Published in final edited form as:
IEEE/ACM Trans Comput Biol Bioinform. 2012 ; 9(3): 809–817. doi:10.1109/TCBB.2012.26.

Faster Mass Spectrometry-based Protein Inference: Junction Trees are More Efficient than Sampling and Marginalization by Enumeration

Oliver Serang and
Department of Neurobiology at Harvard Medical School and the Department of Pathology at Children's Hospital Boston, John F. Enders Research Laboratories, Suite 1155, 320 Longwood Avenue, Boston, MA 02115. Oliver.Serang@Childrens.Harvard.edu

William Stafford Noble
Departments of Computer Science and Engineering and Genome Sciences, University of Washington, Foege Building S-220B, Box 355065, 3720 15th Avenue NE, Seattle, WA 98195. noble@gs.washington.edu

Abstract

The problem of identifying the proteins in a complex mixture using tandem mass spectrometry can be framed as an inference problem on a graph that connects peptides to proteins. Several existing protein identification methods make use of statistical inference methods for graphical models, including expectation maximization, Markov chain Monte Carlo, and full marginalization coupled with approximation heuristics. We show that, for this problem, the majority of the cost of inference usually comes from a few highly connected subgraphs. Furthermore, we evaluate three different statistical inference methods using a common graphical model, and we demonstrate that junction tree inference substantially improves rates of convergence compared to existing methods. The python code used for this paper is available at http://noble.gs.washington.edu/proj/fido.

Index Terms

Mass spectrometry; protein identification; graphical models; Bayesian inference

1 Introduction

In tandem mass spectrometry, proteins in a mixture are digested into peptides, which are separated by hydrophobicity using liquid chromatography (LC) and then further separated by their mass-to-charge ratio (*m/z*). At each combination of elution time and *m/z*, the population of peptides is subsequently fragmented to produce an MS/MS spectrum (Figure 1). This spectrum can be thought of as a collection of statistics that describe the population of peptides. Using existing tools [1], each observed MS/MS spectrum is matched to the peptide that would theoretically produce the most similar statistics. Matched peptide spectrum pairs can be assigned probability-like scores, which estimate the probability that the match is correct [2, 3]. By modeling the relationship between proteins and their constituent peptides, the resulting bipartite graph on proteins and "peptide spectrum matches" (PSMs) can be used to perform statistical inference on the set of proteins initially present in the sample. Because the ultimate goal of many tandem mass spectrometry experiments is protein identification, the primary output of the statistical inference procedure is a posterior probability distribution over proteins. Performing accurate inference is of crucial importance; even small errors can permute the ranking of protein posteriors, and

result in a different set of identified proteins. Efficiency of inference is also paramount; data analysis must be substantially faster than the acquisition process.

As mass spectrometry-based proteomics has grown in prominence, several statistical procedures for protein identification have been proposed. These statistical methods for inference fall into two general classes. The first class is comprised of methods that conflate inference and modeling; they are described procedurally, without derivation from assumptions, and operate in a manner similar to EM, iteratively computing protein scores until convergence is reached. Examples of these methods include ProteinProphet [4] and EBP [5]. The second class of methods emerged in an attempt to improve upon the reliability of methods from the first class; these new methods employed models derived from formally stated assumptions. Methods in this second class do not conflate modeling with inference, and so they are are more robust to different experimental conditions. Examples of methods in this second class include methods which compute posterior probabilities with enumeration (Fido [6] and a hierarchical Bayesian model [7]) or with Gibbs sampling, a type of Markov chain Monte Carlo (MCMC) [8].

The methods in the second class have gained popularity due to their statistical rigor and robustness to large data sets [9]; however, in order to perform exact inference, these methods all have runtimes that may become exponential in the total number of proteins found in each connected subgraph. As mass spectrometry data sets become larger and more complex, performing rigorous inference may become prohibitively expensive.

The *junction tree* (also called “tree decomposition”) has been used to uncover structure in graphical models that would permit exact inference to be performed much more efficiently than with naive enumeration. Specifically, tree decomposition has been used to perform inference on models of gene expression [10] and to estimate probabilities in pedigrees with loops [11]. However, the efficiency of marginalization with junction trees depends on the connectivity of the graph analyzed: some sparsely connected graphs can be decomposed into junction trees where inference becomes trivial, while other highly connected graphs have no performance benefit from junction tree inference. To our knowledge, tree decomposition has never been applied to MS/MS protein inference, and so the practical benefit on protein inference graphs is unknown.

In this paper we use an existing state-of-the-art model [6] and compare the computational costs of using four different inference methods: marginalization by enumeration, two varieties of Gibbs sampling, and junction tree marginalization. We apply these methods to three data sets from organisms of varying biological complexity: *H. influenzae*, yeast, and *C. elegans*. We show that the computational cost of protein inference is dominated by a small number of difficult subgraphs, which each contain several proteins. Furthermore, we show that inference can be performed much more efficiently by using the junction trees, which can compute high-quality posteriors on these graphs in a fraction of the time required by naive enumeration and Gibbs sampling.

2 Materials and Methods

2.1 Data sets

For all of the inference methods we compare, the connectivity of the protein-to-peptide graphs and the coverage for the experiment determines the difficulty of protein inference. We analyzed the complexity of computing protein posteriors using protein lysate data from three model organisms of varying complexity: *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 a set of decoy proteins. 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, we cluster proteins adjacent to identical peptide sets. This can be trivially accomplished by building a dictionary of peptide sets to their adjacent proteins. Clustering the proteins in this way ensures that the complexity of the graph results from distinct proteins that are a challenge to resolve during protein inference. Proteins within the same cluster cannot be distinguished without some additional evidence or prior belief; therefore, clustering is performed to guarantee that computational cost results only from biologically interesting connectivity.

