A practically efficient graph-theoretic approach to protein identification in mass spectrometry

Oliver Serang

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

Doctor of Philosophy

University of Washington

2011

Program Authorized to Offer Degree: Genome Sciences

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Oliver Serang

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

_____

William Stafford Noble

Reading Committee:

_____

William Stafford Noble

_____

Walter L. Ruzzo

_____

Philip Green

Date: _____

Signature_____

Date_____

University of Washington

Abstract

A practically efficient graph-theoretic approach to protein identification in mass spectrometry

Oliver Serang

Chair of the Supervisory Committee:
Professor William Stafford Noble
Department of Genome Sciences

Tandem mass spectrometry has emerged as a powerful tool for the characterization of complex protein samples, an increasingly important problem in biology. The effort to efficiently and accurately perform inference on data from tandem mass spectrometry experiments has resulted in several statistical methods. This thesis rephrases existing methods using a common framework, categorizes them, and discusses them in detail. Each method is analyzed and evaluated by considering the nature of the approach and the outcome and methodology of published comparisons to other methods; the analysis of existing method comparisons is used to comment on the qualities and weaknesses, as well as the overall utility, of these methods. The analysis of existing methods is utilized to propose Fido, a novel Bayesian approach to protein identification; Fido is demonstrated to equal or surpass the state-of-the-art methods. Fido uses a simple generative model of the tandem mass spectrometry process, and employs graph transformations to perform inference efficiently. These graph transformations are then combined with formal graph-theoretic inference procedures to increase the efficiency of inference and facilitate inference on more complex graphs resulting from more sophisticated models of the tandem mass spectrometry process. Extensions of the simple Bayesian model, as well as new directions for the field, are proposed; these proposed changes will help formalize, unify, and improve upon qualitatively similar techniques that are employed by existing methods. A formalized approach improves the quality and reliability with which proteins can be identified in complex mixtures.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my committee and my coauthors Drs. Walter (Larry) Ruzzo, Phil Green, Dave Goodlet, and Michael MacCoss. I am particularly grateful to my advisor Dr. William Noble, who discovered, years before me, that artificial life is second to actual life. I would also like to thank Drs. Thomas Miller and Stephen Walsh for several rewarding challenges in my engineering studies. I am extremely grateful to Drs. Bruce Weir, Jeff Thorne, Hirohisa Kishino, and Spencer Muse, who first introduced me to the beautiful quantitative puzzles in computational biology. I would also like to thank the people with whom I've been close.

DEDICATION

This thesis is dedicated to the dog Laika, who went on a brilliant adventure and never came home.

> *Let all of life be an unfettered howl. Like the crowd greeting the gladiator. Don't stop to think, don't interrupt the scream, exhale, release life's rapture. Everything is blooming. Everything is flying. Everything is screaming, choking on its screams. Laughter. Running. Let-down hair. That is all there is to life.* **–Vladimir Nabokov**

Chapter 1

TANDEM MASS SPECTROMETRY AND ITS APPLICATION TO COMPLEX PROTEIN MIXTURES

1.1 Introduction

Recent advances in DNA sequencing and genomics have made it possible to reconstruct and study the 1918 influenza virus [35], increase the visible color spectrum of mice using human genes [11], and elucidate the process by which some bacteria can reassemble their shattered genomes after exposure to extremely high levels of radiation [37]. But DNA sequencing has its limits; for instance, all cells in the human body are genomically identical despite their vast and observable functional differences. Finding the proteins, which have functional importance, in a group of cells can be much more informative. High-throughput sequencing methods have been applied to RNA [22], an intermediate between DNA and protein, but RNA transcript is often a poor surrogate for protein expression; not all of the transcribed RNA is translated, and once translated, the half-lives of different proteins vary widely [9, 3].

Tandem mass spectrometry has emerged as the most powerful tool for analysis of proteins in complex mixtures [30, 20]. Figure1.1A displays the process by which data is acquired in tandem mass spectrometry-based proteomics. A protein sample is first subjected to enzymatic digestion which breaks the protein into peptides; if a protein is represented as a string of its amino acids, then the peptides produced by digestion will be a set of substrings. Certain digestive enzymes, for example trypsin or chymotrypsin, cut only at specific amino acids, resulting in a small number of possible peptides for each protein. After digestion, the peptide mixture is separated using liquid chromatography (LC), which sequentially elutes peptides according to their hydrophobicity. Subsequently, the population of peptides eluted at a particular retention time are further separated by their precursor mass to charge ratio (m/z) using mass spectrometry; this is accomplished by selecting high intensity peaks from the precursor scan. The population of peptides with a particular retention time and m/z value is then fragmented at certain chemical

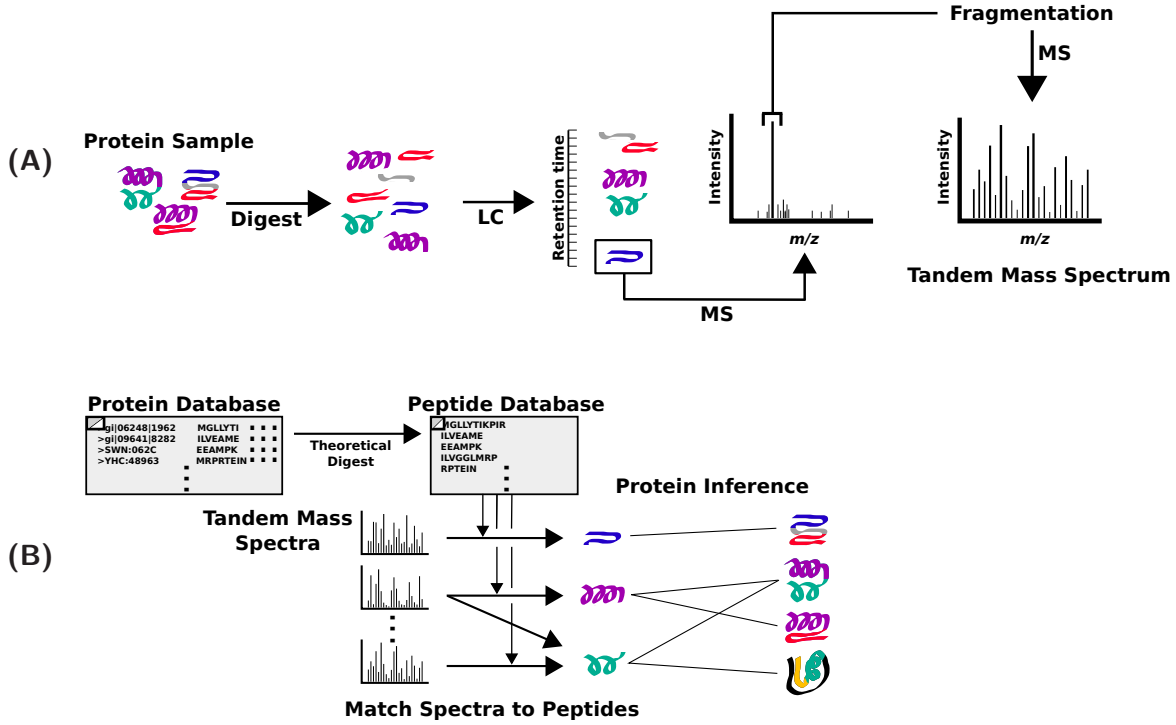


Figure 1.1: **Protein identification using tandem mass spectrometry. (A)** In the tandem mass spectrometry-based proteomics experiment, a collection of proteins is digested into peptides. The peptides are separated by their LC retention time and then by separated by precursor m/z using mass spectrometry (MS). The resulting peptide population is fragmented and subject to a second round of mass spectrometry to produce a tandem mass spectrum. This process is repeated for different LC elution times and precursor m/z values to produce several thousand tandem mass spectra. **(B)** The observed tandem mass spectra are matched to peptides using a search database. In the graphical view of the inference problem, proteins are adjacent to their constituent peptides.

bonds, producing a population of fragments for each peptide. Ideally, the population of peptides fragmented is homogenous. The m/z of each fragment in this population is then measured using mass spectrometry to produce a tandem mass spectrum, a collection of summary statistics about that population of fragments. This process is repeated for different LC retention times and m/z values, resulting in several thousand tandem mass spectra from a single experiment.

The problem of tandem mass spectrometry-based protein identification is to identify the proteins in the original sample from the observed tandem mass spectra (Figure 1.1B). By using existing knowledge from classical genetics, biochemistry and genomics, it is possible to create a database containing an approximate set of all possible proteins of interest for a given sample. The proteins in this database can be associated with the peptides they would be expected to produce when subject to digestion, creating a peptide database; these peptides may consist of only those that result from a perfect digestion, or may include peptides produced via non-standard cleavages. Each observed spectrum is mapped to one or several peptides by comparing it to the predicted theoretical tandem mass spectrum for each peptide in the peptide database [7, 23]. These peptide-spectrum matches (PSMs) are imperfect: error may result from several sources. For instance, matching the spectra to peptides in the peptide database implicitly makes the incorrect assumption that the population of peptides at each hydrophobicity and m/z value is homogeneous. Error can also result from an oversimplified model of how peptides generate observed spectra. Furthermore, it is well-established that some peptides, for instance those with extreme m/z values, are infrequently observed, even when they are present [19, 34].

PSMs are scored by the quality of the match between the observed and theoretical tandem mass spectra [14, 12, 15] and associated with the proteins that would theoretically produce them when digested. These associations between proteins, peptides, and spectra can be represented graphically to form a tripartite graph on proteins, peptides, and spectra by placing edges between proteins and their constituent peptides and between peptides and spectra that match them (Figure 1.1B). All protein identification algorithms start with this tripartite graph and the PSM scores, and then compute either a predicted set of present proteins or a ranking of proteins based on the belief that they are present in the sample.

An ideal protein identification method would maximize sensitivity, the proportion of truly present proteins that are identified, while simultaneously minimizing the false discovery rate

(FDR) [4] of identified proteins. The FDR measures the proportion of identified proteins that are not truly in the sample. The FDR is a useful measure because it quantifies the proportion of protein identifications that would be biologically meaningless in a follow-up experiment. Evaluating a method by estimating the sensitivity and FDR of a method is not trivial, because it is difficult to develop gold standards for complex protein mixtures, which are the most interesting application for tandem mass spectrometry-based protein identification.

1.1.1 Notation

In this thesis, the existing methods for protein identification are described using a common framework. Figure 1.2 presents a graphical view of well-established dependencies in protein inference; these dependencies clearly reflect dependencies inherent in the mass spectrometry process and are used by every protein identification method. The proteins are denoted using a collection of random indicator variables X. Similarly, the peptides are denoted using a collection of random indicator variables Y. The collection of observed spectra and any other associated evidence (for instance, the precursor m/z associated with the spectrum) are encapsulated in a collection D. For simplicity, the objects in the collection D are referred to as "spectra," even though they may be spectra paired with precursor masses or other information. Without loss of generality, a peptide identified at multiple charge states can either be treated as a single peptide or treated as several unique peptides (one for each charge state). Every method discussed can be approached in one of these ways without making specific note of charge state. The index variables i, j, k are used to respectively index the collection of proteins, peptides, and spectra. Table 1.1 provides a quick reference for this notation.

Denote the tripartite graph on all proteins in the protein database, all peptides in the peptide database, and all observed spectra as $G = (E, V)$; let $G' = (E', V')$ denote the "observed graph," a subgraph of G containing only the peptides in G adjacent to observed spectra and only the proteins in G adjacent to those peptides. Note that G depicts a subset of possible dependencies in the general inference problem; a specific model may introduce further dependencies by tying parameters that model the processes depicted in this graph. Conversely, some approaches approximate the known dependencies by removing edges from or altering G. Unless

otherwise stated, all models assume that proteins and peptides not included in the observed graph are absent.

Of particular interest are peptides and spectra like those labeled with an asterisk in Figure 1.2. Peptides in G adjacent to multiple proteins are called "degenerate peptides"; likewise, spectra that match multiple peptides are known as "degenerate spectra." Peptide and spectrum degeneracy are noteworthy because without degenerate peptides and degenerate spectra, G is a tree; inference on that tree can be performed efficiently (unless further dependencies are introduced by tying parameters). Because the proteins and peptides are represented by boolean random variables, the computational cost of a generic naive approach to inference on a general graph may be exponential in both the number of peptides and proteins queried; in practice the number of peptides and proteins of interest is often on the order of ten thousand, making such a naive approach to protein inference practically infeasible.

Let $s_{j,k}$ denote the PSM score for a paired peptide j and spectrum k. $s_{j,k}$ is a rough estimate of $\Pr(Y_j|D_k)$ for $(j,k) \in E$. In practice, PSM scoring algorithms estimate a likelihood proportional to $\Pr(D_k|Y_j = y_j)$ for $(j,k) \in E$. This likelihood is estimated empirically using distributions of features of example PSMs where Y is known. Peptide scoring procedures assume that the spectrum D_k is not degenerate and thus cannot arise from any other peptide; degenerate spectra are eliminated by keeping only the edge to the peptide with the highest likelihood score. By also estimating the prior probability $\Pr(Y_j = y_j)$, these methods then use Bayes rule to estimate:

$$s_{j,k} = \Pr(Y_j|D_k) = \frac{\Pr(D_k|Y_j)\Pr(Y_j)}{\sum_{y_j}\Pr(D_k|Y_j = y_j)\Pr(Y_j = y_j)}$$

In general, $s_{j,k}$ is not a good approximation for $\Pr(Y_j|D)$ for two reasons. First, a peptide may be included in several PSMs with different scores. Second, $s_{j,k}$ does not incorporate protein-level information, which causes covariance in peptides adjacent to the same protein.

1.1.2 Protein grouping

Many of the methods presented below perform some form of protein grouping; that is, they merge multiple proteins together into a single graph node before or during inference. These nodes are then treated as a single protein. Some methods even use the graph connectivity to

remove certain proteins before inference is performed.

Unless otherwise stated, "protein grouping" refers to the process by which proteins are merged together into a single protein group if they are adjacent to identical peptide sets in G. After they are merged, these protein groups are treated as any other protein; therefore, both proteins and protein groups are referred to as "proteins." In addition, some methods remove a protein if it is adjacent to a set of peptides that is a subset of the peptides adjacent to another protein. This principle is denoted as the "Occam's razor principle."

Protein grouping makes evaluation of protein identication methods more difficult because identification of a protein group does not imply that all proteins in the group are present. Instead, a present protein group indicates that at least one protein contained in the group is present. In contrast, the Occam's razor principle does not make interpretation of the identified proteins more difficult, but the principle can make it more difficult to intuitively understand methods that employ it by adding a heuristic behavior to an otherwise probabilistic or numerical method.

1.2 A survey of existing approaches

Several computationally feasible approaches have emerged for performing protein inference from G and the PSM scores. Here these methods are classified as belonging to one of the following categories: set cover methods, iterative methods, and Bayesian methods. Set cover methods approach the problem by accepting a set of peptides based on their associated PSM scores and then accepting a set of proteins based on their adjacency to the accepted PSMs. Iterative methods employ numeric heuristics, which are often probabilistically motivated, to iteratively compute protein scores until convergence is reached. Lastly, Bayesian methods generatively model the mass spectrometry process and then attempt to compute or approximate marginals or a *maximum a posteriori* (MAP) set for the proteins.

1.2.1 Set cover methods

One- and two-peptide rules The one- and two- peptide rules are the simplest methods for protein identification. First, peptides are thresholded so that only peptides paired in a PSM with a score at least τ are present. Second, proteins are similarly thresholded to keep only proteins

Name	Meaning	Indexed by
X	Collection of indicator variables for the proteins	$i, i', \ldots$
Y	Collection of indicator variables for the peptides	$j, j', \ldots$
D	Collection of observed spectra and precursor masses	$k, k', \ldots$
s	Collection of PSM scores	(j, k)
$G = (E, V)$	The graph containing all proteins, peptides, and observed data	–
$G' = (E', V')$	The graph containing all observed data, peptides adjacent to the observed spectra, and proteins adjacent to those peptides	–
x^*	Collection of indicator variables for the proteins identified by a method	$i, i', \ldots$
y^*	Collection of indicator variables for the peptides identified by a method	$j, j', \ldots$

Table 1.1: **Notation reference.** Here each random variable used in is defined. The variables used to index each collection are also presented. Because the notation for indices is consistent, the type of variable accessed in the graph G is known by the name of the variable: $i \in V, i' \in V$ are proteins in G, $j \in V, j' \in V$ are peptides in G, and $k \in V, k' \in V$ are observed spectra. Similarly, $(i, j) \in E$ denotes an edge between protein i and peptide j and $(j, k) \in E$ denotes an edge between peptide j and spectrum k.

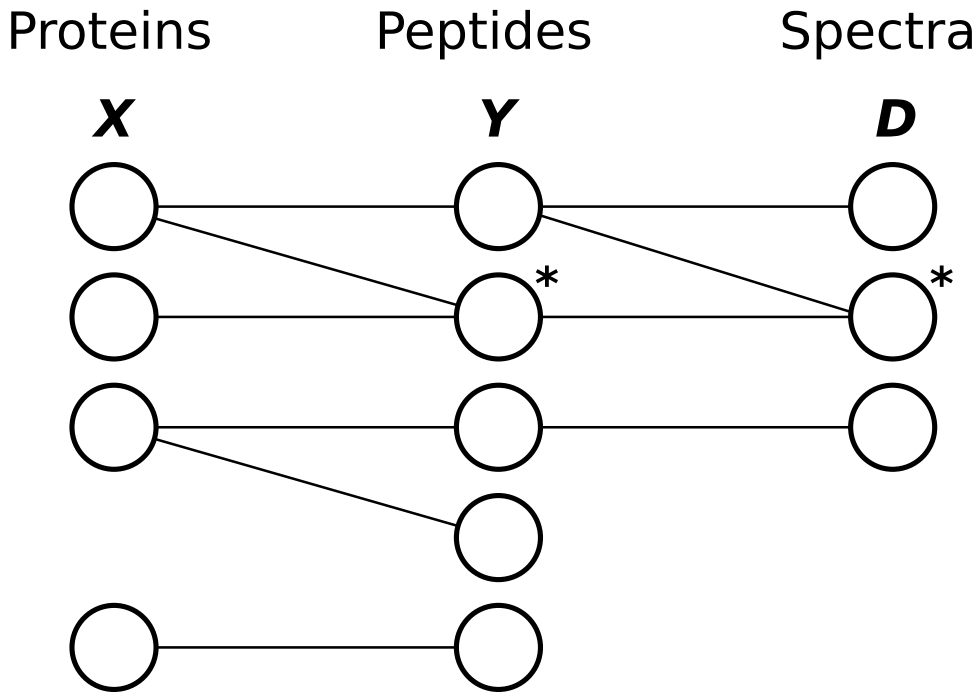


Figure 1.2: **The general graphical view of dependencies in protein inference.** Proteins (X), peptides (Y), and spectra (D) form a tripartite graph with edges indicating well-established dependencies motivated by the mass spectrometry process. Examples of a degenerate peptide and a degenerate spectrum are labeled with asterisks.

adjacent to at least one (or at least two when using the two-peptide rule) present peptides. Formally, the N-peptide rule can be stated as follows:

$$\begin{aligned} y_j^* &= \exists k : (j,k) \in E, s_{j,k} \geq \tau \\ x_i^* &= |\{y_j^* : (i,j) \in E\}| \geq N \end{aligned}$$

Usually, degenerate spectra are eliminated by keeping only the edge pairing a spectrum with its highest-scoring peptide match; although this step is not necessary, it prevents an individual spectrum from matching several peptides. Note that using the one-peptide rule by itself, a single degenerate peptide may force several proteins to be present (for the N-peptide rule, N peptides would be required).

DTASelect DTASelect [32] is a more sophisticated version of the one- and two-peptide rules. First, protein grouping may be performed. Then, DTASelect lets the user select from several criteria to determine which peptides are present, for example, peptides contained in a PSM with a score greater than a threshold or peptides matching at least a certain number of spectra. Similarly, the user manually creates a rule for what proteins are present. Formally, DTASelect allows users to define pairs of scoring functions and thresholds f_Y, τ_Y and f_X, τ_X for peptides and proteins, respectively:

$$\begin{aligned} y_j^* &= f_Y(G,s) \geq \tau_Y \\ x_i^* &= f_X(G,s,y^*) \geq \tau_X \end{aligned}$$

DTASelect can also determine which proteins are similar to a given protein by comparing their sets of adjacent peptides; this similarity measure can be useful to determine situations when a set of peptides can originate from many possible sets of proteins that contain similar sets of adjacent peptides.

IDPicker Frequently, one present protein will produce a handful of degenerate peptides associated with high-scoring PSMs. In such a situation, the one- and two-peptide rules and DTASelect may not only infer that the present protein was in the sample, they may also infer that many other proteins adjacent to these peptides were in the sample. Consequently, degenerate peptides

may lead to a high FDR with these approaches. IDPicker [38] addresses this problem by finding the smallest set of proteins to explain the present peptide set. This smallest protein set defines the "minimum set cover" of the present peptides. Formally, if y is the set of peptides adjacent to the present set of proteins y, IDPicker performs the following optimization:

$$\begin{aligned} y_j^* &= f_Y(G,s) \geq \tau_Y \\ y_j &= \begin{cases} 1, & \exists i : (i,j) \in E, x_i \\ 0, & else \end{cases} \\ x^* &= \underset{x' : y^* \subset y}{\operatorname{argmin}} |x| \end{aligned}$$

Because it defines the present peptides conditional on the present proteins, and then optimizes a constrained set of these proteins, IDPicker models the mass spectrometry process from left to right in the graph in Figure 1.2. In contrast, the one- and two-peptide rules and DTASelect define the peptide and protein relationship conditionally from right to left on the graph in Figure 1.2, despite the fact that the processes underlying mass spectrometry conditional dependency are oriented from left to right.

From a probabilistic perspective, there is an inherently Bayesian flavor to IDPicker. The necessity that all threshold-passing peptides must be explained can be enforced by using a likelihood that is nonzero only when each of those peptides is adjacent to a present protein. Furthermore, large protein sets can be penalized by placing a prior on X that decreases with the cardinality of X (via an arbitrary strictly decreasing positive function h). The MAP estimate of the resulting model will be identical to the minimum set cover found by IDPicker; of the protein sets that result in nonzero likelihood, the smallest will have the highest prior and will therefore be the MAP estimate (at least one nonzero likelihood is guaranteed because each peptide in the

graph must be adjacent to at least one protein, by definition).

$$
\begin{aligned}
y_j^* &= f_Y(G, s) \geq \tau_Y \\
\Pr(D|Y) &= \begin{cases} 1, y^* \subset Y \\ 0, else \end{cases} \\
Y_j|X &= \begin{cases} 1, & \exists i : (i,j) \in E, X_i \\ 0, & else \end{cases} \\
\Pr(X) &\propto h(|X|) \\
D &\perp\!\!\!\perp X|Y
\end{aligned}
$$

IDPicker performs protein grouping; this operation resolves some possible ambiguities where two sets of proteins would both be candidates for the MAP estimate. Note that, in this context, the Occam's razor principle would be redundant, because any protein set containing a protein adjacent in G to a subset of the peptides from another protein could be reduced to explain the same peptides without including the former protein.

1.2.2 Iterative methods

ProteinProphet ProteinProphet [21] is one of the first probabilistically motivated methods for protein identification, and is still one of the most popular and highly regarded. First simplified version of their procedure will be described and then use that to motivate the additional complexity used in ProteinProphet.

