multicover

The MULTICOVER problem formulated in Section 3.1 is an unconstrained version: the solution vector x may contain any nonnegative integers. In other words, each protein may be picked multiple times in order to meet the coverage requirement σ. For the purpose of approximation algorithms, however, we can instead focus on the constrained $0-1$ problem, where each protein can be either picked once, or not picked at all.

Lemma 3.8. *If there is an α-approximation algorithm for constrained multicover, then there is an α-approximation algorithm for* MULTICOVER.

Proof. Consider the unconstrained multicover problem. We first take the LP relaxation of the integer program, and solve the LP in polynomial time to obtain the fractional optimum solution, denoted by OPT_f. From the fractional optimum OPT_f: $x = (x_1, x_2, \ldots, x_m)$, each x_j can be split into an integral part I_j and a fractional part F_j, where $0 < F_j < 1$. Then, $\text{OPT}_f = I + F$. Now consider the "residual" IP:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} x'_j && (3.1) \\ \text{subject to:} \quad & \sum_{j=1}^{m} A_{i,j} \cdot x'_j \geq \sigma'_i && i = 1 \ldots n \\ & 0 \leq x'_j \leq 1 && j = 1 \ldots m \\ & x'_j \in \mathbb{Z} \end{aligned}$$

where

$$\sigma'_i = \max\{\sigma_i - \sum_{j=1}^{m} A_{i,j} I_j,\ 0\}$$

to make sure σ'_i remains nonnegative. Then $F = (F_1, F_2, \ldots)$ forms an optimal solution for the LP relaxation of the above residual IP (since otherwise we can replace it with a better solution for the residual IP and improve OPT_f).

Now, suppose we have an approximate solution A that is at most $\alpha \cdot F$ for some $\alpha > 1$. Then our overall solution is:

$$I + A \ \leq \ I + \alpha F \leq \alpha(I + F) = \alpha(\text{OPT}_f) \leq \alpha \cdot \text{OPT} \qquad \square$$

Therefore, from the standpoint of approximation algorithms, we can focus on the constrained multicover problem. Due to their resemblance, standard techniques for set cover can be applied with similar approximation guarantees, including the simple greedy algorithm (which iteratively picks the set containing the largest number of uncovered elements), or the LP rounding technique using the residual solution F.

3.3.2 An Algorithm for Minimum Protein Types

In Section 3.2, we showed that MINIMUMPROTEINTYPES is NP-hard via a reduction from set cover. While MINIMUMPROTEINTYPES is formulated as a nonlinear integer program,

a harder class of problems in general, our problem is in fact equivalent to the set cover problem.

Theorem 3.9. *Any algorithm for* SETCOVER *can be used to find a solution for* MINIMUMPROTEINTYPES *of the same size.*

Proof. Let Π denote an instance of MINIMUMPROTEINTYPES. Now consider the corresponding set cover instance $\phi(\Pi)$ constructed by replacing the coverage requirement with $\sigma_i = 1$ for each element i. Suppose $\phi(\Pi)$ contains a set cover of size k. Take the chosen sets in $\phi(\Pi)$, and assign the corresponding x_j's with arbitrarily large values so that each coverage requirement σ_i is satisfied. Since the value of the optimization function is the number of x_j's with positive values, we have a solution for MINIMUMPROTEINTYPES of value k. □

This implies that any approximation algorithm for SETCOVER can be used to compute a solution for MINIMUMPROTEINTYPES with the same approximation guarantee, and the inapproximability results for SETCOVER also hold for MINIMUMPROTEINTYPES. See Section 3.6 for inapproximability results on SETCOVER.

Finding the protein abundance σ for Minimum Protein Types. Observe that, in the proof of Theorem 3.9, our algorithm assigns arbitrarily large values to the x_j's chosen by the SETCOVER algorithm. In practice, however, such arbitrarily large values would result in solutions undesirable for biological purposes. To resolve this, we first run the SETCOVER algorithm to choose the smallest set of proteins (which would be assigned positive abundances), and then remove the unchosen proteins from the bipartite graph G. Then, we apply the algorithm for MULTICOVER on the residual bipartite graph. This way, one can be sure that we choose the smallest possible number of proteins, and further, assign the smallest possible abundances to the chosen proteins.

3.3.3 An Algorithm for Minimum Uniform Error

We reduce the problem to a $\{0,1\}$ problem.

Lemma 3.10. *If there is an α-approximation algorithm for constrained* MINIMUMUNIFORMERROR*, then there is an α-approximation algorithm for unconstrained* MINIMUMUNIFORMERROR.

Proof. Let's first spell out the original problem:

$$\begin{aligned} &\text{minimize} \quad \epsilon && && (\text{ILP}_1)\\ &\text{subject to:} \quad \sigma_i - \epsilon \leq \sum_{j=1}^{m} A_{i,j} x_j \leq \sigma_i + \epsilon && i = 1 \dots n\\ &\qquad x_j \in \mathbb{Z}^+ \cup \{0\} && j = 1 \dots m \end{aligned}$$

which we shall call ILP_1. Now, consider the LP relaxation of ILP_1, namely LP_1. Then, by definition, $\text{OPT}(\text{LP}_1) \leq \text{OPT}(\text{ILP}_1)$. Each value assigned to x_j in $\text{OPT}(\text{LP}_1)$ is a fractional value, which can be split into:

$$\text{OPT}(\text{LP}_1) = I + F$$

where $0 < F_j < 1$ for all j. We can then keep the solution given by I, and consider the residual problem given by the following integer program, called ILP_2:

$$\begin{aligned} &\text{minimize} \quad \epsilon && && (\text{ILP}_2)\\ &\text{subject to:} \quad \sigma'_i - \epsilon \leq \sum_{j=1}^{m} A_{i,j} x_j \leq \sigma'_i + \epsilon && i = 1 \dots n\\ &\qquad 0 \leq x_j \leq 1 && j = 1 \dots m\\ &\qquad x_j \in \mathbb{Z} && j = 1 \dots m \end{aligned}$$

where $\sigma'_i = \max\{\sigma_i - \sum_{j=1}^{m} A_{i,j} I_j,\ 0\}$.

Now, consider the optimum solution for the residual integer program, namely OPT(ILP_2). By definition, we have the linear relaxation LP_2 such that OPT(LP_2) $\leq$OPT(ILP_2). For OPT(LP_2), we claim that $F \leq$OPT(LP_2), and F is a feasible solution. To show $F \leq$ OPT(LP_2), suppose otherwise. Then, OPT(LP_2) can be pasted onto I to form:

$$I + \text{OPT}(\text{LP}_2) < I + F \leq \text{OPT}(\text{LP}_1)$$

contradicting the optimality of $OPT(LP_1)$.

Finally, suppose there is an approximate solution A for OPT(LP_2) that is at most $\alpha \cdot F$ for some $\alpha > 1$. Then our overall solution is:

$$I + A \leq I + \alpha \cdot F \leq \alpha(I + F) = \alpha \cdot \text{OPT}(\text{LP}_1) \leq \alpha \cdot \text{OPT}(\text{ILP}_1)$$

Therefore, the overall approach would yield an $\alpha-$approximation algorithm. □

We thus focus on the constrained $0-1$ problem. We can design a randomized rounding scheme where the fractional solution F is used as a probability, as shown in Algorithm 3.3.1.

Algorithm 3.3.1: Randomized Rounding Scheme for MINIMUMUNIFORMERROR

Input: Fractional solution vector F from OPT(LP_1).
Output: Binary solution vector x.

for $j = 1, \ldots, m$ **do**
  $p_j \leftarrow F_j$ (each $0 < F_j < 1$ is treated as a probability.);
  Flip a p_j-biased coin;
  if *heads* **then**
    $x_j \leftarrow 1$;
  end
  else
    $x_j \leftarrow 0$;
  end
end
Return x.

To analyze the rounding scheme, we define a random variable $Y_i = \sum_{(i,j)\in E} x_j$ where $x_j \in \{0,1\}$ is the result of tossing a coin for each j with probability p_j (a Poisson trial). By linearity of expectation, we have:

$$E[Y] = E[Y_1] + E[Y_2] + \ldots = \sum_{(i,j)\in E} E[p_j] = \tau_i.$$

where $\tau_i = \sum_{(i,j)\in E} F_j$, the coverage achieved by F. Since the fractional solution F has at most the optimum error from the integer program, the expected solution $E[Y]$ is optimal.

We now show that Algorithm 3.3.1 gives a small tail probability.

Lemma 3.11. *Algorithm 3.3.1 returns a solution such that*

$$Pr[Y_i > (1+\delta)\tau_i + \frac{6\log n}{\delta^2}] < \frac{1}{n^2}$$

and

$$Pr[Y_i < (1-\delta)\tau_i - \frac{4\log n}{\delta^2}] < \frac{1}{n^2}$$

Proof. To estimate the tail probability, we apply the Chernoff bound [29, 104] on Y_i. For the bound on the upper tail, we have:

$$Pr[Y_i > (1+\delta)\tau_i] < e^{-\tau_i \delta^2/3}$$

and for the bound on the lower tail:

$$Pr[Y_i < (1-\delta)\tau_i] < e^{-\tau_i \delta^2/2}$$

We show each case separately.

Case I. (Upper tail): Suppose $\tau_i \geq \frac{6\log n}{\delta^2}$. Then it follows that:

$$\begin{aligned} \tau_i &\geq \frac{6\log n}{\delta^2} \\ \Rightarrow \frac{\tau_i \delta^2}{3} &\geq 2\log n \\ \Rightarrow \log e^{\tau_i \delta^2/3} &\geq \log(n^2) \\ \Rightarrow e^{-\tau_i \delta^2/3} &\leq \frac{1}{n^2} \end{aligned}$$

and it follows that

$$Pr[Y_i > (1+\delta)\tau_i] < \frac{1}{n^2} \quad \text{if } \tau_i \geq \tfrac{6\log n}{\delta^2} \tag{1}$$

Now consider the case $\tau_i < \frac{6\log n}{\delta^2}$. We introduce an absolute error D such that $(1+\delta)\tau_i = \tau_i + D$. Then we can express as $D = \delta\tau_i$, and rewrite the assumption with the additive error D:

$$\begin{aligned} \tau_i < \frac{6\log n}{\delta^2} &\Rightarrow \tau_i < \frac{6\log n}{(D/\tau_i)^2} \\ &\Rightarrow D < \sqrt{\tau_i \cdot 6\log n} \end{aligned}$$

Moreover, the Chernoff bound for the upper tail now becomes:

$$\begin{aligned} Pr[Y_i > \tau_i + D] &\leq e^{-\frac{D^2}{3\tau_i}} \\ &< e^{-\frac{(\sqrt{\tau_i 6\log n})^2}{3\tau_i}} \quad \text{(by assumption)} \\ &= e^{-2\log n} = \frac{1}{n^2} \end{aligned}$$

Rewriting D in δ gives

$$D < \sqrt{\tau_i \cdot 6\log n} = \frac{6\log n}{\delta^2},$$

and thus we obtain

$$Pr[Y_i > \tau_i + \frac{6\log n}{\delta^2}] < \frac{1}{n^2} \quad \text{if } \tau_i < \frac{6\log n}{\delta^2} \tag{2}$$

Combining (1) and (2), we have

$$Pr[Y_i > (1+\delta)\tau_i + \frac{6\log n}{\delta^2}] < \frac{1}{n^2}.$$

Case II (Lower tail): Consider the lower tail as below.

$$Pr[Y_i > (1-\delta)\tau_i] < e^{-\tau_i \delta^2/2}$$

First, suppose $\tau_i \geq \frac{4\log n}{\delta^2}$. Rearranging this gives $e^{-\tau_i\delta^2/2} \leq \frac{1}{n^2}$, and it follows that

$$Pr[Y_i < (1-\delta)\tau_i] < \frac{1}{n^2} \quad \text{if } \tau_i \geq \frac{4\log n}{\delta^2} \tag{3}$$

Now consider the case $\tau_i < \frac{4\log n}{\delta^2}$. As before, let D be an additive error such that $(1-\delta)\tau_i = \tau_i - D$. Then we have $D = \delta\tau_i$, and it follows that

$$\begin{aligned} \tau_i < \frac{4\log n}{\delta^2} &\Rightarrow \tau_i < \frac{4\log n}{(\frac{D}{\tau_i})^2} \\ &\Rightarrow \frac{D^2}{\tau_i} < 4\log n \\ &\Rightarrow D < \sqrt{\tau_i 4\log n} \end{aligned}$$

Now, the Chernoff bound can be rewritten:

$$\begin{aligned} Pr[Y_i < \tau_i - D] &< e^{-\tau_i \delta^2/2} \\ &= e^{-\frac{D^2}{2\tau_i}} \\ &< e^{-\frac{(\sqrt{\tau_i 4\log n})^2}{2\tau_i}} \\ &= e^{-2\log n} = \frac{1}{n^2} \end{aligned}$$

Rewriting D in δ gives

$$D < \sqrt{\tau_i \cdot 4\log n} = \frac{4\log n}{\delta^2},$$

and thus we obtain

$$Pr[Y_i < \tau_i - \frac{4\log n}{\delta^2}] < \frac{1}{n^2} \quad \text{if } \tau_i < \tfrac{4\log n}{\delta^2} \tag{4}$$

Combining (3) and (4), we have

$$Pr[Y_i < (1-\delta)\tau_i - \frac{4\log n}{\delta^2}] < \frac{1}{n^2}$$ □

As a result, for each s_i, flipping a coin independently with probability p_j for each neighbour w_j results in a value within the error bound of

$$[\,(1-\delta)\tau_i - \frac{4\log n}{\delta^2}\,,\,(1+\delta)\tau_i + \frac{6\log n}{\delta^2}\,]$$

with high probability for both the upper and lower tail. We can then put all the s_i's together to build a bound on the randomized algorithm.

Theorem 3.12. *The randomized rounding scheme finds a solution with optimal expected value for* MINIMUMUNIFORMERROR *with high probability.*

Proof. We use the union bound to put all s_i's together. Consider the upper tail: for each $i = 1 \ldots n$, let A_i denote the event that Y_i is greater than the specified bound. By

Lemma 3.11, $Pr[A_i] \leq \frac{1}{n^2}$ for each A_i. By the union bound, we have:

$$Pr[\cup_i A_i] \leq \sum_i Pr[A_i] = n \cdot \frac{1}{n^2} = \frac{1}{n}$$

Hence, the probability that no s_i falls above the bound is $1 - \frac{1}{n}$.

Similarly, for the lower tail, the probability that any s_i falls below the specified bound is $\frac{1}{n}$. Hence, the probability that no s_i falls below the bound is $1 - \frac{1}{n}$. □

3.3.4 An Algorithm for Minimum Error Sum

Here we discuss the MINIMUMERRORSUM problem where we allow independent error term ϵ_i for each peptide abundance σ_i. As before, we focus on the constrained $0-1$ problem via LP relaxation.

Lemma 3.13. *If there is an α-approximation algorithm for constrained* MINIMUMERRORSUM*, then there is an α-approximation algorithm for unconstrained* MINIMUMERRORSUM.

Proof. Similar to the proof of Lemma 3.10. □

We can thus focus on the constrained, $0-1$ problem. The randomized rounding scheme for MINIMUMUNIFORMERROR shown in Algorithm 3.3.1 can be applied here, too.

Theorem 3.14. *There is a randomized rounding scheme that finds a solution with optimal expected value for* MINIMUMERRORSUM *with high probability.*

Proof. Similar to the proofs of Lemma 3.11 and Theorem 3.12. □

3.4 Experimental Results

In this section, we discuss how our algorithms for the protein quantification problem have been tested empirically against simulated data as well as biological data from AP-MS experiments.

3.4.1 Performance on Simulated Data

String Trypsinization. To generate our simulated test data, we took a collection of 1000 random strings of length 150 over the alphabet of size 4. The abundance of each string w_j was then set to a random value between 1 and 100. We call the resulting multiset of strings $\hat{x}$. Then, we cut the strings at random into approximately 30 fragments to simulate the trypsinization of proteins. The string fragments are counted to form the peptide abundance σ for this input string set.

We acknowledge that the parameters chosen for the generation of protein / peptide data are not based on a realistic model of protein sequences (e.g. alphabet of size 20, realistic length for protein sequences, realistic trypsinization process, etc.). However, note that the performance of our approaches depends directly on the edge density of the bipartite graph constructed from the strings. The chosen parameters ensure that each protein consists of up to 30 peptides, and that the edge density of the resulting bipartite graph is similar to that seen in the real AP-MS data.

We first ran our algorithms on the synthetic data, and compared the results against the protein abundance of the original set of strings $\hat{x}$. Table 3.1 summarizes the results of these comparisons. However, because each problem has its own distinct objective function, comparing the output of our algorithms based on any one objective function is not very meaningful. Furthermore, the abundance $\hat{x}$ of the original set of strings may not be "optimal" for the input vector σ depending on the objective. In fact, as shown in Table 3.1, the original abundance vector $\hat{x}$ does not achieve the optimum objective value

Problem		Objective	Problem		Objective
MULTICOVER	Original $\hat{x}$	6439	MINIMUMUNIFORMERROR	Original $\hat{x}$	0
	Our result	5371		Out result	74
MINIMUMPROTEINTYPES	Original $\hat{x}$	1000	MINIMUMERRORSUM	Original $\hat{x}$	0
	Our result	887		Our result	782

Table 3.1: Output quality of the algorithms compared to the original abundance vector $\hat{x}$ (Objective indicates the value from the corresponding objective function). Note that in MULTICOVER and MINIMUMPROTEINTYPES, our solution achieves better objective values than the original abundance of strings, while in the error minimizing problems (i.e. MINIMUMUNIFORMERROR, MINIMUMERRORSUM) the original string abundances produces no error.

for MULTICOVER and MINIMUMPROTEINTYPES, while our approximate algorithms provide slightly better solutions than $\hat{x}$.

Consequently, we instead base our performance analyses on the distance between the proposed solutions and the original abundance $\hat{x}$. Let x, x' denote two solution vectors. Then, we define the distance between x and x' as the L_1-distance:

$$d(x,x') = \sum_i |x_i - x'_i|$$

where x_i denotes the i-th component of x. We can interpret this L_1-distance between our proposed solution x and the original abundance $\hat{x}$ as the error present in our prediction.

Robustness Analysis. When Gentleman and Huber [60] discussed errors that are prevalent in proteomics, they classified the measurement errors as two distinct types: *stochastic errors* and *systematic errors*. Stochastic errors have random variability, and thus can be reduced by simply repeating the experiment. On the other hand, systematic errors are recurrent in the measurements, which makes them difficult to identify and remove. From the perspective of the protein quantification problem, stochastic errors are embedded into the peptide abundance σ. In our formulation of the problems, MINIMUMUNIFORMERROR and MINIMUMERRORSUM in particular, we attempt to remove the stochastic errors by looking for a solution with the smallest error ϵ.

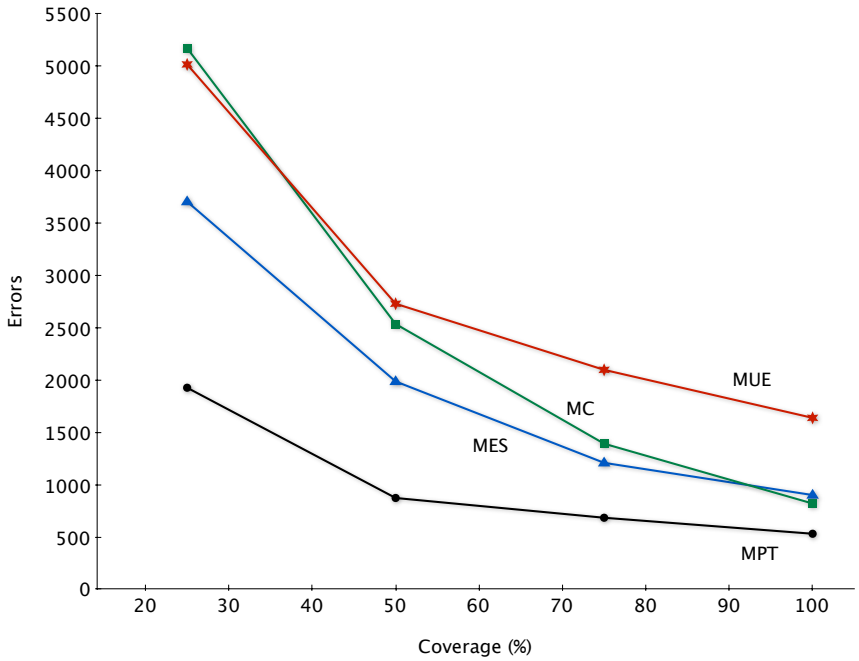


Figure 3.2: Robustness of the algorithms for MPT (MINIMUMPROTEINTYPES), MC (MULTICOVER), MES (MINIMUMERRORSUM) and MUE (MINIMUMUNIFORMERROR). Errors indicate the L_1 distance from the original protein abundances $\hat{x}$ (averaged over 5 distinct partial datasets).

On the other hand, because some peptides are notoriously difficult to identify, and the sensitivity of mass spectrometers is relatively poor, the mass spectrometry data is far from comprehensive. As a result, certain peptides (and their abundances) are missing in data, and form a systematic error that are recurrent throughout the dataset. We test our algorithms against such systematic errors by analyzing the robustness of the algorithms.

In order to test the robustness of our algorithms, we randomly selected and removed[4] a specified proportion of constraints before running the algorithms. For example, if we remove 25% of the constraints (the peptide abundance σ) from the system, solving the problem with the remaining 75% of the constraints can simulate the problem of quantifying the protein abundance with 75% sensitivity of the MS data.

After removing a fixed proportion of the input data (peptide abundance σ), we ran our algorithm for each problem and compared the results in terms of the errors pro-

[4] We note that, by removing a particular peptide abundance σ_i, we simply remove the corresponding constraint in the system rather than setting $\sigma_i = 0$ for that peptide.

duced as estimated by the metric $d(x, \hat{x})$. Figure 3.2 shows the result of this comparison for different input coverage (i.e. percentage of the remaining data). Among the four different problem formulations, MINIMUMPROTEINTYPES showed a closest fit to the original abundance $\hat{x}$ across the different input coverage, while MINIMUMERRORSUM and MULTICOVER showed similar performance until the input coverage fell below 50%. Furthermore, when comparing the amount of errors produced between input coverage at 50% and at 100%, the algorithm for MINIMUMPROTEINTYPES is shown to be the most robust among the four problem formulations.

3.4.2 Protein Quantification from AP-MS Data

To test our algorithms on real biological data, several datasets have been acquired from AP-MS experiments by our collaborators[5]. Spectral count is used to estimate the abundance of peptides, as it measures peptide concentrations in protein mixtures based on the number of tandem mass spectra obtained for each peptide [22, 30, 100]. Recall that each AP-MS experiment consists of a bait protein, together with peptides from its interacting partners, the prey proteins. We obtained the dataset from 3 separate mass spectrometry runs for 3 bait proteins, RPAP3, CDK9, and POLR2A. Since each of these datasets is produced by a distinct mass spectrometry experiment, we treat them as different samples.

As shown in Table 3.2, the three datasets from AP-MS experiments contain approximately 2000 peptides that belong to 1000 proteins. However, when restricted to only the connected components (in the bipartite graph) containing shared peptides, the dataset contains approximately 500 peptides over 100 proteins.

While the nontrivial portion of the problem is smaller in terms of the number of peptides and proteins, the corresponding bipartite graphs are much denser due to the peptides shared by many proteins. In fact, peptides in our datasets are shared by as

[5] Our test data was generously provided by Benoit Coulombe's lab at Institut de recherches cliniques de Montréal (IRCM), and consists of human PPI data from AP-MS experiments.

	Number	CDK9	POLR2A	RPAP3
Overall	proteins	1593	1386	901
	peptides	2592	2861	1689
	components	1320	1211	703
Non-trivial	proteins	125	70	125
	peptides	672	525	589
	components	21	19	26

Table 3.2: Network analysis of prey proteins from AP-MS experiments. Overall indicates the total number of proteins, peptides, and connected components for each dataset, while non-trivial indicates only the connected components containing shared peptides.

many as 19 proteins. For example, Figures 3.3 through 3.5 show the bipartite graphs for the dataset with CDK9, POLR2A, RPAP3 as the bait, respectively.

