BIOINFORMATICS ORIGINAL PAPER

Vol. 28 no. 22 2012, pages 2956–2962
doi:10.1093/bioinformatics/bts540

Data and text mining

Advance Access publication September 6, 2012

A linear programming model for protein inference problem in shotgun proteomics

Ting Huang and Zengyou He*

School of Software, Dalian University of Technology, Dalian 116621, China

Associate Editor: Jonathan Wren

ABSTRACT

Motivation: Assembling peptides identified from tandem mass spectra into a list of proteins, referred to as protein inference, is an important issue in shotgun proteomics. The objective of protein inference is to find a subset of proteins that are truly present in the sample. Although many methods have been proposed for protein inference, several issues such as peptide degeneracy still remain unsolved.

Results: In this article, we present a linear programming model for protein inference. In this model, we use a transformation of the joint probability that each peptide/protein pair is present in the sample as the variable. Then, both the peptide probability and protein probability can be expressed as a formula in terms of the linear combination of these variables. Based on this simple fact, the protein inference problem is formulated as an optimization problem: minimize the number of proteins with non-zero probabilities under the constraint that the difference between the calculated peptide probability and the peptide probability generated from peptide identification algorithms should be less than some threshold. This model addresses the peptide degeneracy issue by forcing some joint probability variables involving degenerate peptides to be zero in a rigorous manner. The corresponding inference algorithm is named as ProteinLP. We test the performance of ProteinLP on six datasets. Experimental results show that our method is competitive with the state-of-the-art protein inference algorithms.

Availability: The source code of our algorithm is available at: https://sourceforge.net/projects/prolp/.

Contact: zyhe@dlut.edu.cn

Supplementary information: Supplementary data are available at *Bioinformatics* Online.

Received on March 28, 2012; revised on July 29, 2012; accepted on August 28, 2012

1 INTRODUCTION

Protein identification using tandem mass spectrometry (MS/MS) is the most widely used tool for detecting proteins from biological samples. In a typical shotgun proteomics experiment (Fig. 1), proteins in a sample are first digested into peptides, and the resulting mixture of peptides is subjected to mass spectrometry to generate tandem mass spectra. After spectra acquisition, the peptide that generates each spectrum is identified with peptide identification algorithms. From these putative peptide identifications, the proteins that are present in the sample are detected with protein inference algorithms.

*To whom correspondence should be addressed.

Computationally, the input for the protein inference problem is a bipartite graph: the left is a set of identified peptides and the right is the set of candidate proteins that have at least one constituent peptide. The inference problem considered here is to find a subset of proteins that are truly present in the sample. However, such protein inference problem is only partially solved since several technical challenges still remain unconquered. One of the most challenging problems is the peptide degeneracy issue, which arises when a single peptide can be mapped to multiple proteins. The performance of protein inference algorithms mainly depends on our capability of assigning these degenerate/shared peptides to proteins that really generate them.

To date, there are already many protein inference algorithms available in the literature (Bern and Goldberg, 2008; Feng *et al*., 2007; Gerster *et al*., 2010; Grobei *et al*., 2009; He *et al*., 2011; Kearney *et al*., 2008; Li *et al*., 2009a, b, 2010; Lu *et al*., 2008; Ma *et al*., 2009; Moore *et al*., 2002; Nesvizhskii *et al*., 2003; Price *et al*., 2007; Qeli and Ahrens, 2010; Ramakrishnan *et al*., 2009a, b; Sadygov *et al*., 2004; Searle, 2010; Serang *et al*., 2010; Shen *et al*., 2008; Slotta *et al*., 2010; Spivak *et al*., 2012; Tabb *et al*., 2002; Weatherly *et al*., 2005; Yang *et al*., 2004; Zhang *et al*., 2007). The reader can refer to a recent survey (Huang *et al*., 2012) for details. Here, we shall discuss briefly how these methods tackle the peptide degeneracy issue and present our research motivation.

Existing protein inference algorithms solve the peptide degeneracy problem in quite different ways. Generally, they fall into two categories, as listed in the subsequent sections.

Inference algorithms in the first category solve the peptide degeneracy problem with some simple rules or assumptions in an implicit manner. One typical example is the widely used two-peptide rule, which regards all candidate proteins that have at least two matching peptides as true positives (TPs). The underlying assumption is that degenerate peptides should belong to all proteins that they can match. In contrast, IDPicker (Ma *et al*., 2009; Zhang *et al*., 2007) formulates the protein inference problem as a set covering problem and solves it with a greedy algorithm. In the greedy selection procedure, proteins that can match the maximal number of uncovered peptides are selected in an iterative manner. The underlying assumption is that each degenerate peptide should be assigned to one protein only.