2.1.1 *H. influenzae*—Trypsinated *H. influenzae* lysate was separated by LC and subjected to MS/MS on an ESI-ITMS instrument. Using SEQUEST [12], mass spectra were searched against the *H. influenzae* proteome (using the UniProt database) combined with a decoy database containing the human proteome. PSMs were scored using PeptideProphet (using default parameters) with no minimum PSM score. This data set contains several highly connected subgraphs, but mostly due to the complexity of the decoy database. As expected, peptides associated with decoy proteins generally receive low scores, and so naive enumeration can be performed by pruning very low-scoring peptides, which essentially introduce no error.

2.1.2 *S. cerevisiae*—Lysate from mid log phase *S. cerevisiae* strain S288C was trypsinated and separated using LC and subject to MS/MS on an LTQ instrument. The resulting tandem mass spectra were searched using Crux [13] against a database containing all yeast ORFs (using the SGD database) and a shuffled decoy copy of each ORF. PSM scores were assigned using PeptideProphet (using default parameters) and with a minimum PSM probability of 0.05. A couple of moderately large connected subgraphs cannot be separated by pruning low-scoring peptides, and so probability estimates from those subgraphs must be approximated if marginalization is to be performed efficiently.

2.1.3 *C. elegans*—*C. elegans* of various developmental stages were grown on plates containing *E.* coli and cleansed of bacterial contamination using sucrose floating. The lysate was sonicated and digested with trypsin and subject to six technical replicate LC-MS/MS analyses on an LTQ instrument. The tandem mass spectra were searched against a database containing the *C. elegans* proteome (UniProt), known contaminants, and a reversed decoy copy of every target protein. PSMs were scored with PeptideProphet (using default parameters), and with a minimum PSM score of 0.05.

2.2 Protein inference model

Protein inference requires definition of a model, which provides an interpretable score for every proposed set of present proteins. Previously, we have introduced the Fido model and have shown that it performs similarly to established methods [6]. Originally, infererence with Fido was performed using marginalization by enumeration. We will briefly describe the model employed by Fido.

Qualitatively, the Fido model creates a directed acyclic graph (DAG) associating proteins to their constituent peptides and spectra to their best-matching peptides. Each protein has an independent prior probability γ of being present. Every peptide has an independent probabilty α of being emitted by an adjacent present protein and an independent probability β of being observed incorrectly due to noise. Each spectrum depends solely on its best-matching peptide, and each peptide is associated with only its best-matching spectrum. A more substantial description can be found in [6]. The conditional probability that each spectrum is observed given the state of its associated peptide is estimated by dividing the posterior estimate from Peptide-Prophet or Percolator by the estimated prior probability.

Because each spectrum depends only on its best-matching peptide and each peptide is conditionally independent of every other peptide given the set of proteins, it is possible to efficiently compute the likelihood of a set of proteins by marginalizing over the set of peptides in linear time.

For simplicity, the edge between each peptide and spectrum can be contracted to replace each associated peptide-spectrum pair with a PSM, resulting in a bipartite graph between proteins and PSMs, shown in Figure 2. The Fido model efficiently computes the likelihood and prior probability for any proposed set of present proteins; therefore, posteriors for each protein can be computed by marginalizing over all proteins in a connected subgraph. This marginalization is performed by enumerating all possible protein configurations for a connected subgraph; the runtime required for this is exponential in the number of proteins in a connected subgraph.

It is important to note that the Fido model shares similar characteristics to the model used by MSBayes, a method that performs inference with Gibbs sampling. Both models define an independent prior probability for each protein and a conditionally independent distribution for each peptide; however, MSBayes does not exploit this conditional independence and jointly samples peptides and proteins. MSBayes also has a more complex emission model; all peptides, not only those paired with an observed spectrum, are considered, and each adjacent peptide-protein pair is given a unique emission probability. This emission probability is predicted using peptide detectability data from another experiment. In contrast, Fido uses a single emission probability α for all peptides and proteins, and estimates its value empirically.

2.3 Approximate inference with the Fido model

A key advantage of the Fido method is the ability to compute high-quality approximations by transforming connected subgraphs into graphs that are equivalent or graphs that will result in very similar posterior estimates. There are two main graph transformations that make this possible: protein clustering and peptide pruning.

The first graph transformation, protein clustering, merges together proteins that are connected to identical sets of observed peptides. Because the protein nodes in the graph have identical connectivity, there will be several equivalent protein sets that will result in the same score; therefore, these equivalent configurations can be collapsed using a binomial transformation, effectively enumerating the *number* of proteins in the cluster that are present. Thus, a substantial speedup is observed, even though no error is introduced.

The second graph transformation, pruning, exploits the fact that degenerate peptides (*i.e.* peptides that are shared among multiple proteins or protein clusters) introduce a dependency between the proteins only when the scores of these peptides are nonzero. Because of this, degenerate peptides identified with scores of exactly zero can be split so that each protein or protein cluster that is adjacent to the peptide is now adjacent to a unique copy. Pruning can be used to split apart connected subgraphs containing many proteins; thus, an equivalent graphical inference problem is created, but where fewer proteins must be jointly enumerated. Likewise, when peptide scores are approximately zero, pruning can be used to separate large connected subgraphs at the cost of a small amount of error introduced. Thus, the user sets a threshold indicating the maximum number of proteins or protein clusters in a connected subgraph and a greedy algorithm separates the graph by pruning the lowest-scoring peptides. When a high marginalization cost is permitted, all proteins with dependencies are enumerated jointly, and no error is introduced. Any desired efficiency can be achieved at the cost of greater error in the posterior probability estimates.

