PepTiger: Search Engine for Error-Tolerant Protein Identification from *de Novo* Sequences

Irina Fedulova[1], Zheng Ouyang[2], Charles Buck[3] and Xiang Zhang*[,3]

[1]*Department of Computational Mathematics and Cybernetics, Moscow State University, Russia*

[2]*Weldon School of Biomedical Engineering, Purdue University, USA*

[3]*Bindley Bioscience Center, Purdue University, USA*

Abstract: In recent years a number of *de novo* sequencing software products became available providing possible partial or complete amino acid sequence tags for MS/MS spectra of peptides. However, for a variety of reasons including spectral chemical noise and imperfect fragmentation these sequence tags almost always contain errors. Additional difficulties arise from actual protein sequence variation and post-translational modifications. We present a search engine named PepTiger which is capable of correctly matching *de novo* sequence tags with errors to protein sequences in a protein database. The algorithm is based on approximate string matching followed by a novel scoring procedure which takes into account mass differences and the string distance between *de novo* sequence and matched peptides and similarities between theoretical and experimental MS/MS spectra. Comparison of PepTiger with other protein identification software shows that PepTiger is better able to assign *de novo* sequence tags with errors to the correct peptide sequences.

INTRODUCTION

Protein identification is a major focus in proteomics. In recent years, tandem mass spectrometry has become a standard tool for protein identification. Typically, proteins are digested with enzymes like trypsin into short peptides by breaking at specific sites along the backbone of the protein. At the next stage, individual peptides can be isolated and further fragmented by collision-induced dissociation (CID) to produce information about the mass-to-charge ratios of resulting fragment ions (MS/MS of parent peptide). In a single experiment, many charged fragments are formed by CID of multiple copies of the same peptide. Since peptides usually break at a peptide-bond when they are fragmented by CID, the resulting spectrum contains information about amino acid composition of peptide. However, this information is typically incomplete due to the chemical structure of the peptide to be sequenced. Additionally, some spectral peaks may result from breakage of non-peptide bonds and from other chemical noise. Finally, co-eluted peptides with similar m/z value further complicate the tandem spectrum. All the above provide a great challenge for reliable interpretation of MS/MS spectra of peptides.

Two approaches are deployed to interpret experimental MS/MS data, database searching and *de novo* sequencing. In the database searching approach a protein database is used to find a peptide for which a theoretically predicted spectrum best matches experimental data. These algorithms differ in the scoring schemes which are used to evaluate the detected matches between candidate peptides and the given experimental spectrum. Most algorithms incorporate a statistical treatment to judge whether the search results are significant. The most popular software products using this approach are Mascot [1] and SEQUEST [2].

*Address correspondence to this author at the Bindley Bioscience Center, Purdue University, USA; E-mail: zhang100@purdue.edu

A significant drawback of database searching algorithms is that these methods rely on a peptide mass filter, where candidate peptides are selected from the database for comparison only if their calculated mass equals the experimentally determined mass of parent ion, within certain error tolerance. The algorithms will therefore fail to find a correct answer if the sample peptide differs from a canonical version in a database due to mutations, posttranslational modifications (PTMs) or database sequence errors, because the calculated mass from the database sequence may no longer match the measured mass. A list of peptide candidates that do not include the correct peptide will provide incorrect answers [3]. To overcome this drawback, an exhaustive search approach was suggested [4] where a virtual database of all modified peptides from a small set of potential modifications is generated and experimental spectra are matched against this enlarged database. This procedure can be extremely computationally expensive. Another technique is to search for peptides without parent mass filter. In this approach, a very short peptide sequence (sequence tag) recovered from the raw spectrum is used to filter the candidate peptides from the database [5-8]. Unfortunately, this method is not suitable for the identification of modified peptides [9] and cannot reliably detect multiple modifications to the same peptide [10].

The second major approach for protein identification is *de novo* sequencing, which is based on utilization of information about amino acid composition of peptides deduced directly from MS/MS spectra. A number of algorithms for reconstruction of partial or complete sequence of a peptide from its MS/MS spectrum (*de novo* sequencing) have been developed [11-17]; among the most popular are Lutefisk [13-14], Sherenga [15], and PEAKS [16]. After the predicted sequences are obtained for peptides by the *de novo* software, a sequence-based database search may be performed. Adapted versions of general homology search tools such as MS-BLAST [18], MS-Shotgun [19] and FASTS [20] can be used in this step. These programs are based on efficient se-

1874-3838/07

quence alignment algorithms and use a modified mutation matrix to score matches with possible sequence variations.

Unfortunately, *de novo* sequences are likely to contain errors due to presence of chemical noise in spectra, the imperfect fragmentation of the peptides and the mass accuracy of current mass spectrometers. Another source of *de novo* sequencing errors is that the mathematical models used in *de novo* software are greatly simplified to minimize the computational requirements [9]. Typical *de novo* errors are simply swaps of adjacent residues or replacements of fragments by another with the same mass. In addition, incomplete data may lead to replacement of several light residues by one or more heavier residues and vice versa. Longer segment replacements are also possible, though less likely. Use of the general homology search programs MS-BLAST, MS-Shotgun and FASTS is therefore limited, as these do not take *de novo* sequencing errors into account. Nevertheless, there are sufficient similarities between sequence candidates that an appropriately modified homology-based database search program should be able to perform sequence database searches using this information [13].

