BIOINFORMATICS ORIGINAL PAPER

Vol. 31 no. 4 2015, pages 532–538
doi:10.1093/bioinformatics/btu678

Sequence analysis

Advance Access publication October 15, 2014

An efficient algorithm for the blocked pattern matching problem

Fei Deng[1], Lusheng Wang[1,*] and Xiaowen Liu[2,3,*]

[1]Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, [2]Department of BioHealth Informatics, Indiana University—Purdue University Indianapolis and [3]Center for Computational Biology and Bioinformatics, Indiana University School of Medicine, Indianapolis, IN 46202, USA

Associate Editor: John Hancock

ABSTRACT

Motivation: Tandem mass spectrometry (MS) has become the method of choice for protein identification and quantification. In the era of big data biology, tandem mass spectra are often searched against huge protein databases generated from genomes or RNA-Seq data for peptide identification. However, most existing tools for MS-based peptide identification compare a tandem mass spectrum against all peptides in a database whose molecular masses are similar to the precursor mass of the spectrum, making mass spectral data analysis slow for huge databases. Tag-based methods extract peptide sequence tags from a tandem mass spectrum and use them as a filter to reduce the number of candidate peptides, thus speeding up the database search. Recently, gapped tags have been introduced into mass spectral data analysis because they improve the sensitivity of peptide identification compared with sequence tags. However, the blocked pattern matching (BPM) problem, which is an essential step in gapped tag-based peptide identification, has not been fully solved.

Results: In this article, we propose a fast and memory-efficient algorithm for the BPM problem. Experiments on both simulated and real datasets showed that the proposed algorithm achieved high speed and high sensitivity for peptide filtration in peptide identification by database search.

Contact: cswangl@cityu.edu.hk or xwliu@iupui.edu

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

Received on June 25, 2014; revised on October 1, 2014; accepted on October 13, 2014

1 INTRODUCTION

With the developments of high-throughput genomics, transcriptomics and proteomics, combining multiple omics data provides a new way for studying complex biological systems. In multiple omics data analysis, tandem mass (MS/MS) spectra are searched against protein databases generated from genomes or RNA-Seq data for peptide identification. These protein databases are often huge. For example, in human microbiome studies, MS/MS spectra are searched against databases containing protein sequences generated from hundreds, even thousands, of genomes (Dewhirst *et al*., 2010; Rudney *et al*., 2010).

Most mass spectrometry (MS)-based peptide identification tools match an MS/MS spectrum against all peptides in a database whose molecular masses are similar to the precursor mass of the spectrum (Craig and Beavis, 2003; Eng *et al*., 1994; Geer *et al*., 2004; Perkins *et al*., 1999). These tools become slow when the database is large. An alternative approach is based on peptide sequence tags: peptide sequence tags are extracted from an MS/MS spectrum and searched against a database to identify a small number of *candidate peptides* that match at least one of the tags, then a rescoring function is applied to report the highest scoring candidate peptide (Ma *et al*., 2011; Mann and Wilm, 1994; Tabb *et al*., 2003; Tanner *et al*., 2005). The trade-off between speed and sensitivity is an important problem in tag-based methods. To increase the sensitivity, it is essential that the set of sequence tags extracted from a spectrum contains a correct one that matches the target peptide. To increase the speed, long sequence tags (length ≥ 5) are preferred because they are efficient in filtering out incorrect peptides. As a result, correct long sequence tags are extremely useful in tag-based methods because they speed up database searches without losing sensitivity. However, correct long sequence tags may not be found in MS/MS spectra because of missing and noise peaks (Jeong *et al*., 2011).

To utilize long sequence tags, Kim *et al*. (2009) proposed to generate a spectral dictionary containing full-length peptide constructions from an MS/MS spectrum for peptide identification. A drawback of this approach is that a spectral dictionary of an MS/MS spectrum with a large precursor mass may contain billions of peptides. Jeong *et al*. (2011) introduced gapped peptides to solve this problem. Since a gapped peptide represents many non-gapped peptides, the use of gapped peptides significantly reduces the sizes of spectral dictionaries and speeds up peptide identification without losing sensitivity.

Similar to the method proposed by Jeong *et al*. (2011), replacing sequence tags with *gapped tags* can achieve both high speed and high sensitivity in tag-based peptide identification. A gapped tag is represented by a sequence of mass values instead of amino acids. Each mass value is the mass of one amino acid residue or the sum of the masses of several consecutive amino acid residues. Many MS/MS spectra do not contain long sequence tags because of missing peaks, but contain long gapped tags.

Gapped tags and gapped peptides give rise to the blocked pattern matching (BPM) problem in which a blocked pattern generated from a gapped tag is searched against a text to find matched substrings. Ng *et al*. (2011) proposed an $O(m)$ algorithm for querying a blocked pattern of length m against a text of length n. However, the time and space needed for the preprocessing step of the algorithm is $O(2^k)$, where k is the length of the longest substring to identify. When $k = 30$, the preprocessing step is slow and the space required by the algorithm is huge.

*To whom correspondence should be addressed.

