Journal of proteome research

pubs.acs.org/jpr

Article

EPIFANY: A Method for Efficient High-Confidence Protein Inference

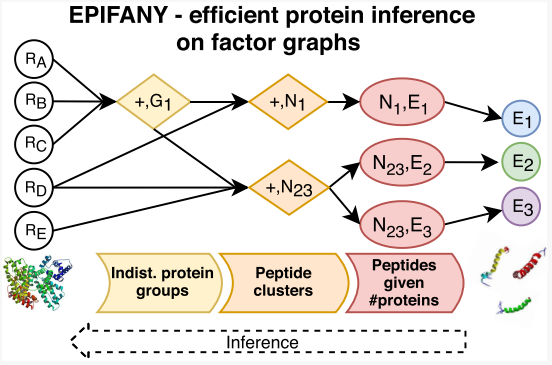
Julianus Pfeuffer,* Timo Sachsenberg, Tjeerd M. H. Dijkstra, Oliver Serang,* Knut Reinert, and Oliver Kohlbacher*

Cite This: *J. Proteome Res.* 2020, 19, 1060−1072

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Accurate protein inference in the presence of shared peptides is still one of the key problems in bottom-up proteomics. Most protein inference tools employing simple heuristic inference strategies are efficient but exhibit reduced accuracy. More advanced probabilistic methods often exhibit better inference quality but tend to be too slow for large data sets. Here, we present a novel protein inference method, EPIFANY, combining a loopy belief propagation algorithm with convolution trees for efficient processing of Bayesian networks. We demonstrate that EPIFANY combines the reliable protein inference of Bayesian methods with significantly shorter runtimes. On the 2016 iPRG protein inference benchmark data, EPIFANY is the only tested method that finds all true-positive proteins at a 5% protein false discovery rate (FDR) without strict prefiltering on the peptide-spectrum match (PSM) level, yielding an increase in identification performance (+10% in the number of true positives and +14% in partial AUC) compared to previous approaches. Even very large data sets with hundreds of thousands of spectra (which are intractable with other Bayesian and some non-Bayesian tools) can be processed with EPIFANY within minutes. The increased inference quality including shared peptides results in better protein inference results and thus increased robustness of the biological hypotheses generated. EPIFANY is available as open-source software for all major platforms at https://OpenMS.de/epifany.



KEYWORDS: *bottom-up proteomics, protein inference, Bayesian networks, convolution trees, loopy belief propagation, iPRG2016*

INTRODUCTION

Ever since the emergence of bottom-up proteomics experiments,[1] mapping the identified peptides back to their most plausible source proteins, the protein inference problem, has been a key problem in proteomics.[2−4] High dynamic range of protein abundance, limitations in digestion, separation, and mass spectrometry result in incomplete coverage of the source proteins by identified peptides. Reconstructing the source proteins originally present in the sample should thus rely on as much of the experimental evidence (i.e., peptide identifications) as possible, which also includes nonunique peptides shared between multiple source proteins. Starting from a notion of a probability of the presence or absence of peptides in the sample, usually expressed by a score, we want to infer the presence or absence of the proteins these peptides originated from. Due to the common presence of ambiguous peptides arising from one or more proteins sharing parts of their amino acid sequence, this is not a trivial task.[2]

The scores for peptides are typically obtained by so-called peptide search engines that match experimentally observed spectra to theoretically derived ones based on the sequences of an in silico-digested database of protein candidates. Those peptide-spectrum matches (PSMs) then need to be scored to be able to quantify the uncertainty in correctness of such a match. Uncertainty in the assignment of a peptide sequence to a spectrum may be a consequence of multiple peptide candidates matching to the same spectrum or a result of imperfect data such as incomplete or noisy spectra as well as incomplete protein databases.[5]

The formulation of protein inference algorithms naturally leads to a representation of the relation between peptides and proteins as a bipartite graph of nodes (proteins and peptides) that are connected with an edge if a peptide is part of the theoretical digest of the (parent) protein (Figure 1). Figure 1 also shows that the ambiguity of peptides across proteins may lead to proteins without unique evidence (e.g., protein E) and, in an extreme case, to experimentally indistinguishable protein groups (e.g., one comprising proteins F and G, which share all of their observed peptides). Reasons for ambiguous peptides are manifold in biology and include, among others, homology, alternative splicing, or somatic recombination. Depending on the degree of ambiguity between the peptides of different proteins, they are often clustered into various types of so-called protein ambiguity groups.[2] It should be noted that those

Received: August 21, 2019
Published: January 24, 2020

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

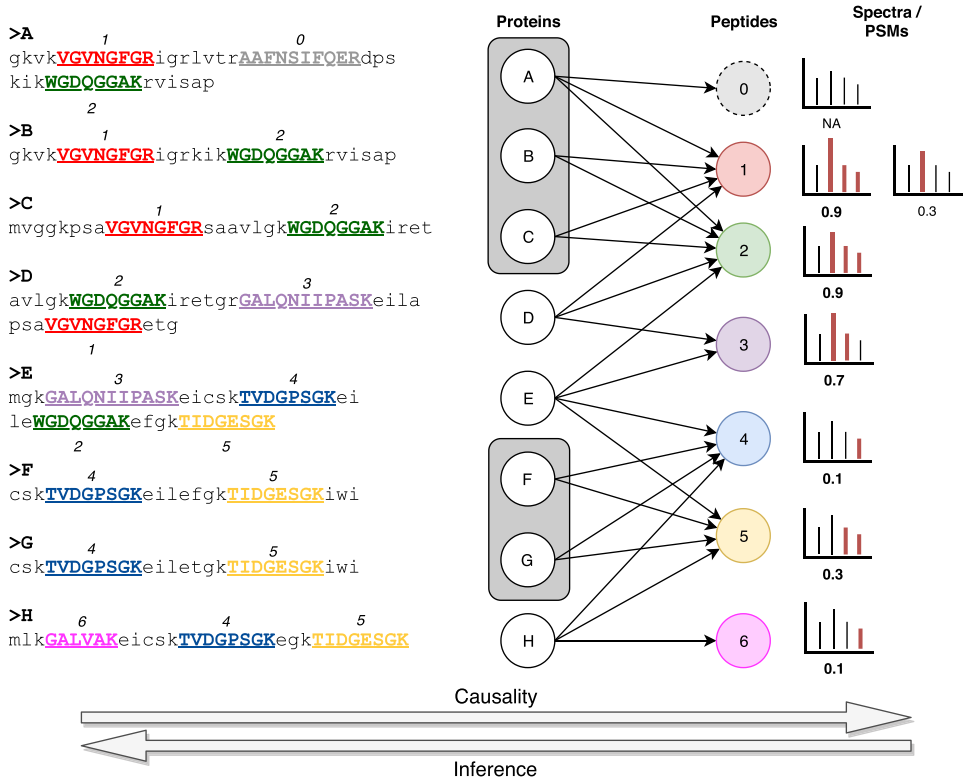


Figure 1. Example of a bipartite protein−peptide graph. Nodes with letters represent potential proteins from the input database. Colored nodes are the peptides from in silico digest with the given enzyme (trypsin). Arrows are drawn when a protein may theoretically generate a peptide. Dashed circles represent experimentally unobserved entities due to missing (NA) peptide-spectrum matches. Red peaks in the sketched hypothetical tandem mass spectra were matched to a theoretical ion of the peptide that matched best to this spectrum. Probability scores roughly follow a dot-product-based score but were invented for the sake of this example. Bold scores highlight the chosen match probability for this peptide (i.e., the maximum probability). The left side shows the used protein database with its tryptic peptides (upper-case bold underlined substrings) following the same color and number scheme as the nodes in the graph. Proteins in the same shaded curved rectangle comprise an experimentally indistinguishable (ambiguity) group. The arrows on the bottom show the general directions of the two processes: causality in the course of the experiment and inference based on the observed data.

groups can be defined solely either based on the experimentally observed peptides or based on all theoretically possible peptides.[6] Preliminary grouping (especially of indistinguishable proteins) is often used automatically by inference algorithms to solve a less ambiguous problem. While this is enough for studies that only need to confirm the presence of any protein in a group (using group-level inference probabilities), other studies look for the effects of a certain isoform on a disease and need to know how likely it is that this specific isoform was expressed (therefore interested in single protein-level results). Comparing results of inference methods on the same level (wherever possible) is an important consideration during benchmarking.