Inference algorithms in the second category treat the peptide degeneracy issue explicitly in terms of conditional probability. Briefly, they either model the conditional probability of one protein being present given a peptide or model the conditional

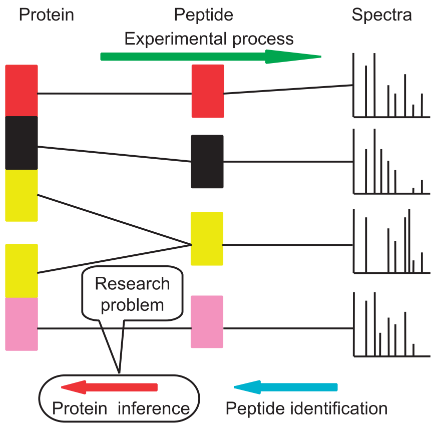

Fig. 1. Protein identification using mass spectrometry in shotgun proteomics. In the experimental process (from left to right), proteins are digested into peptides, which are then subjected to mass spectrometry to produce MS/MS spectra. In the data analysis process (from right to left), there are two major computational problems: peptide identification and protein inference. This article focuses on developing effective algorithms for protein inference

probability of one peptide being present given a protein. ProteinProphet (Nesvizhskii *et al.*, 2003), one of the most widely used protein inference methods, learns 'degenerate peptide weight' using an EM-like algorithm. In fact, such degenerate peptide weight corresponds to the probability of one protein being present conditional on the presence of a given peptide. Alternatively, MSBayesPro (Li *et al.*, 2009b) utilizes the concept of peptide detectability, which is defined as the probability of detecting a peptide in a standard sample by a standard proteomics routine if its parent protein is expressed. Fido (Serang *et al.*, 2010) models the probability with which a sample peptide is generated from a protein containing it with a constant probability. HSM (Shen *et al.*, 2008) considers five types of mechanisms that a peptide can be generated by a protein, i.e. the conditional probability that one peptide is present has five possible values.

The attempts of treating the peptide degeneracy problem rigorously in the second category have obtained promising results; however, they still have some limitations.

First, ProteinProphet employs an EM-like iterative procedure to estimate protein probabilities. This method is described procedurally rather than derived from a well-defined optimization model. In contrast, MSBayesPro, HSM and Fido derive their models from clear, explicitly stated statistical assumptions. However, they formulate the protein inference problem as a combinatorial optimization problem. This means that they may generate different inference results from the same dataset when obtaining the optimal solution is too time-consuming.

Second, Fido and HSM use a very small set of parameters to approximate all possible values that the conditional probability can take. Such a simplification makes it possible to create efficient accompanying algorithms, but it may also limit the capability of achieving better inference performance. In contrast, there are no such limitations in ProteinProphet and MSBayesPro. Unfortunately, the conditional probabilities in ProteinProphet are calculated using a formula that has not been rigorously justified. The peptide detectability values in MSBayesPro are predicted using a complex model trained on other datasets.

Finally, existing methods involve many parameters that are not easy to specify. For example, Fido needs to have a grid search in order to find good values for its three parameters.

The aforementioned observations motivate our research. In this article, we take a step further toward completely solving the protein inference problem with particular emphasis on peptide degeneracy. To that end, we present a linear programming (LP) model for protein inference, which is built on two simple probability equations.

We first introduce the joint probability that both a protein and its constituent peptide are present in the sample. To obtain a linear model, we use a mathematical transformation of this joint probability as the variable. The marginal probability of a peptide being present can be expressed as a formula in terms of the linear combination of these variables. If we assume that the marginal probability of each identified peptide being present is known, the protein inference problem could be formulated as the following optimization problem: 'minimize the number of proteins of non-zero probabilities while the calculated peptide probability should be as close to its known value as possible'. We show that this optimization problem actually can be written as a LP problem, which has only one parameter that is easy to specify and has a clear interpretation. This new protein inference algorithm is named as ProteinLP. Experimental results on six datasets show that our ProteinLP algorithm is a competitive and complementary approach for protein inference.

The main contributions of the work described in this article can be summarized as follows:

- To our knowledge, our work is the first LP formulation for the protein inference problem. Our method guarantees to find the optimal solution.
- Instead of using conditional probability, our model is the first attempt of addressing the peptide degeneracy problem with the joint probability. It greatly simplifies the model without sacrificing the discrimination power.

The rest of this article is organized as follows. In Section 2, we describe our method in detail. Section 3 presents the experimental results and Section 4 concludes the article.

2 METHODS

Given m candidate proteins and n identified peptides, the protein inference problem can be formulated as an optimization problem: select a possibly small subset of candidate proteins that best 'explains' these peptides. Such an optimization problem can be formulated in quite different ways. In this section, we present a LP model for protein inference, which can be solved very quickly with standard LP solver.

We use a vector of indicator variables $(x_1, \ldots, x_j, \ldots, x_m)$ to denote the set of m candidate proteins and another indicator vector $(y_1, \ldots, y_i, \ldots, y_n)$ to denote the set of n identified peptides. In addition, we assume that we know the probability that each peptide is present in the sample, which is provided by peptide identification algorithms such as Mascot (Perkins *et al.*, 1999) or post-processing tools such as PeptideProphet (Keller *et al.*, 2002). The peptide probability vector is denoted by $(z_1, \ldots, z_i, \ldots, z_n)$. Notation and definitions used in this article are summarized in Table 1.