Error-tolerant search engines must be used to differentiate sections of the sequence that are inappropriately assigned by *de novo* software from actual amino acid mutations and posttranslational modifications [21]. At present, a few tools are able to do so; such as CIDentify [13], OpenSea [21] and SPIDER [9]. CIDentify is based on the available FASTA code that is altered to accommodate some typical *de novo* sequencing errors. With this tool the best initial alignment of the candidate sequence to the database is found and then evaluated by consideration of cases where the mass of one query residue is equivalent to the mass of two database residues, the mass of two query residues is equivalent to the mass of one database residue, or the mass of two query residues equals the mass of two different database residues. In each case, replacements are not penalized in final scoring. CIDentify therefore allows replacements of segments with a maximum length of 2 residues.

The OpenSea tool employs a mass-based approach to sequence alignment. *de novo* and database sequences are interpreted as masses of residues, and the masses, rather than the amino acid codes are compared. OpenSea resolves both *de novo* errors and homology mutations, but only if they do not occur at the same positions [9]. Additionally, OpenSea allows only one consecutive substitution, and the alignment process is terminated if additional consecutive substitutions are required to make a match. This constraint can preclude identification of true matches.

The SPIDER [9] software package for protein identification from *de novo* sequence tags accounts for both homology mutations and *de novo* sequencing errors. Moreover, it allows the mutations and errors to occur at the same positions along the sequence. Two types of distances between sequences are introduced: distance that measures the *de novo* sequencing error, and edit distance that measures homology mutations. Total distance is then defined as sum of these two distances. The sequence tag search approach in SPIDER is to find a peptide sequence from the database, such that total distance is minimized. A dynamic programming algorithm computes the total distance based on specific cost functions for *de novo* errors and for common editing operations handling homology mutations. Cost values for correctly assigned letters and incorrectly replaced segments are based on empirical assumption that 80% of the letters can be correctly assigned by the *de novo* software. However, for low-quality spectra this value can be overestimated [16]. At the same time, replacements of segments of arbitrary lengths are allowed. Additionally, only costs for replacements of segments not longer than 3 letters are computed exactly, while costs for longer segments are approximated. Cost values for homology mutations are based on well-known BLOSUM90 [22] table. SPIDER has four match modes with certain assumptions about the data or the database. The simplest match mode is SPIDER exact match, assuming that *de novo* sequence has no error except that L and I, Q and K are exchangeable. Segment match mode allows replacements of segments of arbitrary length, but assumes no homology mutations. In the non-gapped homology match mode it is assumed that all homology mutations are substitutions (no insertion/deletion is allowed), mutations and *de novo* sequencing errors do not exist simultaneously in the same block, and there are no segment replacements with length greater than three. Homology match mode is the most general mode allowing both *de novo* sequencing errors and mutations of proteins.

Both OpenSea and SPIDER allow replacements of segments of arbitrary length with the same mass. However, the number of possible fragments having the same mass increases with length and total weight, leading to increasing possibility of matching two fragments by accident, and therefore, increasing of the false positive rate.

The objective of this work is to develop a search engine that is able to perform protein identification from *de novo* sequence tags that may contain errors. The developed search engine has been implemented into software PepTiger using C++. This alignment method adapted for error-tolerant search is described along with our novel scoring scheme and benchmarking results with our search engine on experimental MS/MS data presented.

ANALYSIS METHOD

Background and Terminology

Similarity searching in a protein database can be reformulated in terms of approximate string matching. This problem has received a lot of attention in computer science for a wide range of applications, but there is a paucity of efforts to adapt general algorithms to the specific field of protein identification. Protein and peptide sequences can be represented by strings where each amino acid residue is encoded by letter from a twenty-letter weighted alphabet Σ. To measure the similarity of two amino acid sequences the distance function between strings should be defined.

In the following discussion, we use capital letters *S*, *A*, *B* to represent strings over alphabet Σ. For any string *S* we denote its length as $|S|$. We also denote $S[i]$ the *i*-th character of *S*, for an integer $i \in 1..|S|$. $S[i..j]$ is a substring of *S* beginning at *i* th character and ending at *j* th character. Superscript *R* denotes the reverse string: for example if *S* = **'abc'**, then S^R = **'cba'**.

The edit distance $d(S_1, S_2)$ between strings S_1 and S_2 is defined as minimum total cost of editing operations needed to transform S_1 to S_2. Typical editing operations are: substitution of one character by another, deletion and insertion of characters. Under these three operations, edit distance is often called Levenshtein edit distance [23]. Since insertions and deletions are allowed, edit distance allows one to compare strings of different lengths. The cost of each edit operation can be defined individually for the particular application. The annotations C_{sub}, C_{ins}, C_{del}, represent costs of substitution, insertion and deletion respectively. The most popular choice for costs is unit cost for each operation. For instance, under this cost the edit distance between **'abc'** and **'ad'** equals to $d(\text{'abc'},\text{'ad'}) = 2$.

An enrichment of the edit distance suggested by Damerau [24] includes transpositions (swaps) of adjacent characters (i.e. a substitution of the form **'ab'** → **'ba'**). The cost of this swap operation is denoted as C_{swap}. We will use subscript D to denote Damerau distance d_D.

The first and most flexible algorithm for edit distance computation is based on dynamic programming [25]. It allows to use arbitrary costs for editing operations. However, this algorithm is also the most inefficient. The value $d(A, B)$ for two strings of lengths $m = |A|$ and $n = |B|$ can be computed by filling a $(m+1)\times(n+1)$ dynamic programming matrix D, in which the cell $D[i, j]$ contains the value $d(A[1..i], B[1..j])$. Detailed description of both algorithms for Levenshtein and Damerau can be found in [29].