We executed our algorithms for MINIMUMPROTEINTYPES and MINIMUMERRORSUM for the nontrivial components (containing shared peptides) of each dataset. Furthermore, for smaller connected components with fewer than 20 proteins, we also computed exact solutions to the corresponding integer programs[6] , and found the results to be promising. In the case of MINIMUMPROTEINTYPES, both the integral optimum solution and our approximate solution often picked the same set of proteins (FP $\sim$ 10%, FN $\sim$ 15%), with only marginal differences in the abundance. In the case of MINIMUMERRORSUM, the sum of errors ($\sum_i \epsilon_i$) in our approximate solution differs from that of integral optimum solution by at most the maximum peptide abundance value (σ_i) in that component. While a pleasant result, this is no surprise since we obtain the significant part of our protein abundance from the optimum of LP relaxation.

Within our dataset, there were various instances where a group of peptides are shared by a large number of proteins. In such instances, our algorithm for MINIMUMPROTEINTYPES picked out only a few proteins as constituent proteins. For example, Figure 3.6 shows a connected component from CDK9 dataset, where all the peptides are shared by variants of the histone family. Among the 14 proteins that share the peptides in this component, only 3 proteins were selected with positive abundance values. This partic-

[6] Gurobi Optimizer (`http://www.gurobi.com/`) was used to solve the integer programs as well as their LP-relaxations.

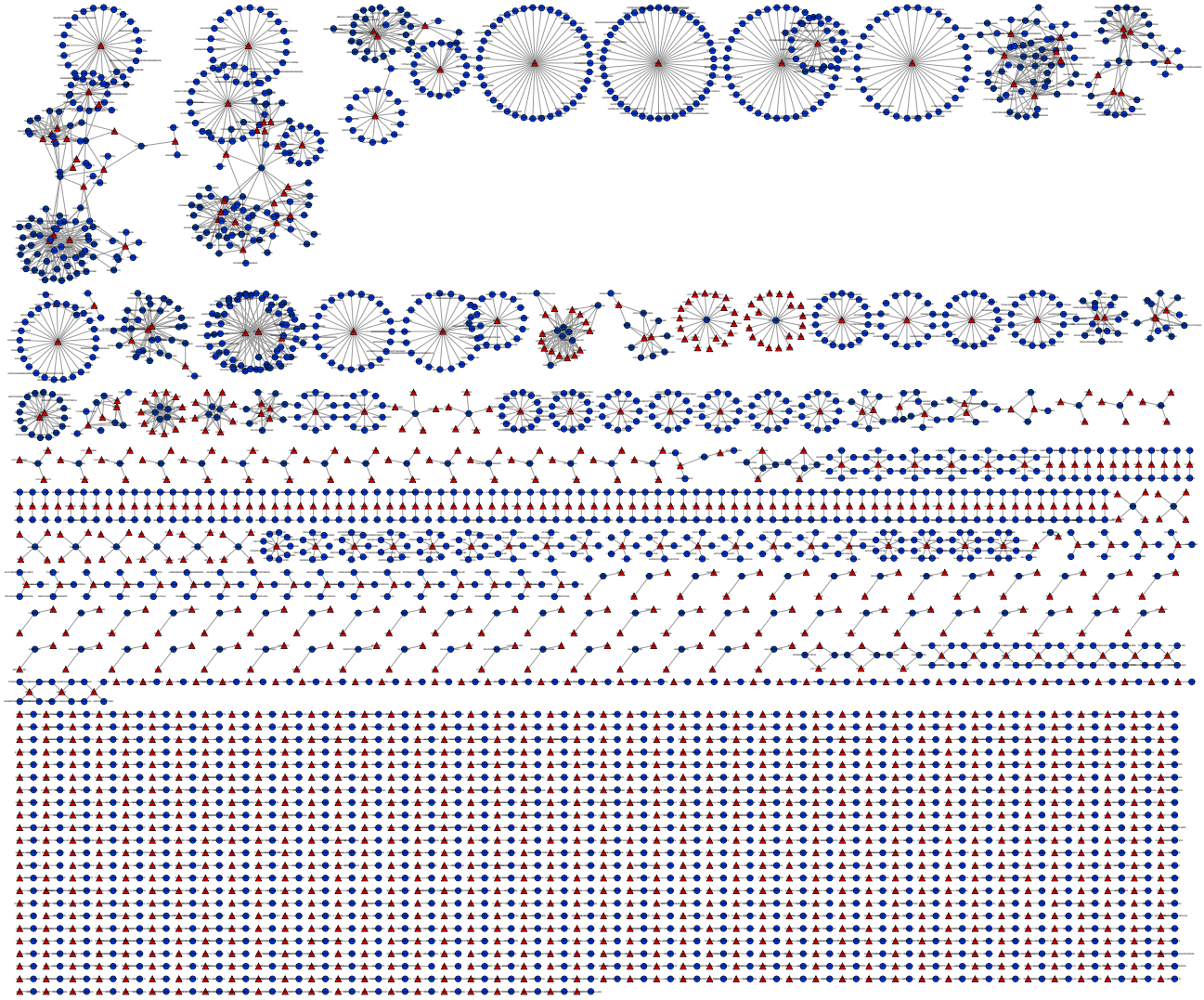


Figure 3.3: Peptide-protein graph constructed from an AP-MS experiment with CDK9 as a bait. Proteins are drawn as red triangular vertices while peptides are drawn as blue round vertices. Note that there are various peptides shared by several different proteins.

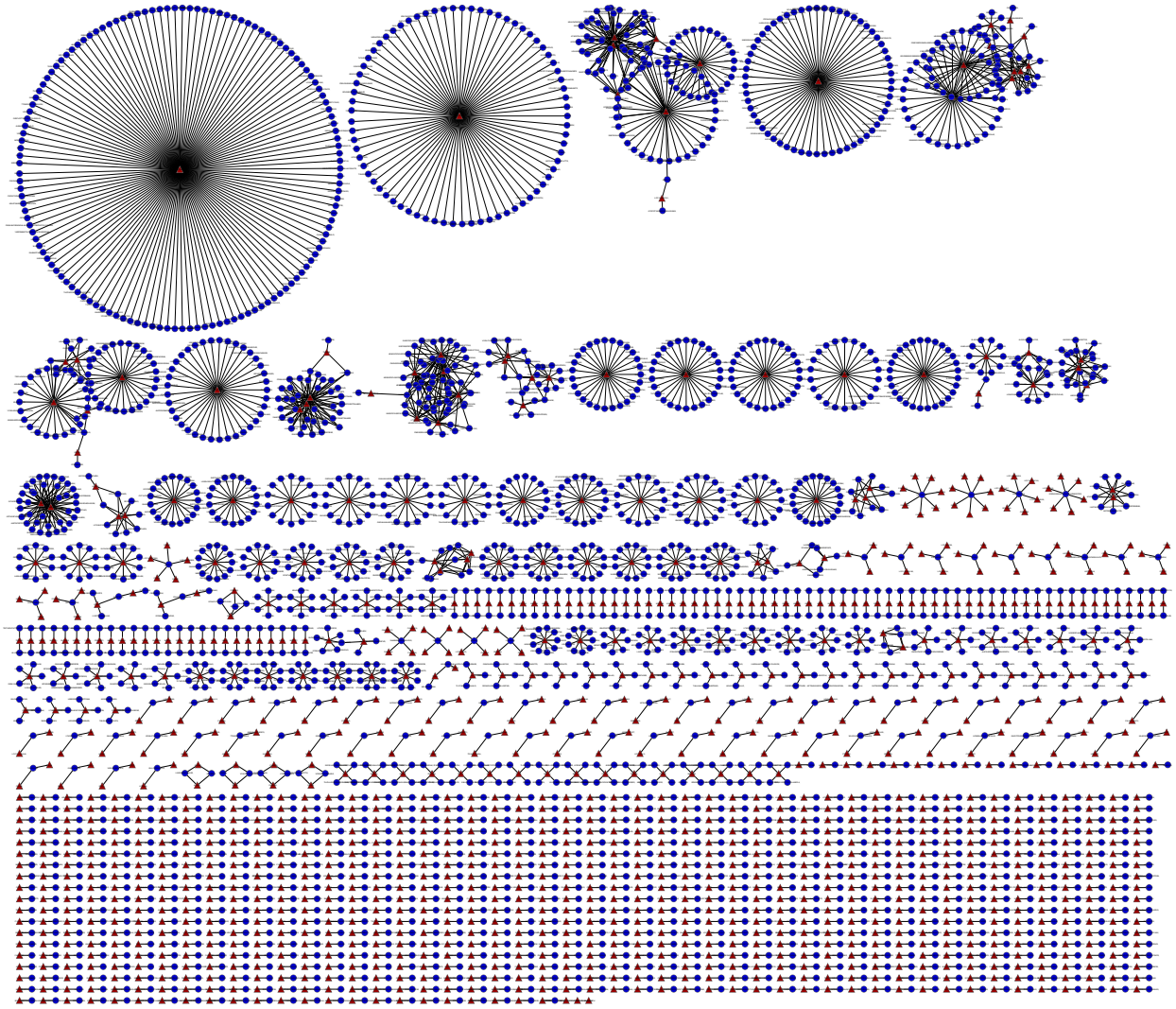


Figure 3.4: Peptide-protein graph constructed from an AP-MS experiment with POLR2A as a bait. Proteins are drawn as red triangular vertices while peptides are drawn as blue round vertices.

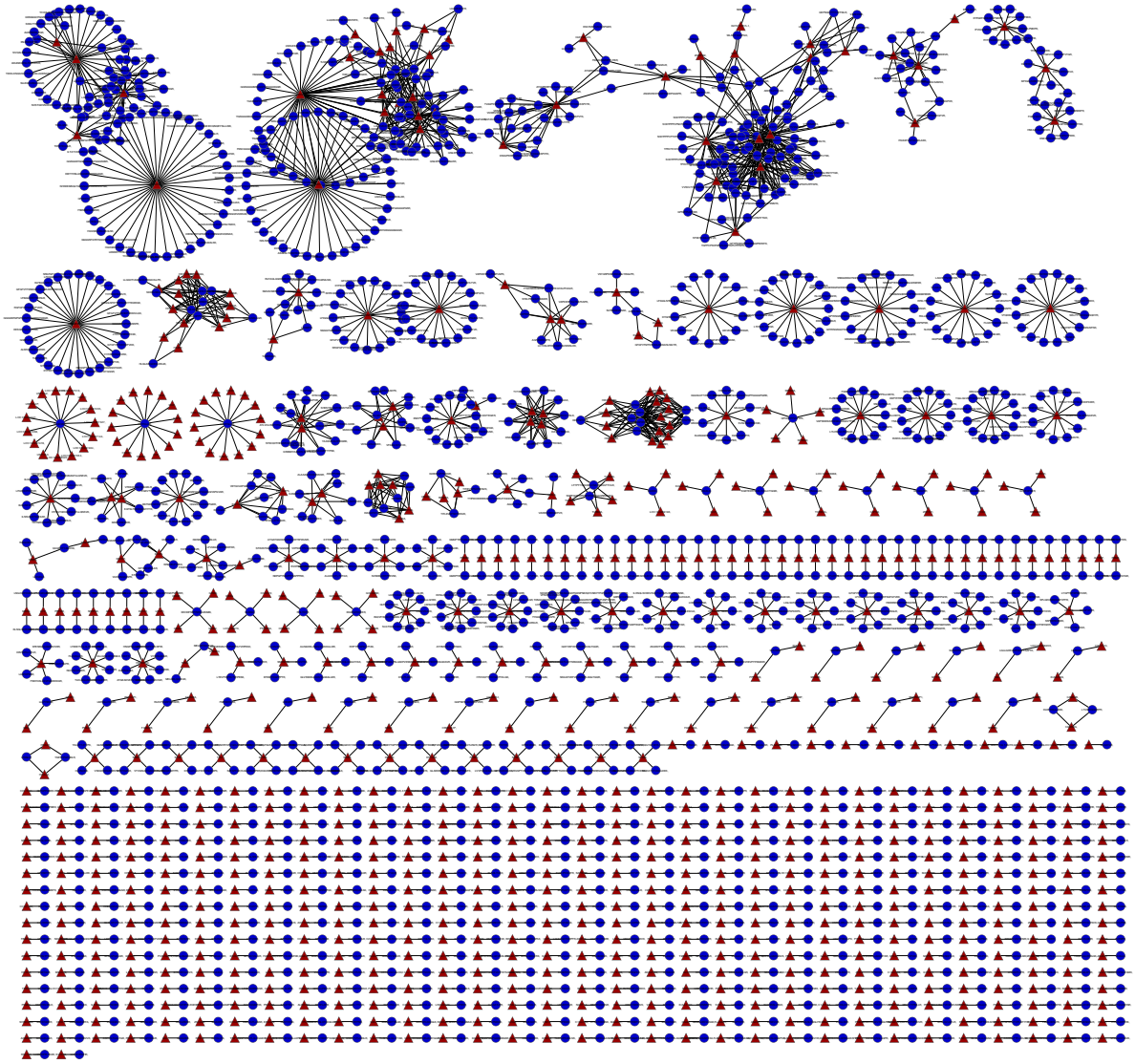


Figure 3.5: Peptide-protein graph constructed from an AP-MS experiment with RPAP3 as a bait. Proteins are drawn as red triangular vertices while peptides are drawn as blue round vertices.

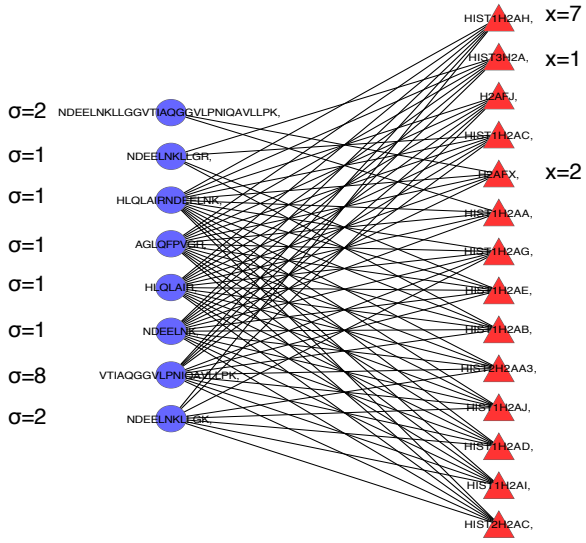


Figure 3.6: A connected component of the bipartite graph from the CDK9 dataset, where all peptides in the component are shared peptides that belong to histones. Our algorithm for MINIMUMPROTEINTYPES chooses only 3 proteins (HIST1H2AH, HIST3H2A and H2AFX) to be in the solution. Blue round vertices indicate peptides while proteins are drawn as red triangular vertices.

ular selection of proteins may not be very accurate due to the relatively small number of peptides. However, the existing approach of simply ignoring shared peptides within MS data would lead us to miss this instance entirely. On the other hand, using the naive approach of assigning each protein the average value of constituent peptide abundances would lead us to selecting all the histones in this component, resulting in poor quantitative data.

Such an effect is more prominent in the example shown in Figure 3.7. Here, the peptide abundances are taken from an AP-MS run with RPAP3 as the bait protein. While the peptides are shared by four distinct proteins, our algorithms for MINIMUMPROTEINTYPES and MINIMUMERRORSUM both picked HNRNP1 and HNRNP2 (heterogeneous ribonucleoprotein particles) with positive abundances. This is indeed a reasonable identification as the remaining candidate proteins are supported by only 1 or 2 constituent peptides.

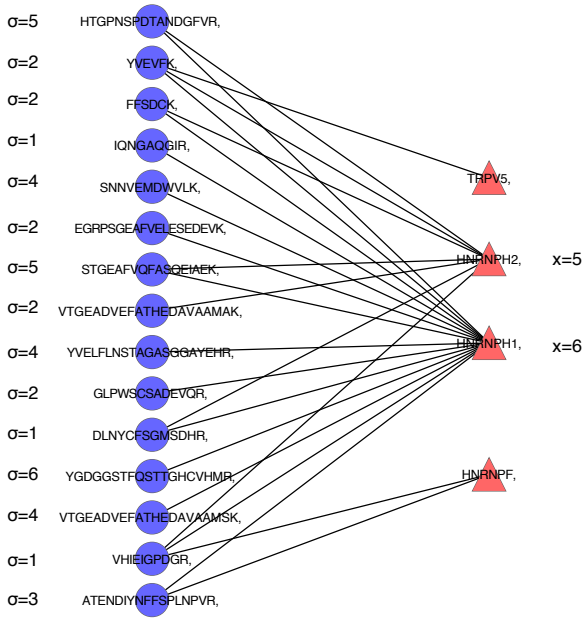


Figure 3.7: A connected component of the bipartite graph from the RPAP3 dataset. Our algorithms for MINIMUMPROTEINTYPES and MINIMUMERRORSUM both select HNRNPH1 and HNRNP2 in the solution while discarding the less supported proteins.

3.5 Discussions

In this chapter, we studied the protein quantification problem by first modelling the mass spectrometry data using mathematical programs. While the four problems formulated are all expressed as integer programs, we studied their hardness from combinatorial standpoints by reductions from SETCOVER and EXACTCOVER. Then, we devised approximation algorithms for the corresponding problems which also showed promising empirical performances when tested on synthetic data as well as biological data.

There remain various extensions to the models discussed here. For example, all four problems (MULTICOVER, MINIMUMPROTEINTYPES, MINIMUMUNIFORMERROR, MINIMUMERRORSUM) model the mass spectrometry data under the assumption that each peptide may occur in a protein at most once. Lifting this assumption can be achieved easily by modifying the adjacency matrix A so that $A_{i,j}$ is the number of times peptide s_i appears in protein w_j, and the resulting problems can be attacked using the same

approaches.

Moreover, one may wish to incorporate into our models the "detectability" $c(i)$ of each peptide[7], indicating the difficulty for correctly detecting a particular peptide. Recent studies have shown that peptide detectability can be estimated reliably [2, 102]. For example, Tang et al. proposed a machine learning approach to estimate the peptide detectability using its sequence length and its neighbouring regions in the parent protein sequence [133]. To incorporate this into our model of MINIMUMERRORSUM, we can minimize the objective function $\sum_i c(i) \cdot \epsilon_i$. The algorithm for the modified objective function need not change, as our randomized rounding scheme simply rounds the fractional solution from the LP relaxation.

In the case of MULTICOVER and MINIMUMPROTEINTYPES, we can integrate this notion of peptide detectability to each *protein* w_j by $p(j) = \sum_{(i,j)\in E} c(i)$. Then, given this penalty function for each protein, we can ask to minimize the total cost of chosen proteins: $\sum_j p(j)x_j$ for MULTICOVER, and $\sum_j p(j)\chi(x_j)$ for MINIMUMPROTEINTYPES. The resulting problems can be solved under the same framework, using algorithms for the corresponding *weighted* set cover problems.

Another factor that can be incorporated into our model is the set of peptides that were *not* observed in the MS data. For every unobserved peptide s_k, we can introduce its absence to our model by adding s_k to the bipartite graph with the abundance value $\sigma_k = 0$. As these are additional constraints to the integer programs, they would help us obtain a more accurate solutions albeit with slower running time. However, as discussed in Section 3.4.1, the coverage of a protein by the identified peptides is typically low due to low sensitivity of mass spectrometers. As a result, we cannot simply interpret undetected peptides as zero abundance, and the addition of these additional constraints should be avoided until the sensitivity of MS data improves.

[7] The detectability $c(i)$ for peptide s_i can be set higher if it is less likely to observe the correct abundance of s_i, say.

3.6 Bibliographic Notes

The set cover problem and its generalization into the multicover problem have been studied extensively in the approximation algorithms community. Lund and Yannakakis [101] showed that SETCOVER cannot be approximated within a factor of $o(\log n)$ unless $\mathsf{P} = \mathsf{NP}$. Moreover, Feige [48] showed that, unless $\mathsf{NP} \subset DTIME(n^{O(\log\log n)})$, there is no $(1-o(1))\ln n$ approximation. Thus the approximability of the set cover problem is essentially resolved if $\mathsf{P} \neq \mathsf{NP}$. When the input set system admits a special underlying structure, the lower bound on the approximation ratio for the general set cover (and multicover) can be improved: for example, Brönnimann and Goodrich [21] showed an improved $O(1)$ approximation ratio when the elements are points on the plane while the sets correspond to disks on the plane. Further, there are various algorithms specialized for different restricted cases of set systems [25, 47] that may provide better approximation ratios. Unfortunately, however, the set systems constructed from our PPI dataset do not admit these underlying structures.

The results discussed in this chapter are to be submitted [89].

Chapter 4

Direct PPI Networks from AP-MS Data

With the help of high-throughput experimental techniques, a large amount of PPI data has recently become available, providing us with a rough picture of how proteins interact in biological systems. However, interaction data from high-throughput experiments often suffers from relatively high error rates and protocol-specific biases. Therefore, inferring the physical PPI network from high-throughput data remains a challenge in systems biology.

As discussed in Chapter 1, affinity purification followed by mass spectrometry identification (AP-MS) is an increasingly popular approach to observe protein-protein interactions (PPI) *in vivo*. One drawback of AP-MS, however, is that it is prone to detecting indirect interactions mixed with direct physical interactions. Therefore, the ability to distinguish direct interactions from indirect ones is of much interest. In this chapter, we propose a simple probabilistic model for the interactions captured by AP-MS experiments, under which the problem of separating direct interactions from indirect ones is formulated. Then, given idealized quantitative AP-MS data, we propose an approach to identify the most likely set of direct interactions that produced the observed data.

This chapter is organized as follows. In Section 4.1, we give a brief overview of related research around the AP-MS technology. Then, Section 4.2 describes the mathematical

modelling of an AP-MS experiment, and formulates an optimization problem. In Section 4.3, we describe the overall algorithm, which is based on a collection of graph theoretic approaches that succeed at inferring a large fraction of the network nearly exactly, followed by a genetic algorithm that infers the remainder of the network. Finally, in Section 4.4, we test the accuracy of our method using both biological and simulated PPI networks. Here, we apply our algorithm to the prediction of direct interactions based on a large set of AP-MS PPI data in yeast [93].

4.1 Background

In an AP-MS experiment, a protein of interest (the *bait*) is first tagged and expressed *in vivo*. The bait is then immuno-precipitated (IP), together with all of its interacting partners (the *preys*), and finally, preys are identified using mass spectrometry. Like Y2H and other high-throughput experimental methods, however, AP-MS suffers from experimental noise. A number of approaches have been proposed to separate true interactions from false-positives. These approaches mostly focus on reducing false-positives due to protein misidentification from MS data [63, 107, 150], on detecting contaminants [19], or a combination of both [26, 33, 35, 111, 122, 123]. These methods often make use of the *guilty-by-association* principle, and quantify the confidence level of an interaction by considering alternative paths between two protein molecules. In this context, a true interaction between bait b and prey p is considered a true positive if, at some point in the set of cells considered, there exists a complex that contains both b and p. One concern with this line of reasoning is that, as the sensitivity of the AP-MS methods improves (and thus the stability of the complexes that can be detected decreases), the transience of detectable interactions will increase to a point where, eventually, every protein may be shown to marginally interact with every other protein.

A key property of AP-MS approaches is that a significant number of the co-purified prey proteins are in fact *indirect* interaction partners of the bait protein, in the sense that

they do not interact physically and directly with the bait, but interact with it through a chain of physical interactions involving other proteins in the complex. Therefore, it is critical, when interpreting AP-MS-derived PPI networks, to correctly interpret the meaning of the term "interaction". Although not designed to identify physical interactions, the increasing amount of AP-MS data available justifies the need for separating direct physical interactions from indirect ones. This is the problem we consider in this chapter:

> "*Given the results of a set of AP-MS experiments, filtered for protein misidentifications and contaminants, how can we distinguish direct (physical) from indirect interactions?*"

Note that since the false-positive filtering methods listed above consider indirect interactions as true-positives, they cannot be used to address this problem. Gordân et al. [65] study the related problem of distinguishing direct vs. indirect interactions between transcription factors (TF) and DNA. While the objective of their study is similar to ours, their method makes use of information specific to TF-DNA interactions (e.g. TF binding data, motifs from protein binding microarrays), and thus is not immediately applicable to the problem on general PPI networks. In fact, to our knowledge, no existing approach seems directly applicable.

4.2 Mathematical Modelling and Problem Formulation

Throughout this chapter, we make the assumption that appropriate methods have been used to reduce as much as possible protein misidentifications and contaminants, in such a way that all interactions detected are either direct or indirect interactions. Our task is to separate the former from the latter. To save from confusion, therefore, we note that false positives (resp. false negatives) henceforth refer to falsely detected (resp. undetected) *direct* interactions inferred by our algorithm.

4.2.1 A Probabilistic Model for AP-MS Data

We first describe a simple model of the AP-MS PPI data that shall be used throughout this chapter. Though admittedly rather simplistic, our model has the benefit of allowing the formulation of a well-defined computational problem.