2.1 Model

Let $\Pr(y_i = 1)$ denote the probability that peptide i is present and $\Pr(x_j = 1)$ denote the probability that protein j is present in the sample. A peptide is present if at least one of its parent proteins is present:

$$\Pr(y_i = 1) = 1 - \prod_{j=1}^{m} \left[1 - \Pr(y_i = 1, x_j = 1)\right], \tag{1}$$

where $\Pr(y_i = 1, x_j = 1)$ denotes the joint probability that peptide i and protein j are both present in the sample.

Similarly, for each protein j, we have

$$\Pr(x_j = 1) = 1 - \prod_{i=1}^{n} \left[1 - \Pr(y_i = 1, x_j = 1)\right]. \tag{2}$$

Through the logarithmic transformation, we convert the product relation in Equations (1) and (2) into the sum relation so as to build a LP model:

$$\ln[1 - \Pr(y_i = 1)] = \sum_{j=1}^{m} \ln\left[1 - \Pr(y_i = 1, x_j = 1)\right] \tag{3}$$

and

$$\ln\left[1 - \Pr(x_j = 1)\right] = \sum_{i=1}^{n} \ln\left[1 - \Pr(y_i = 1, x_j = 1)\right]. \tag{4}$$

Since the peptide probability and protein probability are not linear with respect to the joint probability, we use $p_{ij} = \ln\left[1 - \Pr(y_i = 1, x_j = 1)\right]$ instead of $\Pr(y_i = 1, x_j = 1)$ as the variable of our model. Then, both the peptide probability and protein probability can be expressed as a function of the linear combination of these variables. In other words, we can use the sum of p_{ij} to calculate both peptide probability and protein probability.

From aforementioned analysis, we can see that the joint probability of peptide and protein can serve as the bridge between peptide probability and protein probability. On the one hand, we can use the joint probability to explain the known peptide probability. On the other hand, we can calculate the unknown protein probability and tackle peptide degeneracy issue through joint probability. Therefore, the protein inference problem is equivalent to finding an optimal joint probability matrix, calculated from the matrix $P = (p_{ij})$.

Based on aforementioned observations, we present a LP formulation for the protein inference problem:

$$\max_{P} \sum_{j=1}^{m} t_j \tag{5}$$

$$\forall j : t_j \le 0 \tag{6}$$

$$\forall i,j : p_{ij} \ge t_j \tag{7}$$

$$\forall i : \ln(1 - z_i - \epsilon) \le \sum_{j=1}^{m} p_{ij} \tag{8}$$

$$\forall i : \ln(1 - z_i + \epsilon) \ge \sum_{j=1}^{m} p_{ij} \tag{9}$$

$$\forall i,j : p_{ij} \sim \begin{cases} \le 0 & if\ j \in Ne(i) \\ = 0 & else \end{cases}, \tag{10}$$

where $Ne(i)$ is the set of all proteins that can generate peptide i.

In Figure 2, we provide a vivid illustration on the main idea of this LP model. Some further remarks on the model and constrains are listed below.

- The constraints (8) and (9) control the difference between the observed and calculated peptide probabilities. Here, we regard z_i as the observed peptide probability and $\Pr(y_i = 1)$ as the calculated value where $\sum_{j=1}^{m} p_{ij} = \ln[1 - \Pr(y_i = 1)]$. In constraints (8) and (9), $\epsilon \in [0, 1]$ is the only parameter of our model, which is the difference between the observed and calculated peptide probability. This parameter reflects our confidence on peptide identifications. For instance, $\epsilon = 0$ means that we believe the input peptide probability is perfectly accurate so that we have to adjust the variable p_{ij} to make the equation hold. Hence, this parameter has a clear interpretation and it can be specified with ease. In our implementation, we use $\epsilon = 0$ as the default setting.
- The constraint (7) is to find the minimum value in p_j (the jth column of matrix P). Since only a subset of candidate proteins are truly present in the sample, some protein probability values should be zero. In order to achieve this goal, we control the maximum joint probability assigned to each protein. Since $\ln(1 - x)$ is a monotonic decreasing function, the maximum joint probability $\Pr(y_i = 1, x_j = 1)$ corresponds to the minimum value $\ln\left[1 - \Pr(y_i = 1, x_j = 1)\right]$ in p_j. Then we maximize it in the objective function (5) so as to shrink some protein probabilities to 0.
- The observed peptide probability z_i can be equal to one. This will cause a problem in our implementation since $\ln(1 - x)$ is minus infinity when $x = 1$. To address this problem, we reset the observed peptide probability to 0.99999 when $z_i = 1$.
- $p_{ij} \le 0$ and t_j is the minimum value in p_j so that t_j should be not more than zero, as specified in constraint (6).
- For notation convenience, we use $n \times m$ variables in the LP formulation described earlier in the text. In fact, the actual number of variables is less than $n \times m$ since the peptide–protein bipartite graph is very sparse. As shown in constraint (10), we set all $p_{ij} = 0$ if peptide i is not contained in protein j and consider only the remaining joint probabilities as variables. This greatly improves the running efficiency of our method. Constraint (10) also ensures that $\Pr(y_i = 1, x_j = 1)$ falls into [0,1] since it is a probability value.

