BIOINFORMATICS ORIGINAL PAPER

Vol. 29 no. 14 2013, pages 1768–1775
doi:10.1093/bioinformatics/btt274

Systems biology

Advance Access publication May 10, 2013

A combinatorial approach to the peptide feature matching problem for label-free quantification

Hao Lin, Lin He and Bin Ma*

David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Associate Editor: Martin Bishop

ABSTRACT

Motivation: Label-free quantification is an important approach to identify biomarkers, as it measures the quantity change of peptides across different biological samples. One of the fundamental steps for label-free quantification is to match the peptide features that are detected in two datasets to each other. Although *ad hoc* software tools exist for the feature matching, the definition of a combinatorial model for this problem is still not available.

Results: A combinatorial model is proposed in this article. Each peptide feature contains a mass value and a retention time value, which are used to calculate a matching weight between a pair of features. The feature matching is to find the maximum-weighted matching between the two sets of features, after applying a to-be-computed time alignment function to all the retention time values of one set of the features. This is similar to the maximum matching problem in a bipartite graph. But we show that the requirement of time alignment makes the problem NP-hard. Practical algorithms are also provided. Experiments on real data show that the algorithm compares favorably with other existing methods.

Contact: binma@uwaterloo.ca

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

Received on February 24, 2013; revised on May 7, 2013; accepted on May 8, 2013

1 INTRODUCTION

In proteomics, a quantification experiment compares two or more biological samples with discover peptides that have significant quantity changes across the samples (Heck and Krijgsveld, 2004). Such an experiment can potentially reveal important biomarkers that are relevant to the condition change across the samples (such as comparing a group of disease samples with a control group), and it is becoming an important tool for modern health research.

In the LC–MS (liquid chromatography–mass spectrometry) experiment for peptide quantification, the LC separates the peptides in the complex sample according to the peptides' hydrophobicity and elutes them at different *retention time*. Then the MS measures all the mass values (more precisely, the mass-to-charge ratio) of the co-eluting peptides and produces a mass spectrum at each scanned retention time. From the raw LC–MS data, many 'peptide features' are detected computationally (Zhang *et al.*, 2009). Each correctly detected feature corresponds to a peptide in the sample and mainly consists of three pieces of information: the mass, the retention time and the signal intensity. For the same peptide, the signal intensity is approximately proportional to the abundance of the peptide in the sample. Thus, if one can confidently match the two features of the same peptide in the two samples, the peptide's quantity change can be estimated from the intensity ratio.

Two major experimental approaches exist for peptide quantification: isotopic labeling and label-free (Timms and Cutillas, 2010). They differ mostly by how the peptide features from the two samples are matched. In the isotopic labeling approach, two samples are labeled with different isotopic reagents and mixed together before the LC–MS experiment. For most commercially available labeling reagents, the same peptides from the two samples appear at almost the same retention time and their matching becomes computationally simple. Although the label-free method does not label the samples, it measures the two samples in separate LC–MS runs. Because no isotopic labeling step is needed, the complexity of the experiment is greatly reduced. However, this also imposes a greater computational challenge for the peptide feature matching.

The main difficulty of the feature matching for the label-free method is rooted in the inadequate reproducibility of the LC retention time. Because of factors such as aging, packing and contamination of the LC column, together with additional variability during experiment such as temperature, gradient shape and mixing physics, the retention time from different runs often shows large shifts and distortions. To match peptide features by using their mass and retention time information, the shifts and distortions need to be corrected. This is usually carried out by finding a monotonically increasing function f that maps the time t of one sample to the time $f(t)$ of another. This process is often called the *retention time alignment*, or simply, *time alignment* (Cappadona *et al.*, 2012; Lange *et al.*, 2008; Vandenbogaert *et al.*, 2008).

Note that if the feature matching is available, the time alignment can be solved by fitting the times of the matched features with a smooth function. On the other hand, if the time alignment is known, the feature matching can be carried out by comparing the mass and the corrected time differences between features in the two samples. Although this is still not a trivial problem because of the existence of noise and close by features, its solution is not dauntingly difficult. The real challenge of the feature matching problem lies in the mutual dependence between the time alignment function and the feature matching.