In this article, we present a fast and memory-efficient algorithm for the BPM problem. The time and space complexity of the preprocessing step of the algorithm is $O(n)$. In the application of MS-based peptide identification, the time complexity for querying a blocked pattern is sublinear in n. Experiments on both simulated and real MS/MS data sets showed that the proposed algorithm achieved high speed and high sensitivity for peptide filtration in peptide identification by database search.

2 METHODS

2.1 The BPM problem

Let $\mathcal{N}$ be the set of all natural numbers and Σ a subset of $\mathcal{N}$. A string over $\mathcal{N}$ is called a *pattern string*, and a string over Σ is called a *text string*. When a pattern string is searched against a text string, the pattern string is called a *blocked pattern*. Let $S = s_1, s_2, \ldots, s_n$ be a text string. The mass of a substring $s_i, s_{i+1}, \ldots, s_j$ of S is the sum of all numbers in the substring, i.e. $s_i + s_{i+1} + \cdots + s_j$. Substrings $s_i, s_{i+1}, \ldots, s_j$ and $s_{j+1}, s_{j+2}, \ldots s_k$ are called *consecutive substrings* of S. A *partition* R of S is a sequence of consecutive substrings $A_1, A_2, \ldots, A_k$ of S where A_1 starts from s_1 and A_k ends at s_n. The *mass string* of the partition R is a string comprised of the masses of the consecutive substrings of R. For example, when $\Sigma = \{71, 113, 114, 128\}$, $S = 71, 113, 114, 128$, $A_1 = 71$, $A_2 = 113, 114$ and $A_3 = 128$, the substrings A_1, A_2, A_3 is a partition of S and the corresponding mass string is 71, 227, 128, where 227 is the mass of A_2. A blocked pattern *matches* a text string if there is a partition of the text string such that the mass string of the partition is identical to the blocked pattern. In the previous example, the blocked pattern 71, 227, 128 matches the text string 71, 113, 114, 128.

2.1.1 The BPM problem Given a text $T = t_1, t_2, \ldots, t_n$ over a finite alphabet $\Sigma \subset \mathcal{N}$ and a blocked pattern $P = p_1, p_2, \ldots, p_m$ over $\mathcal{N}$, find all substrings in T that match P.

2.2 An efficient algorithm for the BPM problem

In this subsection, we present an efficient algorithm for the BPM problem, in which the text T is represented by a suffix tree. Although the proposed algorithm works for suffix trees in which each edge is labeled with a text string with one or more letters, here we assume that each edge is labeled with only one letter to simplify the following analysis. Before discussing the new algorithm, we review the steps to search a text string against a suffix tree. To find substrings in T matching a text string $a_1, a_2, \ldots, a_m$, we start from the root N_0 of the suffix tree and find a list of nodes $N_1, N_2, \ldots, N_m$ such that the path from the root to N_i, for $1 \le i \le m$, spells out the prefix $a_1, a_2, \ldots, a_i$. In addition, the letter a_i is the label on the edge connecting N_{i-1} and N_i.

We cannot search the blocked pattern P against the suffix tree directly since the text T is over Σ ($\Sigma \subset \mathcal{N}$), whereas P is over $\mathcal{N}$. Alternatively, we can first convert P into the set Ω of all text strings matching P and then search each text string in Ω against the suffix tree. Let V_i be the set of all text strings whose masses equal p_i. An example of V_i is shown in Supplementary Table S1. Each text string in Ω is the concatenation of a list of text strings $v_1, v_2, \ldots, v_m$ in which $v_i \in V_i$ for $1 \le i \le m$. The number of text strings in Ω is $\Pi_{i=1}^{m} |V_i|$, which may introduce a combinatorial explosion when $|V_i| > 1$ for most p_i in P. As a result, it is not fast to search P by converting it to Ω explicitly.

Prefixes of P are used to speed up blocked pattern searches. A text string S is a *prefix text string* of P if S matches a prefix pattern string of P. Moreover, if a prefix text string of P is identical to a substring of T, we say it is an *identifiable prefix text string* of P. For example, when $\Sigma = \{71, 113, 114, 128\}$, $P = 71, 227, 128$, and $T = 87, 71, 113, 114, 128, 97$, both 71, 113, 114 and 71, 114, 113 are prefix text strings of P. The string 71, 113, 114 is an identifiable prefix text string of P, but 71, 114, 113 is not. If a prefix text string is not identifiable, then all text strings in Ω with the prefix are not identifiable, making it not necessary to generate and search these text strings.

The BPM Algorithm

Input: A text T of length n over a finite alphabet $\Sigma \subset \mathcal{N}$ and a blocked pattern P of length m over $\mathcal{N}$.
Output: All substrings in T that match P.

1. Use a suffix tree to represent T, initialize U_0 as the set containing only the root of the tree, and compute $V_1, V_2, \ldots, V_m$.
2. **For** $i = 1$ to m **do**
3. **For** each node $u \in U_{i-1}$ and each text string $S \in V_i$ **do**
4. Search S against the suffix tree starting from u. If there exists a path from u to a node v that spells out the text string S, add v into U_i.
5. **EndFor**
6. **EndFor**
7. Report all substrings corresponding to U_m and their positions in T.

