BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 5 2014, pages 675–681
doi:10.1093/bioinformatics/btt431

Gene expression

Advance Access publication August 6, 2013

Decoy-free protein-level false discovery rate estimation

Ben Teng, Ting Huang and Zengyou He*

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

Associate Editor: Martin Bishop

ABSTRACT

Motivation: Statistical validation of protein identifications is an important issue in shotgun proteomics. The false discovery rate (FDR) is a powerful statistical tool for evaluating the protein identification result. Several research efforts have been made for FDR estimation at the protein level. However, there are still certain drawbacks in the existing FDR estimation methods based on the target-decoy strategy.

Results: In this article, we propose a decoy-free protein-level FDR estimation method. Under the null hypothesis that each candidate protein matches an identified peptide totally at random, we assign statistical significance to protein identifications in terms of the permutation *P*-value and use these *P*-values to calculate the FDR. Our method consists of three key steps: (i) generating random bipartite graphs with the same structure; (ii) calculating the protein scores on these random graphs; and (iii) calculating the permutation *P* value and final FDR. As it is time-consuming or prohibitive to execute the protein inference algorithms for thousands of times in step ii, we first train a linear regression model using the original bipartite graph and identification scores provided by the target inference algorithm. Then we use the learned regression model as a substitute of original protein inference method to predict protein scores on shuffled graphs. We test our method on six public available datasets. The results show that our method is comparable with those state-of-the-art algorithms in terms of estimation accuracy.

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

Contact: zyhe@dlut.edu.cn

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

Received on May 24, 2013; revised on July 14, 2013; accepted on July 20, 2013

1 INTRODUCTION

Shotgun proteomics is a strategy that is capable of identifying complex protein mixtures by combining high-performance liquid chromatography and mass spectrometry (MS). In shotgun proteomics, the protein identification procedure has two main steps: peptide identification and protein inference (Nesvizhskii *et al.*, 2007). In peptide identification, we search the experimental MS/MS spectra against a protein sequence database to obtain a set of peptide-spectrum matches. In protein inference, we report a set of proteins by assembling peptide identification results (Huang *et al.*, 2012).

*To whom correspondence should be addressed.

Basically, there are two major computational issues in protein identification that have to be solved. On one hand, we need to develop effective and fast identification/inference algorithms at both the peptide level and the protein level. On the other hand, controlling the quality of identified peptides and inferred proteins is as important as developing identification algorithms.

Inferred proteins are more biologically relevant than identified peptides in a proteomics experiment. Therefore, it is vital to control the quality of identification results at the protein level. However, the accurate assessment of statistical significance of protein identifications remains an open question (Spirin *et al.*, 2011).

To date, several research efforts have been made to estimate the protein-level error rate in terms of false discovery rate (FDR). The mainstream approach for FDR estimation is the target-decoy strategy, which searches a target-decoy concatenated database so that the number of false positive protein identifications can be estimated from the number of decoy proteins. For instance, the MAYU method (Reiter *et al.*, 2009) is a typical representative in this category. By adapting the target-decoy strategy to the protein inference task, the MAYU method first assumes that protein identifications containing false positive peptide-spectrum matches are uniformly distributed over the target database. Then, the number of false positive protein identifications is hypergeometrically distributed. As a result, the expected number of false positive protein identifications can be calculated as the probability weighted average. Finally, the protein identification FDR is computed as the ratio of the expected number of false positive protein identifications and the total amount of protein identifications mapping to the target database.

However, this valuable approach has certain drawbacks (Kim *et al.*, 2008). First, searching both the target and the decoy database will certainly double the running time in the protein identification process. Second, the FDR estimation result may be unstable, as we usually use *only* one decoy database with the same size of target database. Finally, the protein FDR value is calculated according to the distribution of decoy peptides across different proteins, making it possible to propagate errors at the peptide level to the protein level in a non-trivial manner.