In the literature, the time alignment and the feature matching are usually dealt with in two separate steps. Most researches in

*To whom correspondence should be addressed.

the literature used heuristic algorithms to find either an initial set of matched feature pairs, or an initial time alignment function. Then they are used to find a new time alignment function or a new set of matched feature pairs, respectively. Naturally, such a procedure can be repeated iteratively to, *hopefully*, get more and more accurate result.

This approach was typified by Li *et al.* (Li *et al.*, 2005), which matched features with similar m/z values as the initial feature matching. Kirchner *et al.* (2007) used the robust point matching method to find an initial feature matching, and then carried out smooth monotone regression to find time alignment. When there are significant time shifts and distortions, as well as the present of noisy false features, the finding of the initial set of feature matching in the aforementioned approaches can become challenging. Note that this problem can be solved if the peptides of all the features are known. Then the features can be matched confidently by checking their peptide identities. Such approaches have already been proposed in previous research (Fischer *et al.*, 2006; Tsou *et al.*, 2010). However, this requires additional MS/MS duty cycles of the instrument, which produce the MS/MS spectra for the identification of the peptides. This reduces the number of MS scans at the same time. As a result, many of the low-abundance peptides from the limited amount of biological samples may not produce strong enough signal in the LC–MS data and become undetectable. Therefore, if time alignment can be achieved without requiring MS/MS, it is advantageous to perform quantification without MS/MS. The peptides can be identified in a separate LC–MS/MS run after the quantification, possibly with an inclusion list that targets the quantified peptide features, and using a less precious sample. In fact, there are even proposals in the literature to identify peptides purely based on the m/z and the *aligned* retention time of a peptide feature (LaMarche *et al.*, 2013). This application definitely requires the time alignment without MS/MS. For these reasons, in this article, we assume the peptide identities are unknown to the alignment algorithm.

Other researchers focused on finding an initial time alignment function. Lange *et al.* (2007) assumed that the time alignment is a linear function: $f(t) = a \times t + b$. A pair of coefficients (a_i, b_i) was calculated from every two pairs of possibly matched features. The correct coefficients (a,b) were estimated by finding a dense cluster of all the calculated (a_i, b_i). Noticing the time alignment is usually non-linear, a number of publications (Bellew *et al.*, 2006; Bylund *et al.*, 2002; Jaitly *et al.*, 2006; Mueller *et al.*, 2007; Pluskal *et al.*, 2010; Podwojski *et al.*, 2009; Radulovic *et al.*, 2004; Silva *et al.*, 2005) only assumed local linearity of the time alignment, and applied linear regression on each local time window of the data. These works mostly differ at the methods used for local linear regression, and for connecting the local linear regression results into a global time alignment function.

Although many of these reviewed methods have been used in practice, none of them defined a clear optimization goal for the peptide feature matching problem. There were usually biological justifications for each step of these published methods. But the property of the final output of a method was unclear. The situation is different from the common practice in traditional algorithmic research, where the optimization goal is usually specified mathematically *before* the algorithm is being developed. This situation is not uncommon in many emerging bioinformatics areas (including peptide quantification) because the biologists often need to have a solution quickly once an experimental method is invented, in which case an *ad hoc* solution has its value. Moreover, given the complexity of biology, the formulation of a tidy mathematical model is often difficult.

But there are certainly disadvantages about this *ad hoc* type of research. An immediate one is that the final outcome of the algorithms is unpredictable without running the software on a specific input. As such, the performance of the algorithm remains at the mercy of the implementation details such as the choice of (many) parameters and sometimes even special considerations hard-coded in the software by a junior programmer to handle a special case of the training (and testing) data. Consequently, a method developed in one laboratory has the tendency to overfit the data of the laboratory and may not work as well in another laboratory or on a future instrument. We advocate that whenever possible, a combinatorial problem should be clearly defined. The separation of the problem formulation, the algorithm development and the program implementation can help reduce the aforementioned overfitting tendency. The field knowledge of the biologists should be used extensively and (almost) exclusively during the problem formulation stage to specify the desired property of the solution. And the algorithm development should strive to compute a solution that meets the specified property, instead of fitting the data that happen to be in the researcher's hands.

