PTMSearch: a Greedy Tree Traversal Algorithm for Finding Protein Post-Translational Modifications in Tandem Mass Spectra

Attila Kertész-Farkas[1], Beáta Reiz[2,3], Michael P. Myers[1], Sándor Pongor[1,2]

[1]International Centre for Genetic Engineering and Biotechnology, Padriciano 99, I-34012 Trieste, Italy, [2]Bioinformatics Group, Biological Research Centre, Hungarian Academy of Sciences, Temesvári krt. 62, H-6701 Szeged, Hungary, [3] Institute of Informatics, University of Szeged, Aradivértanúk tere 1, H-6720, Szeged, Hungary,
kfattila@icgeb.org

Abstract. Peptide identification by tandem mass spectrometry (MS/MS) and database searching is becoming the standard high-throughput technology in many areas of the life sciences. The analysis of post-translational modifications (PTMs) is a major source of complications in this area which calls for efficient computational approaches. In this paper we describe PTMSearch, a novel algorithm in which the PTM search space is represented by a tree structure, and a greedy traversal algorithm is used to identify a path within the tree that corresponds to PTMs that best fit the input data. Tests on simulated and real (experimental) PTMs show that the algorithm performs well in terms of speed and accuracy. Estimates are given for the error caused by the greedy heuristics, for the size of the search space and a scheme is presented for the calculation of statistical significance.

1 Introduction

Tandem mass spectrometry (MS/MS) has become the major tool for high throughput protein analysis in the biomedical field, and the analysis of the data entirely relies on database searching algorithms. The difficulties in analysis arise from both the quantity of data, as current instruments can produce tens of thousands of spectra within an hour, and the quality of data as there are measurement errors, as well as both missing and extraneous data points.

According to the most widely used methodology, peptide sequences, taken from a sequence database, are matched with the experimental MS/MS spectra in order to suggest peptide candidates for further analysis. The algorithmic strategy is seemingly straightforward: protein sequences are divided into peptide segments that in turn are converted into theoretical spectra according to chemical rules and stored in a database [?]. Each experimental MS/MS spectrum is then compared with all theoretical spectrum in the database, and the most similar peptide is stored for further analysis.

The difficulties come from the fact that experimental and theoretical spectra can differ for reasons other than instrument noise. One of the most important sources of variability is the so-called post-translational modifications (PTMs) of proteins. In biological systems, proteins can carry a number of PTMs on their amino acid side-chains which leads to a combinatorial explosion in the number of theoretical spectra to be compared. As there are thousands of already discovered PTMs [**?**], accurate identification of PTMs through analysis of high-throughput MS/MS data is considered a highly challenging problem [**?**], and this is the subject of our work.

Here we present PTMSearch, a greedy-tree search algorithm designed for the identification of PTMs in tandem mass spectrometry data. The approach is based on a tree-representation of the search space in which nodes are amino acid positions and branches represent potential PTMs allowed on the type of amino acid in the given position. In this representation, a path from the root to a leaf represents a peptide with PTMs. PTMSearch uses a greedy traversal procedure to find the path in the tree that best fits to the input data. PTMSearch includes tree pruning heuristics in order to keep the computation time reasonable and polynomial.

The rest of this paper is structured as follows: Section 2 is a brief outline of the spectrum comparison problem and the role of PTMs in the process (optional). Section 3 is a summary of related work. In Section 4 we describe our proposed algorithm and the speedup methodologies. Section 5 describes the calculation of significance used with the algorithm. Section 6 contains the results obtained on experimental datasets and comparisons with other method. Section 7 is a discussion of the results and an outline of future work.

2 MS/MS spectra and PTMs

This section presents a simplified description of mass spectra for readers not familiar with the field. Of the various and continuously changing techniques of mass spectrometry, we selected the most widely used method, collision induced dissociation (CID), to demonstrate the principle of spectrum analysis. Tandem mass spectra peptides (MS/MS spectra, hereafter: spectra) are produced by mass spectrometers by fragmenting peptides (typically up to 20-30 amino acids in length) at given positions of the peptide chain. In theory, fragmentation results in a series of peptide fragments of which two types, the b and y ions are used most frequently for identifying peptides. A theoretical fragmentation pattern can be calculated based on the sequence alone, this is called the theoretical spectrum, the x axis is the molecular mass of the fragment peak (more exactly the mass over charge ratio m/z of the fragment ion) the y axis is proportional to the quantity of the ions. Since predicting the intensity *in silico* has been proven difficult, the intensity in the theoretical spectra are set to unit. If a post-translational modification (PTM) occurs at one position (amino acid) of the peptide (say at the M residue), than the (m/z) of all fragments in which this position is included will be shifted by a value corresponding to the modification. This will result in an altered theoretical spectrum [**?**].