Early inference approaches resorted to simple rule-based conclusions. If a protein is connected to n or more peptides—where n is usually one or sometimes two to avoid so-called one-hit-wonders—then it is declared present; otherwise, it is considered absent. The problem with such approaches, however, is the implicit overcounting of shared peptides. In the presence of very large proteins like titin, false-positive identifications may arise due to matching one of its many peptides. Similarly, titin might be wrongly identified if it just shares enough peptides with truly present proteins. Some methods tackle this problem by ignoring shared peptides (Percolator[7,8]), employing maximum parsimony principles and finding a minimal set of proteins explaining found peptides or PSMs (PIA[4]), iteratively distributing its evidence among all parents (ProteinProphet[9]), or incorporating the evidence in a fully probabilistic manner (Fido,[10] MSBayesPro,[11] MIP-GEM[12]) to make use of the "explaining-away" property of Bayesian networks (BNs).[13] Explaining-away is a term used in probabilistic reasoning to describe the implicit conditional dependency between multiple causes of a common effect (when its probability is nonzero). In this case, knowledge about one cause from other evidence or prior to inference influences our belief about the other causes. In probabilistic models with synergistic parametrizations like the ones in the aforementioned Bayesian tools, this means that if one protein is very likely to be present from its unique evidence, it is already a sufficient explanation for peptides that it shares with other proteins (without evidence) and thereby affects their probability in a negative way. This leads to a probabilistic type of parsimony.

On a gold standard data set, it was shown by The et al.[14] that fully probabilistic models perform among the best in terms of the pure identification task. However, current solutions are computationally demanding. Additionally, in the case of candidate protein databases with many cases of peptides being shared between truly present and absent proteins, the reported probabilities are not a good basis for well-calibrated target-decoy false discovery rates (FDRs) as they yield poor

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

approximations of the true FDR. This leads to over-/underestimation of the true amount of false discoveries.

In our new approach Efficient Protein InFerence for ANY protein−peptide network (EPIFANY), we used a fast approximate inference algorithm called loopy belief propagation (LBP), which has already been shown to perform well in solving other types of probabilistic graphical models (e.g., models used in important information theoretic algorithms like the error-correcting turbo codes[15] as well as on quick medical reference (QMR) disease diagnosis networks[16]). Using LBP, we can achieve drastically improved runtimes than other Bayesian approaches without any approximations on the underlying graph itself. We improved the calibration of the resulting FDRs by introducing an optional regularized model with max-product inference and a greedy protein group resolution based on the reported protein probabilities.

METHODS

In the following section, we define the underlying probabilistic graphical model used to describe the protein inference problem including all of its conditional dependencies. Then, we present the inference algorithm, which allows one to efficiently calculate probabilities for peptides, proteins, and protein groups. In addition, we provide information on the pre- and postprocessing of the data.

Model

The model we chose for protein inference is based on a Bayesian network (BN) representation. The protein−peptide graph (Figure 1) encodes the conditional dependencies of proteins and their peptides. The advantage of this specification of conditional (in-)dependencies is the resulting factorization of the high-dimensional joint distribution into smaller distributions, namely, prior distributions for the proteins and conditional probability distributions (CPDs) for the peptides given their parents. In the case of the binary representation for every peptide and protein, these distributions are discrete and correspond to conditional probability tables (CPTs).

Additionally, the Bayesian network needs to be parametrized. Although the factorization into smaller distributions decreases the number of parameters, each CPT still needs 2^p parameters, where p is the number of parent nodes, to be set or learned. By recognizing the fact that in the generative process from proteins to peptides the presence of any of the parent proteins is enough to potentially produce a peptide, we can reduce the number of parameters further when specifying the conditional probability according to a noisy-OR model.[13] In its original form, the network using the noisy-OR model requires the following parameters:

- γ_ρ prior for protein with index ρ
- $\alpha_{\rho,\varepsilon}$ noisy-OR emission probability of a protein ρ generating peptide ε and
- β_ε noisy-OR leak probability for a peptide ε being generated by chance.

For now, we employ the same simplification used in the Fido[10] algorithm by assuming equality among all α and the presence of only one β. Additionally, a constant prior γ for all proteins prevents biases on the protein level when no further information is available. The parameters α, β, and γ either have to be specified manually or are by default selected from a grid of initial values based on target-decoy classification performance and probability calibration (see the Implementation subsection for details).

One addition to the original model is an option to add prior probabilities for the random variables (RVs) that model the number of proteins that might produce a certain shared peptide. These priors were designed empirically to decrease with a rising number of present parents. In the course of inference, this results in a form of regularization on the number of parent proteins and a more uneven distribution of the evidence from shared peptides starting at the most likely producing proteins (based on their beliefs from the rest of the network), especially in conjunction with the so-called max-product inference. For a more detailed description of the parameterization including equations and an example, please refer to the Supplementary Methods (Section 25.1).

Algorithm

Once the protein inference problem is modeled as a (factorized) probability distribution, probabilities can be inferred on random variables of interest. Usually, probabilities of interest are the marginal probabilities of proteins, protein groups, or peptides, given the evidence on the PSM level, the so-called posterior probabilities.

As peptide-level evidence, the algorithm first reads the PSM probabilities and the associations with their parent proteins from spectra searched with a peptide search engine. It then aggregates PSM probabilities at the peptide level by picking the maximum PSM probability per unmodified peptide sequence and filters out peptides with extremely small probabilities (e.g., below 0.001).

A naive approach to inference in a Bayesian network would be to create a large joint probability table for each connected component of the network and marginalize each protein by aggregating the probabilities of all possible configurations leading to the same state of the current protein of interest. To make this approach viable for at least small to medium-sized problems, previous tools resorted to (Gibbs) sampling[11] or sped up calculations by caching results and making use of symmetries arising due to the chosen model.[10] A possible symmetry to be exploited is the dependence of the peptide probability only on the number of parent proteins (not their exact combination). This symmetry consideration reduces the number of different input configurations in the presence of indistinguishable groups. Although this procedure has been implemented efficiently in tools like Fido, the worst-case runtime is exponential in the number of proteins in a connected component. Reducing the size of the connected components by splitting them at low-probability peptides is a reasonable approximation only if the probability cutoff is not too high. Using the looping version[16] of Pearl's belief propagation algorithm,[17] even nontree structured graphs with cycles (such as all but the most simple protein−peptide networks) can be processed efficiently while keeping flexibility in which types of factors (probability table-based factors, functionlike factors, convolution-tree-based adder factors, etc.) on which sets of random variables are used. Convolution trees[18] (CTs) leverage fast Fourier transformation to calculate the convolution of probability distributions. Additionally, convolutions are performed in a hierarchical procedure to keep the sizes of intermediate tables as small as possible. EPIFANY uses CTs to propagate probabilities between peptide and protein (group) levels efficiently even at peptides with many parents (see Supplementary Methods Section 25.2 for details). To apply loopy belief propagation to our problem, we create a factor graph (see Figure 2 and Implementation

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

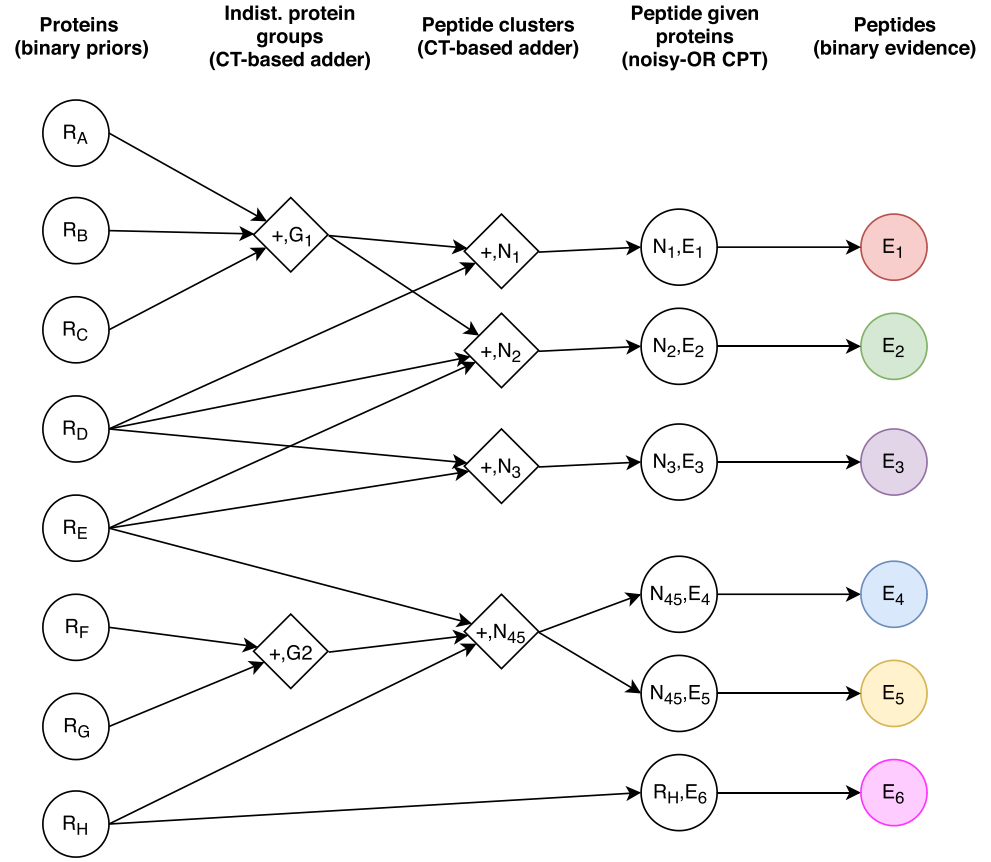


Figure 2. Factor graph created for the loopy belief propagation algorithm based on the example in Figure 1 (same color and letters). Each node represents one factor. Annotations on the factors describe the set of random variables (RVs) comprising the dimensions of its initial potential and in later stages its updated beliefs. Circles are table-based factors; diamonds are convolution-tree (CT)-based probabilistic adders. The set of RVs on CT-based adders is implicitly defined by the union of variables from neighbors on the left side of the graph (indicated by a plus sign) and an output variable. Although edges are displayed unidirectionally to represent the causality, messages will be passed in both directions.