Let $G_{direct} = (V, E_{direct})$ be the graph of direct (physical) interactions over the set V of proteins considered, and let $N(b) = \{p \in V : (b, p) \in E_{direct}\}$ be the set of direct interaction partners of protein b. We model the physical process through which PPIs are identified in an AP-MS experiment as follows. If a bait protein b is in contact with a direct interaction partner $p \in N(b)$, the affinity purification (AP) process (see Section 1.1.2 in Chapter 1) on b will pull down p, which will then be identified through mass spectrometry. In addition, if p interacts with $p' \in N(p)$ at the same time as it interacts with b, protein p' may also be pulled down, even though the two proteins b and p' only *indirectly* interact. In general, any protein x that is connected to b by a series of simultaneous direct interactions may be pulled down by b. As a result, all interaction partners of b (direct or indirect) will be identified together. Figure 4.1 depicts an example of this effect.

In order to distinguish direct physical interactions from indirect ones, the availability of *quantitative* AP-MS data is helpful. Although quantitative AP-MS remains in its infancy, prey abundance can be estimated fairly accurately using various approaches, e.g. the peptide count [55], spectral count [100], sequence coverage [51], protein abundance index [73], and our own approach of quantifying absolute protein abundance discussed in Chapter 3. Combined with the increasing accuracy and sensitivity of mass spectrometers, these methods are becoming more reliable. Throughout the discussions in this chapter, we thus assume that this quantitative data is available to us.

To model the quantitative AP-MS data, we first consider the strength of a direct interaction, which can be measured by the energy required to break it. To give an example, suppose the PPI network consists of only two proteins, b and p. Then, let $A(b, p)$ denote

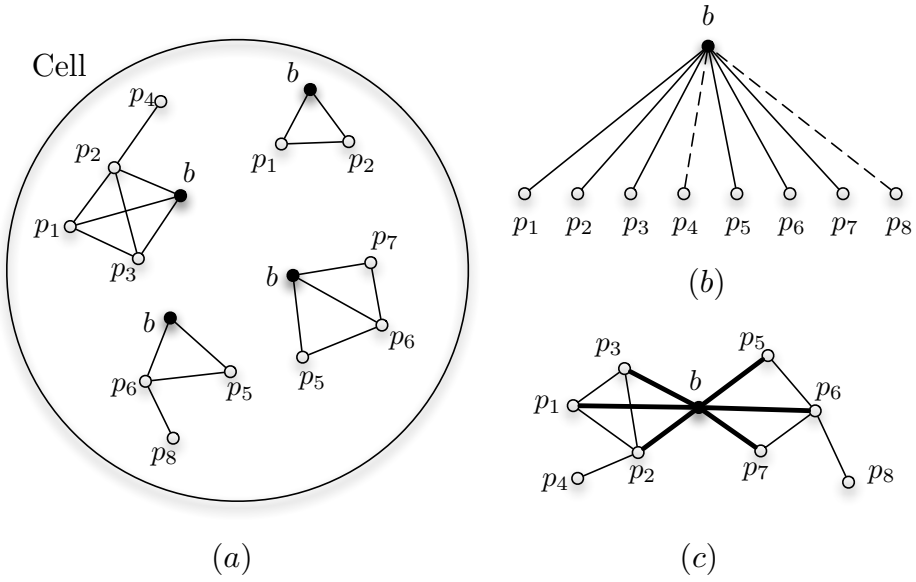


Figure 4.1: A schematic for indirect interactions in AP-MS data. (a) In a cell, multiple copies of a bait protein b are expressed, and interact (directly or indirectly) with other prey proteins $p_1,\ldots,p_8$. (b) After the pull-down on the bait b, MS detects all prey proteins, including indirect interaction partners p_4 and p_8. (c) The direct interaction network should, however, contain only edges between direct interaction partners.

the *observed* abundance of a prey protein p obtained by the bait b, and let $\chi(b,p)$ denote the true abundance of interactions between b and p. Since protein interactions may be disrupted by the AP process, we expect that $A(b,p)$ is correlated with the strength of the interaction between b and p, as well as the true abundance $\chi(b,p)$. We model the strength of an interaction using the probability $\hat{p}(b,p)$ that the interaction between b and p survives the purification process, and assume that the interaction breaks with probability $1-\hat{p}(b,p)$. Then, the observed abundance of protein p obtained from the pull-down on b would be

$$A(b,p) \propto \chi(b,p)\cdot\hat{p}(b,p).$$

As another example, consider now a system of three proteins b, p_1, p_2, where (b,p_1) and (p_1,p_2) form direct interactions, but b and p_2 do not interact directly. As before, let $\chi(b,p_1,p_2)$ denote the true abundance of protein interactions that occur *simultaneously*, i.e., number of times both (b,p_1) and (p_1,p_2) occur together. Then, $A(b,p_2) \propto$

$\chi(b,p_1,p_2)\cdot\hat{p}(b,p_1)\cdot\hat{p}(p_1,p_2)$. In general, the observed amount of protein p obtained upon pull-down of b will be proportional to the probability that b and p remain connected after each edge $(u,v)\in E(G_{direct})$ is broken with probability $\hat{p}(u,v)$. Our goal is then to infer G_{direct} from the set of observed abundances $A(u,v)$. In this paper, we make the following simplifying assumptions:

1. All direct interactions $(u,v)\in E_{direct}$ survive with the *uniform probability* $\hat{p}$, and fail independently with probability $1-\hat{p}$.

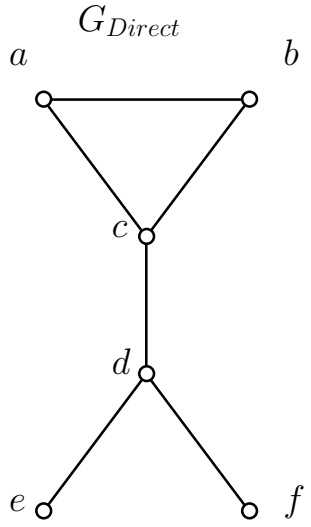
2. All possible direct interactions take place at the *same time*, irrespective of the presence of other interactions, and with the *same frequency*.

Albeit rather strong, these assumptions provide a useful starting point for separating direct interactions from indirect ones (see Section 4.5 for possible relaxation of these assumptions). Despite its simplicity, our mathematical modelling of AP-MS does fit existing biological data reasonably well (see Section 4.4.5). We note that Asthana et al. [5] have proposed an approach similar to our probabilistic graph model to identify novel members of known protein complexes. However, their goal is not to identify indirect interactions, so their approach is not applicable to our problem.

4.2.2 Problem Formulation

We are now ready to formulate the algorithmic problem addressed in this chapter. We henceforth consider the (unknown) direct interaction network G_{direct} as a probabilistic graph, where each edge in G_{direct} survives the AP-MS process with probability $\hat{p}$, and fails otherwise. Let $\mathscr{G}$ denote a random graph obtained from G_{direct} by removing edges in E_{direct} independently with probability $1-\hat{p}$. Then, define $P_{G_{direct}}(u,v)$ to be the probability that vertices u and v remain connected (directly or indirectly) in $\mathscr{G}$:

$$P_{G_{direct}}(u,v)=\Pr[\text{ there exists at least one path from } u \text{ to } v \text{ in } \mathscr{G}\ ].$$



(a)

$P_{G_{Direct}}$

	a	b	c	d	e	f
a	1	$5/8$	$5/8$	$5/16$	$5/32$	$5/32$
b		1	$5/8$	$5/16$	$5/32$	$5/32$
c			1	$1/2$	$1/4$	$1/4$
d				1	$1/2$	$1/2$
e					1	$1/4$
f						1

(b)

Figure 4.2: An example of (a) a direct interaction network G_{Direct} and (b) its connectivity matrix $P_{G_{Direct}}$ calculated with $\hat{p} = \frac{1}{2}$. Assuming each edge of G_{Direct} survives with probability $\hat{p}$, the probability of connectivity between each pair of protein can be estimated via sampling of the probabilistic network.

We call $P_{G_{direct}}$ the *connectivity matrix* of G_{direct}. See Figure 4.2 for an example of a direct interaction network and its connectivity matrix. In Figure 4.2(b), the entry $P_{G_{direct}}(b,d)$ is assigned $\frac{5}{16}$, because, out of all 64 subgraphs of G_{direct}, only 20 of those maintain the vertices b and d in the same component. In general, $P_{G_{direct}}(u,v)$ can be estimated from G_{direct} by straight-forward Monte Carlo sampling. However, its exact computation (known as the *two-terminal network reliability* problem [34, 138]) is #P-complete [138].

A set of AP-MS experiments where all proteins have been tagged and used as baits yields an approximation of $A(x,y)$ for all pairs of proteins (x,y), which can be transformed into an estimate $M(x,y)$ of $P_{G_{direct}}(x,y)$ through an appropriate normalization. We are thus interested in inferring G_{direct} from M:

EXACT DIRECT INTERACTION GRAPH FROM CONNECTIVITY MATRIX (E-DIGCOM)

Given: An $n \times n$ connectivity matrix M

Find: A graph $G = (V,E)$ such that $P_G(u,v) = M(u,v)$ for each $u,v \in V$.

In a more realistic setting, the connectivity matrix P_G would not be observed precisely, and the E-DIGCOM problem may not admit a solution. We are thus interested in an approximate, optimization version of the problem:

APPROXIMATE DIRECT INTERACTION GRAPH FROM CONNECTIVITY MATRIX (A-DIGCOM)
Given: An $n \times n$ connectivity matrix M and a tolerance level $0 \leq \delta \leq 1$
Find: A graph $G = (V, E)$ such that the number of pairs $(u, v) \in V \times V$ such that $|P_G(u, v) - M(u, v)| \leq \delta$ is maximized.

Note that although the complexity of the DIGCOM problems are currently unknown, the reverse problem of finding the connectivity matrix M given an input graph G (network reliability problem) is well studied. Furthermore, related network design problems have been studied extensively in the computer networking community. For example, the *reliable network design* problem is to choose a minimal set of edges over a set a nodes so that the resulting network has at least the prescribed all-pairs terminal reliability; various algorithms including branch-and-bound heuristics [76] and genetic algorithms [38, 39] have been proposed.

4.3 Algorithm for A-DIGCOM

Our algorithm for the A-DIGCOM problem has three main phases outlined as below.

PHASE I. We start by identifying, based on the connectivity matrix M, vertices from G_{direct} with low degree, together with edges incident to them. As most PPI networks exhibit the properties of scale-free networks [10], this resolves the edges incident to a significant portion of the vertices ($\sim 75\%$, in our networks).

PHASE II. At the other end of the spectrum, G_{direct} contains dense clusters that often correspond to protein complexes. We use a quasi-clique finding heuristic to identify such dense clusters from the connectivity matrix M.

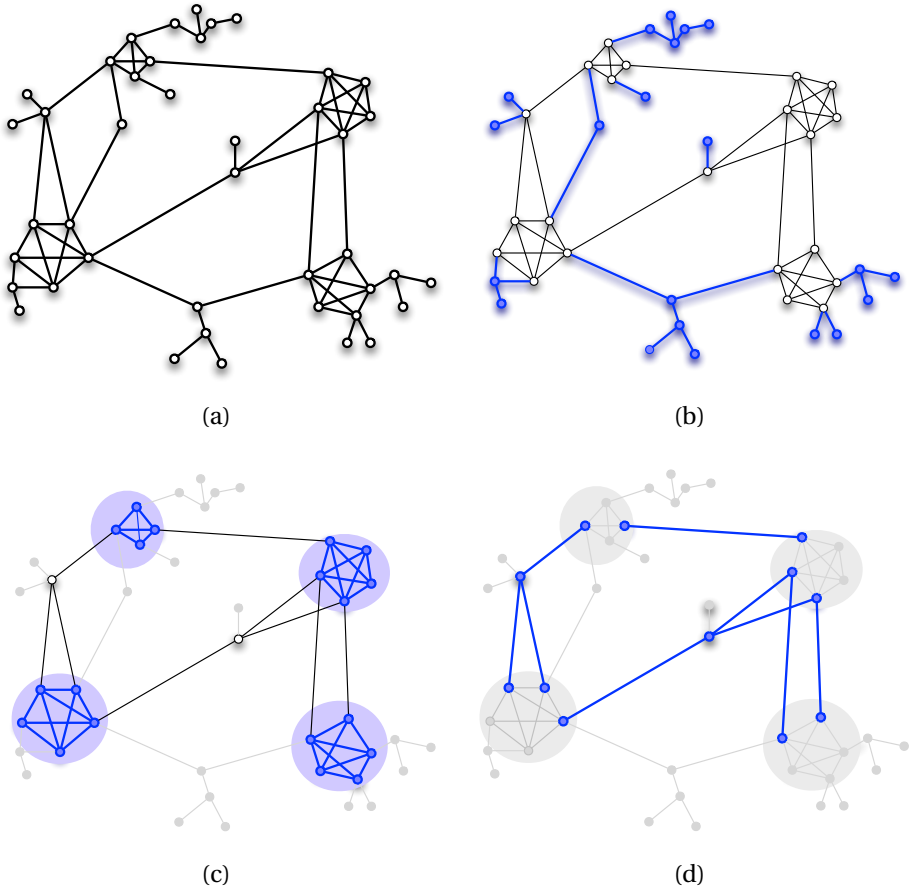


Figure 4.3: The outcome of our direct PPI detection algorithm after each phase. (a) The original solution space; (b) After detecting weakly connected regions; (c) After detecting dense clusters; (d) The genetic algorithm detects the remaining interconnecting regions.

PHASE III. To infer the remainder of the network, we use a novel *genetic algorithm* (see Section 2.4). This highly customized genetic algorithm makes use of the findings from the previous two steps in order to dramatically reduce the dimension of the problem space, and to guide the mating process between parent candidates to create good offspring solutions.

Figure 4.3 gives an example of the output after each phase of our algorithm. In what follows, we describe each phase of the algorithm in detail.

4.3.1 Identification of Weakly Connected Regions

I-a. Finding cut edges

A *cut edge* in a graph G is an edge (u,v) whose removal would result in u and v belonging to two distinct connected components (e.g. edge (c,d) in Figure 4.2(a)). The following theorem allows the identification of all cut edges based on the connectivity matrix P_G.

Theorem 4.1. *A pair of vertices u and v from V forms a cut edge in G if and only if the following two conditions hold:*

(i) $P_G(u,v) = \hat{p}$

(ii) *V can be partitioned into $V = V_u \cup V_v$, where $V_u = \{x \in V : P_G(x,u) \geq P_G(x,v)\}$ and $V_v = \{x \in V : P_G(x,u) < P_G(x,v)\}$, such that $\forall s \in V_u$ and $\forall t \in V_v$, $P_G(s,t) = P_G(s,u) \cdot \hat{p} \cdot P_G(v,t)$.*

Proof. Necessity is trivial. For sufficiency, suppose the conditions (i) and (ii) hold, and (u,v) is not a cut edge. Then, to keep the graph connected, there must be an edge $(s,t) \neq (u,v)$ joining V_u and V_v. Since (s,t) is an edge, $p(s,t) \geq \hat{p}$. However, by assumption, we have $p(s,t) = p(s,u) \cdot \hat{p} \cdot p(v,t) < \hat{p}$, which is a contradiction. □

The above theorem immediately provides an efficient algorithm to test whether a pair of vertices forms a cut edge in time $O(|V|^2)$: recursively find edges satisfying conditions in Theorem 4.1. Observe that removing a cut edge (u,v) allows us to decompose the graph into two connected components (subgraphs induced by V_u and V_v, respectively), and the probability of connectivity between every pair of vertices in V_u (V_v, resp.) remains the same after removing (u,v). Therefore, the submatrices that correspond to V_u and V_v can be treated as independent subproblems, and one can recursively detect cut edges in the remaining subproblems.

Note that a special case of cut edges is an edge whose endpoint is a degree-1 vertex. Because removing a cut edge does not change the connectivity between any two nodes

on the same side of the cut, this allows us to repeatedly identify and *remove* degree-1 vertices, i.e., the corresponding rows and columns in the input matrix. Hence, if the entire graph is assumed to be a tree, this algorithm can reconstruct the graph completely. On the other hand, PPI networks tend to contain many cut edges due to their sparsity. Consequently, this simple characterization for cut edges allows a significant simplification of our problem by identifying all cut edges.

Theorem 4.2. *If G is a tree,* E-DIGCOM *can be solved efficiently.* □

I-b. Finding degree-2 vertices

We now consider the problem of identifying degree-2 vertices from the connectivity matrix M. After degree-1 vertices, which are identified in the previous step, they constitute the next most frequent vertices in the biological networks we studied. While we do not have a full characterization of these vertices, the following theorem gives a set of necessary conditions that lead to a heuristic to predict degree-2 vertices.

Theorem 4.3. *Let s be a degree-2 vertex in G such that $N(s)=\{u,v\}$. Then, the following four conditions must hold.*
(i) **Low connectivity:** *for each $t \in V$, $P_G(s,t) < 2\hat{p} - \hat{p}^2$.*
(ii) **Symmetry:** $P_G(s,u) = P_G(s,v)$.
(iii) **Neighbourhood:** *for each $t \in V - \{s,u,v\}$, $P_G(s,t) < P_G(s,u)$.*
(iv) **"Triangle" inequality:** *for each $t \in V - \{s,u,v\}$, $P_G(s,t) < max\{\, P_G(u,t), P_G(v,t)\,\}$.*

Before we prove this, consider the following results on graph composition. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. Then the following hold.

Fact 4.4. (Series composition) *Suppose $V_1 \cap V_2 = \{c\}$, and a new graph G is constructed by joining G_1 and G_2 at c. Then, for any $s \in V_1 - \{c\}$ and for any $t \in V_2 - \{c\}$, $P_G(s,t) = P_{G_1}(s,c) \cdot P_{G_2}(c,t)$.*

Proof. Since c is a cut vertex, $P_G(s,c) = P_{G_1}(s,c)$, and $P_G(c,t) = P_{G_2}(c,t)$. For the vertices s and t to be connected, all of s, c, t must be in the same connected component. Because every edge removal is independent, we have $P_G(s,t) = P_{G_1}(s,c) \cdot P_{G_2}(c,t)$. □

Fact 4.5. (Parallel composition) *Suppose $V_1 \cap V_2 = \{s,t\}$, and a new graph G is constructed by joining G_1 and G_2 at s and t (possibly leading to parallel edges between s and t). Then,* $P_G(s,t) = P_{G_1}(s,t) + P_{G_2}(s,t) - P_{G_1}(s,t) \cdot P_{G_2}(s,t)$.

Proof. The event that s and t are *not* connected is when s and t are disconnected in both G_1 and G_2. Thus $1 - P_G(s,t) = (1 - P_{G_1}(s,t)) \cdot (1 - P_{G_2}(s,t)) = 1 - P_{G_1}(s,t) - P_{G_2}(s,t) + P_{G_1}(s,t) \cdot P_{G_2}(s,t)$. □

Now we are ready to prove Theorem 4.3.

Proof. We prove each condition separately.

Condition (i) Low connectivity. Since s has degree 2, it becomes disconnected from the rest of the graph with probability $(1-\hat{p})^2$. Thus, $P_G(s,t) \leq 1 - (1-\hat{p})^2 = 2\hat{p} - \hat{p}^2$. The equality can only hold if $P_G(s,u) = P_G(s,v) = 1$, which is impossible.

Condition (ii) Symmetry. Let $P_{G-\{(s,u)\}}(s,u)$ denote the connectivity of s and u when edge (s,u) is removed from E. Then, we have:

$$
\begin{aligned}
P_G(s,u) &= \hat{p} + P_{G-\{(s,u)\}}(s,u) - \hat{p} \cdot P_{G-\{(s,u)\}}(s,u) && \text{(by Fact 4.5)} \\
&= \hat{p} + \hat{p} \cdot P_{G-\{(s,u),(s,v)\}}(v,u) - \hat{p}^2 \cdot P_{G-\{(s,u),(s,v)\}}(v,u) && \text{(by Fact 4.4)} \\
&= \hat{p} + \hat{p} \cdot P_{G-\{(s,u),(s,v)\}}(u,v) - \hat{p}^2 \cdot P_{G-\{(s,u),(s,v)\}}(u,v) \\
&= \hat{p} + P_{G-\{(s,v)\}}(s,v) - \hat{p} \cdot P_{G-\{(s,v)\}}(s,v) && \text{(by Fact 4.4)} \\
&= P_G(s,v) && \text{(by Fact 4.5)}
\end{aligned}
$$

Condition (iii) Neighbourhood Next, we show that for any $t \in V - \{s,u,v\}$, $P_G(s,t) < P_G(s,u)$. For a subgraph $H \subseteq G$, let $p(H)$ denote the probability of observing H from a probabilistic graph G. We can write this probability as

$$p(H) = \hat{p}^{|E(H)|} \cdot (1-\hat{p})^{|E(G)|-|E(H)|}$$

Thus, for any two subgraphs $H_i, H_j \subseteq G$, such that $|E(H_i)| = |E(H_j)|$, $p(H_i) = p(H_j)$. Let $\mathbf{H}(s,t)$ be the set of subgraphs of G where s and t are connected. We can then write the probabilities $P_G(s,t)$ and $P_G(s,u)$ as the sum of the probabilities of these subgraphs.

$$P_G(s,t) = \sum_{H_i \in \mathbf{H}(s,t)} p(H_i)$$
$$P_G(s,u) = \sum_{H_j \in \mathbf{H}(s,u)} p(H_j)$$

Observe that these two sets of subgraphs may overlap:

$$\mathbf{H}(s,t) = \mathbf{H}(s,t,u) \cup \mathbf{H}(s,t,\bar{u})$$
$$\mathbf{H}(s,u) = \mathbf{H}(s,t,u) \cup \mathbf{H}(s,\bar{t},u)$$

where $\mathbf{H}(s,t,u)$ is the set of subgraphs such that s,t, and u are all connected, while $\mathbf{H}(s,t,\bar{u})$ is the set of subgraphs where s,t belong to the same connected component that doesn't contain u. Thus, we now focus on $\mathbf{H}(s,t,\bar{u})$ and $\mathbf{H}(s,\bar{t},u)$. The subgraphs in $\mathbf{H}(s,\bar{t},u)$ can be partitioned as follows, depending on which component v belongs to:

(i) $\{s,u,v\},\{t\}$: Vertices s,u,v belong to the same component, and t belongs to another component.

(ii) $\{s,u\},\{v\},\{t\}$: Vertices s,u belong to the same component, and each of v and t belongs to distinct components.

(iii) $\{s,u\}$, $\{v,t\}$: Vertices s,u belong to the same component, and v,t belong to another component.

It is easy to see that the cases (i) and (ii) are nonempty, since $(s,u),(s,v) \in E(G)$.

Finally, consider the set of subgraphs in $\mathbf{H}(s,t,\bar{u})$. Let H be a subgraph in this set. Since s and u belong to distinct components, $(s,u) \notin E(H)$, and thus $(s,v) \in E(H)$ in order to have s and t connected. We then make the following operation on H to construct H': (1) remove (s,v) from H, and (2) insert (s,u). Then, H' has s and u in the same component, and v and t in another component. Therefore, H' belongs to Case (iii) of $\mathbf{H}(s,\bar{t},u)$. Furthermore, note that H and H' have the same number of edges. Therefore, there is a mapping from subgraphs in $\mathbf{H}(s,t,\bar{u})$ to subgraphs in Case (iii) of $\mathbf{H}(s,\bar{t},u)$ with an equal number of edges. Since the cases (i) and (ii) are nonempty, it follows that $P_G(s,t) < P_G(s,u)$.

Condition (iv) "Triangle" inequality. Given the probabilistic graph G, we partition the subgraphs of G into four cases depending on the existence of the two edges (s,u) and (s,v).

- $(s,u),(s,v) \in E$: occurs with probability $\hat{p}^2$.
- $(s,u) \in E, (s,v) \notin E$: occurs with probability $\hat{p}(1-\hat{p})$.
- $(s,u) \notin E, (s,v) \in E$: occurs with probability $\hat{p}(1-\hat{p})$.
- $(s,u),(s,v) \notin E$: occurs with probability $(1-\hat{p})^2$.