ProteinProphet operates on G', the graph of observed data and the adjacent peptides and proteins; proteins not in G' are given posteriors of zero. First, ProteinProphet performs protein grouping using this graph and eliminates proteins by the Occam's razor principle. Then, ProteinProphet eliminates degenerate spectra by removing all but the highest scoring PSM containing each spectrum. Then ProteinProphet computes peptide scores from the remaining connected PSMs; these scores are treated as approximate peptide probabilities:

$$\begin{aligned} s'_j &= \max_k s_{j,k} \\ &= s_{j,k(j)},\ k(j) = \operatorname*{argmax}_k s_{j,k} \\ &\leq 1 - \prod_k (1 - s_{j,k}) \end{aligned}$$

First, consider a graph with no degenerate peptides. In this case, a simplified version of ProteinProphet assumes that each peptide contributes independent evidence to the protein adjacent to that peptide:

$$\Pr(X_i|D) = 1 - \prod_{j:(i,j)\in E'} \left(1 - s'_j\right)$$

When degenerate peptides are encountered, ProteinProphet partitions each peptide score s'_j among its adjacent proteins:

$$\begin{aligned} s''_{i,j} &= w_{i,j} s'_j \\ \sum_{i:(i,j)\in E'} w_{i,j} &= 1 \end{aligned}$$

Then these partitioned peptide scores s'' are used as if there is no degeneracy:

$$\Pr(X_i|D) = 1 - \prod_{j:(i,j)\in E'} \left(1 - s''_{i,j}\right)$$

Given the protein posteriors, the peptide scores are partitioned so that the size of the partition associated with each protein is proportional to that protein's posterior:

$$w_{i,j} \propto \Pr(X_i|D)$$

Because the sum of weights for each peptide must sum to unity, the proportion constant can be removed thus:

$$w_{i,j} = \frac{\Pr(X_i|D)}{\sum_{i':(i',j)\in E'} \Pr(X_{i'}|D)}$$

$\forall(i,j) \in E', w_{i,j} \leftarrow 1$

while convergence is not reached **do**

$\forall j, s''_{i,j} \leftarrow w_{i,j} s'_j$

$\forall i, \Pr(X_i|D) \leftarrow 1 - \prod_{j:(i,j)\in E'} \left(1 - s''_{i,j}\right)$

$\forall(i,j) \in E', w_{i,j} \leftarrow \Pr(X_i|D) / \sum_{i':(i',j)\in E'} \Pr(X'_i|D)$

end while

Finally, the protein posterior estimates $\forall i, Pr(X_i|D)$ and the partition weights $\forall(i,j) \in E', w_{i,j}$ are iteratively updated in a batch-wise manner, until convergence is reached:

As described, this scheme treats each partitioned peptide score as exact independent evidence to an adjacent protein. For this reason, a single high-scoring non-degenerate peptide j may single-handedly result in a prediction that the adjacent protein i is present. This protein would be given the same posterior as a protein with several high-scoring peptides:

$$\begin{aligned} s'_j &\approx 1 \\ \{i' : (i',j) \in E'\} &= \{i\} \\ \Pr(X_i|D) &= 1 - \prod_{j':(i,j)\in E'} \left(1 - s''_{i,j'}\right) \\ &= 1 - \left(1 - s'_j\right) \\ &\approx 1 \end{aligned}$$

The creators of ProteinProphet make a useful observation: among high-scoring peptides, there are fewer incorrect peptide identifications for peptides associated with proteins that have evidence from several other peptides. ProteinProphet therefore computes a score called "NSP" ("number of sibling peptides") that summarizes other peptide evidence for a particular protein i adjacent to peptide j:

$$NSP_j = \sum_{j'\neq j:\exists i,(i,j')\in E'} s'_j$$

These NSP values are binned to estimate a new peptide score conditional on the NSP bin:

$$\Pr(Y_j|D_{k(j)}, bin(NSP_j) = a) \approx s_j''' = \frac{\Pr(Y_j|D_{k(j)}) \Pr(bin(NSP_j) = a|Y_j)}{\sum_{y_j} \Pr(Y_j = y_j|D_{k(j)} \Pr(bin(NSP_j) = a|Y_j = y_j)}$$

where ProteinProphet approximates the true peptide posteriors with the peptide score: $\Pr(Y_j|D_{k(j)}) \approx s'_j$. The NSP probabilities are estimated by taking the ratio of expected numbers of present peptides in each NSP bin.

$$\Pr(bin(NSP_j) = a|Y_j) \approx \frac{\sum_{j:bin(NSP_j)=a} s_j'''}{\sum_j s_j'''}$$

Finally, the use of NSP is extended to treat each degenerate peptide as several partitionied peptides, one for each adjacent protein. The peptide scores conditioning on this new NSP score NSP' for each protein and peptide will be denoted $s_{i,j}^{(IV)} = \Pr(Y_j|D_{k(j)}, NSP_j^{i'})$

$$NSP_j^{i'} = \sum_{j' \neq j:(i,j') \in E'} s_{i,j}''$$

$$\Pr(bin(NSP_j^{i'}) = a|Y_j) \approx \frac{\sum_i \sum_{j:bin(NSP_j^{i'})=a} w_{i,j} s_{i,j}^{(IV)}}{\sum_i \sum_j w_{i,j} s_{i,j}^{(IV)}}$$

The final algorithm is as follows:

When posterior protein estimates $\Pr(X_i|D)$ are given by any method, some threshold τ_X is used to choose the final set of accepted proteins:

$$x^* = \{i : \Pr(X_i|D) > \tau_X\}$$

$\forall(i,j) \in E', w_{i,j} \leftarrow 1$

while convergence is not reached **do**

$\forall(i,j) \in E'$, compute $s_{i,j}^{(IV)}$

$\forall i, \Pr(X_i|D) \leftarrow 1 - \prod_{j:(i,j)\in E'} \left(1 - w_{i,j} s_{i,j}^{(IV)}\right)$

$\forall(i,j) \in E', w_{i,j} \leftarrow \Pr(X_i|D) / \sum_{i':(i',j)\in E'} \Pr(X_i'|D)$

end while

Scaffold Scaffold [27] employs a novel approach to the problem of spectral degeneracy to perform inference on the observed graph G'. When a spectrum matches multiple peptides, then only the peptides with scores approximately equal to the top-scoring peptide are kept. These remaining peptides are grouped together for that spectrum, creating a "peptide group" with a score equal to the scores of the approximately equal scores of the PSMs from that spectrum and the peptides it contains.

Scaffold resolves peptide degeneracy using a greedy method. Proteins are assigned peptide groups that are not adjacent to any other proteins. The protein scores are equal to the sum of the scores of the assigned peptide groups. Then, degenerate peptide groups (i.e. peptide groups containing peptides ajdacent to multiple proteins) are assigned to the protein with the highest protein score. This process is repeated until no more peptide groups can be assigned, and these ranks are used to represent the belief that a protein is present. The resulting graph with edges connecting proteins to their assigned peptide groups is processed using ProteinProphet, but with no weighting of peptide groups.

In Scaffold, proteins are first grouped using standard protein grouping, but proteins may also be grouped if the sum of the scores of the peptides they do not share is lower than 0.95. The Occam's razor principle is used to discard proteins that don't have unique peptide evidence. In an identical manner to ProteinProphet, the present set of proteins x^* is found by applying some threshold to the sorted list of protein posteriors.

EBP The EBP method [24] is another probabilistically phrased and motivated method, but is ultimately a complex numerical heuristic similar to ProteinProphet. Like ProteinProphet, EBP operates on the observed graph G'. Initially, the EBP method removes spectra degeneracy

using the same method as ProteinProphet, and computing $\Pr(Y_j|D_{k(j)}) \approx s'_j = \max_k s_{j,k}$. EBP partitions the problem of protein identification into two parts. First, the set H consists of proteins that are either present or homologous to a present protein. Second, the set $X \subset H$ consists of proteins that are present in the sample.

Proteins in H must be adjacent to at least one present peptide. Membership in H is calculated using a Poisson distribution to estimate the probability complement to the event that peptide adjacent to the protein is truly present. Membership in X is calculated conditional on membership in H using a weighting scheme similar to ProteinProphet; each weight, where $\sum_i w_{i,j} = 1$, represents the probability that a present peptide j originated from protein i.

In a manner very similar to ProteinProphet's NSP score, EBP computes an approximate abundance estimate v_i for each protein. The abundance estimates are computed as the total sum of weighted peptide scores associated with a protein:

$$v_i = \sum_{j:(i,j)\in E'} w_{i,j} s'_j$$

The abundance estimates are thresholded into abundance class bins $a_i = bin(v_i)$. For each abundance class bin, there is a corresponding set of parameters $\theta_a = (N_a, \tau_a, n_a, \gamma_a, \kappa_a, \lambda_a)$ with the following meanings:

Variable	Meaning
N_a	Number of proteins in bin a
τ_a	Total length of proteins in bin a
n_a	Number of peptide matches to proteins in bin a
γ_a	Proportion of proteins in bin a that are in H
κ_a	Total length of proteins in bin a and in H
λ_a	Number of peptide matches in bin a that are correct

These parameters, along with w, are used to sequentially compute estimates of the following:

$$
\begin{aligned}
&a|w, \forall_j s'_j \\
&\Pr(H_i|D, \theta, a) \\
&\Pr(X_i|H_i, D, \theta, a, w) \\
&\theta|\forall i \Pr(H_i|D, \theta, a) \\
&w|\forall i, \Pr(X_i|D)
\end{aligned}
$$

The value a is updated as stated above by first computing $v_i|w$ and then thresholding it into the appropriate bin. The probability $\Pr(H_i|D, theta, a)$ is estimated by modeling the number of correct peptide identifications matching a protein $i \in H$ as a Poisson process. The parameters of this Poisson process are defined by the parameters in the appropriate bin θ_{a_i}, and by a heuristic value $e^{\sqrt{log(|\{j:(i,j)\in E\}|)}}$, an estimate of the proportional probability that the highest-scoring random match is to one of the peptides adjacent to i. A value proportional to the conditional probability that $i \notin H$ is estimated using the estimated prior probability that $i \notin H$, the product of probabilities that all peptides observed are incorrect identifications and the probability that the observed number of peptides would be produced by the Poisson distribution:

$$
\Pr(\neg H_i|D, \theta, a) \propto \left(1 - \gamma_{a_i}\right) \times \prod_{j:(i,j)\in E'} (1 - s'_j) \times
$$

$$
\Pr\left(|\{Y_j : (i,j) \in E\}| = |\{j : (i,j) \in E'\}| \; ||\{Y_j : (i,j) \in E\}| \sim Poisson\left(\frac{e^{\sqrt{log(|\{j:(i,j)\in E\}|)}}\lambda_{a_i}}{\kappa_{a_i}}\right)\right)
$$

Similarly, a value proportional to the probability that $i \in H$ can be estimated by summing over outcomes with a nonzero quantity m of correct peptides identifications. The resulting summation terms will consist of two Poisson probabilities (one for the correct peptide identifications, and the other for the inforrect peptide identifications) multiplied by the probability that exactly that many peptides are correct matches. The latter probability is estimated by summing over all possible subsets of exactly m present peptides adjacent in G' to i:

$$\Pr(|\{j \ : \ (i,j) \ \in \ E'\}| \ = \ m|D) \ = \sum_{|\{y_j:(i,j)\in E'\}|=m} \prod_{j:(i,j)\in E'} s'_j y_j + (1 - s'_j)(1 - y_j)$$

The probability that a protein is truly present in the sample given that it is adjacent to a truly present peptide is computed by marginalizing out the variable H_i; this marginalization only takes one step because the set $X \subset H$:

$$\begin{aligned}\Pr(X_i|D,\theta,a) &= \sum_{h_i} \Pr(X_i|H_i = h_i, D, \theta, a) \Pr(H_i = h_i|D,\theta,a) \\ &= \Pr(X_i|H_i, D, \theta, a) \Pr(H_i|D,\theta,a)\end{aligned}$$

When computing $\Pr(X_i|H_i, D, \theta, a)$, the method used is nearly identical to the method used to compute $\Pr(X_i|H_i, D, \theta, a)$; the difference is that in computing $\Pr(X_i|H_i, D, \theta, a)$ the weighted value s'' is used in place of all instances of s'.

The entire iterative estimation process is repeated until the estimated values appear to converge. The EBP method is not demonstrated to be a true EM method, despite its probabilistic motivation and description; iterative estimation of posteriors, weights, and other parameters is not equivalent to iteratively maximizing the expectation of the full protein configuration likelihood.

The authors don't state an explicit procedure used for updating w; instead, they indicate that the greatest weight must be given to the protein with the highest current posterior estimate. Presumably, the authors use the same procedure as ProteinProphet for updating w. The present set of proteins x^* is found by applying some threshold to the sorted list of protein posteriors. The authors suggest that a combinatorial function would allow extension to replicate experiments by requiring that the protein be present in a certain number of those experiments.

PANORAMICS The PANORAMICS method [8] similarly uses the observed graph G'. First, peptide scores are normalized using parameters that must be established from a known data set.

Then, proteins are grouped and peptides that produce indistinguishable theoretical spectra are merged. After that, spectral degeneracy is removed by keeping only the edge to the highest-scoring peptide. Peptide probabilities are estimated by estimating the probability that a peptide is absent; the chances a peptide is absent will be computed as the probability that all PSMs containing that peptide are absent by taking the product over the complements of their scores:

$$s'_j = 1 - \prod_k 1 - s_{j,k}$$

If $X_i^{j'}$ denotes the event that peptide j is present as a result of the present protein i, then the probability that protein i is present and the probability that peptide j is present is given using a formula similar to the unweighted formulation of ProteinProphet:

$$\Pr(X_i|D) = 1 - \prod_{j:(i,j)\in E'} 1 - \Pr(X_i^{j'}|D)$$

$$s'_j \approx \Pr(Y_j|D) = 1 - \prod_{i:(i,j)\in E'} 1 - \Pr(X_i^{j'}|D)$$

Finally, by assuming that the probability of observing any peptide given that an adjacent protein is present depends only on the peptide, the probability of the event $X_i^{j'}$ can be rewritten:

$$\Pr(X_i^{j'}|D) = \Pr(X_i|D)\Pr(Y_j|\exists i : (i,j) \in E')$$

By rephrasing $X_i^{j'}$ in this way, it is possible to redefine the posterior protein and peptide probabilities:

$$\Pr(X_i|D) = 1 - \prod_{j:(i,j)\in E'} 1 - \Pr(X_i|D)\Pr(Y_j|\exists i : (i,j) \in E')$$

$$s'_j = 1 - \prod_{i:(i,j)\in E'} 1 - \Pr(X_i|D)\Pr(Y_j|\exists i : (i,j) \in E')$$

Values of $\Pr(X_i|D)$ and $\Pr(Y_j|\exists i : (i,j) \in E')$ that are consistent with these equations are found using the Newton-Raphson method. A present set of proteins x^* is found by applying some threshold to the sorted list of protein posteriors.

1.2.3 Bayesian methods

Hierarchical Statistical Model The hierarchical statistical model [29] assumes that $D \perp\!\!\!\perp X|Y$ and generatively models the process by which proteins create spectra to perform inference on G'. Spectral degeneracy is eliminated by keeping only the edge incident to the peptide with the highest PSM score for any spectrum.

First, the model assumes an independent and identically distributed (iid) prior for all proteins: $\Pr(X_i) = \gamma$. Next, the authors model the process by which a known set of proteins creates a set of peptides as independent processes from each adjacent protein in G'. Their model uses different parameters to model the emission of peptides resulting from different cleavages of that protein; a specific cleavage indicates that the enzyme cut where expected, while a nonspecific cleavage indicates that the enzyme would have to cut at an unexpected location.

$$\Pr(Y_j|X) = 1 - \prod_{i:(i,j)\in E'} 1 - \alpha_{i,j}$$

where

$$\alpha_{i,j} = \begin{cases} \alpha'_N & \text{one nonspecific cleavage} \\ \alpha'_S & \text{one specific cleavage} \\ \alpha'_{NN} & \text{two nonspecific cleavage} \\ \alpha'_{NS} & \text{one specific and one nonspecific cleavage} \\ \alpha'_{SS} & \text{two specific cleavage} \end{cases}$$

The probability of a correct PSM match given the associated peptides is likewise calculated using Z as a random variable that indicates whether a PSM match is correct:

$$\Pr(Z_{j,k(j)}|Y_j = y_j) = \begin{cases} \delta & y_j = 1 \\ 0 & else \end{cases}$$

The authors also model the distribution of PSM scores as a mixture of the PSM score distributions from correct and incorrect PSM matches with mixing proportion λ, where q are factors that influence the score:

$$s \sim Mixture(\{f_{correct}(q), f_{incorrect}(q)\}, \lambda)$$

These likelihoods must be defined in order to perform protein inference. In practice, they are defined as parameterized distributions, and the parameter estimates are made using a separate data set.

Lastly, the probability that the number of peptide hits for present proteins is greater than some threshold m is modeled using parameters ρ_1 and ρ_0:

$$\Pr(|\{i : (i,j) \in E'\}| > m \mid X_i = x_i) = \rho_1^{x_i} \rho_0^{\neg x_i}$$

Then, approximate maximum likelihood estimates (MLEs) of the parameters $\theta = (\gamma, \alpha, \delta, \lambda, \rho_1, \rho_0)$ are computed using EM with hidden variables X, Y, and Z. Finally, using the estimated θ values, the posterior probabilities are estimated for proteins:

$$\Pr(X_i|D) \approx \Pr(X_i|\theta, s, G)$$

These posteriors are thresholded to produce a present set of proteins x^*.

Nested Mixture Model The nested mixture model approach [17] to protein identification transforms G' by copying degenerate peptides so that each adjacent protein is adjacent to its own unique copy of the peptide and the spectra adjacent to the peptide. Spectral degeneracy is also removed by keeping only the highest-scoring edges for each spectrum. This transformation ensures the graph G' is a tree. Because each peptide can only be associated with a single protein, let $i(j)$ denote the protein associated with peptide j.

The model assumes an iid prior for all proteins: $\Pr(X_i) = \gamma$. All peptides adjacent to absent proteins are assumed to be absent, and the peptides adjacent to present proteins are drawn from a mixture model of present and absent peptides:

$$\Pr(Y_j|X_{i(j)} = x_{i(j)}) = \begin{cases} \alpha & x_{i(j)} = 1 \\ 0 & else \end{cases}$$

The number of peptides adjacent to present and absent proteins are of known distributions f_1 and f_0, respectively:

$$|\{j : (i,j) \in E'\}| \mid X_i = x_i \sim f(x_i|\theta_f)$$

$$f(x_i) = \begin{cases} f_1(\theta_f) & x_i = 1 \\ f_0(\theta_f) & else \end{cases}$$

In a similar manner, the distributions of PSM scores containing present and absent peptides are also modeled using distributions g_1 and g_0, respectively:

$$S_{j(k),k} \; | Y_{j(k)} = y_{j(k)} \sim g(y_{j(k)} | \theta_g)$$

$$g(y_{j(k)}) = \begin{cases} g_1(\theta_g) & y_{j(k)} = 1 \\ g_0(\theta_g) & else \end{cases}$$

The distributions f and g are parameterized by θ_f and θ_g, respectively. Because the transformed graph G' is a tree, the peptides are conditionally independent of one another given the associated protein. In a similar manner, the scores are conditionally independent of one another given the associated peptide. The likelihood can then be computed:

$$\begin{aligned} \Pr(D|X = x, \theta) = & \\ & \prod_i \Big[Pr(m = |\{j : (i,j) \in E'\}| \; | m \sim f(x_i, \theta_f)) \\ & \quad \sum_y \prod_{j:i=i(j)} \Big[\Pr(Y_j = y_j | X_i = x_i) \\ & \quad\quad \prod_{k:j=j(k)} \Pr(S_{j,k} = s_{j,k} | S_{j,k} \sim g(y_j | \theta_g)) \Big] \Big] \\ = & \prod_i \Big[Pr(m = |\{j : (i,j) \in E'\}| \; | m \sim f(x_i, \theta_f)) \\ & \quad \prod_{j:i=i(j)} \Big[\sum_{y_j} \Pr(Y_j = y_j | X_i = x_i) \\ & \quad\quad \prod_{k:j=j(k)} \Pr(S_{j,k} = s_{j,k} | S_{j,k} \sim g(y_j | \theta_g)) \Big] \Big] \end{aligned}$$

Similarly, the likelihood constant can be computed and normalized out by summing over all protein states in the joint probability:

$$\sum_x \Pr(D|X=x,\theta)\Pr(X=x) =$$
$$\prod_i \Big[\sum_{x_i} \Pr(X_i = x_i)$$
$$Pr(m = |\{j : (i,j) \in E'\}| \; | m \sim f(x_i, \theta_f))$$
$$\prod_{j:i=i(j)} \Big[\sum_{y_j} \Pr(Y_j = y_j | X_i = x_i)$$
$$\prod_{k:j=j(k)} \Pr(S_{j,k} = s_{j,k} | S_{j,k} \sim g(y_j|\theta_g))\Big]\Big]\Big]$$

The EM algorithm is used to compute approximate MLE estimates of the parameters $\theta = (\gamma, \alpha, \theta_f, \theta_g)$. Posteriors for each protein are computed and these posteriors are thresholded to compute the set of present proteins x^*.

MSBayes MSBayes [18] is takes a novel approach to protein inference; a complex, static model of peptide detectability is used to model the mass spectrometry process for the entire graph G, rather than for the observed graph. The best peptide match for each spectrum determines the peptide score s_j; peptides that are not in the observed graph are given scores of zero.

In MSBayes, each protein has an independent prior $\Pr(X_i|D) = \gamma$; the value of γ is either 0.5 (a uniform prior), or chosen using prior information about the number of proteins in the data set. The peptide scores are assumed to comprise the data, which are conditionally independent of the proteins given the peptides. Each peptide is assumed conditionally independent of other peptides given the proteins. Likewise, scores are assumed to be conditionally independent of

each other given the peptide set:

$$\begin{aligned}
\Pr(D|X=x) &= \Pr(S=s|X=x) \\
&= \sum_y \Pr(S=s|Y=y)\Pr(Y=y|X=x) \\
&= \sum_y \prod_j \Pr(S_{k(j)}=s_{k(j)}|Y_j=y_j)\Pr(Y_j=y_j|X=x) \\
&= \sum_y \prod_j \Pr(S_{k(j)}=s_{k(j)}|Y_j=y_j)\Pr(Y_j=y_j|X=x) \\
\Pr(Y_j=y_j|X=x) &= 1-\prod_{i:(i,j)\in E} 1-x_i\alpha_{i,j}
\end{aligned}$$

$$\Pr(S_{k(j)}=s_{k(j)}|Y_j=y_j) = \frac{\Pr(Y_j=y_j|S_{k(j)}=s_{k(j)})\Pr(S_{k(j)}=s_{k(j)})}{\Pr(Y_j=y_j)}$$

where $\Pr(Y_j=y_j|S_{k(j)}=s_{k(j)})$ is estimated by PeptideProphet [14]. The peptide emission probabilities $\alpha_{i,j}$ are estimated using a static predictor of peptide detectability. This detectability predictor [34] is composed of a neural network that uses 175 features of each protein-peptide pair to predict whether a peptide will be observed given that an adjacent protein is present. The parameters of this model are trained using a "protein standard," a small data set composed of a known set of proteins that have been biochemically purified.

Lastly, Markov chain Monte Carlo (MCMC) is used to jointly sample from the space of proteins and peptides and compute protein posteriors and a MAP protein set as follows:

The posterior of each protein can be estimated by computing the frequency of iterations for which that protein is present in the configuration x. These posteriors are thresholded to estimate the present set of proteins x^*. Alternatively, the MAP protein and peptide set can be computed by storing the joint configuration that results in the highest proportional posterior. This MAP protein set can be treated as the set of present proteins:

$$x^* = x_{MAP}$$

$x, y \leftarrow$ some configuration : $\Pr(D, X = x, Y = y) > 0$

while convergence is not reached **do**

$b_x \leftarrow random(\{i_1, i_2, \ldots\} : |b_x| = u$

$b_y \leftarrow random(\{j_1, j_2, \ldots\} : |b_y| = v$

Denote $X' = \{X_i : i \in b_x\}$

Denote $Y' = \{Y_j : j \in b_y\}$

$\forall x' \forall y', p_{x',y'} \leftarrow$

$\propto \Pr(X' = x', Y' = y' | \forall i \notin b_x X_i = x_i, \forall j \notin b_y Y_j = y_j, D)$

Sample an x', y' proportional to $p_{x',y'}$

$\forall i \in b_x \quad x_i \leftarrow x'_i$

$\forall j \in b_y \quad y_j \leftarrow y'_j$

end while

1.3 Comparison of existing approaches

In the publications originally presenting these methods, some are compared against other existing approaches. This section evaluates each comparison and evaluates these methods relative to each other. Table 1.2 depicts the analysis of each published comparison between a pair of methods. For each method, both the accuracy demonstrated in the original publication, as well as the computational cost required to apply it to large, biologically interesting data sets are used for evaluation.