Fig. 1. An efficient algorithm for the BPM problem

The idea of the BPM algorithm is to speed up blocked pattern searches by removing non-identifiable prefix text strings (Fig. 1). Each identifiable prefix text string corresponds to a unique node in the suffix tree. Let U_i be the set of nodes corresponding to all identifiable prefix text strings that match the prefix $p_1, p_2, \ldots, p_i$ of P. Similar to searching a text string, we initialize U_0 as the set containing only the root of the tree and find U_1, $U_2, \ldots, U_m$ progressively. The set U_m contains the solution to the BPM problem because the identifiable prefix text strings that match $p_1, p_2, \ldots, p_m$ are substrings in T that match P. To find U_i from U_{i-1}, we start from each node $u \in U_{i-1}$ and search for each text string $S \in V_i$. If there exists a node v such that the path from u to v spells out the string S, then the node v is added to U_i. The total number of searches in the iteration, i.e. finding U_i from U_{i-1}, is $|U_{i-1}| \times |V_i|$. After the last set U_m is found, the BPM problem is solved by reporting all substrings corresponding to U_m and their positions in T, which are stored in U_m. Since the text T is represented by a suffix tree, the space complexity of the algorithm is $O(n)$.

2.3 Time complexity

Representing the text T as a suffix tree is a preprocessing step for pattern queries. Its time complexity is $O(n)$. Let G be the largest number in the blocked pattern P, that is, $G = \max_{i=1}^{m} p_i$. We denote by W_k the set of all text strings whose masses equal k. To obtain $V_1, V_2, \ldots, V_m$, we generate a lookup table $W_0, W_1, \ldots, W_G$. The number of operations for computing $W_0, W_1, \ldots, W_G$ is an exponential function of G (see the Supplementary Material). In peptide identification by tandem mass spectra, the value G is not extremely large and the sets $W_0, W_1, \ldots, W_G$ can be computed in a short time (see Section 3.2). Below we focus on the query time complexity of the BPM algorithm (steps 2–7 of the algorithm in Fig. 1).

When prefix text strings of P are long, most of them cannot be found in the text T (i.e. they are not identifiable) and will be removed from further consideration. As a result, the size of U_i is often smaller than $|U_{i-1}| \times |V_i|$. That is the reason why pattern queries in the BPM algorithm are fast.

Let $N = \max_{i=1}^{m} |V_i|$. The running time of searching P against the suffix tree is determined by $\sum_{i=1}^{m} |U_i| \cdot N$. Let L be the length of the longest prefix text string of P. We define X_l as the set of all length l prefix text strings of P and Y_l as the set of all length l identifiable prefix strings of P, for $1 \le l \le L$. Since each node in $\cup_{i=1}^{m} U_i$ corresponds to an identifiable

prefix text string in $\cup_{l=1}^{L} Y_l$, we have $\sum_{i=1}^{m} |U_i| = \sum_{l=1}^{L} |Y_l|$. The expectation of $|Y_l|$ is determined by the size of X_l and the probability that a length l text string is found in T. As a result, the running time of the pattern query is related to the sizes of Y_l and X_l.

Each length l prefix text string in X_l has only one partition $A_1, A_2, \ldots, A_k$ such that the mass string of the partition, which is a length k pattern string, is the same to a prefix of P. Let l_i be the length of A_i for $1 \leq i \leq k$. We say $l_1, l_2, \ldots, l_k$ is the *configuration* of the prefix text string. For example, when $\Sigma = \{2, 3\}$ and P = 3, 6, 4, we have $V_1 = \{(3)\}$, $V_2 = \{(3, 3), (2, 2, 2)\}$ and $V_3 = \{(2, 2)\}$. The configuration of the length 4 prefix text string 3, 2, 2, 2 is 1, 3 because it has only one partition $A_1 = 3$ and $A_2 = 2, 2, 2$ such that the mass string of the partition matches the prefix 3, 6 of P. The total number of different configurations of length l prefix text strings of P is bounded by 2^l.

Let X_l^C be the set of all prefix text strings in X_l with a configuration $C = l_1, l_2, \ldots, l_k$ satisfying $l = \sum_{i=1}^{k} l_i$. Each prefix text string in X_l^C is the concatenation of k text strings $v_1, v_2, \ldots, v_k$ such that $v_i \in V_i$ and the length of v_i is l_i for $1 \leq i \leq k$. We divide V_i into subsets $V_{i,1}, V_{i,2}, \ldots, V_{i,d}$, where d is the length of the longest text string in V_i. The subset $V_{i,j}$ contains all length j text strings in V_i. The size of X_l^C is $\Pi_{i=1}^{k} |V_{i,l_i}|$. Because the probability that a prefix text string in X_l^C is found in T is a function of l, we want to represent the upper bound of $|X_l^C|$ as a function of l. To this end, we introduce an *expansion factor* $r_{i,j} = |V_{i,j}|^{1/j}$ for each set $V_{i,j}$. The largest expansion factor is denoted as $r = \max_{i,j} r_{i,j}$. Since $|V_{i,l_i}| \leq r^{l_i}$ for $1 \leq i \leq k$,

$$|X_l^C| = \Pi_{i=1}^{k} |V_{i,l_i}| \leq \Pi_{i=1}^{k} r^{l_i} = r^{\sum_{i=1}^{k} l_i} = r^l.$$

Because the number of configurations of prefix text strings in X_l is bounded by 2^l, we can prove that $(2r)^l$ is an upper bound of $|X_l|$ and further give the average-case time complexity of a pattern query in the BPM algorithm.