subsection) from our model and initialize messages on all edges in both directions uniformly. In the following, we will use the term "belief" for the current marginal probabilities of a set of variables in a factor. All beliefs on a factor are initialized according to their priors, evidence, and/or the likelihoods calculated based on the parameterization (i.e., noisy-OR parameters and regularization). Lastly, starting from factors that carry initial information, the algorithm iteratively queues, updates, and passes messages between the factors (to update their beliefs) until messages do not change anymore (i.e., convergence is reached in terms of their mean-squared error). By default, messages with the highest residuals, compared to the last message, gain the highest priority for the next iteration.[19] Beliefs are updated by incoming messages following the HUGIN algorithm.[20] Details about the algorithm can be found in the Supplementary Methods on an applied example (Section 25.2). Since the loopy belief propagation is an approximate algorithm on nontree structured parts of the network and convergence is an important concern, messages can additionally be dampened (by a "momentum") in a way such that the updated message is a convex combination of old and new messages.[16] The idea of damping for solving the problem of oscillating messages is further discussed in the Limitations section and on an example in the Supplementary Methods (Section 25.3).

Implementation

EPIFANY starts with the output from probabilistic rescoring tools applied to peptide search engine results. It writes the PSM probabilities and associations from PSMs to proteins into OpenMS' datastructures. After aggregating PSM probabilities at the peptide level and creating the bipartite graph from the peptide mapping to potential protein candidates as described in the previous section, the resulting protein−peptide graph is split into connected components by a depth-first search. Then, using the OpenMP 2.0 API[21] in a dynamic scheduling mode, the processing of the connected components is distributed on as many CPU threads as allowed by a user-specified parameter. For each connected component in the graph, a factor graph is built, equivalent to the Bayesian network specified in the Model section. The Bayesian network represented by the bipartite graph is converted to a factor graph as follows (example shown in Figure 2). First, to allow querying posteriors for indistinguishable protein groups, an additive factor is introduced for all sets of proteins that share the same set of peptides. Then, to reduce loops, save computations, and avoid oscillations later on, we also create peptide cluster factors that calculate and hold the current beliefs for the number of parent proteins for sets of peptides that share the same parent proteins or parent protein groups. Both of these factor types use convolution trees for an efficient adding of discrete random variables.[18] It is worth noting that this hierarchy could also include more intermediate factors, but since no subquadratic

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

algorithm is known to us to create an optimal hierarchy for the parent protein (sub-)sets, the algorithm uses only two levels of aggregation. Also, this aggregation is performed only if the number of contributing proteins or protein groups is greater than one (see, e.g., peptide E_6 in the example). Finally, for each peptide, a factor is added to the graph, which holds the CPT for the probability of a peptide being present given the number of parent proteins. The factor nodes on the very left and very right are just singleton factors to keep track of the beliefs on proteins and peptides. They are initialized with priors (parameter γ for proteins) and evidence probabilities (for peptides, e.g., from Percolator) and, after convergence of the algorithm, hold the final posteriors to be queried. This allows the simultaneous reporting of protein, protein group, and updated peptide-level posteriors as the outcome of the inference on this unified model.

An additional outer loop evaluates the model for (by default 42) points on a three-dimensional grid over the three parameters α, β, and γ based on a convex combination of partial AUC (for target-decoy classification) and posterior probability calibration (i.e., comparing posterior-based and target-decoy-based FDRs). Details on parameter optimization can be found in Supplementary Methods (Section 25.4).

As an optional postprocessing, a greedy protein group resolution can be performed. It implements a probabilistic maximum parsimony model, where protein groups are ordered by their posterior probability, and starting from the best group, each greedily claims all peptides that it potentially generates until all peptides have been claimed. Proteins or protein groups without any remaining evidence are then deleted or, if preferred, implicitly assigned a probability of 0.

Data and Data Preprocessing

To benchmark the main advancements of EPIFANY and to show the different strengths of the tool, we focused on three distinct data sets.

Accuracy and calibration can best be measured on a data set like the iPRG2016 benchmark data with a set of known ground-truth Protein Epitope Signature Tags (PrESTs).[14] Two sets (labeled A and B) of known PrESTs were designed to share a large number of peptides, spiked into an *Escherichia coli* lysate background in three different experiments, and then measured in triplicates: one experiment each exclusively containing one of the spike-in sets (A, B) and a third containing both sets of PrESTs (A + B). For all sets, we evaluated the performance on identifying the PrESTs of A/B/A + B while having the full database of spike-ins (from A + B), entrapment proteins (i.e., intentionally absent PrESTs), and background proteins as potential candidates. All experiments contained an equimolar concentration of each PrEST. To avoid confusion, we will simply use the term "protein" instead of PrEST in the rest of the manuscript.

A small Universal Proteomics Standard 2 data set was additionally analyzed with the same pipeline as for the iPRG data to better evaluate the pros and cons of the methods in detail.

With the goal to measure the scalability of the new tool, a third, larger-scale data set was analyzed. It is part of an unpublished study and consists of two measurements of human immune cells on a long gradient at two different time points in duplicates.

iPRG2016 Data Set. Raw files from the corresponding PRIDE[22] project PXD008425 were converted and centroided with msConvert[23] on all levels. The `fasta` database provided together with the study was used to generate a decoy database through homology-aware (i.e., peptide-based) shuffling of amino acids with the OpenMS' DecoyDatabase[24,25] tool. Then, spectra were searched using Comet (2016.01 rev. 3),[26] allowing a 10 ppm precursor mass tolerance and one missed cleavage for fully tryptic peptides. As fixed modification, we required carbamidomethylation (C). Variable modification was set to oxidation (M). After merging the results over replicates (by creating the union of proteins and concatenating PSMs), we added target-decoy annotations on the protein and PSM levels. To obtain better discrimination through Percolator 3.02, we extracted additional features specific for the Comet search engine (see Section 25.5 in the Supplementary Methods and the corresponding Supplementary Table S1) before running Percolator with standard settings for the PSM score recalibration and basic protein inference. For all other methods tested, we used the PSM-level posterior error probabilities reported by Percolator as the input after filtering them at different levels of PSM confidence—once slightly by removing PSMs with error probabilities higher than or equal to 0.999 (to be consistent with the defaults in Fido and EPIFANY), once at 5% FDR, and once at 1% FDR. For comparability with Percolator, which supports only group-level reports, Fido and EPIFANY were run with group-level inference as well, thereby querying posteriors on the (indistinguishable) protein group level (i.e., reporting the probability of at least one member being present). ProteinProphet and EPIFANY then report both group-level and single protein-level probabilities. PIA's recommended default inference procedure "spectrum extractor" also reports groups. Its "report all" option without any parsimonious steps during inference was additionally tested for completeness.

UPS2 Standard. Raw data for the UPS2 standard in solution was downloaded from http://data.marcottelab.org/MSdata/Data_13/. The highest concentrated sample (30 μL) was chosen to potentially yield the most discoverable proteins. The converted mzML was reuploaded for reproducibility.[27] A database with the 48 known proteins in various concentrations and 6 differently shuffled decoys for each protein was used as provided by the lab. One set of decoys was masked before the Comet search such that full ground-truth information is at no point available for any of the methods. Comet settings were the same as for the iPRG data. The different FDR cutoffs (1 and 5%) at which PSMs were filtered were corrected for the factor of additional protein decoys present. During evaluation of protein FDRs, we included the knowledge about both decoys and entrapment proteins again by counting false positives with a factor of 1/6.

