SPIDER: Software for Protein Identification from Sequence Tags with *De Novo* Sequencing Error

Yonghua Han, Bin Ma, Kaizhong Zhang

Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
{*yhan2,bma,kzhang*}*@csd.uwo.ca*

For the identification of novel proteins using MS/MS, *de novo* sequencing software computes one or several possible amino acid sequences (called sequence tags) for each MS/MS spectrum. Those tags are then used to match, accounting amino acid mutations, the sequences in a protein database. If the *de novo* sequencing gives correct tags, the homologs of the proteins can be identified by this approach and software such as MS-BLAST is available for the matching. However, de novo sequencing very often gives only partially correct tags. The most common error is that a segment of amino acids is replaced by another segment with approximately the same masses. We developed a new efficient algorithm to match sequence tags with errors to database sequences for the purpose of protein and peptide identification. A software package, SPIDER, was developed and made available on Internet for free public use. This paper describes the algorithms and features of the SPIDER software.

Keywords: protein identification; *de novo* sequencing; sequence tags; database search.

1. Introduction

Protein identification is a fundamental problem in Proteomics. Tandem mass spectrometry (MS/MS) is emerging as the standard method for this important protein identification problem [1]. In the current practice of protein identification by using MS/MS, purified proteins are digested into short peptides with enzymes such as trypsin. Then, tandem mass spectra are measured for the peptides with a tandem mass spectrometer. Finally, the spectra are interpreted by computer software to identify the amino acid sequences of the peptides and proteins.

Two approaches exist for the software to interpret the data. The first approach searches in a protein database to find the peptides that match the MS/MS spectra the best [8,22,23,31]. In this paper we call it the database search approach. A number of software programs have been developed for performing such searching tasks, the most popular being Mascot and Sequest [8,22]. The database search approach is effective when the sequences of the target proteins are in the database. But it does not work when the target proteins are not in the database.

The second approach computes the amino acid sequences of the peptides directly from their MS/MS spectra. Such a procedure is called *de novo* sequencing [4,5,6,7,9,12,16,17,24,28,29]. Only after the *de novo* sequencing is done for each of the MS/MS spectra, the obtained sequences (or partial sequences) of the peptides are searched

in a protein database to find the protein that matches [10,25,19]. To distinguish with the database search approach, we call this approach the *de novo* approach. *De novo* approach has the advantage that it works well for both known and novel proteins. When the proteins are not in the database, because *de novo* sequencing can still provide the sequences of the peptides, a homology search program such as BLAST [2,3] can be used to identify the homologs of those peptides, and therefore, the homologs of the target proteins.

The *de novo* approach is gaining more and more attention, especially because the *de novo* sequencing software is getting better and can compute much more accurate sequences than ever before. As a matter of fact, even for proteins that are in the database, researchers have recently reported that the *de novo* approach provides better results than the database search approach for the MS/MS data produced with certain instruments [19,25].

The *de novo* approach needs a *de novo* sequencing software program to compute the peptide sequences, and a similarity search software package to match the peptide sequences in a database. The former step has been widely studied recently [5,6,7,9,12,16,17,24,28,29]. Both commercial software, such as PEAKS [16], and free software, such as Lutefisk [28,29] and those described in other papers [14,9,30], are available. But people are still using the general homology search tools like BLAST for the second step. These general tools are often not suitable for the matching in our application. For example, the peptide sequences generated by the first step are usually short (10-20 amino acids), but the general homology search tools usually anticipate much longer query sequences. So, the parameters of the homology search tools need to be adjusted. Three general purpose homology search programs, FASTA [20], Shotgun [21] and BLAST [2,3], have been modified to three sequence tag searching problems: FASTS [18], MS-Shotgun [13] and MS-BLAST [26].

But more importantly, the *de novo* sequencing software almost always produce peptide sequences with errors. This is determined by the fact that biochemical experiments for protein identification always contain errors, which include the impurity of the sample, the chemical noise in the MS/MS spectra, the imperfect ionization and fragmentation of the peptides in the MS/MS spectrometers [16]. The *de novo* sequencing errors are also because the mathematical models used by *de novo* sequencing software are greatly simplified in order to make the computational problem feasible. In this paper, we will use *sequence tags* to refer to those *de novo* sequences that may contain errors.

According to our experience, in a typical match between a sequence tag with a homolog of the real peptide, more mismatched letters are caused by *de novo* errors than by homology mutations. In such a circumstance, the mathematical foundation that is used to build general homology search tools, such as BLAST, does not hold anymore. As a result, a homology search tool that considering both homology mutations and *de novo* sequencing errors is greatly desired.

2. Related Work

The search programs, FASTS [18], MS-Shotgun [13], and MS-BLAST [26] that were modified from the general purpose homology search programs can take multiple sequence tags as their inputs and identify the homologous proteins. The main difference between these

programs with their "parent" programs is the adjustment of parameters and the simultaneous consideration of multiple sequence tags to improve the protein identification accuracy. However, none of them considers the *de novo* sequencing errors. The errors may increase the matching score of a wrong protein and also decrease the matching score of a correct protein, and therefore create false positives and negatives [25].

Because most *de novo* sequencing software determines the amino acids by examining the mass differences between pairs of peaks in an MS/MS spectrum, the most common *de novo* sequencing error is the replacement of one sequence segment by another with similar mass. In their poster [10], Han et *al.* proposed a simple method to match sequence tags with peptide sequences in the database, allowing the matching of similar-mass segments. The numbers of matching letters and segments are used to measure the quality of the matches. When the correct peptides are in the database, Han et *al.* demonstrated that this segment match method can help identify the correct peptides by searching the partially correct sequence tags in a database. In the same poster [10], they also proposed to combine homology mutations with segment matches in order to identify proteins not in the database. But no algorithm was developed there. The algorithms and the SPIDER program discussed in this paper is the continuation of the research.

Recently, an OpenSea software package [25] was brought to our attention. OpenSea is developed by research effort independent to ours. The software matches partially correct sequence tags with a database to identify the homologous or modified proteins. It considered both *de novo* sequencing errors and homology mutations in its search. However, the *de novo* errors and homology mutations are not allowed to occur at the same positions. It further requires that no *de novo* sequencing errors have length greater than three. Also, OpenSea uses a simple greedy algorithm to do the match, which may not find the optimal alignment between the query and a database sequence. We noticed that OpenSea does a partial function, the "non-gapped homology match mode", of SPIDER. The research conducted in the OpenSea paper [25] also demonstrated that the *de novo* approach is superior to the database search approach for protein identification.