THEOREM 1. *When the size of* Σ *is a constant and* $1 < 2r < |\Sigma|$, *the average-case time complexity of a pattern query in the BPM algorithm is* $O(n^{\log_{|\Sigma|} 2r} N + M)$, *where* M *is the number of matched substrings in* T.

A proof of the above theorem is given in the Supplementary Material. The average-case time complexity is obtained by analyzing the expectation of the size of Y_l. In the worst case, the size of Y_l is bounded by the length of the text. Consequently, the worst-case time complexity of a pattern query in the BPM algorithm is $O(mnN)$.

2.4 Peptide identification by BPM

In peptide identification, blocked patterns are often obtained from gapped tags. A gapped tag of a spectrum is represented by a list of masses, each of which is the distance between two peaks in the spectrum, which may correspond to one or more amino acids. Figure 2 gives an example spectrum for the peptide FTALNQVR from which we can generate a gapped tag 71.04, 227.13, 128.06 rather than a sequence tag ALNQ because the peak for the fragment FTAL is missing. A gapped tag is converted into a blocked pattern by rounding each mass in the gapped tag into an integer. Although rounding masses to integers has been successfully applied to analyzing low accuracy mass spectra (Kim *et al.*, 2008), it may introduce large errors in analyzing high accuracy mass spectra. For example, a large error 0.23 is introduced by rounding the mass 426.23 (with an accuracy of 0.01) of the fragment ion ALNQ in Figure 2 to 426. To avoid this problem, mass values are multiplied by a scale factor before rounded to integers (Liu *et al.*, 2013). For example, when the scale factor is 100, the mass of the amino acid residue A is converted into 7 104. High-accuracy fragment ion peaks and a large scale factor are important in reducing the number of text strings matched to a blocked pattern and speeding up pattern queries. Since the BPM algorithm is designed for high-accuracy MS/MS spectra with high-accuracy

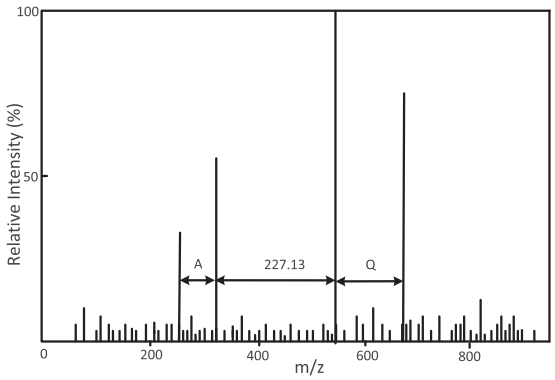

Fig. 2. An example spectrum for the peptide FTALNQVR. A gapped tag 71.04, 227.13, 128.08 is extracted from the spectrum based on the distances between peak pairs, where 71.04 is the mass of the amino acid residue A, 227.13 is the sum of the masses of the amino acid residues L and N and 128.08 is the mass of the amino acid residue Q. A blocked pattern 71, 227, 128 is generated by rounding each mass in the gapped tag to an integer

precursors, it may fail to efficiently search blocked patterns generated from low-accuracy MS/MS spectra.

Blocked patterns can also be generated from sequence tags when local confidence scores of amino acids in the sequence tags are available (Ma *et al.*, 2003; Tabb *et al.*, 2003). In general, amino acids with low confidence scores tend to be incorrect. A sequence tag is converted into a gapped tag by replacing neighboring amino acids that have low confidence scores with the sum of their masses and substituting an amino acid that has a high confidence score with its single mass. The gapped tag is further converted into a blocked pattern by rounding its masses to integers. In practice, gaps in gapped tags are no larger than 500 Da (Ng *et al.*, 2011). When a sequence tag contains several consecutive amino acids with low confidence scores such that the sum of their masses is larger than 500 Da, the sequence tag is broken into two short sequence tags by the large gap from which gapped tags are generated. The threshold for confidence scores determines which amino acids in a sequence tag are combined. When the threshold is lower than the smallest confidence score in a sequence tag, every amino acid in the sequence tag is converted to a mass in the gapped tag. When the threshold is extremely high, all amino acids in a sequence tag is combined to a large mass and the search of the blocked pattern is equivalent to peptide identification by using its precursor mass.

In peptide identification, the residue mass values of the 20 standard amino acid are converted into integers by using a scale factor. The alphabet Σ of text strings is the set of the resulting integers. Since leucine (L) and isoleucine (I) have the same mass, they are treated as the same in blocked pattern searches. As a result, the size of Σ is 19. A protein database is converted to a text over Σ by replacing each amino acid in the database by its corresponding integer in Σ.

Blocked patterns are generated from a tandem mass spectrum and searched against the text translated from a protein database to find candidate peptides. After the candidate peptides are identified, they are filtered by the precursor mass of the spectrum. The resulting peptide-spectrum-matches (PSMs) are further rescored to find the highest scoring one (Supplementary Fig. S2). Any scoring function can be utilized in the rescoring step to improve the sensitivity of peptide identification. Searching blocked patterns against the text generated from a database is an essential step that determines the speed and sensitivity of peptide identification.

