BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 2 2014, pages 234–241
doi:10.1093/bioinformatics/btt642

Data and text mining

Advance Access publication November 8, 2013

PTMTreeSearch: a novel two-stage tree-search algorithm with pruning rules for the identification of post-translational modification of proteins in MS/MS spectra

Attila Kertész-Farkas[1], Beáta Reiz[2], Roberto Vera[1], Michael P. Myers[3] and Sándor Pongor[1,4,*]

[1]Protein Structure and Bioinformatics Group, International Centre for Genetic Engineering and Biotechnology, AREA Research Park, 99 Padriciano, Trieste, Italy, 34149, [2]Institute of Biophysics, Biological Research Centre, Temesvari krt. 62, H-6727 Szeged, Hungary, [3]Protein Networks Group, International Centre for Genetic Engineering and Biotechnology, AREA Research Park, Padriciano 99, 34149 Trieste, Italy and [4]Faculty of Information Technology, Pázmány Péter Catholic University, Práter u. 50/a, H-1083 Budapest, Hungary

Associate Editor: Dr Jonathan Wren

ABSTRACT

Motivation: Tandem mass spectrometry has become a standard tool for identifying post-translational modifications (PTMs) of proteins. Algorithmic searches for PTMs from tandem mass spectrum data (MS/MS) tend to be hampered by noisy data as well as by a combinatorial explosion of search space. This leads to high uncertainty and long search-execution times.

Results: To address this issue, we present PTMTreeSearch, a new algorithm that uses a large database of known PTMs to identify PTMs from MS/MS data. For a given peptide sequence, PTMTreeSearch builds a computational tree wherein each path from the root to the leaves is labeled with the amino acids of a peptide sequence. Branches then represent PTMs. Various empirical tree pruning rules have been designed to decrease the search-execution time by eliminating biologically unlikely solutions. PTMTreeSearch first identifies a relatively small set of high confidence PTM types, and in a second stage, performs a more exhaustive search on this restricted set using relaxed search parameter settings. An analysis of experimental data shows that using the same criteria for false discovery, PTMTreeSearch annotates more peptides than the current state-of-the-art methods and PTM identification algorithms, and achieves this at roughly the same execution time. PTMTreeSearch is implemented as a plugable scoring function in the X!Tandem search engine.

Availability: The source code of PTMTreeSearch and a demo server application can be found at http://net.icgeb.org/ptmtreesearch.

Contacts: pongor@icgeb.org

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

Received on May 3, 2013; revised on October 21, 2013; accepted on November 3, 2013

1 INTRODUCTION

Mass spectrometry is now the *de facto* method used for protein identification in complex biological samples. Subsequent computational analysis of data from even a single sample may be intense and involve the application of a long pipeline of various algorithms to reveal protein identification and structure. (For reviews see Aebersold and Mann, 2003; Becker and Bern, 2011; Deutsch *et al.*, 2008; Jacob, 2010; Johnson *et al.*, 2005; Kertesz-Farkas *et al.*, 2012; MacCoss, 2005; McDonald *et al.*, 2004; Menschaert *et al.*, 2010; Nesvizhskii, 2010; Nesvizhskii and Aebersold, 2004; Neumann and Bocker, 2010; Noble and MacCoss, 2012; Webb-Robertson and Cannon, 2007; Yates *et al.*, 1995.) Difficulties arise when unexpected or missed cleavages, point mutations and post-translational and chemical modifications need to be considered in the analysis, since the inclusion of these structural features of proteins can lead to the combinatorial explosion of the search space. This can lead to increases in search-execution time, decreases in the significance of hits, and increases in the number of false assignments. A solution to this problem involves application of a so-called two-round-search (often termed error-tolerant search) applied extensively in X!Tandem (Craig and Beavis, 2003, 2004; Craig *et al.*, 2004) and Mascot (Creasy and Cottrell, 2002). The first round of this strategy uses 'clean', i.e. unmodified peptides to identify proteins, and then in a second round of analysis, a more exhaustive search is used to identify modified peptides. This principle assumes that at least one high-quality unmodified peptide is present in the sample under analysis.

Identification of post-translational modifications of proteins (PTMs) is an especially difficult task. On average, human proteins are thought to carry about three PTMs per molecule (MacCoss *et al.*, 2002). However, <1% of the proteins in the UniProtKB/Swiss-Prot databases are annotated with PTMs (Tharakan *et al.*, 2010), meaning significant variations from this estimate may exist. The current list of known PTMs is >500, and they can be found in various lookup tables (PTMDBs, PTM databases). A good example is the RESID Database of Protein Modifications (Garavelli, 2003).

A PTM carried on a peptide alters the molecular weight of the peptide and the corresponding fragment ions and results in a shift within the mass spectrum fragment ion series. We term this shift a 'gap'. A PTM search can then be formalized as follows: given a theoretical and an experimental spectrum,

*To whom correspondence should be addressed.

insert one or more gaps into the theoretical spectrum such that it maximizes the number of the overlapping peaks or other scoring functions. The main challenge for the PTM-search algorithms is then to find such gaps that can be interpreted as PTMs in the MS/MS data, while keeping the execution time and the false discovery rate (FDR) low (Benjamini and Hochberg, 1995).

PTM-search algorithms can be categorized into three large groups: targeted, untargeted and *de novo* PTM-search methods (for a review see Ahrne *et al*., 2010). In the case of targeted PTM searches, the user/experimenter has to specify the PTMs that may be present in the sample as an input parameter to the search program and these parameters will then be used to calculate the gaps. For instance, X!Tandem (Craig and Beavis, 2004) considers one type of PTM per amino acid. However, this results in a combinatorial explosion of the search space that increases the execution time and often reduces the significance of the hits. For example, a partial (incomplete) modification, such as phosphorylation of serine would result in a mass shift of ~+80 Da at serine residues and would effectively double the number of serine-containing peptides. This is because each serine residue would have to be annotated with and without the mass shift due to phosphorylation.

Untargeted PTM searches do not require the user to specify expected PTMs but instead use heuristic search algorithms to find modifications on the basis of a database of known PTMs. RESID (Garavelli, 2003) is one such database that can be used in this type of search. For instance, the algorithm PILOT_PTM (prediction via integer linear optimization and tandem mass spectrometry) uses a binary integer optimization model to find PTMs that best match the experimental data under analysis (Baliban *et al*., 2010). MODi (Kim *et al*., 2006) identifies short-sequence tags (3–5-amino acid long) and fills gaps between tags with amino acids that are unmodified or modified by PTMs. PTMSearchPlus (Kertesz *et al*., 2009) uses further information obtained from accurate intact protein mass to identify PTMs in MS/MS data.