The first purpose of this article is to provide a clearly defined combinatorial model for the feature matching problem in Section 2. We will then prove that the feature matching problem is NP-hard in Section 3. In Section 4, a slightly modified optimization goal is proposed, under which a polynomial time algorithm is presented. We show that the solution of the modified problem helps determine an upper-bound and a lower-bound of the optimal solution of the feature matching problem. This results in a practical algorithm for the feature matching problem with a performance guarantee for each given instance. In Section 5, the optimization goal is amended to control the smoothness of the time alignment function for feature matching. A polynomial time algorithm is also presented. Finally, Section 6 examined the performance of the algorithm on real LC–MS data. Not only is the proposed model tidy but the algorithm's performance also compares favorably with other existing methods.

2 THE MAXIMUM FEATURE MATCHING PROBLEM

In this section, we formulate peptide feature matching as a combinatorial optimization problem. A *peptide feature* p is a 2-tuple $(m(p), t(p))$, where $m(p)$ indicates the mass and the $t(p)$ indicates the retention time of the feature in an LC–MS experiment. We assume both $m(p)$ and $t(p)$ are integers, as real numbers can be discretized by allowing a small rounding error. A *sample* consists of a set of features $\{p_1, p_2, \ldots, p_n\}$. Let S and S' be two samples and their retention time range from 1 to T. A *time alignment function* that maps the time of S to the time of S' is a monotonically increasing function $f: [1, T] \mapsto [1, T]$ such that $f(1) = 1$ and $f(T) = T$.

As aforementioned, the retention time of a peptide cannot be measured accurately. First, the unavoidable variations of LC conditions in the two runs can cause systematic drifts of the retention time for all peptides. This systematic error is modeled by the time alignment function f. Second, the retention time of an individual peptide may change independently from other peptides, causing a random error. Suppose two features $p \in S$ and $p' \in S'$ are from the same peptide in the two samples S and S'. Then $|t(p') - f(t(p))|$ models the random error. After a proper time alignment, the random error is usually small. For example, if two LC runs are conducted on the same LC instrument under the same experimental condition, and each lasts for 1 h, then the random error is often <1 min after the time alignment.

For every two features, $p \in S$ and $p' \in S'$. The matching quality of p and p' is a non-negative function $w(\delta_m, \delta_t)$, where $\delta_m = |m(p') - m(p)|$ and $\delta_t = |t(p') - f(t(p))|$. The function w is also called a *weight* function. One example of the weight function is the *unit weight function*, denoted by w_I. Let $\Delta_m \geq 0$ and $\Delta_t \geq 0$ be two thresholds on the allowed mass and time errors, respectively. The unit weight function is

$$w_I(\delta_m, \delta_t) = \begin{cases} 1, & \text{if } \delta_m \leq \Delta_m \text{ and } \delta_t \leq \Delta_t, \\ 0, & \text{otherwise.} \end{cases}$$

The unit weight function essentially treats a pair of features as a match if and only if their mass and time differences are within the allowed error tolerances.

A *peptide feature matching*, or simply, a *feature matching*, is a bijective mapping between two subsets $P \subset S$ and $P' \subset S'$. More specifically, a feature matching provides a set of feature pairs, $M \subset \{(p, p') \mid p \in S, p' \in S'\}$, such that each feature appears in at most one pair in M. Given a time alignment function f and a weight function w, the total weight of the matching M, is defined as

$$w(M) = \sum_{(p,p') \in M} w(|m(p') - m(p)|, |t(p') - f(t(p))|).$$

For label-free quantification, the two studied samples share most of their peptides, and the biological experiments are optimized to minimize the noise and the mass and retention time errors. When the peptide identities for the peptide features are unknown, the most natural combinatorial goal for peptide feature matching is to maximize the total weight of the matching.

The *maximum feature matching problem* (MFM) is, therefore, defined as follows: given two samples S and S' and a weight function w, find a time alignment function f and a feature matching M, such that $w(M)$ is maximized.