In this article, we propose a new method for estimating the FDR at the protein level without searching the decoy database. Our method uses random permutation to assess the statistical significance of each protein in terms of *P*-value and then calculates the final FDR. First, the input for the protein inference problem can be modeled as a bipartite graph: the left is a set of identified peptides and the right is the set of candidate proteins with at least one matched peptide (Huang and He, 2012). From this bipartite graph, a protein inference algorithm calculates the probability or score for each protein. The null hypothesis in our method is that each candidate protein matches an identified peptide totally at random. Under this null hypothesis, we first create multiple random bipartite graphs with the same set of peptides and proteins. Each random bipartite graph has the same structure as the original one, i.e. each protein (peptide) is connected to the same number of peptides (proteins). Then, we run the same protein inference algorithm on these random bipartite graphs, and check if the score of each protein is significantly different on the real graph than on the randomized graphs. That is, we calculate the permutation P-value of one protein as the percentage of random graphs that produce a larger score than its original score. Finally, we sort the proteins according to their P-values and calculate the FDR at different thresholds with permutation P-values as input using the method in Storey and Tibshirani (2003).

Our method has three key steps: (i) generating random bipartite graphs with the same structure; (ii) calculating the protein scores on these random graphs; and (iii) calculating the permutation P-value and FDR. Among these steps, it is relatively easy to perform the first step and the third step. However, it is time-consuming or prohibitive to execute some protein inference algorithms for thousands of times to fulfill step ii. To address this issue, we first train a linear regression model using the original bipartite graph and identification scores given by the target inference algorithm. Then, we use the learned regression model as a substitute of original protein inference method to predict protein scores on shuffled graphs.

Experimental results on several real proteomics datasets show that our method is effective in FDR calculation. Overall, the salient features of the work described in this article can be summarized as follows:

- It can calculate the FDR without using a decoy database.
- It can be applied to evaluate the protein identification results from any algorithm that outputs protein probabilities/scores.

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

In our method, we use permutation test to calculate the P-value of each protein and then use these P-values to calculate the final FDR. Therefore, our method mainly consists of the following steps:

(1) Generating multiple random bipartite graphs with the same structure;
(2) Building a linear regression model with the original bipartite graph and protein identification scores reported by the target protein inference algorithm;
(3) Predicting the protein scores on random graphs using the trained linear regression model;
(4) Calculating the permutation P-value and FDR.

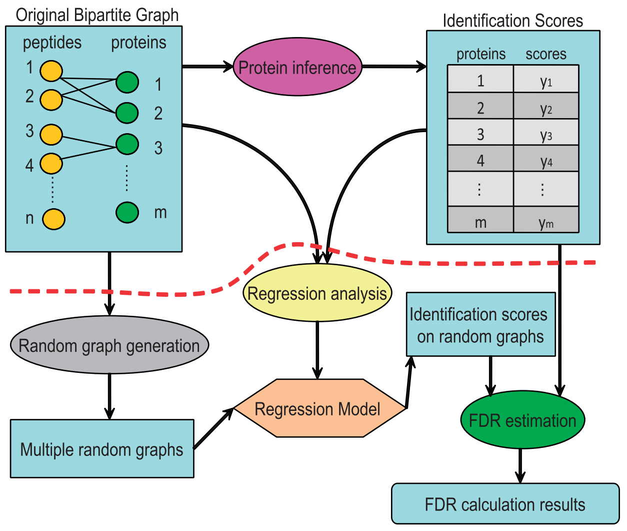

Fig. 1. An overview of our FDR estimation algorithm

Figure 1 gives an overview of this method. In the following, we will explain each step in detail. As step 2 is tightly correlated with step 3, we will elaborate them in the same subsection.

2.1 Generating random bipartite graphs

If we wish to test the null hypothesis that each candidate protein hits an identified peptide at random, it is important to construct random graphs with the same structure as the original one. It ensures that the shuffled graphs are comparable with the original graph, whereby each protein (peptide) is connected to the same number of peptides (proteins).

Given m candidate proteins and n identified peptides, the bipartite graph $G(L, R, E)$ models the input for the protein inference problem. The vertex $j \in L$ corresponds to the jth peptide and $|L| = n$. Similarly, the vertex $i \in R$ represents the ith candidate protein and $|R| = m$. Then, there exists an edge $(j, i) \in E$ if the ith candidate protein contains the jth peptide. We generate a random bipartite graph by the following steps:

(1) Record the degree of each protein/peptide (the number of peptides/proteins that each protein/peptide is connected with) in the original bipartite graph.
(2) Randomly select one protein i and one peptide j whose degrees are both larger than 0, and ensure that there is no edge between them in the current random graph.
(3) Connect protein i and peptide j and decrease the degrees of protein i and peptide j by 1.
(4) If there are no such proteins and peptides, and the number of edges in the current random graph is less than that in the original graph, select one edge (j_e, i_e) in the random graph such that there is no edge between j_e and i in the random graph. Remove edge (j_e, i_e) and add an edge (j_e, i). Increase the degree of protein i_e by 1 and decrease the degree of protein i by 1.
(5) Do steps 2–4 until the generated random graph has the same number of edges as the original graph.