De novo PTM-search algorithms (also called 'unrestricted PTM searches or blind searches') do not use *a priori* information about modifications making it possible to discover novel, previously never reported, modifications. In general, a theoretical spectrum of a peptide sequence is aligned to an experimental spectrum, allowing placement of one or a few gaps in one way or another. These gaps are then reported as putative PTMs. To explain this in more detail, let Δ denote the mass difference between the precursor mass of the experimental spectrum and the parent mass of the theoretical peptide. Some methods have a restriction on gaps, such as the number of the gaps or the sum of the inserted gaps must be $= \Delta$. MS-Alignment (Tsur *et al*., 2005) uses a dynamic programming approach to calculate the best alignment. A similar method has been proposed by (Chiyong *et al*., 2010), but in this case does not take Δ into account during the alignment calculation. P-Mod (Hansen *et al*., 2005), PTMap (Chen *et al*., 2009) and pMatch (Ye *et al*., 2010) place Δ of each amino acid on the theoretical peptide and matches it to the experimental spectrum iteratively. In a similar manner, TwinPeaks (Havilio and Wool, 2007) and DeltAMT (Fu *et al*., 2011) calculate the location of the PTM but TwinPeaks shifts the theoretical peptide over a wide range and derives the PTM mass from the shift providing the highest score. DeltAMT meanwhile assumes that both modified and unmodified versions of the peptides are present in the sample and looks for frequent occurrences of the retention time and parent ion-mass difference between spectral pairs. High concentration of such pairs is assumed to represent modification groups in the sample. OpenSea (Searle *et al*., 2004), in a similar manner to MOD[i], identifies short-sequence tags from the spectrum but fills the gaps between tags without using a PTMDB. SIMS (Liu *et al*., 2008) meanwhile interprets pairs of product ion peaks, which represent potential amino acid residues or intervals, as a means of mapping PTMs. The *de novo* PTM identification algorithms often misplace PTMs on the amino acid sequence or require other corrections to improve accuracy. PTMClust (Chung *et al*., 2011) and PTMFinder (Tanner *et al*., 2008) have been designed to improve the quality of PTM assignments obtained with *de novo* PTM identification.

Building a prefix tree from theoretical peptide sequences constitutes a different kind of approach wherein the problem of PTM identification is reduced to tree traversal (Kertesz-Farkas *et al*., 2011). For a peptide sequence $p_1 \ldots p_n$ of length n, a prefix tree structure can be built wherein the nodes at the i-th level of the tree are labeled with the amino acid p_i and each branch represents different modifications on p_i taken from a PTMDB. At each node, the corresponding theoretical fragment ion masses are calculated and matched to the experimental spectrum, providing a score for the given node. Leaves that have gathered modifications on the path from the root and add up to the parent mass difference are considered feasible solutions. The problem with this approach is the tree's size: a complete traversal is too time-consuming for practical application. A greedy traversal heuristic has previously been proposed that traverses only a fraction of the full search space thus reducing running times to a manageable level (Kertesz-Farkas *et al*., 2011). However, greedy algorithms may fail to find the best solution and report false annotations. Instead of developing more sophisticated tree traversal algorithms for this problem, we decided to develop tree-pruning rules to eliminate unlikely modifications and reduce the search space.

In this article we present a novel method, called PTMTreeSearch, which is designed to identify modifications in experimental spectra using a PTMDB. PTMTreeSearch is executed in two rounds. Using strict tree-pruning rules, the first round of search is aimed at identifying a restricted set of PTM types with high confidence. The second round uses this restricted set of PTM types but runs with relaxed tree-pruning rules so as to identify more modified spectra. To the best of our knowledge, this method is the first that applies a two-stage-search approach to PTM identification. This idea is analogous to the two round peptide identification strategy used by X!Tandem and Mascot, where the first round is used to reduce the search space to likely solutions followed by an error-tolerant, more exhaustive search in the second round.

The rest of this article is structured as follows: Section 2 presents the PTMTreeSearch algorithm along with the tree pruning techniques applied; Section 3 describes the datasets and methods used and Section 4 presents and discusses the results. Finally, Section 5 concludes our findings and outlines further work.

2 METHODS

Let L_a be a list of modification masses (measured in Daltons) for the amino acid a and $n_a = |L_a|$ denote the number of the elements in the list (0.0 Da is always included and represents no modification). A modification mass is the mass difference that the amino acid gains or losses due to the molecular modification. For example $L_c = /0.0,\ -17.0265,\ 47.9847,\ 57.0215,\ 71.0371, \ldots,$ '/' means the amino acid cysteine can be (i) unmodified, (ii) lose 17.0265 Da (via losing ammonia from cysteine), (iii) gain 47.9847 Da (via complete oxidation), (iv) gain 57.0215 Da (via carbamidomethylation) and (v) gain 71.0371 Da (via propionamide), and so on. The molecular mass of cysteine is 121.16 Da and becomes 178.1815 Da after carbamidomethylation. Note that one particular amino acid molecule is not modified with more than one modification at the same time, but an amino acid can be modified with various modifications at different occurrences in the peptide sequence (and in different peptides as well).

Let q be an experimental spectrum represented as a list of location-intensity peak pairs, $p = a_1 a_2 \ldots a_n$ be a peptide sequence, PM(q) and PM(p) denote the precursor mass of q and p, respectively, and $\Delta = \mathrm{PM}(q) - \mathrm{PM}(p)$. The basic idea is to generate the theoretical fragment ion peaks for all modified variations of peptide p and store them in a prefix tree, where a branch at the level i denotes a PTM on amino acid a_i. A tree node at the level i contains a structure $v = \langle s, b, y, m, c \rangle$, where s is a score that quantifies the comparison of the peptide part $a_1 \ldots a_i$ to the experimental spectrum. Then c is the number and m is the sum of the mass of the acquired modifications in the sequence $a_1 \ldots a_i$. The variables b and y store the masses (m/z) of the b- and y-fragment ions that correspond to the $(a_1 \ldots a_i) + m$ and $(a_{i+1} \ldots a_n) + \Delta - m$ fragment ions, respectively.