Note that if f is given, MFM can be easily reduced to the maximum matching problem in a bipartite graph. In the reduction, each feature corresponds to a vertex, and the two sets S and S' are the two vertex sets of the graph. The edge weight between each pair of features is defined by the weight function w. In particular, when w is the unit weight function, the reduction results in the unweighted version of the maximum matching problem. It is well known that polynomial time algorithms exist for maximum matching, for both weighted and unweighted versions (Fredman and Tarjan, 1987; Mucha and Sankowski, 2004). However, for MFM, the time alignment function f needs to be computed simultaneously with the feature matching. This makes MFM a much harder problem.

3 MAXIMUM FEATURE MATCHING IS NP-HARD

THEOREM 3.1. The maximum feature matching problem is NP-hard under the unit weight function.

PROOF. The proof can be found in Supplementary Appendix.

4 A PRACTICAL ALGORITHM FOR MAXIMUM FEATURE MATCHING

In this section, we develop a practical algorithm for MFM. This is achieved by studying a slightly modified version of MFM. Instead of requiring the matching to be a bijective mapping, the new problem only requires the matching to be a surjective mapping. More specifically, a surjective matching M^* is a subset of $\{(p, p') \mid p \in S, p' \in S'\}$, such that a feature $p \in S$ appears at most once in M^*. However, it is possible that a feature $p' \in S'$ appears multiple times. Given a time alignment function f and a weight function w, the weight of the surjective matching M^* can be defined in the same way as in the MFM problem:

$$w(M^*) = \sum_{(p,p') \in M^*} w(|m(p') - m(p)|, |t(p') - f(t(p))|).$$

Given two samples and a weight function w, the *maximum surjective feature matching problem* (SFM) computes a time alignment function and a surjective matching M^*, such that $w(M^*)$ is maximized. We next present a polynomial time algorithm to the SFM problem.

For a sample S and a time i, let $S_i = \{p \in S | t(p) = i\}$ be the subset of features at time i. Let $S_{\leq i} = \{p \in S | t(p) \leq i\}$ be the subset of features with time at most i.

Let $d_{i,j}$ be the maximum weight of a surjective matching between S_i and S' that can be achieved by a time alignment function satisfying $f(i) = j$. As the time of all features in S_i is equal to i and $f(i)$ equals to j, $d_{i,j}$ can be easily computed by finding the best matching of each $p \in S_i$ separately.

Let $D_{i,j}$ be the maximum weight of a surjective matching between $S_{\leq i}$ and S' that can be achieved by a time alignment function satisfying $f(i) \leq j$. If $f(i) < j$, then clearly $D_{i,j} = D_{i,j-1}$. If $f(i) = j$, then the maximum surjective matching includes the maximum surjective matching from $S_{\leq i-1}$ to S', and the maximum surjective matching from S_i to S'. Therefore, $D_{i,j} = D_{i-1,j} + d_{i,j}$. Combining the two cases, we know that $D_{i,j} = \max\{D_{i,j-1},\ D_{i-1,j} + d_{i,j}\}$. With this recurrence relation, it is clear that the SFM problem can be computed with dynamic programming. The algorithm is outlined as follows. The optimal time alignment function f, as well as the surjective matching, can be computed by a standard backtracking.

Algorithm DP-SFM

```
1. For every $1 \leq i \leq T$ and $1 \leq j \leq T$
2.     Compute $d_{i,j}$
3. Let $D_{i,0} = 0$ and $D_{0,i} = 0$ for every $0 \leq i \leq T$.
4. For $i$ from 1 to $T$
5.     For $j$ from 1 to $T$
6.         Let $D_{i,j} = \max\{D_{i,j-1}, D_{i-1,j} + d_{i,j}\}$
7.     Output $D_{T,T}$ as the maximum weight of the surjective matching.
```

After tracing back, all (i,j) pairs on the optimal path form the optimal time alignment function f.

THEOREM 4.1. The SFM can be solved in $O(T^2 + T \times |S| \times |S'|)$ time by algorithm DP-SFM.