We can thus rewrite the probability $P_G(s,t)$ as follows.

$$P_G(s,t) = \hat{p}\cdot(1-\hat{p})\cdot P_{G-\{s\}}(u,t) + \hat{p}\cdot(1-\hat{p})\cdot P_{G-\{s\}}(v,t) + \hat{p}^2 \cdot P_{G-\{s\}}(u \text{ or } v,t) \tag{4.1}$$

where $P_{G-\{s\}}(u \text{ or } v,t)$ denotes the probability that u or v is connected to t in the graph $G-\{s\}$. We write $P_G(u,t)$ and $P_G(v,t)$ similarly:

$$P_G(u,t) = (1-\hat{p}^2)\cdot P_{G-\{s\}}(u,t) + \hat{p}^2 \cdot P_{G-\{s\}}(u \text{ or } v,t) \tag{4.2}$$

$$P_G(v,t) = (1-\hat{p}^2)\cdot P_{G-\{s\}}(v,t) + \hat{p}^2 \cdot P_{G-\{s\}}(u \text{ or } v,t) \tag{4.3}$$

Subtracting (4.1) from (4.2), and (4.1) from (4.3) gives:

$$P_G(u,t) - P_G(s,t) = (1-\hat{p}) \cdot P_{G-\{s\}}(u,t) - \hat{p}(1-\hat{p}) \cdot P_{G-\{s\}}(v,t)$$
$$P_G(v,t) - P_G(s,t) = (1-\hat{p}) \cdot P_{G-\{s\}}(v,t) - \hat{p}(1-\hat{p}) \cdot P_{G-\{s\}}(u,t)$$

If $P_{G-\{S\}}(u,t) > \hat{p} \cdot P_{G-\{S\}}(v,t)$, it follows that $P_G(u,t) > P_G(s,t)$. Otherwise, $P_{G-\{s\}}(u,t) \leq \hat{p} \cdot P_{G-\{s\}}(v,t)$ implies $\hat{p} \cdot P_{G-\{s\}}(v,t) < P_{G-\{s\}}(v,t)$, and it follows that $P_G(v,t) > P_G(s,t)$. This completes the proof. □

These necessary conditions allow us to rule out vertices that cannot have degree greater than 2. As a result, Theorem 4.3 provides a heuristic (Algorithm 4.3.1) for predicting the set of degree 2 vertices as well as the edges incident to them. It is easy to see that Algorithm 4.3.1 runs in time $O(|V|^2)$, and in practice, our studies have shown that vertices satisfying these conditions while having degree higher than two are rare (see Table 4.1).

Unlike the case of degree-1 vertices, we cannot simply remove degree-2 vertices without affecting the remaining entries in the matrix. Therefore, as shown in Algorithm 4.3.1, degree-2 vertices and their incident edges are simply marked as such in the solution, but are not removed from the input matrix.

Algorithm 4.3.1: Prediction of degree 2 vertices

Input: Probability matrix M and vertex set V
Output: Degree 2 vertices V^2 and their incident edges E^2

foreach *vertex* $s \in V$ **do**
  if s *satisfies conditions in Theorem 4.3* **then**
    $V^2 \leftarrow V^2 \cup \{s\}$;
    $u, v \leftarrow$ the two vertices with $M_{s,u} = M_{s,v} = \max\{M_{s,t} : \forall t \in V - \{s\}\}$;
    $E^2 \leftarrow E^2 \cup \{(s,u),(s,v)\}$;
  end
end
Return V^2 and E^2

4.3.2 Identification of Densely Connected Regions

We now turn to the problem of finding densely connected regions in the network. These regions may correspond to protein complexes, where tagging any one of the members of the complex results in the identification of all other members with high probability. While correctly predicting the physical interactions within each complex is a difficult task, separating these dense regions from the remainder of the network is key to improving the accuracy of the genetic algorithm (Phase III).

In order to build an algorithm for detecting dense regions, we consider the connectivity between nodes in an n-clique K_n, which we denote as CliqueConn(n). First, let us recall a classical result in graph enumeration.

Lemma 4.6. [62, 67] *Let γ_n denote the number of connected simple graphs over n labeled vertices. We can count this number by the recurrence:*

$$\gamma_n = \begin{cases} 1 & \text{if } n = 1 \\ 2^{\binom{n}{2}} - \frac{1}{n}\sum_{i=1}^{n-1}\left(i\binom{n}{i}\cdot 2^{\binom{n-i}{2}}\cdot\gamma_i\right) & \text{otherwise} \end{cases}$$

Using this combinatorial result, we can construct the formula of CliqueConn(n) when $\hat{p} = \frac{1}{2}$, which is the value we used for our analyses. The formula for other values of $\hat{p}$ can be formulated using a similar proof.

Lemma 4.7. *Let G be a complete, probabilistic graph on n vertices with $\hat{p} = \frac{1}{2}$. Then, for any two vertices in G, the probability that they are connected is:*

$$CliqueConn(n) = \sum_{i=2}^{n} \frac{\binom{n}{i-2}\gamma_i}{2^{\binom{n}{2}-\binom{n-i}{2}}}$$

Proof. We write the probability as follows.

$$CliqueConn(n) = \frac{\sigma_2 + \sigma_3 + \ldots + \sigma_n}{2^{\binom{n}{2}}}$$

Algorithm 4.3.2: Detecting dense clusters in the network

Input: Probability matrix M, and minimum cluster size k
Output: Set of possible k-cliques in G_{direct}.

$t \leftarrow \text{CliqueConn}(k)$
$G_t \leftarrow (V, E_t)$, where $E_t = \{(u,v) : u, v \in V \text{ and } M(u,v) \geq t\}$
foreach *connected component* $S \subseteq V$ **do**
  Find cliques of size k in S
end
Return cliques discovered.

where σ_i denotes the number of subgraphs of K_n with i vertices, where the two given vertices u and v are connected. To compute σ_i, there are $\binom{n}{i-2}$ ways to pick $i-2$ vertices (in addition to u and v) for the connected components, and γ_i ways to keep them all connected, and finally, $2^{\binom{n-i}{2}}$ ways to assign edges among the remaining vertices. So we have

$$\sigma_i = \binom{n}{i-2} \cdot \gamma_i \cdot 2^{\binom{n-i}{2}},$$

and finally, we have

$$CliqueConn(n) = \sum_{i=2}^{n} \frac{\binom{n}{i-2}\gamma_i}{2^{\binom{n}{2}-\binom{n-i}{2}}} \qquad \square$$

Lemma 4.7 gives rise to a heuristic (see Algorithm 4.3.2) that finds a set of dense regions covering the network. While the algorithm guarantees to identify all cliques of size $k' \geq k$ contained within G_{direct}, sets of vertices that do not form a k-clique may also be reported, provided that they are sufficiently connected among themselves, possibly via vertices outside the dense cluster. However, for sufficiently large values of k, we found this to be a very rare occurrence. While finding cliques in a graph is a computationally intensive task in general, the construction of G_t (in Algorithm 4.3.2) for large values of k creates few small connected components and leaves the remaining vertices isolated. Therefore, in practice, Algorithm 4.3.2 can be implemented to run in a reasonable amount of time (see Table 4.5(ii)).

Based on the connectivity matrix M, our algorithm identifies (possibly overlapping)

clusters of proteins of size at least k such that, for every pair u, v in each cluster, $M(u,v) \geq t_k$ for some threshold t_k. For appropriately chosen values of k and t_k, the discovered clusters correspond to cliques in G_{direct} with high accuracy (see Section 4.4.6).

The dense regions discovered at this phase provide us with (1) the set of edges within each dense region; and (2) sparse cuts between disjoint dense regions. The edge set within each cluster will be used for initial candidates in the genetic algorithm, whereas the cuts defined by the clusters will be used as crossover points during the crossover operation in the genetic algorithm.

4.3.3 Cut-based Genetic Algorithm

To predict the remaining section of the network, we use a customized genetic algorithm that aims at finding an optimal solution to the A-DIGCOM problem. We first devise a solution to a generalization of the A-DIGCOM problem, and then show how the results of Phase I and Phase II of the algorithm are used to improve the performance.

Genetic algorithms have been shown to be an effective family of heuristics for a wide variety of optimization problems [64], including network design under connectivity constraints [38, 39]. A genetic algorithm models a set of candidate solutions as individuals of a population. From this population, pairs of promising candidate solutions are mated, and their offspring solutions inherit properties of the parents with some random mutations. Over generations, this process of natural selection improves the fitness of the population.

The A-DIGCOM problem is a hard optimization problem, because (i) the size of the search space is huge – $2^{\binom{n}{2}}$ for a graph of size n, and (ii) there is no known polynomial-time algorithm to evaluate a proposed candidate solution (i.e. compute P_G from G). For these reasons, a straight-forward genetic algorithm implementation failed to produce satisfactory results (data not shown). Instead, we use a more sophisticated approach by making use of the results obtained in previous sections in order to reduce the search

space and to guide the mating operations for more effective search.

The genetic algorithm aims to solve a generalization of A-DIGCOM. First, we allow each edge (u,v) in the network to survive with a non-uniform probability $\hat{p}(u,v)$, instead of one constant probability $\hat{p}$ over all edges. Secondly, we assume that we are given two sets of edges E_{YES} and E_{NO} that indicate the set of edges that are guaranteed to be in the solution, and guaranteed not to be in the solution, respectively. This will later allow us to factor in the outcomes of the previous sections. Therefore, the edges whose presence remains to be determined are $E_{MAYBE} = \overline{E_{YES} \cup E_{NO}}$.

Encoding of candidate solutions. To represent a candidate solution, we first create a hash table that maps each putative edge in E_{MAYBE} to an integer. Each candidate is then encoded as a list of integers (edges). Edges in E_{YES}, which are part of all solutions, are not explicitly listed, in order to save space. Since the networks we consider are sparse ($|E| = O(|V|)$), such an encoding technique significantly reduces the space requirements.

Initial population. The initial population of candidates is generated using a preferential attachment model [10] using the following observations: (i) The average connectivity of vertex u, $\text{avgCon}(u) = \frac{1}{|V|}\sum_{v \in V - \{u\}} M(u,v)$ is strongly positively correlated with the degree of u in G_{direct}; (ii) the age of a vertex, measured by when the vertex was introduced to the graph, is positively correlated with the degree of the vertex. Therefore, during the generation of each candidate, we choose the next vertex to be added with probability proportional to its average connectivity. This results in a candidate solution where the degree of most vertices is likely to be close to their true degree in G_{direct}. Furthermore, in order to create candidates that are clustered similarly to the true direct interaction graph, we include the set of edges predicted by Algorithm 4.3.2 to each initial candidate.

Fitness function. The fitness of a candidate solution G, fitness(G) is obtained by first estimating the probability matrix P_G using 500 Monte Carlo samples. To compute these samples, we break each edge independently with probability $\hat{p}$, and count the number of times each pair of vertices remain connected. Given 500 samples, we then count the number of vertex pairs (u,v) whose estimated connectivity $P_G(u,v)$ is within the tolerance level δ, i.e., $M(u,v)\pm\delta$ (see Section 4.4.2 on how to choose δ).

Crossover. The crossover operation needs to hybridize two parent candidates to produce offsprings preserving the good properties of the parents. This operation will be guided by a randomly chosen balanced cut $V = V_1 \cup V_2$. Let G_1 and G_2 denote the two parent networks and let $E_1(G_i)$ and $E_2(G_i)$ denote the edges of G_i such that both endpoints lie in V_1 and V_2, respectively. Furthermore, let $E_{1,2}(G_i)$ denote the edges of G_i that cross from V_1 and V_2. Mating G_1 and G_2 results in two children $G' = (V, E')$, and $G'' = (V, E'')$ such that:

$$E' = E_1(G_1) \cup E_2(G_2) \cup (E_{1,2}(G_1) \cap E_{1,2}(G_2))$$
$$E'' = E_2(G_1) \cup E_1(G_2) \cup (E_{1,2}(G_1) \cap E_{1,2}(G_2))$$

While choosing a random cut as the crossover point is a reasonable strategy to construct a new pair of offsprings, our studies have shown that a planned strategy in choosing the crossover points results in better performance with less chance of premature convergence. In particular, if the crossover point is chosen as a dense cut in the parent networks, then the connectivity among vertices within each partition would be deteriorated significantly. This results in offsprings with much poorer fitness than their parents. On the other hand, if the parents are hybridized at a sparse cut, the connectivity among vertices within each partition is disrupted much less. Therefore, crossover operations are best done by selecting sparse balanced cuts ($|V_1| \approx |V_2|$). Finding sparse balanced cuts is a well-studied problem in combinatorial optimization, for which various approximation algorithms exist [96, 140]. However, these algorithms assume that the graph itself,

not the connectivity matrix M, is given as input. We therefore use a simple heuristic that avoids cutting through the dense regions of the network. To generate these sparse cuts, we contract each dense region identified in Algorithm 4.3.2 to a single vertex, and then generate a weighted (by the number of vertices in each dense region) balanced partition of the vertices at random.

Mutation. In order to introduce variability to the population of candidates, a small number of edges ($5-10\%$) are randomly inserted or deleted. Moreover, observe that the child network constructed as above may not remain connected. Aside from the random mutation, therefore, we employ a simple local search that greedily adds edges to keep the network connected.

Genetic algorithm parameter selection. The various parameters of the genetic algorithm were selected based on the resulting performance on the Yu et al. data set. Two main parameters that affect the performance significantly are the population size and the selection criteria. For selection criteria, we tested several different selection criteria by setting the probability of choosing a candidate as a parent. The best compromise between running time and accuracy was obtained using a population size of 500 and, a selection probability for a parent proportional to fitness(c_i)$-$minFit, where minFit is the fitness of the worst candidate in the population.

4.3.4 Restricting the Solution Space for GA

While our genetic algorithm offers a plausible method for the A-DIGCOM problem, one can reduce the size of the solution space, which typically results in faster convergence to better solutions, using the results in Theorem 4.1 and 4.3. First, recall that finding all cut edges decomposes the problem into independent subproblems on 2-edge-connected components. Second, the identification of degree-2 vertices defines two sets of edges E_{YES} and E_{NO} that constitute all putative edges incident to the identified degree-2 ver-

tices. In other words, E_{MAYBE} forms the subgraph of G *induced* by the set V^{3+} of vertices with degree ≥ 3.

Furthermore, observe that the edges in E_{YES} form parallel paths between vertices in V^{3+}. A classical result in network reliability (see Fact 4.5) suggests that these parallel paths can be merged into a single meta-edge whose reliability can be efficiently computed. To be more formal, consider the set

$$\{\mathscr{P}_1(u,v),\mathscr{P}_2(u,v),\ldots,\mathscr{P}_k(u,v)\}$$

which is the set of paths between u and v in E_{YES}. These paths can then be replaced by a single edge (u,v) with survival probability $\hat{p}(u,v) = 1 - \Pi_{i=1\ldots k}(1-\hat{p}^{|P_i(u,v)|})$. By merging every set of parallel paths, we obtain a compact network over V^{3+} that efficiently encodes the edges in E_{YES}. Since our genetic algorithm handles the case where the edge survival probability is non-uniform, this compact encoding results in substantial gains in the running time for estimating the fitness of the candidates, as well as in time and space requirements for handling large population sizes. In our applications, this allows us to remove approximately $70-75\%$ of the original set of vertices.

4.4 Experimental Results

In this section, we provide the details on how our algorithm is tested on various datasets. Then, the accuracy of the predicted direct PPI network is evaluated using simulated data. Finally, our algorithm is tested on a well-known yeast AP-MS dataset to compare against PPI datasets from other sources.

4.4.1 Randomized Hill-climbing Algorithm

For performance comparisons, we tested our algorithm against a simple randomized hill-climbing approach. In this approach, we start with a randomly chosen spanning tree G^1 of the vertex set V. At the i^{th} iteration, we first sample the connectivity probability P_{G^i} of G^i, using the Monte Carlo simulation. Then, we randomly pick a vertex pair u, v with probability proportional to

$$\frac{D(u,v)}{\sum_{\forall i,j \in V} D(i,j)},$$

where $D(u,v) = |M(u,v) - P_{G^i}(u,v)|$. If the selected pair u, v are connected by an edge in G^i, but $M(u,v) > P_{G^i}(u,v)$, then we remove (u,v) from G^i. On the other hand, if u and v are not connected by an edge, but $M(u,v) < P_{G^i}(u,v)$, then we add (u,v) to G^i. We repeat this local optimization heuristic while making sure the candidate solution remains connected.

4.4.2 Choosing a Tolerance Level δ and Handling Numerical Errors

In order to deal with numerical errors from the Monte Carlo sampling, we use an additive tolerance level δ. Note that the sampling process for estimating the probability matrix P_G is a binomial process, which, by the central limit theorem, is closely approximated by a normal distribution. The confidence interval is largest when the estimated probability is equal to 0.5, in which case we obtain a confidence interval of

$$\hat{p} \pm \delta = \hat{p} \pm \frac{z_{1-\alpha/2}}{2\sqrt{n}},$$

where $\hat{p}$ denotes the fraction of samples where the two vertices are connected after n samples, and $z_{1-\alpha/2}$ is the z-value for desired level of confidence. Using this formula, we conclude:

1. When $n = 20000$ (the input matrix M by sampling the test networks), we obtain a 95% confidence interval of size at most $2 \cdot \delta = 2 \cdot 0.007 = 0.014$.

2. When $n = 500$ (the connectivity matrix by sampling each candidate solution in our genetic algorithm), the 95% confidence interval is of size at most $2 \cdot \delta = 2 \cdot 0.04 = 0.08$.

With the chosen tolerance level δ, we modify our algorithm as appropriate each time we compare two connectivity probabilities. For example, in Theorem 4.1, the first condition $P_G(u,v) = \hat{p}$ is modified to $P_G(u,v) \in [\hat{p}-\delta, \hat{p}+\delta]$; and in Theorem 4.3, we modify the first condition $P_G(s,t) < 2\hat{p} - \hat{p}^2$ to $P_G(s,t) < 2\hat{p} - \hat{p}^2 + \delta$.

4.4.3 Generation of Scale-free Networks

In order to generate artificial scale-free networks, we used two generation models: the preferential attachment model, and the duplication model. In the preferential attachment model, we evaluated the degree distribution of the two biological networks (G_{Yu} and G_{DIP}) and used the Barabási-Albert algorithm to construct a scale-free network with attachment factor 1.5 (each iteration adds a new vertex with $1-2$ edges attached to existing vertices). In the duplication model, at each iteration, we randomly pick a vertex to duplicate with probability proportional to its degree and randomly drop the duplicated edges with probability at 0.5 in order to fit the degree distributions and sparsity of biological networks. While there are various other random graph models proposed for PPI networks, these scale-free network models provided a sufficiently diverse set of initial population for our genetic algorithm.

4.4.4 Calculation of Connectivity Matrix from Peptide Counts

The peptide count of a prey protein in an AP-MS experiment is the number of different peptides that have been observed by mass spectrometry for that protein. We note that

the peptide counts are biased towards preys with longer protein sequences, and to rectify this propensity, we normalized the abundance data by the protein sequence lengths to obtain the abundance ratios $R(i,j)$. In order to turn the normalized abundance ratios into the connectivity matrix for our probabilistic graph model, we used a simple logistic function

$$\hat{M}(i,j) = \frac{1}{1+e^{-\frac{R(i,j)-\alpha}{\beta}}}$$

where the parameters α, β are chosen so that the computed distribution of $\hat{p}$ fits the simulated connectivity distribution of G_{Yu}, using a χ^2 test ($\alpha = 2.8921, \beta = -0.6318$). In the cases where $R(i,j)$ differ from $R(j,i)$, we choose the average of the two entries to symmetrize the matrix.

4.4.5 Model Validation

To test our approach, we first sought to validate our model of AP-MS data. To this end, we used one of the most comprehensive yeast AP-MS networks, obtained by Krogan et al. [93]. The dataset reports the Mascot score [114] and the number of peptides detected for each bait-prey pair. The complete set of interactions reported contains 2186 proteins and 5496 interactions (Krogan et al. Table S6); we call the resulting network $G_{KroganFull}$. The authors identified a subset of these interactions as high-confidence, based on their Mascot scores (Krogan et al. Table S5). We call this set of high-confidence interactions $G_{KroganHigh}$; this network consists of 1210 proteins and 2357 interactions. We expect that $G_{KroganHigh}$ is relatively rich in direct interactions, whereas the complete set of interactions $G_{KroganFull}$ consists in part of indirect interactions.

Considering $G_{KroganHigh}$ as a direct interaction network, we simulated Monte Carlo sampling to estimate $P_{G_{KroganHigh}}$, using $\hat{p} = 0.5$ and 50,000 samples, which yields a 95% confidence interval of size at most 0.007 on each $P_{G_{KroganHigh}}(u,v)$ entry. Next, we normalized the peptide counts of the interactions in $G_{KroganFull}$ using protein lengths. We then compared $P_{G_{KroganHigh}}$ to the normalized peptide counts of the interactions in $G_{KroganFull}$.

We expect that a significant fraction of low-confidence interactions in $G_{KroganFull} - G_{KroganHigh}$ are likely to be indirect interactions. If our model is correct, their peptide counts should then be correlated with the corresponding entries in $P_{G_{KroganHigh}}$. Indeed, the positive linear correlation between the predicted connectivity $P_{G_{KroganHigh}}$ and the observed normalized peptide counts is very significant (regression p-value of 8.17×10^{-11}, Student t-test). Furthermore, this correlation is strongest when $\hat{p} \approx 0.5$, as compared to $\hat{p} = 0.3$ or 0.7, justifying the use of this value in our subsequent analyses.

4.4.6 Accuracy of the Algorithm

The ideal validation of the accuracy of our algorithm would involve (i) constructing a connectivity matrix M using actual quantitative AP-MS data; (ii) predicting direct interactions based on M using our algorithm; and then (iii) comparing our predictions to experimentally generated direct interaction data. Yeast 2-Hybrid (Y2H) experiments are less prone to detect indirect interactions than are AP-MS methods, and several large-scale efforts have been reported [53, 74, 137]. Unfortunately, for a number of technical reasons, the overlap between AP-MS PPI networks and Y2H networks remains very small [149]. As a consequence, Y2H data cannot be used directly to validate predictions made on AP-MS data. Instead, we had to rely on partially-synthetic data set, where an actual network of high-quality Y2H interactions is assumed to form the direct interaction graph, and a connectivity matrix is generated from it using Monte Carlo sampling, under our model. Two sets of Y2H interactions were used: (i) G_{Yu} is the network constructed from the gold standard dataset of Yu et al. [149]. This network consists of 1090 proteins and 1318 interactions with high confidence of direct interactions; (ii) G_{DIP} is the core, high-quality, physical interaction network of yeast, available at the DIP database, version 20090126CR [146], consisting of 1406 proteins and 1967 interactions.

These biological networks were complemented with two artificial 1000-vertices networks. The first was generated using the preferential attachment model (PAM) [10]. For the second, we used the duplication model (DM) [13], which, in contrast to the PAM,

generates graphs containing several dense clusters. The resulting artificial "direct" interactions graphs are called G_{PAM} and G_{DM} and contain $1500-2000$ interactions each. We then used the Monte Carlo sampling approach described above to estimate the connectivity matrices $P_{G_{Yu}}$, $P_{G_{DIP}}$, $P_{G_{PAM}}$, and $P_{G_{DM}}$. These will form the input to our inference algorithm, whose output will then be compared to the corresponding direct interaction graph. It is important to note that these input matrices are not perfectly accurate and may contain sampling errors. However, it is easy to bound the size of the errors with high probability and use this as a tolerance level within our algorithm. We also note that the results presented in this section only aim at evaluating the performance of the inference algorithm on input data that was generated exactly according to our probabilistic model. As such, the error rates reported may be considered as lower bounds for those on actual biological data. An assumption-free evaluation is provided later in this section.

Identification of weakly connected vertices

Theorem 4.1 provides an efficient algorithm that guarantees the identification of all cut edges, provided that the given connectivity matrix is precise. We say that a vertex v is a *1-cut vertex* if *all* edges incident on v are cut edges. By applying Theorem 4.1 recursively to detect cut edges and decomposing the graph into two connected components, we can detect and remove all 1-cut vertices from the input connectivity matrix. Table 4.1 (i) reports the number of 1-cut vertices that are detected by the recursive algorithm from Theorem 4.1. In both the Yu and the DIP network, 1-cut vertices constitute approximately 50% of the network, and identifying them allows a significant reduction in the problem size. We note that the inaccuracies in the input connectivity matrices could, in principle, have introduced errors in the detection of cut edges. However, this rare event was never observed on any of our networks.

Algorithm 4.3.1 (see Methods) guarantees to efficiently identify all degree-2 vertices (again, provided that the connectivity matrix is known), but may also incorrectly flag some higher-degree vertices. As seen in Table 4.1 (ii), nearly all degree-2 vertices were

Network	Total	(i) 1-cut vertices				(ii) Degree 2 vertices				Remaining
		real	pred.	FDR(%)	FNR(%)	real	pred.	FDR(%)	FNR(%)	
Yu	1090	552	552	0	0	195	207	7.7	2.05	331
DIP	1406	656	656	0	0	309	326	5.82	0.64	424
PAM	1000	457	457	0	0	351	363	3.58	0.28	180
DM	1000	323	323	0	0	117	126	11.9	5.12	551

Table 4.1: Performance of detecting weakly connected vertices. Number of vertices detected as (i) 1-cut vertices, and (ii) degree 2 vertices in the real network and the predicted network. False discovery (FDR) and false negative (FNR) ratios are given in percentage. Remaining: the number of vertices remaining after identifying 1-cut vertices and degree 2 vertices.

identified, with a low false-discovery rate ranging from 6% to 9%. Moreover, the false-positives incorrectly detected as degree-2 vertices indeed had small degrees, and their predicted neighbours were mostly correct (but incomplete) predictions. Flagging degree-2 vertices reduces the problem size further by 15% to 36%.