Now we recursively define the tree and the values stored in the node structures as follows: the root node is $\langle 0, 0, \mathrm{PM}(q), 0, 0 \rangle$ at level 0. If $v = \langle s, b, y, m, c \rangle$ is a node in the tree at the level i $(0 \le i < n)$, then the node $v_j = \langle s + h, b + m_{ai+1} + m_j, y - m_{ai+1} - m_j, m + m_j, c' \rangle$ is a child of the node v at level $i+1$, where

- m_{ai+1} is the mass of the amino acid a_{i+1},
- m_j is the j-th modification in the list L_{ai+1},
- h is the sum of the intensities of the experimental peaks that matches to the theoretical peaks $b + m_{ai+1} + m_j$ and $y - m_{ai+1} - m_j$ within the small ion match tolerance,
- $c' = c + 1$ if $m_j \neq 0$, otherwise $c' = c$ (that is c' counts the PTMs on the $a_1 \ldots a_{i+1}$ fragment).

The node v_j at level $i+1$ represents that the amino acid a_{i+1} in the peptide p is modified with the j-th modification from L_{ai+1}. If v is a node at level n then v is a leaf node. A leaf $l = \langle s, b, y, m, c \rangle$ is called a feasible solution if $\Delta = m$ (up to a small precursor mass tolerance); that is the mass of the peptide with the acquired modification masses is equal to the precursor mass of the query spectrum. Note that h can be calculated in an efficient way by storing the peaks of the experimental spectrum in an ordered list ordered by the peaks' m/z locations.

The score of the spectrum q and peptide p comparison is the maximum score of the feasible solutions, if there is any, otherwise 'null' is returned. This goal can be found with any kind of tree traversal method – depth-first traversal algorithms are a good example in this case. The modifications on the peptide can then be extracted from the path between the root and the best goal leaf.

Note that all nodes at level i correspond to the i-th amino acids in peptide p, and all have the same n_{ai} number of children. Hence, the tree is balanced and all leaves have the same depth. This gives the size of the search space:

$$|T| = 1 + \sum_{j=1}^{n} \prod_{i=1}^{j} n_{a_i}$$

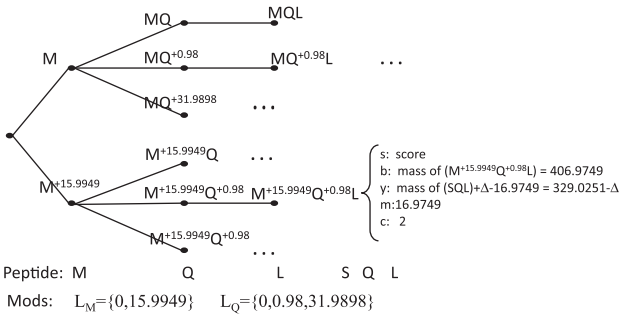

Fig. 1. Illustration of a computational tree representation of the search space of the peptide MQLSQL, where the amino acid M can be oxidized, Q can carry 0.98 and 31.9898 modifications, and where the curly bracket shows the structure of a node. Each path from the root to the leaf represents a modified peptide and each branch represents an insertion of a modification to the peptide

which makes the time complexity of the traversal algorithm impractical. Note also that the size of the search space does not depend on the experimental spectrum. In the next subsection we define pruning techniques in order to maintain the run time polynomial and make it appropriate for real applications. An example computational tree is given in Figure 1.

2.1 Search space reduction via tree pruning

In theory, for a peptide of 10 residues each modifiable by three modifications, the search space will be a tree consisting of around four million nodes. Checking all these nodes means a complete traversal of the tree would require prohibitively long calculations for practical applications. However, we can define simple rules for eliminating potential modifications and/or combinations that are unlikely to occur. This is a rule-based reduction of the search space, which corresponds to the pruning of the tree. Namely, if a rule is not fulfilled at a certain node, a sub-tree beyond the node in question is eliminated, and, as a result, tree traversal will become faster. We use the following rules.

(i) Limiting the number of modifications: Let MB be a user-defined upper bound on the number of modifications allowed on any peptide. The algorithm eliminates the node $v = \langle s, b, y, m, c \rangle$ along with the corresponding sub-tree from the search space if $c > \mathrm{MB}$. We noticed that applying a large MB value (say 5) leads to a disproportionally large number of erroneous annotations in short peptides (data not shown). This bias can be removed by applying a length-dependent upper bound, $\mathrm{MB}' = \min(\mathrm{MB}, n/3)$, where n is the length of the peptide in question.

(ii) Excluding adjacent, compensatory modifications: In the database of modifications there are 102 pairs in which the mass change is identical but of adverse sign. We term these 'pairs-compensatory' modifications. For instance, hydroxylation of lysine to hydroxyl-lysine leads to a mass gain of 15.9994 Da, while reduction of serine to alanine (or threonine to α-aminobutyrate) leads to a mass loss of 15.9994 Da. The algorithm would automatically allow these pairs to be included since the overall mass of the peptide would not change. Currently we disallow such compensatory modifications if they are adjacent (but are not necessarily on two subsequent amino acids). If the sum of the peak shifts caused by two adjacent PTMs is 0 within a certain tolerance, the node and the respective sub-tree will be pruned from the tree.

(iii) Excluding non-productive modifications: In the first approximation, a PTM is accepted if adding its mass shift to the corresponding theoretical peaks (type b- and its complementary peak

of type y-) will cause the modified peak to match with one peak of the experimental spectrum. Formally, the node $v_j = \langle s', b', y', m', c' \rangle$ is to be deleted if $c' > c$ and $s = s'$ with respect to its parent node $v = \langle s, b, y, m, c \rangle$.

(iv) Setting a threshold on matching intensities: In practice, an experimental-theoretical spectrum comparison can be considered acceptable if the total intensity of the matching peaks exceeds a certain threshold. The algorithm applies such a threshold and prunes a node when the score of the current node plus a predicted score for the path from the current node to leaves does not exceed a certain value. Formally, let S be the sum of the intensity of the d most intense experimental peaks, where d is the number of the theoretical peaks of the peptide and let matching intensities (score ratio (SR)) be a user-defined threshold ($0 < SR < 1$). The node $v = \langle s, b, y, m, c \rangle$ at level i is to be eliminated if $(s/S + (n-i)/n) < SR$. Score estimation for a path connecting a node to a leaf is a widely used technique in A*-algorithms (Hart *et al.*, 1968).