The validity of the published evaluation of each pair of methods is also considered. Traditionally, decoy database methods have been used to evaluate mass spectrometry-based protein identifications [6]. This appraoch introduces proteins into the protein database that are known to be absent; these absent proteins, known as "decoys," may come from a species known not to contribute to the sample, or may be generated by shuffling or reversing the original protein database. The proteins comprising the original protein database (before decoys are introduced) are called "target" proteins. The decoy proteins can be used to estimate the FDR of a given set of protein identifications; if decoy proteins are favored no more and no less than absent targets, and if the number of decoys and targets are equal, then for each decoy protein found

in a predicted present protein set x^*, it is expected that one incorrect target protein in x^* is also present. This target-decoy approach makes assumptions that are known to be incorrect regarding the target and decoy databases, which result in several caveats to using it to estimate the FDR.

Below is a detailed description of the caveats to the published comparisons between pairs of methods, as well as the outcome of the competition to produce an analysis of each method's accuracy. Overall, this analysis of a method's accuracy, along with the method's efficiency and compuational scalability, is used as criteria to predict and evaluate its utility for identifying proteins in large data sets from complex protein mixtures.

In [38], IDPicker is compared to the one- and two-peptide rules using experiments from three data sets: a protein standard containing 49 proteins, proteins from yeast cells, and proteins from human. For each data set, the authors choose a peptide threshold τ_Y by controlling the peptide FDR estimated using a decoy database. The IDPicker method substantially increases the specificity of the method, while slightly lowering the sensitivity. It is trivial to observe that IDPicker will always select a subset of the proteins chosen by the one- or two-peptide rule when using the same peptide threshold for both methods; therefore, IDPicker can never have superior sensitivity, and so it is appropriate to use a more lax peptide threshold for IDPicker in order to compare it the one- and two-peptide rules. Even so, it is practically and theoretically clear that IDPicker will result in substantially greater specificity, especially in instances where there are many degenerate peptides. Each degenerate peptide adjacent to a present protein has a large chance of receiving a high score; using the one- and two- peptide rules, all proteins adjacent to that degenerate peptide will be included in x^*, even if only one of them was present.

Furthermore, degenerate target peptides are much more likely to be adjacent to target proteins; the expected scores of target peptides, which consist of a mixture of present and absent proteins, will be higher than the expected score of decoy peptides, which are necessarily absent. Target protein identifications are often used as a surrogate for true positive protein identifications when there is no ground truth regarding the set of present proteins. Therefore, methods like the one- and two-peptide rules are more likely to include absent target proteins rather than decoy proteins, resulting in an overestimated sensitivity and underestimated FDR.

Understandably, the IDPicker method is less efficient than the computationally trivial one-

		One- and two-peptide	IDPicker	ProteinProphet	EBP	PANORAMICS	Hierarchical	Nested Mixture	MSBayes
One- and two-peptide	Accuracy		■□□□						
	Scalability		■■■□						
IDPicker	Accuracy	■■■□							
	Scalability	■□□□							
ProteinProphet	Accuracy				■■□□	■■■□	■■□□	■■■□	■■□□
	Scalability				■■■□	■□□□	■■■■	■■□□	■■□□
EBP	Accuracy			■■□□					
	Scalability			■□□□					
PANORAMICS	Accuracy			■□□□					
	Scalability			■■■□					
Hierarchical	Accuracy			■■□□				–	
	Scalability			□□□□				□□□□	
Nested Mixture	Accuracy			■□□□			–		
	Scalability			■■□□			■■■■		
MSBayes	Accuracy			■■□□					
	Scalability			■■□□					

Table 1.2: **Published method comparisons.** Published comparisons between methods are analyzed and evaluated. The entries in a row depict the evaluations of the labeled method relative to other methods from each column. Each cell categorizes the outcome of a comparison between two methods. The various symbols indicate whether the row method performed much worse (□□□□), slightly worse (■□□□), essentially the same (■■□□), slightly better (■■■□) or significantly better (■■■■) than the column method. For each pair of methods, the accuracy and the scalability are compared. Accuracy evaluates the ability of a method to identify many present proteins at a low FDR when applied to an unseen data set. Scalability evaluates the computational efficiency of a method and whether it can be applied to large, biologically interesting data sets.

and two-peptide rules. Solving minimum set cover is NP-complete [13]; however, like many NP-complete problems, it can often be solved efficiently in practice, and when an exact solution is inefficient, it can be approximated using established approaches.

ProteinProphet is one of the most popular and seminal protein identification methods, and so many methods have been compared to it. In [24], the EBP method is compared to ProteinProphet using several replicate experiments on a protein standard consisting of 18 purified proteins [16]. Using two protein thresholds $\tau_X \in \{0.9, 0.7\}$, EBP achieves a greater specificity (one fewer decoy protein identified), but a lower sensitivity (one and two fewer present proteins identified, respectively). Furthermore, EBP is shown to be conservative; the FDR estimated computing the expected value of the complement of the posterior probabilities of proteins included in x^* [5] is substantially higher than the true FDR. For this reason, a fairly high protein threshold would be required to achieve greater sensitivity. This higher protein threshold is likely to increase the FDR. For this reason, it is concluded that the method has not been demonstrated to be superior to ProteinProphet. EBP may even lower the interpretability of the protein posteriors due to its conservative estimates.

The iterative procedures underpinning EBP are very similar to ProteinProphet and are mostly very efficient; however, when estimating the probabilities $\Pr(H_i|D, \theta_{a_i})$ and $\Pr(T_i|D, \theta_{a_i}, w)$, there is a sum over all subsets of observed peptides adjacent to a protein. This term requires summing over the power set, which grows exponentially with the number of observed peptides adjacent to any protein. Unless this sum of products can be transformed into a product of sums using dynamic programming or some other procedure, it will become prohibitively inefficient to perform on data sets from complex organisms like human, where the number of peptides adjacent to a protein can be very large.

In [8], ProteinProphet is compared with PANORAMICS on a data set from the plant *Arabidopsis thaliana*. The authors calibrated parameters for their peptide score in a rigorous manner using a protein standard; the protein standard is different enough from the *A. thaliana* data set that these parameter estimates are unlikely to provide an unfair advantage to their method. The authors then search the data against two different databases. The first database is the *A. thaliana* proteome (targets) combined with reversed copies of every target protein. The second database is the NCBI NR database, which contains over 3.1 million proteins. For both searches,

the protein threshold τ_X was varied to produce a receiver operating characteristic (ROC) curve, which plots the number of true positive protein identifications against the number of false positive protein identifications. When the spectra were searched against the *A. thaliana* data set, ProteinProphet identifies more target proteins from the *A. thaliana* proteome in the low FDR region. In general, ProteinProphet performs very well at low FDR thresholds, and it is common to see comparisons in which competing methods outperform ProteinProphet only once the FDR becomes higher.

When these spectra are searched against the combined NCBI NR (target) and NCBI NR reversed (decoy) database, PANORAMICS identifies many more targets than ProteinProphet in the moderate FDR region. The authors suggest that this increased number of identified targets indicates an increased sensitivity compared to ProteinProphet; however, the paper does not indicate whether these NCBI NR proteins are actually from *A. thaliana*. The demonstrated tendency to distinguish targets from decoys, regardless of species, is actually quite detrimental; in practice, the organisms producing the sample data are almost always known, and the challenge is to separate and distinguish the present target proteins from the absent target proteins. A preference for target proteins, regardless of species, can be the result of degenerate peptides from a single present protein that allow several other adjacent proteins to be included in the set x^*. For this reason, PANORAMICS appears to be slightly less reliable than ProteinProphet, which includes fewer and fewer target proteins as the protein threshold τ_X is lowered.

The principal strength of the PANORAMICS method is its elegant simplicity, which casts the protein identification problem as a numeric equation; approaches to numerically solve such equations are extremely efficient. Their numeric solution is not demonstrated to be unique, but it is an appealing heuristic.

In [29], ProteinProphet is again compared, this time to the hierarchical model. The two methods are both used to analyze a data set consisting of 23 peptides together with 12 purified proteins. The proteins are enzymatically digested, and the resulting peptide mixture is treated as containing 35 present "proteins"; the 23 peptides are treated as single-peptide proteins. The authors use the two methods to identify peptides at different peptide thresholds, and they demonstrate that ProteinProphet has a slightly greater sensitivity at a low FDR and a lower sensitivity at a higher FDR. They do not perform a similar comparison for proteins; instead, they

choose a single cuttoff at $\tau_X = 0.8$ and show that ProteinProphet and the hierarchical model identify the same number of present proteins and decoy proteins. The *ad hoc* choice of protein threshold is uninformative, especially given the tendency of ProteinProphet to outperform other methods in the low FDR region.

Unfortunately, the hierarchical model computes a sum over all peptide and protein configurations, and the paper does not discuss the resulting computational cost. Depending on the implementation, the cost of computing posteriors will grow with either the exponential or the factorial of the number of variables (or the number of proteins, if the sum of products over peptides is transformed into a product of sums) in a connected subgraph; in practice, this poor scalability makes the hierarchical model computationally prohibitive for so many data sets that the method is not practically useful.

In [17] the nested mixture model is compared to ProteinProphet on a yeast data set using an unspecified decoy database of artificial proteins. The nested mixture model has slightly higher sensitivity than ProteinProphet at low FDR, which is fairly impressive. Overall, the method performs similarly to ProteinProphet; however, the utility of the nested mixture model is low, because the treatment of degenerate peptides (essentially assuming that no peptides are degenerate) will cause the model to perform poorly on data sets from complex organisms such as human, whose graphs feature substantial peptide degeneracy. Furthermore, the treatment of degenerate peptides resembles the one- and two-peptide rules, and may result in an overestimate of sensitivity, because target peptides are more likely to have higher scores and target proteins are more likely to share degenerate peptides with other target proteins. Also, like the one- and two-peptide rules, the independent treatment of peptides makes this method very computationally efficient.

In [18], ProteinProphet is compared to MSBayes on a protein standard composed of 49 proteins [38] using a detectability model with hundreds of parameters chosen from another replicate of the same data set. MSBayes is shown to have a higher specificity but a lower sensitivity using a protein threshold of $\tau_X = 0.5$. Considering that the threshold is so lax, the fact that ProteinProphet is still more specific indicates that the MAP protein set is very permissive. Furthermore, the fact that so many parameters for the detectability model are estimated on essentially the same data set makes the results less meaningful; even though

the detectability model is shown to perform fairly well using parameters estimated from other data sets [34], it is highly probable that even fairly small changes in the quality of the peptide detectability estimates may result in large changes in the set of identified proteins.

MSBayes is not as efficient as ProteinProphet, but this may be because MSBayes is implemented in an interpreted language. The underlying MCMC (python) procedure could be reimplemented in a more efficient, compiled language to be roughly the same speed as ProteinProphet. The MCMC procedure jointly samples protein and peptide states, despite the fact that the model permits peptides to be conditionally independent of each other given the protein set. Using this conditional independence would permit the peptides to be marginalized out in linear time given a sampled protein state, dramatically reducing the space that needs to be sampled. Furthermore, d-separation can be exploited to ensure that a block sampling chooses protein blocks so that every protein in the block shares a peptide with another protein in the block. Otherwise, a block will be d-separated by proteins whose states have been sampled, and can be sampled independently, rather than jointly.

Chapter 2

FIDO: A NOVEL BAYESIAN APPROACH TO PROTEIN INFERENCE

In this chapter, a novel Bayesian method for computing posterior protein probabilities is presented. The approach, called "Fido," (an acronym of "fast identification with optimization") is motivated by a desire to derive a model using a few relatively simple assumptions, but also to create accompanying algorithms that make computation very efficient. Such a model will permit evaluation of the assumptions and allow us to systematically make improvements in a manner that is difficult with many current approaches. The Fido model uses only three parameters, which can be easily estimated using the same data set used for identifying proteins. With respect to the peptide deneracy problem, the Fido model rewards protein sets that contain independent evidence in addition to degenerate peptides. In particular, the model allows a protein with strong independent supporting evidence to "explain away" supporting data that is shared with other proteins. Thus, the Fido method automatically apportions information from degenerate peptides during the marginalization procedure, rather than requiring an *ad hoc* adjustment.

Using this model, a series of three mathematical transformations are introduced; these transformations alter the graph and substantially increase the computational efficiency of computing posterior probabilities, while still recognizing peptide degeneracy. The resulting algorithm is mathematically equivalent to the result achieved by *marginalizing* [33], the process of computing every possible set of present proteins and evaluating their net contribution. In contrast to sampling, marginalizing yields an exact, closed-form solution in a finite amount of time. Naively marginalizing would require enumerating every possible set of proteins, which is exponentially complex and hence computationally infeasible even for small problems, but the optimized marginalization procedure is significantly more efficient and computes the same result as the naive approach. Using the Fido method, it is possible to compute discriminative and interpretable posterior probabilities quickly, even on large data sets. This combination of efficiency and rigor allows us to compute accurate and well-calibrated posterior probabilities quickly, and

lays the groundwork for more complex models and more optimized procedures.

2.1 Data sets

The Fido method is compared to ProteinProphet [21] on four data sets, yeast lysate [12], *H. influenzae* lysate [21], the ISB 18 mix protein standard [16], and *C.* elegans lysate [10], and compared to MSBayes [18] on one additional data set, the Sigma Aldrich 49 protein standard [38]. These methods were chosen to compare to because they represent the state-of-the-art for methods that can perform inference on graphs containing degenerate peptides. Each collection of spectra was searched against a combined database of *target* and *decoy* proteins. For the purposes of the analyses, when a protein identification method identifies a protein from the database, that identification is considered a "true positive" or a "false positive," depending on whether the protein is a target or a decoy, respectively. It should be noted that treating the targets as true positives is not perfectly correct, because the target database is actually a mixture of true and false positives. Consequently, the empirical estimates of true positive counts may be slightly inflated; however, because these estimates of the true positives are only used as a relative comparison between methods, the slight bias introduced will not influence the comparison. Summary statistics describing each data set are given in Table 2.1.

H. influenzae *H. influenzae* lysate was digested with trypsin and analyzed by LC-MS/MS on an ESI-ITMS machine. The spectra were searched with SEQUEST [7] against a database containing *H. influenzae* (targets) and human proteins (decoys). The resulting PSMs were scored using PeptideProphet with a minimum peptide probability of 0.0.

Yeast *Saccharomyces cerevisiae* strain S288C were grown to mid log phase on rich media at 30°C. The proteins were digested with trypsin and analyzed using data dependent acquisition and LC-MS/MS, using an LTQ machine. The resulting spectra were searched using Crux [23] against a database consisting of all yeast ORFs plus a shuffled version of each ORF. PSMs were assigned probabilities by PeptideProphet using a minimum peptide probability of 0.05.

	H. influenzae	Yeast	ISB 18	*C.* elegans	Sigma 49
matched spectra	33350	35236	$1.1\text{x}10^6$	42091	32700
target proteins	1709	6734	34	23932	49
decoy proteins	88299	6734	1709	23932	31227
decoy database	human	shuffled	*H. influenzae*	reversed	human and reversed human

Table 2.1: **Data set sizes.** The table lists, for the five data sets, the number of fragmentation spectra produced, the number of proteins in the target and decoy databases, and the type of decoy proteins used. All numbers are reported before any analysis was performed; Table 2.2 reports the number of proteins and PSMs that are actually matched and connected in the bipartite graph. Proteins were counted using their unique accession numbers, so this is the true size of the database and is invariant to redundancy or homology. The different decoy databases are taken from existing publications of protein and peptide identification algorithms, and were selected to demonstrate that the Fido method performs well regardless of the decoy database used.

ISB 18 mix The ISB 18 protein data was created using proteins purchased from Sigma Aldrich. This data set consisted of four prepared samples, which were analyzed on a variety of mass spectrometry machines with several technical replicates. The proteins were digested together using trypsin, and in one of the four samples, digestion was aided by sonication. These peptides were analyzed using LC-MS/MS on a variety of machines, including LTQ, LCQ Deca, Q-TOF, QSTAR, AGILENT XCT Ultra, Applied Biosystems ABI 4800, AppliedBiosystems 4700, and Thermofinnigan LTQ-FT. The spectra obtained from each experiment were searched using SEQUEST against a database containing these 18 proteins, a set of closely related homologs (obtained from the authors), which are indistinguishable from or may have been purified with the 18 proteins, possible contaminants, and a collection of *H. influenzae* proteins. PSMs were assigned probabilities by using PeptideProphet with a minimum peptide probability of 0.05. The contaminant proteins were not treated as present or absent, because they were identified using the Trans Proteomic Pipeline (http://tools.proteomecenter.org/wiki/index.php?title=Software:

TPP), which includes the ProteinProphet algorithm. The replicate experiments were analyzed in two ways: individually, and after pooling all experiments together.

C. elegans *C. elegans* were grown to various developmental stages on peptone plates containing *E.* coli. After removal from the plate, bacterial contamination was removed by sucrose floating. The lysate was sonicated and digested with trypsin and subject to six technical replicate LC-MS/MS analyses using LTQ machine and data dependent acquisition. The spectra were searched against a database containing the target proteins, the *C. elegans* proteome and known contaminants, as well as a reversed copy of every target protein. PeptideProphet was run using a minimum PSM probability of 0.05.

Sigma 49 mix The Sigma 49 mixture was prepared using 49 human proteins from Sigma Aldrich. The proteins were digested with trypsin and subjected to three replicate LC-MS/MS analyses using a Thermo LTQ machine. The spectra were searched using MyriMatch16 [31] against a database composed of all Swiss-Prot (54.2) proteins with the _HUMAN as well as a reversed copy of each protein. During the database search, any spectra that matched multiple peptide sequences and that also received equal scores for these matches were excluded. The remaining PSMs were scored using PeptideProphet and any PSMs with probability less than 0.05 were thrown out.

2.2 Results

2.2.1 A probability model for scoring candidate solutions

To compute the desired protein posterior probabilities, the tandem mass spectrometry process is modeled using a generative Bayesian probability model. The Fido model follows directly from a series of seven simple assumptions, which are illustrated in Figure 2.1 and described in detail below. First, however, some terminology is introduced. A peptide is *emitted* by a protein if that peptide was created by digesting a protein, retrieved by the precursor scan, and analyzed by the fragmentation scan. A peptide is created by the *noise model* if the peptide was identified by the fragmentation scan but the scan was not derived from that peptide.

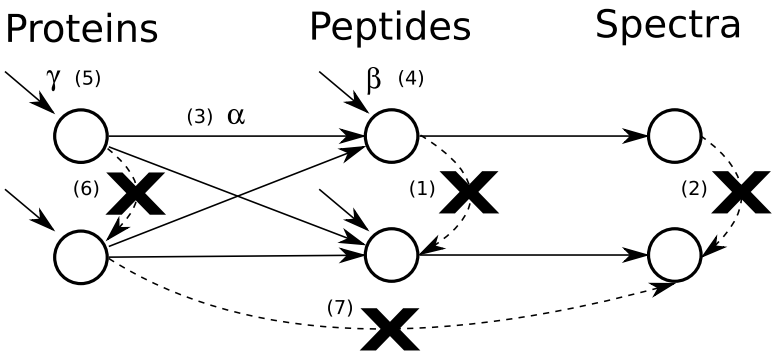


Figure 2.1: **Assumptions.** The assumptions of the model are illustrated graphically and numbered by their corresponding assumption numbers from Section 2.2.1. Solid arrows represent dependencies. Peptides depend on the proteins and the noise model; present proteins emit associated peptides with probability α, and peptides that are not created by associated proteins are created by the noise model with probability β. Spectra depend exclusively on the best-matching peptide. Proteins have identical and independent prior probabilities γ. The marked-out dashed arrows depict dependencies that do not exist within the model.

Qualitatively, the assumptions underlying the Fido model are as follows (more rigorous and complete quantitative descriptions of these assumptions are presented in the chapter "Quantitative derivation of inference and graph transformations"):

- The process by which one peptide is retrieved from the precursor scan does not influence the retrieval of other peptides from the precursor scan given the set of proteins in the sample.

- The process by which a spectrum is created and observed does not influence the creation and observation of other spectra given the set of peptides selected by the precursor scan.

- The event in which a sample peptide is generated from a present protein containing it occurs with constant probability α. This event is independent of all other emission events. Although the probability that a peptide is retrieved may depend on properties of the peptide, the model can account for these variations by adjusting the probability of the

PSM. Adjusting the probability of a PSM is equivalent to adjusting the probability that the peptide was retrieved from the precursor scan. These events are only observable in conjunction, so it is not possible to distinguish between the event where a peptide is retrieved from the precursor scan but its spectrum is mistakenly assigned and the event where a peptide is not retrieved from the precursor scan and undergoes no fragmentation scan.

- The event that a truly absent peptide (i.e. not created by an associated protein) is erroneously observed occurs with constant probability β.

- The prior event that any protein is present in the sample occurs with probability γ. It would be possible to later introduce a more complex prior, but doing so may increase the runtime of the algorithm.

- The prior probabilities of all proteins are independent.

- Each spectrum depends exclusively on the peptide that it best matches and is paired with to form a PSM.

The probability model that results from these assumptions is sufficient to compute the likelihood of a particular set of proteins given the the observed set of spectra, which is proportional to the probability that these proteins would create the observed spectra:

$$
\begin{aligned}
& L(X = x|D) \propto \Pr(D|X = x) \\
= \quad & \sum_{y} \prod_{j} \Pr(D_{k(j)}|Y_j = y_j) \Pr(Y_j = y_j|X = x) \\
= \quad & \sum_{y_1} \sum_{y:y_1} \prod_{j} \Pr(D_{k(j)}|Y_j = y_j) \Pr(Y_j = y_j|X = x) \\
= \quad & \sum_{y_1} \Pr(D_{k(1)}|Y_1 = y_1) \Pr(Y_1 = y_1|X = x) \sum_{y:y_1} \prod_{j \neq 1} \Pr(D_{k(j)}|Y_j = y_j) \Pr(Y_j = y_j|X = x) \\
= \quad & \prod_{j} \sum_{y_j} \Pr(D_{k(j)}|Y_j = y_j) \Pr(Y_j = y_j|X = x)
\end{aligned}
$$

where X is the set of present proteins, Y is the set of present peptides, D represents the observed spectra, and j is used to index the peptides. Both X and Y are random variables representing the truly present protein and peptide sets; x and y are specific values taken on by these random variables. The first equation above removes uncertainty from the unknown peptide set Y by marginalizing over all possible peptide sets (i.e., all possible values Y can take on, which is denoted by summing over y). For example, if the set of spectra match 10,000 distinct peptides, then the enumeration over all possible values of y must consider $2^{10,000}$ possibilities.

Values proportional to $\Pr(D_j|Y_j = y_j)$ are computed using PeptideProphet and $\Pr(Y_j = y_j|X = x)$ using the model of peptide generation. This former is actually computed by an intermediate step in the PeptideProphet algorithm and can be recovered by applying Bayes' rule to PeptideProphet's probability scores and prior probability estimates. The conditional independence of peptides given proteins allows us to compute the sum over all peptide sets in linear time (rather than exponential time), by transforming the sum into an equivalent product over peptide indices. Essentially, the procedure between Equations (2) and (4) can be repeated on the right sum in Equation (4) using a different peptide index. This operation can be continued inductively on each successive sum, effectively unrolling the sum of products into a product of sums. In the product of sums form, each sum has only two states (a particular peptide is present or absent), so each term in the product is trivial, permitting the likelihood of a set of proteins to be computed in linear time relative to the number of peptides. From a graphical model perspective, once the set of proteins is specified, all of the PSMs are disconnected from each other, making an independent graph for each PSM. The likelihood is thus computable by a product over these independent graphs. A more complete derivation, as well as other derivations used for the Fido model and optimizations, are provided in the chapter "Quantitative derivation of inference and graph transformations."