The sequence of editing operations transforming S_1 to S_2 can be written in the form of alignment. Alignment is a set of two strings $\tilde{S}_1$ and $\tilde{S}_2$ of even length over an alphabet $\tilde{\Sigma} = \Sigma + \{\text{'-'}\}$. Strings $\tilde{S}_1$ and $\tilde{S}_2$ are derived from S_1 and S_2 by insertion of empty characters **'-'**. Alignment is usually written as a two-row matrix, where $\tilde{S}_1$ is written in the first row, and $\tilde{S}_2$ in the second row. In such a representation, each column represents an edit operation: substitution of the character $S_1[i]$ by the character $S_2[j]$ $\begin{pmatrix} S_1[i] \\ S_2[j] \end{pmatrix}$, deletion of the ith character of S_1 $\begin{pmatrix} S_1[i] \\ \text{'-'} \end{pmatrix}$ or insertion of the jth character of S_2 $\begin{pmatrix} \text{'-'} \\ S_2[j] \end{pmatrix}$. We assume that no column contains empty characters in both rows, so that the alignment may have at most $|S_1| + |S_2|$ columns. The cost of alignment is the total cost of its editing operations. Optimal alignment corresponds to the edit (minimum) distance between strings. Optimal alignment is not unique despite the fact that edit distance for every two strings is unique.

For example, two different alignments for strings **'abc'** and **'ad'** are shown in Equation (1).

$$A_1 = \begin{pmatrix} \text{'abc'} \\ \text{'ad -'} \end{pmatrix} \quad A_2 = \begin{pmatrix} \text{'abc'} \\ \text{'a - d'} \end{pmatrix} \tag{1}$$

An optimal alignment can be recovered by tracing back from the cell $D[m, n]$. At each cell $D[i, j]$ adjacent cells $D[i-1, j-1]$, $D[i-1, j]$ and $D[i, j-1]$ are checked, determining which cell gave rise to the current cell, and so on back to $D[0, 0]$. At some cells we can have multiple choice of substitution, insertion or deletion. Final alignment depends on which option is given the highest priority.

PepTiger Distance

Due to the possibility of *de novo* errors a common edit distance does not fit well for protein similarity search. To accommodate typical *de novo* substitutions of segments with equivalent masses, an extension to Damerau edit distance has been defined in PepTiger.

Denote as h the length of an aligned segment with possible *de novo* sequencing error, which is tolerable by PepTiger. Algorithm 1 explains computation of PepTiger distance d_P. Algorithm is based on the same dynamic programming method as algorithms for Levenshtein and Damerau distance calculations. However, in addition to common deletion, insertion and substitution operations, an operation of block substitution of length up to h is considered. Both strings are searched for fragments of possibly different lengths, but such that their masses are close within accuracy ϑ (lines 18-26). If such substitution block is found, its cost is compared to other possible costs of editing operations (lines 28-30).

Relative edit distance $\alpha(S_1, S_2)$ is used to compare peptide sequences of different lengths and defined as

$$\alpha(S_1, S_2) = \frac{d(S_1, S_2)}{\max\{|S_1|, |S_2|\}} \tag{2}$$

where d is either Levenshtein, Damerau or PepTiger distance. This equation satisfies $0 \le \alpha \le 1$.

Once distance function is defined we can formulate the *de novo* error-tolerate protein identification problem using *de novo* sequence candidate.

Problem 1. Given the protein database $\{T^q, q = 1..Q\}$, *de novo* sequence candidate P and threshold value ε, return the set of all q, i^q, j^q such that $\alpha_P(T^q[i^q..j^q], P) < \varepsilon$.

That is, identify all proteins and peptides from the protein database that match P in the sense of a defined distance function. From the problem statement it follows that PepTiger can work only with complete *de novo* sequences without gaps.

There exists a number of efficient algorithms for Levenshtein edit distance computation and some for

Algorithm 1

1. $D[0, 0] := 0$
2. **for** $i := 1$ **to** m **do**
3. $\quad D[i, 0] := D[i-1, 0] + C_{del}$

4. **end for**
5. **for** $j := 1$ **to** n **do**
6. $D[0, j] := D[0, j-1] + C_{ins}$
7. **end for**
8. **for** $i := 1$ **to** m **do**
9. **for** $j := 1$ **to** n **do**
10. **if** $A[i] = B[j]$
11. $Sub := 0$
12. **else**
13. $Sub := C_{sub}$
14. **end if**
15. $D[i, j] := \min \begin{Bmatrix} D[i-1, j-1] + Sub \\ D[i-1, j] + C_{del} \\ D[i, j-1] + C_{ins} \end{Bmatrix}$
16. $foundBlock :=$ false
17. **if** $Sub \neq 0$
18. **for** $h_a := 1$ **to** h **do**
19. **for** $h_b := 1$ **to** h **do**
20. **if** $\left|\text{Mass}(A[i-h_a..i-1]) - \text{Mass}(B[j-h_b..j-1])\right| < \vartheta$
21. $foundBlock :=$ true
22. $iBlock := i - h_a$
23. $jBlock := j - h_b$
24. **end if**
25. **end for**
26. **end for**
27. **end if**
28. **if** $foundBlock =$ true **and** $D[iBlock, jBlock] + C_{block} < D[i, j]$
29. $D[i, j] := D[iBlock, jBlock] + C_{block}$
30. **end if**
31. **end for**
32. **end for**
33. $d_P = D[m, n]$

Damerau edit distance computation. Most are based on efficient ways of dynamic programming matrix computation either by filling only necessary parts of a matrix [26] or by exploiting the intrinsic parallelism of the computer when it works on bits [27, 28]. Both approaches are based on monotone properties of a dynamic programming matrix due to special choices for editing operation costs (unit cost for every operation; see [29] for a survey of current techniques for approximate string matching).