(v) Equilibrated matching frequencies: We define matching frequency as the number of theoretical peaks matching with the experimental peaks divided by the total number of the theoretical peaks. By prescribing equilibrated matching frequencies, we require that the matching frequency of fragments carrying 0, 1, 2,...PTMs, respectively, should be roughly the same. For instance, if a peptide has 10 unmodified fragments, 20 fragments carrying one PTM and 10 fragments carrying two PTMs, we expect that the ratio of matching peaks should be roughly the same, say 40%, in all three groups. For quantifying the evenness of the distribution, we define the modified peptide entropy score (MPE), which is based on the formula of Shannon entropy (see Supplementary Materials Section S1 for the formal definition). MPE is 1.0 when the modified fragment ions match to experimental peaks evenly, while MPE is 0.0 when only one type of (modified or non-modified) fragment matches. We note that MPE does not take into account the number of the matching peaks and it can give 1.0 as a value for poor but evenly matching frequencies (such as those that occur in random spectrum-peptide comparisons). Hence, MPE does not substitute the usual scoring function but rather complements it. A high MPE score is a necessary but not sufficient condition for identifying modified peptides. This rule is applied on feasible solutions (leaves)—i.e. it filters solutions but not tree nodes. Hence application of this rule does not speed up the calculations, but improves accuracy by eliminating ambiguous annotations.

PTMTreeSearch builds, explores and applies the pruning rules simultaneously, i.e. it does not build a full tree before pruning it. PTMTreeSearch builds the tree recursively, starting from the root node and applies the tree pruning rules in each newly created node. In addition, tree traversal follows a depth-first search strategy, meaning that only the path from the root to the current node is kept in memory. As a consequence, the RAM requirement of PTMTreeSearch is low. We note that rules defined here can be applied independently from each other. Supplementary Material Section S3 shows a flowchart and a pseudo code of the algorithm.

2.2 Two-stage searching

Pruning parameters [MB, non-productive modifications (NPM), SR, MPE] determine the size of the search space. Strict parameter settings (small MB, NPM, high SR and high MPE) allow identification of high confidence PTMs in good quality spectra in relatively short times. Conversely, loose parameter settings (high MB, without NPM, low SR and low MPE), allow the analysis of a larger search space and give rise to a larger set of PTMs at the expense of running time. In order to keep the benefits of both strategies, PTMTreeSearch is implemented in a two-round fashion. The first round uses strict parameter settings and serves to identify a smaller set of likely PTM types, while looser parameter settings applied in the second round identify all possible occurrences of this restricted set of PTM types. This approach is based on the assumption that each modification type present in a sample has to be found in at least one good quality spectrum. This idea builds on the two-step database search principle used by programs such as X!Tandem and Mascot.

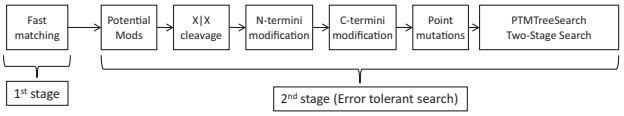

Fig. 2. The spectrum identification pipeline of the X!Tandem including PTMTreeSearch

2.3 Implementation, availability, and remarks

PTMTreeSearch is written in C++ using MPICH (message passing interface) parallelization technology. It is implemented as a pluggable scoring function (MacLean *et al.*, 2006) within the X!!Tandem search engine (Bjornson *et al.*, 2008), which is a parallel version of the X!Tandem program. PTMTreeSearch consists of two main classes. One class is called `PTMTreeSearchScore`. It derives from the class `mscore_tandem` and it implements the PTMTreeSearch method. The other class is called `PTMTreeSearch`. It derives from the `mrefine` class and initializes the variables, loads the modifications, launches the spectrum-peptide comparisons using PTMTreeSearchScore and collects the modifications found. The PTMTreeSearch class carries out the two-stage searching. The source code of PTMTreeSearch along with an installation guide and the parameter specifications can be found at http://net.icgeb.org/ptmtree search/doc. For demonstration purposes, we developed a web server that contains the X!!Tandem search engine with PTMTreeSearch included (http://net.icgeb.org/ptmtreesearch). We note that X!Tandem uses a binning technique for peak representation, while in contrast PTMTreeSearch uses the exact measured location of the peaks and the accurate mass of the modifications. The latter was chosen because it provides a higher accuracy than a binning approach. Figure 2 shows PTMTreeSearch within the X!Tandem pipeline.

X!!Tandem implements reading of the input experimental spectra, filters and manages the results and outputs it in a standard manner. From X!Tandem, as it is specified in the input parameter file, PTMTreeSearch obtains the parameter settings, such as the tolerance parameters, fragment ions to score (a-, b-, c-, x-, y-, z- ions) and considers partial, missed cleavage and multiply charged precursor ions. Fixed modifications defined in X!Tandem are not included up to the allowed PTM limit MB.

2.4 Methods and datasets

2.4.1 Collections of known PTMs We collected different types of PTMs from four public sources: (i) OMSSA (http://www.ncbi.nlm.nih.gov/ IEB/ToolBox/CPP_DOC/lxr/source/src/algo/ms/omssa/mods.xml) (Geer *et al.*, 2004), (ii) Unimod (http://www.unimod.org/downloads.html), (iii) Uniprot http://www.uniprot.org/docs/ptmlist and (iv) ResId (ftp://ftp. ebi.ac.uk/pub/databases/RESID/) (Garavelli, 2003). The data were downloaded October 10, 2011. We compiled a unique list of amino acid modifications, which resulted in 524 modification types.

2.4.2 Spectrum datasets used for testing 'The Aurum' dataset is a publicly available dataset that contains 9832 singly charged spectra generated on an ABI 4700 MALDI TOF/TOF instrument from 246 purified and trypsin-digested protein samples. This dataset was explicitly designed for testing novel MS/MS algorithms and tools (Falkner *et al.*, 2007).

The data were downloaded from ProteomeCommons.org Tranche network via the following hash: HnxUzQuuP7BIqF10aetLtjwnffOwuOMAfDvg2BFmenNe9Ue MgprBFh7+wtpbcWnXqMk2KY-8z9VjmwqXYDbQ0pTNqIx4AAAAAA SJlaw== =. 'HSPP2A' is a dataset containing 29 583 spectra (20 773 doubly and 8706 triply, 474 quadruply, 26 quintuply and four hexuply charged spectra) and was obtained with an LTQ mass spectrometer. The data were obtained from trypsin digested proteins of the human protein phosphatase 2A system (Glatter *et al*., 2009) and downloaded from www.peptideatlas. org/repository/publications/Glatter2008. The Universal Proteomics Standard I (Sigma) dataset (UPS1) contains 3368 singly charged spectra obtained from 50 trypsin-digested proteins using an Applied Biosystems 4800 MALDI TOF/TOF instrument (Bish *et al*., 2008). The data were obtained from the authors.