PROOF. The correctness of the algorithm follows from the discussion earlier in the text. We only need to prove the time complexity. The computation of each $d_{i,j}$ in line 2 takes at most $O(|S_i| \times |S'|)$ time. Therefore, the whole for loop at lines 1 and 2 takes time $O\left(\sum_{1 \le i,j \le T} |S_i| \times |S'|\right) = O(T \times |S| \times |S'|)$. After $d_{i,j}$ is computed and stored in memory, each execution of line 6 takes constant time. Thus, the for loops from line 4 to line 6 take $O(|T|^2)$ time. ■

Note that the most expensive part of the time complexity in Theorem 4.1 is the time $O(T \times |S| \times |S'|)$ for the computation of all $d_{i,j}$. When the weight function w satisfies some properties, it is possible to speed-up the computation of $d_{i,j}$.

COROLLARY 4.2. When the unit weight function $\mathrm{w_I}$ is used, SFM can be solved in time

$$O\left(T^2 + T \times |S| + |S| \times |S'|\right).$$

PROOF. We only need to show that $d_{i,j}$ can be computed with time $O(T \times |S| + |S| \times |S'|)$ for all $1 \le i \le T$ and $1 \le j \le T$. For each $p \in S$, let $\mathcal{J}_p = \{j |$ there is $p' \in S'$ such that $|m(p') - m(p)| \le \Delta_m$ and $|t(p') - j| \le \Delta_t\}$. Then $\mathcal{J}_p$ can be computed by the following algorithm:

Algorithm JP

```
1. Let $\mathcal{J}_p = \emptyset$.
2. For each $p' \in S'$ such that $|m(p') - m(p)| \le \Delta_m$
3.     $\mathcal{J}_p = \mathcal{J}_p \cup [t(p') - \Delta_t, t(p') + \Delta_t]$
```

Note that $\mathcal{J}_p$ is the union of many intervals with equal length. Therefore, there is a simple data structure to store $\mathcal{J}_p$ so that the union operation in line 3 can be done in $O(1)$ time. The detail of this data structure is provided in the Supplementary Appendix. Thus, the complexity of the above algorithm is $O(|S'|)$. After $\mathcal{J}_p$ is calculated, $d_{i,j}$ can be calculated by $d_{i,j} = |\{p \in S_i | j \in \mathcal{J}_p\}|$. And the calculation for all $1 \le i \le T$ and $1 \le j \le T$ can be carried out more efficiently with the following algorithm:

Algorithm DIJ

```
1. For each $1 \le i \le T$
2.     Calculate $\mathcal{J}_p$ for each $p \in S_i$ with algorithm JP.
3.     Let $d_{i,j} = 0$ for each $1 \le j \le T$.
4.     For each $p \in S_i$
5.         For each $j \in \mathcal{J}_p$
6.             Let $d_{i,j} = d_{i,j} + 1$.
```

As algorithm JP takes $O(|S'|)$ time for each $p \in S$, the accumulated time cost for line 2 is $O(|S| \times |S'|)$. As $|\mathcal{J}_p| \le T$, line 6 is repeated at most $O(\sum_{i=1}^{T} |S_i| \times T) = O(|S| \times T)$ times. Therefore, the total time complexity for algorithm DIJ is $O(T \times |S| + |S| \times |S'|)$. ■

Remark: If we sort S' by mass values, then in line 2 of algorithm JP, we can retrieve all p' such that $|m(p') - m(p)| \le \Delta_m$ by a binary search, without enumerating all $p' \in S'$. Because usually $\Delta_m \ll R$, and $|S'| > T$, this trick provides significant speed gain on real life instances.

In the rest of this section, we examine the relation between the SFM and the MFM problems.

LEMMA 4.3. Suppose two instances of the SFM and MFM share the same input. Then the weight of the maximum feature matching (MFM) is less than or equal to the weight of the maximum SFM.

PROOF. A bijective mapping is also surjective. Thus, a solution of MFM is also a solution of SFM. ■

On the other hand, let $M^* \subset \{(p,p') \,|\, p \in S, p' \in S'\}$ be a solution for SFM. We can easily modify M^* into a suboptimal solution for MFM by selecting only one pair of features from M^* for every $p' \in S'$. In fact, if a better suboptimal solution is desired, the following algorithm can be used to compute a suboptimal solution of MFM based on the optimal solution of SFM.