After repeatedly detecting and removing 1-cut vertices and degree-2 vertices from the problem space, the edges adjacent to approximately 70% of the vertices are detected with very low error rate. The remaining vertices only constitute approximately 30% of the original network. We call this remaining subset the *hard core* of the connectivity matrix. Because it is more densely connected than the rest of the network, the topology of hard core is more difficult to reconstruct.

Running our algorithm on the PAM simulated data yields similar resolution and error rates as on the Y2H networks. However, our DM network is found to be less amenable to these strategies, leaving 55% of vertices unresolved and resulting in an error rate approximately twice that seen for other networks. This is simply due to the fact that networks generated by the duplication model do not contain as many 1-cut vertices or degree 2 vertices when compared to other networks, including biological Y2H networks.

Identification of dense regions

Our dense region detection algorithm aims at identifying all edges that belong to a k-clique in G_{Direct}, for a given value of k. Table 4.2 reports the accuracy of the algorithm.

Network	$k=7$				$k=6$				$k=5$			
	real	pred.	FD(%)	FN(%)	real	pred.	FD(%)	FN(%)	real	pred.	FD(%)	FN(%)
Yu	42	54	22.22	0	66	104	36.53	0	146	308	55.19	5.48
DIP	0	42	100	0	86	112	26.79	4.65	184	266	35.34	6.52
PAM	0	31	100	0	0	45	100	0	0	96	100	0
DM	254	346	28.32	2.36	194	267	29.21	2.57	488	718	34.96	4.30

Table 4.2: Performance of quasi-clique predictions. Number of edges that belong to maximal cliques of size k. Real: actual number of edges that belong to maximal cliques of size k; pred: predicted number of maximal k-clique edges; FD(false discovery ratio): percentage of false positives in the predicted set; FN(false negative ratio): percentage of false negatives in the real set.

As expected, our algorithm achieves extremely high sensitivity for clique edges. However, the false-discovery rate is quite high, especially for the case of DIP and PAM, $k=7$. This is due to the fact that distinguishing a 5-clique from, say, a quasi-clique of size 7 is extremely difficult, causing false-positive predictions. We note however that these erroneous predictions are mostly inconsequential, as the intra-cluster topology of each dense region shall only be used in generating the initial candidate solutions for the genetic algorithm.

Cut-based genetic algorithm

The various parameters of the genetic algorithm (population size, mate selection probability, mutation rate, etc.) were optimized for the running time and accuracy of the solution based on G_{Yu}. Although our genetic algorithm could in principle be used on any connectivity matrix, running it on the full matrix of >1000 proteins is impossible: the search space is huge, and the amount of time required to evaluate the fitness of a given candidate solution is too large. However, as discussed previously, applying first the 1-cut and degree-2 vertex detection algorithms significantly reduces the problem size and makes it accessible to our genetic algorithm. Table 4.3 (i) reports the accuracy of the genetic algorithm predictions on the hard core of each connectivity matrix. We note that since the network to be inferred is relatively highly connected, the problem is significantly more difficult than the identification of 1-cuts and degree-2 vertices. Indeed, the false-discovery and false-negative rates range from 35% to 55% for most datasets.

Network	(i) reduced network				(ii) overall network			
	real	pred.	FDR(%)	FNR(%)	real	pred.	FDR(%)	FNR(%)
Yu	563	552	43.65	44.76	1318	1390	14.96	10.31
DIP	931	890	35.50	38.34	1967	2041	17.34	14.23
PAM	473	421	49.88	55.39	1538	1462	16.14	20.28
DM	1138	1295	43.39	35.58	1869	1804	32.81	35.15

Table 4.3: Performance of genetic algorithm and overall algorithm. (i) Performance of the genetic algorithm on the reduced network obtained from removing 1-cut vertices and degree-2 vertices. (ii) Overall performance of the combined prediction pipeline on the complete connectivity matrix. Real and predicted describe the number of edges in the real and predicted networks, respectively.

For a comparison, an algorithm that would pick edges randomly would achieve 98.75% false-discovery and false-negative rates.

Combining the three phases of the algorithm, the overall error rate obtained on each data set ranges from 10 to 20% false-discovery and false-negative rates, except for the DM data set, which fares considerably worse, for the reasons explained earlier.

To the best of our knowledge, there has been no other efforts to solve the DIGCOM problems (neither exact, nor approximate version). We thus compared our approach to a simple hill climbing search algorithm on the Yu et al. data set (see Methods). We let this algorithm run over several days (as opposed to few hours spent using our approach), with multiple restarts, and discovered that it provides very poor sensitivity and specificity (see Table 4.4 for the best results obtained). This is not surprising since the hill climbing method is highly dependent on the initial solution (in this case, a spanning tree chosen randomly based on the connectivity matrix) and the search space is simply

Network	Real	Predicted	FDR(%)	FNR(%)
(i) Hill-climbing	1318	2108	86.67	78.68
(ii) Hill-climbing + weakly conn. nodes	1318	1517	43.57	35.05
(iii) Our approach (GA)	1318	1390	14.96	10.31

Table 4.4: Comparison of our method to simple hill-climbing approach. (i) Accuracy of the hill-climbing approach used over the complete network; (ii) Accuracy of the hill-climbing approach after fixing the weakly connected nodes using our algorithm; (iii) Accuracy of our combined pipeline using the genetic algorithm.

Network	(i) 1-cut & degree 2 vertices (secs)	(ii) Quasi-clique predictions	(iii) Genetic algorithm
Yu	0:00:29	0:31:02	15:00:00
DIP	0:00:41	0:48:14	15:00:00
PAM	0:00:19	0:29:49	15:00:00
DM	0:00:24	1:18:03	15:00:00

Table 4.5: Running times for each phase of our algorithm. (i) Detecting 1-cut vertices and degree two vertices; (ii) Predicting quasi-clique clusters; and (iii) Running the genetic algorithm. We are reporting the average run time over three runs on each network. The implementation was tested on a Powermac G5 2Ghz with 4GB of RAM. Note that the genetic algorithm was run for a fixed amount of time, and the top scoring candidates achieved the quality as shown in Table 3. The times are shown in *hh:mm:ss* format.

too large to exhaustively search for the a good initial solution. We also tested the hill-climbing approach in the same setting as the genetic algorithm, i.e. combining it with the 1-cut edges and degree-2 vertices detection algorithms. Here, the hill-climbing approach showed a better sensitivity and specificity than the pure hill-climbing approach, but still performed much worse than our genetic algorithm (Table 4.4). Furthermore, the improvement over the pure hill-climbing approach was mostly due to the high sensitivity and specificity of our algorithm for detecting weakly connected vertices.

To provide an idea of the running times, Table 4.5 gives the empirical data from our experiments. The first two phases (detecting weakly connected vertices and recognizing dense regions) were run within seconds while the genetic algorithm was run for a fixed amount of time.

4.4.7 Inferring Direct Interactions from AP-MS Experimental Data

In order to apply our algorithm to biological data from AP-MS experiments, we used the raw data reported by Krogan et al. [93] for the 2186 putative interactions of $G_{KroganFull}$. We only considered the subnetwork of tagged proteins, and further focussed our efforts on the analysis of 77 proteins that are well separated in the tag-induced subnetwork. Quantitative abundance estimates were derived from the peptide counts reported for each prey, and an experimentally derived connectivity matrix M was obtained after normalization (see Section 4.4.4). Our full prediction algorithm was then run on the es-

timated connectivity matrix, resulting in a direct interaction graph prediction we call G_{Kim} that consists of 164 interactions (see Table 4.6). The network G_{Kim} was compared to $G_{KroganHigh}$, the set of high-confidence interactions reported by Krogan et al., and to $G^{Top}_{KroganHigh}$, a subset of $G_{KroganHigh}$ consisting of the 164 (to compare against G_{Kim}) most confident interactions they reported.

Both $G_{KroganFull}$ and $G_{KroganHigh}$ overlap G_{Kim} quite substantially. These three sets of predictions were then compared against a set of high-quality binary interactions from G_{Yu}. In Y2H experiments, the interaction partners are separately screened using a genetic readout. Therefore, interactions from G_{Yu} are believed to be direct, and thus used to test against the predictions from AP-MS data. On the other hand, these interactions may reflect only a subset of all direct interactions among the 77 proteins.

As shown in Figure 4.4, our results show that the high-confidence AP-MS data $G_{KroganHigh}$ exhibited very little overlap with the direct binary interaction set G_{Yu}. 72.6% of interactions in $G_{KroganHigh}$ are disjoint from G_{Yu}, and 25% of G_{Yu} remains undetected by $G_{KroganHigh}$. Furthermore, even the top scoring set of interactions $G^{Top}_{KroganHigh}$ showed high discrepancy ratios against G_{Yu}. In contrast, G_{Kim} produced by our algorithm coincides with G_{Yu} with better sensitivity and specificity. Given the crudeness of the method in translating the AP-MS data into a connectivity matrix, our algorithm has thus performed relatively well in predicting direct interactions from real AP-MS data.

4.5 Discussions

Approaches for determining bait-prey abundance remain in their infancy, and to date, no large-scale PPI networks have this type of quantitative data. As these approaches improve in accuracy, so will the results of our method. Furthermore, as the sensitivity of AP-MS pipelines improves, the fraction of indirect interactions detected will also increase, thereby making the ability to distinguish them even more critical.

Protein A	Protein B	Protein A	Protein B	Protein A	Protein B
YOL012C	YKR048C	YNL031C	YJL115W	YLR418C	YIL035C
YBL003C	YGL241W	YNL031C	YBL003C	YDR054C	YGL173C
YBL003C	YKR048C	YNL031C	YLL022C	YDR054C	YLR418C
YBL003C	YGL207W	YNL031C	YIL035C	YER112W	YCR077C
YBL003C	YIL035C	YNL031C	YGL207W	YER112W	YDR432W
YGR103W	YIL035C	YNL031C	YER164W	YMR229C	YDR432W
YGR103W	YMR229C	YOR039W	YER164W	YOR206W	YMR229C
YGR103W	YOR272W	YER164W	YGL207W	YHL034C	YOR310C
YGR103W	YKR048C	YBL002W	YOL012C	YHL034C	YDR432W
YGR103W	YNL189W	YBL002W	YDR224W	YHL034C	YLR175W
YGR103W	YLR347C	YBL002W	YKR048C	YHL034C	YMR229C
YMR304W	YGR159C	YBL002W	YBR245C	YHL034C	YGL173C
YMR304W	YDR432W	YBL002W	YGL207W	YOL076W	YGR159C
YMR304W	YLR197W	YBL002W	YIL035C	YOL076W	YDR432W
YMR075W	YBL002W	YLL022C	YPL001W	YNL088W	YGR159C
YMR075W	YDR432W	YBR010W	YBR245C	YCR077C	YDR432W
YMR075W	YNL189W	YBR010W	YGL207W	YER146W	YCR077C
YOL004W	YAL034C	YBR010W	YDR432W	YHR064C	YGL173C
YOL004W	YMR075W	YBR010W	YKR048C	YOR048C	YDR432W
YOL004W	YDR432W	YBR010W	YPL001W	YGL241W	YKR048C
YOL004W	YIL035C	YBR010W	YBL002W	YDR225W	YGL207W
YOR310C	YGL120C	YBR010W	YLL022C	YDR225W	YKR048C
YOR310C	YNL189W	YCR060W	YGR159C	YDR225W	YGL241W
YLR455W	YOR310C	YJR041C	YNL189W	YGL207W	YOR039W
YLR455W	YOR206W	YJR041C	YDR432W	YGL207W	YGL244W
YLR455W	YNL088W	YJR041C	YMR125W	YGL207W	YIL035C
YLR455W	YMR229C	YJR041C	YPL178W	YGL207W	YBR279W
YLR455W	YDR496C	YNL209W	YDR432W	YGL207W	YOL145C
YBR009C	YPL001W	YNL209W	YER146W	YGL207W	YNL088W
YBR009C	YGL207W	YNL209W	YKR048C	YGL207W	YGL173C
YNL030W	YPL001W	YGL244W	YLR418C	YGL207W	YOR061W
YNL030W	YBR009C	YGL244W	YOL145C	YDR224C	YGL207W
YNL030W	YLL022C	YOR123C	YGL207W	YDR224C	YKR048C
YNL030W	YGL207W	YOR123C	YGL244W	YGL120C	YGR159C
YKR024C	YMR229C	YDR234W	YHR064C	YGL120C	YDR432W
YKR024C	YGL173C	YOR061W	YOR039W	YMR128W	YGL120C
YBR103W	YDR432W	YIL035C	YBR009C	YGR272C	YHL034C
YDL229W	YBR169C	YIL035C	YDR172W	YGR272C	YDR432W
YDL229W	YMR229C	YIL035C	YOR061W	YMR014W	YDR432W
YDL229W	YGL207W	YIL035C	YER164W	YMR014W	YGR272C
YDL229W	YDR432W	YPL153C	YGR159C	YNL147W	YGL173C
YDL229W	YKR048C	YPL153C	YNL189W	YDR432W	YDL060W
YDL229W	YGR159C	YJL115W	YOR061W	YGL147C	YDR432W
YDL229W	YHR064C	YJL115W	YIL035C	YPL178W	YLR347C
YOL145C	YBR279W	YJL115W	YPL153C	YPL178W	YMR125W
YOL145C	YDR172W	YLR410W	YGR159C	YGR159C	YOR310C
YOL145C	YOR123C	YLR410W	YKR048C	YML062C	YNL189W
YBR279W	YLR418C	YLR197W	YOR310C	YML062C	YGR159C
YBR279W	YIL035C	YLR197W	YLR347C	YMR125W	YNL189W
YBR279W	YGL244W	YDL040C	YMR229C	YNL139C	YLR347C
YBR279W	YOR123C	YDL040C	YLR175W	YNL139C	YNL189W
YDR365C	YMR229C	YDL040C	YLR197W	YNL189W	YLR347C
YDR365C	YGR159C	YLR418C	YGL207W	YDR289C	YNL189W
YNL031C	YPL001W	YLR418C	YOR123C	YDR289C	YLR347C
YNL031C	YBR009C	YLR418C	YNL189W		

Table 4.6: The set of direct interactions among 77 yeast proteins, predicted by our algorithm. The connectivity matrix is generated from normalized peptide counts, and then our algorithm is run to predict the direct interactions.

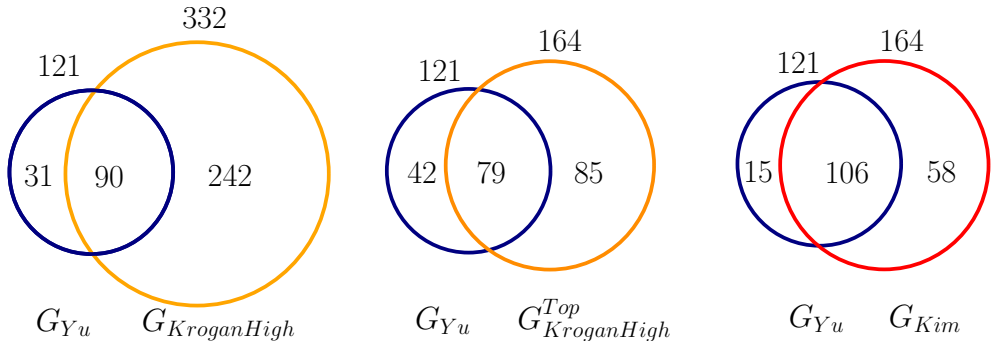


Figure 4.4: Inferring direct interactions from actual AP-MS dataset. Overlap between the Y2H interaction network of Yu et al. and various AP-MS-based networks: (a) High-confidence set of interactions from Krogan et al. (b) The set of 164 highest scoring interactions from Krogan et al. (c) The set of 164 interactions predicted as direct interactions by our algorithms, based on the AP-MS data from Krogan et al.

In this chapter, we lay the groundwork for modelling indirect interactions in AP-MS experiments. We formulate the DIGCOM problems, which aim at distinguishing direct interactions from indirect ones, and provide a set of theoretical and heuristic approaches that are shown to be highly accurate on both biological PPI networks and simulated networks. Despite the unrealistic assumptions that should be relaxed, our results show that the predicted set of interactions fits the experimental data reasonably well. In addition, applying our algorithms to a large-scale AP-MS data set from Krogan et al. results in predictions that overlap Y2H data approximately 35% more often than the equivalent number of top-scoring interactions reported by these authors.

The DIGCOM problems raise a number of challenging, yet fascinating computational and mathematical problems. Is the solution to the exact DIGCOM problem, if it exists, always unique? We suspect it is. What is the computational complexity of the exact and approximate DIGCOM problems? We believe they are NP-hard, and possibly not even in NP. Are there types of graph substructures, other than those discussed here, that can be unambiguously inferred from P_G? Are there special properties of PPI networks, other than the power-law degree distribution, of which an algorithm can take advantage to make more accurate predictions and/or provide approximation or probabilistic guarantees?

The model and algorithm proposed here are only a first step toward an accurate detection of direct interactions from AP-MS data. Several generalizations and improvements are worth investigating. First, the abundance of an interaction is not constant and needs to be modelled more accurately. Second, the strength of all physical interactions is non-uniform, and some interactions may be more prone to disruption by the affinity purification process than others. Given sufficient quantitative AP-MS data, one may study a generalization of the DIGCOM problems that aims at identifying not only the set of direct interactions, but also their individual strengths and abundances. While modelling these aspects is in theory possible, the amount and quality of experimental data required is currently unavailable, and the computational complexity of the resulting problems are likely to be daunting.

Perhaps a more significant limitation of our model is that all direct interactions are assumed to occur simultaneously, though it is clear that certain interactions are either mutually exclusive, or restricted to specific subcellular compartments or conditions. One can investigate approaches to decompose the observed network into a family of simultaneously occurring interactions. In order to do so, complementary experimental data, such as comprehensive protein localization assays or cell cycle expression data, would be required to reduce the space of possible solutions in a biologically meaningful manner.

An additional assumption that may need to be relaxed is the independence of the edge failures, which may not hold in cases where the loss of an interaction between two proteins causes a significant destabilization of the larger complex they belong to. Unfortunately, in the presence of strong dependencies between edge failures, it becomes almost impossible to distinguish direct from indirect interactions. Nonetheless, it may be possible to at least identify complexes where such dependencies hold, by studying subsets of proteins for which the AP-MS data differs significantly from our model.

4.6 Bibliographic Notes

The results discussed in this chapter have been published in [88].

Chapter 5

Hypergraph Modelling of PPI Data

PPI networks are traditionally modelled as graphs whose vertices represent proteins, and two vertices are joined by an edge if and only if the *two* corresponding proteins interact with each other. However, many recent discoveries of protein complexes as building blocks of the proteome led us to believe that protein-protein interactions may involve *any* number of interaction partners [7, 58, 93]. Consequently, various methods have been proposed for detecting protein complexes (e.g. [7, 90, 129]). We discussed some of these methods in detail in Chapter 1.

One of the limitations of the existing approaches is that the identified protein complexes are often disjoint, and the sparse inter-cluster bridges remain as binary interactions. Some recently proposed methods do allow finding overlapping clusters [1, 7, 98], but most of these approaches are designed to first identify dense clusters, and then try to merge the nearby clusters. While intuitive, such a strategy often fails to find a globally optimum structure, if, for example, the optimization criteria is to minimize the number of clusters. In fact, none of these methods guarantees to find the optimum solution for the problem they are designed for. As a result, finding a comprehensive model for PPI networks as a collection of protein complexes remains unsolved to date.

This chapter is concerned with the problem of modelling binary PPI data as a hyper-

graph, i.e., a set system where each set represents a protein complex. In order to model this problem, let $G = (V, E)$ denote the combinatorial graph constructed from the widely available binary PPI data (e.g. Y2H interaction data, or the direct AP-MS data produced from Chapter 4). Then, each protein complex in the PPI network corresponds to a set S of vertices in G, where each pair of vertices in S share an edge between them. In other words, each protein complex corresponds to a *clique* in G. Our problem of finding a hypergraph (with possibly a small number of sets, or *hyperedges*) can then be stated as finding a *clique cover* of minimum cardinality.

(EDGE) CLIQUE COVER. Given a graph $G = (V, E)$, find a smallest collection Q of cliques in G such that every edge in E belongs to at least one clique in Q.

A similar problem can be defined as a partitioning problem:

(EDGE) CLIQUE PARTITION. Given a graph $G = (V, E)$, find a smallest collection Q of cliques in G such that every edge in E belongs to *precisely* one clique in Q.

Therefore, we draw our attention to the clique cover problem, with an application to hypergraph modelling of PPI data. In fact, our theoretical pursuit on the clique cover problem gives rise to efficient algorithms for sparse networks (including PPI networks), where the graph sparsity is measured by various graph parameters.

Related Work. In addition to graph theoretic aspects, the clique cover problem has been studied extensively from the standpoint of computational complexity. In particular, there are numerous studies concerning approximability and fixed-parameter tractability. Clique cover is NP-hard in general [110], even when the input graph is planar [24] or has bounded degree [71]. Furthermore, Lund and Yannakakis have shown that clique cover is not approximable within a factor of $|V|^{\epsilon}$ for some $\epsilon > 0$ unless $\mathsf{P} = \mathsf{NP}$ [101], thereby removing the hope of good approximation algorithms in the general case. In the case of clique *partition* problem, the problem has also been shown to be NP-complete

for various restricted classes of graphs [23, 56, 57, 72].

A parameterized problem is *fixed-parameter tractable* (FPT) if it can be solved in $f(k) \cdot |I|^{O(1)}$ time, where f is a computable function depending on some parameter k, independent of the input size $|I|$. Recently, Gramm et al. [66] showed that the clique cover problem is FPT when the size of the cover is chosen for the parameter k. Similarly, Mujuni and Rosamond [105] have shown that the clique partition problem is FPT with the output size chosen as the parameter. These algorithms run in polynomial time in the input size but exponential time in the number of cliques in the solution. As a result, these algorithms are well suited for dense graphs where a few cliques can cover the entire graph, but they are not suitable for sparse graphs that require a large number of small cliques in the solution.

In this chapter, we fill the gap by designing efficient algorithms for such sparse graphs. We first introduce some definitions and related results in Section 5.1. Then, in Section 5.2, we present an exact algorithm for clique cover when the input graph has bounded treewidth. Section 5.3 discusses the problem restricted to planar graphs, and we provide a polynomial time approximation scheme. Finally, we show the performance of our algorithm from experimental studies in Section 5.4. In particular, our algorithm shows efficient and practical running time for both real and simulated biological networks. Furthermore, our PTAS for planar graphs shows a clear trade-off between the approximation ratio and the running time when tested against random planar graphs.

5.1 Preliminaries

Throughout this chapter, we focus on the clique cover problem for sparse networks. Where possible, we shall discuss how the algorithms for the clique cover problem can be modified to solve the clique partition problem. Various measures of network sparsity have been proposed in the past, with possibly the best known being treewidth. We recall the definition of tree decomposition from Section 2.2.

Definition 5.1. [120] *Tree decomposition of a graph $G = (V, E)$ is a pair $(X = \{X_i | i \in I\}, T = (I, F))$ where each node $i \in I$ is associated with a set of vertices $X_i \subseteq V$, such that*

(1) Each vertex belongs to at least one node: $\bigcup_{i \in I} X_i = V$.

(2) Each edge is induced by at least one node: $\forall (v, w) \in E$, there is an $i \in I$ with $v, w \in X_i$.

(3) For each $v \in V$, the set of nodes $\{i \in I | v \in X_i\}$ induces a subtree of T.

Then, the width of a tree decomposition (X, T) is defined as $max_{i \in I} |X_i| - 1$, and the *treewidth* of a graph G, denoted tw(G), is the minimum width over all tree decompositions of G.

An alternative definition of treewidth can be given using k-trees.

Definition 5.2. [3] *A k-tree is defined recursively as follows:*

(i) The complete graph on k vertices is a k-tree, and

(ii) A k-tree G with $n+1$ vertices ($n \geq k$) can be constructed from a k-tree H with n vertices by adding a vertex adjacent to exactly k vertices that form a k-clique in H.