2.4.3 Protein sequences Computational identification of PTMs requires a collection of protein sequences, which are selected by the experimenter, and will be dependent on the goal of the experiment. In order to compare various algorithms on an equal footing, we compiled uniform collections of protein sequences that were then used with each PTM finder program used in this work. Each spectral dataset was submitted to the X!Tandem program using the IPI.Human v3.81 protein sequence dataset (Kersey *et al*., 2004), and the proteins that passed to the second round were collected in separate FASTA files (Jeong *et al*., 2012). For statistical analysis, we constructed decoy datasets (Elias and Gygi, 2007) from reversed sequences (Moore *et al*., 2002). The target and reverse datasets were unified in concatenated FASTA files before use by various programs. Supplementary Material Section S8 shows a summary of the datasets and the corresponding parameters used.

2.4.4 Performance evaluation The performance was evaluated using the receiver operator characteristic (ROC) curve technique (Green and Swets, 1966; Sonego *et al*., 2008). Annotated spectra were ranked according to a matching score. Spectra annotated with target peptide sequences were then plotted as a function of the number of spectra annotated with a decoy peptide sequence by varying a threshold over the ranking variable range. This plot gives a monotonously increasing curve, the ROC curve. A higher running ROC curve indicates better performance. The false discovery rate (FDR) was calculated as the ratio of the number of the decoy hits over the number of positive hits at a certain threshold t using the following formula (Kall *et al*., 2007):

$$\mathrm{FDR} = \frac{\text{Number of decoy}(t)}{\text{Number of target}(t)}.$$

The ROC plot can also be used to compare methods at the level of the same FDR. For instance, FDR = 1% is a straight line in the ROC plot, and the intersections with the ROC curves indicate the number of target and decoy peptides found at the same level of FDR. The FDR = 100% would coincide with the diagonal $x = y$ line.

2.4.5 PTM finding programs 'InsPecT' [version 2012.01.09 (Tsur *et al*., 2005)] was downloaded from the project web site (http://proteomics.ucsd.edu/Software/Inspect.html) and installed on Linux. Inspect was running in 'unrestrictive' mode in order to find modifications in a *de novo* way. No fixed or partial modifications were included as parameters. The parameter MaxPTMSize was set to 200 Da. The results files were then post-processed by ComputeFDR.jar script (part of Inspect) and the field InspectFDR was used as a ranking variable for the ROC analysis.

MOD[i] v3.01 (Kim *et al*., 2006) was run remotely at the web server available at http://prix.hanyang.ac.kr/modi/search.jsp. The runs were performed without fixed modifications and by allowing all 615 variable modifications present in the database of MOD[i]. The modification range was set from −50 to 200 Da. The ROC analysis was carried out using the 'Probability' field as ranking parameter. 'SIMS' (Liu *et al*., 2008) was obtained from the authors. SIMS was used with the 'FULL_TRYPTIC FALSE' parameter setting and the modification range was set to 0–200 Da. No fixed or variable modifications were used. 'X!Tandem' version 10-12-01 (Craig and Beavis, 2004) was downloaded from www.thegpm.org. For the experimental calculations only, the source code of X!Tandem was slightly altered such that it uses the complete restricted protein sequences in the second round in order to allow FDR and ROC calculations based on target/decoy comparisons. The LogE values taken from the X!Tandem output file were used for FDR and ROC calculations.

2.4.6 Computational environment All programs (except MOD[i], which was run on a web server) were executed on a Linux cluster consisting of one frontend and 20 backend nodes, each equipped with 2.2 Ghz CPU and 2 GB memory. The execution time was calculated as if the experiments were run on a single CPU computer in order to make the execution time comparable with other methods.

3 RESULTS AND DISCUSSIONS

3.1 Comparison of tree-pruning strategies

PTMTreeSearch has three parameters (MB, SR, NPM) that control the size of the search space. In this section we investigate their impact on accuracy and execution time. First, we fixed MB = 1 and MPE = 0.0, and we defined six search strategies (Table 1) for the SR and NPM, in which the size of the search space gradually decreases from complete search (no tree pruning) to restricted search (strict tree pruning).These search strategies were applied to the three spectrum datasets and the identified peptides (spectra that are assigned to a peptide sequence), the modified peptides (peptides that carry at least one modification) and the modification types (a modification from the PTMDB, such as carbamidomethylation of cysteine or oxidation of methionine) were counted at the level of 1% FDR. Note that the number of identified peptides includes both modified and unmodified peptides. The exact numbers can be found in the Supplementary Materials Table S3, Section S4. On each dataset, the relative increase or decrease in speed, compared to the full search (Search Strategy 1) were calculated, and averaged over the three spectrum datasets. These are illustrated in Figure 3. As expected, stricter pruning rules reduced the execution time (PTMTreeSearch became 3–4-times faster) however they also reduce the number of the identified modification types and annotated modified peptides. We have evaluated the six search strategies using MB = 2, and the results are presented in the Supplementary Materials Table S4, Figure S8 in Section S4. These results show that larger search space can result in higher execution times and fewer annotations at FDR = 1%. When tree-pruning rules are applied, the execution time decreases significantly and the number of annotated peptides increases. In order to further improve the performance we decided to implement a two-stage-search process.

3.2 Two-stage search

In the first stage of this strategy, strict tree pruning rules are used to identify trusted modification types from PTMDB. The modification types found in the first stage are filtered according to the MPE criterion as described in Supplementary Materials Sections S1 and S2. A more exhaustive search is then performed in the second round, using this relatively small set of trusted

Table 1. Search strategy (SS) names and definitions

SS Name	Full	NPM	SR2	SR4	SR6	SR8
NPM	No	Yes	Yes	Yes	Yes	Yes
SR	0	0	0.2	0.4	0.6	0.8

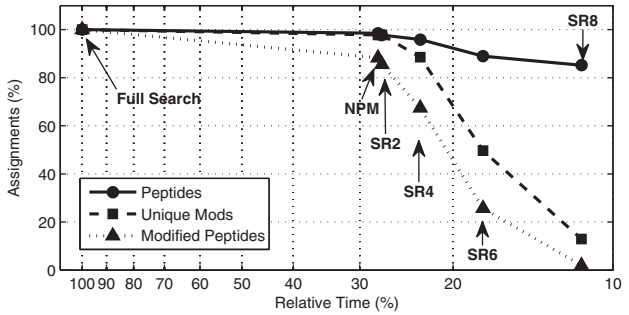

Fig. 3. The effect of tree pruning on execution times and the number of assignments (peptides, modified peptides, modification types). The abbreviations of the tree-pruning strategies are explained in the text. The data were normalized by taking the number of assignments and the running time values of the full database search as 100%, and then the data obtained on the three datasets (Aurum, UPS1, HSPP2A) were averaged. Two-stage search