However, dynamic programming matrix for PepTiger distance does not hold monotone property due to possible block substitutions. Due to this non-monotone behavior of PepTiger distance it is impossible to define a threshold for selection of candidates for alignment reconstruction. Therefore we would have to check every peptide. But the main constraint is that for the protein identification problem we may set editing operation costs based on mutation matrix, for example BLOSUM90 [22]. Fast algorithms for edit distance computation require unit cost for edit operations. This makes applying efficient techniques for approximate text searching mentioned above unfeasible. Instead, we extract all possible substrings (peptides) from the text (protein) and compute PepTiger distance separately for each substring with the simple dynamic programming method described. That is a large number of small dynamic programming matrices are calculated followed by alignment recovery and further processing.

Reduction of the Alignment Model

The number of all possible substrings in the protein database is too large for complete evaluation of each PepTiger distance for the query *de novo* sequence. Some assumptions are made to narrow down the search space. First, a researcher may fix the maximum number of possible insertions and deletions thereby limiting candidate peptide lengths. A filtering technique is also employed whereby a potential match is first detected with a simple procedure and then verified with more complex distance computation. Specifically, for a potential match we require an exact match between short substrings of length l (l-mers) in the query and candidate database peptides. If such an exact match is detected, positions of the beginning and ending of the potential match in the protein are determined and PepTiger distance calculated. This technique improves the search speed greatly with only a small sacrifice in sensitivity [9]. Efficient l-mer exact match detection is implemented by first building a tree of all the l-mers in the query and then threading text through the tree. Once the terminal leaf is reached, a potential match is detected.

As long as insertions and deletions are allowed, adjacent positions for the beginning and end of the potential match should also be checked.

The number of adjacent positions to check is defined by the total number of insertions and deletions (see Fig. **1**).

Scoring Scheme

The scoring scheme used in PepTiger is based on three components: a string component based on PepTiger distance f_P, a mass component f_M and a spectral component f_S.

(a)

X	X	X	X	A	B	C	**D**	**E**	**F**	G	H	S	X	X	X	X
					A	W	**D**	**E**	**F**	K	S					

$d_p = 4$

(b)

X	X	X	X	A	B	C	**D**	**E**	**F**	G	H	S	X	X	X	X
					A	W	**D**	**E**	**F**	K	-	S				

$d_p = 2$ if Mass(GH) = Mass(K)

(c)

X	X	X	X	A	B	C	**D**	**E**	**F**	G	H	S	X	X	X	X
				A	W	-	**D**	**E**	**F**	K	-	S				

$d_p = 0$ if Mass(BC) = Mass(W) and Mass(GH) = Mass(K)

Fig. (1). Sample peptide alignment procedure in PepTiger.

All components are normalized to a minimum value of 0 and maximum value of 1. Final scoring is determined as a weighted sum of all three components.

$$\text{Score} = w_P f_P + w_M f_M + w_S f_S \quad (3)$$

By adjusting the weights w_P, w_M, w_S, the researcher is able to fit a scoring scheme to a particular task. The only requirement for these weights is that the unit total sum: $w_P + w_M + w_S = 1$. Thus, scoring is normalized between 0 and 1 with larger values corresponding to better match quality.

The *string component* is based on PepTiger relative distance α_P, calculated in the previous step. To reduce the number of false positive results, a maximum length of exactly matched segments in the alignment is taken into account in the final scoring. The probability of an accidental occurrence of a long exact match is small. For example, consider two alignments with equal PepTiger distance $A_1 = (*...*...*..*)$ and $A_2 = (.........****)$. Here the dot denotes a match and asterisks denote a mismatch in the alignment. *de novo* errors which have been detected are regarded as a match. Alignment A_2 is more likely to be correct than A_1 if the last four mismatched characters represent a *de novo* error undetected in the computation of the PepTiger distance because the mismatch length is greater than h. Denote as L the length of alignment, the relative maximum length of matched segment is given by

$$\mu = \frac{\text{max length of matched segment}}{L} \quad (4)$$

A non-normalized string component is given by a formula $\tilde{f}_P = 1 - \alpha_P + \mu$. It is our claim that maximum value of $\tilde{f}_P = 2$, and its minimum value depends on the threshold value for PepTiger distance ε, length of alignment L and length l of exactly matched l-mer on which the potential match detection step is based: $\min(\tilde{f}_P) = 1 - \varepsilon + l / L$. The derivations of these formulae are available from authors upon email request. Therefore, normalized string component of scoring can be obtained by

$$f_P = \frac{\tilde{f}_P - \min(\tilde{f}_P)}{2 - \min(\tilde{f}_P)} \quad (5)$$

which satisfies $0 \le f_P \le 1$.

At this step, the researcher can also take into account protein digestion errors in the query by penalizing non-tryptic peptides. A small penalty value p can be subtracted from f_P. In reality, there are few peptides in a proteolytic digest without at least one correct terminus [30].