Definition 5.3. *A graph is a partial k-tree if it is a subgraph of a k-tree.*

Then, the class of partial k-trees is equivalent to the class of graphs with treewidth k. Note that this recursive definition provides a simple algorithm to construct a graph of treewidth k; we shall use this approach to generate a set of synthetic test data in Section 5.4.1. In general, it is NP-complete to determine the treewidth of a graph [3]. However, when k is fixed, graphs with treewidth k can be recognized, and width k tree decompositions can be constructed, in linear time [17].

From the empirical studies of our input PPI data (shown in Section 5.4, Table 5.1), the treewidth of PPI networks is often small compared to the network sizes. We thus assume,

where applicable, that the input graph has a bounded treewidth, and its optimum tree decomposition can be constructed efficiently. Furthermore, for ease of exposition, we assume that the decomposition tree T admits a *nice* structure as defined below.

Definition 5.4. [92] *A tree decomposition (X, T) is called nice if the tree T is rooted, and for each node $i \in I$, one of the following holds:*

1. LEAF*: node i is a leaf of T, and $|X_i| = 1$.*
2. JOIN*: node i has exactly two children j_1 and j_2 such that $X_i = X_{j_1} = X_{j_2}$.*
3. INTRODUCE*: node i has exactly one child j, and $X_i = X_j \cup \{v\}$.*
4. FORGET*: node i has exactly one child j, and $X_j = X_i \cup \{v\}$.*

Figure 5.1 illustrates an example of these node types. It is easy to see that if $\text{tw}(G) \leq k$, then G also admits a nice tree decomposition of width $\leq k$, with $O(n)$ tree nodes: given an arbitrary decomposition tree T, one can repeatedly split each node X_i until all nodes satisfy the conditions above.

Another closely related graph parameter is *branchwidth*. We recall its definition from Section 2.2.

Definition 5.5. [121] *A branch decomposition (T, ϕ) of a graph G is characterized by a ternary tree*[1] *T, and a bijection ϕ from the leaves of T onto the edges of G.*

Let e be a tree edge in T. Removing e from T partitions into T_1 and T_2, and this partition induces a partition of edges in G, called an *e-separation*, associated with the leaves of T_1 and T_2. The set of vertices in G that are shared by both G_1 and G_2 is called the *middle-set* of e, and the *width* of this separation is the number of vertices in the middle-set.

Given a branch decomposition (T, ϕ), the width of this branch decomposition is the maximum width over all e-separations in T, and the *branchwidth* of G, denoted $\text{bw}(G)$,

[1] A tree T is a ternary tree if every non-leaf node has degree 3.

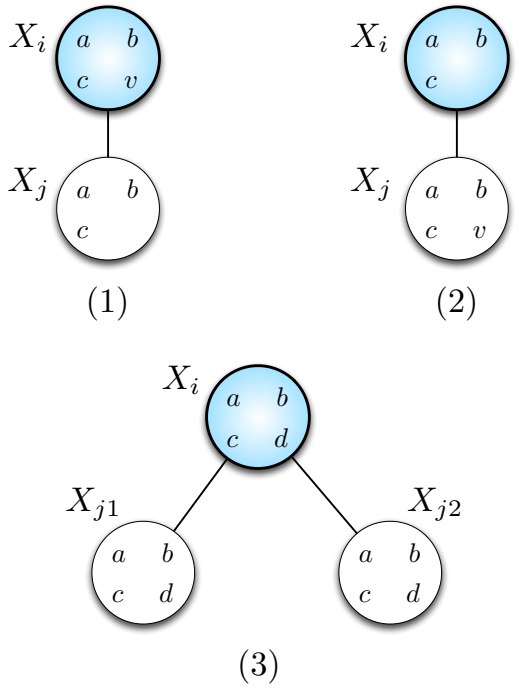


Figure 5.1: Node types in a nice tree decomposition: (1) introduce node: $X_i = X_j \cup \{v\}$; (2) forget node: $X_j = X_i \cup \{v\}$; (3) join node: $X_i = X_{j_1} = X_{j_2}$

is the minimum width over all branch decompositions. It is well known that the branchwidth is closely related to the treewidth of graph [121]: $\text{bw}(G) \leq \text{tw}(G)+1 \leq \frac{3}{2}\,\text{bw}(G)$. For planar graphs, Fomin and Thilikos gave an upper bound on the branchwidth:

Theorem 5.6. [52] *For any planar graph G, $bw(G) \leq \sqrt{4.5n} \approx 2.122\sqrt{n}$.*

5.2 Clique Cover for Graphs with Bounded Treewidth

In this section, we design a dynamic programming algorithm for finding a minimum clique cover for a graph G where a *nice* tree decomposition (X, T) is given. Let k denote the width of that decomposition. First, let us define $E(X_i)$ to be the set of edges in the subgraph induced by the vertices in X_i. Furthermore, we let V_i denote the *union* of all vertices in X_i *and* its descendent nodes. Similarly, let G_i denote the subgraph of G induced by the vertices V_i. Finally, for some $v \in V(G)$, let $\delta(v)$ denote the set of edges that

are incident to v in G.

We shall in fact design an algorithm for a generalization of the clique cover problem, where we are given a subset S of edges that are *already covered*, i.e., our solution need not cover S, but may use these edges in the cliques. Then, the original clique cover problem is a special case where $S = \emptyset$. Since our dynamic programming is formulated around the decomposition tree T, we often speak of a subgraph G_i where a certain subset S of edges is already covered, denoted by $G_i(S)$. Now we can define a cost function:

$$C_i(S) = \text{ minimum size of clique cover for } G_i \text{ where } S \text{ is already covered.}$$

Then, our final solution is precisely $C_r(\emptyset)$ where r is the root of T. Our dynamic programming algorithm will proceed from the leaves of T up to its root, computing, for each node X_i, the value of $C_i(S)$ for every possible subset S of $E(X_i)$. Depending on the type of node, $C_i(S)$ is computed differently.

LEAF NODE. Suppose i is a leaf node. Then, by the definition of nice tree decomposition, $|X_i| = 1$, and thus $C_i(S) = 0$ for all $S \subseteq E(X_i)$, trivially.

FORGET NODE. Suppose i is a forget node. Then, it has one descendant $X_j = X_i \cup \{v\}$ for some unique vertex v.

Lemma 5.7. *If i is a Forget node, then for any subset $S \subseteq E(X_i)$, $C_i(S) = C_j(S)$.*

Proof. Note that since $X_i \subset X_j$, the corresponding graphs G_i and G_j are the same. Furthermore, since v is not in X_i, $S \cap \delta(v) = \emptyset$. Therefore, $S \cap E(X_i) = S \cap E(X_j)$ and we have $C_i(S) = C_j(S)$. □

INTRODUCE NODE. Suppose i is an introduce node. Then, it has one descendant node X_j such that $X_i = X_j \cup \{v\}$ for some unique vertex v. Consider an arbitrary clique cover $\mathscr{W}$ for $G_i(S)$. Since the cliques in $\mathscr{W}$ can be partitioned into Q_v = cliques containing v and $Q_{\overline{v}} = \mathscr{W} - Q_v$, the following recurrence relation holds for each introduce node.

Lemma 5.8. *If i is an Introduce node, then for any subset $S \subseteq E(X_i)$:*

$$C_i(S) = min\{\, |Q_v| + C_j(S \cup E(Q_v)) \,:\, Q_v \text{ is a clique cover for } \delta(v) - S \,\}$$

Proof. Let $\mathscr{W}$ be a clique cover for $G_i(S)$. We partition the cover $\mathscr{W} = Q_v \cup Q_{\overline{v}}$ as defined, and consider the edgeset in Q_v. Since these edges are covered by Q_v, $Q_{\overline{v}}$ needs only cover $G_i - (S \cup Q_v)$. Moreover, since the cliques in $Q_{\overline{v}}$ do not contain v, $Q_{\overline{v}}$ is a cover for $G_j - (S \cup Q_v)$. Therefore, $|Q_{\overline{v}}| \geq C_j(S \cup Q_v)$, and the lemma follows. □

To compute $C_i(S)$, we need to consider all possible clique covers, Q_v, for $\delta(v) - S$. Since $|X_i| \leq k$, this is the number of ways to partition k vertices, and is given by the kth Bell number $B(n) \leq k!2^k$.

JOIN NODE. Finally, suppose i is a join node. Then, it has two children nodes j_1 and j_2 such that $X_i = X_{j_1} = X_{j_2}$, and $G_i = G_{j_1} \cup G_{j_2}$. Therefore, a clique cover for G_i contains cliques that belong to G_{j_1} or G_{j_2}. We need to ensure not to double count cliques that belong in both G_{j_1} and G_{j_2}.

Lemma 5.9. *Let $S \subseteq E(X_i)$ be the set of already covered edges, and let $R = E(X_i) - S$ be the edges to be covered. Then,*

$$C_i(S) = min\{\, C_{j_1}(S \cup R_2) + C_{j_2}(S \cup R_1) \,:\, \forall R_1 \subseteq R \text{ and } R_2 = R - R_1 \}$$

Proof. Assuming S is already covered, let $\mathscr{W}$ be a minimum clique cover for $G_i(S)$ with cost $C_i(S)$. By definition, any clique that belongs to both G_{j_1} and G_{j_2} must also belong to X_i. Thus, the cliques in $\mathscr{W}$ can be partitioned as $\mathscr{W} = Q_1 \cup Q_2 \cup Q_3$, where

$$Q_1 = \{q \in \mathscr{W} \mid q \subseteq V_{j_1} \text{ and } q \not\subseteq X_i\}$$
$$Q_2 = \{q \in \mathscr{W} \mid q \subseteq V_{j_2} \text{ and } q \not\subseteq X_i\}$$
$$Q_3 = \{q \in \mathscr{W} \mid q \subseteq X_i\}.$$

Thus, $|\mathscr{W}| = |Q_1| + |Q_2| + |Q_3|$. Now, the edgeset R can be partitioned to $R = R_1 \cup R_2$ such that:

$$R_1 = \{e \in R \mid e \text{ covered by } Q_1 \text{ or } Q_3\}$$
$$R_2 = \{e \in R \mid e \text{ covered only by } Q_2\} = R - R_1$$

By definition, $Q_1 \cup Q_3$ needs to cover the edges in R_1. Furthermore, Q_3 needs to cover the edges in $G_{j_1} - E(X_{j_1})$, and thus $|Q_1 \cup Q_3| \geq C_{j_1}(S \cup R_2)$. On the other hand, the cliques in Q_2 only need to cover the edges in R_2 together with $G_{j_2} - E(X_{j_2})$, and thus $|Q_2| \geq C_{j_2}(S \cup R_1)$, and the result follows. □

Therefore, we can compute $C_i(S)$ for any given subset $S \subseteq E(X_i)$. Observe that the recurrence relation looks at all possible bipartitions of R. Since the number of edges in $E(X_i)$ is at most $\binom{k}{2}$, we need to check at most $2^{\binom{k}{2}}$ different partitions of R. Once the bipartition of R is fixed, it takes constant time to look up the values from C_{j_1} and C_{j_2}.

Note that, while these recurrence relations calculate the size of clique covers, they are also constructive: a little bookkeeping at each node will allow us to construct the optimal cover at the root node.

5.2.1 Running Time of Treewidth-based Algorithm

For each node $i \in I$, we compute $C_i(S)$ for every $S \subseteq E(X_i)$. Since $\text{tw}(G) = k$, each node contains at most k vertices. Let ρ denote the maximum number of edges induced in any node. Then we need to consider 2^ρ cases. Then, for each fixed $S \subseteq E(X_i)$, we carry out one of the four recurrence relations. Leaf nodes and Forget nodes can be computed in constant time. An Introduce node can be computed in $O(B(k) \cdot k)$ time, where $B(k)$ is kth Bell number. Finally, a Join node takes $O(2^\rho)$ to compute. Therefore, the dynamic programming algorithm takes $2^\rho \cdot \max\{B(k) \cdot k, 2^\rho\} \cdot O(n) = O(4^\rho n)$ time overall. While ρ can be as large as $\binom{k}{2}$ in theory, this is rarely the case as shown in our experimental tests

(see Section 5.4, Table 5.1).

Theorem 5.10. *There is a linear time algorithm for computing minimum clique cover for graphs with fixed treewidth k.* □

5.2.2 Modifications for Clique Partition

It is straightforward to modify the above algorithm to solve the clique partition problem: instead of assuming that edges already covered can be re-used to form other cliques, we simply delete those edges and solve for remaining edgeset. If we redefine the cost function $C_i(S)$ to be the size of the minimum clique partition for the graph $G_i - S$, the same recurrence holds for each node type. The only difference is when, at an Introduce node, finding a local solution for $\delta(v)$, we look for a clique partition rather than a cover.

5.3 Planar Clique Cover

In this section, we study the clique cover problem restricted to planar graphs, and present a PTAS for planar graphs. While planar graphs are possibly the most restricted class of interest for the clique cover problem (the largest clique being just K_4), the problem remains NP-hard [136]. Furthermore, as treewidth is unbounded in planar graphs [68, 120], simply applying our algorithm from Section 5.2 would result in an exponential running time.

Instead, we shall design an exact polynomial time algorithm for planar graphs with bounded *branchwidth*. As we shall see, the algorithm runs in $O(2^k n)$ time with k being the branchwidth, and since $\mathrm{bw}(G) \leq \sqrt{4.5n}$ when G is planar, this would be the first subexponential algorithm for the clique cover problem on planar graphs.

Then, we will use our exact algorithm to construct a polynomial time approximation scheme. Baker [8] has proposed a divide-and-conquer technique to design approximation schemes for various optimization problems on planar graphs. We will show that her

technique can be applied to the planar clique cover problem, using our exact algorithm as a subroutine, resulting in a $(1+\epsilon)$ approximation algorithm.

5.3.1 Clique Cover for Planar Graphs with Bounded Branchwidth

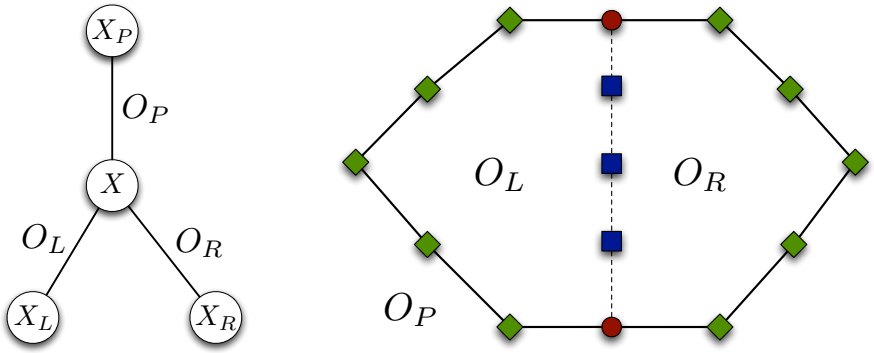
As with its counterpart, treewidth, it is NP-complete to determine if a graph has a branch decomposition of width at most k in general, but this decomposition can be found in linear time when k is fixed. We thus assume that the input graph G is given together with a branch decomposition (T,ϕ) of width at most k. Now pick an arbitrary edge e of T, and subdivide it to create a root node r. Then each tree node X is associated with a subset of edges in $E(G)$, namely the leaf nodes of the subtree rooted at X. We let $E(S)$ denote the edges in the subgraph induced by a subset of vertices S.

Define the middle-set of X, denoted by $\mathrm{mid}(X)$, to be the middle-set of the edge between X and its parent node. Since the root node r has no parent, set $\mathrm{mid}(r)=\emptyset$. Then, we create a table $W_X[\cdot]$ indexed by a subset $\mathscr{F}$ of edges as follows:

$$W_X[\mathscr{F}] = \text{minimum clique cover for edges in } X \text{ with } \mathscr{F} \text{ already covered.}$$

Since $\mathrm{mid}(X)$ is a cutset in G, we can paste together solutions from each subproblem by computing only the entries $W_X[\mathscr{F}]$ where $\mathscr{F}$ is a subset of edges in $E(\mathrm{mid}(X))$.

Before describing the recurrence relation for this table, we study the middle-set of three adjacent edges. Consider the sphere-cut branch decomposition for planar graphs, as studied by Dorn et al. [41]; here, each middle-set defines a closed curve (namely, a *noose*) on the planar embedding of the input graph that intersects only the vertices in the middle-set. Let X be a tree node with two children nodes X_L and X_R, and a parent node X_P. The three edges adjacent to X define 3 middle-sets which we denote by O_P, O_L, O_R for parent edge, left and right child edge, respectively. Since $O_R-(O_P\cup O_L)=\emptyset$ and $O_L-(O_P\cup O_R)=\emptyset$, the vertices of $O_P\cup O_L\cup O_R$ can be partitioned as follows:



(a) (b)

Figure 5.2: A sphere cut decomposition at a node X and its planar embedding. (a) A tree node X has three adjacent edges, each of which defines a middle-set that forms a simple curve called a noose; (b) The middle-set of O_P is drawn as solid lines. Inside the noose O_P are O_L and O_R. Here, portal vertices P are drawn as red circles, intersection vertices are drawn as blue squares, and symmetric difference vertices are drawn as green diamonds.

- Portal vertices $P = O_L \cap O_R \cap O_P$
- Intersection vertices $I = O_L \cap O_R - P$
- Symmetric Difference vertices $D = O_P - (P \cup I)$

Figure 5.2 gives an example of a sphere cut decomposition at a tree node X.

Lemma 5.11. *The table $W_X[\mathscr{F}]$ can be calculated as*

$$W_X[\mathscr{F}] = min\{\, W_{X_1}[\mathscr{F} \cup F_2] + W_{X_2}[\mathscr{F} \cup F_1] \,:\, F_1 \cup F_2 \text{ is a partition of } E(I \cup P)\,\}$$

Proof. For an arbitrary clique cover for X with $\mathscr{F}$ already covered, consider the cliques covering the edges $E(I \cup P)$. Observe that, by planarity of G, any clique intersecting with $I \cup P$ only contains either vertices of X_1 or vertices of X_2. Therefore, we can partition the edges in $E(I \cup P)$ into F_1 and F_2, where F_1 is to be covered by cliques in X_1, and F_2 is covered by cliques in X_2. For each partition, the solution from the subproblem $W_{X_1}[\mathscr{F} \cup F_2]$ together with the solution from $W_{X_2}[\mathscr{F} \cup F_1]$ gives the solution for $W_X[\mathscr{F}]$. Since we consider all possible partitions of $E(I \cup P)$, the lemma follows. □

The algorithm runs in a bottom-up manner. Observe that to compute each state $\mathscr{F}$ of a node, we need to consider all bipartitions of $E(I \cup P)$. Since $I \cup P \subseteq \text{mid}(X_1)$, $|E(I \cup P)| = O(k)$, and thus there are $2^{O(k)}$ such partitions. Moreover, to compute for all states $\mathscr{F}$, we consider $2^{O(k)}$ subsets of edges in $E(\text{mid}(X))$. Altogether, the above dynamic programming algorithm runs in $2^{O(k)}O(n)$ time.

Lemma 5.12. *There is a linear time algorithm to compute a minimum clique cover for planar graphs with bounded branchwidth.* □

5.3.2 Modifications for Clique Partition

As with our treewidth-based algorithm, the above algorithm can be easily modified to solve the clique partition problem: rather than assuming $\mathscr{F}$ is already covered, one can simply delete those edges and solve for the remaining edges.

5.3.3 Baker's Technique on Planar Graphs

Baker [8] has proposed a general approach to design approximation algorithms for various NP-hard problems on planar graphs. Here, we show how this technique, together with our exact algorithm in Section 5.3.1, results in a $(1+\epsilon)$ approximation algorithm.

Baker's technique is a divide-and-conquer approach, where the input graph is decomposed into *layers* of subgraphs defined by the distance from a chosen vertex. Applying this technique to the planar clique cover problem, we obtain Algorithm 5.3.1. In short, we pick an arbitrary vertex r, and define the level of each vertex of $G - \{r\}$ as distance from r. Then, we slice up the graph into layers of subgraphs, where each subgraph is solved independently. The clique covers for the layers are then merged to construct a solution for the entire graph G. The edges on the boundary of each layer are covered by cliques from adjacent layers, which can be bounded to obtain an approximation ratio. See Algorithm 5.3.1 for details, and Figure 5.3 shows a schematic of the approach.

Algorithm 5.3.1: PTAS for Planar Clique Cover

Input: A planar graph $G=(V,E)$, $0<\epsilon<1$
Output: A clique cover C of G.

$r \leftarrow$ an arbitrary vertex of G;
foreach *vertex* $v \in V \setminus \{r\}$ **do**
  level(v) $\leftarrow$ distance from r;
end
$k \leftarrow \lceil 2/\epsilon \rceil$;
for $i = 0,1,\ldots,k-1$ **do**
  for $j = 0,1,2\ldots$ **do**
    $G_{ij} \leftarrow$ subgraph of G induced by the vertices at levels $jk+1$ through $(j+1)k+i$;
    $C_{ij} \leftarrow$ minimum clique cover for G_{ij};
  end
end
for $i = 0,1,\ldots,k-1$ **do**
  $C_i \leftarrow \bigcup_j C_{ij}$.
end
Return a clique cover from $\{C_0,\ldots,C_{k-1}\}$ with the minimum size.

Lemma 5.13. *Algorithm 5.3.1 finds a clique cover of weight at most* $(1+\epsilon)$ *OPT.*

Proof. Let Q^* denote the optimum solution, and given some congruence class $i \mod k$, let us assume that the above approximation scheme divides the problem into m pieces. Since the solution for each piece is solved exactly, we have

$$|C_i| = \sum_{j=1}^{m} |C_{ij}| \leq \sum_{j=1}^{m} |Q^*_{ij}|,$$

where Q^*_{ij} denotes the optimum solution restricted to the graph G_{ij}. We therefore need to compare the two values $\sum_{j=1}^{m} |Q^*_{ij}|$ and $|Q^*|$. To compare these, consider the number of cliques in Q^* that contain vertices at levels $i \mod k$. For planar graphs, each clique belongs to at most 2 consecutive levels. Therefore, for at least one value of i, $0 \leq i \leq k$, there are at most $\lceil \frac{2}{k} \rceil |Q^*|$ cliques containing a vertex at level $i \mod k$. Since these cliques

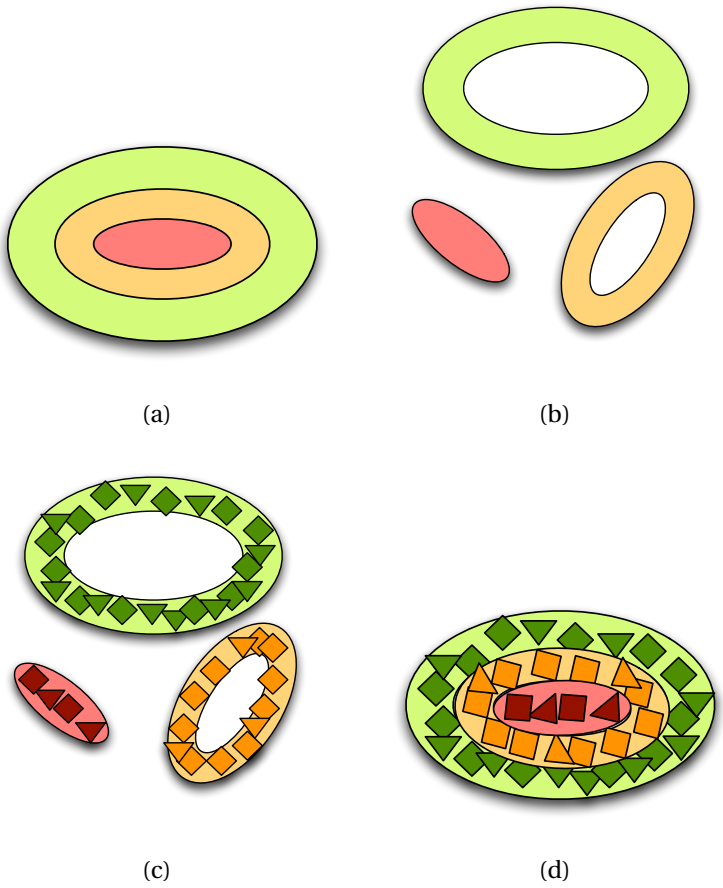


Figure 5.3: Planar clique cover using Baker's technique. (a) For each value of $0 \leq i \leq k-1$, the graph is sliced into layers; (b) Each layer is treated as an independent subproblem; (c) Each layer is solved using our clique cover algorithm on planar graphs with bounded branchwidth; (d) The solutions from the layers are pasted together. The number of edges doubly-covered by cliques from adjacent layers can be bounded to give the approximation ratio.

are double counted in the sum $\sum_{j=1}^{m} |Q_{ij}^*|$, we have

$$\sum_{i=1}^{m} |Q_i^*| \leq (1+\frac{2}{k})|Q^*|$$

and it follows that $|C_i| \leq (1+\frac{2}{k})|Q^*|$. □

What remains now is an algorithm to compute the clique cover for each G_{ij}. Note that Lemma 5.12 provides an algorithm for planar graphs with bounded branchwidth, and thus it suffices to show that each G_{ij} also has bounded branchwidth. Tamaki's the-

orem [132] directly provides an answer.