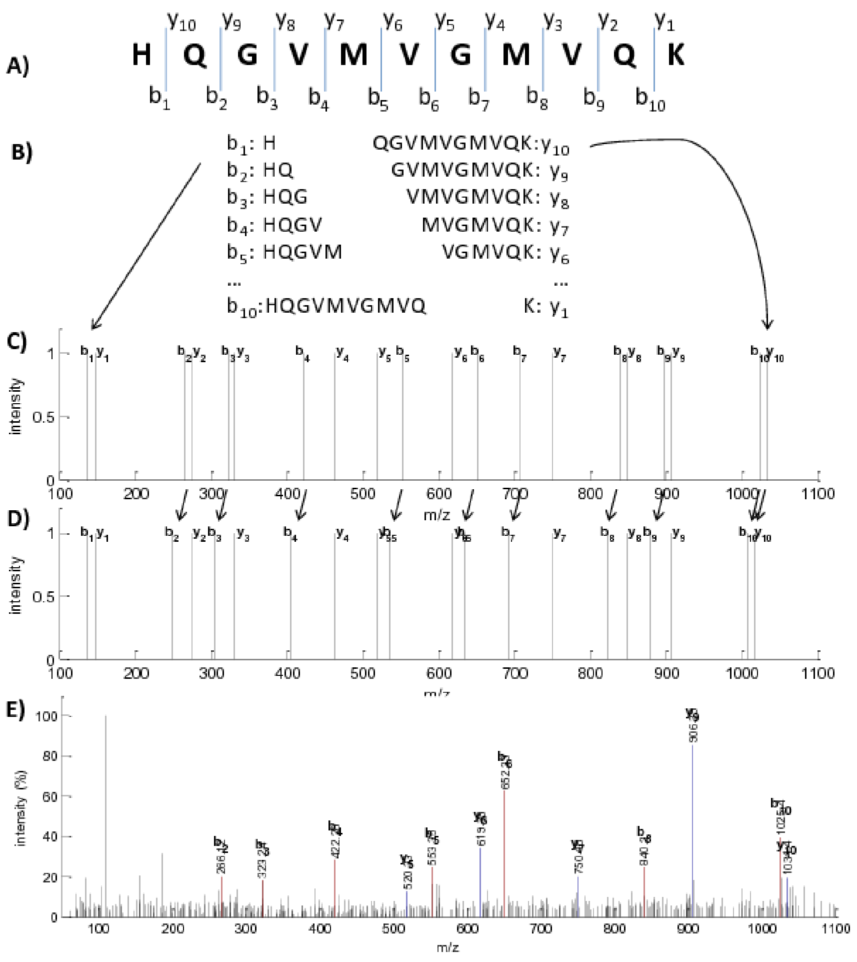

Fig. 1. A simplified outline of peptide fragmentation, shown on the example peptide HQGVMVGMVQK A) In the mass spectrometer, peptides are fragmented at certain points of the backbone. B) In theory, the resulting b and y ions form a regular series. C) This series is represented as a theoretical spectrum. D) If a modification is introduced, say at the 2nd residue (Q) of the sequence, all b and y ions that contain that residue will be shifted by a value corresponding to the molecular mass of the modification (see arrows). In this example we show the effect of deamidation at Q in position 2, which decreases the molecular mass by 17.03 Daltons (small negative shift indicated by arrows). E) An experimental spectrum of the same peptide recorded by a CID mass spectrometer. The annotated peaks correspond to the $b - y$ fragment ions. Some $b - y$ ions e.g. b_1, y_9 are not observed in this example, all other peaks can be considered noise.

Eventhough experimental spectra, produced by the mass spectrometer may contain only some of the peaks predicted by the theoretical spectra, the peak intensities (y axis) vary in a broad range and the spectrum is always laden with noise peaks, a classical numerical similarity measure should be able to tell if the experimental spectrum is closer to the modified or to the unmodified spectrum, even if the overlaps of noise peaks with modified peaks can obscure the results.