3 Results

3.1 Inference procedures

Posterior probabilities for the Fido model can be estimated using alternative approaches to naive enumeration. In this paper we compare four procedures and evaluate their efficiency on real protein inference problems: they are marginalization by enumeration, Gibbs sampling, collapsed Gibbs sampling, and junction tree inference. Before the marginalization-based inference procedures (*i.e.* marginalization by enumeration and junction tree inference) are run, pruning can be used to achieve smaller connected subgraphs; peptides with nonzero scores can be pruned to compute approximate posteriors with varying degrees of error and speed. A similar comprimise between runtime and accuracy can be achieved with sampling procedures by varying the depth of sampling.

3.1.1 Enumeration—Marginalization can be trivially performed by enumerating and summing over all possible protein configurations for each connected subgraph; however, the cost of enumeration is exponential in the number of proteins in a connected subgraph, and is prohibitively slow when many proteins share a subgraph.

3.1.2 Gibbs sampling—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.

In our Gibbs sampler, we sampled from configurations defined by states for all peptides and proteins. Using a “blocking” approach similar to MSBayes, we proposed all possible changes to a random block of peptides and a random block of proteins. We used block sizes of three, which was found to yield best performance for MSBayes.

3.1.3 Collapsed Gibbs sampling—Because our model (like MSBayes) 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. We exploited this conditional independence by creating a “collapsed Gibbs sampler,” which samples some unobserved random variables and marginalizes out the remaining variables. In the collapsed Gibbs sampler, we sampled protein configurations only and marginalized out the peptide configuration given the sampled protein configuration. We used the same blocking strategy for proteins, sampling all possible perturbations to a random block of size three.

3.1.4 Junction tree—We also investigated junction tree inference, a more sophisticated approach to marginalization by enumeration. Naive enumeration 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 this enumeration, 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 3A can be thought of as two subgraphs that are “d-separated” by protein R_3; given a known boolean value for R_3, the graph separates into two disjoint subgraphs. Removing R_3 and its edges leaves two graphs, each containing two proteins; therefore, the entire graph can be

marginalized in $2 \times (2^2 + 2^2) = 2^4$ protein configurations rather than the 2^5 protein configurations required by naive enumeration.

In general, a protein set that results in a useful d-separation in the graph should be marginalized 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 set 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 [14]; however, in practice greedy heuristics can be used to create trees with low treewidth. Figure 3B 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 R_3 d-separates (R_1, R_2) from (R_4, R_5).

We performed tree decomposition using a minimum neighbor heuristic (nodes were eliminated in ascending order of the number of remaining neighbors in the graph). Inference was performed using our implementation of the Hugin algorithm [15], which efficiently computes the marginal posterior probabilities using the junction tree.

3.2 The cost of naive enumeration in real protein inference graphs

In order to investigate the difficulty of marginalization on real data sets, we divided each data set into connected subgraphs and counted the number of proteins in each connected subgraph. In the first row of Figure 4, the red solid series shows the distribution of naive enumeration 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 enumeration 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 enumeration to be practially useful. The unpruned *C. elegans* data contains no connected subgraphs with more than six proteins; therefore, naive enumeration is quite practical. It is somewhat surprising that naive enumeration 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 clustered together. These proteins are not necessarily identical; they simply contain the same set of observed peptides. Without clustering these proteins, the cost of naive enumeration for the *C. elegans* data would be much higher. Just as sparse graphs are highly separable, highly connected graphs result in components with identical connectivity, and so they benefit most from clustering

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. In our previous paper, we called this process "pruning" [6]. Because pruning can sometimes reduce the cost of marginalization without introducing any error, we consider the cost of naive enumeration when PSMs with very low PeptideProphet scores are pruned. The second row of Figure 4 shows the cost of naive enumeration 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 enumeration 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. We therefore analyzed the cost of naive enumeration 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 shows the cost of naive enumeration for the entire proteome of each of these species. As expected, naive enumeration 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 clustered, which may substantially increase the cost of inference.

3.3 The value of tree decomposition

For each connected subgraph, we also created the junction tree and computed the number of protein configurations required for marginalization using junction tree inference. A disparity between the cost of naive enumeration and the cost of junction tree indicates subgraphs with tree decompositions that can be marginalized more efficiently than with the naive approach. In Figure 4 we overlayed the distributions of cost for junction tree inference using a blue dashed series.

In the first row of Figure 4, we show 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 enumeration: 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 enumeration 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 enumeration. The improvement to the *C. elegans* data set is negligable, largely because the most computationally expensive subgraph for naive enumeration requires a modest 2^{6} steps.

The second row of Figure 4 shows the cost of junction tree inference after pruning the data. There is no improvement over naive enumeration for the pruned *H. influenzae* data set, which is quite easy when using naive enumeration. 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 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 enumeration, 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 enumeration is noteworthy: in one case, junction tree inference improves upon naive enumeration 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 enumeration, including subgraphs requiring 2^{89}, 2^{59}, and 2^{50} steps.