Algorithm 1 shows a straightforward implementation of this generation method.

Algorithm 1 Random-graph-generation

```
Input: Original graph $G(L, R, E)$
Output: Random graph $G'(L, R, E')$ with the same structure as $G$
 1: $E' \leftarrow null$
 2: Initialize the degree of each protein and each peptide
 3: while $|E'| < |E|$ do
 4:     randomly select one protein $i$ with $Degree(i) \geq 1$
 5:     if all $j$ with $Degree(j) \geq 1$ make $(j, i) \in E'$ then
 6:         select $(j_e, i_e) \in E'$ such that $(j_e, i) \notin E'$
 7:         remove $(j_e, i_e)$ from $E'$ and add $(j_e, i)$ to $E'$
 8:         $Degree(i_e)++, Degree(i)--$
 9:         continue
10:     end if
11:     randomly select one peptide $j$ with $Degree(j) \geq 1$
12:     while $(j, i) \in E'$ do
13:         re-select one peptide $j$ with $Degree(j) \geq 1$
14:     end while
15:     $Degree(i)--,\ Degree(j)--$
16:     $E' \leftarrow E' \cup \{(j, i)\}$
17: end while
18: return $G'(L, R, E')$
```

The most important advantage of our method for generating the random graph is that it does not require any parameter. There exist some other algorithms that can generate random graphs with the same structure as the original graph. For example, the method in Gionis *et al.* (2007) is more simple and easier to understand than our method because only local swaps are performed in generating a random graph. However, this method needs to specify the number of local swaps and it is difficult to decide which parameter value is the best.

To make the protein inference results comparable, the graph randomization procedure should keep all the features of the original graph unchanged except the matches between proteins and peptides. The features of a graph include the number of nodes, the number of edges, the number of connected components and so on. As our method only ensures that the individual degree of each node (protein or peptide) is preserved, one may wonder whether the number of connected components will change substantially. In the Supplementary Figure S1, we plot the distribution of the number of connected components in random graphs. It shows that the difference between the random graph and the real graph with respect to the number of connected components is small. Therefore, protein inference is comparable between the original graph and random graphs, although the numbers of connected components of these graphs are not exactly the same.

2.2 Predicting the protein scores on random graphs

It is time-consuming and inconvenient to execute some protein inference algorithms for thousands of times. Therefore, we build a linear regression model to predict the protein scores reported by these algorithms.

We use a response variable $y_i \in R$ to denote the identification score of protein i and a predictor vector $x_i \in R^n$ to represent the set of associated peptide probabilities or scores. The element x_{ij} in vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ is constructed by the following way:

$$x_{ij} = \begin{cases} \text{probability of peptide } j, & (j, i) \in E \\ 0. & (j, i) \notin E \end{cases} \quad (1)$$

That is, if there is an edge between protein i and peptide j in the bipartite graph, the probability of peptide j will be used as the corresponding predictor value. As there are m candidate proteins, we have m observation pairs (x_i, y_i).

Essentially, all existing protein inference algorithms use the bipartite graph or the equivalent predictor vectors as input to calculate protein scores/probabilities. If we have executed a protein inference algorithm on the original bipartite graph, then we have the protein scores generated from this algorithm. From a machine learning perspective, it is possible to 'learn' a similar model that can be used to perform protein inference in the future. As a result, we can apply our method to simulate the identification results of any protein inference algorithm without knowing its algorithmic details even when its executable program is not available to us. Based on this motivation, here we build a linear regression model to realize this goal. In this approach, we search a coefficient vector β to minimize the residual sum of squares:

$$\min_\beta f(\beta) = \min_\beta [\sum_{i=1}^{m} (y_i - x_i^T \beta)^2]. \quad (2)$$

For protein i, when we have a new peptide contribution vector $\hat{x}_i$, with the estimated coefficient vector $\hat{\beta}$, we can easily use the linear model to get the identification score $\hat{y}_i$:

$$\hat{y}_i = \hat{x}_i^T \hat{\beta}. \quad (3)$$

2.3 Calculating the permutation *P*-value and FDR

After obtaining the protein scores calculated on these random graphs, we can compute the permutation *P*-value and FDR.