The *mass component* of scoring takes into account any difference in molecular weight of the matched database and experimental peptides:

$$\Delta m = MW_{\text{experimental}} - MW_{\text{matched\_peptide}}.$$

This mass difference should be small and within the range of instrument variation. However, it could be very large due to unknown and/or ignored modifications of the experimental peptide. PepTiger provides a list of possible chemical and posttranslational modifications (PTMs) [31, 32]. PepTiger supports two types of modification, fixed and variable. Fixed modifications are applied to every instance of the specified residue or terminus by simply changing its mass. Variable modifications may, or may not be present. By checking all possible variable modification sites in a database peptide sequence, theoretical masses with modifications are calculated and compared with the mass of experimental peptide. Once optimal variable modifications are identified, a special normalization function is applied to the mass difference according to Equation 6.

$$f_M = \exp\left(-\frac{\left(|\Delta m| - \sum_{i=1}^{n} p_i m_i \right)^2}{\sigma^2} \right) \quad (6)$$

where σ is parameter reflecting instrument tolerance (peak width), n is the maximum number of user defined variable modifications, p_i is the number of i th modification applied to the matched peptide sequence. If no variable modification is necessary, $p_i = 0$. However, the number of possible conditions grows geometrically with the number of variable modifications. When a large number of variable modifications is expected, a better approach is to reduce the weight w_M of the mass component of scoring.

The *spectral component* (f_S) is similar to spectral scoring functions used for database search and *de novo* sequencing algorithms. Theoretical spectra for matched peptides from the database are compared to experimental MS/MS spectra. The basic assumption is that the greater the number of high abundance peaks that are matched by theoretically predicted ions, the more likely the matched peptide from the database is correct [16]. For this approach, raw MS/MS data is preprocessed to reduce the number of potential false positive peak matches. Peak centering is performed as previously described [33]. This is followed by deconvolution of the doubly and triply charged species to singly charged ions according to the method of Wehofsky and Hoffmann [34]. For each theoretically predicted peak a reward is then calculated as suggested by the authors of the PEAKS software [16]. The reward depends on many factors including the abundance of corresponding experimental peaks, the mass error between theoretical and experimental mass values and co-existence of the x, y-H_2O, y-NH_3 (or a, c, b-H_2O, b-NH_3) ions. If there is no peak close to given theoretical mass, the reward is a negative constant value.

The total reward $\tilde{f}_S$ reflects the similarity between theoretical and experimental spectra. The last step is normalization of the total reward to $0 \le f_S \le 1$. Taking into account the fact that the upper and lower boundaries of total reward are hard to determine due to their dependence of experimental spectrum, a simple sigmoid function is used for normalization:

$$f_S = \frac{1}{1 + \exp(-\beta \tilde{f}_S)} \quad (7)$$

This function satisfies $0 < f_S < 1$ for any $\tilde{f}_S$. The graph of this function is shown in Fig. (**2**). The parameter β should be determined to satisfy the condition that correct peptide matches would obtain a f_S value near 1. For exam-

ple, if a total reward larger than Z is typical for correct matches and we require that in this case f_S should be larger than $1-\delta$, then an approximate value for β can be obtained by the formula

$$\beta = \log\left((1-\delta)/\delta\right)/Z \quad (8)$$

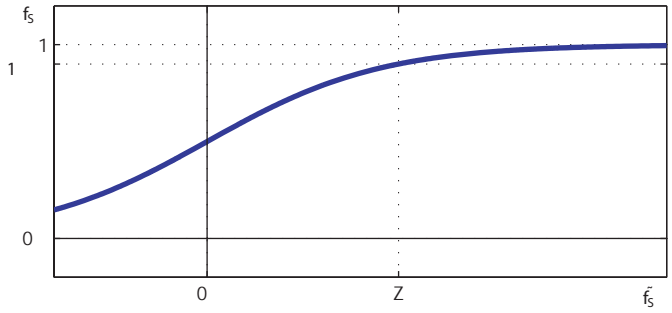

Fig. (2). Illustration of the normalization function for scoring in PepTiger.

RESULTS AND DISCUSSION

PepTiger matches *de novo* sequence tags with database sequences based on approximate string matching followed by a novel scoring procedure. This procedure takes into account string distance between *de novo* sequence and matched peptides, mass differences, and similarity between theoretical and experimental MS/MS spectra. PepTiger uses information of experimental MS/MS spectrum of peptide being identified for scoring. This approach leverages the potential errors introduced during *de novo* sequencing. Two datasets were used to test the performance of PepTiger. Test Dataset I was generously provided by Dr. Richard Johnson, an author of Lutefisk [13, 14]. Test Dataset II was provided by Dr. Hamid Mirzaei.

Test Dataset I contains 144 ion trap MS/MS spectra. Most of these spectra are obtained from proteins bovine serum albumin, chicken ovalbumin, mouse myosin regulatory light chain 2 and skeletal muscle isoform. All cysteine residues in these spectra are carbamidomethylated (45 spectra) and 7 spectra have oxidated methionine. Therefore, carbamido-methylation of cysteine was taken into account as a fixed chemical modification, whereas oxidation of methionine was regarded as variable modification. Test sequence tags were obtained with the *de novo* sequencing software PEAKS 3.0 [16]. Sequencing parameters for all spectra were chosen as default PEAKS values for an ion trap instrument: parent mass error tolerance and fragment mass error tolerance were both 0.3 Da, "Enzyme and PTM" parameters were "Trypsin with Cam". All spectra were automatically centroided and deconvoluted by PEAKS. Only the top scoring *de novo* sequence was selected for further analysis. Swiss-Prot Release 46.5 of 12-Apr-2005 (178 940 proteins) was used for testing the search engine.