It is also possible to integrate the findings of sequence tags from MS/MS and the sequence tag search in the database into one program. GutenTag [27] is a program that implements a similar idea. GutenTag first produces some short pieces of the *de novo* sequence. Then a database is searched to find the matches of the pieces. GutenTag scores the proteins according to five factors: a tag match, a mass gap match at either side of the tag, and a tryptic-termini match on either side of the peptide. However, GutenTag does not consider homology mutations or modifications. As a result, GutenTag is not suitable for the identification of unknown or modified proteins [25].

In this paper, we present SPIDER's methods to do sequence tag searching. SPIDER's approach takes into account of both homology mutations and *de novo* sequencing errors. Moreover, it allows the mutations and errors to occur at the same positions of the sequence. Also, the method we develop here can be straightforwardly extended to identify proteins with post-translational modifications [25]. An interesting bonus feature of SPIDER's algorithm is that it can "guess" the most likely real sequence by combining the information of the partially correct sequence tag and the homolog of the real sequence.

The rest of the paper is organized as follows: Section 3 models the sequence tag search problem as a very special alignment problem by introducing an "intermediate" real sequence; Section 4 presents a dynamic programming algorithm to find the optimal alignment; Section 5 studies a "simplified" version of SPIDER's alignment problem. The time complexity is therefore reduced; Section 6 introduces the features and heuristics used in SPIDER software; Section 7 gives the testing results on real sequence tags.

3. SPIDER's Alignment Model

When a sequence tag is aligned to a database sequence, two types of editing operations should be considered. One is the *de novo* sequencing error, which replaces one segment of letters by another segment with the same mass. The other is the homology mutations between the real sequence and the database sequence.

Let Σ be our alphabet, i.e., all the amino acids. Let X be the sequence tag, Y be the real peptide sequence, and Z be Y's homolog in a database. Very often, Y is slightly different from both X and Z. The difference between Y and X is caused by *de novo* sequencing error, and the difference between Y and Z is caused by homology mutations. We use $f(X, Y)$ to denote the distance function that measures the *de novo* sequencing error that changes Y to X, and use $g(Y, Z)$ to denote the edit distance between Y and Z. Then, the SPIDER distance between X and Z, denoted by $d(X, Z)$, is naturally defined by $d(X, Z) = f(X, Y) + g(Y, Z)$. Because we usually do not know what Y is, we define

$$d(X, Z) = \min_{Y}(f(X, Y) + g(Y, Z)).$$

In the following we will introduce a general way to define $f(\cdot, \cdot)$ and $g(\cdot, \cdot)$. The actual definitions used in SPIDER software will be discussed in Section 6.1. We note that SPIDER's algorithm that will be introduced in Section 4 can be straightforwardly used to many other definitions of $f(\cdot, \cdot)$ and $g(\cdot, \cdot)$.

When a segment Y_0 with mass m is replaced by a different segment X_0 due to *de novo* sequencing error, we associate it with a cost $f'(m) > 0$. Because any practical *de novo* sequencing software has a reasonable success rate, we do not anticipate to see many long segments being computed wrongly. Also, when m is larger, the number of different segments with the same mass m is also larger, and therefore the *de novo* sequencing software has a smaller chance to output this specific segment X_0. For these two reasons, $f'(m)$ is generally greater for greater m. However, $f'(m)$ can be very dependable to m and usually is not a monotonic function. For completeness, if a letter a is assigned correctly by the *de novo* sequencing software, we also associate it with a cost $f^*(a)$. The *de novo* sequencing cost, $f(X, Y)$ is then defined by the sum of costs of segment replacements and correct letter assignments between X and Y.

For example, Let $X =$ `LSCFAV` and $Y =$ `EACFAV`. Because we only consider replacements of segments with same mass, and all masses of the amino acids (letters) are known, the alignment in Figure 1(a) can be constructed easily. Then the *de novo* sequencing cost is equal to $f'(218.1) + f^*(\texttt{C}) + f^*(\texttt{F}) + f^*(\texttt{A}) + f^*(\texttt{V})$, where $218.1 \approx m(\texttt{LS}) \approx m(\texttt{EA})$.

```
X:   [LS]CFAV                  [LS]CFAV
Y:   [EA]CFAV     EACFAV       [EA]CFAV
Z:                ETCF-V       [ET]CF-V
        (a)         (b)           (c)
```

Fig. 1. (a) The alignment between a *de novo* sequence tag X and the real sequence Y. Same-mass segment replacements are allowed. (b) The alignment between a real sequence Y and a database sequence Z. Substitutions and insertions/deletions are allowed. (c) The alignment of the three sequences X, Y, Z constructed from (a) and (b).

Let $g'(a, b)$ be the distance between two amino acids a and b, and $\bar{g} > 0$ be the insertion/deletion cost. $g(Y, Z)$ is defined to be the conventional edit distance between Y and Z under the scoring scheme $g'(a, b)$ and $\bar{g}$.

Because our algorithm works generally on any reasonally defined $f'(m)$, $f^*(a)$, $g'(a, b)$ and $\bar{g}$, we will not worry about the actual selection of these functions until when we discuss the software implementation in Section 6.1. However, in an examplary system, these functions can be the negative logarithms of the probabilities of the corresponding events. The advantage of using probabilites is that the weight between $f(X, Y)$ and $g(Y, Z)$ is calibrated automatically.

Once the distance functions are defined, a sequence tag search problem is then to find a peptide sequence from the database, such that its SPIDER distance to the query sequence is minimized. This can be done by enumerating all the database sequences and computing the SPIDER distance. Therefore, the key to solve this problem is an efficient algorithm to compute $d(X, Z)$ for any given X and Z. Moreover, the core problem is to find this "intermediate" sequence Y, for the following two reasons:

(1) Once Y is known, $f(X, Y)$ and $g(Y, Z)$ can be computed relatively easily.

(2) Y is likely the real peptide sequence, knowing that it is homologous to Z and a *de novo* sequence software program computes an partially correct sequence tag X.

Using the above discussed distance model, an optimal alignment of the three sequences, X, Y, and Z can be constructed by merging the alignment of (X, Y) and the alignment of (Y, Z) (See Figure 1). In the alignment, the columns involved in a single segment replacement is called a *block*. Similarly, if a letter of Y is correctly assigned in X, we also call that column a *block*. Therefore, in the alignment of Figure 1(c), there are five blocks. The first block involves a segment replacement and the other four blocks are all single-column blocks. The *length of a block* is the maximum length of the X and Z segments in that block. Therefore, the five blocks in Figure 1(c) have lengths 2, 1, 1, 1, and 1, respectively.

From the definition of our cost function, it is obvious that $d(X, Z)$ can be computed "block-wisely" as follows.