If our null hypothesis that each protein matches an identified peptide by chance is true, then there is no significant difference between the score of each protein in the original graph and those calculated from the random graphs. Therefore, we can calculate the permutation *P*-value of one protein as the percentage of random graphs that produce a larger score than its score generated on the original graph. More precisely, the *P*-value of the ith protein is computed as follows:

$$p_i = \frac{\#\{y_i^t > y_i\}}{K}, \quad (4)$$

where y_i is the original score of the ith protein, y_i^t is the score of the ith protein produced from the tth random graph, $\#\{y_i^t > y_i\}$ denotes the number of random graphs on which protein i has a larger score than y_i and K is the total number of random bipartite graphs.

Based on these *P*-values, we can calculate the FDR and the pFDR value using the method described in Storey and Tibshirani (2003):

$$\widehat{FDR}(\gamma) = \frac{\pi_0 \gamma}{Pr(P \leq \gamma)}, \quad (5)$$

$$p\widehat{FDR}(\gamma) = \frac{\pi_0 \gamma}{Pr(P \leq \gamma)\{1 - (1 - \gamma)^m\}}, \quad (6)$$

where γ denotes the rejection threshold and π_0 is the proportion of false positives. $Pr(P \leq \gamma)$ represents the probability that a *P*-value P is no more than γ. The pFDR (positive false discovery rate) is a modified version of the FDR. As γ becomes small, FDR is driven by the fact that the rejection threshold is small rather than the fact that the 'rate that discoveries are false' is small. In contrast, pFDR can avoid this disadvantage. Therefore, in this article, we use pFDR as the final estimated value.

We estimate π_0 by the following equation:

$$\hat{\pi}_0(\lambda) = \frac{\#\{p_i > \lambda\}}{(1 - \lambda)m}, \quad (7)$$

where p_i is the *P*-value of each candidate protein, λ is a parameter and $\#\{p_i > \lambda\}$ denotes the number of proteins whose *P*-values are larger than λ. And we use the method in Storey (2002) to pick λ.

A natural estimate of $Pr(P \leq \gamma)$ is:

$$\widehat{Pr}(P \leq \gamma) = \frac{\#\{p_i \leq \gamma\}}{m} = \frac{R(\gamma)}{m}, \tag{8}$$

where $R(\gamma) = \#\{p_i \leq \gamma\}$ represents the number of candidate proteins with P-value $\leq \gamma$.

With the estimated $\hat{\pi}_0(\lambda)$ and $\widehat{Pr}(P \leq \gamma)$, we can easily get the pFDR value at a given threshold γ.

3 EXPERIMENT

3.1 Datasets

In our experiments, we use six publicly available datasets: 18 mixtures (Klimek *et al.*, 2008), Sigma49 (Tabb *et al.*, 2007), Yeast (Ramakrishnan *et al.*, 2009a), DME (Brunner *et al.*, 2007), HumanMD (Ramakrishnan *et al.*, 2009b) and HumanEKC (Ramakrishnan *et al.*, 2009a). The first three datasets have the ground-truth protein reference sets and the other three have no such reference sets. The details of these datasets are described in the Supplementary Table S1 and S2.

We use X!Tandem (David and Cottrell, 2004) as the peptide identification method. As we compare our method with MAYU (Reiter *et al.*, 2009), all MS/MS data are searched against both target and decoy protein databases. The decoy database is generated using the Trans-Proteomic Pipeline (TPP) v4.6 software. In reality, it is not necessary to search a decoy database when using our method. During the database search, we use default search parameters and accept the peptides reported by PeptideProphet with probabilities >0.05 as the input. For any peptide sequence that is matched by multiple spectra with different scores, we choose the highest identification score.