Search Engine Parameters

The value l for the length of l-mer in the potential match detection procedure was taken as 3. The majority of tags in Test Dataset I (119 of 144, 83%) have exact matches of 3 consecutive letters with the correct protein and 25 tags (17%) don't satisfy this condition ("bad" tags). These cannot be correctly identified by PepTiger due to the reduction of the alignment model procedure. Therefore, the upper boundary for the success rate of this test dataset is 83%.

The number of matches depends on the threshold value. The optimal value of threshold $\varepsilon = 0.7$, was obtained empirically by examination of test dataset, where a tradeoff between sensitivity and specificity was considered. PepTiger distance was measured between each *de novo* sequence and its corresponding correct database protein (see Fig. **3**). Most of the tags with $\alpha_P > 0.7$ are included in the 25 "bad" tags. The length h of a segment containing *de novo* errors for which PepTiger distance is tolerant was taken as 2, as with the CIDentify software. Larger values dramatically decrease the search speed. Our approach is different from that of SPIDER and OpenSea which allow *de novo* errors of arbitrary length. However, the large value for threshold ε allows accepting matches with *de novo* errors longer than two residues, and at the same time an accurate scoring scheme helps to reject most of false positive matches.

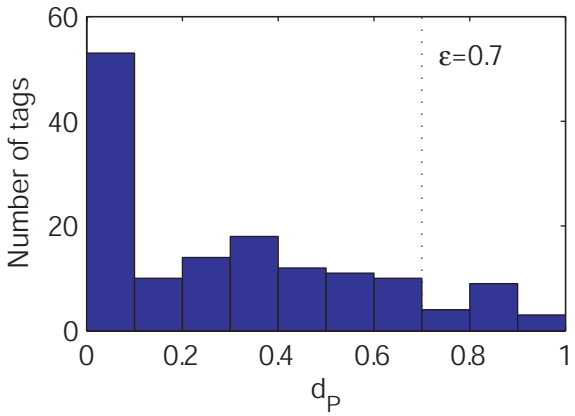

Fig. (3). Determination of PepTiger threshold ε.

Performance of Direct Search

By direct search we mean that all target proteins are present in the database. Each sequence tag was searched individually with PepTiger, SPIDER and CIDentify against SwissProt database.

SPIDER was tested in its 4 possible match modes: exact match, segment match, non-gapped homology and homology. Weights for scoring components of PepTiger were taken as: $w_P = 0.5$, $w_M = 0.25$, $w_S = 0.25$. These values were determined empirically to maximize the success rate over the entire test dataset. Carbamidomethylation was selected as a fixed modification and oxidation of methionine as a variable modification. A maximum of two insertions and two deletions was allowed for the search. Performance details for each algorithm can be provided by authors upon email request.

Table **1** summarizes the performance of PepTiger, SPIDER and CIDentify on Test Dataset I. The value of each cell in the table represents number of correct answers for *de novo* sequences from the test dataset among the Top 1 and Top 5 reported matches. In each case, PepTiger assigned the highest number of correct peptides as the top candidates, 88 (61%) and 96 (67%), respectively. The exact match mode of SPIDER gave the smallest number of correct matches in

Table 1. Peptide Identification Software Performance

Search Engine	Test Dataset I						Test Dataset II	
	Charge > 1 (127 spectra)		Charge = 1 (17 spectra)		All charges (144 spectra)		Charge = 1 (28 spectra)	
	Top 1	Top 5	Top1	Top 5	Top 1	Top 5	Top 1	Top 5
PepTiger	79(62%)	87(69%)	9(53%)	9(53%)	88(61%)	96(67%)	24(86%)	24(86%)
SPIDER homology	65(51%)	70(55%)	8(47%)	8(47%)	73(51%)	78(54%)	23(82%)	24(86%)
SPIDER non-gapped homology	66(52%)	72(57%)	5(29%)	7(41%)	71(49%)	79(55%)	22(79%)	24(86%)
SPIDER segment	70(55%)	74(58%)	6(35%)	7(41%)	76(53%)	81(56%)	23(82%)	24(86%)
SPIDER exact	18(14%)	18(14%)	5(29%)	5(29%)	23(16%)	23(16%)	15(54%)	15(54%)
CIDentify	60(47%)	67(53%)	10(59%)	10(59%)	70(49%)	77(53%)	23(82%)	24(86%)

each cases, 23 (16%). This is due to the very stringent requirement of the matching criteria. The other three SPIDER match modes and CIDentify provided similar performance. Separately each match mode of SPIDER performed more poorly than PepTiger. However, the combination of matched results from all four SPIDER match modes correctly ranked more database sequences as the top candidates than did PepTiger (see supplement material). Ninety-nine peptides were ranked as Top 1 sequence candidates by at least one of SPIDER's matching modes, while 88 peptides were ranked as Top 1 by PepTiger. It should be noted that these observations are based on known correct peptide sequences, which is not the case in a real experiment. In the experimental setting it is often not known which peptide is in the sample. Further, factors such as peptide concentration and ionization efficiency can effect MS detection. Therefore, one can not predict which SPIDER match mode will provide the correct sequence.

Practically, the success rate of SPIDER ranking the correct sequence as Top 1 should be calculated as: mean value of three SPIDER matching modes: SPIDER segment, SPIDER non-gapped homology and SPIDER homology:

$$SPIDER_{total} = \text{mean}(53\% + 49\% + 51\%) = 51\%$$

We didn't include SPIDER exact mode success rate into this formula because all sequences identified by exact match mode can also be identified by SPIDER segment match mode, so the exact match mode is a subset of segment match mode.