Table 1. Notations and definitions

Notations	Definitions
$(1, \ldots, i, \ldots, n)$	All n peptides identified by peptide identification algorithms
$(1, \ldots, j, \ldots, m)$	All m proteins that might have generated these n peptides
$(y_1, \ldots, y_i, \ldots, y_n)$	Peptide vector: indicator variables of peptides' presences if peptide i is present, $y_i = 1$; otherwise $y_i = 0$
$(x_1, \ldots, x_j, \ldots, x_m)$	Protein vector: indicator variables of proteins' presences
$(z_1, \ldots, z_i, \ldots, z_n)$	The probabilities of peptides' presences estimated by peptide identification algorithms or PeptideProphet

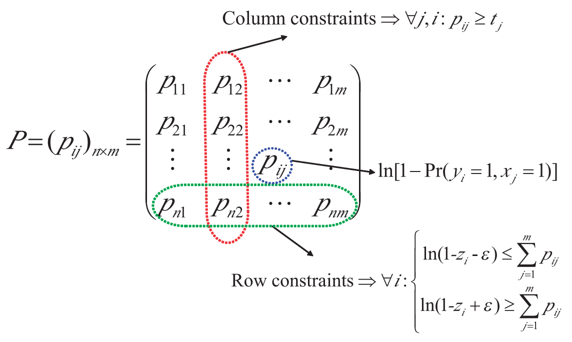
Column constraints $\Rightarrow \forall j,i: p_{ij} \geq t_j$
$P=(p_{ij})_{n\times m}=$
$\ln[1-\Pr(y_i=1,x_j=1)]$
Row constraints $\Rightarrow \forall i:$
$\ln(1-z_i-\varepsilon) \leq \sum_{j=1}^{m} p_{ij}$
$\ln(1-z_i+\varepsilon) \geq \sum_{j=1}^{m} p_{ij}$

Fig. 2. $P=(p_{ij})$ is a $n \times m$ matrix, where p_{ij} is equal to $\ln[1-\Pr(y_i=1, x_j=1)]$, and $\Pr(y_i=1, x_j=1)$ is the joint probability that peptide *i* and protein *j* are both present in the sample. In the model, the linear program has two kinds of constraints: column constraints and row constraints. The row constraints require that for each peptide *i*, the difference between the observed peptide probability and the calculated peptide probability should be no greater than a threshold ϵ. The column constraints can shrink some protein probabilities to 0

- In the model, we group the proteins with the same set of identified peptides together and regard each group as a single entity.
- Our LP model is quite flexible and can be extended easily. For instance, we currently assign a global deviation threshold ϵ to all peptides. In fact, we can also use an individual deviation threshold ϵ_i for each peptide *i*. This will provide us the possibility of assigning larger deviation thresholds to certain peptides that are suspected to be error-prone.

After obtaining the solution matrix *P*, the protein probability is calculated as:

$$\Pr(x_j = 1) = 1 - \prod_{i=1}^{n} e^{p_{ij}}. \qquad (11)$$

3 EXPERIMENTS

3.1 Data

We use six datasets in our experiments. All the datasets are publicly available. Among these six datasets, 18 mixtures (Klimek *et al.*, 2008), Sigma49 and yeast (Ramakrishnan *et al.*, 2009a) have a corresponding protein reference set as the set of ground-truth proteins. An identified protein is labeled as a true identification if it is present in the protein reference set. Another three datasets are DME (Brunner *et al.*, 2007), HumanMD (Ramakrishnan *et al.*, 2009b) and HumanEKC (Ramakrishnan *et al.*, 2009a), which have no reference sets and we use the target-decoy strategy for performance evaluation. In this target-decoy strategy, the MS/MS spectra are searched against a mixed protein database containing all target protein sequences and an equal number of decoy sequences. Then, we consider an identified protein as a true identification if it comes from the target protein database. The detailed information about the six datasets can be found in the supplementary Tables S1 and S2.

3.2 Database search

The search engine used in our experiment is X!Tandem (v2010.10.01.1) (David and Cottrell, 2004). For 18 mixtures, Sigma49 and yeast datasets, all MS/MS data are searched against their own protein sequence databases. For DME, HumanMD and HumanEKC, the spectra need to search against both target and decoy protein databases.

During the database search, we use default search parameters wherever possible, assuming that parameters have already been optimized. Some important parameter values are listed in the following: fragment monoisotopic mass error = 0.4 Da; parent monoisotopic mass error = 100 ppm; minimum peaks = 15 and minimum fragment $m/z = 150$.

Peptide probabilities are computed using PeptideProphet included in Trans-Proteomic Pipeline (TPP) v4.5. Any peptide identifications with probability <0.05 are excluded in the input. For any peptide sequence that is matched by multiple spectra with different scores, we choose the highest identification score.