3.2 Parameter setting

First, we run the protein inference algorithm on the original bipartite graph to obtain the original score of each protein and use these scores to build a linear regression model. Second, we predict the score of each protein in random graphs with the regression model and calculate the P-value and FDR. The protein inference algorithms used in our experiment are ProteinProphet (Nesvizhskii *et al.*, 2003) and ProteinLP (Huang and He, 2012). We run ProteinProphet included in the Trans-Proteomic Pipeline (TPP) v4.6 software with the default parameter values and use ProteinLP by setting $\epsilon = 0$. As both ProteinProphet and ProteinLP output the presence probability of each protein, we transform the score predicted by the regression model into the interval [0, 1]. If the score is >1, we set the score to 1; if the score is <0, we set the score to 0. In addition, the number of random graphs we choose is 10 000.

3.3 Results

For the first three datasets with reference sets, we apply our method and MAYU to estimate the FDR and check the difference between the estimated FDR and the ground-truth FDR. We set $\{0.05, 0.1, \ldots, 0.95, 1\}$ as the threshold, respectively. For each threshold, the smaller the difference, the better the performance.

As shown in Figure 2, for the first two datasets, our method is comparable with MAYU, while our method can provide a more accurate FDR estimation on the Yeast dataset when the P-value is larger than 0.2. We can also see that both our method and MAYU have huge deviations from the real protein FDRs for the first two synthetic datasets probably because 18 mixtures and Sigma49 do not have characteristics of those complex proteomics datasets generated from real samples. Thus, the experimental result on the more complex Yeast dataset indicates that both methods can perform relatively well on real data, and the advantage of our method begins to be visible.

For the other three datasets, we compare our method with MAYU and the naive target-decoy method. For MAYU, we set $\{0.05, 0.1, ..., 0.95, 1\}$ as the threshold, respectively. When using the naive target-decoy method, FDR is calculated by doubling the ratio of the number of decoy proteins and the total number of protein identifications in the reported proteins. As shown in Figure 3, the performance of our method is comparable with the naive target-decoy method and MAYU.

One important parameter in our method is the number of random graphs. We test the influence of different numbers of random graphs by comparing absolute difference between the estimated FDR and the true FDR at each threshold. We choose $\{1000, 3000, 5000, 7000, 10000\}$ as the parameter value, respectively. As shown in Figure 4, we can see that the absolute differences are almost the same when using different numbers of random graphs. This means that our method is insensitive to the number of random graphs when the value is >1000.

We also list the real π_0 and the estimated $\hat{\pi}_0$ on six datasets in Table 1. For the three datasets without reference sets, we calculate the real π_0 by doubling the ratio of the number of decoy proteins in the datasets and the total number of protein identifications in the datasets. As shown in Table 1, the estimated $\hat{\pi}_0$ is far from its true value on 18 mixtures and Sigma49. This explains why the absolute FDR differences on these two datasets are large.

Now we test the running efficiency of our method. The running time of our method and MAYU on six datasets is provided in Table 2. All the experiments are tested on the DELL Studio XPS 8100 workstation with 2.80 GHz CPU and 8 G main memory. It shows that our method is efficient in practice and faster than MAYU.

4 CONCLUSION

In this article, we propose a novel protein-level FDR estimation method. We assume that each candidate protein matches an identified peptide totally at random. Then, we use random permutation for assessing the statistical significance of each protein in terms of P-value and calculate the final FDR. The main advantage is that our method can calculate FDR without searching a decoy database. Experimental results on six proteomics datasets demonstrate the superiority of our method.

In the future work, we will extend our method to validate the inference results of the algorithms that do not report protein probabilities or scores. We plan to train a logistic regression model, which can assign a corresponding probability to