The goal of the scoring scheme is to reliably distinguish correct matches from the incorrect. The number of correct answers having top score value indicates the quality of the scoring scheme. The relationship between the matches for which PepTiger gave a high score value (Score > 0.7) and correct matches having Top 1 score is depicted in Fig. (**4**).

PepTiger has correctly matched 88 database peptide sequences as the top candidates. Out of these 88 matches, 77 matches have a score larger than 0.7 while the other 11 matches have an average score 0.65. On the other hand, of the 96 matches with a score higher than 0.7, 19 did not rank the correct database sequences as Top 1 candidate. Out of these 19 matches, 3 correct matches were among top 5, and 5 correct matches were among top 10 reported matches. It can be concluded, that large score values indicate a correct answer with high confidence.

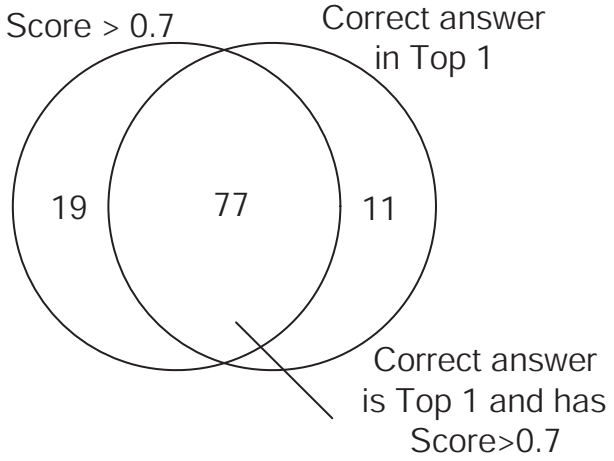

Fig. (4). Venn diagram illustrating the relationship between PepTiger high scores and highest ranked matches.

Most peptides from tandem MS spectra of Test Dataset I were doubly charged. Singly charged peptide ions typically do not generate information rich tandem MS spectrum and are often excluded from CID. Table **1** shows that PepTiger did a very good job in analyzing spectra from multiple charged ions compared with other software packages. It also demonstrated better performance than SPIDER in analyzing spectra from singly charged ions, but performed slightly worse in this respect than did CIDentify. A concern with this result is the small amount of testing data.

There were only 17 spectra generated by singly charged peptides in the Test Dataset I. To further examine the performance of PepTiger on spectra of singly charged peptides, we digested three proteins, transferrin, human serum albumin, and cytochrome c. Each digest was analyzed on a ABI QStar instrument using the information dependent acquisition (IDA) mode and selecting the top three abundance ions for fragmentation without excluding singly charged ions.

All tandem spectra were first analyzed using SEQUEST for peptide identification. There were 35 singly charged tryptic peptides identified by SEQUEST from all three experi-

ments. For each of these, a tandem spectrum with the highest Sequest Xcorr value was selected for Test Dataset II. This dataset was first analyzed by PEAKS using the same parameters as used for the Test Dataset I. There were 28 *de novo* sequencing tags generated by PEAKS satisfying l-mer condition (having 3-mer exact match with correct peptide sequence).

PepTiger was then used to match these 28 *de novo* sequencing tags with the protein database using the same analysis parameters as for Test Dataset I.

PepTiger ranked 24 correct sequences as Top 1 while CIDentify ranked 23 correct sequences as Top 1 (Table **1**). The exact mode of SPIDER ranked just 15 correct sequences as Top 1 candidate. The segment and homology search mode of SPIDER also ranked 23 sequences correctly as Top 1 while the non-homology search mode of SPIDER ranked 22 correct sequences as the Top 1 candidate. Comparing these results with those from the Test Dataset I, we conclude that PepTiger performs same as or better than SPIDER and CIDentify in matching *de novo* sequence tags with protein database sequences. Moreover, PepTiger demonstrates significantly superior matching of correct database sequences for *de novo* sequence tags generated from multiple charged peptides.

As for SPIDER, PepTiger requires that *de novo* sequence tags satisfy the l-mer requirement. The typical length of l-mer is three amino acid residues. The success rate of PepTiger also depends on the quality of *de novo* sequencing. With increased accuracy of *de novo* sequencing algorithms to correctly generate l-mers, the performance of PepTiger is significantly improved. In case of the Test Dataset I, the success rate of PepTiger ranking the correct sequence of multiple charged peptides as Top 1 will be improved from 62% to 73% if all *de novo* sequence tags meet l-mer requirement.

CONCLUSIONS

PepTiger provides a novel and powerful scoring scheme. Our results demonstrate that PepTiger identifies more *de novo* sequences with typical *de novo* errors from the test dataset than do other popular protein identification software packages. The advantage of PepTiger is its novel scoring scheme, which takes into account not only similarity between sequences of *de novo* tags and peptides from the database, but also the similarity between experimental MS/MS spectra and theoretical spectra for peptides in the database. By allowing *de novo* errors of arbitrary lengths the other packages sometimes produce incorrect matches that are rejected by taking into account experimental MS/MS information. PepTiger's approach combines advantages of both database search and *de novo* sequencing approaches for protein identification from experimental MS spectra.

ACKNOWLEDGMENTS

Drs Richard Johnson and Hamid Mirzaei kindly provided experimental spectra. IF and ZO were supported by CRDF grant RC1-2493-MO-04 at the early stage of this research. CB and XZ were supported by NCI Grant U24CA126480.