3.3 Protein inference

We compare our method with ProteinProphet (Nesvizhskii *et al.*, 2003), MSBayesPro (Li *et al.*, 2009b) and Fido (Serang *et al.*, 2010). All these three algorithms treat the peptide degeneracy issue explicitly in terms of conditional probability, and their software packages are publicly available. For the proteins that cannot be distinguished with respect to identified peptides, ProteinProphet, Fido and ProteinLP put all of them into the same group. Whenever we refer to the number of TPs or false positives (FPs) identified at a threshold or use these values in a calculation, all proteins in the group are reported and the group probability is used as their protein probabilities. Alternatively, we can select one representative from each protein group in the performance comparison for these three algorithms (please check the supplementary Section 1 for details).

3.3.1 ProteinProphet We run ProteinProphet included in the TPP (v4.5) software with the default parameter values.

3.3.2 MSBayesPro We first obtain the predicted peptide detectabilities from http://darwin.informatics.indiana.edu/applications/PeptideDetectabilityPredictor/. This website currently only predicts scores of tryptic peptides. For those non-tryptic peptide identifications, we assign detectability scores to them by ourselves. The principle is peptide detectability = median (predicted detectability scores from the same parent protein)/3. Then, we run MSBayesPro for the first time with the peptide probability file and peptide detectability file as input to estimate the protein priors. Finally, we run MSBayesPro for the second time to obtain the protein probabilities using priors from the first run as additional input. The probability of each protein is reported according to the value of Positive_Probability_by_memorizing no matter what MAP_state_by_Memorizing value is.

3.3.3 Fido We run Fido with its default parameter setting.

3.3.4 ProteinLP We use Glpk for Java (v4.47) as the LP solver and set $\epsilon = 0$ in the experiment for ProteinLP.

3.4 Results

We evaluate the performance of different methods by creating a curve, which plots the number of TPs as a function of *q*-value. An identified protein is labeled as a TP if it is present in the corresponding protein reference set or target protein sequence database and as a FP otherwise. Given a certain probability threshold t, suppose there are T_t TPs and F_t FPs, the false discovery rate (FDR) is estimated as $\mathrm{FDR}_t = F_t/(F_t + T_t)$. The corresponding q-value is defined as the minimal FDR that a protein is reported: $q_t = \min_{t' \le t} \mathrm{FDR}_{t'}$. The curve is produced by varying the probability threshold t. The probabilities of top-scoring proteins in the several methods are all equal to one, and the order of these proteins in the output file is random. Thus, we skip these proteins with same probabilities and start from the one with a different score when calculating the q-value.

Figure 3 plots the number of TPs identified by four methods at different q-values. Some important observations are summarized as follows.

First, none of these four methods can always achieve the best performance over all datasets. Throughout six datasets, our method is stable and never performs the worst. Globally,