Large-Scale Data. After searching the spectra of four runs with MSGF+[28] (unspecified number of missed cleavages, 10 ppm precursor mass tolerance, fully specific trypsin/P as enzyme, fixed methylthio (C) modification, variable oxidation (M)) against the whole human part of the UniProt database (SWISS-PROT + TrEMBL)[29] (release 2017_06, 70 939 proteins plus peptide-level pseudo-reversed decoys appended) and merging the samples, the data set can be summarized by the following numbers:

- 119 921 proteins (matched by at least one PSM);
- 533 218 different peptide sequences;

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

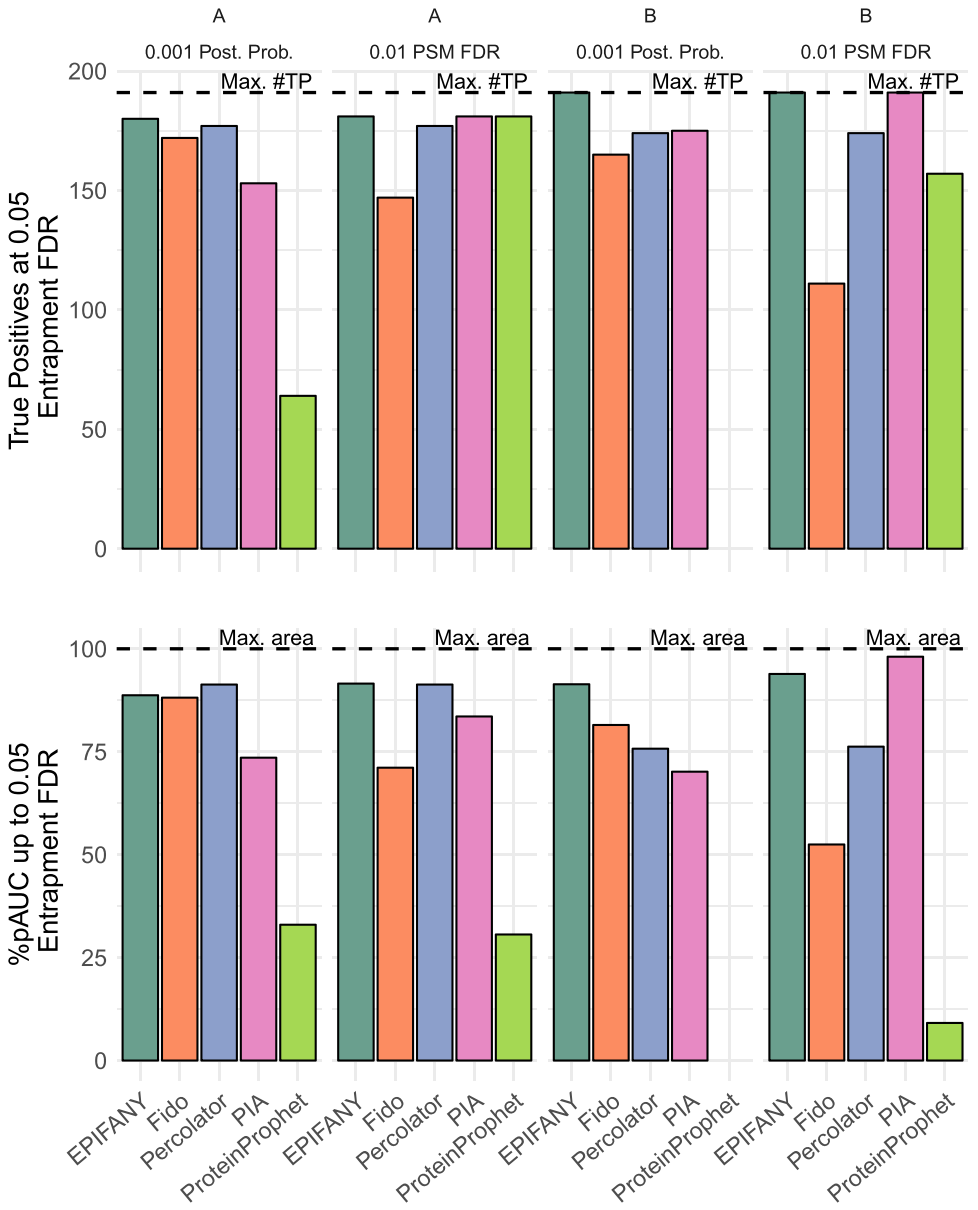


Figure 3. Summary statistics on the identification performance of different inference methods at a 5% protein entrapment FDR on the iPRG2016 data set, samples "A" and "B". PSMs were filtered at two distinct levels (at 0.001 posterior probability and a stricter 0.01 PSM FDR). Upper: Number of true-positive proteins found. The maximum number of true positives according to the ground-truth database given is 191 for B and 192 for A, as indicated by a dashed horizontal line. Lower: The percentage of the maximum area under the partial receiver operating curve (% pAUC) as a measure of how quickly methods accumulate true positives at increasing FDR levels until the chosen cutoff at 5% protein entrapment FDR.

- 807 663 PSMs (including tied matches due to isoleucine/leucine ambiguities or modification locations); and
- 734 522 spectra.

Again, PSMs were rescored and probabilities extracted via Percolator 3.02 by training its support vector machine on a subset of 250 000 PSMs for speed and memory efficiency. Since, currently, there is no way to run protein inference in Percolator separately, the corresponding options were set to be activated as well. The identifications with MSGF scores as well as Percolator scores and features were made available publicly for reproducibility.[30] To evaluate the computationally most demanding task for the Bayesian approaches, Fido was run in the single protein-level mode (no groups to exploit symmetries). EPIFANY calculates both levels simultaneously by default. Other parameters were left with their defaults in all tools, except for filtering out PSMs under 0.001 probability in ProteinProphet and PIA (to be comparable with the defaults in Fido and EPIFANY). Times and peak memory usage were measured with the Unix utility `time` on a two-socket Intel Xeon X5570 machine (i.e., 16 possible threads) with 64 GB of RAM.

Benchmarked Tools and Tool-Specific Adaptions. The tools tested represent a diverse set of algorithms. PIA 1.3.10 was chosen as the spectrum-level parsimony approach considering shared peptides. Percolator 3.02 represents aggregation-based approaches on unique peptides only. ProteinProphet (compiled from TPP 5.1) was included as a commonly used iterative and pseudo-probabilistic heuristic, which considers shared peptides as well. Lastly, Fido (in the version shipped with OpenMS) is the representative of

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

Bayesian methods with a very similar model to EPIFANY's but with a different inference procedure. The selection is also based on other recent evaluations of protein inference methods.[14,31]

Since PIA accepts the OpenMS' idXML format by default, the only change that was done was a renaming of the PSM score type name resulting from Percolator to OpenMS' posterior error probability type so that PIA accepts the scores and interprets them as error probabilities. Inference was performed with and without PIA's own FDR estimation. The main manuscript shows the results directly on Percolators' PEPs as they yield a fairer comparison.

For ProteinProphet, OpenMS' IDFileConverter was used to write the error probabilities from Percolator into a pepXML file that is compatible with ProteinProphet to make sure that all tools start from the same set of scores. The protein FDR estimation procedures of the tools were used whenever possible. For ProteinProphet, we used the same FDR estimation procedure as for EPIFANY with a concatenated target-decoy database and the equation $\widehat{\mathrm{FDR}} = \frac{(N_{\mathrm{decoy}} + 1)}{N_{\mathrm{target}}}$.[32] Groups are counted as decoy if they consist of only decoy proteins. Unless explicitly stated otherwise, FDRs were converted to *q*-values and are used synonymously throughout the text.

RESULTS AND DISCUSSION

EPIFANY Shows Improved Identification Performance

Identification performance of protein inference was benchmarked using the iPRG2016 data set, which was specifically designed to include 192/191 ground-truth proteins and a database with additional entrapment sequences. This allowed a comparison of tools at specific known entrapment protein FDRs with different cutoffs at the PSM and protein levels. Further, a more detailed investigation of the impact of unfiltered PSMs on a UPS2 data set with varying concentrations of known proteins was performed.

Unpooled Samples A/B with PSMs Filtered at 0.001 Posterior Probability. Results on the loosely filtered, unpooled experiments of the iPRG2016 study show that EPIFANY (with greedy group resolution enabled) yields the highest count of known true-positive proteins among the tested methods (Figure 3, top). On sample B, EPIFANY identifies 9.14% more true positives (TPs) at 5% FDR than the second-best method for this measure, PIA. Considering the low number of missing true positives to be identified, this increase is of even greater importance. On both samples, PIA and ProteinProphet do not perform well on just mildly filtered sets of PSMs. Additionally, a special case occurs for sample B, where ProteinProphet's reported proteins are first found at entrapment *q*-values higher than 5%, and therefore an empty bar is shown. The performance of ProteinProphet might be explained by the different pipeline[33] usually used to create its input. Furthermore, ProteinProphet performs a different and more aggressive protein grouping not only by aggregating indistinguishable groups but also by subsuming groups into more general protein ambiguity groups.

However, since this data set is limited in the number of known proteins that can be found (191 in sample B, 192 in A), the number of identifications at a certain cutoff does not indicate the full strength of our new method. Not only does it identify more known spike-ins, it also finds them at overall lower entrapment FDRs than most other methods and thus at a threshold where it is reporting fewer false positives. This is evident in the quickly rising curve of the receiver operating characteristic (ROC). On sample B, EPIFANY is covering the largest partial area under the curve of all tested tools (truncated at nine false positives equaling the maximum number to reach a 5% entrapment FDR; abbreviated pAUC). On the other sample (A), Percolator slightly outperforms EPIFANY in this regard. This pAUC measure is summarized in the lower part of Figure 3.

Together with Fido, EPIFANY is also the only tool reporting all correct proteins of sample B after all—though Fido finds all 191 present proteins at an entrapment FDR of 47% (beyond the cutoff in the figures). The other tools tested on loosely filtered sample B (even without any protein FDR cutoff) completely ignore or filter some true positives, most likely due to missing/insufficient evidence, e.g., missing unique peptides in Percolator (which reaches 187 true proteins on B at its maximum FDR of 56%). On sample A, no method is able to identify all true positives at any protein-level cutoff.