Definition 5.14. *Given a plane-embedded graph G, the* face-incidence graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *of G consists of vertices $\mathcal{V}$ from faces of G, and two vertices in $\mathcal{V}$ are joined by an edge if and only if the corresponding faces in G share a vertex.*

Theorem 5.15. [132] *Given a planar graph G, there is a linear time algorithm that finds a branch decomposition with width at most the radius of the face-incidence graph of G.*

Since the face-incidence graph of G_{ij} has a bounded radius, each G_{ij} is a planar graph with bounded branchwidth. Therefore, our algorithm from Lemma 5.12 can compute the minimum clique cover for the subgraph in each G_{ij}.

Recall that our dynamic programming algorithm for graphs with branchwidth $\leq k$ runs in time $2^{O(k)}O(n)$. Due to Theorem 5.6, directly applying this algorithm to planar graphs gives an *exact* solution in time $2^{O(\sqrt{n})}O(n)$. On the other hand, the PTAS using Baker's technique involves decomposing the graph into n/k layers, and solving each piece exactly in time $2^{O(k)}O(n/k)$. Trying for every congruent classes between 1 and $k-1$, the overall algorithm takes $k \cdot \lceil \frac{n}{k} \rceil \cdot 2^{O(k)}O(n/k) \approx 2^{O(k)}O(\frac{n^2}{k})$ to obtain a solution of value at most $(1+\frac{2}{k})OPT$. Therefore, while our PTAS provides an approximation with varying degree of approximation ratio, if one wants to get a solution any closer than $(1+\frac{1}{\sqrt{n}})OPT$, one may be better off running our dynamic programming algorithm directly to the graph to obtain an exact solution. This trade-off is clearly shown empirically in Table 5.2.

Theorem 5.16. *There is a* PTAS *for planar clique cover.* □

5.4 Experimental Results

We have implemented the described algorithms and tested them against both real biological PPI data and simulated data. As a comparison, we considered Gramm et al.'s algorithm [66] for the decision problem version of clique cover: given the size of clique

cover k as an input parameter, their algorithm works by first applying a set of reduction rules to reduce the problem instance, and then using a search tree algorithm on the reduced instance, in time exponential in k. While their algorithm works well in cases where the solution size is small, it performed poorly against our test data which contains several hundreds of cliques. This is mainly because their reduction rules did not significantly decrease the size of our input graphs (especially biological networks): the solutions for the reduced instances would still contain a large number of cliques, resulting in inefficient running time for the search tree algorithm. As their input parameter is the size of the minimum clique cover, it motivates our development of approaches using edge sparsity for these networks.

5.4.1 Simulated and Biological Networks

Each of our algorithms was tested on both simulated and actual biological networks. First, Krogan et al. [93] obtained an extensive dataset on yeast protein interactions. Taking the largest connected component from the dataset, with 323 vertices and 742 edges, we formed a model network, G_{Krogan}. Various studies have shown that PPI networks exhibit the properties of scale-free networks [10]. Many generative models for scale-free networks have also been proposed, and we used the two most frequently used models [10, 32] to create test graphs for our algorithm: (1) preferential attachment model (denoted by G_{PAM}), and (2) duplication model (denoted by G_{DM}). In both cases, we set the parameters of generative models so that the resulting networks show similar characteristics to that of real PPI data; density of $|E| \approx 2|V|$, and degree distribution $P(k) \sim k^{-\gamma}$ where $\gamma \approx 1.7$.

To investigate the behaviour of our algorithms on denser graphs, we generated a set of partial k-trees. Recall that a k-tree is a maximal graph with treewidth k such that no edge can be inserted without increasing its treewidth, and a graph is a partial k-tree if it is a subgraph of a k-tree. The partial k-trees of given treewidth have been generated by first generating a k-tree, and randomly removing edges to obtain desired edge density.

	n	m	tw	ρ	tree decomp. (hh:mm)	clique cover algo. (hh:mm)	# of cliques
Krogan	323	742	11	21	2:13	6:08	482
PAM	300	634	14	29	2:41	9:21	421
DM	300	794	21	43	4:13	17:47	583

Table 5.1: Performance of treewidth-based exact clique cover algorithm; we show the treewidth of each graph, maximum # of edges per tree node (ρ), time taken to compute an optimal tree decomposition and a clique cover, and size of the solution.

Since the generation process of k-trees is similar to that of the preferential attachment model, this allows us to create graphs with higher density than G_{PAM} while preserving low treewidth.

5.4.2 Performance of the Treewidth and Branchwidth-based Algorithms

Table 5.1 reports results obtained on real and simulated biological networks. Both Krogan's PPI network and simulated networks exhibit relatively low treewidths for their size. Figure 5.4 gives a more comprehensive view of the running times from empirical testing. As expected, the running time increases linearly with n for graphs with fixed treewidths (Figure 5.4(a)), but exponentially with treewidth k for graphs with fixed n (Figure 5.4(b)). Running times for partial k-trees (Figures 5.4(c) and (d)) follow the same trends, although they are somewhat higher due to the higher edge density.

While our branchwidth-based exact algorithm was designed for planar graphs, it is easy to modify the algorithm to handle non-planar graphs with bounded branchwidth (but at the expense of higher time complexity due to non-planarity). Figure 5.5 shows that, in practice, the running time of the treewidth-based algorithm grows slower than that of the branchwidth-based algorithm, allowing it to handle graphs with larger treewidths.

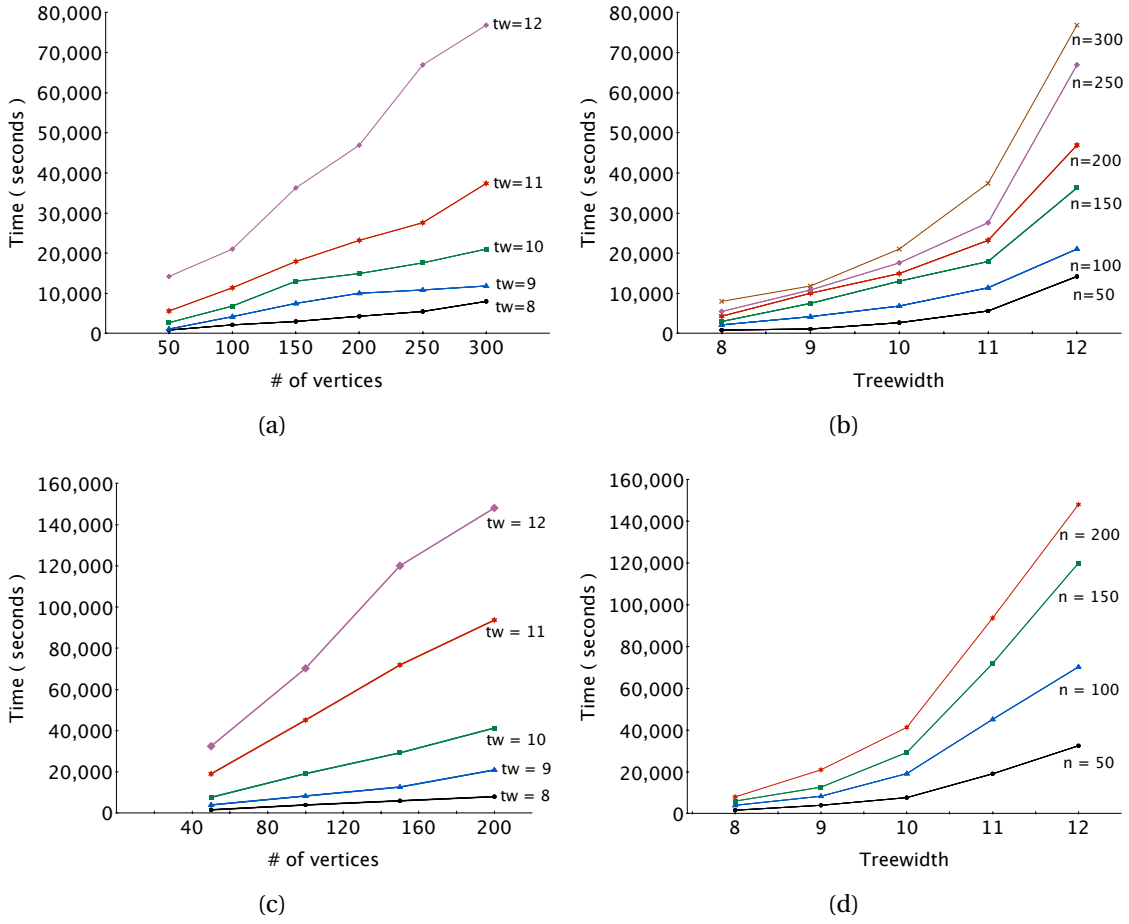
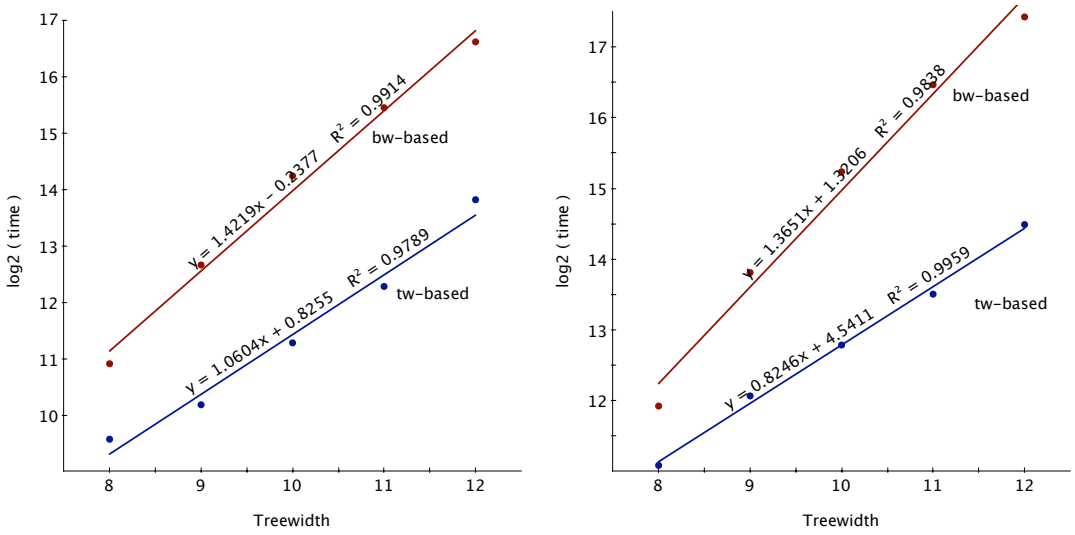


Figure 5.4: Performance of the treewidth-based algorithm on simulated networks: (a) scale-free networks (PAM) with fixed treewidth; (b) scale-free networks (PAM) with fixed graph size; (c) partial k-trees with fixed treewidth; (d) partial k-trees with fixed graph size.



(a) (b)

Figure 5.5: Performance comparison of treewidth-based algorithm vs. branchwidth-based algorithm on scale-free networks; y-axes are shown in log scale to exemplify the difference in exponents in the running time. (a) scale-free networks with 50 vertices; (b) scale-free networks with 100 vertices; data points with the same treewidth values in each chart are taken from the same network.

5.4.3 Performance of PTAS for Planar Graphs

To test our PTAS on planar graphs, we generated a set of random planar graphs using the simple algorithm by Denise and Vasconcellos [40]. Then, we ran both our exact branchwidth-based algorithm and the PTAS with varying values of ϵ to explore the trade-off between running time and quality of the solution. As shown in Table 5.2, the promised approximation ratios are almost exactly realized and substantial speed-ups are obtained for relatively large values of ϵ, compared to the exact branch-decomposition based algorithm. The running time increases exponentially with $1/\epsilon$, and the exact algorithm starts to become faster than the PTAS when ϵ becomes sufficiently small.

5.4.4 Clique Cover in Biological Networks

When executed on the yeast protein-protein interaction network of Krogan et al. [93], our treewidth-based algorithm finds a clique cover that includes 93 cliques of size 5 or

ϵ	# of cliques	time (hh:mm:ss)
0.5	159	00:12:08
0.2	124	00:39:17
0.1	116	02:36:18
0.05	113	03:53:42
OPT	109	02:46:31

Table 5.2: Performance of PTAS for planar graph with $n = 200$, $m = 407$, $tw = 10$. OPT was obtained using the branch-decomposition based algorithm.

more. While the PPI data may not admit a unique clique cover on the network, we manually verified that most discovered cliques correspond to known complexes, such as the RNA polymerase II, the RSC complex, the mediator complex, and the 20S proteasome. In addition, three highly overlapping complexes, SWR1 (a chromatin remodelling complex), NuA4 (a histone acetyltransferase complex), and INO80 (another chromatin remodelling complex) are correctly identified, despite the fact that SWR1 and NuA4 share three subunits (ARP4, GOD1, and YAF9) and SWR1 and INO80 share four (ARP4, GOD1, RVB1, RVB2). This suggests that our algorithm is capable of identifying biologically relevant protein complexes, even those that share a significant number of subunits.

5.5 Discussions

In this chapter, we studied the clique cover problem on sparse networks as measured by treewidth and branchwidth, with an application to protein-complex discovery in PPI networks. We gave exact polynomial-time algorithms for graphs with bounded treewidth and bounded branchwidth, and built on the latter using Baker's technique to obtain a polynomial time approximation scheme for planar graphs.

Our empirical studies show that the biological networks as well as synthetic networks with similar characteristics (e.g. edge density, degree distribution) indeed exhibit low treewidth, and our algorithms showed practical running times on these test networks. Moreover, our branchwidth based PTAS algorithm shows practical running time for computing solutions close to the optimal.

In proteomics research, existing experimental methods for detecting binary interactions often suffer from false negatives, i.e., some edges are not detected in the experiments. Therefore, in the direction towards hypergraph modelling of PPI networks, one may wish to cover the edges with quasi-cliques: Here, the optimization function needs to be modified slightly. One possible formulation may be to find a minimum cardinality quasi-clique cover, where a quasi-clique is defined by some lower bound on the edge density. On the other hand, there may be classes of graphs other than the ones discussed here that admit polynomial time exact algorithms, for example, graphs with bounded genus or bounded degree.

5.6 Bibliographic Notes

The results discussed in this chapter have been published in [14].

Chapter 6

Conclusion and Future Directions

Researchers in systems biology strive to uncover complex interactions in biological systems, and protein-protein interactions lie at the heart of their efforts. As opposed to the classical reductionist paradigm, systems biologists consider biological phenomena as a complex system, and thus high throughput experimental technologies are vital to their studies. Indeed, the recent development of high throughput techniques has provided a huge momentum – the amount of PPI data available is rapidly growing, and large-scale studies of the human proteome are also underway.

Unfortunately, high throughput technologies remain in their infancy, and produce a large amount of data with relatively poor accuracy. As a result, the accumulating data in the literature presents us with both an opportunity and a challenge. While the analysis of the PPI networks provides invaluable information on the inner workings of the cell, the significant amount of noise within the data makes it difficult to handle. AP-MS is a good example of such a technology that suffers from noisy output data.

Noise in proteomics data can often be classified as two distinct types [60]: *stochastic errors* and *systematic errors*. Stochastic errors are measurement errors with random variability, which can be reduced by simply repeating the experiment. Systematic errors, on the other hand, are recurrent in the measurements, and thus require a careful

modelling of every step of the experiment in order to correctly interpret the data.

In this thesis, we considered the overall pipeline of the AP-MS method, and looked for sources of systematic errors. Here we highlight the novel contributions put forward in this thesis, and discuss future research directions.

Protein Quantification with Shared Peptides. Chapter 3 is concerned with protein quantification from quantitative MS data. While significant progress has been made, the problem of protein identification (and quantification) still remains unresolved due to shared peptides prevalent in many MS datasets [108]. We proposed several approaches to protein quantification in the presence of shared peptides by defining a set of optimization problems based on a clean combinatorial model.

One of the limitations in our approach comes from the assumption that every peptide can be detected by MS equally well – and thus the errors in the peptide abundances are equally likely. While we have empirically tested the robustness of our algorithms by varying the coverage of the data (see Section 3.4.1), an approach that directly models the noisy MS data would likely yield more practical results.

Open Problem 6.1. *Does there exist an approach for protein quantification with shared peptides that incorporates peptide detectability?*

Peptide detectability may be estimated either using a suitably large MS/MS dataset [2, 133] or using the physical properties of each peptide [99]. In Section 3.5, we discussed possible ways to extend our combinatorial approach by incorporating peptide detectability as weighted errors. Alternatively, we may also attack this problem using a probabilistic framework where observed peptide abundances are thought to be random variables that incorporate detectability. Then, we can look for protein abundances maximizing the probability of observing the given peptide abundances using various Bayesian approaches.

Predicting Direct PPI Network. In Chapter 4, we studied the problem of distinguishing direct interactions from indirect ones within the PPI data from AP-MS experiments. We tackled this problem by proposing a probabilistic graph model, and formulated the DIGCOM problems to identify the direct interaction network from quantitative PPI data. Our algorithm consists of three main phases: (1) Identify weakly connected vertices (and their neighbours); (2) Discover dense clusters in the network; (3) Identify remaining direct interactions via a genetic algorithm.

As discussed in Section 4.5, the DIGCOM problems raise a number of challenging computational problems to investigate further. From the standpoint of computational complexity, the hardness of the problems are not yet known, and we conjecture that they are NP-hard.

Open Problem 6.2. *What is the computational complexity of* E-DIGCOM *and* A-DIGCOM*?*

On the other hand, the probabilistic graph model is built around several assumptions. For example, the abundance of protein complexes is not constant, and the strength of all physical interactions is non-uniform as some interactions may be more prone to disruption by the affinity purification process than others.

Open Problem 6.3. *Given sufficient AP-MS data, can we generalize the* DIGCOM *problems to handle individual interaction strengths and abundances?*

In fact, if we were given the individual interaction strength for each pair of proteins (possibly from a complementary dataset such as protein co-crystallization or physical models), the second and third phases of our algorithm can be easily modified to handle non-uniform interaction probabilities. The first phase of the algorithm would require more careful modifications to handle arbitrary survival probability for each edge.

PPI Networks as Hypergraphs. In Chapter 5, we considered the problem of modelling PPI networks as hypergraphs via the (edge) clique cover problem on the binary PPI network. In particular, we devised an exact algorithm for graphs with bounded treewidth

which showed promising performance on both simulated data and real PPI data. Furthermore, our theoretical pursuit on clique cover for planar graphs with bounded branch-width resulted in a PTAS for planar graphs.

However, existing experimental techniques for detecting binary interactions often suffer from false negatives, i.e., some edges are missing in the PPI network. Consequently, protein complexes that should ideally form cliques instead appear as quasi-cliques or simply dense subgraphs. In the direction towards modelling PPI networks as hypergraphs, therefore, we may wish to relax the assumption that each protein complex forms a clique.

Open Problem 6.4. *Does there exist a quasi-clique cover algorithm?*

Quasi-cliques can be defined in various ways: one possible definition of a quasi-clique is a dense subgraph with a lower bound on the edge density. Using this definition, one may generalize our algorithm for graphs with bounded treewidth. Following our algorithm, the same type of dynamic programming algorithm can be conceived – however, our approach of finding cliques in the local neighbourhood of a vertex should now be extended to multi-hop neighbours depending on the edge density, resulting in an increase in the running time. On the other hand, one may devise an algorithm specialized for graphs with other properties of PPI networks, for example the scale-freeness of networks.

There are many problems in PPI networks to pursue further studies, and the work presented in this thesis opens the door to several opportunities with important implications in systems biology. One of the main pursuits in the field is a comprehensive understanding of the proteome as a complex dynamic system, and any improvements in tackling the problems discussed here would take us one step closer towards this goal.

Bibliography

[1] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics (Oxford, England)*, 22(8):1021–1023, Apr. 2006.

[2] P. Alves, R. J. Arnold, M. V. Novotny, P. Radivojac, J. P. Reilly, and H. Tang. Advancement in protein inference from shotgun proteomics using peptide detectability. *Pacific Symposium on Biocomputing*, pages 409–420, 2007.

[3] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *Society for Industrial and Applied Mathematics. Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[4] S. Arora. *Computational Complexity: A Modern Approach*. Cambridge University Press, 1 edition, 2009.

[5] S. Asthana. Predicting protein complex membership using probabilistic network reliability. *Genome Research*, 14(6):1170–1175, 2004.

[6] J. Azé, T. Bourquard, S. Hamel, A. Poupon, and D. Ritchie. Using Kendall-Tau Meta-Bagging to Improve Protein-Protein Docking Predictions. In *Lecture Notes in Bioinformatics 7036, PRIB 2011*, pages 284–295, 2011.

[7] G. D. Bader and C. W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC bioinformatics*, 4:2, Jan. 2003.

[8] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1), 1994.

[9] M. Bantscheff, M. Schirle, G. Sweetman, J. Rick, and B. Kuster. Quantitative mass spectrometry in proteomics: a critical review. *Analytical and bioanalytical chemistry*, 389(4):1017–1031, Oct. 2007.

[10] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286(5439):509–512, Oct. 1999.

[11] P. L. Bartel, J. A. Roecklein, D. SenGupta, and S. Fields. A protein linkage map of Escherichia coli bacteriophage T7. *Nature Genetics*, 12(1):72–77, Jan. 1996.

[12] H. M. Berman, T. N. Bhat, P. E. Bourne, Z. Feng, G. Gilliland, H. Weissig, and J. Westbrook. The protein data bank and the challenge of structural genomics. *Nature structural biology*, 7 Suppl:957–959, Nov. 2000.

[13] A. Bhan, D. Galas, and T. Dewey. A duplication growth model of gene expression networks. *Bioinformatics*, 18:1486–1493, 2002.

[14] M. Blanchette, E. Kim, and A. Vetta. Clique Cover on Sparse Networks. In *The 9th SIAM Meeting on Algorithm Engineering & Experiments (ALENEX), Kyoto, Japan*, 2012.

[15] M. Blatt, S. Wiseman, and E. Domany. Superparamagnetic Clustering of Data. *Phys. Rev. Lett.*, 76:3251–3254, Apr 1996.

[16] H. Bodlaender and T. Kloks. A simple linear time algorithm for triangulating three-colored graphs. *Journal of Algorithms*, 15(1):160–172, 1993.

[17] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[18] H. L. Bodlaender, M. R. Fellows, M. T. Hallett, H. T. Wareham, and T. J. Warnow. The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. *Theoretical Computer Science*, 244(1-2):167–188, 2000.

[19] A. Breitkreutz, H. Choi, J. Sharom, L. Boucher, V. Neduva, B. Larsen, Z.Y.Lin, B. Breitkreutz, C. Stark, G. Liu, J. Ahn, D. Dewar-Darch, Z. Qin, T. Pawson, A. Gingras, A. I. Nesvizhskii, and M. Tyers. Global architecture of the yeast kinome interaction network. *Science*, 2010.

[20] S. Brohée and J. van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC bioinformatics*, 7:488, 2006.

[21] H. Brönnimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14:463–479, 1995. 10.1007/BF02570718.

[22] P. C. Carvalho, J. Hewel, V. C. Barbosa, and J. R. Yates. Identifying differences in protein expression levels by spectral counting and feature selection. *Genetics and molecular research : GMR*, 7(2):342–356, 2008.

[23] M. R. Cerioli, L. Faria, T. O. Ferreira, C. A. J. Martinhon, F. Protti, and B. Reed. Partition into cliques for cubic graphs: planar case, complexity and approximation. *Discrete Applied Mathematics*, 156(12):2270–2278, 2008.

[24] M.-S. Chang and H. Müller. On the tree-degree of graphs. In *Graph-theoretic Concepts in Computer Science (Boltenhagen, 2001)*, pages 44–54, 2001.

[25] C. Chekuri, K. L. Clarkson, and S. Har-Peled. On the set multi-cover problem in geometric settings. In *Proceedings of the 25th Annual Symposium on Computational geometry*, SCG '09, pages 341–350, New York, NY, USA, 2009. ACM.