In peptide identification by tandem mass spectra, the numbers (letters) in P are usually not extremely large. When the largest gap in gapped tags

is 500 Da and the scale factor is 100, the largest number in P is 50 000. In this case, the number N in Theorem 1 can be treated as a constant value. In addition, $\log_{|\Sigma|} 2r$ in Theorem 1 is usually smaller than 1. In this application, the average-case time complexity for a pattern query is sublinear in the size of the database.

The numbers in a blocked pattern may have small errors that are introduced in measuring the m/z values of fragment ions. To account for these errors, an error tolerance is allowed when blocked patterns are searched against a text. In the BPM algorithm, a text string is included into the set V_i if the difference between p_i and the sum of all the numbers in the text string is within a predefined error tolerance.

3 RESULTS

We implemented the BPM algorithm in Java and tested it on both simulated and real datasets. All the tests were performed on a Linux (64-bit) desktop PC with a 1.4 GHz CPU and 32 GB RAM. We were not able to compare the BPM algorithm with that described in Ng *et al*. (2011) because it is not available.

3.1 Dataset

A dataset generated from human cell lysate (Frese *et al*., 2011) was used to test the BPM algorithm. In the preparation of the dataset, the protein mixture was analyzed on an Orbitrap Velos (Thermo Fisher Scientific) coupled with a high-performance liquid chromatography system. High-accuracy MS and MS/MS spectra were collected at a resolution of 30 000 and 7500, respectively. Triplicate higher energy collisionally activated dissociation datasets were then acquired, of which only one with 37 810 tandem mass spectra was selected to evaluate the BPM algorithm. Details of the experiment can be found in Frese *et al*. (2011).

3.2 Running time analysis

The human proteome database (about 12 MB) was downloaded from the Swiss-Prot database. We constructed a suffix tree from the human proteome database using the Ukkonen's algorithm (Ukkonen, 1995) implemented by Nelson and ported to Java by Havsiyevych (The source code of the implementation can be found at http://illya-keeplearning.blogspot.com/2009/04/suffix-trees-java-ukkonens-algorithm.html). The running time for the suffix tree construction was about 30 s.

In the implementation of the BPM algorithm, the scale factor is 100. Let Σ be the alphabet generated from the 19 amino acids (the amino acids I and J are treated as the same). In practice, gaps typically do not exceed 500 Da (Ng *et al*., 2011), and the largest number in blocked patterns is 50 000. For each number k between 0 and 50 000, we generated the set W_k of all text strings on Σ such that the sum of the numbers in the text string is the same to k with an error tolerance of 5. The size of the lookup table $W_1, W_2, \ldots, W_{50000}$, that is, $\sum_{k=1}^{50000} |W_k|$, is about 3.34×10^6. The running time for generating the lookup table was about 1 s. In the following analyses, we will ignore the preprocessing time for generating the suffix tree and the lookup table and only consider spectral query time unless otherwise stated.

The value $\log_{|\Sigma|} 2r$ defined in Section 2 is the exponent that determines the time complexity of the BPM algorithm. We estimated the value of r when the largest number in a blocked pattern is 50 000. We divided W_k into subsets $W_{k,1}, W_{k,2}, \ldots, W_{k,j}, \ldots$, where $W_{k,j}$ contains all length j text strings in W_k. Let $r_k = \max_j |W_{k,j}|^{\frac{1}{j}}$. The histogram of r_k shows that $r_k \leq 5.7$ (Supplementary Fig. S3). Since $\log_{19}(2 \times 5.7) \approx 0.83$, the average-case time complexity of the BPM algorithm is sublinear in the size of the database in this application.

When a blocked pattern of length m is generated from a sequence tag with n amino acids, we define n/m as the *gap ratio* of the pattern. For example, the blocked pattern 71, 227, 128 can be generated from a sequence tag ALNQ (71, 113, 114, 128), and the gap ratio is 4/3. We tested the BPM algorithm on simulated data to find the relationship between the running time and the gap ratio. Blocked patterns were generated directly from peptides in the human proteome database: A total of 1 million peptides of length 20 were randomly selected from the database. Given a peptide $A = a_1, a_2, \ldots, a_{20}$, a block $A[i, j]$ $(1 \leq i \leq j \leq 20)$ denotes the substring of $a_i, a_{i+1}, \ldots, a_j$. To construct a blocked pattern from A, we first partition it into blocks of length k (possibly except for the last block), i.e. $A[1, k], A[k+1, 2k], \ldots, A[lk+1, 20]$ $(l = \lceil \frac{20}{k} \rceil - 1)$, and then sum up the rounded scaled masses (the scale factor is 100) of the amino acids in each block. The resulting list of $\lceil \frac{20}{k} \rceil$ numbers is a blocked pattern with a gap ratio about k. Four patterns were constructed for each peptide for k = 1, 2, 3 and 4. When the gap ratio increases from 2 to 4, the range of the largest extension factors r for the blocked patterns changes from [1,4.3] to [2.1,5.7] (Supplementary Fig. S4). The blocked patterns were then searched against the suffix tree constructed from the human proteome database. The average query time for blocked patterns was fast for all four gap ratios (Supplementary Fig. S5). When the gap ratio was 4, the average query time was less than 0.003 s per pattern. In addition, the average query time significantly increased as the gap ratio increased, so large gaps should be avoided to guarantee that the query time is fast in practice.