2.2.2 Computing posterior probabilities for each protein

By applying Bayes rule to the likelihood proportional to $Pr(D|X = x)$ and marginalizing over all possible protein configurations x, it is possible to compute a posterior probability for each protein. This approach appears to be prohibitively expensive, because a naive implementation

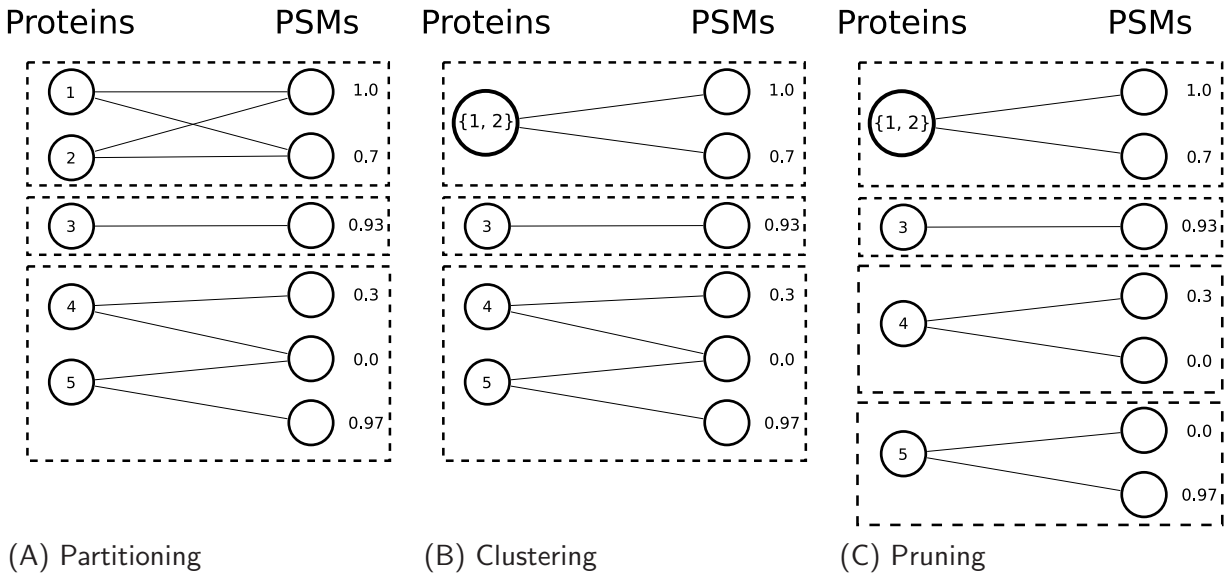


Figure 2.2: **Three speedups. (A)** The graph is partitioned into connected subgraphs (enclosed by dashed boxes). Posterior probabilities can be computed individually for each connected subgraph. **(B)** Proteins with identical connectivity are clustered together. In this example, proteins 1 and 2 are clustered to more efficiently enumerate the power set. **(C)** Graph components joined only by zero-probability peptides are separated by creating a copy of each of these peptides for each subsection. This operation further divides existing partitions to create partitions containing fewer proteins.

of this marginalization requires explicitly enumerating every possible set of proteins (a so-called "power set"). The computational cost of enumerating this power set is exponential in the number of proteins, making the naive implementation impractical for even small data sets. However, in practice, it is not necessary to enumerate the power set of peptides, because peptides are assumed to be conditional independent given the protein configuration; therefore, when a protein set is specified, these assumptions permit marginalization over all peptide sets in linear time using a single product.

In order to make computation of posterior probabilities computationally feasible for large data sets, three graph-transforming procedures are introduced: partitioning, clustering, and pruning.

These procedures, illustrated in Figure 2.2, dramatically increase the efficiency of computing posterior probabilities for the proteins. All of these procedures are described thoroughly in the chapter "Quantitative derivation of inference and graph transformations."

Speedup #1: Partitioning

In the Fido model, a protein is dependent on other proteins within connected subgraphs, but not dependent on proteins that share no peptides with proteins in the connected subgraph. This property is exploited to compute posterior probabilities for proteins in a subgraph by enumerating over the power set of proteins in the subgraph. The original graph can be trivially partitioned into connected subgraphs by performing depth-first search on the undirected graph. When a specific digest, such as trypsin, is used, this transformation considerably decreases the number of protein sets that need to be evaluated.

Speedup #2: Clustering

In the Fido probability model, it can be demonstrated that proteins with identical connectivity can be clustered together to compute their posterior probabilities with greater efficiency (a more thorough derivation is described in the chapter "Quantitative derivation of inference and graph transformations"). In Figure 2.2, proteins 1 and 2 are indistinguishable; therefore, the case in which protein 1 is present and protein 2 is absent has the same probability as the case in which protein 1 is absent and protein 2 is present. Thus, these two proteins can be merged into a single node (Figure 2.2B), which can occupy three distinct states: a state with both proteins absent, a state with only a single protein present, and a state with both proteins present. The state where a single protein is present must now be counted twice because there are two ways for it to occur. Using this transformation, it is possible to enumerate the power set in three steps rather than four. Generally, merging n proteins reduces the number of states that must be enumerated from 2^n to $n + 1$.

Speedup #3: Pruning

Furthermore, it is proven that within a connected subgraph, any two partitions of proteins that are only connected by peptides with a probability of zero can be transformed into two subgraphs that do not connect to one another. These zero-probability peptides are often produced by PeptideProphet when the best spectrum match for the peptide is a very poor match. Because they are a special case, each zero-probability peptide implies two necessary events: first, the peptide cannot be emitted by any protein, and second, the peptide cannot be created by the noise model; for if the peptide were emitted by a protein or created by the noise model, it would necessarily raise its probability above zero, resulting in a contradiction. When two protein partitions within a subgraph are connected only through zero-probability peptides, then neither partition may emit any of those zero-scoring peptides.

The pruning operation copies each zero-probability peptide so that each of these protein partitions connects to its own copy; therefore, these necessary events remain the same, except the event that the peptides are not created from noise is now counted twice instead of once, because a copy has been added. This overcounting can be corrected easily, transforming the original problem into two partitioned subproblems. A more thorough derivation is shown in the chapter "Quantitative derivation of inference and graph transformations."

In Figure 2.2C, proteins 4 and 5 are only connected by a zero-probability peptide. The only possible events that would produce the observed data require that neither protein 4 nor 5 emit the peptide and require that the peptide not be created by the noise model. Creating a copy of the peptide for each of these proteins and then correcting so that the noise model is only counted once will produce the same posterior probability for each protein.

Table 2.2 illustrates the effects of these three speed-ups on five different data sets. The first three rows of the table indicate the size of the input graph, the next four rows list the size of the corresponding search space initially and after each of the three graph transformations, and the remaining row shows the runtime of the algorithm. In the most extreme case, *H. influenzae*, the graph transformations reduce the size of the search space by nearly 10,000 orders of magnitude. By reducing the theoretical complexity of the procedure, these graph optimizations lead to efficient runtimes, as shown in the last row of Table 2. In comparison, ProteinProphet took

	H. influenzae	Yeast	ISB 18	*C.* elegans	Sigma 49
PSMs	29123	10390	21166	4944	23964
proteins	32748	3742	1777	4303	392
edges	60844	12202	21720	7332	24392
$Log_2\#$	32764.6	11495.6	1833.7	4316.3	407.6
$Log_2\#$ PRT	935.2	90.4	71.6	40.0	92.3
$Log_2\#$ PRT, CLST	72.6	46.96	71.6	16.2	92.3
$Log_2\#$ PRT, CLST, PRUNE	18.3	46.96	31.4	16.2	16.9
Runtime	1.4s (0.1s)	1.6s (0.2s)	8.6s (4.3s)	1.3s (0.1s)	1.0s (0.1s)

Table 2.2: **Utility of the Optimizations.** The table lists, for the five data sets, the size of the protein identification graph (using the combined target and decoy databases) in terms of the number of PSMs (after identical peptides are merged), number of proteins, and number of PSM-to-protein edges, as well as the log of the size of the full search space and the search space after each of the three optimizations are successively applied. PRT indicates partitioning, CLST indicates clustering, and PRUNE indicates pruning (using a theshold of 10^{-3}). As before, the proteins are counted by their unique accession numbers; however, peptides are counted using their sequence, so degenerate peptides are only counted once. The final row lists the running time necessary to compute the posterior probabilities for a fixed value of the parameters α, β, and γ using a variable threshold and allowing for up to 2^{18} marginalization steps for each subgraph (using a single-core standard desktop computer). The first runtime listed is the total execution time (including file I/O to read the data); the runtime in parentheses is the time necessary to only run the marginalization procedure. The *H. influenzae* data has the highest degree of peptide degeneracy, because it includes a human decoy database.

13.3s on the yeast data and 10.7s on the ISB 18 data. MSBayes took 2m23.5s on the Sigma 49 data. Due to lack of access to the proper files, it was not possible to time ProteinProphet on the *H. influenzae* data.

Unfortunately, even after the transformations, the search space associated with the larger data sets is still prohibitively large. In order to guarantee the efficiency of the Fido method on large data sets, the original problem can be approximated by pruning low-scoring PSMs as if they were zero-scoring PSMs; in practice, this approximation generally achieves great efficiency with little error, because the likelihoods computed using PeptideProphet are hardly sacrosanct. With this approximation, the pruning procedure creates subgraphs with many fewer proteins. Because the user is only interested in using the smallest threshold that will sufficiently break apart the connected subgraphs, this process is performed recursively and divide each subgraph using a successively greater flooring threshold. This process is continued until the total number of steps necessary for marginalization is less than a user-specified value. The result is that, rather than choosing one strict threshold for the entire data set, the user can specify a permitted computational complexity, and then different thresholds are employed to ensure that the method is as efficient as the user requires.

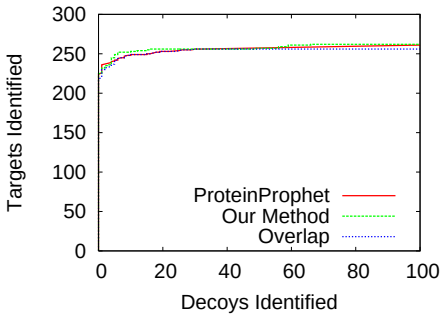
Occasionally, it is necessary to apply the pruning procedure to a PSM with a larger probability. In these cases, a collection of proteins are connected through a collection of high-scoring PSMs. These cases are already known to be difficult; in the extreme case, when all PSMs have probability 1.0, this problem closely resembles the NP-hard minimal set coverage problem (except, in the Fido method, marginalization requires that each permutation of present and absent protein states must be considered). Fortunately, any error introduced by pruning will only distort the probabilities of proteins connected in this way; therefore, accurate protein posteriors may be achieved as long as these cases are relatively rare. Chapter "Quantitative derivation of inference and graph transformations" shows the distribution of PSM probabilities that must be pruned to achieve no more than 2^{18} marginalization steps, and demonstrate that few pruned PSMs have probabilities greater than zero. When such a PSM is pruned, the two partitions it joins are approximated as being independent (even though they may not be). In these cases the Fido method behaves similarly to the first iteration of ProteinProphet, by treating the peptides as independent.

2.2.3 Comparison of the Fido method to ProteinProphet and MSBayes

A C++ implementation of the Fido method is evaluated using the five data sets described in Materials and Methods. The source code of this implementation is publicly available (http://noble.gs.washington.edu/proj/fido). Starting from the scored peptides, each method computes a probability for each protein, and these probabilities are used to rank the proteins. Groups of identically connected proteins are merged for evaluation, and are treated as a single protein group. All references to the number of target proteins or decoy proteins identified at a threshold or use these values in a calculation, each protein group is counted once, rather then once for each protein it contains. Groups containing both targets and decoys are not counted in evaluation; such groups are so infrequent that their treatment doesn't visibly change the figures presented.

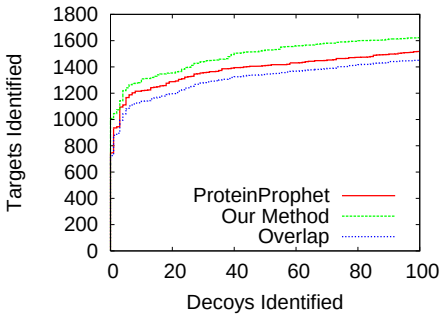
From each ranked list of proteins, the method producing the ranked list is evaluated by creating a receiver operator characteristic (ROC) curve, which plots true positive counts (i.e., the number of target proteins) as a function of false positive counts (the number of decoy proteins). A curve is produced by varying the probability threshold above which a protein is deemed to be present. Because the performance in the low false positive rate is particularly interestin, the plot depicts the ROC curve out to 100 false positives along the x-axis. Each ranked list of proteins is also evaluated using a calibration FDR plot, which plots the empirical FDR as a function of the estimated FDR. The empirical FDR is calculated as the number of decoys identified divided by the total number of proteins identified. The FDR is estimated from posteriors by exploiting the fact that the probability that a protein is absent is equal to one minus the posterior assigned to the protein; therefore, by assuming that the posterior probabilities are independent, it is possible to estimate the FDR for any set of proteins by computing the expected number of false positives (found by the number of proteins minus the sum of their posteriors) divided by the number of proteins identified at the threshold. If the Fido method is perfectly calibrated and if the empirical FDR estimate is accurate, then the empirical FDR and estimated FDR should be equal at every threshold.

Figure 2.3 shows, for each data set, ROC curves for the Fido method and either ProteinProphet or MSBayes. Fido is compared against ProteinProphet for the first four data sets, and



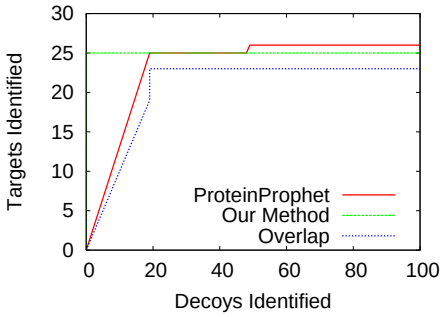
(A) *H. influenzae*

$\alpha^* = 0.16, \beta^* = 0.00, \gamma^* = 0.1$



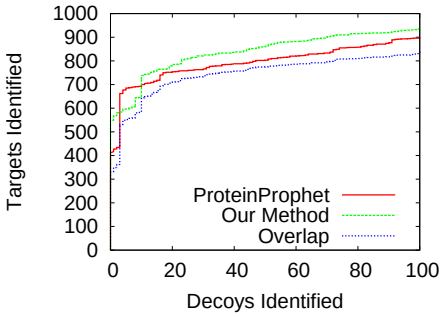
(B) Yeast

$\alpha^* = 0.06, \beta^* = 0.00, \gamma^* = 0.7$



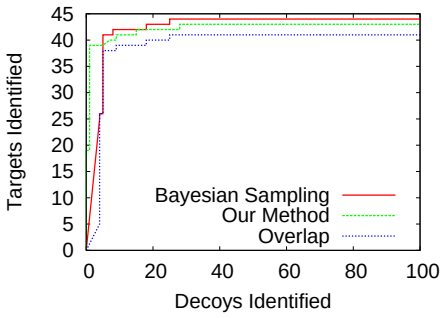
(C) ISB 18

$\alpha^* = 0.11, \beta^* = 0.05, \gamma^* = 0.1$



(D) *C. elegans*

$\alpha^* = 0.01, \beta^* = 0.00, \gamma^* = 0.7$



(E) Sigma 49

$\alpha^* = 0.36, \beta^* = 0.05, \gamma^* = 0.9$

Figure 2.3: **ROC plots.** For each data set, the plot shows the number of true positives as a function of the number of false positives. The Fido method is compared to ProteinProphet in **(A)–(D)** and compared to the ABLA model of MSBayes in **(E)**. The figure also shows the overlap between the sets of true positive proteins found at each false positive level. The Fido method is run on each data set with the same set of parameters used for the same data set in Figure 2.4.

MSBayes for the Sigma 49 protein mixture. ProteinProphet has previously been demonstrated to perform similarly to MSBayes on this data set [18]. Fido is not compared against MSBayes on any other data sets, because the model it employs was trained to be used for the Sigma 49 protein mixture; hence, attempting to compare performance on another data set would be unfair. The ISB 18 protein data set includes many replicate analyses, so the ROC for protein inference uses the results from pooling these replicate data sets. On the yeast data set, the Fido method performs better than ProteinProphet. On the *H.* influenzae data, the Fido method performs nearly identically to ProteinProphet. On the Sigma 49 data set, the Fido method performs similarly to MSBayes, but achieves a smaller minimum FDR. For the pooled ISB 18 data set, a substantial improvement is observed over ProteinProphet in the low false positive region. The diagonal line produced by ProteinProphet for the ISB 18 data set corresponds to 25 proteins that each receive a probability of exactly 1.0; because this data set includes many spectra, many PSMs associated with the decoy proteins are assigned scores larger than zero, and the ProteinProphet algorithm assigns these decoy proteins scores of 1.0.

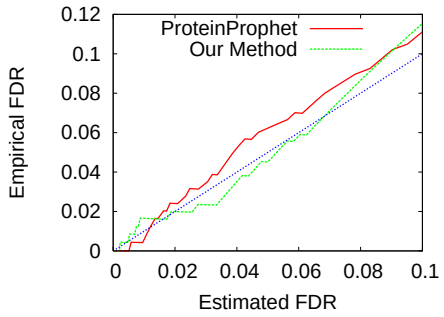
In addition to ROC curves, the plots in Figure 2.3 contain series labeled "Overlap," corresponding to the number of proteins identified by both methods at a given number of false positives. In every case, the overlap line is very close to the ROC curves, indicating that the methods are consistent with one another and identify a largely overlapping set of proteins at each score threshold.

Table 2.3 depicts the sensitivity of the methods at different empirical FDRs. The Fido method outperforms ProteinProphet on the yeast data and performs significantly better than ProteinProphet on the ISB 18 data set. On the *H. influenzae* data set, the Fido method performs almost identically to ProteinProphet; this similarity can also be observed in the ROC plot in Figure 2.3A. On the *C.* elegans data set, the Fido method performs better at the 0.0 FDR, worse at the 0.01 FDR and better for higher FDRs. On the Sigma 49 data set, the Fido method performs better than MSBayes, which does not achieve a FDR less than 0.10.

Figure 2.4 shows, for each data set, the calibration of the posterior probabilities assigned by the different methods. In these figures, the estimated FDR is compared to the empirical FDR. On the yeast data set the Fido method has better calibration accuracy compared to ProteinProphet. On the Sigma 49 data set, the Fido model achieves better calibration than MSBayes. On the

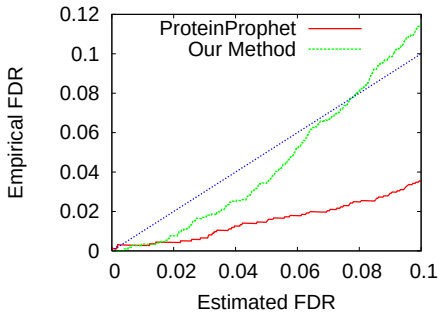
FDR	*H. influenzae*		Yeast		ISB 18		*C. elegans*		Sigma 49	
	F	PP	F	PP	F	PP	F	PP	F	MSB
0.0	**225**	224	**1008**	745	**25**	—	**549**	412	**19**	—
0.01	235	**237**	**1320**	1225	**25**	—	602	**687**	**19**	—
0.05	**254**	249	**1603**	1471	**25**	—	**849**	788	**39**	—
0.10	**256**	255	**1778**	1566	**25**	—	**954**	900	**39**	—

Table 2.3: **True Positive Identifications vs. Empirical FDR.** The table lists, for the five data sets, the number of true positive identifications that the methods achieve at the greatest empirical FDR not exceeding 0.0, 0.01, 0.05, and 0.10. Methods are abbreviated as PP = ProteinProphet, MSB = MSBayes, and F = Fido, the optimized Bayesian marginalization method. The true positive counts are boldfaced if that method is better for that data set and at that particular empirical FDR. All FDR values for ProteinProphet applied to the ISB 18 data are missing because the minimum FDR attainable was greater than 0.43. All FDR values for MSBayes applied to the Sigma 49 data are missing because the minimum FDR attainable was greater than 0.10.



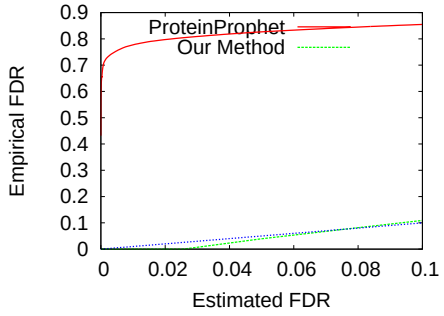
(A) *H. influenzae*

$\alpha^* = 0.16, \beta^* = 0.00, \gamma^* = 0.1$



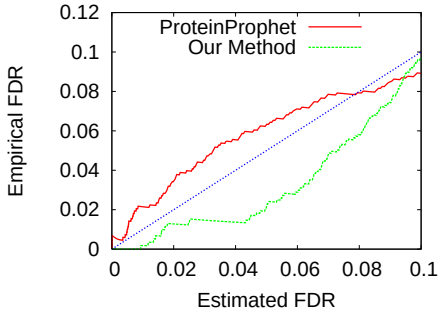
(B) Yeast

$\alpha^* = 0.06, \beta^* = 0.00, \gamma^* = 0.7$



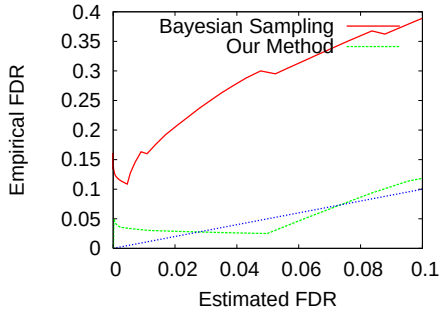
(C) ISB 18

$\alpha^* = 0.11, \beta^* = 0.05, \gamma^* = 0.1$



(D) *C. elegans*

$\alpha^* = 0.01, \beta^* = 0.00, \gamma^* = 0.7$



(E) Sigma 49

$\alpha^* = 0.36, \beta^* = 0.05, \gamma^* = 0.9$

Figure 2.4: **FDR calibration plots.** For each data set, the plot shows the decoy database estimate of the empirical FDR as a function of the estimated FDR. The Fido method is compared to ProteinProphet in **(A)–(D)** and compare to the ABLA model of MSBayes in **(E)**. The line along which the two axes are equal is also shown; this line represents an ideally calibrated model (without considering bias introduced in the estimation of the empirical FDR). The Fido method was run on each data set with the same set of parameters used for the same data set in Figure 2.3.

ISB 18 data set the Fido method is much better calibrated than ProteinProphet. On the *H. influenzae* data set, both the Fido model and ProteinProphet are very well calibrated and achieve similar results. On the *C.* elegans data set, the Fido method's calibration is similar or slightly inferior to ProteinProphet.

estimates will necessarily include some error. In particular, all decoy proteins are known false positives but not all target proteins are always present. As a result, target-decoy estimation may underestimate the empirical FDR in Figure 2.4. However, this observation does not alter the conclusion that the Fido model is similarly or better calibrated compared to ProteinProphet and MSBayes. For four of the five sets (*H. influenzae*, ISB 18, *C.* elegans, and Sigma 49) the FDR calibration curve is nearly identical to or below the curve from ProteinProphet, indicating that the Fido method is at least as conservative as ProteinProphet. Furthermore, on these data sets ProteinProphet is less conservative than an ideal model. Due to the high level of agreement among the algorithms in Figure 2.3, it is reasonable to assume that both curves would similarly move upward; therefore, any negative bias to the empirical FDR estimation would move both curves similarly upward, causing the Fido model to remain better calibrated than ProteinProphet. In other words, after correcting for absent targets, it is preferable to have a more conservative model, and the Fido model is more conservative on these data sets. Furthermore, the ISB 18 and Sigma 49 data sets consist of several proteins directly purified into the sample. In these cases, there should be little or no error to the estimated empirical FDR, because no proteins from the target database should be absent.