modification types in conjunction with more relaxed tree pruning rules. In practice, we used MB = 1, with NPM, SR = 0.4 and MPE = 0.99 as the parameter settings for the first round. The impact of the MPE on the number of the modification types and on the results can be found in the Supplementary Materials (Section S2). A comparison of PTMTreeSearch with MPi Preview v1.0 is shown in Supplementary Material Section S6. For the second stage, we used the following parameter settings: SR = 0.3, without NPM, and MPE = 0.00, with various values (between 1 and 5) for MB. The number of allowed modifications, MB, is a critically important parameter since it regulates the search space size in all PTM finding algorithms. Untargeted PTM finders used in practice frequently produce prohibitively long running times if more than two or three modifications are allowed, while *de novo* PTM finders usually allow one modification per peptide. The effect of MB on the performance of PTMTreeSearch is shown in Figure 4. Data used for the figure can be found in the Section S5 in the Supplementary Materials. In these experiments, MB was varied only in the second round of the search. The data show that allowing more modifications dramatically increases the running time of PTMTreeSearch. Allowing four to seven modifications increases the running times by 10–50 times, respectively. It is also apparent that allowing more than three or four modifications does not improve the number of correct assignments. The change in correct assignments is simply due to the fact that the combinations of more modifications can easily produce random matches which will then impair the significance of the hits. Therefore an optimal settings for the two-stage search process we propose are MB = 1, using NPM, SR = 0.4, MPE = 0.99, for the first round and SR = 0.3, MPE = 0.0, without NMP and MB = 1, 2, 3. The

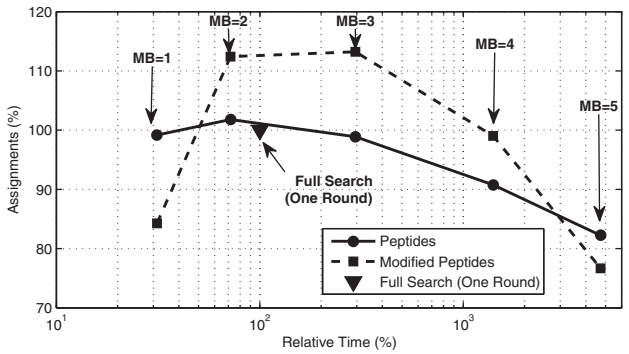

Fig. 4. The effect of allowed modifications on the number of assignments (number of identified peptides, modified peptides) in the two-stage search. The general parameter settings were (MB = 1, with NPM, SR = 0.4, MPE = 0.99) for the first and (SR = 0.3, without NPM, MPE = 0.0) with MB = 1,..., 5 for the second stage, respectively. MB was varied in round 2 only. The data were normalized by taking the number of assignments and the running time values of the full database search as 100%, and then the data obtained on the three datasets were averaged

Section S5 in the Supplementary Materials presents results with various parameter settings.

3.3 Comparison of PTMTreeSearch with other state-of-the-art methods

The performance of the optimized two-stage PTMTreeSearch was compared to InsPecT, MOD[i] and SIMS using the ROC analysis and shown in Figure 5. At any given FDR, PTMTreeSearch identifies more peptides and misses relatively fewer than the other methods. Figure 6 shows that PTMTreeSearch identifies 73% more and misses 13% less peptides on average in comparison to the other methods at FDR = 1%. For a detailed comparison and execution times see Supplementary Tables S12 and S13, respectively. Supplementary Excel Table 3 presents a summary about known modification types found in UPS1 dataset. If one PTM is allowed, the running time of PTMTreeSearch compares quite favorably to those of the other programs. If we allow more modifications, PTMTreeSearch becomes slower. For these cases, we cannot compare the accuracies since InsPecT and SIMS allow only one modification per peptide.

We also evaluated PTMTreeSearch on the PTM benchmarking dataset of The Proteome Informatics Research Group (iPRG) of the Association of Biomolecular Resource Facilities (ABRF), which was designed for a programming contest in 2012 (http://www.abrf.org/index.cfm/group.show/ProteomicsInformaticsResearchGroup.53.htm#R_4). This spectrum dataset has been spiked with 69 additional synthetic peptides, with modified phosphorylation, methylation, acetylation, nitro, sulfation, etc. We compared PTMSearch with the two best anonymous contest participants, 71755v and EK93128i, as well as Inspect, SIMs and MODi. Supplementary Figure S12 shows that PTMTreeSearch identifies 35% more peptides than the two best submissions, compared at FDR = 1%.

Supplementary Excel Table 1 summarizes the modification types found in this dataset. Supplementary Excel Table 2 shows a separate list of hits found on the 69 spiked modified

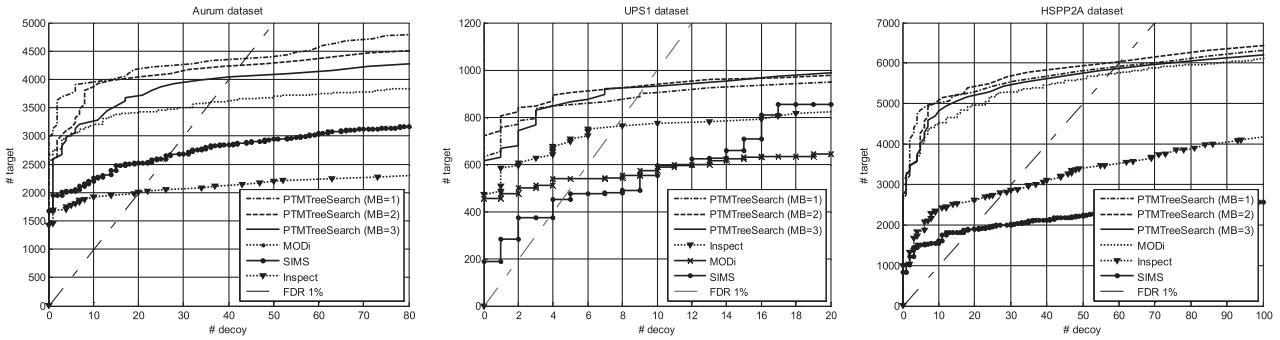

Fig. 5. ROC comparisons of the PTMTreeSearch to the state-of-the-art methods. The intersection of the ROC curve and the straight line indicates the number of target and decoy peptides annotated at 1% FDR