When applied to data without stringent PSM FDR filtering, PIA achieves only consistently moderate pAUCs (around 70−75%). Therefore, we evaluated all tools again with the often-suggested precutoff of 0.01 (in the Supplementary Results also with 0.05) PSM FDR.[34]

Unpooled Samples A/B with PSMs Filtered at 1% FDR. Different methods are affected by prefiltering on the PSM level in various degrees. While Fido struggles to perform well on 1% cutoffs on both unpooled samples, PIA now slightly outperforms EPIFANY (Figure 3) on the pAUC measure of sample B. EPIFANY still finds all known true positives at the lowest entrapment FDR (0.035 vs 0.04 in PIA) and achieves the best agreement between reported FDR and entrapment FDR (Supplementary Figure S1). Supplementary Table S5 suggests that EPIFANY's parameter optimization actually missed configurations with almost perfect pAUCs (due to balancing calibration and not having information about the labels of entrapment proteins).

On sample A with a strict 1% PSM FDR cutoff however, it is again EPIFANY (almost tied with Percolator) that shows the overall steepest ROC curve. PIA's performance first recovers at around 2% entrapment FDR and finds, together with EPIFANY, most (181) of the 192 known true positives at 5% protein entrapment FDR (Figure 3). Percolator behaves robust to additional data from low-scoring PSMs but misses out on several true positives on sample B in both strictly and loosely filtered test cases. Although ProteinProphet's number of identified true positives (TPs) benefits significantly from prefiltering, its pAUC does not.

Identification Performance at 1% Protein Entrapment FDR. Since a cutoff of 1% FDR on the protein level is often suggested for reporting, we compared performance at this stricter protein-level cutoff as well. However, we want to emphasize that due to the small number of known true positives, an evaluation of 1% entrapment FDR is now defined based on the ranking of the first two false-positive entrapment proteins only. This results in both performance measures (pAUC and number of TP) being very similar and overall less robust (Supplementary Figure S21).

The general trend that can be seen at a 5% protein entrapment FDR is also valid at this stricter FDR: PIA (180 TP on sample B; 1% PSM FDR) and Percolator (175 TP on sample A; both PSM-level cutoffs) can reach top identification

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

performance but also can fall behind significantly, as can be seen on some of the tested configurations (e.g., max. 126 TP for PIA on A and max. 75 TP for Percolator on B). Fido is robust across samples without a strict PSM cutoff (min. 150 TP) but does not perform well at strict PSM-level filtering. Even more pronounced now is the reduced performance of ProteinProphet (max. 55 TP in any combination). EPIFANY shows robust performance across all tested combinations of the sample and PSM-level filtering and identifies 147 true-positive proteins on sample B and 164 on sample A (independent of any tested PSM filtering); however, its full strength here is better reflected in the pAUCs at higher, more stable cutoffs, which implicitly include lower ones (Figure 3, bottom), or in the full ROC curves shown in the Supplementary Results.

Potential Advantages of Unfiltered PSM Input for Protein Inference. While a filtering on the PSM level before inference indeed seems beneficial to identification performance for at least PIA and ProteinProphet on the subset of known proteins, we argue that in general with Bayesian methods (as with most machine learning approaches), inference on as much data as possible may have several advantages as well.

However, specific examples can be shown more easily on smaller ground-truth data sets such as the universal proteomics standard 2 (UPS2), where it can be observed that with full information available, EPIFANY is able to improve the ranking of low-concentrated true proteins (e.g., lysozyme C) compared to some decoys (Figure 4). With milder filtering at the PSM level, inference methods can rely on additional data from low-scoring PSMs. Decoys that acquired one or few high-scoring PSMs can then more easily be penalized based on the number of bad PSMs that simultaneously match them. EPIFANY on unfiltered PSMs therefore achieves the highest true-positive count at almost all protein FDRs across all analyzed cutoffs and methods (Figure 4). With strict FDR cutoffs on the PSM level, even medium-scoring peptides of the true proteins are often not considered since their PSMs do not pass the threshold. Therefore, with stricter cutoffs, much of each method's performance depends solely on the ordering among the top-scoring peptides of each protein as established by the PSM rescoring procedures (here Percolator).

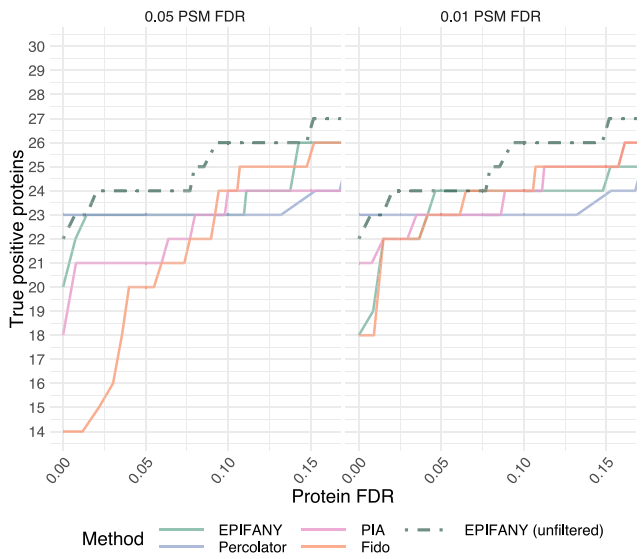


Figure 4. Number of true-positive UPS2 proteins (max. 48) plotted against the FDR at the protein level. No protein groups were present. The subplots from left to right show the results of protein inference on PSMs filtered at increasingly strict cutoffs. Left: PSMs were filtered at 0.05 PSM target-decoy FDR. Right: PSMs were filtered at 0.01 PSM target-decoy FDR. The results from a run of EPIFANY on completely unfiltered data were overlaid as a dashed line over both of the subplots. Note that the *y*-axis was cut at the minimal number of true positives observed.

Another advantage of less strict cutoffs is that previously indistinguishable proteins (above the desired protein FDR threshold) may become distinguishable. On iPRG2016 sample B for example, EPIFANY on loosely filtered PSMs is able to resolve 3 additional indistinguishable protein groups (of previously 16 on filtered data).

In any case, good performance on both strictly filtered and almost unfiltered data shows the increased robustness of our method. Supplementary Figures S10 and S11 also suggest that additional PSMs from other sources (in this example wider precursor search windows and additional variable modifications) can be more easily tolerated or even used advantageously in a Bayesian setting (see the curves of Fido and EPIFANY). With those settings, PIA struggles in the very low FDR ranges even with strict cutoffs.

FDR Estimates of EPIFANY Are More Realistic than Other Estimates

Before the introduction of a regularized model and greedy resolution as implemented in EPIFANY, Bayesian methods were shown to report overly optimistic FDRs in the case of data sets like the one tested here.[14] This is due to the fact that on iPRG2016 samples A and B although only a small number of proteins are known to be present in the sample, the spectra were searched against an additional equal number of absent proteins designed to share peptides with the present ones along with an even bigger number of 1000 absent random entrapment proteins. Unregularized models using standard (sum-product) inference like Fido then would conservatively assign probabilities far from 0 to proteins with similar or no unique evidence that share one or more peptides with truly present proteins. Regularized max-product inference in EPIFANY now makes the assumption that it is less likely for peptides to be generated by many proteins and preferentially distributes the evidence among proteins with the highest unique evidence. Greedy resolution additionally makes a definite, unprobabilistic choice among those proteins based on their posterior probability. By postponing this decision until the end of an identification pipeline, however, reliable uncertainty estimates are available up to the very last step. Even compared to conservative methods ignoring shared peptides for FDR estimation (e.g., Percolator), FDRs reported by EPIFANY are often closer to the known entrapment FDRs (also across different PSM cutoffs on A and B). The differences between reported FDR and observed entrapment FDR can be seen, e.g., for the barely filtered sample B in Figure 5, which shows up to a 15% cutoff (since higher cutoffs are usually not of interest). The cutoff was set this high to generate more robust measures of calibration (by being able to include more point estimates for each method). For sample A, the increase in calibration compared to Percolator is seen mainly in the higher FDRs, but it calibrates as well or better than PIA with simultaneously improved identification performance. Additional pseudo-ROC curves and calibration plots for unpooled samples A and B, different cutoffs, and the full FDR ranges can be found in Supplementary Figures S1−S6.