It cannot be certain whether the Fido method is better calibrated on the remaining yeast data set. However, at the 0.05 estimated FDR level (which will not be influenced by the potential empirical FDR bias), the empirical FDR is estimated to be 0.034. Even if the empirical FDR was underestimated by 50%, the Fido method would be nearly perfectly calibrated. For the Fido method to have significantly worse calibration than ProteinProphet, the bias towards absent targets would need to be substantial.

The Fido probability model requires the estimation of three free parameters, α, β, and γ. Parameters were empirically chosen so that they jointly maximizes the ROC_{50} score (the average sensitivity when allowing between zero and 50 false positives) and minimizes the mean squared error (MSE) from an ideally calibrated probability. The calibrated MSE is computed by

integrating the square of the difference between the estimated and the empirical FDR over the estimated FDR range $[0, 0.1]$. A rough three-dimensional grid search is then performed in the range $[0.01, 0.76]$ at resolution of 0.05 for α, in the range $[0.00, 0.80]$ at resolution 0.05 for β, and in the range $[0.1, 0.9]$ at resolution 0.1 for γ. For each triplet of parameters, both the ROC_{50} and the calibration MSE are computed. For each data set, the triplet of parameters is selected to result in an acceptable compromise between the most accurate model and the best-calibrated model. In order to demonstrate that this comprimise can be achieved objectively, this comprimise was phrased as an optimization problem, by which the quantity $(1 - \lambda)MSE - \lambda ROC_{50}$ was minimized, where λ is a parameter selected to emphasize ROC_{50} or MSE; a λ approaching 1.0 will shift the emphasis to the most accurate model, and a λ approaching 0.0 will result in a more calibrated model. Every data set uses $\lambda = 0.15$.

Because three parameters are chosen for each data set, it cannot be certain that the observed differences in performance between the Fido method and ProteinProphet or MSBayes are not partially due to overfitting. On the other hand, the optimal α and β parameter values are similar for these data sets.

Furthermore, the influence of the γ parameter is limited because it is estimated with very low resolution, and very few bits of precision are used to define it. Also, the γ parameter almost exclusively contributes to calibration because it upweights or downweights all proteins in a similar manner; using a fixed γ of 0.5, which is equivalent to using a uniform prior for all protein sets, and performing the grid search for only α and β resulted in nearly identical ROC figures. The risk of overfitting is also decreased because parameters are chosen by jointly optimizing both the accuracy and calibration, which are independent values. To demonstrate the robustness of the Fido model to slightly suboptimal parameters, the values of α, β, and γ that were selected using the *H. influenzae* data set were applied to a each of the experiments in the ISB 18 data set. The Fido method attains a greater ROC_{50} score than ProteinProphet for $193/236$ (81%), even when using parameters chosen from completely different data.

Table 2.4 shows that the Fido method compares favorably to ProteinProphet and MSBayes when identifying proteins that contain a high-scoring degenerate peptide. On all of these data sets, the Fido method identifies no decoy proteins that contain a high-scoring degenerate peptide. Furthermore, it does so without blindly introducing a systematic bias against such proteins.

	H. influenzae				Yeast			
	F		PP		F		PP	
	TP	FP	TP	FP	TP	FP	TP	FP
simple proteins	228	1	227	1	983	5	886	2
proteins with degeneracy	2	0	6	0	251	0	163	1
	ISB 18				*C.* elegans			
	F		PP		F		PP	
	TP	FP	TP	FP	TP	FP	TP	FP
simple proteins	13	0	15	87	420	4	445	14
proteins with degeneracy	10	0	11	6	173	0	273	1
	Sigma 49							
	F		MSB					
	TP	FP	TP	FP				
simple proteins	27	1	36	5				
proteins with degeneracy	5	0	5	2				

Table 2.4: **Accuracy on Proteins Containing Degenerate Peptides.** The table lists, for the five data sets, the number of true positive and false positive proteins identified by each method using a probability threshold of 0.90. The proteins identified are separated into two classes: "proteins with degeneracy" are proteins that share a high-scoring (≥ 0.90) peptide with another protein and "simple proteins" do not share such a peptide with another protein. Proteins that share a high-scoring peptide only through protein grouping are not treated as proteins with degeneracy in order to prevent any artifacts due to the treatment of grouped proteins. Methods are abbreviated as PP = ProteinProphet, MSB = MSBayes, and F = Fido, the optimized Bayesian marginalization method. When high-scoring degenerate peptides must be resolved, the Fido protein identification method offers a favorable trade-off between sensitivity and specificity.

For instance, on the yeast data, 88 proteins are identified with degenerate peptides without introducing any false positives. On the ISB 18 and Sigma 49 data sets, the Fido method identifies nearly the same number of target proteins containing high-scoring degenerate peptides as the competing methods but without identifying any decoy proteins. In contrast, ProteinProphet and MSBayes identify six and two decoy proteins on these data sets, respectively. The only data set where the Fido method does not increase either the sensitivity or specificity without sacrificing the other is the *H. influenzae* data, in which the Fido method identifies four fewer degenerate target proteins but still maintains perfect specificity. These proteins are not a significant percent of the targets identified. It should be noted that on the *C.* elegans data set, many fewer target and decoy proteins are identified that contain high-scoring degenerate peptides, but on this data set, ProteinProphet is overly permissive, while the Fido method is overly conservative (as shown by Figure 2.4D). Lowering the protein threshold to 0.8 on the Fido method yields a superior sensitivity and identical or superior specificity for both categories of proteins.

Rather than treating the PeptideProphet values as probabilities and making an *ad hoc* correction, the Fido method analyzes all of the data in a gestalt manner. As a result, the Fido emission model can prevent a degenerate peptide from being counted twice, and the noise model in the Fido method can prevent spurious evidence from accumulating and awarding an absent protein a high score. On large, somewhat noisy data sets like the ISB 18 data set, ProteinProphet effectively aggregates a great deal of noise, resulting in many decoy proteins with estimated probabilities of 1.0. In contrast, the Fido method gives these decoy proteins smaller probabilities than ProteinProphet, and more importantly, gives the decoy proteins smaller probabilities relative to several target proteins.

Figure 2.5 shows a decoy protein (`gi|1573516|gb|AAC22189.1|`) that matches several PSMs, but the majority of these PSMs have fairly low scores and are the result of the enormous number of spectra. In contrast, most of the PSMs associated with the target protein `sp|P02643|TNNI2_RABIT` have scores above 0.99. The Fido method estimates that the target and decoy proteins have respective posterior probabilities of 1.0 (which is higher than any decoy protein posterior) and 0.00092. ProteinProphet assigns both proteins posteriors of 1.0, preventing them from being effectively ranked. For completeness, the decoy protein `gi|1573522|gb|AAC22195.1|`, which shares a common PSM with the target, is also shown.

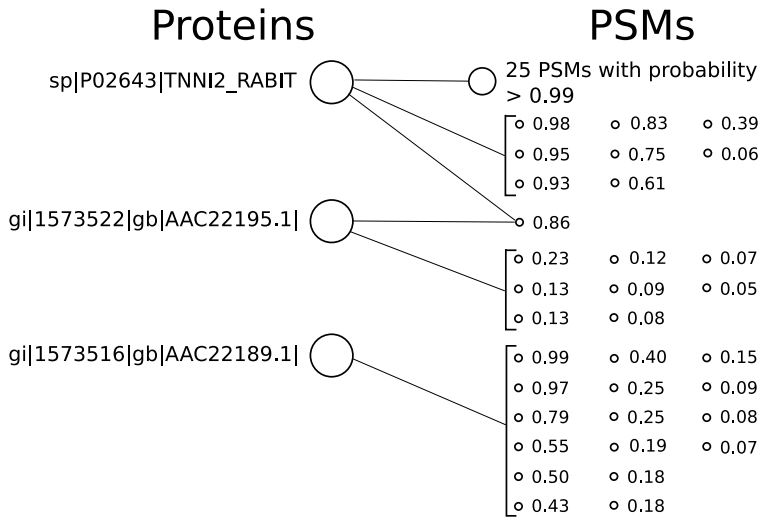


Figure 2.5: **Example of differently ranked ISB 18 proteins.** An example of three proteins from the ISB 18 data set are shown: one target (`sp|P02643|TNNI2_RABIT`) and two decoys (`gi|1573522|gb|AAC22195.1|` and `gi|1573516|gb|AAC22189.1|`). ProteinProphet assigns posteriors of 1.0 to the target protein `sp|P02643|TNNI2_RABIT` and to the decoy protein `gi|1573516|gb|AAC22189.1|`.

The Fido method likewise accumulates the relatively weak evidence supporting this decoy protein to estimate a weak posterior of 0.019 (which is lower than any target protein posterior). ProteinProphet estimates a 0.0 probability for the protein `gi|1573522|gb|AAC22195.1|`.

In general, even after partitioning, clustering and pruning zero-probability peptides, it cannot be sure that the running time of the Fido algorithm will not be prohibitively high. Therefore, as described above, the user is allowed to specify the log of the maximum size of the search space. If, after the graph transformations, one or more connected components contain too many nodes, then the probabilities of low-scoring PSMs in the offending components are temporarily set to zero, increasing the separability. The threshold for this zeroing procedure is adjusted recursively, on a per-component basis, to achieve the desired search space size (and, hence, running time). To test how the performance of the Fido method varies as the size of the search space is adjusted, the *H. influenzae* analysis was run multiple times with a fixed value for α and β but different search space sizes. Figure 2.6 shows that the performance (as measured by ROC_{50} score) improves in a step-wise fashion as the search space increases.

2.3 Discussion of accuracy and calibration evaluation

Using a simple and straightforward probability model, it has been demonstrated that protein posteriors can be efficiently computed using efficient marginalization with respect to a given collection of PSMs. The resulting posteriors provide rankings that often out-perform accurate and widely used existing methods, thus providing evidence that finding an exact or near-exact solution to this problem is beneficial.

Note that the timing results shown in Table 2.2 are slightly unfair, because the Fido procedure was run once rather than multiple times to select the parameters α and β. In the experiments reported in Figure 2.3, for example, the Fido procedure was run $\sim$1900 times; however, it is not expected that users will require such an exhaustive search in practice. Some parameter values do not realistically model the mass spectrometry process; for instance $\alpha = 0, \beta = 1$ would award all identified proteins with identical scores. Furthermore, empirically, the optimal parameters from the dense grid search are all fairly similar and would easily be found by a much lower resolution grid search. For instance, a search over $\alpha \in \{0.01, 0.04, 0.09, 0.16, 0.25, 0.36\}$, $\beta \in \{0.01, 0.025, 0.05\}$, $\gamma \in \{0.1, 0.5, 0.9\}$ requires 54 iterations $(9 \times 3 \times 3)$ and produces

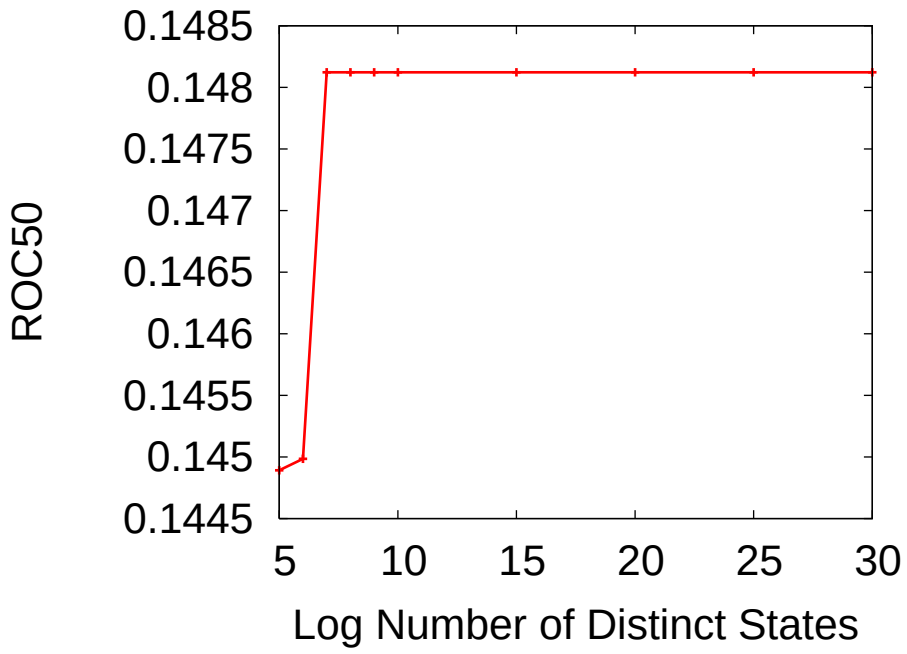


Figure 2.6: **Accuracy against the number of states.** The accuracy of the procedure on the *H. influenzae* data (measured by unnormalized ROC_{50} score) increases as the number of states allowed grows. A larger number of states corresponds to less aggressive pruning; a smaller number of states corresponds to pruning many moderately-scoring PSMs. The number of states can be thought of as the number of possible protein and peptide configurations that must be enumerated when marginalizing a connected subgraph with these optimizations. The *H. influenzae* has the most complex graph connectivity (due to the human decoy database used), as illustrated by the number of edges in Table 2.2.

produces similar parameter sets and very similar results. Running the Fido method 54 times (i.e. loading the data once and marginalizing 54 times) would take less time on the yeast data and 20 seconds longer than ProteinProphet on the ISB 18 data. Note, however, that the Fido method results in substantially better performance on the latter.

When the marginalization is too expensive, a pruning procedure efficiently approximates the full marginalization. Importantly, the effects of this approximation are restricted to the graph components in which they are carried out. Thus, when a PSM with a non-zero probability must be split during the pruning procedure, the pruning will only change the probabilities assigned to the proteins in that subgraph. In the future, it may be possible to bound the error introduced by pruning around non-zero nodes. This bound could be used to design pruning strategies that would minimize the error incurred. When multiple proteins share a large group of high-scoring peptides, the problem becomes very similar to the minimum set coverage problem, which is already known to be very difficult; it resides in the complexity class NP-Hard, making it equivalent to the Traveling Salesperson Problem. Hence, in these scenarios even a locally inaccurate approximation may be welcome if the runtime is efficient.

A relatively simple enumeration strategy was used to select the values of the three parameters, α, β, and γ. A more rigorous approach would set these parameters using cross-validation. However, such an approach would still require users to provide a decoy database. α, β, and γ are shown to be robust across different data sets: Fido attains a greater ROC_{50} score than ProteinProphet for 81% (193 of the 236) ISB 18 replicate data sets, even when using parameters chosen from the *H. influenzae* data set. It would be interesting, therefore, to investigate strategies for estimating these parameter values without using a decoy database.

ProteinProphet's analysis depends heavily on the data set being analyzed. ProteinProphet's strength and weakness are derived from its implicit assumptions and its reliance on PeptideProphet. ProteinProphet implicitly assumes that PeptideProphet scores are true, unbiased probability estimates. ProteinProphet does not use protein length or the number of associated peptides to correct the PeptideProphet scores associated with a protein. In a similar manner, ProteinProphet does not correct for the total number of spectra observed. When myriad spectra are observed, nearly every peptide has a strong chance of being matched to a spectrum with a high PeptideProphet score. When ProteinProphet's assumptions prove to be reasonable, they

add extra information and result in a very accurate and well-calibrated model (e.g., the *H. influenzae* data set). On data sets where these implicit assumptions are not helpful and may even be harmful, ProteinProphet performs similarly or worse than the Fido method (e.g., the yeast and ISB 18 data sets).

One significant difference between the Fido model and ProteinProphet is that the Fido model explicitly allows the possibility that a high-scoring PSM is the result of an error. In data sets containing many spectra, the chance of a protein associating with an erroneous high-scoring PSM becomes higher. This effect can be represented in the Fido model by using a larger value of the β parameter. It is important to note that even if the β parameter is larger than the α parameter, this does not mean that a peptide is more probably created from noise rather than from an associated protein. This is because β is used as the probability that a peptide is matched to a fragmentation scan given that the peptide was absent. When each protein is associated with several identified peptides, then the collective effect of these peptides make it very unlikely that they are absent, substantially lowering the influence of the noise model. ProteinProphet does consider the number of sibling peptides (NSP), the sum of other peptide scores sharing a protein association with a candidate peptide, when interpreting a score from PeptideProphet. But the manner by which this NSP correction lowers inflated peptide scores will not remove a systematic bias from all peptide scores.

The different manner in which the Fido model handles noise is indicative of a larger fundamental difference in how the Fido model employs PeptideProphet scores. ProteinProphet uses PeptideProphet scores as true probabilities and conditions on the NSP score to distinguish multi-hit proteins from so-called "one-hit wonders." Without conditioning on NSP, association with a single high-scoring peptide may be indistinguishable from association with many high-scoring peptides. In contrast, the Fido model removes the prior probability estimates used by PeptideProphet and converts them back into discriminant score-based likelihoods. The difference is that ProteinProphet initially interprets PeptideProphet scores as the probability that a peptide is present given a paired spectrum was observed, whereas the Fido method initially uses these scores to compute a value proportional to the probability that a spectrum would be created given that its paired peptide was present. For a given hypothesized set of present proteins, the Fido model will compute the likelihood that the spectra were observed given that set of proteins was

present. This subtle distinction lets the Fido model use protein-level information when utilizing the PeptideProphet scores to compute protein posteriors. A protein associated with many high-scoring peptides will score higher than a protein associated with a single high-scoring peptide, but without using an iterative heuristic like ProteinProphet's NSP score.

In a similar manner, the Fido method uses protein-level information in a rigorous, gestalt manner when handling degenerate peptides, rather than by using a correction in hindsight. ProteinProphet partitions every degenerate peptide's probability between its associated proteins. Initially, the peptide is split equally between them to compute the protein probabilities, but in the next iteration, each protein is afforded a stake proportional to its most recent probability. Like the NSP correction, this approach works well, but it does so in a somewhat opaque manner; hence, identifying its implicit assumptions and improving their effects is not trivial. Rather than heuristically partitioning each identified peptide amongst its associated proteins, the Fido model allows a protein with strong independent supporting evidence to "explain away" supporting data that is shared between itself and other proteins (e.g., degenerate peptides). This happens simply because the likelihood increases only slightly by including the protein with little independent supporting evidence, but the likelihood decreases substantially when a protein with substantial independent supporting evidence is omitted. In this manner, the Fido method automatically apportions information from degenerate peptides during the marginalization procedure and does not require an *ad hoc* adjustment.

Like other methods, the Fido model's assumptions are not perfect. But because the Fido model is derived from clear, explicitly stated statistical assumptions, it may be possible to evaluate their accuracy and replace them with more relaxed assumptions. For instance, when using data dependent acquisition, it may be inaccurate to assume that the process of retrieving one peptide from the precursor scan does not influence the process of retrieving other peptides; the population of peptides with equal hydrophobicity effectively compete in the MS1 to be selected for collision induced dissociation. The assumptions defining the noise model and mapping each spectrum to a unique peptide could be improved by treating spurious peptide identifications as mismatches between the observed spectrum and peptides that produce similar spectra at that precursor mass. The protein prior may be improved for certain samples by using more complex priors that more aggressively enforce competition between proteins for ownership of shared pep-

tides. But perhaps the most fruitful avenue for future work involves relaxing the assumptions regarding the peptide emission model and the noise model. Using the current model, it has been observed that likelihood estimates of α and β are not as good as the empirical estimates, suggesting that relaxed assumptions will better model the type of data observed in practice. It is intuitive that the likelihood estimates do not match the empirical estimates, because likelihood estimates essentially infer a uniform prior on all α, β pairs despite the inherent relatedness between the parameters (the contribution of β decreases as α increases or as graph connectivity increases). A more accurate model of these probabilities may preserve the presented optimizations, while taking into account peptide-specific information. Furthermore, a more sophisticated model would make far greater use of the entire graph by including peptides that are not matched to any spectra (without unfairly penalizing proteins that contain many undetectable peptides).

Chapter 3

QUANTITATIVE DERIVATION OF INFERENCE AND GRAPH TRANSFORMATIONS

This chapter uses simple, clearly stated assumptions to rigorously derive the Fido model, and prove that the graph transformations result in equivalent inference problems that can be solved more efficiently.

3.1 Statistical notation

Using standard statistical notation, random variables are given capital letters. For a random variable (for instance X) a particular value that can be taken on by that random variable is given with the corresponding lowercase letter (x). The event that the random variable takes the value is noted $X = x$.

For any random variable X, a sum over the variable x indicates the sum over all possible outcomes, or the power-set, which is equivalent to the sum over all possible values for the random variable X. If X_i indicates whether protein i is present or absent, then a sum over x_i will sum over the outcomes where X_i is present (written X_i) and the case when it is absent (written $\neg X_i$).

X	array of indicators for presence of proteins in the sample, indexed by $i, i', \ldots$
Y	array of indicators for presence of peptides in the sample, indexed by $j, j', \ldots$
D	observed data, with paired spectra and total masses indexed by k
$m_{i,j}$	indicator for protein i is expected to produce peptide j using the digest (equivalent to $(i,j) \in E$)
$\Delta_{i,j}$	the event that peptide j is produced by emission from a protein
H_j	the event that peptide j is produced by an event other than $\Delta_{i,j}$

The size of these arrays is denoted $size(X), size(Y), \ldots$.

The numbered equations correspond to a function in the final implementation; the accompanying source code (found at http://noble.gs.washington.edu/proj/fido).

The models of Δ and H are parameterized in terms of unknown rates α and β.

3.2 Assumptions

Assumption $A1$**:** Conditional Independence of Y_j and $Y_{j'}$ given X

$$\Pr(Y = y|X = x) = \prod_j \Pr(Y_j|X = x)$$

Assumption $A2$**:** Conditional Independence of D_k and $D_{k'}$ given Y

$$\Pr(D|Y = y) = \prod_k \Pr(D_k|Y = y)$$

Assumption $A3$**:** Probability of predicted peptide creation

$$\Pr(\Delta_{i,j}|X_i) \quad = \quad \begin{cases} \alpha, & m_{i,j} = 1 \\ 0, & else \end{cases}$$

Assumption $A4$**:** Probability of spontaneous peptide emission given the peptide is not created by proteins

$$\Pr(H_j|\forall i,\ \neg\Delta_{i,j}) \quad = \quad \beta$$

Assumptions $A5, A6$**:** Prior, independent belief on presence of proteins

$$\forall i, \Pr(X_i) = \gamma$$

$$\Pr(X = x) = \gamma^{|x|}(1 - \gamma)^{size(X)-|x|}$$

Assumption $A7$**:** Conditional Independence of D and X given Y

$$\Pr(D|Y = y, X = x) = \Pr(D|Y = y)$$

Assumption $A8$**:** Dependence of D_k only on best matching peptide Y_j

$$\Pr(D_k|Y = y) = \Pr(D_k|Y_{j(k)} = y_{j(k)}),$$

$$\Pr(D_k|Y_{j(k)} = y_{j(k)}, \alpha, \beta) = \Pr(D_k|Y_{j(k)} = y_{j(k)})$$

given $j(k)$ is the index of the best matching peptide to spectrum and precursor mass pair k

$$j(k) = \arg\max_{j'} \Pr(Y_{j'}|D_k)$$

$$k(j) = \arg\max_{k'} \Pr(Y_j|D_{k'})$$

Furthermore, a unique $j(k)$ is assumed to exist for each k, and a unique $k(j)$ is assumed to exist for each j.

Assumption $A9$**:** Independence of predicted peptide creation

$$\Pr(\Delta_{i,j}, \Delta_{i',j}) = \Pr(\Delta_{i,j}) \Pr(\Delta_{i',j})$$

3.3 Method

3.3.1 Using likelihoods from adjusted PeptideProphet probabilities

It is important to distinguish peptide probabilities that use protein information from peptide probabilities that do not use protein information (as well as their corresponding priors). PeptideProphet estimates a posterior probabilitie for each peptide without utilizing information in the associations between proteins and peptides. The Fido model can make use of the PeptideProphet estimates by using them to compute likelihoods that a peptide emitted it's paired spectrum.

By default, this document assumes that protein information is being used, but when protein information is not being used (for instance when PeptideProphet values are used), then conditioning on Q indicates that protein-level information is not available. For instance $\Pr(Y_j)$ is the prior probability that peptide Y_j is present using protein information, whereas $\Pr(Y_j|Q)$ is the prior probability of Y_j given by PeptideProphet. This prior probability should take into account

the assumed charge state for this PSM. The correct prior values for different charge states are found in the pepXML output of PeptideProphet.