Let Y be such that $f(X, Y) + g(Y, Z)$ is minimized. That is, $d(X, Z) = f(X, Y) + g(Y, Z)$. In an alignment of X, Y and Z, let X_i, Y_i, and Z_i be the segments of the three sequences in the i-th block, respectively. Then

$$d(X, Z) = f(X, Y) + g(Y, Z)$$

$$= \sum_i f(X_i, Y_i) + \sum_i g(Y_i, Z_i)$$
$$= \sum_i [f(X_i, Y_i) + g(Y_i, Z_i)]. \quad (1)$$

The following lemma tells us Y_i is optimal in each block.

Lemma 1. *Y_i is the sequence that minimizes $f(X_i, Y_i) + g(Y_i, Z_i)$. I.e. $d(X_i, Z_i) = f(X_i, Y_i) + g(Y_i, Z_i)$.*

Proof. If there is another sequence Y_i' such that $f(X_i, Y_i') + g(Y_i', Z_i) < f(X_i, Y_i) + g(Y_i, Z_i)$, replacing Y_i with Y_i' in sequence Y will decrease the value of (1), which is a contradiction to the optimity of Y. □

Definition 1. SPIDER's sequence alignment problem Given a sequence tag X and a database sequence Z, compute a real sequence Y and an alignment of X, Y and Z, so that the cost $f(X, Y) + g(Y, Z)$ is minimized.

4. Dynamic Programming for Optimal Alignment

In this section, we present a dynamic programming algorithm to compute $d(X, Z)$. In order to compute $d(X, Z)$, we first calculate another function $\alpha(m, Z)$ which is defined below.

Let Z be a segment and m be a mass, define

$$\alpha(m, Z) = \min_{Y:m(Y)=m} g(Y, Z). \quad (2)$$

That is, $\alpha(m, Z)$ denotes the minimum edit distance between Z and a segment with mass m. Here we require $m(Y)$ to be exactly equal to m for the simplicity of presentation. In practice, it is easy to modify the algorithms described in this section to compute $\min_{Y:m(Y)\approx m} g(Y, Z)$. And therefore small mass errors are allowed in the same-mass segment replacement.

Because of the use of edit distance in $g(Y, Z)$, m can be different from $m(Z)$. As a result, the computation of $\alpha(m, Z)$ is not trivial. Lemmas 2 and 3 study the computations. Recall that $\bar{g}$ is the insertion/deletion cost and $g'(a, b)$ is the distance between two amino acids a and b.

Lemma 2.

$$\alpha(m, Z[i..j]) = \min \begin{cases} \min_{y\in\Sigma} \alpha(m - m(y), Z[i..j]) + \bar{g}, \\ \alpha(m, Z[i..(j-1)]) + \bar{g}, \\ \min_{y\in\Sigma}(\alpha(m - m(y), Z[i..(j-1)]) + g'(y, Z[j])) \end{cases} \quad (3)$$

Proof. Let $Y = Y_1 y$ be the optimal sequence that minimizes $\alpha(m, Z[i..j])$ and consider the optimal alignment between Y and $Z[i..j]$. There are three cases: (A) y is deleted (insertion in Z), (B) $Z[j]$ is deleted, or (C) y is aligned with $Z[j]$, as illustrated in Figure 2.

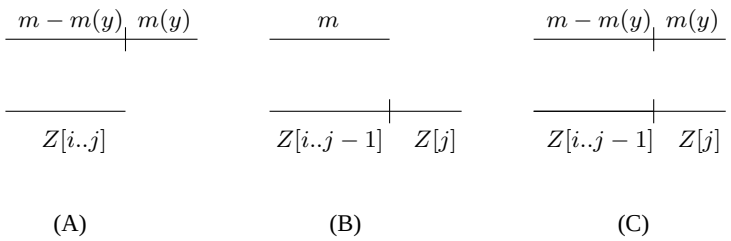

Fig. 2. Three cases for the alignment between a mass gap and sequence Z: (A) insertion in Z; (B) deletion in Z; and (C) match.

If y is deleted, then there is a cost, $\bar{g}$, of deleting y and Y_1 is the optimal sequence that minimizes $\alpha(m - m(y), Z[i..j])$. Therefore, $\alpha(m, Z[i..j]) = \alpha(m - m(y), Z[i..j]) + \bar{g} = \min_y \alpha(m - m(y), Z[i..j]) + \bar{g}$.

If $Z[j]$ is inserted, there is a cost inserting $Z[j]$ and Y is the optimal sequence that minimizes $\alpha(m, Z[i..(j-1)])$. Therefore $\alpha(m, Z[i..j])$ = $\alpha(m, Z[i..(j-1)])$ + $\bar{g}$.

If y is aligned with $Z[j]$, there is a substitution cost of $g'(y, Z[j])$ and Y_1 is the optimal sequence that minimizes $\alpha(m - m(y), Z[i..(j-1)])$. Therefore $\alpha(m, Z[i..j])$ = $\alpha(m - m(y), Z[i..(j-1)])$+ $g'(y, Z[j])$. □

Lemma 2 gives the recurrence relation that computes $\alpha(m, Z[i..j])$. The base of the recurrence is studied in the following lemma.

Lemma 3.

$$\begin{aligned}\alpha(0, nil) &= 0.\\ \alpha(0, Z[i..j]) &= \alpha(0, Z[i..(j-1)] + \bar{g}.\\ \alpha(m, nil) &= \min_{y\in\Sigma} \alpha(m - m(y), nil]) + \bar{g}.\end{aligned}$$

Proof. The first two equations are trivial. For the third equation, let $Y = Y_1 y$ be the optimal sequence that minimizes $\alpha(m, nil)$. Then Y_1 must be the optimal sequence that minimizes $\alpha(m - m(y), nil)$. Consequently, the third equation holds. □

Let A be a table, where $A[m, i, j]$ has value $\alpha(m, Z[i..j])$. From Lemma 3 and Lemma 2, A can be straightforwardly computed by the following dynamic programming algorithm in $O(M \times |Z|^2 \times |\Sigma|)$ time, where $M = m(X)$.

Algorithm DP-α

Input: Two sequences X and Z.
Output: Array A.

1. for i from 1 to $|Z|$ do
   $A[0, i, i-1] = 0$.
2. for $i \leq j$ do
   $A[0, i, j] = A[0, i, j-1] + \bar{g}$.
3. for m from 1 to $m(X)$ do
   for i from 1 to $|Z|$ do
   $A[m, i, i-1] = \min_{y \in \Sigma} A[m - m(y), i, i-1]) + \bar{g}$.
4. for m from 1 to M do
   for i from 1 to $|Z|$ do
   for j from i to $|Z|$ do
   $$A[m, i, j] = \min \begin{cases} \min_{y \in \Sigma} A[m - m(y), i, j] + \bar{g}, \\ A[m, i, j-1]) + \bar{g}, \\ \min_{y \in \Sigma}(A[m - m(y), i, j-1] + g'(y, Z[j])). \end{cases}$$
5. Output A.