Algorithm SMFM

```
1. Compute an optimal solution for SFM using the same input. Let $f$ be
   the optimal time alignment function. Let $W$ be the optimal weight.
2. Let $\tilde{w}(p,p') = w(|m(p') - m(p)|, |t(p') - f(t(p))|)$ for every $p \in S$ and
   $p' \in S'$.
3. Treat $\tilde{w}(p,p')$ as the edge weight in a complete bipartite graph $S \times S'$,
   and compute a maximum bipartite matching.
4. Output the maximum bipartite matching as the suboptimal solution of
   MFM, and $W$ as the upper bound of the optimal weight.
```

THEOREM 4.4. Algorithm SMFM computes a suboptimal solution for MFM, and an upper bound for the optimal weight.

PROOF. The theorem is an immediate consequence of Lemma 4.3.

As algorithm SMFM outputs both a suboptimal solution and an upper bound for the optimal weight, one can effectively assess the performance of the algorithm for each given instance.

5 VARIATIONS OF THE MAXIMUM FEATURE MATCHING PROBLEM

5.1 Weight function

The unit weight function w_I is conceptually simple, and the mass and time error thresholds Δ_m and Δ_t can be easily determined by the technician who operates the instrument according to experience. However, it is sometimes desirable to use a continuous weight function to give different weight to different time errors. It has been shown that in real data, the random retention time error after the time alignment satisfies a normal distribution (Felinger, 1998). Let $\epsilon_i = t(p_i') - f(t(p_i))$ be the random time errors of a pair of matched features (p_i, p_i') after the time alignment. Then $\Pr(\epsilon_i) = \frac{1}{\sigma\sqrt{2\pi}} \mathbf{e}^{-\frac{1}{2}\left(\frac{\epsilon_i}{\sigma}\right)^2}$. Assume the random error of different features is independent to each other, then the probability of all the errors in the matching is $\prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \mathbf{e}^{-\frac{1}{2}\left(\frac{\epsilon_i}{\sigma}\right)^2}$. By taking the logarithm, it is easy to see that maximizing the probability aforementioned is equivalent to maximizing $\sum_{i=1}^{n} -\epsilon_i^2$. As the weight function needs to be non-negative, we define the following weight function w_2:

$$w_2(\delta_m, \delta_t) = \begin{cases} \Delta_t^2 - \delta_t^2, & \text{if } \delta_m \le \Delta_m \text{ and } \delta_t \le \Delta_t, \\ 0, & \text{otherwise.} \end{cases}$$

5.2 Gap penalty

In the definition of the MFM and SFM problems, not much restriction is put on the time alignment function *f*, except that it has to be monotonically increasing. However, it is sometimes beneficial to have some smoothness requirement for *f*. This is mostly for a practical concern: a smooth function needs fewer data points to fit than an arbitrary function does.

Let $[l_i, r_i]$ $(i = 1, \dots, k)$ be the maximal time intervals such that $r_i - l_i > 1$ and $f(t)$ remain constant in each interval. These are called the type-I gaps. The gap length for $[l_i, r_i]$ is $r_i - l_i$. Let $[l'_i, r'_i]$ $(i = 1, \dots, k')$ be the maximal time intervals such that there is no t such that $f(t) \in [l'_i, r'_i]$. These are called the type-II gaps. The gap length for $[l'_i, r'_i]$ is $l'_i - r'_i + 1$. By requiring *f* to be smooth, we essentially want to penalize these two types of gaps with a gap penalty function $g(k) > 0$ for a length-k gap.