3.4 Convergence of inference methods

In order to compare empirically the various inference algorithms, we chose a moderately large subgraph of 22 proteins from the *S. cerevisiae* data set. We chose this subgraph because it is non-trivial for naive enumeration 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. In Figure 5A, we show the directed protein-to-PSM graph. In Figure 5B, we show 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 enumeration requires 2^{22} steps.

Figure 6 shows the tradeoff between error and efficiency when analyzing this subgraph using the four inference algorithms. For all methods we used the optimal α, β, and γ parameters for this data set from our previous paper [6]. On the x-axis, we plot the execution time and on the y-axis, we plot the largest absolute error introduced compared to doing exact marginalization. We implemented all methods 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 [6] 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 6 shows the average time-error relationship after replicating 10 times for each sampler; these replicate series were all very close to the average. Naive enumeration 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 enumeration, particularly when the graph is unpruned; however, when aggressive pruning of high-scoring peptides is permitted, naive enumeration is superior to junction tree inference because of the substantially lower overhead.

4 Discussion

We have 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 5, 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 [16]. It is important to note that junction tree inference and sophisticated variants of Gibbs sampling and collapsed Gibbs sampling would be trivially compatible with other Bayesian models (*e.g.* MSBayes [8] and the hierarchical Bayesian model [7]).

Furthermore, Figure 4 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 approaches we are aware of currently 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.

Acknowledgments

Funding NIH Award R01 EB007057.

References

1. Nesvizhskii AI, Vitek O, Aebersold R. Analysis and validation of proteomic data generated by tandem mass spectrometry. Nature Methods. 2007; vol. 4(no. 10):787–797. [PubMed: 17901868]
2. Keller A, Nesvizhskii AI, Kolker E, Aebersold R. Empirical statistical model to estimate the accuracy of peptide identification made by MS/MS and database search. Analytical Chemistry. 2002; vol. 74:5383–5392. [PubMed: 12403597]
3. Käll L, Canterbury J, Weston J, Noble WS, MacCoss MJ. A semi-supervised machine learning technique for peptide identification from shotgun proteomics datasets. Nature Methods. 2007; vol. 4:923–925.
4. Nesvizhskii AI, Keller A, Kolker E, Aebersold R. A statistical model for identifying proteins by tandem mass spectrometry. Analytical Chemistry. 2003; vol. 75:4646–4658. [PubMed: 14632076]
5. Price TS, Lucitt MB, Wu W, Austin DJ, Pizarro A, Yokum AK, Blair IA, FitzGerald GA, Grosser T. EBP, a program for protein identification using multiple tandem mass spectrometry datasets. Molecular Cell Proteomics. 2007; vol. 6(no. 3):527–536.
6. Serang O, MacCoss MJ, Noble WS. Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data. Journal of Proteome Research. 2010; vol. 9(no. 10):5346–5357. [PubMed: 20712337]
7. Shen C, Wang Z, Shankar G, Zhang X, Li L. A hierarchical statistical model to assess the confidence of peptides and proteins inferred from tandem mass spectrometry. Bioinformatics. 2008; vol. 24:202–208. [PubMed: 18024968]
8. Li, YF.; Arnold, RJ.; Li, Y.; Radivojac, P.; Sheng, Q.; Tang, H. A Bayesian approach to protein inference problem in shotgun proteomics. In: Vingron, M.; Wong, L., editors. Proceedings of the Twelfth Annual International Conference on Computational Molecular Biology. Vol. vol. 12. Berlin, Germany: Springer; 2008. p. 167-180.ser. Lecture Notes in Bioinformatics
9. Serang O, Noble WS. A review of statistical methods for protein identification using tandem mass spectrometry. Statistics and Its Interface. 2012; vol. 5(no. 1):3–20.
10. Dojer N, Gambin A, Mizera A, Wilczynski B, Tiuryn J. Applying dynamic bayesian networks to perturbed gene expression data. BMC Bioinformatics. 2006; vol. 7(no. 1):249. [PubMed: 16681847]
11. Totir L, Fernando R, Abraham J. An efficient algorithm to compute marginal posterior genotype probabilities for every member of a pedigree with loops. Genetics Selection Evolution. 2009; vol. 41:1–11. 10.1186/1297-9686-41-52.

12. Eng JK, McCormack AL, Yates JR 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. 1994; vol. 5:976–989.

13. Park CY, Klammer AA, Käll L, MacCoss MP, Noble WS. Rapid and accurate peptide identification from tandem mass spectra. Journal of Proteome Research. 2008; vol. 7(no. 7):3022–3027. [PubMed: 18505281]

14. Arnborg S, Corneil DG, Proskurowski A. Complexity of finding embeddings in a *k*-tree. SIAM Journal on Algebraic and Discrete Methods. 1987; vol. 8(no. 2):277–284.

15. Andersen, SK.; Olesen, KG.; Jensen, FV. HUGIN, a shell for building Bayesian belief universes for expert systems. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1990. p. 332-337.

16. Weiss Y. Correctness of local probability propagation in graphical models with loops. Neural Computation. 2000; vol. 12(no. 1):1–41. [PubMed: 10636932]

Biographies

Oliver Serang Oliver Serang received a B.S. in computer engineering from North Carolina State University, Raleigh in 2006 and a Ph.D. in Genome Sciences from the University of Washington, Seattle in the lab of William Noble in 2011. He is now a postdoctoral research fellow in the Steen lab in the Department of Pathology at Children's Hospital Boston and the Department of Neurobiology at Harvard Medical School. His research interests include probabilistic modeling, algorithms for statistical inference and optimization, and biomarker detection.