One of the simplest similarity measure for two spectra (an experimental and a theoretical spectrum) is the so called shared peak count (SPC), which is the number of peaks that are on the same position (m/z) within a given tolerance range, called (fragment) ion tolerance. Formally, for an experimental spectrum q and theoretical spectrum t, the number of shared peaks can be computed as,

$$SPC(q,t) = |\{(q_i, t_j) : |q_i - t_j| < \delta\}|, \tag{1}$$

where a_i is the location (m/z) of ith in the spectrum a and δ is the ion tolerance. In the case of PTMs, the algorithms must take into account the mass difference of the precursor ion and the resulting fragment ions, as the algorithms typically compare experimental and theoretical spectra coming from precursors with similar masses. Since we do not know which PTM to expect we need to produce theoretical spectra for all of them. There are many different kinds of protein modifications (the Unimod database lists over 500 types of them `www.unimod.org`) and in most cases the modifications are only partial i.e. both the modified and the unmodified amino acid can be present. The result is a combinatorial explosion which makes the accurate identification of PTMs an especially challenging problem. For instance, if we allow 5 of the 10 most frequent modifications to occur in a peptide, the search space grows 3 orders of magnitude, but the growth is more dramatic if instead of 10 types of modifications we wish to consider all of roughly 500 known types [**?**].

3 Related work

There is a large body of literature on the computational identification of PTMs in MS/MS spectra (for excellent reviews see [**?**]). The methods fall in 3 large categories:

– Targeted PTM identification. In this case, the user/experimenter has to define the types of PTMs and input these as parameters to the program.
– Untargeted PTM identification. This approach uses a full list of known modifications as a vocabulary, such as UniMod `www.unimod.org` or ResId `http://www.ebi.ac.uk/RESID/`.
– Unrestricted PTM identification. (also called de novo or blind PTMs identification). Here, no assumptions are made about the PTMs, but all shifts that obey the chemical rules (and are recurrent in a dataset) are considered as potential PTMs. Here, novel PTMs can be identified, but unfortunately the statistical significance of the hits are often low, especially when more then one PTM per peptide is allowed.

In practice, search engines running on single CPU computers can handle up to 4-5 modifications per peptide, above this limit the search space becomes too large and analysis often becomes too time-consuming [**?**]. From the many programs available we mention two:

MS-Alignment. The Pevzner group has proposed a model for unrestrictive PTM identification that seeks to identify an optimal alignment between the experimental and the theoretical spectrum that maximizes the overlap of the peaks between the two spectra via peak shifting [**?**]. One shift represents one PTM and the size of the shift will represent the molecular mass of the PTM. The method is analogous to finding an optimal edit distance between two strings. The program is freely available and an online web server can be found at: `http://proteomics.ucsd.edu/LiveSearch/`.

PILOT_PTM.(Prediction via Integer Linear Optimization and Tandem mass spectrometry) uses a binary integer optimization model for finding PTMs that best match the experimental data [**?**]. This is an untargeted method i.e. PILOT_PTM uses a vocabulary of PTMs. Given a template peptide sequence of amino acids, the model will seek to determine the optimal set of modifications among a "universal" list of PTMs. Then the obtained binary linear model is solved by relaxing the variables and cutting plane techniques.

We describe an alternative approach in which the PTM search space is mapped to a tree structure and finding an optimal set of PTMs is reduced to a tree traversal problem. The resulting tree is still too big for the classical, exhaustive tree traversal algorithms (such as depth-first search, breadth-first search [**?**]), so we developed a greedy traversal algorithm.

4 Method: The PTMSearch algorithm

The PTMSearch algorithm compares an experimental spectrum to a peptide sequence in order to identify PTMs untargeted way. Briefly, the algorithm generates all possible modifications of a peptide sequence and selects the set of modifications for which the precursor mass of the modified peptide is within a small tolerance of the precursor mass of the spectrum. If there is more than one such set of PTMs, the algorithm selects the one that shares the maximum number of peaks with the experimental spectrum (eqn 1). Since the search space is prohibitively large, PTMSearch uses a greedy approach and speedup techniques in finding the optimum.

Let q be an experimental spectrum and $p = a_1a_2 \dots a_n$ be a peptide sequence and let's denote the precursor mass of q and p by $PM(q), PM(p)$ respectively. Let L_a be a list of modification masses for the amino acid a and $n_a = |L_a|$ denote the number of the elements in the list. A modification mass is the mass difference that the amino acid gains or looses due to the molecular modification. For example $L_c = \{-17.0265, 47.9847, 57.0215, 71.0371, \dots\}$ means the amino acid cysteine can loose 17.065Da (occurs via losing of ammonia from cysteine), can gain 47.9847Da (occurs via complete oxidation), can gain 57.0215Da (occurs via carbamidomethylation), etc. respectively. The molecular mass of the cysteine is 121.16Da which becomes 178.1815 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 modification at different occurrences in the peptide sequence (and in different peptides as well).

The basic idea is to generate all modified variations of peptide p and store them in a prefix tree, where a branch at the level i denotes PTM on amino acid a_{i+1}.