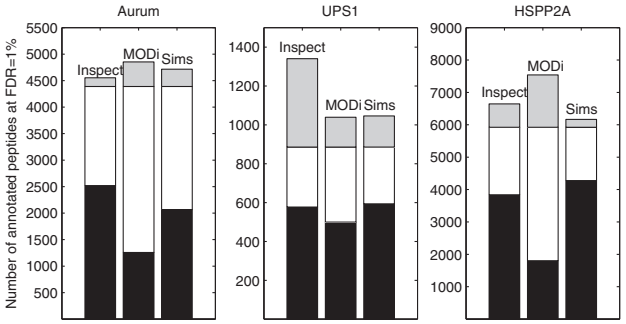

Fig. 6. Pair-wise comparison of results obtained with PTMTreeSearch to the state-of-the-art methods. The grey bar denotes the number of the spectra annotated with other program (indicated on the top of the bar). The white bar denotes the number of the spectrum annotated with both the PTMTreeSearch and the other program. The black bar denotes the number of the spectrum annotated only with PTMTreeSearch program

peptides. In this comparison, PTMTreeSearch was also one of the best performers, although this, depended on the number of modifications allowed (See Supplementary Materials Section S8 for further details).

We have compared the current version of the PTMTreeSearch to its previous version (Kertesz-Farkas *et al*., 2011). In the earlier version, the full search space was traversed using a greedy algorithm allowing for backtracking with a limited size priority double-ended queue. On average, on our datasets, the greedy approach identifies 36% more modification types, but 5% fewer peptides, 18% fewer modified peptides and takes three and half times longer than the current version of PTMTreeSearch. Data are shown in the Section S7 in Supplementary Materials. The time increase is due to the priority queue operations, which are logarithmic in time. The current version of PTMTreeSearch using tree-pruning rules and a two-round search strategy provided more assignments in shorter running times.

4 CONCLUSIONS

In this article, we have presented a novel algorithm called PTMTreeSearch for identification of PTM in tandem mass spectrometry data using a large collection of known amino acid modifications. The method employs a prefix tree for search space representation where branches represent PTMs and tree pruning techniques are used to eliminate unlikely solutions and to reduce the execution time. Moreover, it uses a two-stage search, where a small set of trusted modification types are identified in the first round, and this restricted set of modification types is used in the second round to identify PTMs in an exhaustive manner.

We believe PTMTreeSearch and the state-of-the-art methods (such as Inspect, SIMS, MOD[i]) are able to find the PTMs in a complex biological sample. The question remains whether modifications are identified with high confidence or if the modified peptide annotation vanishes among the decoy matches. In our opinion the computational tree provides a natural way to represent the search space for finding modifications and avoiding redundant calculations. The tree pruning techniques and the two-stage search provide more accurate modification identification at a reasonable time cost, and identifies more peptides than current state-of-the art PTM finder algorithms at the same FDR level.

We implemented PTMTreeSearch as a plug-in of the X!Tandem framework, and hence PTMTreeSearch is fully compatible with all data and output formats of X!Tandem. In addition, PTMTreeSearch fixes one drawback of X!Tandem - namely, that X!Tandem cannot detect different modifications on the same type of amino acid (not on the same side-chain). For example, with PTMTreeSearch it becomes possible to search for multiple modifications of cysteine without multiple searches. Simply put, if the corresponding modifications are part of the user-defined PTMDB, PTMTreeSearch will consider them in various combinations on the peptide sequences at the same time. Finally, we note that PTMTreeSearch interprets peak shifts using known PTMs, so it cannot identify PTMs that neither induce peak shifts nor alter the type of the product ions.

ACKNOWLEDGEMENTS

The authors thank Drs. Yoonsung Joh, Eunok Paek and the MODi team for their help with using MODi program as well as to Dr Max Bingham (freelance, Rotterdam, The Netherlands) for editorial assistance prior to submission.

Funding: Arturo Falaschi Fellowship program (AKF) (to the work at ICGEB, Trieste); Informatics PhD Program of the University of Szeged, Hungary (BR) and Hungarian National Research Foundation OTKA [grant number K.84335 BIOIN]

(to the work at BRC Szeged); grants TET_10-1-2011-0058, TAMOP-4.2.1.B_11/2/KMR-2011-0002 and TAMOP-4.2.2/B-10/1-2010-0014 (to the work at Pázmány University, Budapest).

Conflict of Interest: none declared.

REFERENCES

Aebersold,R. and Mann,M. (2003) Mass spectrometry-based proteomics. *Nature*, **422**, 198–207.

Ahrne,E. *et al.* (2010) Unrestricted identification of modified proteins using MS/MS. *Proteomics*, **10**, 671–686.

Baliban,R.C. *et al.* (2010) A novel approach for untargeted post-translational modification identification using integer linear optimization and tandem mass spectrometry. *Mol Cell Proteom.*, **9**, 764–779.

Becker,C.H. and Bern,M. (2011) Recent developments in quantitative proteomics. *Mutation Res.*, **722**, 171–182.

Benjamini,Y. and Hochberg,Y. (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. Royal Stat. Soc. Series B (Methodological)*, **57**, 289–300.

Bish,R.A. *et al.* (2008) Conjugation of complex polyubiquitin chains to WRNIP1. *J. Proteome Res.*, **7**, 3481–3489.

Bjornson,R.D. *et al.* (2008) X!!Tandem, an improved method for running X!tandem in parallel on collections of commodity computers. *J. Proteome Res.*, **7**, 293–299.

Chen,Y. *et al.* (2009) PTMap—a sequence alignment software for unrestricted, accurate, and full-spectrum identification of post-translational modification sites. *Proc. Natl Acad. Sci. USA*, **106**, 761–766.

Chiyong,K. *et al.* (2010) Unrestricted identification of post translational modifications from tandem mass spectra datasets. In: *Proceedings of the International Conference on Bioinformatics and Biomedical Technology (ICBBT), 2010, Sanya, China*. pp. 244–247.

Chung,C. *et al.* (2011) Computational refinement of post-translational modifications predicted from tandem mass spectrometry. *Bioinformatics (Oxford, England)*, **27**, 797–806.

Craig,R. and Beavis,R.C. (2003) A method for reducing the time required to match protein sequences with tandem mass spectra. *Rapid Commun. Mass Spectrom.*, **17**, 2310–2316.

Craig,R. and Beavis,R.C. (2004) TANDEM: matching proteins with tandem mass spectra. *Bioinformatics (Oxford, England)*, **20**, 1466–1467.

Craig,R. *et al.* (2004) Open source system for analyzing, validating, and storing protein identification data. *J. Proteome Res.*, **3**, 1234–1242.

Creasy,D.M. and Cottrell,J.S. (2002) Error tolerant searching of uninterpreted tandem mass spectrometry data. *Proteomics*, **2**, 1426–1434.