William S. Noble William Stafford Noble (formerly William Noble Grundy) received the Ph.D. in computer science and cognitive science from UC San Diego in 1998. After a one-year postdoc with David Haussler at UC Santa Cruz, he became an Assistant Professor in the Department of Computer Science at Columbia University. In 2002, he joined the faculty of the Department of Genome Sciences at the University of Washington. His research group develops and applies statistical and machine learning techniques for modeling and understanding biological processes at the molecular level.

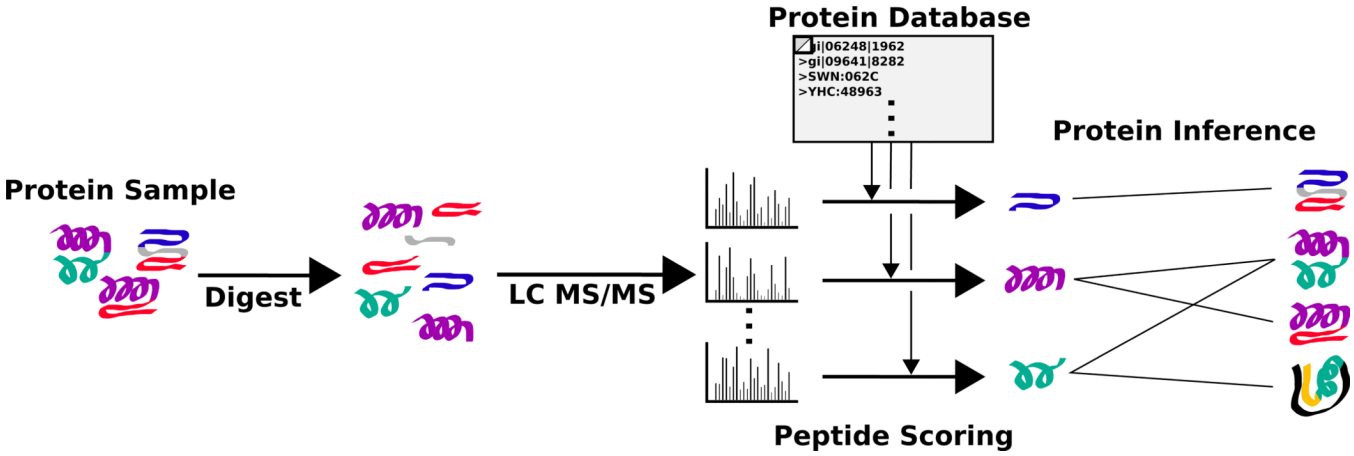

Fig. 1. Protein identification using tandem mass spectrometry

In the tandem mass spectrometry approach to proteomics, a protein mixture is first digested into peptides. These peptides are separated by hydrophobicity using LC and then by precursor *m/z*. Each resulting peptide population is fragmented to produce an MS/MS spectrum. These spectra are matched to peptides using a search database and the matches are scored. The graph mapping proteins to their scored constituent peptides is used for protein inference.