In the following, let $D[i, j] = d(X[1..i], Z[1..j])$ and $m[i', i] = m(X[i'..i])$. $m[i', i]$ can be pre-computed. Lemma 4 studies the recurrence relation of $D[i, j]$, and Lemma 5 studies the base of the recurrence.

Lemma 4.

$$D[i, j] = \min \begin{cases} D[i, j-1] + \bar{g}, \\ D[i-1, j] + \bar{g} + f^*(X[i]), \\ D[i-1, j-1] + g'(X[i], Z[j]) + f^*(X[i]), \\ \min_{1 \leq i' \leq i, 1 \leq j' \leq j}(D[i'-1, j'-1] + \alpha(m[i', i], Z[j'..j]) + f'(m[i', i])) \end{cases}$$

Proof. Let Y be the sequence that minimizes $D[i, j]$ and consider the optimal alignments between $X[1..i]$ and Y and Y and $Z[1..j]$. Let $Y = Y_1 y$. Four cases can happen as illustrated in Figure 3.

(A) $Z[j]$ is deleted.

Then Y is the sequence that minimizes $D[i, j-1]$ and there is a deletion cost of $\bar{g}$. Therefore $D[i, j] = D[i, j-1] + \bar{g}$, which is the first item in the min operation.

(B) $y = X[i]$ is not *de novo* error, but y is deleted.

Then there is cost of $f^*(X[i])$ associated with $X[i]$ and a deletion cost of $\bar{g}$ associated with y. Y_1 is the sequence that minimizes $D[i-1, j]$. Therefore $D[i, j] = D[i-1, j] + \bar{g} + f^*(X[i])$, which is the second item in the min operation.

(C) $y = X[i]$ is not *de novo* error, and y aligned with $Z[j]$.

Then there is a substitution cost $g'(X[i], Z[j])$ and Y_1 is the sequence that minimizes $D[i-1, j-1]$. Therefore $D[i, j] = D[i-1, j-1] + g'(X[i], Z[j]) + f^*(X[i])$, which is the third item in the min operation.

(D) $X[i'..i]$ is a segment of *de novo* error, which is aligned with segment $Z[j'..j]$ in the optimal alignment.

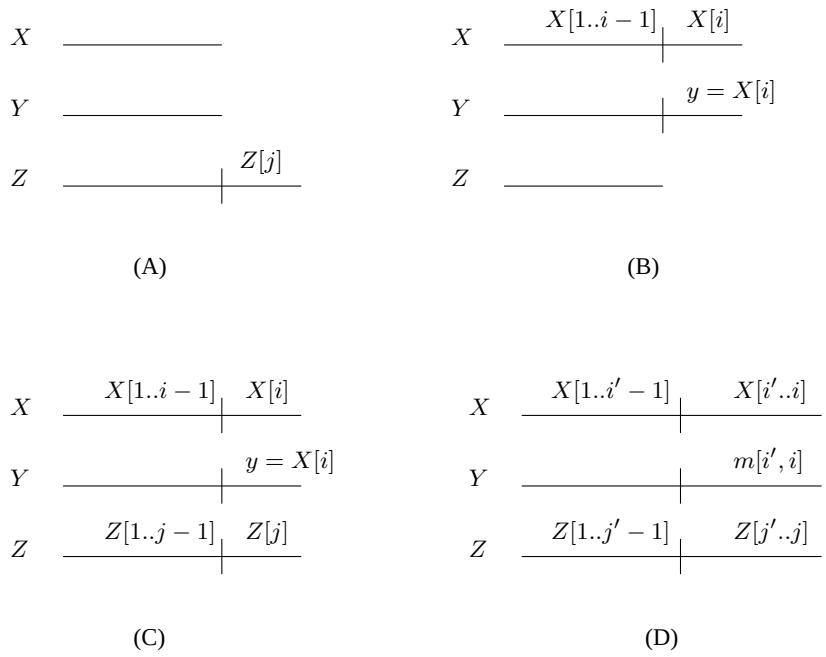

Fig. 3. The four cases for the alignment of X, Y and Z: (A) $Z[j]$ is deleted; (B) $y = X[i]$ is not *de novo* error, but y is deleted; (C) $y = X[i]$ is not *de novo* error, and y aligned with $Z[j]$; and (D) $X[i'..i]$ is a segment of *de novo* error, which is aligned with segment $Z[j'..j]$ in the optimal alignment.

In this case, the cost of the *de novo* error is $f'(m[i', i])$, the cost of aligning the suffix of Y with mass $m[i', i]$ with $Z[j'..j]$ is $\alpha(m[i', i], Z[j'..j])$, and the cost between $X[1..i' - 1]$ and $Z[1..j' - 1]$ is $D[i' - 1, j' - 1]$. Therefore $D[i, j] = \min_{1 \le i' \le i, 1 \le j' \le j}(D[i' - 1, j' - 1] + \alpha(m[i', i], Z[j'..j]) + f'(m[i', i]))$, which is the fourth item in the min operation. □

Lemma 5.

$$D[0, 0] = 0.$$
$$D[0, j] = D[0, j - 1] + \bar{g}.$$
$$D[i, 0] = \min \begin{cases} D[i - 1, 0] + \bar{g} + f^*(X[i]), \\ \min_{1 \le i' \le i}(D[i' - 1, 0] + \alpha(m[i', i], nil) + f'(m[i', i])) \end{cases}$$

Proof. The first two equations are trivial. For the third equation, Z has length 0. If $X[i]$ is maintained in Y and then deleted in Z, the cost is $D[i - 1, j] + \bar{g} + f^*(X[i])$. If a suffix $X[i', i]$ is replaced by a mass gap $m[i', i]$ in Y and then deleted in Z, the cost is $D[i' - 1, 0] + \alpha(m[i', i], nil) + f'(m[i', i])$. All i' satisfying $1 \le i' \le i$ should be tried to minimize the cost. □

From Lemmas 4 and 5, a dynamic programming algorithm can be designed straightforwardly to compute all $D[i, j]$ in $O(|X|^2 \times |Z|^2)$ time, providing that $\alpha(m, Z[i..j])$ has

already been computed and stored by Algorithm DP-α. Here we omit the details. Therefore, the computation of $\alpha(\cdot,\cdot)$ and $D[\cdot,\cdot]$ takes $O((|X|^2 + m(X)) \times |Z|^2)$ time in total. And a standard backtracking procedure can be used to construct the optimal alignment between X and Z, as well as the optimal Y such that $d(X,Z) = f(X,Y) + g(Y,Z)$.