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

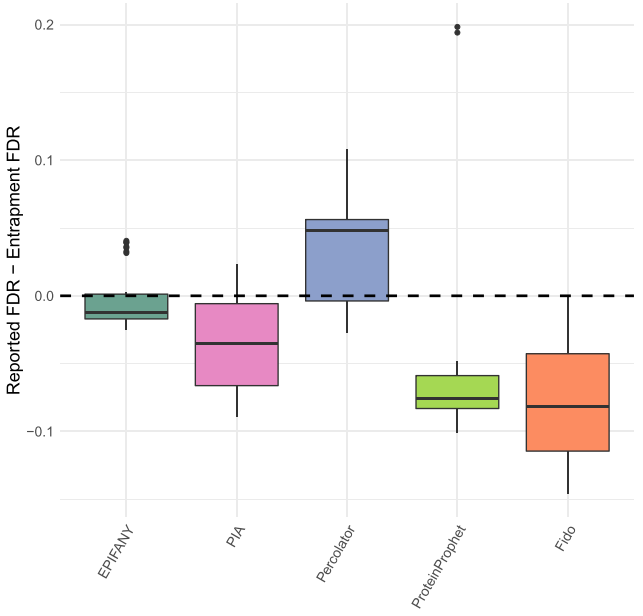


Figure 5. Absolute deviations from reported FDR from different tools and settings to the true entrapment FDR (on the subset of proteins with known presence/absence) on the iPRG2016 data set (experiment B with a 0.001 probability cutoff at the PSM level). The boxplots aggregate data points for each method until a 15% entrapment FDR. Lower deviations imply better calibration, with perfect calibration being indicated by the dashed horizontal line. Data above the horizontal line signify conservative estimates, while data below the line signify overly optimistic estimates.

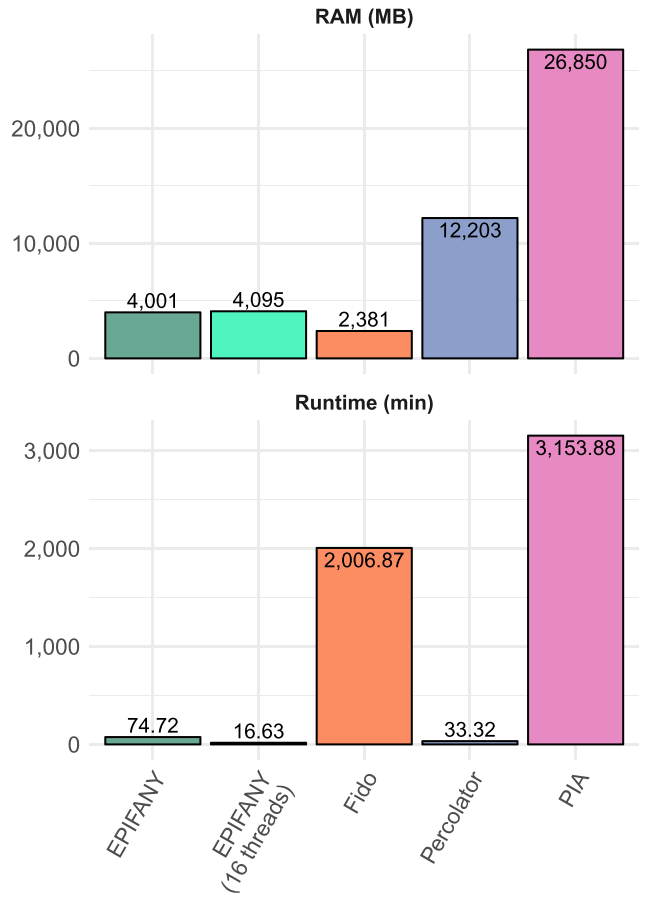


Figure 6. Memory usage in megabytes (upper) and runtimes in minutes (lower) on the large-scale human data set filtered at 0.001 PSM probability. ProteinProphet did not terminate within a week of runtime and was thus not included in the figure. EPIFANY uses one thread by default. Another bar represents EPIFANY with multithreading on 16 threads enabled via the thread parameter ("EPIFANY (16 threads)".).

Additionally, with the optional greedy postprocessing step turned off, regularized inference in EPIFANY on its own offers a balance between better calibration (at least in low entrapment FDR ranges of up to almost 2%) on the one extreme (A or B) and identification performance on the other (A + B), where it recovers the remaining known true proteins between entrapment FDR ranges of 3 and 8% (compare Supplementary Figures S1−S6 vs S7−S9). This is due to the fact that proteins, which were identified through shared peptides alone, are then assigned a lower probability than the proteins with additional unique evidence but still outscore many of the decoy or false entrapment proteins. Heuristic approaches like greedy, parsimonious, and "unique peptide" methods cannot model the actually present sharedness in such data sets. "Report all" methods (e.g., through the corresponding option in PIA) perform well on that extreme but are not an alternative for data sets deviating from this very optimistic assumption.

Scalable Algorithms Allow Application to Large-Scale Data Sets with Vastly Disparate Discoveries

A common challenge with inference on generative Bayesian models was their scalability due to the speed of the method, which is inherently related to the complexity of the underlying model. The efficiency of EPIFANY enables full Bayesian inference on problem sizes that were previously intractable given the used model. On the loosely filtered human data set searched against the full UniProt database, even non-Bayesian approaches struggle with the high connectivity in the resulting protein−peptide graph. This is evident in Figure 6, which shows the runtimes and memory consumption of the different tools. Fido took longer than a day of runtime (33.5 h), ProteinProphet did not finish after a week, and PIA required more than 2 days of processing (without considering compilation of the intermediate graph format used by PIA). While multithreaded EPIFANY (16 min) is even faster than (subset-trained) Percolator (33 min), it should be noted that most of Percolator's runtime consists of peptide rescoring and internal training of a support vector machine model. The same argument holds for memory usage. Although running Percolator just for protein inference is, to our knowledge, currently not possible, it is likely that the actual runtime on that data set will be in the single-digit minutes. EPIFANY and Fido were both run with parameter optimization enabled. Runtimes for EPIFANY are roughly linear in the number of parameter sets tested although there are sets that lead to more oscillation and therefore may take longer than others. Concerning multithreading, none of the methods except for EPIFANY has the option to pass the number of threads to be used. However, it seems from the log that PIA automatically distributes work across all of them. Percolator supports automatic multithreading across three threads. EPIFANY was run once with 1 thread and once with all 16 threads enabled. A full 16-fold speedup is yet far from being achieved due to the different sizes in connected components processed by each thread and the synchronization required during parameter

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

evaluation. For completeness, Fido on the easier problem of "group inference only" took 5.5 h less (28 h).

Except for the lower theoretical complexity of the algorithm compared to methods like Fido with a similar model, it is difficult to make absolute statements about the scalability of its runtime due to its dependency on multiple factors (e.g., the size of connected components, their connectivity, and convergence; Supplementary Figure S20).

In general, it starts with a bigger overhead than all of the other methods due to the allocation of the tables and messages for every parameter set. Even very small components where posteriors could be quickly calculated naively are currently processed with the general LBP algorithm. This leads to disadvantages on small- and/or low-complexity data sets (e.g., on filtered iPRG2016 A + B, EPIFANY needed 86 s instead of 2.5−36 s like the other methods on a MacBook Pro 2014; Supplementary Table S2). EPIFANY starts to gain from the theoretical advancements on medium−complex data as in the case of loosely filtered PSMs from the A + B sample. Therefore, it already is faster (183 s) than PIA (302 s) and ProteinProphet (498 s). Fido still benefits from the very efficient implementation on small connected components and completes in 80 s. For comparison, the full Percolator run on all spectra including inference took 312 s (see also Supplementary Table S3). Figure 7 additionally shows the scalability of our method based on different posterior probability filter thresholds on the large-scale data set. A linear trend is observed here but cannot be guaranteed in general as explained above.

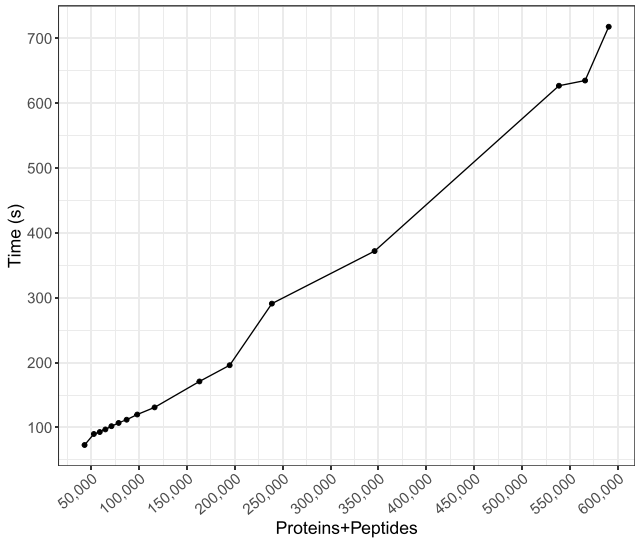


Figure 7. Runtime in seconds of EPIFANY (16 threads) for different PSM-level thresholds on the human data set. The total resulting number of potential peptides and proteins was used as the *x*-axis.

It is expected that in data sets with more spectra than our human data set, the complexity and size of separate connected components will at some point reach saturation (on the same protein database) such that the growth in runtime of inference methods decreases. More and more PSMs will start hitting the same peptides and form less new paths between previously independent proteins until all peptides are observed.

The efficiency of EPIFANY should allow an application of Bayesian methods also on databases with a higher amount of ambiguities such as proteogenomics or metaproteomics databases as well as in the context of post-translationally modified proteoforms.