More formally, the tree node at the level i is a structure $v = \langle s, b, y, m, c\rangle$, where s is a score of the node, c is the number and m is the sum of the mass of the acquired modifications in the sequence $a_1 \dots a_i$. Finally the variables b and y store the masses (m/z) of the b and y fragment ions that correspond to the $a_1 \dots a_i$ and $a_{i+1} \dots a_n$ fragment ions respectively.

We recursively define the tree and the values stored in the node structures as follows: The root node is $\langle 0, 0, PM(q), 0, 0\rangle$. If $v = \langle s, b, y, m, c\rangle$ is a node in the tree at the level $i \quad (0 \le i < n)$ then the node $v_0 = \langle s', b + m_{a_{i+1}}, y - m_{a_{i+1}}, m, c\rangle$ is a child of the node v at level $i+1$, $m_{a_{i+1}}$ is the mass of the amino acid a_{i+1}, and $s' = s + \widetilde{s}(b') + \widetilde{s}(y')$, where

$$\widetilde{s}(x) = \begin{cases} 1 & \text{if } \exists j : |q_j - x| < \delta \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where q_j denotes the (m/z) location of the jth peak in the spectrum q and δ is the ion match tolerance. The nodes $v_j = \langle s', b + m_{a_{i+1}} + m_j, y - m_{a_{i+1}} - m_j, m + m_j, c + 1\rangle$ are children of the node v at level $i+1$, where $j = 1, \dots, n_{a_{i+1}}$, m_j is the mass of the jth modification in $L_{a_{i+1}}$. s' is defined the same as earlier. The node v_j at level $i+1$ represents that the amino acid a_{i+1} in the peptide p is modified with the jth modification from $L_{a_{i+1}}$. If the node v is at the level n, then node v is a leaf. Denote $w(v) = s$ as the score of the node v. A leaf $\langle s, b, y, m, c\rangle$ is called a goal leaf if $|PM(q) - PM(p)| = m$ 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.

We denote this computation tree T and the subtree of T rooted at a node v is denoted T_v. The number of the nodes in the tree T_v is denoted by $|T_v|$.

The score of the spectrum q is the maximum of the scores of the goal leaves, denoted $H(q) = \max\{w(v) \mid v \text{ is a goal leaf in tree } T\}$. This goal can be found with any kind of tree traversal methods like Depth-first, Breadth-first traversal algorithms. The modifications on the peptide then can be extracted from the path between the root and the best goal leaf.

Note that, all nodes at level $i \quad (0 \le i < n)$ correspond to the ith amino acids in p, all have the same $n_{a_i} + 1$ children respectively, and the tree is balanced and all leaves have the same depth. Thus, the number of the nodes in the tree is $|T| = \prod_{i=1}^{n}(n_{a_i} + 1)$, which makes the time complexity of the traversal algorithm impractical. Note that the size of the search space does not depend on the experimental spectrum q. In the next subchapter we define pruning techniques in order to maintain the run time polynomial and appropriate for real applications.

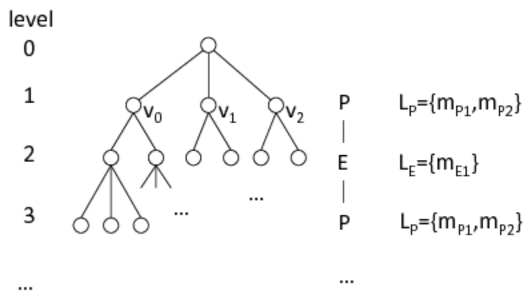

Fig. 2. Sketch of the computation tree for a peptide. PEP stands for the peptide sequence starting from above. The nodes at the level i represent the $a_1 \dots a_i$ peptide fragments modified in all possible ways.

4.1 Speedup techniques

In this section we present various speedup solutions that prune the search space and make the algorithm suitable for real-life applications.

Limiting the number of PTMs If we restrict the number of the PTMs on a peptide to K, then a node $\langle s, b, y, m, c\rangle$ can be deleted from the tree if $c > K$. Thus, the number of the nodes $|T'|$ in the pruned tree T' can be estimated by the following formula

$$m^K \binom{n}{K} \le |T'| \le M^K \binom{n}{K} \le \underbrace{\frac{M^K}{K!}}_{\text{const.}} n^K, \tag{3}$$

where $m = \min_a\{n_a\}$ and $M = \max_a\{n_a\}$, which gives that the number of the nodes in the tree T' is $O(n^K)$.