REFERENCES

[1] Perkins, D.N.; Pappin, J.D.C.; Creasy, D.M.; Cottrell, J.S. *Electrophoresis*, **1997**, *20*, 3551.
[2] Eng, J.K.; McCormack, A.L.; Yates, J.R.I. *J. Am. Soc. Mass. Spectrom.*, **1994**, *5*, 976.
[3] Davis, M.T.; Spahr, C.S.; McGinley, M.D.; Robinson, J.H.; Bures, E.J.; Beierle, J.; Mort, J.; Yu, W.; Luethy, R.; Patterson, S.D. *Proteomics*, **2001**, *1*, 108.
[4] Yates, J.R.I.; Eng, J.K.; McCormack, A.L.; Schieltz, D. *Anal. Chem*., **1995**, *67*, 1426.
[5] Mann, M.; Wilm, M. *Anal. Chem.*, **1994**, *66*, 4390.
[6] Sunyaev, S.; Liska, A.J.; Golod, A.; Shevchenko, A.; Shevchenko, A. *Anal. Chem.*, **2003**, *75*, 1307.
[7] Tabb, D.L.; Saraf, A.; Yates, J.R.I. *Anal. Chem.*, **2003**, *75*, 6415.
[8] Mortz, E.; O'Connor, P.B.; Roepstorff, P.; Kelleher, N.L.; Wood, T.D.; McLafferty, F.W.; Mann, M. *Proc. Natl. Acad. Sci. USA*, **1996**, *93*, 8264.
[9] Han, Y.; Ma, B.; Zhang, K. *J. Bioinform. Comput. Biol.*, **2005**, *3*, 697.
[10] Searle, B.C.; Dasari, S.; Wilmarth, P.A.; Turner, M.; Reddy, A.P.; David, L.L.; Nagalla, S.R. *J. Proteome Res.*, **2005**, *4*, 546.
[11] Bartels, C. *Biomed. Environ. Mass Spectrom.*, **1990**, *19*, 363.
[12] Fernandez-de-Cossio, J.; Gonzalez, J.; Besada, V. *Comput. Appl. Biosci.*, **1995**, *11*, 427.
[13] Taylor, J.A.; Johnson, R.S. *Rapid Commun. Mass Spectrom.*, **1997**, *11*, 1067.
[14] Taylor, J.A.; Johnson, R.S. *Anal. Chem.*, **2001**, *73*, 2594.
[15] Dancik, V.; Addona, T.A.; Clauser, K.R.; Vath, J.E.; Pevzner, P.A. *J. Comput. Biol.*, **1999**, *6*, 327.
[16] Ma, B.; Zhang, K.; Hendrie, C.; Liang, C.; Li, M.; Doherty-Kirby, A.; Lajoie, G. *Rapid Commun. Mass Spectrom.*, **2003**, *17*, 2337.
[17] Heredia-Langner, A.; Cannon, W.R.; Jarman, K.D.; Jarman, K.H. *Bioinformatics*, **2004**, *20*, 2296.
[18] Shevchenko, A.; Sunyaev, S.; Loboda, A.; Shevchenko, A.; Bork, P.; Ens, W.; Standing, K.G. *Anal. Chem.*, **2001**, *73*, 1917.
[19] Huang, L.; Jacob, R.J.; Pegg, S.C.-H.; Baldwin, M.A.; Wang, C.C.; Burlingame, A.L.; Babbitt, P.C. *J. Biol. Chem.*, **2001**, *276*, 28327.
[20] Mackey, A.J.; Haystead, T.A.J.; Pearson, W.R. *Mol. Cell. Proteomics*, **2002**, *1*, 139.
[21] Searle, B.C.; Dasari, S.; Turner, M.; Reddy, A.P.; Choi, D.; Wilmarth, P.A.; McCormack, A.L.; David, L.L.; Nagalla, S.R. *Anal. Chem.*, **2004**, *76*, 2220.
[22] Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, Vol. *89*, **1992**.
[23] Levenshtein, V. *Soviet Physics Doclady*, **1966**, *10*, 707.
[24] Damerau, F.A. *Commun. ACM*, **1964**, *7*, 171.
[25] Wagner, R.; Fisher, M. *J. ACM*, **1974**, *21*, 168.
[26] Ukkonen, E. *J. Algor*., **1985**, *6*, 132.
[27] Myers, G.A. *J. ACM*, **1999**, *46*, 395.
[28] Proc. of the Prague Stringology Conference (PCS 2002) A bit-vector algorithm for computing Levenshtein and Damerau edit distances, **2002**.
[29] Navarro, G. *ACM Computing Surveys*, **2001**, *33*, 31.
[30] Johnson, R.S.; Davis, M.T.; Taylor, J.A.; Patterson, S.D. *Methods*, **2005**, *35*, 223.
[31] Ji, J.; Chakraborty, A.; Geng, M.; Zhang, X.; Amini, A.; Bina, M.; Regnier, F.E. *J. Chromatogr. A*, **2000**, *745*, 197.
[32] Gygi, S.O.; Rist, B.; Gerber, S.A.; Turacek, F.; Gelb, M.H.; Aebersold, R. *Nat. Biotechnol.*, **1999**, *17*, 994.
[33] Gentzel, M.; Kocher, T.; Ponnusamy, S.; Wilm, M. *Proteomics*, **2003**, *3*, 1597.
[34] Wehofsky, M.; Hoffmann, R. *Eur. J. Mass Spectrom.*, **2002**, *7*, 39.

Received: June 18, 2007 Revised: August 03, 2007 Accepted: August 07, 2007