We also emphasize that an application of a Bayesian method incorporating results from shared peptides may lead to different discoveries compared to methods currently used on big data sets that ignore this complication.[7,35] On the largest of the tested data sets for example, Percolator considers only 26 182 of the 51 576 potential target proteins (with at least one PSM above the 0.001 probability cutoff as used in EPIFANY).

Implementation Grants Flexibility in Input and Output Information

In addition to overall increased performance, EPIFANY's general Bayesian framework also permits the inclusion of auxiliary information (besides PSM data) in a convenient manner. The new implementation allows the usage of arbitrary protein priors for any protein that allows integration of information from complementary RNA-seq experiments, which has been shown to improve protein identification.[36] Due to its modular graph structure, it is also easy to add additional probabilistic evidence on the peptide level in the future. Since the factor graph includes both single proteins and indistinguishable protein groups, both results can be output simultaneously. It is not possible to report single protein-level probabilities of proteins in indistinguishable groups in either PIA—its "report all" option allows that but skips parsimonious inference altogether—or Percolator. Another novelty is the reporting of updated peptide-level posteriors that can be used to rescore and improve the FDR peptide level, yielding increased target-decoy classification performance by up to 6% (measured on the full target-decoy-based AUC). This effect arises from the fact that the information from sibling peptide[9] in the graph was now propagated and incorporated into every peptide's posterior.

Limitations

Since the algorithm's grid search explores multiple parametrizations of the model, there might be settings in which, at some point during LBP, contradicting messages are generated that have disagreeing beliefs about the probability of a protein or peptide (e.g., from evidence of different parts of the graph). This can often be solved by a stepwise increase in damping in later iterations (see also Supplementary Methods Section 25.3). In extreme cases, however, it can lead to interruptions in the inference on that part of the graph (since, e.g., zero probabilities cannot be recovered). In the latter case, we evaluate the (failing) parameter set by using the prior probabilities of the affected proteins (assuming no knowledge could be gained for those proteins). In the case of nonconvergence, due to too many iterations, the current beliefs for a protein are used. However, those cases are rare and usually seen in very large components caused by extreme parameter sets and evidence only.

Also, due to the fact that the parameter estimation is based on target-decoy annotations, our method is affected by the composition of the decoy database. However, as shown in Supplementary Figure S3, different random shuffles of the iPRG2016 database resulted in comparable identification performances with a median partial AUC of 92% and a median absolute deviation of one percent point.

Furthermore, group-level inference is still based on experimentally indistinguishable proteins, which hinders reproducibility of the specific groupings across multiple runs.[6] If other types of groupings are to be performed, this

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

has to be reflected in the input before running inference (e.g., by merging protein IDs beforehand).

Availability

EPIFANY runs on all major platforms (Windows, Linux, OSX). It is available under an open-source license (BSD three-clause) at https://openms.org/epifany. This website also contains demo data, a manual, binary installers for all platforms, and links to the source code repository. EPIFANY relies on the Evergreen inference library released by O. Serang under an MIT license, which can be found under https://bitbucket.org/orserang/evergreenforest.[37]

CONCLUSIONS AND OUTLOOK

With EPIFANY, we present a new approach to efficient Bayesian protein inference in proteomics that combines excellent inference quality with good runtimes. We also showed that inference on unfiltered data can yield improved identification rates at widely accepted protein FDR thresholds. The underlying method certainly can be improved upon. As mentioned in the Limitations section, the results depend on the convergence of the algorithm. Convergence is generally affected by the current parametrization of the model as well as the connectivity and evidence in the graph; to which degree however still has to be investigated. In the case of suboptimal results on a connected component, the algorithm could, in the future, try different message-scheduling types or resort to heuristics. Additionally, the current experimental options of peptide rescoring and user-defined priors are worth further research. Incorporation of additional evidence especially from the MS1 level, replicates, or multiple PSMs is a viable extension, too; however, initial tests with precursor mass and retention time deviations yielded noisy, generally disappointing results so far. Regularization of protein groups could be improved by facilitating user-defined protein groups (e.g., by gene or theoretical digest). Reintroduction of proteotypicity with discretized α parameters per peptide instead of a single α per data set could help in the discrimination of otherwise indistinguishable proteins. In general, parameter estimation via grid search is a very time-consuming part of the algorithm, and although different parameter sets can be distributed across machines on even larger-scale data, learning speed might be improved by learning on a subset of the graph or completely circumvented by including the parameters as hyperparameters into the probabilistic model. Concerning the impact on quantification, it should be noted that optionally, greedily assigned shared peptides could now be used in quantifying tools as well. While this procedure boosts the number of peptides to be used for quantification, it should be handled with care since those quantities might be distorted due to actual contributions from the greedily removed proteins. It is then even more important to use quantification approaches that filter peptides with heavily disagreeing profiles.[38] Another promising approach would be to incorporate peptide quantities into the model, e.g., in a way that they influence the degree of regularization of each peptide. After correction for their response factors, shared peptides that show significantly higher intensities than the unique peptides of a protein are more likely to have been produced by multiple parent proteins. This might also help estimate when a complete greedy approach on a data set is too conservative.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jproteome.9b00566.

extendedGridAUCsGreedyB001.xlsx: Supplementary Table S5 with actual entrapment AUCs of an extended grid search of our parameters (XLXS)

scripts_and_workflows.zip: Supplementary Data 1 with a collection of scripts and workflows used in the preparation of this manuscript (ZIP)

Tables S1 (Comet) and S2 (MSGFPlus), extracted search engine-specific features for Percolator; Figures S1−S11, more detailed information on the iPRG2016 results for different samples and PSM cutoffs; Figures S12−S16, guided example of the message passing algorithm; Figures S17 and S18, visualization of potentially oscillating parts of a graph; Figure S19, effect of differently shuffled decoy databases on EPIFANY's parameter estimation; Tables S3 and S4, runtimes of the different methods on the iPRG2016 A + B sample for a measure of scalability; Figure S20, runtimes versus sizes of connected components; Figure S21, alternative results on the 1% protein entrapment FDR level; detailed descriptions of the model, algorithm, convergence, and parameter estimation; summarization of Figures S1−S11 (PDF)

AUTHOR INFORMATION

Corresponding Authors

Julianus Pfeuffer − *Applied Bioinformatics, Department of Computer Science and Institute for Bioinformatics and Medical Informatics, University of Tübingen, 72076 Tübingen, Germany; Algorithmic Bioinformatics, Department of Bioinformatics, Freie Universität Berlin, 14195 Berlin, Germany;* orcid.org/0000-0001-8948-9209; Email: pfeuffer@informatik.uni-tuebingen.de

Oliver Serang − *Department of Computer Science, University of Montana, Missoula, Montana 59812, United States;* orcid.org/0000-0003-1245-7051; Email: oliver.serang@umontana.edu

Oliver Kohlbacher − *Biomolecular Interactions, Max Planck Institute for Developmental Biology, 72076 Tübingen, Germany; Applied Bioinformatics, Department of Computer Science, Institute for Bioinformatics and Medical Informatics, and Quantitative Biology Center, University of Tübingen, 72076 Tübingen, Germany; Institute for Translational Bioinformatics, University Hospital Tübingen, 72076 Tübingen, Germany;* orcid.org/0000-0003-1739-4598; Email: oliver.kohlbacher@uni-tuebingen.de

Authors

Timo Sachsenberg − *Applied Bioinformatics, Department of Computer Science and Institute for Bioinformatics and Medical Informatics, University of Tübingen, 72076 Tübingen, Germany*

Tjeerd M. H. Dijkstra − *Biomolecular Interactions, Max Planck Institute for Developmental Biology, 72076 Tübingen, Germany*

Knut Reinert − *Algorithmic Bioinformatics, Department of Bioinformatics, Freie Universität Berlin, 14195 Berlin, Germany*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jproteome.9b00566

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

Author Contributions

O.S., K.R., and O.K. designed the experiments. O.S. and J.P. wrote the implementation of the method. J.P. performed data analysis and wrote the manuscript. T.S. wrote auxiliary tools and functions for data preprocessing and export. J.P., O.S., O.K., and T.M.H.D. interpreted the results. All authors helped in drafting and read as well as approved the final manuscript.

Funding

The authors are thankful for the support of the funding agencies and their following grants: J.P., T.S., and O.K. were funded by the Bundesministerium für Bildung und Forschung (BMBF) under award number 031A535A (German Network for Bioinformatics Infrastructure). O.S. received funding from the University of Montana small grant 325476 (UGP 2018: Development of Probabilistic Cardinal Models) and the National Institute of General Medical Sciences of the National Institutes of Health under Award Number P20GM103546. This material is based on work supported by the National Science Foundation under grant no. 1845465. T.M.H.D. was funded by the European Union through the EU-H2020 ICT-644727 COGIMON project. Funding for O.K. was also provided by the European Union through EU project EPIC-XS grant number 823839 (H2020-INFRAIA).