Figure 3 plots a distribution of the number of the modification per amino acids in our PTM database. See more details about PTM dataset we used in the Experimental results chapter. There is one amino acid that can be modified by 32 different PTMs, so $M = 32$ and there are 5 amino acids that cannot be tagged by any modification, so $m = 0$. We can say loosely that the expected number N_L of modifications per amino acid is around 5 or 6.

Greedy Tree Traversal In principle the tree can be traversed by any of the known tree traversal algorithms. Here, we present a Greedy Tree Traversal (GTT) algorithm that inserts a node to a queue Q when it is visited at the first time, deletes when all of its children have been visited, and continues the search from the node with the highest score in the Q. When the size of Q exceeds a certain limit Q_T, the node with the lowest score is deleted along with the corresponding subtree.

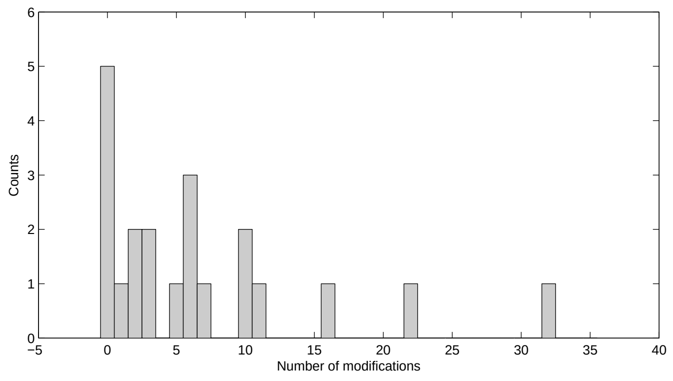

Fig. 3. Distribution of the number of PTMs per amino acids in our PTM database.

The queue Q can be implemented as a priority double ended queue, where the ordering operator of elements is based on the relation "$<$" over the score of the nodes. In this case, a new node is inserted at the end of the nodes with equal scores in Q. This algorithm can be described best by the following pseudocode.

Algorithm 1 Greedy Tree Traversal (GTT)

```
input: Tree $T$
output: best goal (or NULL if there is no goal leaf)
1 put($Q$,root);
2 while $Q$ is not empty
3       $v$ = front($Q$);
4       create a non-visited child $v_j$ of node $v$;
5       if all children of $v$ has been visited then pop_first($Q$);
6       if $v_j$ is a goal leaf then update best goal;
7       else put($Q$,$v_j$);
8       if size($Q$) > $Q_T$ then pop_last($Q$);
9 return best goal (or NULL if there is no goal leaf);
```

It may happen that the true goal leaf is eliminated from the tree by deleting a node in the 8th code line. Now, we give an estimate about the probability of eliminating the true goal under the following assumptions: The probability of a node v matches to a peak by chance (i.e. the score of a child of v is increased due to one of the fragment ions match with a noise peak) is $p = 2 * m * \delta / PM(q)$, where m is the number of the peaks in q. We assume that the peaks are evenly and independently distributed in the experimental spectrum. Let p_e be the probability of that a (non-noise) peak is missing from the spectrum independently from other peaks (because either it was not observed or was filtered out in a preprocessing step).

Theorem 1. *Using the assumptions and parameters above the probability $P(\epsilon)$ of eliminating the true goal from the search space is $P(\epsilon) = N_L \cdot p(K + \sum_{j=1}^{H} p_e^j)$, where $H = Q_T/(N_L)^K$, $Q_T > M$ is the bound of the size of the queue Q, and K is the limit of the PTMs on a peptide to be identified.*

Proof. First case: Let's assume the GTT is on the right path to the true goal at node v at level i, and the b_{i+1} and y_{n-i+1} peaks are not missing. GTT can leave the true path iff one of the children l of the node v hits a random match by chance before the right child is extended. In this case GTT will visit all nodes of the T_l because all nodes in T_l have greater weight than any other nodes in Q. If $|T_l| > Q_T$, then the node v will be deleted from the Q and the right path to the best node will be lost. Since the path from the root to the true goal has K branches, each has N_L branches, then the probability of this error $p_{\epsilon_1} = K \cdot N_L \cdot p$.

Second case: Let's assume GTT is on the right path to the true goal at node v at level i, v has a match, and the b_{i+1} and y_{n-i+1} peaks are missing. v has the unique and greatest score $w(v)$ in Q. ($w(v)$ must be unique, since v had a match and v's parent has smaller weight. If there is another node x such that $w(x) = w(v)$ in the Q, then GTT first traversals T_x, deletes x from Q, then returns back to T_v.) GTT starts visiting the nodes in T_v Breadth-first way until a node matches to a peak by chance. Q can store Q_T nodes of T_v, $H = Q_T/N_L^K$ levels. So, H right peaks must be missing from the spectra successively, before GTT starts deleting nodes of T_v from Q. The probability of that H consecutive peaks are missing is p_e^H. Thus, the probability of error of the second case is $p_{\epsilon_2} = p \cdot N_L \cdot \sum_{j=1}^{H} p_e^j$.