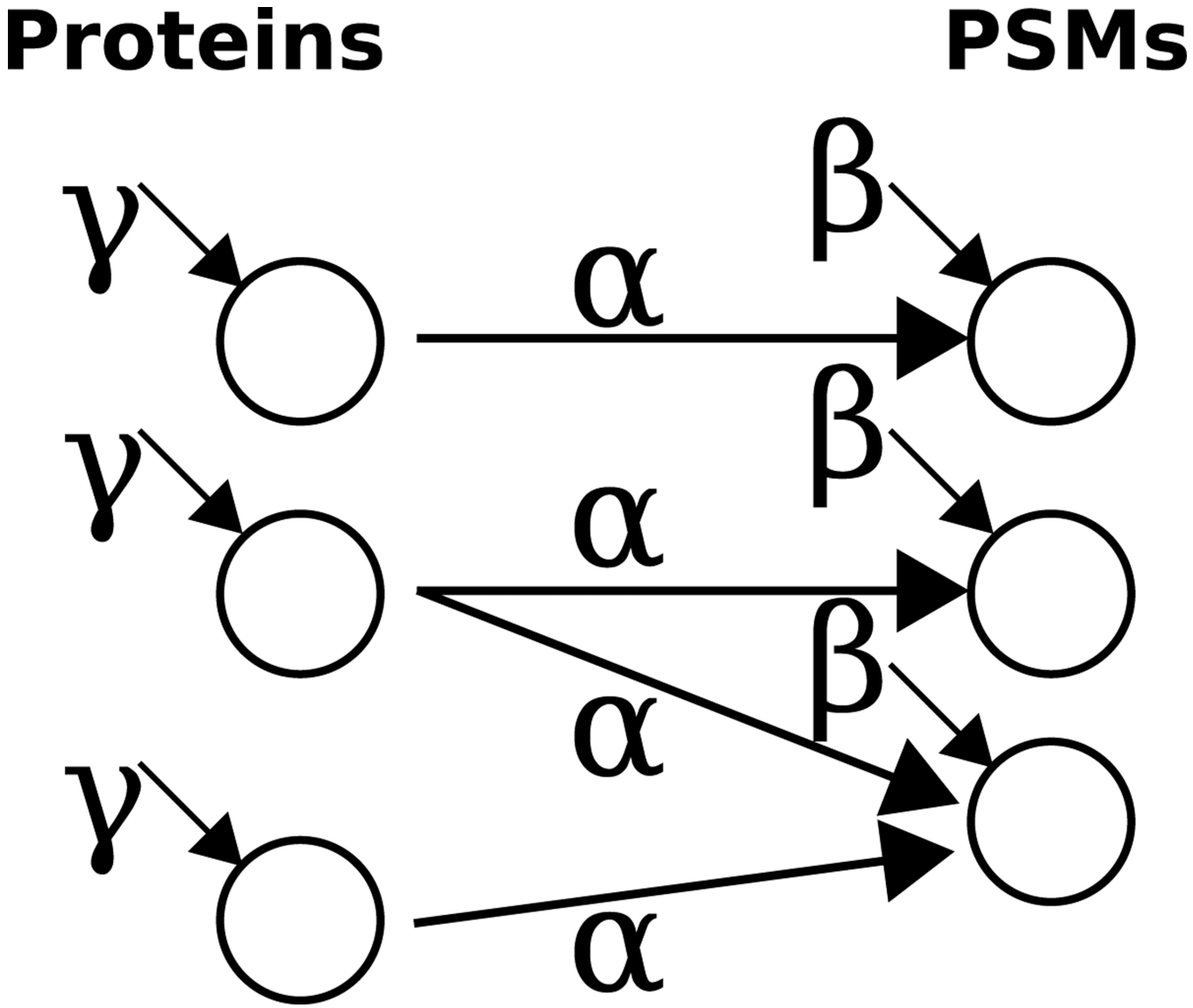

Fig. 2. Graphical view of the Fido model

In the Fido model peptides depend only on proteins which would produce them when digested. Spectra only depend on their best-matching peptides; these peptide-spectrum pairs are merged into PSMs. Proteins have prior probability γ that they are present, and a present protein has probability α of emitting an adjacent peptide. Peptides have probability β that they will be created due to some error.

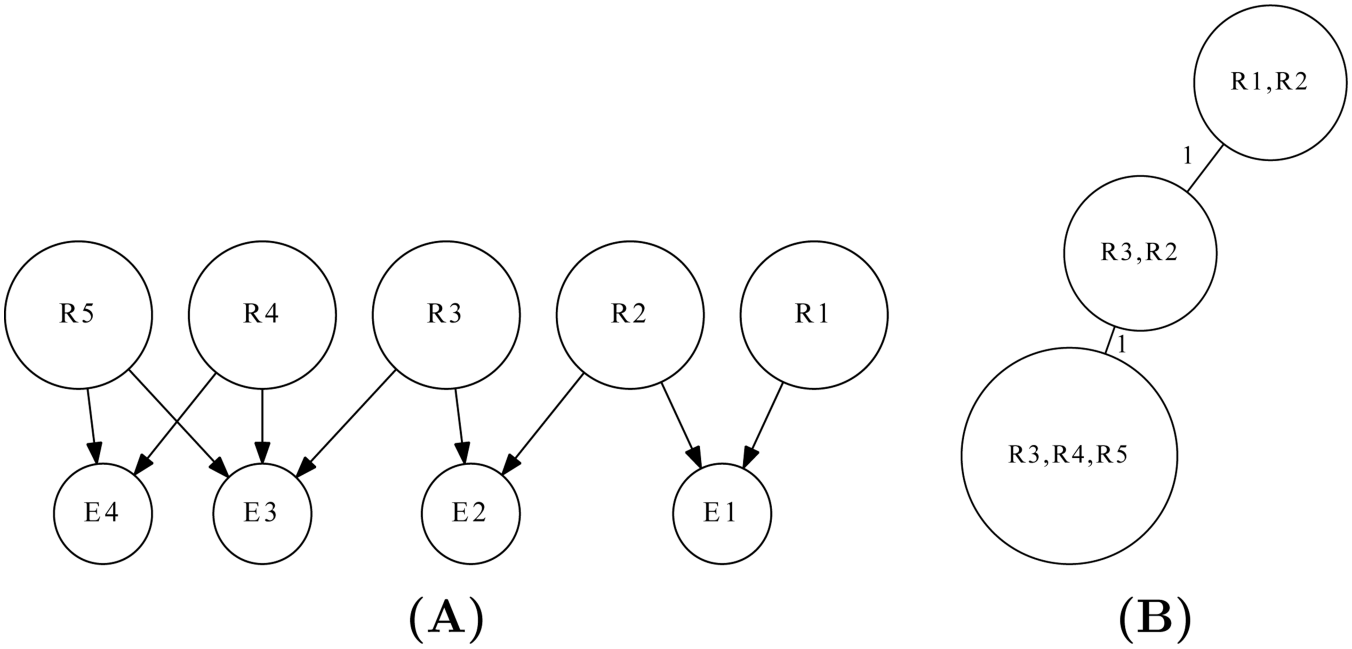

Fig. 3. Decomposing a graph for protein inference

(A) An example DAG for a five protein inference problem. Proteins, labeled with prefix *R*, are shown as large circles and peptides, labeled with prefix *E*, are shown as smaller circles. Choosing a specific state for protein R_3 effectively removes R_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. Weights on the edges indicate the size of the overlap between adjacent nodes.