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The authors thank Xiao Liang for the fruitful discussions and ideas on this and related topics as well as implementations in earlier stages of this work and for the happy atmosphere she brought to the office. The authors also thank Julian Uszkoreit for the generation of the input graph file on the large data set for his tool PIA and his help with the settings. Additionally, they thank Prof. Lothar Jänsch and his group for the permission to use their data for benchmarking.

■ REFERENCES

(1) Zhang, Y.; Fonslow, B. R.; Shan, B.; Baek, M.-C.; Yates, J. R. Protein Analysis by Shotgun/Bottom-up Proteomics. *Chem. Rev.* **2013**, *113*, 2343−2394.
(2) Nesvizhskii, A. I.; Aebersold, R. Interpretation of shotgun proteomic data: the protein inference problem. *Mol. Cell. Proteomics* **2005**, *4*, 1419−1440.
(3) Serang, O.; Noble, W. A review of statistical methods for protein identification using tandem mass spectrometry. *Stat. interface* **2012**, *5*, 3−20.
(4) Uszkoreit, J.; Maerkens, A.; Perez-Riverol, Y.; Meyer, H. E.; Marcus, K.; Stephan, C.; Kohlbacher, O.; Eisenacher, M. PIA: An Intuitive Protein Inference Engine with a Web-Based User Interface. *J. Proteome Res.* **2015**, *14*, 2988−2997.
(5) Cottrell, J. S. Protein identification using MS/MS data. *J. Proteomics* **2011**, *74*, 1842−1851.
(6) Serang, O.; Moruz, L.; Hoopmann, M. R.; Käll, L. Recognizing uncertainty increases robustness and reproducibility of mass spectrometry-based protein inferences. *J. Proteome Res.* **2012**, *11*, 5586−5591.
(7) The, M.; MacCoss, M. J.; Noble, W. S.; Käll, L. Fast and Accurate Protein False Discovery Rates on Large-Scale Proteomics Data Sets with Percolator 3.0. *J. Am. Soc. Mass Spectrom.* **2016**, *27*, 1719−1727.
(8) Käll, L.; Canterbury, J. D.; Weston, J.; Noble, W. S.; MacCoss, M. J. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nat. Methods* **2007**, *4*, 923−925.
(9) Nesvizhskii, A. I.; Keller, A.; Kolker, E.; Aebersold, R. A Statistical Model for Identifying Proteins by Tandem Mass Spectrometry. *Anal. Chem.* **2003**, *75*, 4646−4658.
(10) Serang, O.; MacCoss, M. J.; Noble, W. S. Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data. *J. Proteome Res.* **2010**, *9*, 5346−5357.
(11) Li, Y. F.; Arnold, R. J.; Li, Y.; Radivojac, P.; Sheng, Q.; Tang, H. A bayesian approach to protein inference problem in shotgun proteomics. *J. Comput. Biol.* **2009**, *16*, 1183−1193.
(12) Gerster, S.; Qeli, E.; Ahrens, C. H.; Bühlmann, P. Protein and gene model inference based on statistical modeling in k-partite graphs. *Proc. Natl. Acad. Sci. U.S.A.* **2010**, *107*, 12101−12106.
(13) Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann Publishers Inc.: San Mateo, CA, 1988; pp 143−237.
(14) The, M.; Edfors, F.; Perez-Riverol, Y.; Payne, S. H.; Hoopmann, M. R.; Palmblad, M.; Forsström, B.; Käll, L. A Protein Standard That Emulates Homology for the Characterization of Protein Inference Algorithms. *J. Proteome Res.* **2018**, *17*, 1879−1886.
(15) Berrou, C.; Glavieux, A.; Thitimajshima, P. *Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-codes*, 1. Proceedings of ICC '93 - IEEE International Conference on Communications, 1993; pp 1064−1070.
(16) Murphy, K. P.; Weiss, Y.; Jordan, M. I. *Loopy Belief Propagation for Approximate Inference: An Empirical Study*, Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, 1999; pp 467−475.
(17) Pearl, J. *Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach*, Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, Aug 18−20 1982; pp 133−136.
(18) Serang, O. The probabilistic convolution tree: efficient exact Bayesian inference for faster LC-MS/MS protein inference. *PLoS One* **2014**, *9*, No. e91507.
(19) Elidan, G.; McGraw, I.; Koller, D. *Residual Belief Propagation: Informed Scheduling for Asynchronous Message Passing*, Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence, 2006; pp 165−173.
(20) Jensen, F. V.; Lauritzen, S. L.; Olesen, K. G. Bayesian updating in causal probabilistic networks by local computations. *Comput. Stat. Q.* **1990**, *4*, 269−282.
(21) OpenMP Architecture Review Board. OpenMP Application Program Interface Version 2.0.2002, 2002. https://www.openmp.org/wp-content/uploads/cspec20.pdf.
(22) Perez-Riverol, Y.; et al. The PRIDE database and related tools and resources in 2019: improving support for quantification data. *Nucleic Acids Res.* **2019**, *47*, D442−D450.
(23) Chambers, M. C.; et al. A cross-platform toolkit for mass spectrometry and proteomics. *Nat. Biotechnol.* **2012**, *30*, 918−920.
(24) Sturm, M.; Bertsch, A.; Gröpl, C.; Hildebrandt, A.; Hussong, R.; Lange, E.; Pfeifer, N.; Schulz-Trieglaff, O.; Zerck, A.; Reinert, K.; Kohlbacher, O. OpenMS - An open-source software framework for mass spectrometry. *BMC Bioinf.* **2008**, *9*, 163.
(25) Röst, H. L.; et al. OpenMS: A flexible open-source software platform for mass spectrometry data analysis. *Nat. Methods* **2016**, *13*, 741−748.
(26) Eng, J. K.; Jahan, T. A.; Hoopmann, M. R. Comet: An open-source MS/MS sequence database search tool. *Proteomics* **2013**, *13*, 22−24.
(27) Pfeuffer, J. UPS2 standard in solution converted to mzML with entrapment and decoy appended protein databases, 2019. https://doi.org/10.5281/zenodo.3531639.
(28) Kim, S.; Gupta, N.; Pevzner, P. A. Spectral probabilities and generating functions of tandem mass spectra: a strike against decoy databases. *J. Proteome Res.* **2008**, *7*, 3354−3363.
(29) Bairoch, A.; Apweiler, R. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.* **2000**, *28*, 45−48.

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072

(30) Pfeuffer, J. PSM Probabilities from Percolator on merged IDs from four human samples, 2020. https://doi.org/10.5281/zenodo.3531682.
(31) Audain, E.; Uszkoreit, J.; Sachsenberg, T.; Pfeuffer, J.; Liang, X.; Hermjakob, H.; Sanchez, A.; Eisenacher, M.; Reinert, K.; Tabb, D. L.; Kohlbacher, O.; Perez-Riverol, Y. In-depth analysis of protein inference algorithms using multiple search engines and well-defined metrics. *J. Proteomics* **2017**, *150*, 170−182.
(32) Levitsky, L. I.; Ivanov, M. V.; Lobas, A. A.; Gorshkov, M. V. Unbiased False Discovery Rate Estimation for Shotgun Proteomics Based on the Target-Decoy Approach. *J. Proteome Res.* **2017**, *16*, 393−397.
(33) Deutsch, E. W.; Mendoza, L.; Shteynberg, D.; Slagel, J.; Sun, Z.; Moritz, R. L. Trans-Proteomic Pipeline, a standardized data processing pipeline for large-scale reproducible proteomics informatics. *Proteomics* **2015**, *9*, 745−754.
(34) Uszkoreit, J.; Perez-Riverol, Y.; Eggers, B.; Marcus, K.; Eisenacher, M. Protein Inference Using PIA Workflows and PSI Standard File Formats. *J. Proteome Res.* **2019**, *18*, 741−747.
(35) Savitski, M. M.; Wilhelm, M.; Hahne, H.; Kuster, B.; Bantscheff, M. A Scalable Approach for Protein False Discovery Rate Estimation in Large Proteomic Data Sets. *Mol. Cell. Proteomics* **2015**, *14*, 2394−2404.
(36) Proffitt, J. M.; Glenn, J.; Cesnik, A. J.; Jadhav, A.; Shortreed, M. R.; Smith, L. M.; Kavanagh, K.; Cox, L. A.; Olivier, M. Proteomics in non-human primates: utilizing RNA-Seq data to improve protein identification by mass spectrometry in vervet monkeys. *BMC Genomics* **2017**, *18*, No. 877.
(37) Serang, O. The p-convolution forest: a method for solving graphical models with additive probabilistic equations, arXiv:1708.06448. arXiv.org e-Print archive, 2017. https://arxiv.org/abs/1708.06448.
(38) Zhang, B.; Pirmoradian, M.; Zubarev, R.; Käll, L. Covariation of Peptide Abundances Accurately Reflects Protein Concentration Differences. *Mol. Cell. Proteomics* **2017**, *16*, 936−948.

https://dx.doi.org/10.1021/acs.jproteome.9b00566
J. Proteome Res. 2020, 19, 1060−1072