3.3 Evaluation on the real dataset

To evaluate the performance of the BPM algorithm, we generated a set of PSMs from the human lysate dataset. MS-GF+ (Kim *et al*., 2008) was employed to search each spectrum in the dataset against a target-decoy concatenated human proteome database. With 1% spectrum level false discovery rate (FDR), MS-GF+ identified 13 135 spectra (in about 18 min), which are assumed correct and used as a gold standard for the following evaluation. The 13 135 spectra are called *identifiable spectra*. For brevity, the set of 13 135 spectra is referred to as the HUMAN dataset.

3.3.1 Evaluation criteria In peptide identification, a set of blocked patterns are extracted from a spectrum and searched against a protein database to find candidate peptides that contain at least one blocked pattern. If one of the extracted blocked patterns is correct (in the target peptide) and the filtration based on the blocked patterns reports a set of candidate peptides that contains the target one, then the spectrum is said to be efficiently filtered. The filtration efficiency is defined as follows:

$$\text{filtration efficiency} = \frac{\text{Number of identifiable spectra that can be efficiently filtered}}{\text{Number of identifiable spectra}}.$$

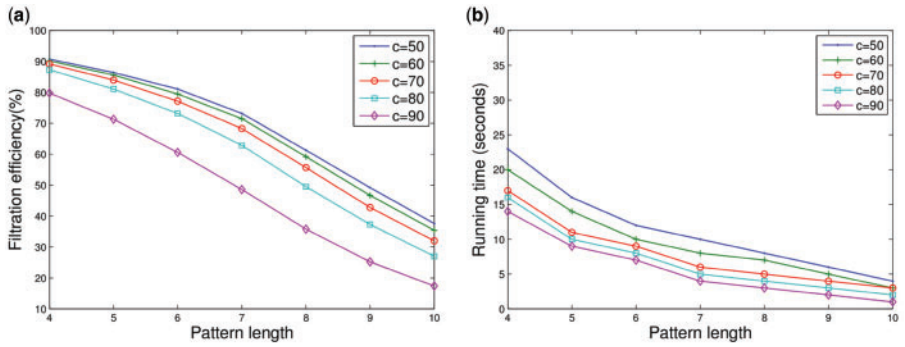

Fig. 3. Performance of the BPM algorithm with various settings of the pattern length and the confidence score threshold c on the HUMAN dataset: (a) the filtration efficiency and (b) the running time

The second evaluation criterion is the query speed of blocked patterns. The speed of peptide identification is also related to the number of the candidate peptides reported by blocked pattern-based filtration, which is the third evaluation criterion.

3.3.2 Pattern generation We performed *de novo* sequencing on the HUMAN dataset using PEAKS 6.0 (Ma *et al.*, 2003) and reported 10 peptides with the highest scores for each spectrum. Each peptide was converted into a blocked pattern as follows. Consecutive amino acids were replaced by the sum of their masses if they all had a local confidence score $< c$ (a predefined threshold), whereas an amino acid with a local confidence score $\geq c$ was replaced by its single mass. The masses were multiplied by the scale factor 100 and rounded to integers.

3.3.3 Pattern length and confidence score thresholds We tested the filtration efficiency and speed of the BPM algorithm by varying the two parameters, i.e. the length of blocked patterns and the threshold c for confidence scores, on the HUMAN dataset. To obtain length l blocked patterns, we enumerated all length l substrings of the blocked patterns generated from the dataset. The filtration efficiency of the algorithm decreases and the speed of the algorithm improves with the increase of the length of blocked patterns (Fig. 3). As the patterns become longer, the average number of candidate peptides for each spectrum decreases (Table 1). Particularly, when the pattern length $\geq$7, most spectra contain only 0 or 1 candidate peptide. When the pattern length is 5 or 6, the BPM algorithm achieves a good balance between the speed and filtration efficiency. Moreover, when the pattern length is fixed, the filtration efficiency decreases as the parameter c increases, while the running time is almost the same (Fig. 3). Therefore, a relative low cutoff value c should be used in practice.

There are three parameters: the pattern length, the confidence score threshold and the number of blocked patterns, which affect the filtration efficiency of the BPM algorithm. When the confidence score threshold is fixed, the number of blocked patterns extracted from a spectrum decreases when the pattern length increases and more shorter blocked patterns can be extracted from a spectrum than longer ones. The high-filtration efficiency of short blocked patterns has two possible reasons: the large number of short patterns and the short pattern length. To find which is the main reason, we tested the filtration efficiency of the BPM algorithm by fixing the confidence score threshold and the number of patterns and varying the pattern length. The experiment results show that the main reason that short gapped tags have high-filtration efficiency is the short pattern length, not the large number of short patterns (see the Supplementary Material).

Table 1. The average number of candidate peptides reported by the filtration based on blocked patterns for the spectra in the HUMAN dataset

Pattern length	4	5	6	7	8	9	10
Number of candidate peptides	4.18	1.15	0.84	0.74	0.60	0.47	0.36

Note: The local confidence score cutoff c is fixed as 60.