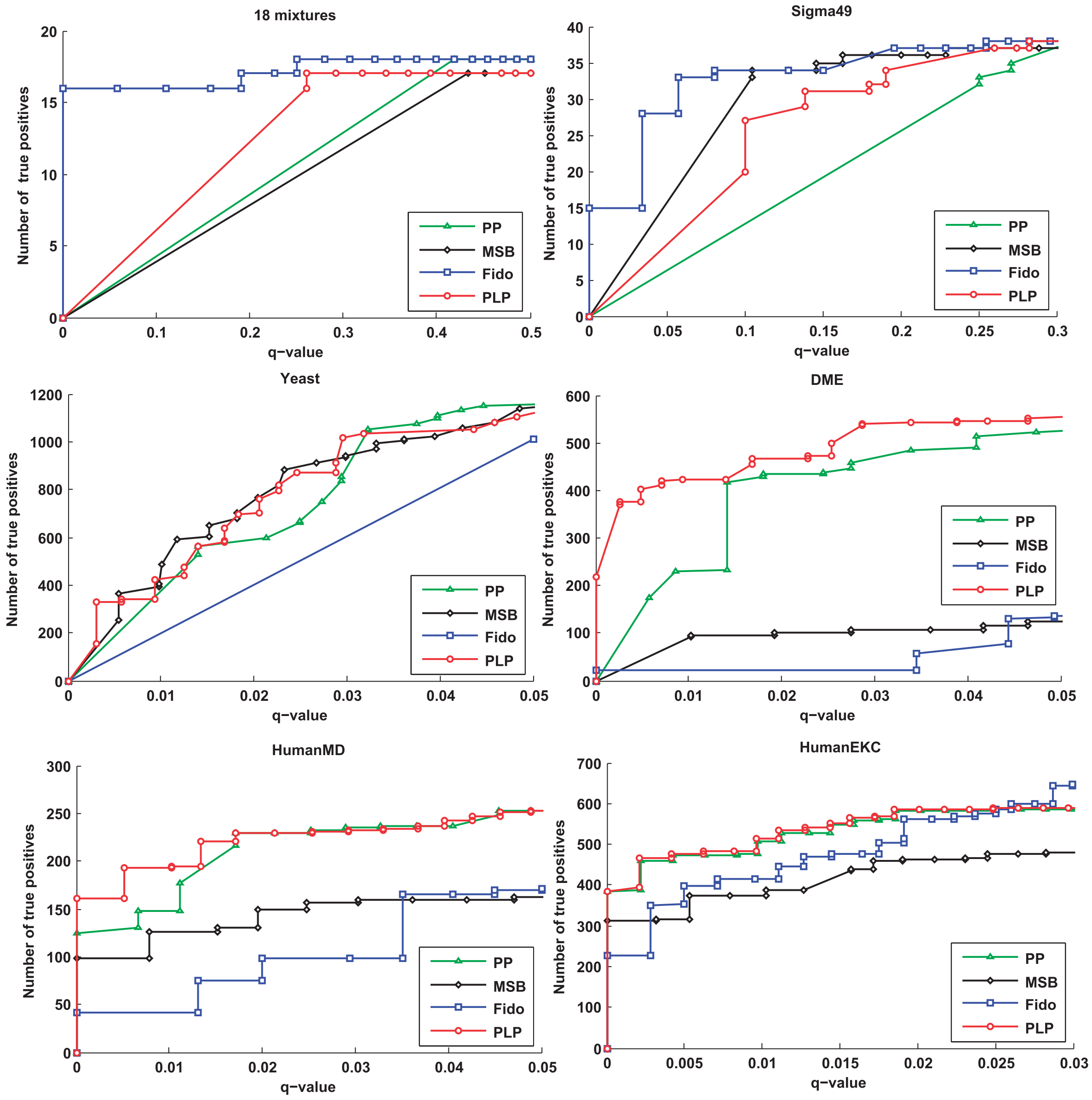

Fig. 3. Identification performance comparison among ProteinLP (PLP), Fido, ProteinProphet (PP) and MSBayesPro (MSB). Because people are particularly interested in the performance of different algorithms when the q-value or FDR is very small, we only plot the curve up to 0.05 along the x-axis for yeast, DME and HumanMD datasets. Fido has a minimum non-zero q-value of 0.08 on yeast dataset, which is >0.05. To plot the curve of Fido within the slot of [0,0.05], we use the maximal number of TPs achieved at q-value = 0.08 as the value of y-axis at q-value = 0.05. Note that such an operation overestimates the actual performance of Fido on the yeast data. Since the maximum q-value for HumanEKC is <0.04, we choose 0.03 as the maximal value of x-axis. We cannot set the q-value range very small for 18 mixtures and Sigma49 datasets since the probabilities of top-scoring proteins in the several algorithms are all equal to one, hence we have to ship these proteins with same probabilities and then calculate the q-value of the first appearing protein with a different probability

ProteinLP is approximately (or tied with other algorithms) the best inference algorithm on four datasets (yeast, DME, HumanMD and HumanEKC) and the second best on 18 mixture data. Locally, it beats ProteinProphet five times, outperforms both MSBayesPro and Fido four times.

Second, ProteinLP has the largest number of TPs among the highest ranking proteins when q-value = 0 (i.e. 0 FPs) on three datasets: DME, HumanMD and HumanEKC. Other three algorithms can also achieve such a property on some datasets. The number of these datasets is 1, 0 and 2 for ProteinProphet, MSBayesPro and Fido, respectively.

Finally, the experimental results from simple 18 mixture to complex human data show the trend that ProteinLP is more powerful on processing the MS data generated from real samples.

To compare the capability of different methods in tackling the peptide degeneracy issue, we present the identification results of four methods when inferring proteins containing a high-scoring degenerate peptide in Table 2. For each dataset, we count the number of TPs and FPs identified by ProteinProphet, MSBayesPro, Fido and ProteinLP with the same number of reported proteins. Then, we divide the identified proteins into two classes: 'degenerate proteins', which contain a high-scoring degenerate peptide and 'simple proteins' which do not contain any such degenerate peptide. From Table 2, we have the following observations.

First, ProteinProphet and ProteinLP can identify more degenerate proteins than MSBayesPro and Fido in most cases. This is because both ProteinProphet and ProteinLP tend to assign a degenerate peptide to the parent protein with more identified peptides. As a result, some degenerate proteins will have much higher probability than other proteins. In contrast, MSBayesPro and Fido do not have such a tendency. When we let different methods report the same number of proteins, ProteinProphet and ProteinLP will return more degenerate proteins since these proteins are ranked more front by these two methods.

Second, MSBayesPro can always report the least number of FP degenerate proteins on 18 mixtures, Sigma49 and yeast at the cost of identifying less TP degenerate proteins. All four methods report zero FP degenerate proteins on HumanMD and HumanEKC datasets.

Third, ProteinLP is able to identify more TP degenerate proteins than the other three methods on DME, HumanMD and HumanEKC datasets. Our method never reports the most FP degenerate proteins. Moreover, ProteinLP identifies the least number of FP degenerate proteins on DME dataset.

Overall, MSBayesPro is more powerful in controlling the false discovery rate with respect to degenerate proteins, whereas our method offers a reasonable trade-off between TP and FP rates.

Using the same set of identified proteins in Table 2, we also plot two groups of Venn diagrams to further check the overlap and difference among (degenerate) proteins identified by different inference algorithms in supplementary Figures S3 and S4. These figures show that the set of proteins identified from the same dataset by different methods can vary significantly. Moreover, ProteinLP can always report some additional (degenerate) proteins that have never been identified by the other three methods on all datasets except Sigma49. This fact further confirms that ProteinLP can serve as a strong and complementary approach for protein inference.