The above-discussed algorithm is a global alignment version and one can easily extend it to a pattern matching version where the goal is find a substring of Z which has minimum distance to X, as follows.

Now let $D[i,j] = \min_{1 \le j' \le j+1} d(X[1..i], Z[j'..j])$, the formulas for computing $D[i,j]$ are exactly the same as before except the initialization which is list bellow.

Lemma 6.

$$
\begin{aligned}
&D[0,0] = 0.\\
&D[0,j] = 0.\\
&D[i,0] = \min \begin{cases} D[i-1,0] + \bar{g} + f^*(X[i]), \\ \min_{1 \le i' \le i}(D[i'-1,0] + \alpha(m[i',i], nil) + f'(m[i',i])) \end{cases}
\end{aligned}
$$

Without giving further detail, we claim that the algorithms in this section can be modified to find local alignment and gapped alignment. The algorithms can also be used under many other score functions, as well as to find the alignment that maximizes a score function.

5. Improving the Efficiency

Although running in polynomial time, the algorithm introduced in Section 4 is not practical, because the algorithm needs to be used millions of times to align the query sequence with every peptide in the database. In this section, we try to "simplify" our alignment model in order to get a more efficient alignment algorithm. It is easy to see that the time complexity comes from the large value of m in (3). As a result, we will try to "simplify" our model by using a "bad-block" cost to approximate the function $\alpha(m,\cdot)$ when m is large. The reason we put quotation marks around "simplify" is that the description of the model and algorithm becomes much more complicated, although the time complexity is reduced.

We visually examined many sequence tags obtained by *de novo* sequencing and noticed that the segment replacements of length 1, 2 and 3 are the most common error. Therefore, if we calculate the costs of length 1, 2 and 3 segment replacements accurately, and only approximate the costs of the longer segment replacements, the alignment obtained will be a good approximation to the optimal alignment. In general, let Δ be a positive integer. The blocks with segment replacements of length greater than Δ are called *bad blocks*. In contrast, all the other blocks are called *good blocks*.

The costs of good blocks can be pre-computed by the dynamic programming algorithm in Section 4 and stored in a *SPIDER Segment Substitution Scoring matrix (S4-matrix)*. The S4-matrix contains all the values of $d(X,Z)$ for all segments X and Z of length at most Δ. For each pair of X and Z, the S4-matrix also records the intermediate segment Y such that $d(X,Z) = f(X,Y) + g(Y,Z)$.

To simplify the computation, the costs of bad blocks are only estimated. Suppose a bad block has a letters in X, and b letters in Z, the cost of the block is then defined by $\alpha \times \min(a, b) + \beta \times |a - b| + \gamma$. That is, α is a mismatch cost in a bad block, and β is a insertion/deletion cost in a bad block, and γ is a bad-block open penalty. Usually we require that $\alpha \leq 2\beta$.

Definition 2. SPIDER's bad-block sequence alignment problem: Let S_4 be an S4-matrix. Let α, β and γ be three positive numbers. For any given two sequences X and Z, construct an alignment so that the sum of the good and bad block costs is minimized.

In order to represent good blocks and bad blocks, two matrices G and B are used. Let $G[i, j]$ be the maximum score of the alignments between $X[1..i]$ and $Z[1..j]$ ending at a good block; and $B[i, j]$ be the maximum score of the alignment between $X[1..i]$ and $Z[1..j]$ ending in a bad block.

The recursive formulas for computing G and B are given in (4 - 6). (4) initializes matrices G and B, (5) computes $B[i, j]$ and (6) computes $G[i, j]$.

$$\begin{array}{l} G[0,0] = 0;\ G[0,j] = 0;\ G[i,0] = \bar{g}; \\ B[0,0] = \gamma;\ B[0,j] = \gamma;\ B[i,0] = \gamma + \beta. \end{array} \tag{4}$$

$$B[i,j] = \min \begin{cases} B[i-1][j-1] + \alpha, \\ B[i-1][j] + \beta, \\ B[i][j-1] + \beta, \\ G[i-1][j-1] + \alpha + \gamma, \\ G[i-1][j] + \beta + \gamma, \\ G[i][j-1] + \beta + \gamma \end{cases} \tag{5}$$

$$G[i,j] = \min \begin{cases} G[i-1][j] + \bar{g}, \\ G[i][j-1] + \bar{g}, \\ \min_{i-\Delta<i' \leq i; j-\Delta<j' \leq j}(G[i'-1][j'-1] + S_4(X[i', i], Z[j', j])), \\ \min_{i-\Delta<i' \leq i; j-\Delta<j' \leq j}(B[i'-1][j'-1] + S_4(X[i', i], Z[j', j])) \end{cases} \tag{6}$$

The proofs of the correctness of those formulas are omitted. Because Δ is a constant, it takes $O(1)$ time to calculate (5) and (6) for a single pair of (i, j). So, a dynamic programming can be designed straightforwardly to compute all $G[i, j]$ and $B[i, j]$ in $O(|X| \times |Z|)$ time. And a backtracking can construct the optimal alignment, as well as the optimal Y such that $d(X, Z) = f(X, Y) + g(Y, Z)$.

6. Software Implementation

In this section, we introduce the software SPIDER. SPIDER implements the algorithm in Section 5, as well as some other heuristic algorithms. SPIDER is available on the Internet for free public use at http://bif.csd.uwo.ca/spider.

6.1. *The scoring functions and S4 matrix*

The parameters that determine the software's behavior include

- The *de novo* cost function $f'(m)$ for a mass value m, and $f^*(a)$ for a letter a.
- The homology score matrix $g'(a, b)$ for two letters a and b, and $\bar{g}$ for the insertion/deletion cost.
- The good block length upper bound Δ.
- The α (mismatch), β (insertion/deletion), and γ (bad block open) penalties in a bad block.

In SPIDER, we empirically assume that 80% of the letters can be correctly assigned by the *de novo* sequencing software. Let $P_1 = 0.8$. Let $f^*(a) = -\log_2 P_1$. Let $S(m)$ be the set of segments whose mass is approximately m. $S(m)$ can be computed by dynamic programming using the following recursive definition, $S(m) = \sum_{a\in\Sigma} S(m - m(a))$. The probability that the *de novo* sequencing software assigns a specific *wrong* segment to fill that "mass gap" m can be approximated by $\frac{1-P_1}{|S(m)|-1}$. We subtract one from $|S(m)|$ because $S(m)$ also contains the correct sequence. Then we define $f'(m) = -\log_2 \frac{1-P_1}{|S(m)|-1}$.