[26] J. Chen, W. Hsu, M. L. Lee, and S. Ng. Increasing confidence of protein interactomes using network topological metrics. *Bioinformatics*, 22(16):1998–2004, Aug. 2006.

[27] J. Chen, W. Hsu, M. L. Lee, and S.-K. Ng. Systematic assessment of high-throughput experimental data for reliable protein interactions using network topology. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI '04, pages 368–372, Washington, DC, USA, 2004. IEEE Computer Society.

[28] Q. Cheng, P. Berman, R. Harrison, and A. Zelikovsky. Efficient alignments of metabolic networks with bounded treewidth. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 687–694, 2010.

[29] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):pp. 493–507, 1952.

[30] H. Choi, D. Fermin, and A. I. Nesvizhskii. Significance analysis of spectral count data in label-free shotgun proteomics. *Molecular & Cellular Proteomics*, 7(12):2373–2385, December 2008.

[31] V. Choi. Yucca: an efficient algorithm for small-molecule docking. *Chemistry & biodiversity*, 2(11):1517–1524, Nov. 2005.

[32] F. Chung, L. Lu, T. G. Dewey, and D. J. Galas. Duplication models for biological networks. *Journal of Computational Biology*, 10(5):677–687, Oct. 2003.

[33] P. Cloutier, R. Al-Khoury, M. Lavallée-Adam, D. Faubert, H. Jiang, C. Poitras, A. Bouchard, D. Forget, M. Blanchette, and B. Coulombe. High-resolution mapping of the protein interaction network for the human transcription machinery and affinity purification of RNA polymerase II-associated complexes. *Methods (San Diego, Calif.)*, 48(4):381–386, Aug. 2009.

[34] C. J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, Inc., 1987.

[35] S. R. Collins, P. Kemmeren, X.-C. Zhao, J. F. Greenblatt, F. Spencer, F. C. P. Holstege, J. S. Weissman, and N. J. Krogan. Toward a comprehensive atlas of the physical interactome of Saccharomyces cerevisiae. *Molecular & cellular proteomics : MCP*, 6(3):439–450, Mar. 2007.

[36] B. Coulombe, M. Blanchette, and C. Jeronimo. Steps towards a repertoire of comprehensive maps of human protein interaction networks: the Human Proteotheque Initiative (HuPI). *Biochemistry and Cell Biology*, 86(2):149–156, Apr. 2008.

[37] T. Dandekar, B. Snel, M. Huynen, and P. Bork. Conservation of gene order: a fingerprint of proteins that physically interact. *Trends in Biochemical Sciences*, 23(9):324 – 328, 1998.

[38] B. Dengiz, F. Altiparmak, and A. Smith. Efficient optimization of all-terminal reliable networks, using an evolutionary approach. *Reliability, IEEE Transactions on*, 46(1):18–26, 1997.

[39] B. Dengiz, F. Altiparmak, and A. Smith. Local search genetic algorithm for optimal design of reliable networks. *Evolutionary Computation, IEEE Transactions on*, 1(3):179–188, 1997.

[40] A. Denise and M. Vasconcellos. The random planar graph. *Congressus Numerantium*, 113:61–79, 1996.

[41] F. Dorn, E. Penninkx, H. Bodlaender, and F. Fomin. Efficient exact algorithms on planar graphs: exploiting sphere cut branch decompositions. *Algorithms–ESA 2005*, 3669:95–106, 2005.

[42] B. Dost, N. Bandeira, X. Li, Z. Shen, S. P. Briggs, and V. Bafna. Accurate mass spectrometry based protein quantification via shared peptides. *Journal of Computational Biology*, 19, 2012.

[43] B. Dost, T. Shlomi, N. Gupta, E. Ruppin, V. Bafna, and R. Sharan. QNet: a tool for querying protein interaction networks. *Journal of Computational Biology*, 15(7):913–925, 2008.

[44] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, Dec. 1998.

[45] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *J. Graph Algorithms and Applications*, 3(3):1–27, 1999.

[46] D. Eppstein. Diameter and treewidth in minor-closed graph famillies. *Algorithmica*, 27:275–291, 2000.

[47] G. Even, D. Rawitz, and S. Shahar. Hitting sets when the VC-dimension is small. *Inf. Process. Lett.*, 95(2):358–362, July 2005.

[48] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[49] S. Fields and S. O. A novel genetic system to detect protein-protein interactions. *Nature*, 340(6230):245–6, 1989.

[50] R. L. Finley and R. Brent. Interaction mating reveals binary and ternary connections between Drosophila cell cycle regulators. *Proceedings of the National Academy of Sciences of the United States of America*, 91(26):12980–12984, Dec. 1994.

[51] L. Florens, M. Washburn, J. Raine, R. Anthony, M. Grainger, J. Haynes, J. Moch, N. Muster, J. Sacci, D. Tabb, and et al. A proteomic view of the Plasmodium falciparum life cycle. *Nature*, 419(6906):520–526, 2002.

[52] F. V. Fomin and D. M. Thilikos. New upper bounds on the decomposability of planar graphs. *Journal of Graph Theory*, 51(1):53–81, 2006.

[53] M. Fromont-Racine, J. Rain, and P. Legrain. Toward a functional analysis of the yeast genome through exhaustive two-hybrid screens. *Nat Genet*, 16(3):277–282, July 1997.

[54] A. Galarneau, M. Primeau, L.-E. Trudeau, and S. W. Michnick. β-Lactamase protein fragment complementation assays as in vivo and in vitro sensors of protein-protein interactions. *Nature Biotechnology*, 20(6):619–622, June 2002.

[55] J. Gao, G. Opiteck, M. Friedrichs, A. Dongre, and S. Hefta. Changes in the protein expression of yeast as a function of carbon source. *J Proteome Res*, 2(6):643–649, 2003.

[56] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. WH Freman,1979, 1979.

[57] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[58] A. Gavin, M. Bösche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, J. M. Rick, A. Michon, C. Cruciat, M. Remor, C. Höfert, M. Schelder, M. Brajenovic, H. Ruffner, A. Merino, K. Klein, M. Hudak, D. Dickson, T. Rudi, V. Gnau, A. Bauch, S. Bastuck, B. Huhse, C. Leutwein, M. Heurtier, R. R. Copley, A. Edelmann, E. Querfurth, V. Rybin, G. Drewes, M. Raida, T. Bouwmeester, P. Bork, B. Seraphin, B. Kuster, G. Neubauer, and G. Superti-Furga. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415(6868):141–147, 2002.

[59] L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant. Open mass spectrometry search algorithm. *Journal of proteome research*, 3(5):958–964, Aug. 2004.

[60] R. Gentleman and W. Huber. Making the most of high-throughput protein-interaction data. *Genome Biology*, 8(10):112, 2007.

[61] S. A. Gerber, J. Rush, O. Stemman, M. W. Kirschner, and S. P. Gygi. Absolute quantification of proteins and phosphoproteins from cell lysates by tandem MS. *Proceedings of the National Academy of Sciences*, 100(12):6940–6945, 2003.

[62] E. Gilbert. Enumeration of labeled graphs. *Canad. J. Math*, 8:405–411, 1956.

[63] J. Gilmore, D. Auberry, J. Sharp, A. White, K. Anderson, and D. Daly. A bayesian estimator of protein-protein association probabilities. *Bioinformatics*, 24(13):1554–5, 2008.

[64] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989.

[65] R. Gordân, A. J. Hartemink, and M. L. Bulyk. Distinguishing direct versus indirect transcription factor-DNA interactions. *Genome Research*, 19(11):2090–2100, Nov. 2009.

[66] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction and exact algorithms for clique cover. *ACM Journal of Experimental Algorithmics*, 13, 2009.

[67] J. Gross and J. Yellen. *Handbook of Graph Theory (Discrete Mathematics and its Applications)*. CRC, 2003.

[68] R. Halin. *S*-functions for graphs. *Journal of Geometry*, 8:171–186, 1976. 10.1007/BF01917434.

[69] E. Hartuv, A. O. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, and R. Shamir. An algorithm for clustering cDNA fingerprints. *Genomics*, 66(3):249 – 256, 2000.

[70] Y. Ho, A. Gruhler, A. Heilbut, G. D. Bader, L. Moore, S. Adams, A. Millar, P. Taylor, K. Bennett, K. Boutilier, L. Yang, C. Wolting, I. Donaldson, S. Schandorff, J. Shewnarane, M. Vo, J. Taggart, M. Goudreault, B. Muskat, C. Alfarano, D. Dewar, Z. Lin, K. Michalickova, A. R. Willems, H. Sassi, P. A. Nielsen, K. J. Rasmussen, J. R. Andersen, L. E. Johansen, L. H. Hansen, H. Jespersen, A. Podtelejnikov, E. Nielsen, J. Crawford, V. Poulsen, B. D. Sørensen, J. Matthiesen, R. C. Hendrickson, F. Gleeson, T. Pawson, M. F. Moran, D. Durocher, M. Mann, C. W. V. Hogue, D. Figeys, and M. Tyers. Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry. *Nature*, 415(6868):180–183, 2002.

[71] D. N. Hoover. Complexity of graph covering problems for graphs of low degree. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 11:187–208, 1992.

[72] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, and R. E. Stearns. The complexity of planar counting problems. *SIAM Journal on Computing*, 27(4):1142–1167 (electronic), 1998.

[73] Y. Ishihama, Y. Oda, T. Tabata, T. Sato, T. Nagasu, J. Rappsilber, and M. Mann. Exponentially modified protein abundance index ($emPAI$) for estimation of absolute protein amount in proteomics by the number of sequenced peptides per protein. *Mol Cell Proteomics*, 4(9):1265–1272, 2005.

[74] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences of the United States of America*, 98(8):4569–4574, 2001.

[75] T. Ito, K. Tashiro, S. Muta, R. Ozawa, T. Chiba, M. Nishizawa, K. Yamamoto, S. Kuhara, and Y. Sakaki. Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 97(3):1143–1147, Feb. 2000.

[76] R.-H. Jan, F.-J. Hwang, and S.-T. Chen. Topological optimization of a communication network subject to a reliability constraint. *Reliability, IEEE Transactions on*, 42(1):63–70, 1993.

[77] J. Janin, K. Henrick, J. Moult, L. T. Eyck, M. J. E. Sternberg, S. Vajda, I. Vakser, S. J. Wodak, and Critical Assessment of PRedicted Interactions. CAPRI: a Critical Assessment of PRedicted Interactions. *Proteins*, 52(1):2–9, July 2003.

[78] L. J. Jensen and P. Bork. Not comparable, but complementary. *Science*, 322(5898):56–57, 2008.

[79] H. Jeong, S. P. Mason, A. L. Barabasi, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 05 2001.

[80] R. Jin, S. Mccallen, C.-C. Liu, Y. Xiang, E. Almaas, and X. J. Zhou. Identifying dynamic network modules with temporal and spatial constraints. In *Pacific Symposium on Biocomputing*, pages 203–214, 2009.

[81] S. Jin, D. Daly, D. Springer, and J. Miller. The effects of shared peptides on protein quantitation in label-free proteomics by LC/MS/MS. *Journal of Proteome Research*, 7(1):164–169, 2008.

[82] R. Jothi and P. T. M. Computational approaches to predict protein-protein and domain-domain interactions. In M. I. and Z. A., editors, *Bioinformatics Algorithms: Techniques and Applications*. Wiley Press, 2008.

[83] V. Kann. *On the Approximability of NP-complete Optimization Problems*. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, 1992.

[84] R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad and spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[85] M. Karas and F. Hillenkamp. Laser desorption ionization of proteins with molecular masses exceeding 10,000 daltons. *Analytical Chemistry*, 60(20):2299–2301, 1988.

[86] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 40(4):85–103, 1972.

[87] S. M. Khan, B. Franke-Fayard, G. R. Mair, E. Lasonder, C. J. Janse, M. Mann, and A. P. Waters. Proteome analysis of separated male and female gametocytes reveals novel sex-specific Plasmodium biology. *Cell*, 121(5):675–687, June 2005.

[88] E. Kim, A. Sabharwal, A. Vetta, and M. Blanchette. Predicting direct protein interactions from affinity purification mass spectrometry data. *Algorithms for Molecular Biology*, 5(1):34, 2010.

[89] E. Kim, A. Vetta, and M. Blanchette. Protein quantification with shared peptides using multi cover algorithms. *in preparation*, 2011.

[90] A. D. King, N. Przulj, and I. Jurisica. Protein complex prediction via cost-based clustering. *Bioinformatics (Oxford, England)*, 20(17):3013–3020, Nov. 2004.

[91] D. S. Kirkpatrick, S. A. Gerber, and S. P. Gygi. The absolute quantification strategy: a general procedure for the quantification of proteins and post-translational modifications. *Methods*, 35(3):265 – 273, 2005.

[92] T. Kloks. *Treewidth: Computations and Approximations (Lecture Notes in Computer Science)*. Springer, 1 edition, Sept. 1994.

[93] N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis, T. Punna, J. M. Peregrín-Alvarez, M. Shales, X. Zhang, M. Davey, M. D. Robinson, A. Paccanaro, J. E. Bray, A. Sheung, B. Beattie, D. P. Richards, V. Canadien, A. Lalev, F. Mena, P. Wong, A. Starostine, M. M. Canete, J. Vlasblom, S. Wu, C. Orsi, S. R. Collins, S. Chandran, R. Haw, J. J. Rilstone, K. Gandi, N. J. Thompson, G. Musso, P. St Onge, S. Ghanny, M. H. Y. Lam, G. Butland, A. M. Altaf-Ul, S. Kanaya, A. Shilatifard, E. O'Shea, J. S. Weissman, C. J. Ingles, T. R. Hughes, J. Parkinson, M. Gerstein, S. J. Wodak, A. Emili, and J. F. Greenblatt. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature*, 440(7084):637–643, Mar. 2006.

[94] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, Dec. 2010.

[95] M. Lavallée-Adam, P. Cloutier, B. Coulombe, and M. Blanchette. Modeling Contaminants in AP-MS/MS Experiments. *Journal of Proteome Research*, 10(2):886–895, 2011.

[96] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 48:787–832, 1999.

[97] E. D. Levy and J. B. Pereira-Leal. Evolution and dynamics of protein interactions and networks. *Current Opinion in Structural Biology*, 18(3):349 – 357, 2008. <ce:title>Nucleic acids / Sequences and topology</ce:title>.

[98] M. Li, J. Wang, J. Chen, Z. Cai, and G. Chen. Identifying the overlapping complexes in protein interaction networks. *International Journal of Data Mining and Bioinformatics*, 4(1):91–108, 2010.

[99] Y. F. Li, R. J. Arnold, H. Tang, and P. Radivojac. The Importance of Peptide Detectability for Protein Identification, Quantification, and Experiment Design in MS/MS Proteomics. *Journal of Proteome Research*, 9(12):6288–6297, 2010.

[100] H. Liu, R. G. Sadygov, and J. R. Yates. A model for random sampling and estimation of relative protein abundance in shotgun proteomics. *Analytical chemistry*, 76(14):4193–4201, July 2004.

[101] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[102] P. Mallick, M. Schirle, S. S. Chen, M. R. Flory, H. Lee, D. Martin, J. Ranish, B. Raught, R. Schmitt, T. Werner, B. Kuster, and R. Aebersold. Computational prediction of proteotypic peptides for quantitative proteomics. *Nature Biotechnology*, 25(1):125–131, Jan. 2007.

[103] S. W. Michnick. Protein fragment complementation strategies for biochemical network mapping. *Current Opinion in Biotechnology*, 14(6):610 – 617, 2003.

[104] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, Jan. 2005.

[105] E. Mujuni and F. Rosamond. Parameterized complexity of the clique partition problem. *In Proc. Fourteenth Computing: The Australasian Theory Symposium (CATS 2008)*, 2008.

[106] M. Narayanan, A. Vetta, E. E. Schadt, and J. Zhu. Simultaneous clustering of multiple gene expression and physical interaction datasets. *PLoS Computational Biology*, 6(4):e1000742, 13, 2010.

[107] A. Nesvizhskii. Protein identification by tandem mass spectrometry and sequence database searching. *Methods Mol Biol.*, 367:87–119, 2007.

[108] A. I. Nesvizhskii and R. Aebersold. Interpretation of shotgun proteomic data: the protein inference problem. *Molecular & cellular proteomics : MCP*, 4(10):1419–1440, Oct. 2005.

[109] W. M. Old, K. Meyer-Arendt, L. Aveline-Wolf, K. G. Pierce, A. Mendoza, J. R. Sevinsky, K. A. Resing, and N. G. Ahn. Comparison of label-free methods for quantifying human proteins by shotgun proteomics. *Molecular & Cellular Proteomics*, 4(10):1487–1502, 2005.

[110] J. Orlin. Contentment in graph theory: covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406–424, 1977.

[111] P. Pei and A. Zhang. A topological measurement for weighted protein interaction network. In *Computational Systems Bioinformatics Conference, 2005. Proceedings. 2005 IEEE*, pages 268–278, 2005.

[112] M. Pellegrini, E. M. Marcotte, M. J. Thompson, D. Eisenberg, and T. O. Yeates. Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 96(8):4285–4288, Apr. 1999.

[113] M. Penrose. *Random Geometric Graphs (Oxford Studies in Probability)*. Oxford University Press, USA, July 2003.

[114] D. N. Perkins, D. J. C. Pappin, D. M. Creasy, and J. S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20(18):3551–3567, 1999.

[115] P. A. Pevzner, V. Dancík, and C. L. Tang. Mutation-tolerant protein identification by mass spectrometry. *Journal of computational biology : a journal of computational molecular cell biology*, 7(6):777–787, 2000.

[116] U. Pieles, W. Zürcher, M. Schär, and H. Moser. Matrix-assisted laser desorption ionization time-of-flight mass spectrometry: a powerful tool for the mass and sequence analysis of natural and modified oligonucleotides. *Nucleic Acids Research*, 21(14):3191–3196, 1993.

[117] N. Przulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[118] O. Puig, F. Caspary, G. Rigaut, B. Rutz, E. Bouveret, E. Bragado-Nilsson, M. Wilm, and B. Séraphin. The tandem affinity purification (TAP) method: a general procedure of protein complex purification. *Methods (San Diego, Calif.)*, 24(3):218–229, July 2001.

[119] G. Rigaut, A. Shevchenko, B. Rutz, M. Wilm, M. Mann, and B. Séraphin. A generic protein purification method for protein complex characterization and proteome exploration. *Nature Biotechnology*, 17(10):1030–1032, 1999.

[120] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of treewidth. *Journal of algorithms*, 7(3):309–322, 1986.

[121] N. Robertson and P. D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory. Series B*, 52(2):153–190, 1991.

[122] R. Saito, H. Suzuki, and Y. Hayashizaki. Interaction generality, a measurement to assess the reliability of a protein-protein interaction. *Nucl. Acids Res.*, 30(5):1163–1168, Mar. 2002.

[123] R. Saito, H. Suzuki, and Y. Hayashizaki. Construction of reliable protein-protein interaction networks with a new interaction generality measure. *Bioinformatics*, 19(6):756–763, Apr. 2003.

[124] M. E. Sardiu, Y. Cai, J. Jin, S. K. Swanson, R. C. Conaway, J. W. Conaway, L. Florens, and M. P. Washburn. Probabilistic assembly of human protein interaction networks from label-free quantitative proteomics. *Proceedings of the National Academy of Sciences of the United States of America*, 105(5):1454–1459, Feb. 2008.

[125] A. Schrijver. *Combinatorial Optimization (3 volume, A,B, & C)*. Springer, 1 edition, Feb. 2003.

[126] R. Sharan and R. Shamir. CLICK: a clustering algorithm with applications to gene expression analysis. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, Aug. 2000.

[127] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of Escherichia coli. *Nat Genet*, 31(1):64–68, 05 2002.

[128] 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, July 2009.

[129] V. Spirin and L. A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences of the United States of America*, 100(21):12123–12128, Oct. 2003.

[130] M. P. H. Stumpf, T. Thorne, E. de Silva, R. Stewart, H. J. An, M. Lappe, and C. Wiuf. Estimating the size of the human interactome. *Proceedings of the National Academy of Sciences*, 105(19):6959–6964, 2008.

[131] K. Suhre. Inference of gene function based on gene fusion events. *Springer Protocols*, 396, November 2007.

[132] H. Tamaki. A linear time heuristic for the branch-decomposition of planar graphs. In G. D. Battista and U. Zwick, editors, *European Symposium on Algorithms (ESA)*, volume 2832 of *Lecture Notes in Computer Science*, pages 765–775. Springer, 2003.

[133] H. Tang, R. J. Arnold, P. Alves, Z. Xun, D. E. Clemmer, M. V. Novotny, J. P. Reilly, and P. Radivojac. A computational approach toward label-free protein quantification using predicted peptide detectability. *Bioinformatics (Oxford, England)*, 22(14):e481–8, 2006.

[134] K. Tarassov, V. Messier, C. R. Landry, S. Radinovic, M. M. S. Molina, I. Shames, Y. Malitskaya, J. Vogel, H. Bussey, and S. W. Michnick. An *in vivo* map of the yeast protein interactome. *Science*, 320(5882):1465–1470, 2008.

[135] J. A. Taylor and R. S. Johnson. Sequence database searches via de novo peptide sequencing by tandem mass spectrometry. *Rapid Communications in Mass Spectrometry*, 11(9):1067–1075, 1997.

[136] R. Uehara. NP-complete problems on a 3-connected cubic planar graph and their application. Technical report, Tokyo Woman's Christian University, Tokyo, Sept. 1996.

[137] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J. M. Rothberg. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature*, 403(6770):623–627, 2000.

[138] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[139] S. M. van Dongen. *Graph clustering by flow simulation*. PhD thesis, University of Utrecht, The Netherlands, 2000.

[140] V. Vazirani. *Approximation Algorithms*. Springer, 2004.

[141] C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417(6887):399–403, 05 2002.

[142] A. J. Walhout, R. Sordella, X. Lu, J. L. Hartley, G. F. Temple, M. A. Brasch, N. Thierry-Mieg, and M. Vidal. Protein interaction mapping in C. elegans using proteins involved in vulval development. *Science (New York, N.Y.)*, 287(5450):116–122, Jan. 2000.

[143] X. Wang, J. Venable, P. LaPointe, D. M. Hutt, A. V. Koulov, J. Coppinger, C. Gurkan, W. Kellner, J. Matteson, H. Plutner, J. R. Riordan, J. W. Kelly, J. R. Yates, and W. E. Balch. Hsp90 cochaperone Aha1 downregulation rescues misfolding of CFTR in cystic fibrosis. *Cell*, 127(4):803–815, Nov. 2006.

[144] C. M. Whitehouse, R. N. Dreyer, M. Yamashita, and J. B. Fenn. Electrospray interface for liquid chromatographs and mass spectrometers. *Analytical Chemistry*, 57(3):675–679, 1985. PMID: 2581476.

[145] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.

[146] I. Xenarios, L. Salwínski, X. J. Duan, P. Higney, S. M. Kim, and D. Eisenberg. DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res*, 30(1):303–305, January 2002.

[147] A. Yamaguchi, K. F. Aoki, and H. Mamitsuka. Graph complexity of chemical compounds in biological pathways. In *Genome Informatics Vol. 14*, pages 376–377, 2003.

[148] P. Ye, B. D. Peyser, X. Pan, J. D. Boeke, F. A. Spencer, and J. S. Bader. Gene function prediction from congruent synthetic lethal interactions in yeast. *Molecular systems biology*, 1:2005.0026, 2005.

[149] H. Yu, P. Braun, M. A. Yildirim, I. Lemmens, K. Venkatesan, J. Sahalie, T. Hirozane-Kishikawa, F. Gebreab, N. Li, N. Simonis, T. Hao, J. Rual, A. Dricot, A. Vazquez, R. R. Murray, C. Simon, L. Tardivo, S. Tam, N. Svrzikapa, C. Fan, A. de Smet, A. Motyl, M. E. Hudson, J. Park, X. Xin, M. E. Cusick, T. Moore, C. Boone, M. Snyder, F. P. Roth, A. Barabasi, J. Tavernier, D. E. Hill, and M. Vidal. High-quality binary protein interaction map of the yeast interactome network. *Science*, 322(5898):104–110, Oct. 2008.

[150] B. Zhang, B. Park, T. Karpinets, and N. Samatova. From pull-down data to protein interaction networks and complexes with biological relevance. *Bioinformatics*, 24(7):979–86, 2008.

[151] B. Zhang, N. C. VerBerkmoes, M. A. Langston, E. Uberbacher, R. L. Hettich, and N. F. Samatova. Detecting differential and correlated protein expression in label-free shotgun proteomics. *Journal of Proteome Research*, 5(11):2909–2918, Nov. 2006.