3.3.4 Comparison with a baseline algorithm We compared the running time of the BPM algorithm with a trivial baseline algorithm. Given a blocked pattern $P = p_1, p_2, \ldots, p_m$ and a text $T = t_1, t_2, \ldots, t_n$, the baseline algorithm simply examines, for each position i, $1 \leq i \leq n$, if there is a prefix of $t_i, t_{i+1}, \ldots, t_n$ that matches P and outputs matched prefixes. The time complexity of the algorithm is $O(nL)$, where L is the length of the longest text string that matches P. By setting the pattern length as 4 and the confidence score threshold as 60, a total of 170 149 blocked patterns were extracted from the HUMAN dataset using the described method. Although the baseline algorithm took about 9 h to search all the patterns against the human proteome database, the BPM algorithm took only less than 1 min, which is a significant 540 times speedup.

3.3.5 Comparison with sequence tags For each spectrum in the HUMAN dataset, we generated both ordinary patterns (sequence tags) and blocked patterns from the 10 peptides reported by PEAKS. When ordinary patterns were generated, the confidence score threshold was set as 0, and all amino acids in a peptide were replaced by their single masses. Note that the length of a blocked pattern may not be the same to its corresponding sequence tag. For example, the length of the sequence tag ALNQ (71, 113, 114, 128) is 4, and the length of the blocked pattern 71, 227, 128 is 3. To compare blocked patterns with ordinary patterns, we redefine the length of a blocked pattern as the length of the sequence tag (i.e. number of amino acids) from which the blocked pattern is generated.

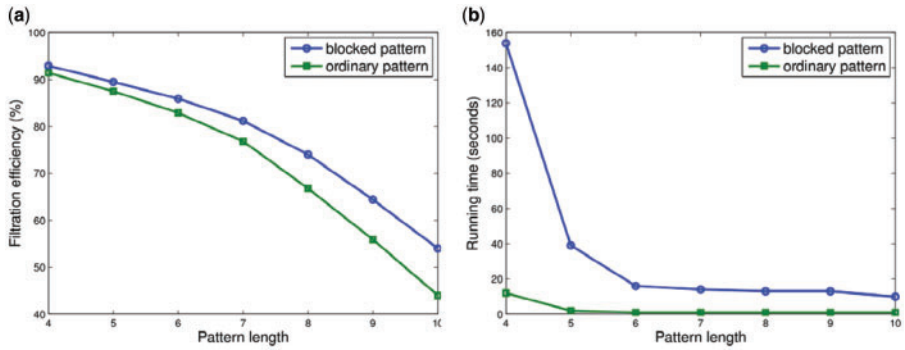

Fig. 4. Comparisons between blocked patterns and ordinary patterns: (a) the filtration efficiency and (b) the running time

Both the blocked patterns and ordinary patterns were searched against the suffix tree. The parameter c was fixed as 60 for generating blocked patterns. The BPM algorithm with blocked patterns achieved a higher filtration efficiency than that with ordinary patterns (Fig. 4a). The difference in the filtration efficiency increases as the pattern length increases (Fig. 4a). By contrast, the running time of the BPM algorithm with blocked patterns is longer than that with ordinary patterns (Fig. 4b), especially when pattern length is 4. However, the difference in the running time becomes smaller when the pattern length increases. Particularly, when the pattern length ≥ 6, the running time is within 20 s for both methods. The experiments demonstrated that replacing sequence tags with blocked patterns improved the filtration efficiency without significant increase of the running time.

3.3.6 Comparison with MASCOT To further investigate the filtration efficiency of the BPM algorithm, we ran MASCOT (Perkins *et al.*, 1999) on the whole human cell lysate dataset with 37 810 tandem mass spectra. The error tolerances for precursor masses and fragment ion masses were set as 20 ppm and 0.05 Da, respectively. Using the target-decoy approach, MASCOT identified 10 953 spectra and 9284 distinct peptides with 1% spectrum level FDR in 6.1 min. Without any filtering algorithm, MS-GF+ identified 13 135 spectra and 10 800 peptides with the same FDR. It identified 2141 peptides missed by MASCOT and missed 625 peptide identified by MASCOT. When the BPM algorithm (pattern length = 4, confidence score threshold = 60) was used as a filtering step for MS-GF+, a total of 982 peptides were missed because of the filtration, and MS-GF+ identified 11 845 spectra and 9818 peptides. MS-GF+ with the BPM filtering algorithm identified 1456 peptides missed by MASCOT and missed 922 peptides identified by MASCOT. When coupled with the BPM algorithm, the sensitivity of MS-GF+ is still comparable to MASCOT because of the high-filtration efficiency of the BPM algorithm.

3.3.7 Searching spectra against the human genome Most existing software tools for peptide identification by database search are slow when the protein database is huge. We evaluated the speed of the BPM algorithm by searching the HUMAN dataset against the six-frame translation of the human genome. This approach has found many important applications in proteogenomics studies (Andersen and Mann, 2001; Bitton *et al.*, 2010; Yates *et al.*, 1995).