There are two special pairs of amino acids, (I, L) and (K, Q), that are hard to distinguish by mass spectrometry. The reason is that $m(\mathtt{I}) = m(\mathtt{L})$ and $m(\mathtt{K}) \approx m(\mathtt{Q})$. Also, $m(\mathtt{AG}) = m(\mathtt{GA}) = m(\mathtt{Q})$. As a result, it is easier for the *de novo* sequencing software to make mistakes on these amino acids. SPIDER slightly adjusted f^* and f' to reflect this fact, as follows: $f^*(I) = f^*(L) = -\log_2(P_1/2)$; $f^*(K) = f^*(Q) = -\log_2(2P_1/3)$; $f'(m(I)) = -\log_2(P_1/2)$; $f'(m(K)) = f'(m(Q)) = -\log_2((1 - \frac{2}{3}P_1)/3)$.

$g(a, b)$ is set to be $-\log_2(Pr(a|b))$, where $Pr(a|b)$ is the probability that letter b is aligned with a letter a in a protein alignment. $Pr(a|b)$ is derived from a BLOSUM90 [11] matrix as follows. Let $Pr(a)$ be the probability that a position of a protein sequence is a. $Pr(a)$ can be computed by a simple statistics in a protein database. Let $B[a, b]$ be the BLOSUM90 score between a and b. From the definition of the BLOSUM matrix [11], we can derive that $B[a, b] = -2\log_2 \frac{Pr(a|b)}{Pr(a)}$. Therefore, we define $g(a, b) = -\log_2 Pr(a|b) = -(B[a, b]/2 + \log_2 Pr(a))$. We also set $\bar{g} = 10$ empirically.

The S4 matrix is computed for all pairs of segments with lengths no greater than $\Delta = 3$. Larger Δ values may give better result but it also unfortunately increases the memory requirement for the S4-matrix exponentially.

The bad-block related costs are empirically set to $\alpha = 6$, $\beta = 6$, $\gamma = 12$.

6.2. *Techniques to Speed Up the Search*

The protein databases that SPIDER searches are usually large. For example, the release 42.12 (15-Mar-2004) of Swiss-Prot database contains 146193 proteins comprising 53888514 amino acids. As a result, it may take a long time to compare the query with every position in the database. Therefore, we implemented the "seeding" heuristics [2,15] to screen out the likely-matching positions, which are then further compared. More specifically, we require the query and the database entry to have three consecutive letter matches before they are further aligned by the algorithm in Section 5. The finding of the three letter

matches can be done efficiently by first building an index table of all the 3-mers of either query or database, then scanning through the other sequence and retrieving the matches in the index table. Once a hit (a three-letter match) is found, we do an extension to both directions using the heuristic method in Section 6.3. If a significant local alignment is found by the "non-gapped homology match mode", we apply the algorithm in Section 5 to build a good alignment.

The heuristics described above is what people usually do in general homology search tools. Our testing showed that the heuristic also works well in SPIDER's search. The speed is greatly improved and the sensitivity of the search is not sacrificed very much.

6.3. *Other Match Modes*

Our algorithm in Section 5 assumes both the *de novo* sequencing error and the homology mutations. Moreover, the error and mutations can occur simultaneously in the same block of the alignment. This is very sophisticated model. In practice, however, there are cases where the situations are simpler. E.g., the target proteins may be in the database therefore we need not to consider the mutations. Or people are willing to use a simpler model to gain speed over search quality. In order to meet the different needs at different situations, we also developed several other match modes in SPIDER software, according to different error models. In this section, we will introduce the four match modes of SPIDER.

Exact Match Mode.

Exact match mode assumes the *de novo* sequencing result has no error, except that L and I, Q and K can be exchangeable. [a]

It further assumes that the target protein is in the database. In this mode, users are also allowed to use a notation, $[m]$, to specify a mass gap of value m. The following shows an alignment between a query tag with mass gaps and a database match.

```
Tag:      [258.1]TLMEYLE[114.0]PK
Match:    [EE   ]TLMEYLE[N    ]PK
```

In such an alignment, $m(\texttt{EE}) \approx 258.1$ and $m(\texttt{N}) \approx 114.0$. The mass gaps are often produced by some *de novo* sequencing software, e.g., Lutefisk, when it cannot confidently determine a segment. Also, if the *de novo* sequencing software such as PEAKS [16] outputs a confidence level to each amino acid assignment, people can generate these mass gaps by turning the low-confidence letters to their mass values [10,25].

This mode runs very fast but requires high confidence *de novo* sequencing results.

Segment Match Mode.

Segment match mode allows segment replacement errors; however, assumes no homology mutations. In one of our examples, the *de novo* sequencing software output a wrong sequence tag ARPKWTPTLVMPSR. Using the segment search, the correct sequence,

[a] because $m(\texttt{I}) = m(\texttt{L})$ and $m(\texttt{Q}) \approx m(\texttt{K})$.

KVPQVSTPTLVEVSR, was found in the database. The alignment between the tag and the correct sequence is the following:

```
Tag:     [AR]PK[W ]TPTLV[MP]SR
Match:   [KV]PQ[VS]TPTLV[EV]SR
```

where mass(AR) $\approx$ mass(KV), mass(W) $\approx$ mass(VS), and mass(MP) $\approx$ mass(EV). The difference between K and Q is usually discarded in this match mode, although users have the option to choose to distinguish them in SPIDER.

So, there are 9 matched letters and 3 matched segments. These numbers are used to define the score of the match. Mass gaps are also allowed in our segment search. For example, because mass(KV)=mass(A)+156.2 and mass(PQ)=225.3, the following alignment is a segment match:

```
Tag:    [156.2A][225.3]VSTPTLVEVSR
Match:  [KV    ][PQ   ]VSTPTLVEVSR
```

In this mode, we use the seeding heuristic to first find hits. Once a hit is found, the algorithm extends both sequences to the same direction, starting from the hit. A "current total mass" value is computed to recorded the total mass of the amino acids that have been extended in the corresponding sequence. The algorithm always extends one more amino acids in the sequence with a smaller "current total mass" value. A block is formed whenever the two "current total mass" values are approximately equal. This procedure can be done in linear time. And therefore, this modes runs very fast. When the queries are batched, this mode, on average, searches the SwissProt database in no more than 1 second per query on a 3GHZ Pentium IV PC. This mode is very useful when the target protein is in the database.

Non-gapped Homology Match Mode.