In this document, conversion to probabilities that do not depend on the proteins will be accomplished using the fact that

$$\Pr(D_{k(j)}|Y_j) = \Pr(D_{k(j)}|Y_j, Q)$$

using assumption $A8$.

3.3.2 Method given known α, β, γ

Let the proteins and peptides be partitioned so that all proteins in each partition have no connections to peptides in another partition, and all peptides in each partition have no connections to proteins in another partition. This can be trivially accomplished by tracing the graph with depth first search. Denote the protein set corresponding to partition u as $x^{(u)}$ and the peptides in the partition $y^{(u)}$.

$$\Pr(X^{(u)} = x^{(u)}|D) = \frac{\Pr(D|X^{(u)} = x^{(u)}) \Pr(X^{(u)} = x^{(u)})}{\sum_{x^{(u)'}} \Pr(D|X^{(u)} = x^{(u)'}) \Pr(X^{(u)} = x^{(u)'})}$$

This probability can be defined in terms of a likelihood function:

$$\Pr(X^{(u)} = x^{(u)}|D) = \frac{L(X^{(u)} = x^{(u)}|D) \Pr(X^{(u)} = x^{(u)})}{\sum_{x^{(u)'}} L(X^{(u)} = x^{(u)'}|D) \Pr(X^{(u)} = x^{(u)'})}$$

$$\begin{aligned}
\Pr(D|X^{(u)} = x^{(u)}) &= \sum_{y^{(u)}} \Pr(D|Y^{(u)} = y^{(u)}) \Pr(Y^{(u)} = y^{(u)}|X^{(u)} = x^{(u)}) \\
&= \sum_{y^{(u)}} \prod_k \Pr(D_k|Y^{(u)} = y^{(u)}) \Pr(Y^{(u)} = y^{(u)}|X^{(u)} = x^{(u)}) \\
&= \sum_{y^{(u)}} \prod_{j \neq i} \prod_k \Pr(D_k^{(j)}) \prod_j \Pr(D_{k(j)}^{(u)}|Y_j^{(u)} = y_j^{(u)}) \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)}) \\
\Pr(D_{j(k)}^{(u)}|Y_j^{(u)} = y_j^{(u)}) &= \Pr(Y_j^{(u)} = y_j^{(u)}|D_{j(k)}^{(u)}, Q) \frac{\Pr(D_{k(j)}^{(u)}|Q)}{\Pr(Y_j^{(u)} = y_j^{(u)}|Q)}
\end{aligned}$$

$$\Pr(D|X^{(u)} = x^{(u)}) =$$

$$\prod_j \left[\prod_k \Pr(D_k^{(j)}|Q) \right] \sum_{y^{(u)}} \prod_j \frac{\Pr(Y_j^{(u)} = y_j^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_j^{(u)} = y_j^{(u)}|Q)} \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)})$$

$$L(X^{(u)} = x^{(u)}|D) =$$

$$\sum_{y^{(u)}} \prod_j \frac{\Pr(Y_j^{(u)} = y_j^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_j^{(u)} = y_j^{(u)}, Q)} \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)})$$

$$= \frac{\Pr(Y_1^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_1^{(u)}, Q)} \Pr(Y_1^{(u)}|X^{(u)} = x^{(u)})$$

$$\sum_{y^{(u)}:y_1^{(u)}} \prod_{j \neq 1} \frac{\Pr(Y_j^{(u)} = y_j^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_j^{(u)} = y_j^{(u)}, Q)} \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)})$$

$$+ \frac{\Pr(\neg Y_1^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(\neg Y_1^{(u)}, Q) \Pr(\neg Y_1^{(u)}|X^{(u)} = x^{(u)})}$$

$$\sum_{y^{(u)}:\neg y_1^{(u)}} \prod_{j \neq 1} \frac{\Pr(Y_j^{(u)} = y_j^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_j^{(u)} = y_j^{(u)}, Q)} \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)})$$

$$= \frac{\Pr(Y_1^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_1^{(u)}, Q)} \Pr(Y_1^{(u)}|X^{(u)} = x^{(u)})$$
$$\sum_{y^{(u)}:y_1^{(u)}} \prod_{j\neq 1} \frac{\Pr(Y_j^{(u)} = y_j^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_j^{(u)} = y_j^{(u)}, Q)} \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)})$$
$$+ \frac{\Pr(\neg Y_1^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(\neg Y_1^{(u)}, Q) \Pr(\neg Y_1^{(u)}|X^{(u)} = x^{(u)})}$$
$$\sum_{y^{(u)}:y_1^{(u)}} \prod_{j\neq 1} \frac{\Pr(Y_j^{(u)} = y_j^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_j^{(u)} = y_j^{(u)}, Q)} \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)})$$

$$= \left(\frac{\Pr(Y_1^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_1^{(u)}, Q)} \Pr(Y_1^{(u)}|X^{(u)} = x^{(u)}) + \frac{\Pr(\neg Y_1^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(\neg Y_1^{(u)}, Q)} \Pr(\neg Y_1^{(u)}|X^{(u)} = x^{(u)}) \right)$$
$$\sum_{y^{(u)}:y_1^{(u)}} \prod_{j\neq 1} \frac{\Pr(Y_j^{(u)} = y_j^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_j^{(u)} = y_j^{(u)}, Q)} \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)})$$

$$\cdots$$

$$= \prod_j \sum_{y_j^{(u)}} \frac{\Pr(Y_j^{(u)} = y_j^{(u)}|D_{k(j)}^{(u)}, Q)}{\Pr(Y_j^{(u)} = y^{(u)}, Q)} \Pr(Y_j^{(u)} = y_j^{(u)}|X^{(u)} = x^{(u)})$$

where the likelihood constant does not depend on X, α, β, γ. This final transformation uses the conditional independence assumption to convert the sum over all peptide sets into a product over peptide indices. Thus the likelihood can be computed for each protein set without computing the sum over all peptide sets.

$$\Pr(Y_j^{(u)}) = \sum_{x^{(u)'}} \Pr(Y_j^{(u)}|X^{(u)} = x^{(u)'}) \Pr(X^{(u)} = x^{(u)'})$$

$$\begin{aligned}
\Pr(Y_j^{(u)}|X^{(u)}=x^{(u)}) &= 1-\Pr(\neg H_j, \bigcap_{i:x_i^{(u)}} \Pr(\neg\Delta_{i,j})) \\
&= 1-\Pr(\neg H_j|\bigcap_{i:x_i^{(u)}} \Pr(\neg\Delta_{i,j}))\Pr(\bigcap_{i:x_i^{(u)}} \Pr(\neg\Delta_{i,j})) \\
&= 1-(1-\beta)\prod_{i:x_i^{(u)},m_{i,j}} (1-\alpha) \\
&= 1-(1-\beta)(1-\alpha)^{|\{x_i^{(u)}:m_{i,j}\}|} \\
&= \Pr(Y_j^{(u)}|\ \ |\{X_i^{(u)}:m_{i,j}\}|=|\{x_i^{(u)}:m_{i,j}\}|)
\end{aligned}$$

$$\Pr(Y_j) = \sum_{k=0}^{size(\{X^{(u)}:m_{i,j}\})} \Pr(Y_j^{(u)}|\ \ |\{X_i^{(u)}:m_{i,j}\}|=k)\Pr(|\{X_i^{(u)}:m_{i,j}\}|=k) \tag{3.1}$$

$$\Pr(Y_j^{(u)}|\ \ |\{X_i^{(u)}:m_{i,j}\}|=k) = 1-(1-\beta)(1-\alpha)^k \tag{3.2}$$

$$\Pr(|\{X_i^{(u)}:m_{i,j}\}|=k) = \binom{size(|\{X^{(u)}:m_{i,j}\}|)}{k}\gamma^k(1-\gamma)^{size(|\{X^{(u)}:m_{i,j}\}|)-k} \tag{3.3}$$

Let $c^{(u)}$ be a collection of sets of proteins from partition i that have idential connectivity in the graph. Let $m_{\nu,j}$ indicate $m_{i,j}$ for all $i \in c_\nu^{(u)}$. Also, use $\nu(i)$ as shorthand for the index of the collection containing protein i.

$$\Pr(Y_j^{(u)}|\ \ |\{X_i^{(u)}:m_{i,j}\}|=k) = \Pr(Y_j^{(u)}|\ \ \sum_{\nu:m_{\nu,j}} |\{X_i^{(u)}:i\in m_{\nu,j}\}|=k)$$

Define a new random variable N as the sum of the present proteins in each collection, and write the previous probability in terms of this variable:

$$\begin{aligned}
N_\nu^{(u)} &= |\{X_i^{(u)}:i\in c_\nu^{(u)}\}| \\
\Pr(Y_j^{(u)}|\ \ |\{X_i^{(u)}:m_{i,j}\}|=k) &= \Pr(Y_j^{(u)}|\ \ \sum_{\nu:m_{\nu,j}} N_\nu^{(u)}=k)
\end{aligned}$$

$$\Pr(Y_j^{(u)}|N^{(u)}=n^{(u)}) = Pr(Y_j^{(u)}|\ \ |\{X_i^{(u)}:m_{i,j}\}|=\sum_{\nu:m_{\nu,j}} N_\nu^{(u)}) \tag{3.4}$$

Summing over all sets of proteins $X^{(u)}$ is now equivalent to summing over the indistinguishable possible values of $N^{(u)}$ and multiplying each term by the number of possible ways it could be made using protein sets.

$$\begin{aligned}\sum_{x^{(u)}} \Pr(D|X^{(u)}=x^{(u)}) \Pr(X^{(u)}=x^{(u)}) &= \sum_{n^{(u)}} \Pr(D|N^{(u)}=n^{(u)}) \Pr(N^{(u)}=n^{(u)}) \\ \sum_{x^{(u)}} L(X^{(u)}=x^{(u)}|D) \Pr(X^{(u)}=x^{(u)}) &= \sum_{n^{(u)}} L(N^{(u)}=n^{(u)}|D) \Pr(N^{(u)}=n^{(u)})\end{aligned}$$

$$L(N^{(u)}=n^{(u)}|D) = \prod_j \sum_{y_j^{(u)}} \frac{\Pr(Y_j^{(u)}=y_j^{(u)}|D_{k(j)})}{\Pr(Y_j^{(u)}=y_j^{(u)}|Q)} \Pr(Y_j^{(u)}=y_j^{(u)}|N^{(u)}=n^{(u)}) \tag{3.5}$$

$$\Pr(N^{(u)}=n^{(u)}) = \prod_\nu \Pr(N_\nu^{(u)}=n_\nu^{(u)}) \tag{3.6}$$

$$\Pr(N_\nu^{(u)}=n_\nu^{(u)}) = \binom{|c_\nu^{(u)}|}{n_\nu^{(u)}} \gamma^{n_\nu^{(u)}} (1-\gamma)^{|c_\nu^{(u)}|-n_\nu^{(u)}} \tag{3.7}$$

$$\begin{aligned}\Pr(X_i^{(u)}|D) &= \sum_{x^{(u)}:x_i^{(u)}} \Pr(X^{(u)}=x^{(u)}|D) \\ &= \sum_{n^{(u)}:n_{\nu(i)}^{(u)}>0} \Pr(N^{(u)}=n^{(u)}, X_i^{(u)}|D) \\ \Pr(N^{(u)}=n^{(u)}, X_i^{(u)}|D) &= \frac{\Pr(D|N^{(u)}=n^{(u)}, X_i^{(u)}) \Pr(N^{(u)}=n^{(u)}, X_i^{(u)})}{\Pr(D)} \\ &= \frac{\Pr(D|N^{(u)}=n^{(u)}) \Pr(N^{(u)}=n^{(u)}, X_i^{(u)})}{\Pr(D)} \\ &= \frac{\Pr(D|N^{(u)}=n^{(u)}) \Pr(X_i^{(u)}|N^{(u)}=n^{(u)}) \Pr(N^{(u)}=n^{(u)})}{\Pr(D)} \\ &= \Pr(N^{(u)}=n^{(u)}|D) \Pr(X_i^{(u)}|N^{(u)}=n^{(u)}) \\ \Pr(N^{(u)}=n^{(u)}|D) &= \frac{\Pr(D|N^{(u)}=n^{(u)}) \Pr(N^{(u)}=n^{(u)})}{\sum_{n^{(u)\prime}} \Pr(D|N^{(u)}=n^{(u)\prime}) \Pr(N^{(u)}=n^{(u)\prime})}\end{aligned}$$

$$\Pr(N^{(u)} = n^{(u)}|D) = \frac{L(N^{(u)} = n^{(u)}|D)\Pr(N^{(u)} = n^{(u)})}{\sum_{n^{(u)\prime}} L(N^{(u)} = n^{(u)\prime}|D)\Pr(N^{(u)} = n^{(u)\prime})} \tag{3.8}$$

The vector of these posterior probabilities can be defined to further save computation by computing all of them in one pass over the set $\{\forall n^{(u)\prime}\}$:

$$\Pr(X^{(u)}|D) = \sum_{n^{(u)}} \Pr(N^{(u)} = n^{(u)}|D)\Pr(X^{(u)}|N^{(u)} = n^{(u)}) \tag{3.9}$$

$$\tag{3.10}$$

Where $\Pr(X^{(u)}|N^{(u)} = n^{(u)})$ is a vector defined as follows:

$$\Pr(X^{(u)}|N^{(u)} = n^{(u)})_i = \frac{n^{(u)}}{|c^{(u)}_{\nu(i)}|} \tag{3.11}$$

$$\tag{3.12}$$

3.3.3 Method given unknown α, β, γ

Likelihood estimates of the model parameters α, β, γ attained by marginalizing out X and Y to compute $\Pr(D|\alpha, \beta, \gamma)$, as well as marginalized estimates using a uniform prior on α, β, γ, are substantially biased, and do not yield good performance in practice. For that reason, parameters are chosen empirically by simultaneously optimizing target-decoy discrimination and FDR calibration.

The grid search performed need not cover the entire range $[0, 1]$; for instance, α values approaching 1.0 do not accurately model the mass spectrometry process, in which many present peptides are frequently unobserved. High-resolution grid search is likewise unecessary; near 0, higher resolution is more important. Empirically, a rough three-dimensional grid search in the range $[0.01, 0.76]$ at resolution of 0.05 for α, in the range $[0.00, 0.80]$ at resolution 0.05 for β, and in the range $[0.1, 0.9]$ at resolution 0.1 for γ yields similarly low optimal α and β values; therefore, even this low-resolution range is excessive.

Increasing the sparsity of the graph (pruning)

In practice, the number of proteins in each partition is often small enough that the algorithm is efficient. However, this is sometimes not the case. A single protein partition can be split by assuming that the peptides that join two subpartitions, $Y^{(0)}$, are not present. This assumption introduces no error when these peptides receive a zero score from PeptideProphet.

Note that normally j iterates over every index where it is used, however some statements here need to limit to $Y^{(0)}$, and so are explicitly referred to as $j^{(0)}$.

First it is demonstrated that the error will be zero in this case:

$$\begin{aligned}
|\Pr(X_i|D) - \Pr(X_i, Y^{(0)} = \{\}|D)| &= |\sum_{y^{(0)}} \Pr(X_i, Y^{(0)} = y^{(0)}|D) - \Pr(X_i, Y^{(0)} = \{\}|D)| \\
&= \sum_{y^{(0)} \neq \{\}} \Pr(X_i, Y^{(0)} = y^{(0)}|D) \\
&\leq \sum_{y^{(0)} \neq \{\}} \Pr(Y^{(0)} = y^{(0)}|D) \\
&\leq \sum_{j} \Pr(Y_j^{(0)}|D)
\end{aligned}$$

Because, writing $Y^{(0)'}$ as the elements other than $Y_1^{(0)}$,

$$\sum_{y^{(0)} \neq \{\}} \Pr(Y^{(0)} = y^{(0)}|D) = \sum_{y^{(0)'} \neq \{\} \wedge (\neg y_1^{(0)} \vee y_1^{(0)})} \Pr(Y^{(0)'} = y^{(0)'}|D)$$

$$= \Pr(Y_1^{(0)}|D) + \sum_{y^{(0)'} \neq \{\}} \Pr(Y^{(0)'} = y^{(0)'}, \neg Y_1^{(0)}|D)$$

$$\leq \Pr(Y_1^{(0)}|D) + \sum_{y^{(0)'} \neq \{\}} \Pr(Y^{(0)'} = y^{(0)'}|D)$$

and continuing inductively, then

$$\leq \cdots \leq \sum_j \Pr(Y_j^{(0)}|D)$$

Since $\Pr(Y_j^{(0)}|D) = 0$ when $\Pr(Y_j^{(0)}|D_{k(j)}, Q) = 0$ (as long as $\Pr(D) > 0$ which can be shown since $\forall r,\ \Pr(Y = y|X = x) > 0$ as long as $\alpha, \beta \in (0, 1)$, meaning that $\Pr(Y = y) > 0$, and $\Pr(D|Y = y) > 0$ trivially for some $Y = y$, and so $\Pr(D) > 0$) then any peptide given a zero score by PeptideProphet can be assumed to be absent. When a peptide is given a very small score by PeptideProphet, then the problem can be approximated more efficiently by assuming the score was zero. In the future, it would be useful to develop a bound on the error introduced by making this approximation, since it would suggest a strategy for which peptides should be changed to zero scored.

Second, it is demonstrated that assuming $Y^{(0)} = \{\}$ allows the proteins to be divided into two partitions, $X^{(1)}$ and $X^{(2)}$ with peptides $Y^{(1)}$ that associate only with $X^{(1)}$, $Y^{(2)}$ that associate only with $X^{(2)}$, and $Y^{(0)}$ that associate with both $X^{(1)}$ and $X^{(2)}$, then:

$$\begin{aligned}\Pr(D, Y^{(0)} = \{\}|X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}) = \\ \sum_{y^{(1)}, y^{(2)}} \Pr(D^{(1)}|Y^{(1)} = y^{(1)}) \Pr(D^{(2)}|Y^{(2)} = y^{(2)}) \Pr(D^{(0)}|Y^{(0)} = \{\}) \\ \Pr(Y^{(1)} = y^{(1)}|X^{(1)} = x^{(1)}) \Pr(Y^{(2)} = y^{(2)}|X^{(2)} = x^{(2)}) \\ \Pr(Y^{(0)} = \{\}|X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)})\end{aligned}$$

$$\begin{aligned}\Pr(Y^{(0)} = \{\}|X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}) = \\ \frac{\Pr(Y^{(0)} = \{\}|X^{(1)} = x^{(1)}, X^{(2)} = \{\}) \Pr(Y^{(0)} = \{\}|X^{(2)} = x^{(2)}, X^{(1)} = \{\})}{\prod_{j^{(0)}} \Pr(\neg H_{j^{(0)}} | \bigcap_{i:x_i} \neg \Delta_{i,j^{(0)}})}\end{aligned}$$

$$\Pr(D, Y^{(0)} = \{\} | X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}) =$$

$$\frac{1}{\prod_{j^{(0)}} \Pr(\neg H_{j^{(0)}} | \bigcap_{i:x_i} \neg \Delta_{i,j^{(0)}})}$$

$$\sum_{y^{(1)}} \Pr(D^{(1)} | Y^{(1)} = y^{(1)}) \Pr(D^{(0)} | Y^{(0)} = \{\})$$

$$\Pr(Y^{(2)} = y^{(2)} | X^{(2)} = x^{(2)}) \Pr(Y^{(0)} = \{\} | X^{(2)} = x^{(2)}, X^{(2)} = \{\})$$

$$\sum_{y^{(2)}} \Pr(D^{(2)} | Y^{(2)} = y^{(2)})$$

$$\Pr(Y^{(2)} = y^{(2)} | X^{(2)} = x^{(2)}) \Pr(Y^{(0)} = \{\} | X^{(2)} = x^{(2)}, X^{(1)} = \{\})$$

$$=$$

$$\frac{1}{\Pr(D^{(0)} | Y^{(0)} = \{\}) \prod_{j^{(0)}} \Pr(\neg H_{j^{(0)}} | \bigcap_{i:x_i} \neg \Delta_{i,j^{(0)}})}$$

$$\sum_{y^{(1)}} \Pr(D^{(1)} | Y^{(1)} = y^{(1)}) \Pr(D^{(0)} | Y^{(0)} = \{\})$$

$$\Pr(Y^{(2)} = y^{(2)} | X^{(2)} = x^{(2)}) \Pr(Y^{(0)} = \{\} | X^{(2)} = x^{(2)}, X^{(2)} = \{\})$$

$$\sum_{y^{(2)}} \Pr(D^{(2)} | Y^{(2)} = y^{(2)}) \Pr(D^{(0)} | Y^{(0)} = \{\})$$

$$\Pr(Y^{(2)} = y^{(2)} | X^{(2)} = x^{(2)}) \Pr(Y^{(0)} = \{\} | X^{(2)} = x^{(2)}, X^{(1)} = \{\})$$

$$=$$

$$\frac{1}{\Pr(D^{(0)} | Y^{(0)} = \{\}) \prod_{j^{(0)}} \Pr(\neg H_{j^{(0)}} | \bigcap_{i:x_i} \neg \Delta_{i,j^{(0)}})}$$

$$\Pr(D^{(1)}, D^{(0)}, Y^{(0)} = \{\} | X^{(1)} = x^{(1)}, X^{(2)} = \{\})$$

$$\Pr(D^{(2)}, D^{(0)}, Y^{(0)} = \{\} | X^{(2)} = x^{(2)}, X^{(1)} = \{\})$$

Finally, the likelihood can be defined:

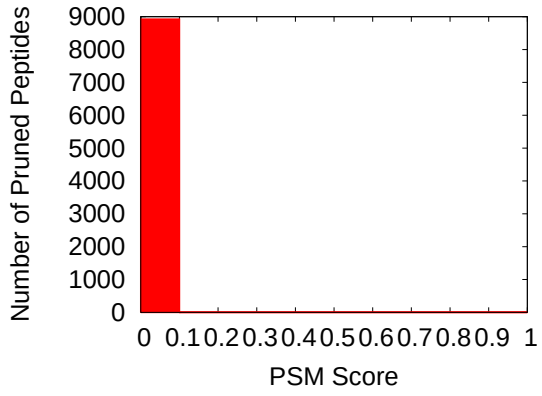
$$L(X^{(1)}, X^{(2)}|Y^{(0)} = \{\}, D) = \frac{1}{\prod_{j^{(0)}} \Pr(\neg H_{j^{(0)}} | \bigcap_{i:x_i} \neg \Delta_{i,j^{(0)}})} L(X^{(1)} = x^{(1)}, X^{(2)} = \{\}|Y^{(0)} = \{\}, D^{(1)}, D^{(0)}) L(X^{(2)} = x^{(2)}, X^{(1)} = \{\}|Y^{(0)} = \{\}, D^{(2)}, D^{(0)}) \tag{3.13}$$

where the likelihood constant does not depend on X, α, β, γ.

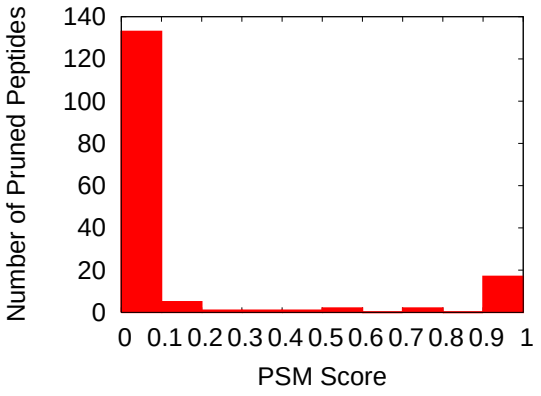
The resulting probability and likelihood is equivalent (aside from the leading correction in the previous formula) to those computed after disconnecting $X^{(1)}$ from $X^{(2)}$ by duplicating the peptides that they share so that each $X^{(1)}$ and $X^{(2)}$ associate with their own copy of $Y^{(0)}$ and $D^{(0)}$.