ProteinLP requires only one parameter: ϵ. We choose $\epsilon = 0$ as the default setting. To test the effect of this parameter, we run ProteinLP over a rough grid of ϵ that ranges from 0 to 0.9. We omit parameter value of 1.0 since all the protein probabilities are zero under this parameter setting. We use the number of TPs at certain q-value threshold as the performance metric to assess the effect of different parameters. We choose 0.3 as the q-value threshold for 18 mixtures and Sigma49 and 0.01 for all the other four datasets, respectively. As shown in Figure 4, the performance of our method is sensitive to different parameter specifications, and $\epsilon = 0$ is not the best choice. To address the parameter selection problem, we develop an entropy-based approach for setting a proper value automatically (see supplementary Section 2 for details).

Table 2. Accuracy on proteins containing degenerate peptides

	PP		MSB		Fido		PLP		PP		MSB		Fido		PLP	
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP
	18 mixtures								*Sigma49*							
Simple proteins	17	8	17	11	17	9	17	9	27	1	32	6	30	5	30	1
Degenerate proteins	1	5	0	3	1	4	0	5	5	10	4	1	5	3	5	7
	Yeast								*DME*							
Simple proteins	366	4	469	7	373	2	398	4	38	0	145	18	129	11	33	0
Degenerate proteins	164	4	62	0	106	57	132	4	136	1	11	1	34	1	142	0
	HumanMD								*HumanEKC*							
Simple proteins	70	0	119	1	111	6	64	0	147	0	184	0	180	0	143	0
Degenerate proteins	54	0	4	0	7	0	60	0	49	0	12	0	16	0	53	0

For the six datasets, we count the number of true positives and false positives identified by ProteinProphet (PP), MSBayesPro (MSB) and Fido and ProteinLP (PLP) among their top- k ranked proteins, where k is 31, 43, 538, 175, 124 and 196 for 18 mixtures, Sigma49, yeast, DME, HumanMD and HumanEKC datasets, respectively. The value of k is determined according to the number of proteins with probability of 1.0 reported by ProteinProphet. We divide the identified proteins into two classes: 'degenerate proteins' are proteins that share a high-scoring ($\geq$ 0.90) peptide with another protein and 'simple proteins' do not share such a peptide with any other protein.

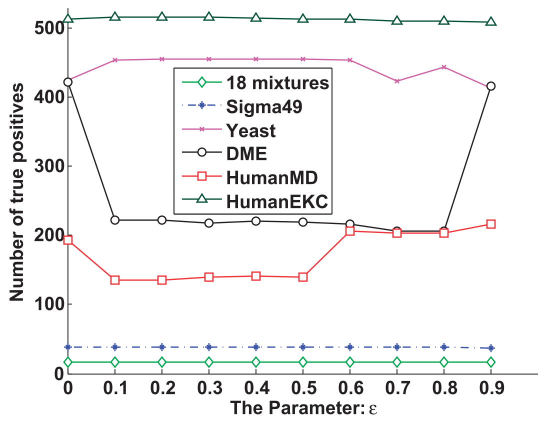

Fig. 4. Effect of parameter ϵ on ProteinLP. We use the number of TPs under some q-value threshold to assess the effect. The threshold is 0.3 for 18 mixtures, Sigma49 and is 0.01 for yeast, DME, HumanMD and HumanEKC

4 CONCLUSION

To solve the peptide degeneracy problem, we need to know which protein really generates the degenerate/shared peptide. The most natural idea is to model or infer the conditional probability of one peptide (protein) being present given a protein (peptide). However, we may still need the protein probability as the variable in the mathematical formulation besides the conditional probability. This will lead to a hard optimization problem that cannot be optimally solved. In fact, the joint probability that both a protein and its constituent peptide are present in the sample can also provide similar discrimination information for assigning a degenerate peptide to the right protein. Therefore, the main advantage of ProteinLP over other methods is the use of joint probability as the variable, which avoids modeling the protein probability and the conditional probability simultaneously so that the optimization formulation is greatly simplified.

In the future work, we plan to incorporate some supplementary information such as protein–protein interactions into the LP model to help solving the degeneracy issue. For example, if we know that there is an interaction between two proteins, then it can be expected that the existence of one protein may lead to the presence of another protein. To utilize such interaction information, we can introduce a linear constraint on the probability difference between two interacting proteins to enforce their coexistence.

Funding: The Natural Science Foundation of China (61003176).

Conflict of Interest: none declared.

REFERENCES

Bern,M. and Goldberg,D. (2008) Improved ranking functions for protein and modification-site identifications. *J. Comput. Biol.*, **15**, 705–719.

Brunner,E. *et al.* (2007) A high-quality catalog of the drosophila melanogaster proteome. *Nat. Biotechnol.*, **25**, 576–583.

David,C.M. and Cottrell,J.S. (2004) Unimod: protein modifications for mass spectrometry. *Proteomics*, **4**, 1534–1536.