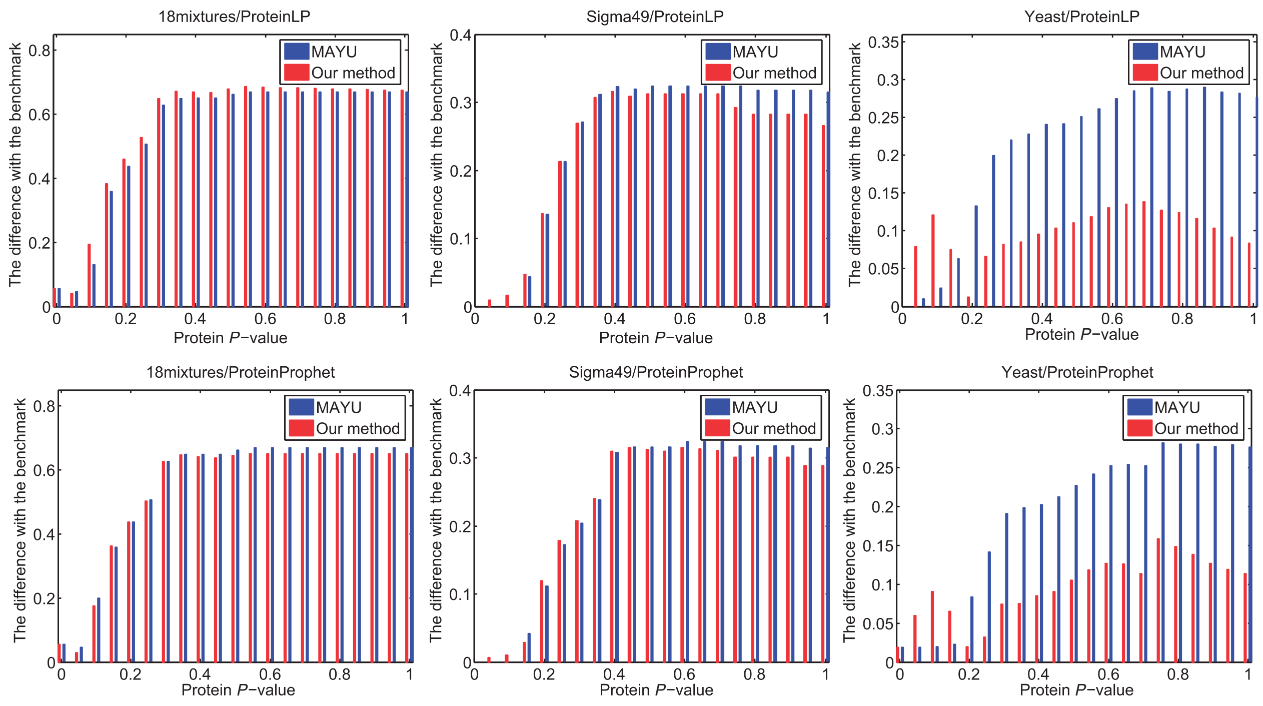

Fig. 2. Performance on 18 mixtures, Sigma49 and Yeast when ProteinLP (upper) and ProteinProphet (lower) are used as the target protein inference algorithm, respectively. The difference with the benchmark = |the estimated FDR − true FDR|. For each threshold, the smaller the difference, the better the performance. For 18 mixtures, 18 standard proteins and 15 contaminants are marked as the ground-truth. For Sigma49, all accessions in the final list of correct proteins provided by the ABRF2007 bioinformatics committee are used as the ground-truth