3.4 Analysis of approximation errors from pruning

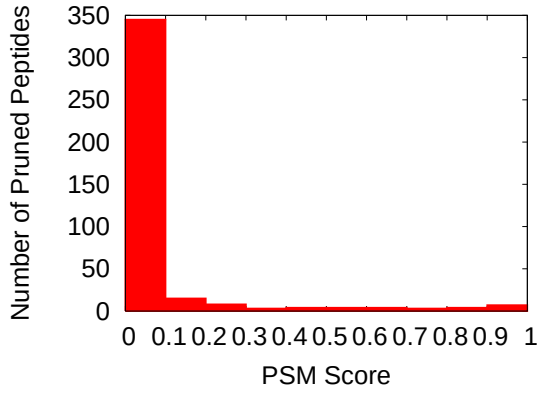
The calculation of posterior probabilities is exact when only zero-scoring PSMs are used for pruning. When PSMs with small nonzero scores are used, then a small approximation error is introduced. Occassionaly, it may be necessary to prune a PSM with a large score in order to achieve the desired separability. Even in this case, the error introduced may be small and is confined to the proteins in the same connected subgraph; therefore, the overall approximation error depends not only on the highest-scoring PSM pruned, but on the quantity of PSMs pruned and their scores. Figure 3.1 plots the distribution of pruned PSM scores necessary to require no more than 2^{18} marginalization steps for any subgraph. It is demonstrated that there are not many high-scoring PSMs that need to be pruned, even to achieve this strict level of separability. The yeast data requires a couple of high-scoring PSMs to be pruned, but the proteins associated with these PSMs are also suppported by other PSMs; therefore, the error is still minimal and only influences these few proteins.



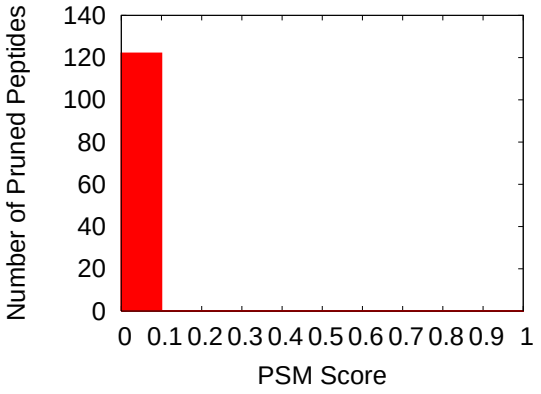
(A) *H. influenzae*



(B) Yeast



(A) ISB 18



(B) Sigma 49

Figure 3.1: **Distribution of pruned PSM scores.** For each data set, a histogram of the scores of pruned PSMs is plotted. The figure shows that almost all pruned PSMs have very low scores. The *C. elegans* data set is not shown because it does not require pruning to achieve this much separability.

Chapter 4

EXTENDING THE FIDO METHOD WITH FORMAL GRAPH-THEORETIC PROCEDURES

4.1 Methods for graphical inference

The efficiency with which inference can be performed is extremely important: the computational cost of inference has impeded the use of more complex and realistic models of the tandem mass spectrometry process. In this chapter, the efficiency and accuracy with which various inference methods estimate posterior probabilities is evaluated by using the Fido model. Specifically, the naive marginalization approach used by Fido after graph transformation is compared to three alternative approaches: two MCMC methods, and junction tree marginalization. Both MCMC methods attain successively more accurate estimates of posteriors by sampling from possible outcomes of unkown random variables. Specifically, two Gibbs samplers are used; both store a current configuration of unobserved random variables and propose and sample from changes to this configuration. The first MCMC approach samples the states of both proteins and peptides using a Gibbs sampler similar to MSBayes. The second Gibbs sampler approach, called a "collapsed Gibbs sampler," samples only the states of proteins, and then exploits conditional independence to efficiently marginalize out the peptides given that protein configuration.

The third method considered is "junction tree inference." A junction tree is a tree where each tree node is a set of vertices from the original graph and where any vertex found in two tree nodes must be found on the path between them [26]. Some sparsely connected graphs can be decomposed into junction trees that can be marginalized significantly faster than using naive marginalization. The speedup introduced depends on the connectivity of the graph; some graphs produce junction trees that can be marginalized very efficiently, while other graphs produce junction trees that are no more efficient to marginalize than with the naive approach. In this chapter, the practical benefit of junction tree inference is investigated for real protein-to-peptide graphs using a from-scratch implementation of tree decomposition and the Hugin algorithm [1].

In this chapter, it is shown that the cost of protein inference is dominated by a few highly connected subgraphs of proteins. For many real protein inference graphs, junction tree inference can compute posteriors much more efficiently than naive marginalization. Likewise, posterior estimates from a collapsed Gibbs sampler converge much more quickly than using the standard Gibbs sampler. Both junction tree inference and collapsed Gibbs sampling can compute high-quality posteriors in a fraction of the time required by naive marginalization and the Gibbs sampling approaches.

4.2 Data sets

For each inference method, the connectivity of the protein-to-peptide graphs and the coverage for the experiment determines the difficulty of protein inference. The complexity of computing protein posteriors was analyzed using the three model organism protein lysate data sets used to evaluate Fido: *H. influenzae*, *S. cerevisiae*, and *C. elegans*. For each data set, the spectra were searched against a database comprised of the target organism's proteome and the same set of decoy proteins used to evaluate Fido. The net connectivity of these target and decoy databases, as well as the coverage of the experiment, determine the complexity of protein inference for each data set. For all data sets, proteins adjacent to identical peptide sets were grouped to ensure the complexity results from distinct, biologically interesting proteins, that are a challenge to resolve during protein inference. Even if these proteins were not grouped, the clustering graph transformation would ensure that identically connected proteins do not substantially contribute to the cost of inference.

4.3 Results

4.3.1 Alternative graphical approaches

Posterior probabilities for the Fido model can be estimated using alternative approaches to naive marginalization, whereby all possible protein (or protein cluster) states are enumerated for each connected subgraph after graph transformations. A popular approach to approximating posteriors, MCMC, does so using a random walk through the space of unobserved random variables. This random walk is chosen so that its corresponding Markov chain has a stationary

distribution equal to the desired posteriors. Gibbs sampling is a specific random walk procedure, which starts with some initial state for the unobserved random variables and proposes several random state changes. A value proportional to the probability of each proposed configuration is computed, and a new configuration is selected based on these relative proportional probabilities.

The Gibbs sampler randomly samples from configurations defined by states for all peptides and proteins. Using a "blocking" approach similar to MSBayes, all possible changes are proposed jointly for a random block of peptides and a random block of proteins. Block sizes of three were used for both the peptides and proteins, because that was found to yield best performance for MSBayes.

Because the Fido model uses independent protein priors and treats peptides as conditionally independent given the protein set, it is possible to efficiently marginalize over all peptide configurations when the protein configuration is known. In the same manner as the marginalization strategy used by Fido, this conditional independence can be exploited when given a protein configuration to marginalize out all sets of peptides in linear time. This idea was used to create a "collapsed Gibbs sampler," which samples some unobserved random variables and marginalizes out the remaining variables. In the collapsed Gibbs sampler, only protein configurations were sampled and marginalized out the peptide configuration given the sampled protein configuration. The same blocking strategy was used for these protein blocks, sampling all possible perturbations to a random block of size three.

In addition to sampling approaches, junction tree inference, a more sophisticated approach to exact marginalization, was also considered. Naive inference computes posterior probabilities by "marginalizing" or summing over all protein configurations in a connected subgraph; the number of configurations, and thus the computational cost of naive marginalization, is exponential in the number of proteins in the subgraph. However, it is possible to marginalize some subgraphs more efficiently. For example, the graph in Figure 4.1A can be thought of as two subgraphs that are "d-separated" by protein X_3; given a known boolean value for X_3, the graph separates into two disjoint subgraphs. Removing X_3 and its edges leaves two graphs, each containing two proteins; therefore, the entire graph can be marginalized in $2 \times \left(2^2 + 2^2\right) = 2^4$ protein configurations rather than the 2^5 protein configurations required by naive marginalization.

In general, a protein set that results in a useful d-separation in the graph should be marginal-

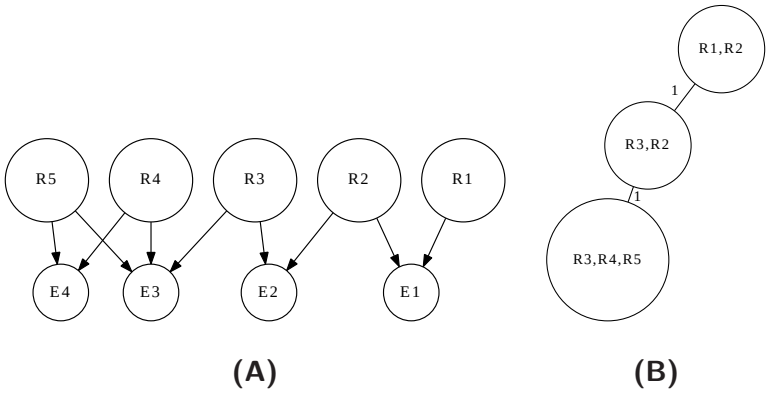


Figure 4.1: **Decomposing a graph for protein inference. (A)** An example DAG for a five protein inference problem. Proteins, labeled with prefix X, are shown as large circles and peptides, labeled with prefix Y, are shown as smaller circles. Choosing a specific state for protein X_3 effectively removes X_3 and its edges from the graph and propagates these values towards its sucessors. This results in two subgraphs that can be marginalized independently. **(B)** The protein subgraph of the junction tree depicts dependencies between proteins.

ized out late so that it has a specific value when other variables are marginalized out. Choosing a collection of nodes that d-separates the graph corresponds to placing those nodes late in the "elimination ordering." The most efficient elimination ordering for marginalizing all variables corresponds to the tree decomposition (or junction tree) with minimum treewidth. Decomposing a graph into a tree of minimum treewidth is NP-hard [2]; however, in practice greedy heuristics can be used to create trees with low treewidth. Figure 4.1B shows that this protein inference problem can be decomposed into a tree of four nodes with treewidth two. For this reason, marginalization can be performed by visiting $2^3 + 2^2 + 2^2 = 2^4$ protein configurations. Effectively, the junction tree shown makes use of the fact that X_3 d-separates (X_1, X_2) from (X_4, X_5).

Tree decomposition was performed using a minimum neighbor heuristic (nodes were eliminated in ascending order of the number of remaining neighbors in the graph). Inference was performed using an implementation of the Hugin algorithm, which efficiently computes the marginal posterior probabilities using the junction tree.

4.3.2 The cost of naive marginalization in real protein inference graphs

In order to investigate the difficulty of marginalization on real data sets, each data set was divided into connected subgraphs and the number of proteins in each connected subgraph was counted. In the first row of Figure 4.2, the red solid series shows the distribution of naive marginalization costs for the unpruned graphs from each data set; the x-axis of each figure is the value of the log marginalization cost and the y-axis is the number of connected subgraphs with log marginalization cost at or exceeding that value. The *H. influenzae* data set contains roughly 10 connected subgraphs with marginalization cost at or exceeding 2^{25} and maximum marginalization cost 2^{120}; naive marginalization on this data set is far too computationally expensive to be performed. The unpruned *S. cerevisiae* data has only two subgraphs with more than 10 proteins: one contains 22 proteins and the other contains 54. While enumerating 2^{22} protein configurations is on the high side of what can be performed efficiently, 2^{54} configurations is too high for naive marginalization to be practially useful. The unpruned *C. elegans* data contains no connected subgraphs with more than six proteins; therefore, naive marginalization is quite practical. It is somewhat surprising that naive marginalization is feasible on the data from the most complex organism and with the highest coverage; however, closer examination reveals that some highly connected proteins can be successfully grouped together. These proteins are not necessarily identical; they simply contain the same set of observed peptides. Without grouping these proteins, the cost of naive marginalization for the *C. elegans* data would be much higher.

Proteins connected solely by PSMs with scores very close to zero can be separated without introducing any error. A PSM with a score of zero requires it was neither created by a protein or by the noise model. These same necessary events correspond to a graph in which the zero-scoring PSM is copied so that each adjacent protein has its own copy; therefore, by simply adjusting for the number of copies made, it is possible to transform one graph into subgraphs with fewer proteins without any error using the pruning transformation. Because pruning can sometimes reduce the cost of marginalization without introducing any error, the cost of naive marginalization is considered after pruning PSMs with very low PeptideProphet scores. The second row of Figure 4.2 shows the cost of naive marginalization after pruning PSMs with

scores that are practically zero (i.e. less than 0.001). After pruning these low scoring PSMs, the *H. influenzae* data contains no connected subgraphs more than six proteins, making naive marginalization trivial. The *H. influenzae* data was searched against a target-decoy database comprising both the *H. influenzae* proteome and the human proteome. Generally, peptides associated with human proteins should receive very low scores; therefore, pruning is very effective at reducing the cost of marginalization. In contrast, pruning is of little use on the *S. cerevisiae* data because larger subgraphs are connected by high-scoring peptides found in yeast proteins. Because *C. elegans* was already quite easy, there is essentially no benefit from pruning.

For each experiment, the graph containing only the peptides matched to observed spectra and only the proteins adjacent to those peptides is a subgraph of the entire bipartite graph for the entire search database; all proteins, peptides, and edges in the graph from an experiment must also be found in the full graph produced by the search database. For this reason, the complexity of the graph from the search database provides an upper bound on the complexity of protein inference for any data set searched against that proteome. The cost of naive marginalization is therefore analyzed on graphs produced by digesting each target proteome *in silico* and including all fully tryptic peptides with lengths in range $[6, 50]$ residues and masses in range $[200, 7200]$ daltons. The third row of Figure 4.2 shows the cost of naive marginalization for the entire proteome of each of these species. As expected, naive marginalization on the *H. influenzae* data is trivial, corroborating the prediction that the difficult subgraphs are the result of the human decoy database. The full *S. cerevisiae* proteome is more difficult than the observed data, but the distribution has a similar shape. On the other hand, the *C. elegans* proteome is far more difficult. Despite the fairly good coverage of the data set, the observed data produces a graph that is only a fraction of the graph produced by digesting the full proteome. An experiment that covered a larger portion of the *C. elegans* proteome would not only produce subgraphs that were more highly connected, it would also increase the chances of connecting a protein with a peptide that it does not share with other proteins; such a peptide would prevent a protein from being grouped, which may substantially increase the cost of inference.

4.3.3 The value of tree decomposition

For each connected subgraph, a corresponding junction tree was created; this tree was used to compute the number of protein configurations required for marginalization using junction tree inference. A disparity between the cost of naive marginalization and the cost of junction tree indicates subgraphs with tree decompositions that can be marginalized more efficiently than with the naive approach. Figure 4.2 overlays the distributions of cost for junction tree inference using a blue dashed series.

The first row of Figure 4.2 shows the cost of inference on the unpruned graphs. On the *H. influenzae* data, the cost of junction tree inference is substantially lower than the cost of naive marginalization: there are only two subgraphs requiring more than 2^{25} steps with junction tree inference, whereas there are around 10 subgraphs requiring more than 2^{25} steps using naive inference. Furthermore, the most difficult subgraph for junction tree inference requires just over 2^{40} steps; in comparison, the most difficult subgraph for naive marginalization requires more than 2^{120} steps. The *S. cerevisiae* data exhibits a more modest improvement from junction tree inference, but the cost of the most expensive subgraph drops from 2^{54} to 2^{40} when using junction tree inference rather than naive marginalization. The improvement to the *C. elegans* data set is negligable, largely because the most computationally expensive subgraph for naive marginalization requires a modest 2^6 steps.

The second row of Figure 4.2 shows the cost of junction tree inference after pruning the data. There is no improvement over naive marginalization for the pruned *H. influenzae* data set, which is quite easy when using naive marginalization. As noted above, the large subgraphs in the *H. influenzae* data set are the result of matches to the human decoy database; spectra that match peptides adjacent only to decoy proteins generally have very low scores and were successfully pruned, thus reducing the number of proteins in connected subgraphs. These subgraphs are so small that junction tree inference offers no benefit. Junction tree inference improves the pruned *S. cerevisiae* data about as well as it improves the unpruned *S. cerevisiae* data. The larger connected subgraphs on this data set have high-scoring PSMs that come from target proteins; therefore, pruning barely changes the *S. cerevisiae* graph. Likewise, pruning has nearly no influence on the *C. elegans* data set, and so the improvement of junction tree inference is

negligable, just as it was for the unpruned data.

The third row of Figure 4.2 shows the cost of junction tree inference on the graph produced from the entire proteome of the target organism. On the *H. influenzae* data, junction tree inference provides no advantage over naive marginalization, which is already quite easy. The distribution of costs is nearly identical to that of the pruned *H. influenzae* data, confirming that the complexity of inference for the unpruned *H. influenzae* data was the result of the human decoy database. The *S. cerevisiae* data set is more interesting, and the improvement of junction tree inference over naive marginalization is noteworthy: in one case, junction tree inference improves upon naive marginalization by reducing the number of steps necessary from 2^{72} to just under 2^{50}. The full *C. elegans* proteome data has the most dramatic improvement from junction tree inference. The largest connected subgraph contains over 2000 proteins, but it corresponds to a junction tree that can be solved in 2^{89} steps. Furthermore, junction tree inference requires no more than 2^{19} steps for all remaining subgraphs; aside from the largest subgraph, there are 24 other subgraphs that demand more than 2^{19} steps using naive marginalization, including subgraphs requiring 2^{89}, 2^{59}, and 2^{50} steps.

4.3.4 Convergence of inference methods

A moderately large subgraph of 22 proteins from the *S. cerevisiae* data set was used to compare empirically the various inference algorithms. This subgraph is particularly useful because it is non-trivial for naive marginalization and it can't be substantially separated without pruning high-scoring PSMs, but exact inference is still feasible; therefore, the exact posteriors can be used as a gold standard to evaluate and compare the other methods. Figure 4.3A, shows the directed protein-to-PSM graph. Figure 4.3B shows the junction tree produced by that graph. Each node in the junction tree is labeled with the number of proteins found in the node; the number of variables in each node determines the cost of junction tree inference. The largest node contains 14 proteins, and the total cost of enumerating all protein states for each junction tree node is $2^{14.644}$. Naive marginalization requires 2^{22} steps.

Figure 4.4 shows the tradeoff between error and efficiency when analyzing this subgraph using the four inference algorithms. All methods used the optimal α, β, and γ parameters

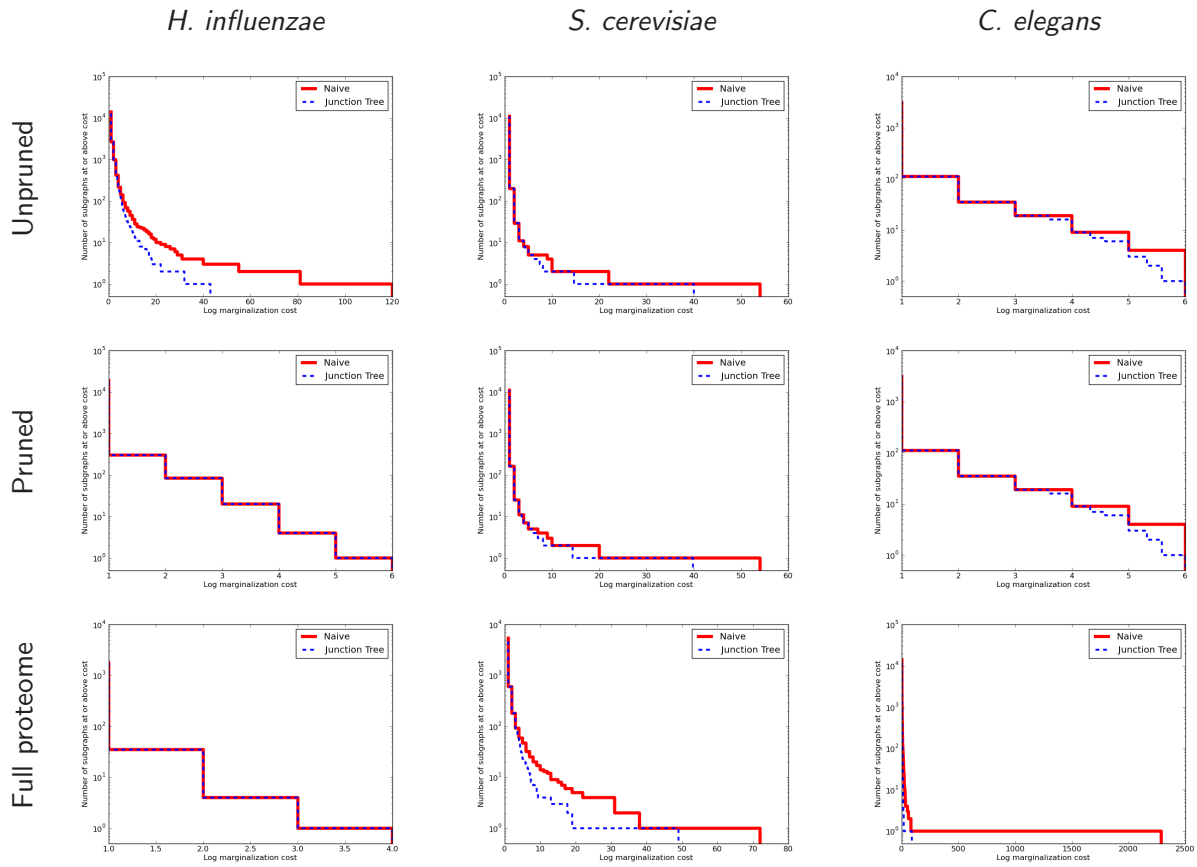


Figure 4.2: **Distribution of marginalization complexities.** For each organism, the following are plotted: the distribution of marginalization costs for the connected subgraphs found in the unpruned graph, the pruned graph, and the graph from the full target proteome. The x-axis measures the log marginalization cost of a subgraph, and the y-axis shows the number of subgraphs with marginalization cost at or exceeding that cost. The distribution of costs using naive marginalization is shown using the red solid series and the distribution of costs using junction tree inference is shown using the blue dashed series.

previously found to simultaneously yield the highest discrimination and calibration. The x-axis plots the execution time and the y-axis plots the largest absolute error introduced compared to doing exact marginalization. All methods were implemented using a shared python framework. All times are reported in user time. It should be noted that the relative times of the python implementations are useful for comparison, but the actual times presented are not representative of a fast implementation in a compiled language; for instance, the C++ version available in [28] is substantially faster than the corresponding python implementation. Using the largest absolute posterior error was motivated by the fact that a single significant perturbation to a protein's posterior estimate disrupts the overall ranking of the proteins even if the average error of each posterior is small. The Gibbs sampler and the collapsed Gibbs sampler were each run for successively longer periods while periodically logging the error of their estimates. Both samplers were started at random configurations of variables with nonzero likelihood and given a 100 iteration burn-in. Because the samplers are stochastic, Figure 4.4 shows the average time-error relationship after replicating 10 times for each sampler; these replicate series were all very close to the average. Naive marginalization and junction tree inference were run after successively pruning the graph more and more aggressively. Exact inference on the unpruned graph introduces no error and was used as the baseline. As more PSMs with higher scores are pruned, marginalization becomes more efficient, but the largest posterior error becomes larger.

Not surprisingly, the collapsed Gibbs sampler outperforms the Gibbs sampler, which mixes less efficiently because it samples the peptide configuration as well as protein configuration. Also not surprising is the fact that junction tree inference is superior to naive marginalization, particularly when the graph is unpruned. When the computation time is very small, the samplers perform worse than the marginalization procedures; however, as more time is allowed, the collapsed Gibbs sampler becomes competitive with junction tree inference and offers a superior tradeoff between error and time by avoiding using high-scoring PSMs for pruning. When around 2000 seconds are allowed, junction tree inference obtains nearly the exact answer, which will be approached asymptotically by the samplers.