Then, the final probability of eliminating the true goal from the search space can be obtained by summing the cases since they can happen independently. Thus, we have

$$P(\epsilon) = p_{\epsilon_1} + p_{\epsilon_2} = K \cdot N_L \cdot p + p \cdot N_L \cdot \sum_{j=1}^{H} p_e^j = N_L \cdot p(K + \sum_{j=1}^{H} p_e^j). \quad (4)$$

□

Consequence 1. Deleting a node from the tree polynomially reduces the search space. If we decrease the Q_T threshold, more node will be eliminated however the error will grow only linearly.

Consequence 2. The probability $P(\epsilon)$ can lessen with more accurate resolution of the spectra, (smaller δ).

Consequence 3. The probability $P(\epsilon)$ can lessen via removing more noise peaks from the experimental data in order to decrease the chance of random matches.

Remark 1. This greedy approach can be considered as a special case of the $A^\star$ algorithm [**?**], where the distance-plus-cost function is defined $h(n) = g(n) + f(n)$ for a node n, where $g(n) = w(n)$ is the score from the root to the node n and $f(n)$ is an estimation of the score from the node n to a leaf, here $f(n) \equiv 0$, the constant zero function. Note that h is a monotone function.

Remark 2. Let $Q_T = \infty$. If Q is implemented as a stack, then GTT traverses the tree in a Depth-first manner. If implemented as a First-In-First-Out queue, GTT will function as a Breadth-first search.

Remark 3. Note that, if the query spectrum is compared to an incorrect peptide sequence, then it doesn't really matter whether or not we lose the best goal.

Fast Match heuristics In the process of database search, the experimental spectrum q is compared to peptide p (from the peptide database), if $|PM(q) - PM(p)| < \Delta$, where Δ is a precursor mass tolerance (usually Δ <1 Dalton).

PTMs can add or subtract several tens or hundreds Daltons with respect to the precursor mass of the experimental spectrum, thus the precursor mass tolerance needs to be widened. In this way a query is compared to a very large of number peptides that will substantially increase the execution time.

PTMSearch includes an additional filtering step we term Fast Match, wherein a theoretical peptide t is compared to the experimental spectrum q if $SPC(q, t) > F$, where F is a given threshold. If we set $F = 0$, this technique can miss the correct peptide if both the first and the last amino acids of the peptide carry PTMs because in this case all the fragment ions are shifted.

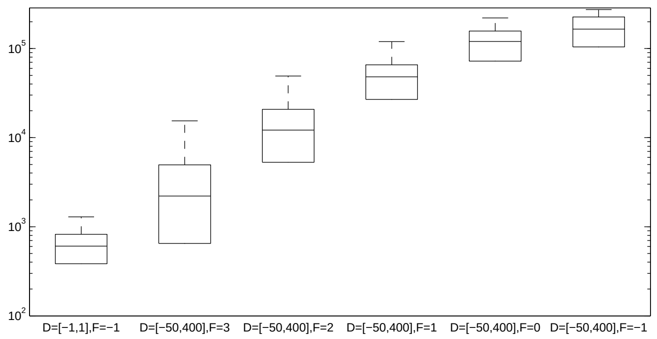

Fig. 4. Boxplot diagram about the number of the peptides to be processed per each spectrum.

Figure 4 shows the number of the peptides to be compared with the query, at various parameter settings. Simply put, widening of the mass range from D=[-1,1] to D= [-50,400] increases the number of the peptide comparisons by two orders of magnitude. However, the application of Fast Match can significantly reduce the number of comparisons. The data in the plot were calculated on the experimental dataset described in chapter 6.

5 Significance calculation of a hit

In the literature, the number of the random matches between two spectra are modeled either by Poisson or by hypergeometrical distributions [?,?]. The search for PTMs can be pictured as a generating all modified theoretical spectra, comparing them to the experimental spectrum, and picking the one with the maximum score. It is known that the distribution of the maximum of random numbers tends to an extreme value distribution [?]. Thus, we use an extreme value distribution to calculate the statistical significance (p-value) of the hit to measure the probability the hit occurred by chance.