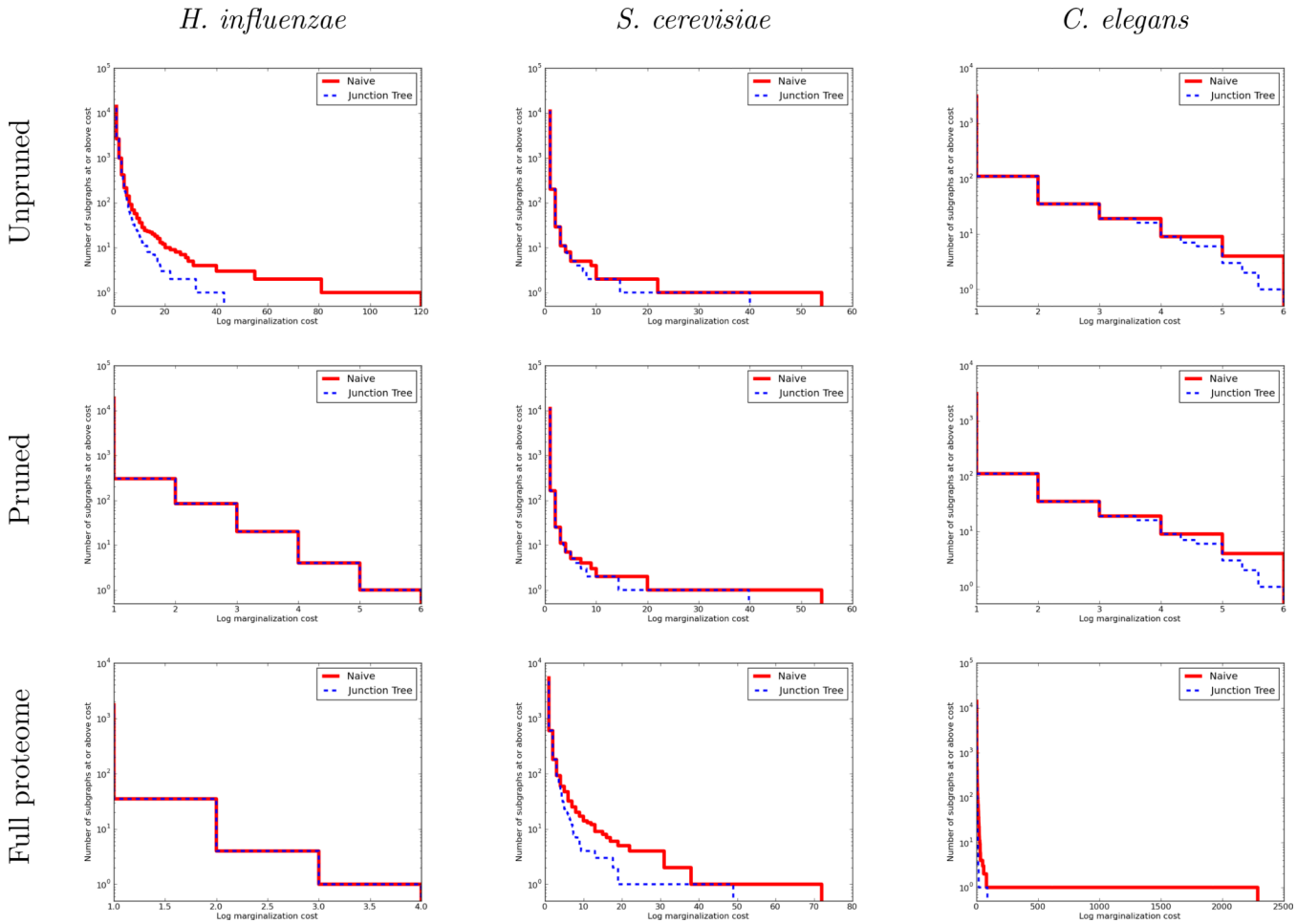

Fig. 4. Distribution of marginalization complexities

For each organism we plot 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.