In this match mode, we makes three assumptions:

(1) All the homology mutations are substitutions. I.e., no insertion/deletion is allowed.

(2) Mutations and *de novo* sequencing errors do not exist simultaneously in the same block.

(3) There are no segment replacements with length greater than three.

Under these assumptions, we use a greedy algorithm to find a good alignment around a hit found by the seeding heuristic. During the extension process, if a mismatch happens, a subroutine similar to our segment match mode is called to find length ≤ 3 segment matches between the sequence tag and the database sequence. If this mass-based match fails, we make the assumption that a substitution happened, and the BLOSUM90 [11] score for the two amino acids at the current position is assigned to this substitution. Then the extension is continued. We keep the local max scores for the extensions at both directions of the hit. The region that gives the highest local score will be recorded and output later. An example of alignment is shown as following. The square brackets indicate the same mass replacements.

```
Tag:      CCQ[W ]DAEAC[AF][NN][PG]K
```

	SwissProt	Human
SPIDER exact	35	13
SPIDER segment	**90**	36
SPIDER non-gapped homology	76	49
SPIDER homology	78	**52**
MS-BLAST	52	24

Table 1. The comparison of SPIDER and MS-BLAST

```
          ||     | |||           |
Match:    CCA[AD]DKEAC[FA][VE][GP]K
```

This mode is almost as fast as the segment match mode. It is very useful when there are only very few *de novo* sequencing errors and mutations. We also allow mass gaps in the query of this mode. We note that this mode of SPIDER is very similar to the OpenSea software [25] developed by independent research effort.

Homology Match Mode.

This is the mode we discussed in Sections 5, 6.1, and 6.2. Homology search allows both *de novo* sequencing errors and mutations of proteins. This mode is suitable for the search of proteins that are not present in the database. The homolog of the protein, however, must be in the database.

This match mode is considerably slower than the other modes but still runs reasonably fast. When the queries are batched, on average, the searches in the SwissProt database take 3 seconds per query sequence on a 3GHZ Pentium IV PC. One reason the search is fast is the use of the S4 matrix and the limitation of $\Delta = 3$ as described in Sections 5, 6.1, and 6.2.

7. Experiments

De novo sequencing software PEAKS 2.0 [16] was used to obtain 144 sequence tags from 144 Ion Trap MS/MS spectra. The spectra were generously provided by Dr. Richard Johnson who is an author of Lutefisk [28,29]. Most of these spectra were obtained from proteins ALBU_BOVIN, MLE3_RATR, PHS2_RABIT and OVAL_CHICK. All these 144 sequence tags were searched individually against two database, SwissProt (146193 proteins) and SwissProt human protein database (28926 proteins), using four match modes of SPIDER and MS-BLAST, respectively. All these proteins are in SwissProt but not human database. However, homologs of all the proteins can be found in the human database. As a result, searching in SwissProt database can test the ability of peptide identification, and searching in human database can test the ability of finding homologs. The numbers of successfully identified peptides or homologs are shown in table 1.

When searching in SwissProt database, all target proteins are in the database. Therefore, SPIDER segment match mode had the best performance since it does not impose any limitation on the length of segments replacement errors. However, when search is done in

the human database, the SPIDER homology match mode identified the greatest number of homologs. Except for the exact match mode, all the other three match modes of SPIDER perform better than MS-BLAST on the searches of both databases. Also, SPIDER found most sequences that were found by MS-BLAST.

The following is one of several examples from our data, where *de novo* errors and homology mutations occur at same positions.

```
      Tag(X):  CCQ[W ]DAEAC[AF]<NN><PG>K
     Real(Y):  ||| AD |A|A| FA  VE  GP |
Database(Z):  CCK[AD]DKETC[FA]<EE><GK>K
```

In this example, a partially correct tag $X =$ CCQWDAEACAFNNPGK is searched for in human database, and a homolog $Z =$ CCKADDKETCFAEEGKK of the real peptide Y was found. It is noteworthy that the correct sequence Y was not known before the search, and Y is not in the database either. However, SPIDER constructed it correctly from both X and Z, except that the VE in Y can be also EV. We also note that there are both *de novo* errors and mutations in the blocks surrounded by angle brackets. For this reason, the non-gapped homology mode was not able to find this match.

Because all the proteins are known to be in SwissProt database, it is not surprising that SPIDER segment search in SwissProt achieved the best performance, about 63% success rate, in Table 1. [b] This rate is not higher mainly because the data quality produced by an Iontrap MS/MS spectrometer are generally much lower than other MS/MS spectrometers such as Q-Tof. Therefore, despite that PEAKS is considered to be the top *de novo* sequencing software, many sequence tags it produced on this particular dataset contain too many errors, and do not have enough information for the matching with the correct peptides. This difficult dataset was selected deliberately to demonstrate the ability that SPIDER corrects the *de novo* sequencing errors.

8. Conclusion

De novo sequencing using MS/MS data often produces sequence tags with errors. Using the sequence tags with errors to search for the homologs of the real peptides in a database is a very useful and challenging problem. We modeled this problem as the SPIDER's alignment problem, and provided an efficient algorithm to find the optimal alignment. To further speed up the search, a "simplified" model was developed and efficient algorithm was also developed under this model. Finally, the SPIDER software was developed to implement our algorithms and several heuristics for the best performance in different situations.

Our method not only can find the homologs using sequence tags, but also can "guess" the real peptide sequences at the same time. Our method can be straightforwardly extended to incorporate the post-translational modifications in the same manner of incorporating homology mutations.

[b]When all target proteins are in the database, it is likely that software such as SEQUEST and Mascot, which match uninterpreted spectra with peptides in the database, will give better results than SPIDER's approach. The first column of Table 1 is here for the comparison with MS-BLAST only.

Acknowledgement

This research was undertaken, in part, thanks to funding from NSERC, PREA, and the Canada Research Chairs Program. BM thanks Drs. Ming Li and Sheng Yu for their discussion at the early stage of this research.