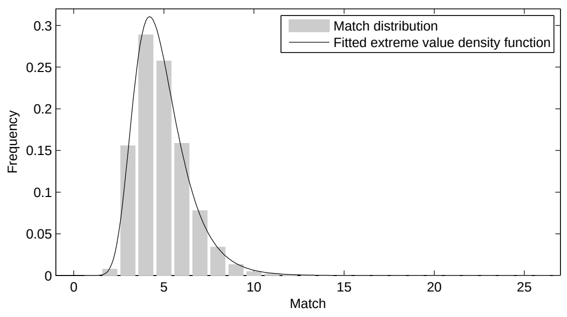

Fig. 5. Distribution of the matches on a real dataset. The solid line is a fitted extreme value probability density function on the experimental data. The number of the allowed PTMs is set to 3.

In Figure 5 we compared a real (experimental) spectrum against the decoy dataset, where the limit of the PTMs $K = 3$. The figure shows that the extreme value distribution fits the data well.

6 Experimental results

For the experimental tests we used MatLab (version R2010b)as a frame to load the data and evaluate the results. PTMSearch (and GTT) was implemented in C++ and used as a module in MATLAB. Computational experiments were carried out on a Linux cluster with 20 nodes and one frontend node, each with 2.2Ghz CPU and 1GByte memory.

IPIHuman (version 3.71) downloaded from `http://www.ebi.ac.uk/IPI/IPIhelp.html` was used as the protein sequence database (86309 sequences). These sequences were cut into peptides using the tryptic digestion rule in the Cleavage routine of Matlab's Bioinformatic Toolbox. This *in silico* digestion resulted in 727707 unique peptides. For the calculation or theoretical fragment ions we used the program THEOSPEC 2.0 (available at: `http://sourceforge.net/projects/protms/files/theospec/`)

The list of modifications were downloaded from the OMSSA project site (`http://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP_DOC/lxr/source/src/`

`algo/ms/omssa/mods.xml`) on Dec. of 2010. This file contains 207 records of PTMs. We have considered only chemical modifications of amino acids. Therefore the current iteration does not take into account N- and C-terminal modifications, such as might be found by the removal of the initiator methionine. Thus, we used PTM records that are tagged by "MSModType Value = 0" in the mods.xml file. In the end we obtained 142 PTMs. The distribution of the number of the modifications per amino acids can be found in Figure 3.

The precursor mass tolerance was set to $\Delta = 0.4$, the fragment ion match tolerance was $\delta = 0.4$, and PTMs mass range was set to $D = [-50, 400]$ Da. The Fast Match parameter was $F = 1$. The size of the queue Q was $Q_T = 50 * \log(n)$ based on the author's observations.

For statistical significance calculation we used a decoy dataset consisting theoretical spectra calculated from reversed peptide sequences [**?**,**?**]. During database search, the query was first compared with the theoretical peptide dataset, and then with the decoy dataset [**?**]. For the evaluation, we plotted the number of the positives as a function of the False Positive Rate (FPR) at various thresholds.

6.1 Results on a toy datasets

We generated *toy dataset* by randomly selecting 200 peptides from the theoretical database and generating the corresponding theoretical fragment ion peaks. Then, we randomly added 2-3 PTMs from our list to each of the peptide spectra. This dataset was then searched against the peptide database with PTMSearch. We found that our method found all the generated modifications (Data not shown).

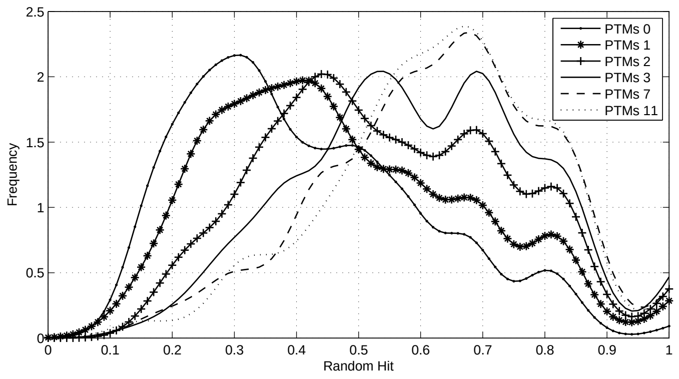

Fig. 6. The distribution of the random hits as a function of the PTMs limit K on the toy dataset. Here the number of the matches were divided the length of the peptide as a normalization.

For an illustration we examined how the limit K of the PTMs interferes with the expected number of the random matches. We found that, as expected, increasing the limit of PTMs increases the number of the random matches, which in turn lessens the statistical significance of the true hit. This is shown in Figure 6. However, this effect could be avoided by increasing the instruments accuracy, thereby decreasing the ion match tolerance δ parameter.