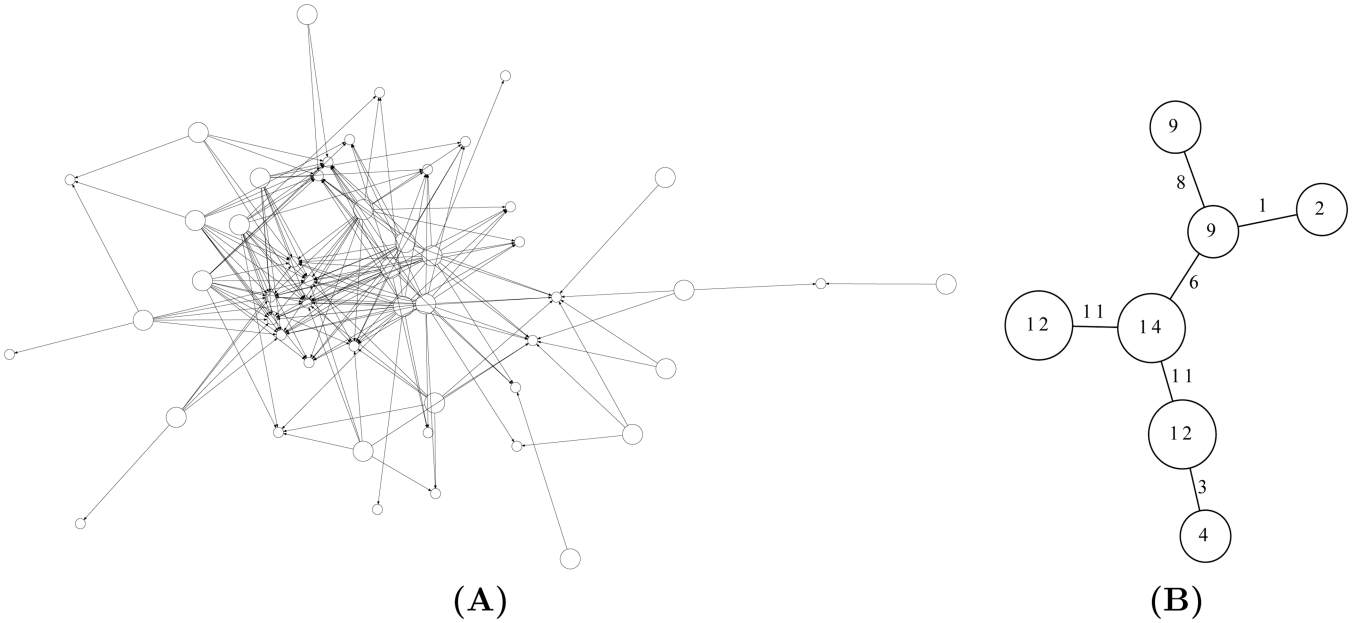

Fig. 5. A difficult connected subgraph and its tree decomposition

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 these proteins, all peptides shared by them 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.

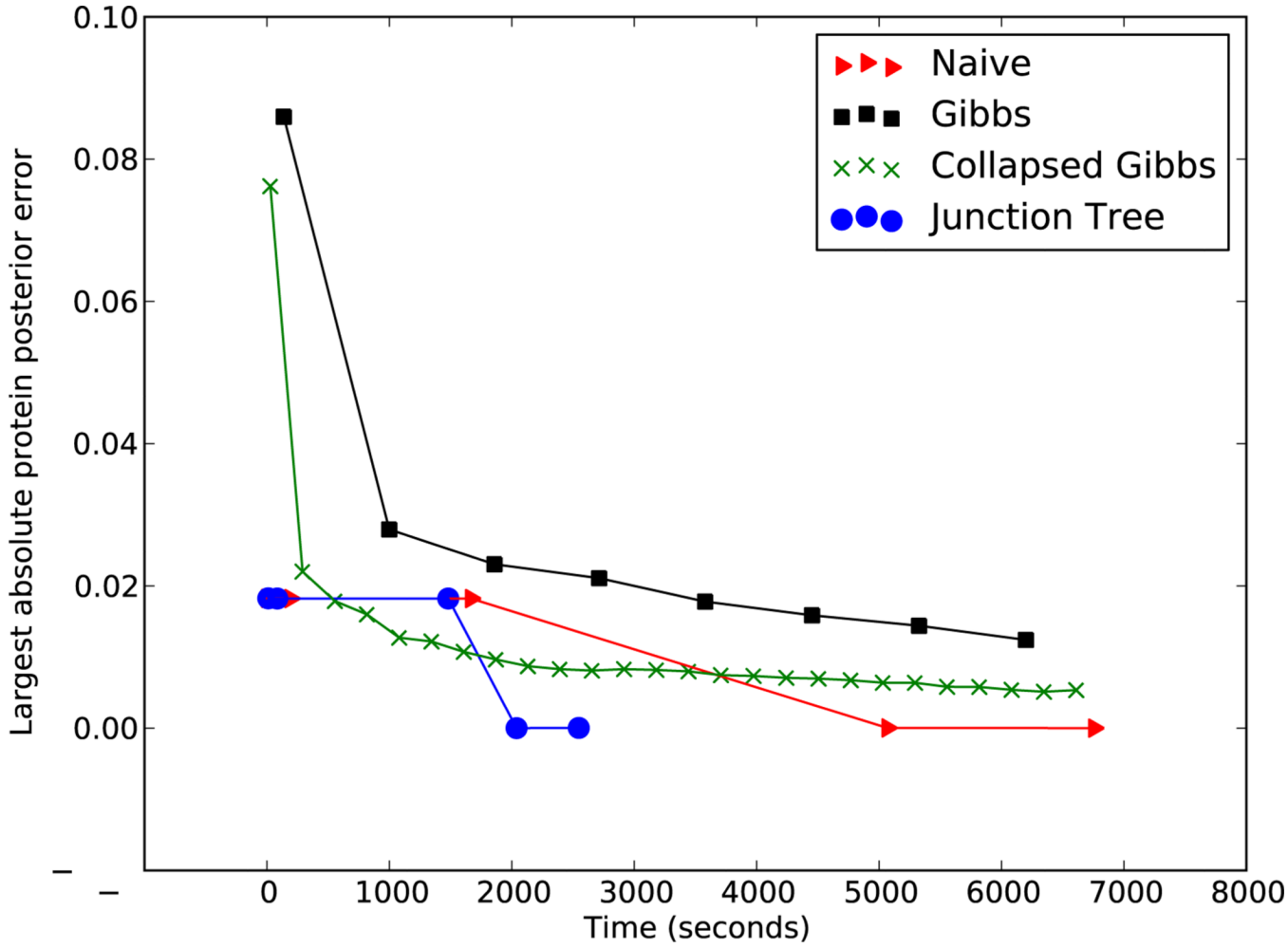

Fig. 6. Convergence of various inference methods

For each method, we plot the log relationship between the largest posterior error and runtime for protein inference on the subgraph from Figure 5. 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. For the Gibbs sampler and collapsed Gibbs sampler, error bars indicate the smallest and largest errors achieved in the 10 replicate runs.