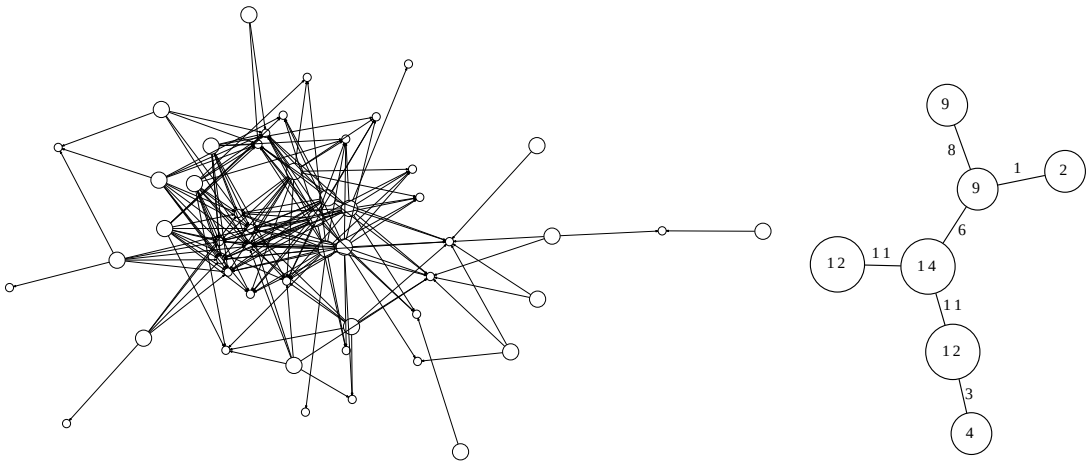


Figure 4.3: **A difficult connected subgraph and its tree decomposition.** Pruning introduces no error when pruned peptides have very small scores; however, some connected subgraphs may not be separable without pruning higher-scoring peptides. **(A)** In a subgraph taken from the *S. cerevisiae* data set, 22 dependent proteins cannot be separated without pruning high-scoring peptides. Many proteins in this subgraph share several peptides; in order to separate groups of proteins, all peptides shared by the groups must be pruned. Proteins are shown as large circles and peptides are shown as smaller circles. **(B)** Each node in the junction tree is labeled by the number of proteins it contains and each edge is labeled by the size of the intersection between its incident nodes. Junction tree inference requires enumerating the power-set for the collection of proteins found in each junction tree node; therefore, inference can be performed on the order of the sum of the cost of computing those power sets. For this connected subgraph, marginalization would require enumerating $2^{14.644}$ protein states, a fraction of the 2^{22} protein states enumerated by naive marginalization.

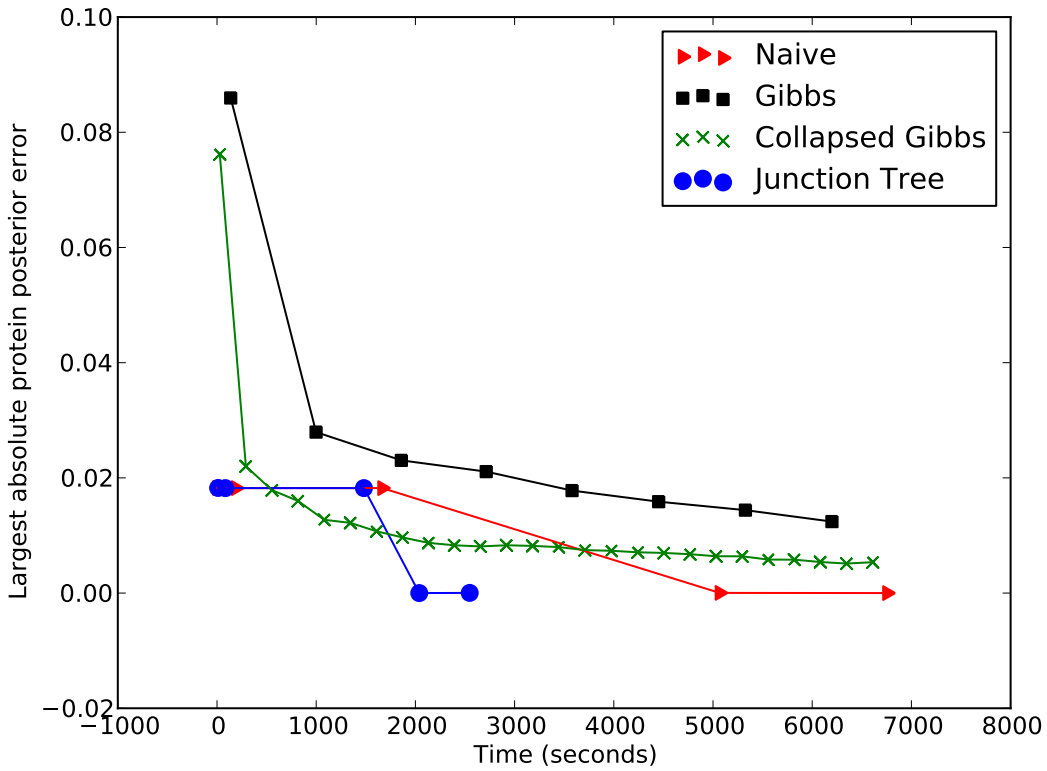


Figure 4.4: **Convergence of different methods.** For each method, the relationship between the largest posterior error and runtime for protein inference on the subgraph from Figure 4.3 is shown. The Gibbs sampler runtimes were varied by allowing more iterations. Junction tree inference and naive marginalization runtimes were varied by successively pruning the graph and allowing PSMs with sequentially higher scores to be pruned. The Gibbs sampler and collapsed Gibbs sampler series are averages of 10 replicate runs, none of which vary substantially from the average shown.

4.4 Discussion of graphical methods for efficient inference

It has been demonstrated that junction tree inference can be substantially more efficient than naive marginalization for protein inference. Some subgraphs, though dramatically more efficient to marginalize using junction tree inference, are still computationally infeasible without pruning. For instance, one connected subgraph in the *S. cerevisiae* data set requires 2^{54} steps using naive inference and 2^{40} steps using junction tree inference; while 2^{40} steps may still be infeasible, the substantial decrease in the cost of inference may be enough to make the problem approachable by other methods. For instance, given the states of all proteins in the largest node in the junction tree of Figure 4.3, the remaining marginalization problem becomes trivial; therefore, rather than perform the full marginalizatin using the junction tree, it may be favorable to use a more sophisticated collapsed Gibbs sampler, which samples the configuration of proteins in a few highly connected junction tree nodes and then performs junction tree inference on the reminaing tree. Such an approach would substantially reduce the dimensionality of the sampling problem. Other hybrids between sampling and marginalization may also prove fruitful. For instance, it is always possible to obtain a fast approximation of posteriors using marginalization and pruning. This fast estimate may be used to initialize the sampling process or could be used in place of a uniform proposal distribution.

Junction tree inference may also be used to generalize the pruning procedure to enable faster and more accurate approximation. Essentially, pruning exploits cases where the joint distribution can be approximted as the product of multiple independent distributions. A general search for more exploitable instances of this phenomenon could be performed by analyzing the messages sent by the message passing algorithm used to perform junction tree inference. In a similar manner, it may be possible in some instances to avoid marginalizing over variable configurations with extremely low likelihoods without noticably altering the final posterior estimates. It would also be interesting to compare these approaches to other iterative approximations like loopy belief propagation [36].

Furthermore, Figure 4.2 shows that the cost of inference is dominated by a few highly connected subgraphs. Even in cases when the posterior estimates of the proteins in these graphs must be estimated with some error, it may be possible to obtain a high-quality estimate of what

proteins are present in a complex mixture. Rather than focusing on computing exact posteriors where it is very difficult, it may be possible to estimate a bound on the posterior estimates and present the ultimate ranked list of proteins using a partial rather than a total ordering based on whether there is a provable hierarchy for a pair of proteins.

Formal graphical methods provide a general framework that can be adapted for new models to see whether they are computationally feasible. General inference methods that work well in practice on real protein inference graphs will enable the development of more complex models that reflect real processes in tandem mass spectrometry. For instance, all existing approaches, to my knowledge, perform protein inference on a bipartite graph, when in actuality the data forms a tripartite graph; including edges between a spectrum and the second and third-best matching peptides may allow spurious high-scoring peptides to be explained away by peptides adjacent to proteins with independent supporting evidence. In a similar manner, modeling competition between peptides in data-dependent acquisition may help explain why peptide detectability varies widely between experiments. Enabling efficient inference on more realistic models of the mass spectrometry process will contribute dramtically to our ability to analyze protein data and improve our understanding of the biological processes inside the cell.

Chapter 5

THE FUTURE OF STATISTICAL ANALYSIS FOR PROTEIN INFERENCE

Quantitative methods, which estimate numeric protein scores instead of simply imperatively computing the protein set x^*, have a distinct advantage over nonquantitative methods, because quantitative methods offer information regarding the confidence of the proteins identified. Furthermore, among quantitative methods, probabilistic methods offer easily interpreted scores; a marginal protein posterior probability indicates the chance that a certain protein is present in the sample. In contrast, with nonprobabilistic quantitative methods, a score threshold of τ_X may be appropriate for one data set, but on another larger data set, the protein threshold should be stricter $\tau'_X > \tau_X$. This property leads to a method that must be run and adjusted manually to achieve satisfactory results.

Probabilistically motivated heuristics, like the iterative methods presented in Chapter 1, often provide very useful and satisfying ways to approach a problem initially, but these method do not provide an intuitive understanding of the inference problem itself; therefore, numeric heuristics can be very difficult to improve and extend. Furthermore, because these methods are defined procedurally, rather than derived from clearly stated assumptions, they can be brittle. For example, ProteinProphet is often extremely accurate and efficient; however, for some large data sets, like the combined replicates from the protein standard of 18 proteins, ProteinProphet becomes inaccurate and unreliable. Post-processing methods like MAYU [25] attempt to compute more rigorous FDR estimates after ProteinProphet is run, but the entire approach of "fixing" the results after the fact, rather than improving the gestalt process of identification, is reminiscient of the very approach that sometimes makes ProteinProphet unreliable.

Several assumptions stated or implicit in these various approaches are quite similar. For instance, the one- and two-peptide rules, DTASelect, ProteinProphet, EBP, Scaffold, PANORAMICS, and the nested mixture model all make assumptions that make the graph into a tree by somehow partitioning peptides among their adjacent proteins. This shared assumption is no

coincidence: inference on a tree is substantially easier than inference on a general graph. Thus, rather than starting with a formal and rigorous formulation of the protein identification problem and then making it more efficient, these methods change the question they ask so that it becomes easier to answer. Likewise, every method presented makes assumptions to remove spectral degeneracy, which results in a subgraph on peptides and spectra composed of disjoint trees.

Rather than making assumptions that make the graph (or a snapshot of the graph in one iteration) into a tree, the hierarchical model, MSBayes, and Fido attempt to solve the actual problem by modeling the mass spectrometry process generatively and then taking steps to compute protein posteriors or the MAP protein set. In the case of the hierarchical statistical model [29], a naive treatment of inference on the non-tree topology of the graph results in a runtime exponential in the number of connected proteins in a subgraph; this runtime makes the procedure of little use in practice. MSBayes, on the other hand, avoids an exponential runtime by using MCMC. Fido uses graph transformations so that the cost of marginalization or approximate marginalization is not prohibitively large.

Because the Bayesian methods approach protein inference by modeling the mass spectrometry process and then taking the necessary steps to make inference feasible, they have a convenient modularity that allows the models to be easily improved; for instance, prior information regarding the number of present proteins or prior information regarding peptide detectability can be easily integrated into these models. Furthermore, if peptide detectability is partly determined by features of each peptide, then the emission model can be relaxed to allow tied parameters $\alpha_j, \alpha_{j'}, \ldots$; these tied parameters will introduce many implicit edges into the graph, increasing the computational cost of inference. Nevertheless, the α values could be fixed by sampling, and marginalization could efficiently be performed for each set of α values. In contrast, iterative methods are much more difficult to improve in this way. For instance, it would be quite difficult to come up with a way to correctly adjust ProteinProphet's protein-peptide weights to account for prior information on the number of present proteins. Bayesian approaches, on the other hand, don't partition the peptides among adjacent proteins; instead, they take into account the fact that shared peptides may only be present in a single protein by "explaining away" the shared peptide when an adjacent protein is present. Because Bayesian methods model mass

spectrometry in an intuitive way, it is trivial to see how these methods could incorporate prior information into the protein prior for a model.

In a similar manner, a noise model is a necessity for inference on large data sets. Intuitively, each PSM has a chance of incorrectly matching a spectrum with a fairly high score. In a very large data set, this chance becomes nontrivial and is responsible for erratic behavior in many protein identification methods. Because iterative methods do not model the mass spectrometry process generatively, it is difficult to conceive of a reliable means of adding a noise model; in contrast, for Bayesian methods this is trivial and only requires considering the impact of that particular noise model on the computational efficiency.

It is easy to see that much more accurate and sophisticated models of the mass spectrometry process are possible. For example, peptide detectability is best modeled in a sample-specific way, not statically. Although some of the factors that determine peptide detectability are invariant between experiments, others will depend on the experiment and on the specific proteins present. For instance, a highly abundant protein can result in highly abundant adjacent peptides, some of which may outcompete other less-abundant present peptides with similar mass and retention time; these less abundant peptides will not be selected during the precursor scan, and will not be detectable. Bayesian methods are very well-suited to incorporate more complex models of peptide detectability, which could easily model experiment-specific peptide detectability trends. Similarly, Bayesian methods can be easily extended to the full tripartite graph. The spectral degeneracy removed by all methods offers a great deal of unused information: for example, a spectrum with a spurious highest scoring peptide match is likely to have a lower ranked peptide match from a present peptide. Using protein-level information, the present peptide may have other evidence, which could upweight the chances it produced the degenerate spectrum, and subsequently downweight the effect of the spurious PSM.

These more complex models of mass spectrometry will introduce greater computational burden; however, that must not dissuade us from modeling the process as accurately as possible. First of all, there is a wealth of information on efficient graphical inference, including approximation techniques such as loopy belief propagation [36]; such techniques could substantially improve the efficiency of inference even with accurate models.

Furthermore, even if the cost of inference using a more accurate model were prohibitive, a

better generative model of the mass spectrometry process could be applied to evaluation. The target-decoy strategy is so flawed that it makes rigorous comparison of methods very difficult. On one hand, degenerate peptides may increase the probability estimates of absent target proteins more than decoy proteins, yielding an incorrectly optimistic estimate of sensitivity and FDR. On the other hand, because high-scoring decoy peptides are going to stratify uniformly over the decoy proteins, whereas high-scoring target peptides are more likely to cluster to fewer present target proteins, target-decoy databases may overestimate the number of absent targets and yield overly pessimistic FDR estimates [25]. Robust generative models could supplement or replace the target-decoy strategy, and be used to evaluate the protein sets proposed by different methods.

Models of the mass spectrometry process and inference techniques are not the only facets of protein inference that could benefit from increased formality and attention. The problems of evaluating methods and reporting results are also ripe for fundamental improvements and novel approaches. For instance, posterior-based rankings on proteins don't account for non-independence between the included proteins: if a higher-scoring protein is chosen, that choice should decrease the rank of another protein that is adjacent to an overlapping set of peptides. Furthermore, the MAP approach does not adequately illustrate the relative advantage of the MAP set compared to the protein set yielding the second highest posterior. Often these poor results lead to biologists interpreting results themselves, even after substantial statistical inference is performed: it is common to ask, "Are there any other sets of proteins that might result in the identified set x^* using this method?" Ideally, rigorous statistical procedures should evaluate methods by mapping all possible sets of present proteins to all possible sets of identified proteins and estimating the probability that a method would make each identification from each present set. I posit that such an approach could successfully avoid the problems of target-decoy competition by directly asking about uncertainty among target proteins. If such an approach could be made computationally efficient, then it could dramatically improve the utility of mass spectrometry and substantially improve the confidence with which protein samples can be analyzed.

Above all, it is important to recognize that a number of very similar approaches to protein identification have been repeatedly applied in different packaging. When a field is in its infancy,

heuristics are the natural approach to take; heuristics are fast sketches of our qualitative goals, and they are often easy to implement. Nevertheless, as a field grows into maturity, it is increasingly important to approach problems formally. Mass spectrometry is currently experiencing an exciting watershed moment reminiscient of the early ages of the genomics era, where formal approaches to open problems promise to not only substantially improve our understanding of the processes that drive life, but also to propose questions that may result in novel statistical concepts and methods with myriad other applications.

BIBLIOGRAPHY

[1] S. K. Andersen, K. G. Olesen, and F. V. Jensen. *HUGIN, a shell for building Bayesian belief universes for expert systems*, pages 332–337. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[2] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[3] A. Belle, A. Tanay, L. Bitincka, R. Shamir, and E. K. OShea. Quantification of protein half-lives in the budding yeast proteome. *Proceedings of the National Academy of Sciences*, 103(35):13004–13009, 2006.

[4] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B*, 57:289–300, 1995.

[5] B. Efron, R. Tibshirani, J.D. Storey, and V. Tusher. Empirical Bayes analysis of a microarray experiment. *Journal of the American Statistical Association*, 96(456):1151–1161, 2001.

[6] J. E. Elias and S. P. Gygi. Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nature Methods*, 4(3):207–214, 2007.

[7] J. K. Eng, A. L. McCormack, and J. R. Yates, III. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*, 5:976–989, 1994.

[8] J. Feng, D. Q. Naiman, and B. Cooper. Probability-based pattern recognition and statistical framework for randomization: modeling tandem mass spectrum/peptide sequence false match frequencies. *Bioinformatics*, 23(17):2210–2217, 2007.

[9] S. P. Gygi, Y. Rochon, B. Franza, and R. Aebersold. Correlation between protein and mrna abundance in yeast. *Mol. Cell. Biol.*, 19(3):1720–1730, 1999.

[10] M. R. Hoopmann, G. E. Merrihew, P. D. von Haller, and M. J. MacCoss. Post analysis data acquisition for the iterative MS/MS sampling of proteomics mixtures. *Journal of Proteome Research*, 8(4):1870–1875, 2009.

[11] G. H. Jacobs, G. A. Williams, H. Cahill, and J. Nathans. Emergence of Novel Color Vision in Mice Engineered to Express a Human Cone Photopigment. *Science*, 315(5819):1723–1725, 2007.

[12] L. Käll, J. Canterbury, J. Weston, W. S. Noble, and M. J. MacCoss. A semi-supervised machine learning technique for peptide identification from shotgun proteomics datasets. *Nature Methods*, 4:923–25, 2007.

[13] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[14] A. Keller, A. I. Nesvizhskii, E. Kolker, and R. Aebersold. Empirical statistical model to estimate the accuracy of peptide identification made by MS/MS and database search. *Analytical Chemistry*, 74:5383–5392, 2002.

[15] S. Kim, N. Gupta, and P. A. Pevzner. Spectral probabilities and generating functions of tandem mass spectra: a strike against decoy databases. *Journal of Proteome Research*, 7:3354–3363, 2008.

[16] J. Klimek, J. S. Eddes, L. Hohmann, J. Jackson, A. Peterson, S. Letarte, P. R. Gafken, J. E. Katz, P. Mallick, H. Lee, A. Schmidt, R. Ossola, J. K. Eng, R. Aebersold, and D. B. Martin. The standard protein mix database: a diverse data set to assist in the production of improved peptide and protein identification software tools. *Journal of Proteome Research*, 7(1):96–1003, 2008.

[17] Q. Li, M. J. MacCoss, and M. Stephens. A nested mixture model for protein identification using mass spectrometry. *Annals of Applied Sciences*, 4(2):962–987, 2010.

[18] Y. F. Li, R. J. Arnold, Y. Li, P. Radivojac, Q. Sheng, and H. Tang. A Bayesian approach to protein inference problem in shotgun proteomics. In M. Vingron and L. Wong, editors, *Proceedings of the Twelfth Annual International Conference on Computational Molecular Biology*, volume 12 of *Lecture Notes in Bioinformatics*, pages 167–180, Berlin, Germany, 2008. Springer.

[19] 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*, 25:125–131, 2006.

[20] A. I. Nesvizhskii. A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. *Journal of Proteomics*, 73(11):2092 – 2123, 2010.

[21] A. I. Nesvizhskii, A. Keller, E. Kolker, and R. Aebersold. A statistical model for identifying proteins by tandem mass spectrometry. *Analytical Chemistry*, 75:4646–4658, 2003.

[22] Q. Pan, O. Shai, L. J. Lee, B. J. Frey, and B. J. Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. advanced online publication, November 2008.

[23] C. Y. Park, A. A. Klammer, L. Käll, M. P. MacCoss, and W. S. Noble. Rapid and accurate peptide identification from tandem mass spectra. *Journal of Proteome Research*, 7(7):3022–3027, 2008.

[24] T. S. Price, M. B. Lucitt, W. Wu, D. J. Austin, A. Pizarro, A. K. Yokum, I. A. Blair, G. A. FitzGerald, and T. Grosser. EBP, a program for protein identification using multiple tandem mass spectrometry datasets. *Molecular Cell Proteomics*, 6(3):527–536, 2007.

[25] L. Reiter, M. Claassen, S. P. Schrimpf, M. Jovanovic, A. Schmidt, J. M. Buhmann, M. O. Hengartner, and R. Aebersold. Protein identification false discovery rates for very large proteomics data sets generated by tandem mass spectrometry. *Molecular and Cellular Proteomics*, 8(11):2405–2417, 2009.

[26] N. Robertson and P. D. Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[27] Brian C. Searle. Scaffold: A bioinformatic tool for validating ms/ms-based proteomic studies. *PROTEOMICS*, 10(6):1265–1269, 2010.

[28] O. Serang, M. J. MacCoss, and W. S. Noble. Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data. *Journal of Proteome Research*, 9(10):5346–5357, 2010.

[29] C. Shen, Z. Wang, G. Shankar, X. Zhang, and L. Li. A hierarchical statistical model to assess the confidence of peptides and proteins inferred from tandem mass spectrometry. *Bioinformatics*, 24:202–208, 2008.

[30] H. Steen and M. Mann. The ABC's (and XYZ's) of peptide sequencing. *Nature Reviews Molecular Cell Biology*, 5:699–711, 2004.

[31] D. L. Tabb, C. G. Fernando, and M. C. Chambers. Myrimatch: highly accurate tandem mass spectral peptide identification by multivariate hypergeometric analysis. *Journal of Proteome Research*, 6:654–661, 2007.

[32] D. L. Tabb, W. H. McDonald, and J. R. Yates, III. DTASelect and Contrast: tools for assembling and comparing protein identifications from shotgun proteomics. *Journal of Proteome Research*, 1(1):21–26, 2002.

[33] M. M. Tai. A mathematical model for the determination of total area under glucose tolerance and other metabolic curves. *Diabetes Care*, 17(2):152–154, February 1994.

[34] 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*, 22:e481–e488, 2006.

[35] J. K. Taubenberger, A. H. Reid, A. E. Krafft, K. E. Bijwaard, and T. G. Fanning. Initial Genetic Characterization of the 1918 Spanish Influenza Virus. *Science*, 275(5307):1793–1796, 1997.

[36] Yair Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1):1–41, 2000.

[37] K. Zahradka, D. Slade, A. Bailone, S. Sommer, D. Averbeck, M. Petranovic, A. B. Lindner, and M. Radman. Reassembly of shattered chromosomes in Deinococcus radiodurans. *Nature*, 443(7111):569–573, September 2006.

[38] B. Zhang, M. C. Chambers, and D. L. Tabb. Proteomic parsimony through bipartite graph analysis improves accuracy and transparency. *Journal of Proteome Research*, 6(9):3549–3557, 2007.

VITA

Oliver Serang was raised outside Philadelphia until he was 12 and then in North Carolina where he fell in love with programming on a TI-85 graphing calculator (a necessary love of code optimization followed). Oliver graduated with a B.S. in Computer Engineering at North Carolina State University and is currently a graduate student in the department of Genome Sciences in Seattle, Washington, where he acquired a compilation of every good song ever done by anybody. Oliver has always loved challenging mathematical puzzles that a child can understand (e.g. "How many distinct ways can you count the votes in a tied election so that candidate A is never behind candidate B?" http://en.wikipedia.org/wiki/Catalan_number). He is a great fan of jumping into bodies of water, storms, the smell of wood fires, and the peptide PHADTHAI/L

.