6.2 Calculations on real data

The Aurum dataset has been designed to provide a standard, manually curated dataset of experimental spectra for comparison and evaluation of newly developed algorithmic approaches and is freely available [**?**]. The dataset contains 1834 MS/MS spectra obtained on 246 known, individually purified and trypsin-digested protein samples recorded on MALDI TOF/TOF instruments. The spectra were preprocessed as described in OMSSA [**?**].

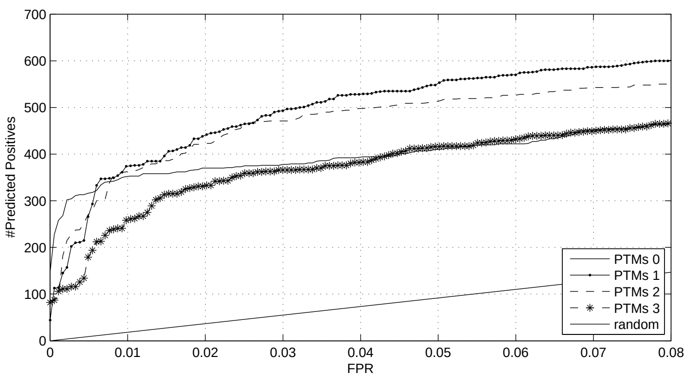

Fig. 7. The number of identified peptides at various PTM limit as a function of FPR.

The Figure 7 show a plot of the number of the positively predicted spectrum as a function of the FPR on the Aurum dataset. Table 1 shows detailed results obtained with PTMSearch at FPR=1% and FPR=5% levels. Columns show the number of the identified peptides at various PTM limits, while rows show the distribution of singly-, doubly-, and triply-modified peptides, respectively. For example, PTMSearch has identified 514 peptides at 5% FPR level, of which 243 peptides contained no modifications, 180 peptides were tagged by exactly 1 PTM, and 91 peptides were tagged by exactly 2 PTMs.

PTMSearch reported an unexpectedly large number of propionamide modifications of cysteine in the Aurum dataset. Including triply modified peptides PTMSearch found 138 cysteines that carry this Acrylamide adduct. We then used X!Tandem [**?**] to perform targeted search for this modification and could

Table 1. The number of the peptides identified with PTMSearch at 1% and 5% of FPR respectively.

FPR	PTMs	0	1	2	3
1%	sum	352	374	362	258
	0	352	240	203	138
	1		134	113	62
	2			46	28
	3				30
5%	sum	410	553	514	416
	0	410	295	243	194
	1		258	180	105
	2			91	42
	3				75
Time(min)		11	120	754	1281

1281 minutes is approx. 21 hours.

only validate 58 propionamide modifications. This data further supports the importance of untargeted search for PTM discovery.

This filtered Aurum dataset was submitted to the online version of MS-Alignment at `http://proteomics.ucsd.edu/LiveSearch/` [?]. Here the precursor mass tolerance and the ion tolerance was the same as in the PTMSearch case. The database was the IPI.Human (v.3.81), The modification mass range was set to $D = [-200, 200]$. Results are presented in the Table 2. Note that, the run time cannot be compared to PTMSearch run time since neither the system architecture nor the load of the MS-Alignment server were known. Limit of PTMs was set to 1. MS-Alignment has identified 287 peptides with at most one PTM, while PTMSearch identified 553 peptides under same conditions.

Table 2. Results obtained with MS-Alignment. PTMs limit is 1.

FPR	PTMs 0	PTMs 1	sum	Time
5%	196	91	287	5.5 days

7 Discussion, future plans

Protein identification based on tandem mass spectrometry and database search is fast growing field in continuous need of new computational approaches. The untargeted PTM search algorithm described targets one of the most difficult problems in this field, identification of post-translationally modified proteins.

The novel features of the approach are the tree-representation of the search space, the greedy heuristics based on chemical rules and the speedup techniques that allow one to bring down the computational speed to a manageable level.

We investigated the computational properties of the heuristics and showed that the limits imposed on PTMs decrease the search space to polynomial. We gave an estimate to the error introduced by the greedy heuristics and presented a significance calculation scheme appropriate for this algorithm.

The current work was undertaken in order to demonstrate the feasibility of the tree-traversal approach. There are many further improvements possible, for instance one can use spectrum similarity measures instead of the simple SPC employed here. The current algorithm is apparently not substantially faster as compared to other PTM identification methods, however an exact comparison could not be made because of code availability problems. Nevertheless, a substantial speedup can be expected upon massive parallelization or porting the algorithm to GPU processors, since tree-traversal algorithms are known from this perspective. We plan to include the algorithm into a free online search engine where those computational aspects that were outside the scope of this work (data filtering, protein inference etc.) will also be dealt with.