References

1. Aebersold, R.; and Mann, M. "Mass spectrometry-based proteomics", *Nature* **2003, 422**, 198-207.
2. Altschul, S.F.; Gish, W.; Miller, W.; Myers, E.W.; and Lipman, D.J. "Basic local alignment search tool", *J. Mol. Biol.* **1990, 215**, 403–410.
3. Altschul, S.F.; Madden, T.L.; Schäffer, A.A.; Zhang, J.; Zhang, Z.; Miller, W.; and Lipman, D.J. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", *Nucleic Acids Res.* **1997, 25**, 3389-3402.
4. Bafna, V.; and Edwards, N. "On de novo interpretation of tandem mass spectra for peptide identification", *RECOMB 2003*, Berlin, Germany, 2003.
5. Bartels, C. "Fast algorithm for peptide sequencing by mass spectroscopy", *Biomed. Environ. Mass Spectrom.* **1990, 19**, 363-368.
6. Chen, T.; Kao, M-Y.; Tepel, M.; Rush, J.; and Church, G. "A dynamic programming approach to *de novo* peptide sequencing via tandem mass spectrometry", *J. Comp. Biology* **2001, 8(3)**, 325-337.
7. Dančík, V.; Addona, T.; Clauser, K.; Vath, J.; and Pevzner, P. "*De novo* protein sequencing via tandem mass-spectrometry", *J. Comp. Biology* **1999, 6**, 327-341.
8. Eng, J.K.; McCormack, A.L.; and Yates, J.R.,III "An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database", *J. Am. Soc. Mass Spectrom.* **1994, 5**, 976-989.
9. Fernández-de-Cossío, J.; Gonzalez, J.; Betancourt, L.; Besada, V.; Padron, G.; Shimonishi, Y.; and Takao, T. "Automated Interpretation of High-Energy Collision-Induced Dissociation Spectra of Singly-Protonated Peptides by "SeqMS", a Software Aid for *De Novo* Sequencing by MS/MS", *Rapid Commun. Mass Spectrom.* **1998, 12**, 1867-1878.
10. Han,Y.; Ma, B.; and Zhang K. "Gapped Sequence Tag Query for Improving *De Novo* Sequencing Results", **2003**, *ASMS*.
11. Henikoff, S.; and Henikoff, J.G. "Amino acid substitution matrices from protein blocks", *P*roc. Natl. Acad. Sci. U.S.A. **1992, 89**, 10915-10919.
12. Hines, W.M.; Falick, A.M.; Burlingame, A.L.; and Gibson, B.W. "Pattern-based algorithm for peptide sequencing from tandem high energy collision-induced dissociation mass spectra", *J. Am. Sco. Mass. Spectrom.* **1992, 3**, 326-336.
13. Huang, L.; Jacob, R.J.; Pegg, S.C.-H.; Baldwin, M.A.; Wang, C.C.; Burlingame, A.L.; and Babbitt, P.C. "Functional Assignment of the 20 S Proteasome from Trypanosoma brucei Using Mass Spectrometry and New Bioinformatics Approaches", *J. Biol. Chem.* **2001, Vol. 276, Issue 30**, 28327-28339.
14. Lu, B.; and Chen, T. "A suboptimal algorithm for *de novo* peptide sequencing via tandem mass spectrometry", *J. Comp. Biology* **2003, 10**, 1-12.
15. Ma, B.; Tromp, J.; and Li, M. "PatternHunter: faster and more sensitive homology search", *Bioinformatics* **2002, 18(3)**, 440-445.
16. Ma, B.; Zhang, K.; Lajoie, G.; Doherty-Kirby, A.; Hendrie, C.; Liang, C., and Li, M. "PEAKS: powerful software for peptide *de novo* sequencing by MS/MS", *Commun. Mass Spectrom.* **2003, 17(20)**, 2337-2342.
17. Ma, B.; Zhang, K.; and Liang C. "An effective algorithm for the peptide *de novo* sequencing from MS/MS spectrum", to appear in *Journal of Computer and System Sciences.*.

18. Mackey, A.J.; Haystead, T.A.J. and Pearson, W.R. "Getting More for Less: Algorithms for Rapid Protein Identification with Multiple Short Peptide Sequences", *Mol. Cell. Proteomics* **2002, 1**, 139-147.
19. Mann, M.; and Wilm, M. "Error-tolerant identification of peptides in sequence databases by peptide sequence tags", *A*nal. Chem. **1994, 66**, 4390-4399.
20. Pearson, W.R.; and Lipman, D.J. "Improved tools for biological sequence comparison", *Proc. Natl. Acad. Sci.* U.S.A. **1988, 85**, 2444-2448.
21. Pegg, S.C.; and Babbitt, P.C. "Shotgun: getting more from sequence similarity searches", *Bioinformatics* **1999, 15**, 729-740.
22. Perkins, D.N.; Pappin, D.J.C.; Creasy, D.M.; and Cottrell, J.S. "Probability-based protein identification by searching sequence database using mass spectrometry data", *E*lectrophoresis **1999, 20**, 3551-3567.
23. Pevzner, P.A.; Dančík, V.; and Tang, C. "Mutation tolerant protein identification by mass spectrometry", *J. Comp. Biology* **2000, 6**, 777-787.
24. Sakurai, T.; Matsuo, T.; Matsuda, H.; and Katakuse, I. "Paas3: A computer program to determine probable sequence of peptides from mass spectrometric data" *B*iomed. Mass spectrum. **1984, 11(8)**, 396-399.
25. Searle, B.C.; Dasari, S.; Turner, M.; Reddy, A.P.; Choi, D.; Wilmarth, P.A.; McCormack, A.L.; David, L.L.; and Nagalla, S.R. "High-throughput identification of proteins and unanticipated sequence modifications using a mass-based alignment algorithm for MS/MS *de novo* sequencing results", *Anal. Chem.* **2004, 76**:2220-2230.
26. Shevchenko, A.; Sunyaev, S.; Loboda, A.; Shevchenko, A; Bork, P.; Ens, W.; and Standing, K.G. "Charting the proteomes of organisms with unsequenced genomes by MALDI-quadrupole time-of-flight mass spectrometry and BLAST homology searching", *Anal Chem.* **2001, 73(9)**, 1917-26
27. Tabb, D.L.; Saraf, A.; and Yates, J.R., III "GutenTag: High-Throughput Sequence Tagging via an Empirically Derived Fragmentation Model", *Anal. Chem.* **2003, 75**, 6415-6421.
28. Taylor, J.A.; and Johnson, R.S. "Sequence database searches via *de novo* peptide sequencing by tandem mass spectrometry", *Rapid Commun. Mass Spectrom.* **1997, 11**, 1067-1075.
29. Taylor, J.A.; and Johnson, R.S. "Implementation and uses of automated *de novo* peptide sequencing by tandem mass spectrometry", *Anal. Chem.* **2001, 73**, 2594 - 2604.
30. Uttenweiler-Joseph, S.; Neubauer, G.; Christoforidis, S.; Zerial, M.; and Wilm, M. "Automated de novo sequencing of proteins using the differential scanning technique", *Proteomics* **2001, 1**, 668-682.
31. Yates, J.R.I.; Eng, J.K.; McCormack, A.L.; and Schieltz, D. "Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database", *Anal. Chem.* **1995, 67**, 1426-36.