Deutsch,E.W. *et al.* (2008) Data analysis and bioinformatics tools for tandem mass spectrometry in proteomics. *Physiol. Genom.*, **33**, 18–25.

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

Falkner,J.A. *et al.* (2007) Validated MALDI-TOF/TOF mass spectra for protein standards. *J. Am. Soc. Mass Spectrom.*, **18**, 850–855.

Fu,Y. *et al.* (2011) DeltAMT: a statistical algorithm for fast detection of protein modifications from LC-MS/MS data. *Mol. Cell Proteom.*, **10**, M110 000455.

Garavelli,J.S. (2003) The RESID database of protein modifications: 2003 developments. *Nucleic Acids Res.*, **31**, 499–501.

Geer,L.Y. *et al.* (2004) Open mass spectrometry search algorithm. *J. Proteome Res.*, **3**, 958–964.

Glatter,T. *et al.* (2009) An integrated workflow for charting the human interaction proteome: insights into the PP2A system. *Mol. Syst. Biol.*, **5**, 237.

Green,D.M. and Swets,J.A. (1966) *Signal Detection Theory and Psychophysics*. Wiley, New York.

Hansen,B.T. *et al.* (2005) P-Mod: an algorithm and software to map modifications to peptide sequences using tandem MS data. *J. Proteome Res.*, **4**, 358–368.

Hart,P.E., Nilsson,N.J. and Raphael,B. (1968) A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics*, **4**, 100–107.

Havilio,M. and Wool,A. (2007) Large-scale unrestricted identification of post-translation modifications using tandem mass spectrometry. *Anal. Chem.*, **79**, 1362–1368.

Jacob,R.J. (2010) Bioinformatics for LC-MS/MS-based proteomics. *Methods Mol. Biol.*, **658**, 61–91.

Jeong,K. *et al.* (2012) False discovery rates in spectral identification. *BMC Bioinform.*, **13 (Suppl 16)**, S2.

Johnson,R.S. *et al.* (2005) Informatics for protein identification by mass spectrometry. *Methods*, **35**, 223–236.

Kall,L. *et al.* (2007) Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. *J. Proteome Res.*, **7**, 29–34.

Kersey,P.J. *et al.* (2004) The International Protein Index: an integrated database for proteomics experiments. *Proteomics*, **4**, 1985–1988.

Kertesz-Farkas,A. *et al.* (2011) PTMSearch: a greedy tree traversal algorithm for finding protein post-translational modifications in tandem mass spectra. In: *Proceedings of the 2011 European conference on Machine Learning and Knowledge Discovery in Databases - Volume Part II*. Springer-Verlag, Athens, Greece.

Kertesz-Farkas,A. *et al.* (2012) Database searching in mass spectrometry based proteomics. *Curr. Bioinform.*, **7**, 221–230.

Kertesz,V. *et al.* (2009) PTMSearchPlus: software tool for automated protein identification and post-translational modification characterization by integrating accurate intact protein mass and bottom-up mass spectrometric data searches. *Anal. Chem.*, **81**, 8387–8395.

Kim,S. *et al.* (2006) MODi: a powerful and convenient web server for identifying multiple post-translational peptide modifications from tandem mass spectra. *Nucleic Acids Res.*, **34**, W258–W263.

Liu,J. *et al.* (2008) Sequential interval motif search: unrestricted database surveys of global MS/MS data sets for detection of putative post-translational modifications. *Anal. Chem.*, **80**, 7846–7854.

MacCoss,M.J. (2005) Computational analysis of shotgun proteomics data. *Curr. Opin. Chem. Biol.*, **9**, 88–94.

MacCoss,M.J. *et al.* (2002) Shotgun identification of protein modifications from protein complexes and lens tissue. *Proc. Natl Acad. Sci. USA*, **99**, 7900–7905.

MacLean,B., Eng,J.K., Beavis,R.C. and McIntosh,M. (2006) General framework for developing and evaluating database scoring algorithms using the TANDEM search engine. *Bioinformatics*, **22**, 2830–2832.

McDonald,W.H. *et al.* (2004) MS1, MS2, and SQT-three unified, compact, and easily parsed file formats for the storage of shotgun proteomic spectra and identifications. *Rapid Commun. Mass Spectrom.*, **18**, 2162–2168.

Menschaert,G. *et al.* (2010) Peptidomics coming of age: a review of contributions from a bioinformatics angle. *J. Proteome Res.*, **9**, 2051–2061.

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. (2010) A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. *J. Proteom.*, **73**, 2092–2123.

Nesvizhskii,A.I. and Aebersold,R. (2004) Analysis, statistical validation and dissemination of large-scale proteomics datasets generated by tandem MS. *Drug Discov. Today*, **9**, 173–181.

Neumann,S. and Bocker,S. (2010) Computational mass spectrometry for metabolomics: identification of metabolites and small molecules. *Anal. Bioanalyt. Chem.*, **398**, 2779–2788.

Noble,W.S. and MacCoss,M.J. (2012) Computational and statistical analysis of protein mass spectrometry data. *PLoS Comput Biol.*, **8**, e1002296.

Searle,B.C. *et al.* (2004) High-throughput identification of proteins and unanticipated sequence modifications using a mass-based alignment algorithm for MS/MS de novo sequencing results. *Anal. Chem.*, **76**, 2220–2230.

Sonego,P. *et al.* (2008) ROC analysis: applications to the classification of biological sequences and 3D structures. *Brief. Bioinform.*, **9**, 198–209.

Tanner,S. *et al.* (2008) Accurate annotation of peptide modifications through unrestrictive database search. *J. Proteome Res.*, **7**, 170–181.

Tharakan,R. *et al.* (2010) Data maximization by multipass analysis of protein mass spectra. *Proteomics*, **10**, 1160–1171.

Tsur,D. *et al.* (2005) Identification of post-translational modifications via blind search of mass-spectra. In: *Proceedings/IEEE Computational Systems Bioinformatics Conference, CSB, San Francisco, CA, USA*. pp. 157–166.

Webb-Robertson,B.J. and Cannon,W.R. (2007) Current trends in computational inference from mass spectrometry-based proteomics. *Brief. Bioinform.*, **8**, 304–317.

Yates,J.R. 3rd *et al.* (1995) Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. *Anal. Chem.*, **67**, 1426–1436.

Ye,D. *et al.* (2010) Open MS/MS spectral library search to identify unanticipated post-translational modifications and increase spectral identification rate. *Bioinformatics (Oxford, England)*, **26**, i399–i406.