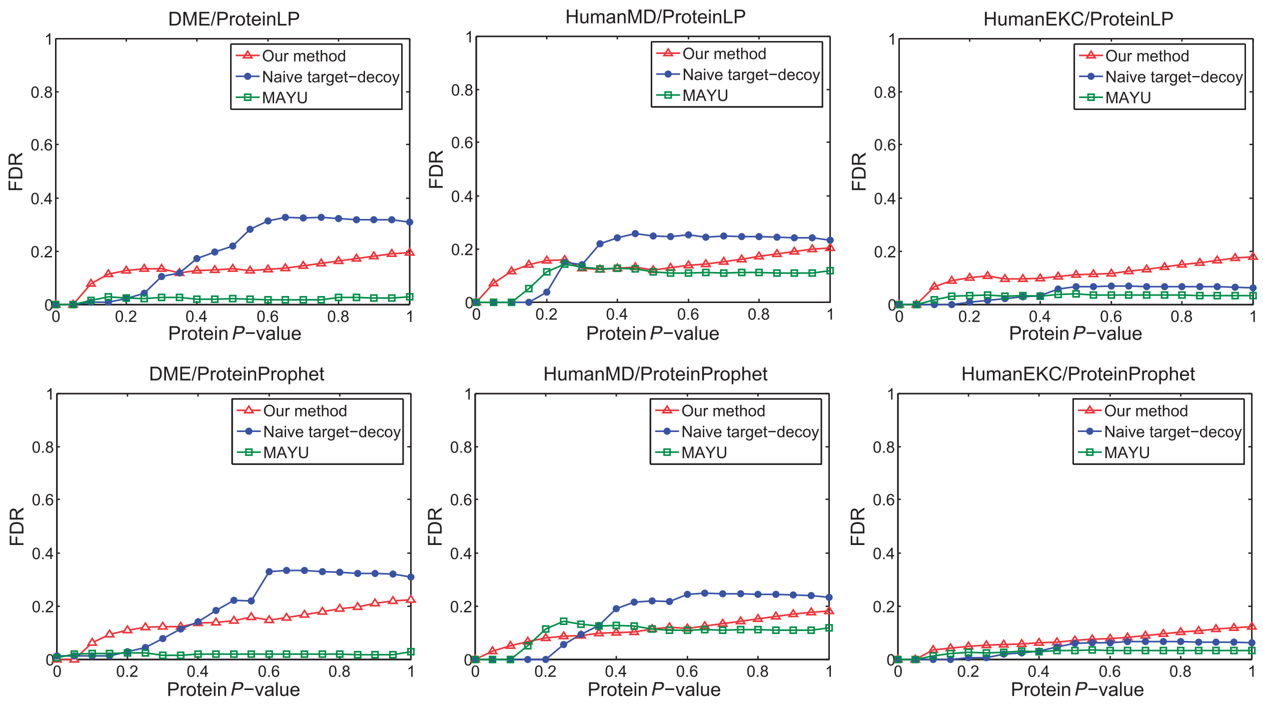

Fig. 3. Performance on DME, HumanMD and HumanEKC when ProteinLP (upper) and ProteinProphet (lower) are used as the target protein inference algorithm, respectively