The complete human genome (Homo_sapiens.GRCh37.74) was downloaded from Ensembl. The translation of each chromosome starts from the first, second and third nucleotide on each strand according to the standard genetic code regardless of the position of the start codon. The HUMAN dataset was searched against the translated putative protein sequences using both MS-GF+ and the BPM algorithm. The translated protein database was split into parts of about 100 MB to reduce the memory usage of MS-GF+ and the BPM algorithm. The largest memory usage of the BPM algorithm for searching a 100 MB database was about 12 GB. The average running time for MS-GF+ was 3 s per spectrum while it was 0.3 s for the BPM algorithm when the pattern length was 4 (Supplementary Fig. S7). The running time of the BPM algorithm decreased with the increase of the pattern length. When the pattern length was 6, the running time was about 0.08 s per spectrum.

4 CONCLUSIONS

Peptide identification of MS/MS spectra by database search has been dominated by the approach of comparing a spectrum against all peptides in the protein database whose molecular masses are similar to the precursor mass of the spectrum. However, in the era of big data biology, this approach may become slow with the rapid growth of the sizes of protein databases. An alternative approach based on sequence tags or blocked patterns (gapped tags) is fast for identifying peptides when the database is huge. In this article, we proposed an efficient BPM algorithm for filtering peptides using blocked patterns and tested it on simulated and real datasets. The experiments showed that the BPM algorithm achieved high speed in filtration of peptides, even when the database is huge, e.g. the six frame translation of the human genome. We also compared blocked patterns with sequence tags for peptide filtration and found that blocked patterns outperformed sequence tags in terms of the sensitivity of peptide filtration. Because the BPM algorithm is fast and sensitive, it is a promising tool for searching MS/MS spectra against huge protein databases. In practice, the length of blocked patterns plays an important role in the performance of the BPM algorithm. The experiments on the length of blocked

patterns showed that the BPM algorithm achieved a good balance between the filtration efficiency and the running time when the length was 5 or 6.

Funding: This work was supported by National Science Foundation of China (NSFC 61373048 to L.W. and F.D.), a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 122511 to L.W. and F.D), and a startup fund provided by Indiana University-Purdue University Indianapolis (to X.L.).

Conflict of interest: none declared.

REFERENCES

Andersen,J.S. and Mann,M. (2001) Mass spectrometry allows direct identification of proteins in large genomes. *Proteomics*, **1**, 641–650.

Bitton,D.A. *et al.* (2010) An integrated mass-spectrometry pipeline identifies novel protein coding-regions in the human genome. *PLoS One*, **5**, e8949.

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

Dewhirst,F.E. *et al.* (2010) The human oral microbiome. *J. Bacteriol.*, **192**, 5002–5017.

Eng,J.K. *et al.* (1994) An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *J. Am. Soc. Mass Spectrom.*, **5**, 976–989.

Frese,C.K. *et al.* (2011) Improved peptide identification by targeted fragmentation using CID, HCD and ETD on an LTQ-Orbitrap Velos. *J. Proteome Res.*, **10**, 2377–2388.

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

Jeong,K. *et al.* (2011) Gapped spectral dictionaries and their applications for database searches of tandem mass spectra. *Mol. Cell. Proteomics*, **10**, M110.002220.

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

Kim,S. *et al.* (2009) Spectral dictionaries: integrating de novo peptide sequencing with database search of tandem mass spectra. *Mol. Cell. Proteomics*, **8**, 53–69.

Liu,X. *et al.* (2013) Identification of ultramodified proteins using top-down tandem mass spectra. *J. Proteome Res.*, **12**, 5830–5838.

Ma,B. *et al.* (2003) PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry. *Rapid Commun. Mass Spectrom.*, **17**, 2337–2342.

Ma,Z.Q. *et al.* (2011) ScanRanker: quality assessment of tandem mass spectra via sequence tagging. *J. Proteome Res.*, **10**, 2896–2904.

Mann,M. and Wilm,M. (1994) Error-tolerant identification of peptides in sequence databases by peptide sequence tags. *Anal. Chem.*, **66**, 4390–4399.

Ng,J. *et al.* (2011) Blocked pattern matching problem and its applications in proteomics. In: *Proceedings of 15th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2011)*. Vancouver, BC, Canada: Springer-Verlag Berlin Heidelberg. pp. 298–319.

Perkins,D.N. *et al.* (1999) Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, **20**, 3551–3567.

Rudney,J. *et al.* (2010) A metaproteomic analysis of the human salivary microbiota by three-dimensional peptide fractionation and tandem mass spectrometry. *Mol. Oral Microbiol.*, **25**, 38–49.

Tabb,D.L. *et al.* (2003) GutenTag: high-throughput sequence tagging via an empirically derived fragmentation model. *Anal. Chem.*, **75**, 6415–6421.

Tanner,S. *et al.* (2005) InsPecT: identification of posttranslationally modified peptides from tandem mass spectra. *Anal. Chem.*, **77**, 4626–4639.

Ukkonen,E. (1995) On-line construction of suffix trees. *Algorithmica*, **14**, 249–260.

Yates,J.R. *et al.* (1995) Mining genomes: correlating tandem mass spectra of modified and unmodified peptides to sequences in nucleotide databases. *Anal. Chem.*, **67**, 3202–3210.