Feng,J. *et al.* (2007) Probability model for assessing proteins assembled from peptides sequences inferred from tandem mass spectrometry data. *Anal. Chem.*, **79**, 3901–3911.

Gerster,S. *et al.* (2010) Protein and gene model inference based on statistical modeling in k-partite graphs. *Proc. Natl Acad. Sci. USA*, **107**, 12101–12106.

Grobei,M.A. *et al.* (2009) Deterministic protein inference for shotgun proteomics data provides new insights into Arabidopsis pollen development and function. *Genome Res.*, **19**, 1786–1800.

He,Z. *et al.* (2011) A partial set covering model for protein mixture identification using mass spectrometry data. IEEE/ACM Trans. *Comput. Biol. Bioinform.*, **8**, 368–380.

Huang,T. *et al.* (2012) Protein inference: a review. *Brief. Bioinform.*, **13**, 586–614.

Kearney,P. *et al.* (2008) Protein identification and peptide expression resolver: harmonizing protein identification with protein expression data. *J. Proteome Res.*, **7**, 234–244.

Keller,A. *et al.* (2002) Empirical statistical model to estimate the accuracy of peptide identifications made by MS/MS and database search. *Anal. Chem.*, **74**, 5383–5392.

Klimek,J. *et al.* (2008) The Standard Protein Mix Database: a diverse dataset to assist in the production of improved peptide and protein identification software tools. *J. Proteome Res.*, **7**, 96–103.

Li,J. *et al.* (2009a) Network-assisted protein identification and data interpretation in shotgun proteomics. *Mol. Syst. Biol.*, **5**, 303.

Li,Q. *et al.* (2010) A nested mixture model for protein identification using mass spectrometry. *Ann. Appl. Stat.*, **4**, 962–987.

Li,Y.F. *et al.* (2009b) A Bayesian approach to protein inference problem in shotgun proteomics. *J. Comput. Biol.*, **16**, 1–11.

Lu,B. *et al.* (2008) Improving protein identification sensitivity by combining MS and MS/MS information for shotgun proteomics using LTQ-Orbitrap high mass accuracy data. *Anal. Chem.*, **80**, 2018–2025.

Ma,Z.Q. *et al.* (2009) IDPicker 2.0: improved protein assembly with high discrimination peptide identification filtering. *J. Proteome Res.*, **8**, 3872–3881.

Moore,R.E. *et al.* (2002) Qscore: an algorithm for evaluating sequest database search results. *J. Am. Soc. Mass Spectrom.*, **13**, 378–386.

Nesvizhskii,A.I. *et al.* (2003) A statistical model for identifying proteins by tandem mass spectrometry. *Anal. Chem.*, **75**, 4646–4658.

Perkins,D.N. *et al.* (1999) Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, **20**, 3551–3567.

Price,T.S. *et al.* (2007) EBP: protein identification using multiple tandem mass spectrometry datasets. *Mol. Cell. Proteomics*, **6**, 527–536.

Qeli,E. and Ahrens,C.H. (2010) PeptideClassifier for protein inference and targeted quantitative proteomics. *Nat. Biotechnol.*, **28**, 647–650.

Ramakrishnan,S.R. *et al.* (2009*a*) Mining gene functional networks to improve mass-spectrometry based protein identification. *Bioinformatics*, **25**, 2955–2961.

Ramakrishnan,S.R. *et al.* (2009*b*) Integrating shotgun proteomics and mRNA expression data to improve protein identification. *Bioinformatics*, **25**, 1397–1403.

Sadygov,R.G. *et al.* (2004) Statistical models for protein validation using tandem mass spectral data and protein amino acid sequence databases. *Anal. Chem.*, **76**, 1664–1671.

Searle,B.C. (2010) Scaffold: a bioinformatic tool for validating MS/MS-based proteomic studies. *Proteomics*, **10**, 1265–1269.

Serang,O. *et al.* (2010) Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data. *J. Proteome Res.*, **9**, 5346–5357.

Shen,C. *et al.* (2008) A hierarchical statistical model to assess the confidence of peptides and proteins inferred from tandem mass spectrometry. *Bioinformatics*, **24**, 202–208.

Slotta,D.J. *et al.* (2010) MassSieve: panning MS/MS peptide data for proteins. *Proteomics*, **10**, 3035–3039.

Spivak,M. *et al.* (2012) Direct maximization of protein identifications from tandem mass spectra. *Mol. Cell. Proteomics*, **11**, M111.012161.

Tabb,D.L. *et al.* (2002) DTASelect and Contrast: tools for assembling and comparing protein identifications from shotgun proteomics. *J. Proteome Res.*, **1**, 21–26.

Weatherly,D.B. *et al.* (2005) A heuristic method for assigning a false-discovery rate for protein identifications from mascot database search results. *Mol. Cell. Proteomics*, **4**, 762–772.

Yang,X. *et al.* (2004) DBParser: web-based software for shotgun proteomic data analyses. *J. Proteome Res.*, **3**, 1002–1008.

Zhang,B. *et al.* (2007) Proteomic parsimony through bipartite graph analysis improves accuracy and transparency. *J. Proteome Res.*, **6**, 3549–3557.