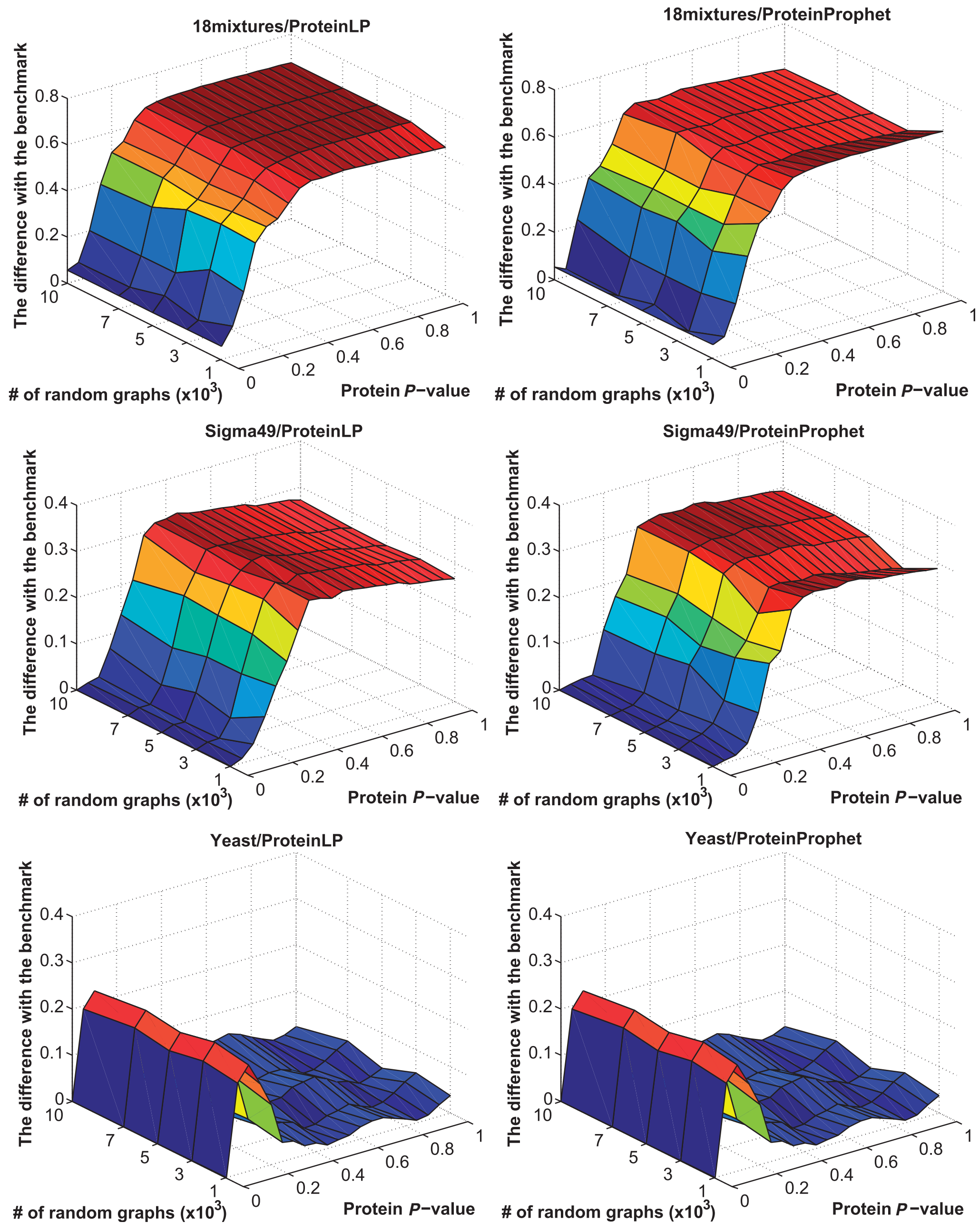

Fig. 4. The influence of different numbers of the random graphs on three datasets with reference sets

Table 1. Real π_0 and the estimated $\hat{\pi}_0$ on six datasets

Datasets	Real π_0	Estimated $\hat{\pi}_0$_1	Estimated $\hat{\pi}_0$_2
18 Mixtures	0.698	0.024	0.090
Sigma49	0.333	0.068	0.048
Yeast	0.331	0.248	0.196
DME	0.308	0.195	0.222
HumanMD	0.232	0.202	0.180
HumanEKC	0.063	0.177	0.121

Note: $\hat{\pi}_0$_1 and $\hat{\pi}_0$_2 are estimated when ProteinLP and ProteinProphet are used as the target protein inference algorithm, respectively.

Table 2. The running time of our method and MAYU on six datasets

Datasets	MAYU	Our method (ProteinLP)	Our method (ProteinProphet)
18 Mixtures	3 s	2 s	2 s
Sigma 49	30 s	2 s	2 s
Yeast	12 s	106 s	90 s
HumanEKC	41 s	35 s	35 s
DME	35 s	30 s	30 s
HumanMD	41 s	13 s	13 s

Note: MAYU can only report the FDR value at one threshold in each run, whereas our method can estimate a series of FDR values at all the thresholds. The execution time listed here for MAYU is the average running time over different thresholds. The number of random graphs in our method is fixed to be 10 000.

each reported protein. Meanwhile, as π_0 can affect the FDR estimation results significantly, we will find more accurate π_0 estimation method in the future.

Funding: This work was supported by the Natural Science Foundation of China under Grant No. 61003176.

Conflict of Interest: none declared.

REFERENCES

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.

Gionis,A. *et al.* (2007) Assessing data mining results via swap randomization. *ACM Trans. Knowl. Discov. Data*, **1**, 14.

Huang,T. and He,Z. (2012) A linear programming model for protein inference problem in shotgun proteomics. *Bioinformatics*, **28**, 2956–2962.

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

Kim,S. *et al.* (2008) Spectral probabilities and generating functions of tandem mass spectra: a strike against decoy databases. *J. Proteome Res.*, **7**, 3354–3363.

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

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

Nesvizhskii,A.I. *et al.* (2007) Analysis and validation of proteomic data generated by tandem mass spectrometry. *Nat. Methods*, **4**, 2405–2417.

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

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

Reiter,L. *et al.* (2009) Protein identification false discovery rates for very large proteomics data sets generated by tandem mass spectrometry. *Mol. Cell. Proteomics*, **8**, 787–797.

Spirin,V. *et al.* (2011) Assigning spectrum-specific p-values to protein identifications by mass spectrometry. *Bioinformatics*, **27**, 1128–1134.

Storey,J.D. (2002) A direct approach to false discovery rates. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, **64**, 479–498.

Storey,J.D. and Tibshirani,R. (2003) Statistical significance for genomewide studies. *Proc. Natl Acad Sci. USA*, **100**, 9440–9445.

Tabb,D.L. *et al.* (2007) Myrimatch: highly accurate tandem mass spectral peptide identification by multivariate hypergeometric analysis. *J. Proteome Res.*, **6**, 654–661.