This is analogous to the gaps in the pairwise sequence alignment. But a key difference is that here we prefer many smaller gaps over a few large gaps, which is the opposite of the sequence alignment. As a result, although the sequence alignment normally used a concave gap penalty function, here we choose to use a convex gap penalty function, such as $g(k) = k^2$. And the total gap penalty of the time alignment function *f* is defined as

$$g(f) = \sum_{i=1}^{k} g(r_i - l_i) + \sum_{i=1}^{k'} g(r'_i - l'_i + 1).$$

The *gapped-MFM* problem is to find a bijective feature matching M and a time alignment f to maximize score $(M, f) = w(M) - g(f)$. Similarly, the *gapped-SFM* problem is to find a surjective feature matching M^* and a time alignment f to maximize score $(M^*, f) = w(M^*) - g(f)$.

We design a dynamic programming algorithm for the gapped-SFM problem. Let $K > 0$ be the maximum allowed gap length. Let S_i, $S_{\le i}$ and $d_{i,j}$ be as defined in Section 4. Let $N_{i,j}$ be the maximum score achieved by features in $S_{\le i}$ and a time alignment function satisfying $f(i) = j$ and $f(i-1) < j$. Let $M_{i,j}$ be the maximum score achieved by features in $S_{\le i}$ and a time alignment function satisfying $f(i) = j$.

From the definition of $N_{i,j}$, we know that $[f(i-1)+1, f(i)-1]$ is a probable type-II gap. Let $k = f(i) - f(i-1) - 1$ be the gap length. Then, $f(i-1) = f(i) - k - 1 = j - k - 1$; therefore,

$$N_{i,j} = \max_{0 \le k \le K} \{M_{i-1, j-k-1} + d_{i,j} - g(k)\} \tag{1}$$

To compute $M_{i,j}$, assume that $i - k$ is the least number such that $f(i-k) = j$. Then $[i-k, i]$ is a probable type-I gap; therefore,

$$M_{i,j} = \max_{0 \le k \le K} \{N_{i-k,j} + \sum_{l=i-k+1}^{i} d_{l,j} - g(k)\} \tag{2}$$

From Equations (1) and (2), it is straightforward to develop a dynamic programming algorithm to compute $N_{i,j}$ and $M_{i,j}$ simultaneously. The time complexity will be $O(T^2K)$ plus the time needed by computing $d_{i,j}$. Therefore, for a general weight function w, the time complexity is $O(T^2K + T \times |S| \times |S'|)$. Here, the algorithm and proof details are omitted.

As MFM is NP-hard, gapped-MFM with a general gap penalty is also NP-hard. Algorithm SMFM in Section 4 can be modified to provide a suboptimal solution for gapped-MFM and an upper bound to the optimal score. The only required modification is to replace SFM by gapped-SFM in line 1 of the algorithm.

6 EXPERIMENTAL RESULTS

The performance of our algorithms was compared with three other state-of-the-art software tools, msInspect (Bellew *et al.*, 2006), MZmine2 (Pluskal *et al.*, 2010) and MultiAlign (LaMarche *et al.*, 2013) by using real LC–MS datasets. Our algorithms include (i) algorithm SMFM with the weight function w_2 and (ii) the algorithm with weight function w_2 and a gap penalty $g(k) = 10k^2$, as described in Section 5. For the rest of the section, the first algorithm will be denoted by *SMFM*, and the second algorithm will be denoted by *SMFM-g*.

Five LC–MS datasets produced from the yeast proteome by three different laboratories were chosen for our comparison purpose. These were all public datasets made available by previous publications (Askenazi *et al.*, 2011; Nagaraj *et al.*, 2012; Swaney *et al.*, 2008). The names of the datasets and the number of features detected by msInspect in each of them are listed in Table 1. These five datasets are aligned with one another under different settings. More specifically, the alignments **Coon1.F3 versus Coon2.F4** and **Mann.1 versus Mann.2** are datasets from the same laboratory on the same instrument in the same experiment. These reflect the easiest test cases, as the LC conditions do not vary too much. The alignments **iPRG versus Coon2.F4** and **Coon2.F4 versus Mann.1** reflect the most challenging test cases, as the aligned datasets were from different laboratories and the LC conditions across different laboratories present the largest possible variations. However, as they were all produced from the yeast proteome, there should be a significant number of peptides shared by the datasets. Therefore, a robust feature matching algorithm should still be able to match these common peptides' features, despite the existence of large retention time distortion and noises. More detailed information about the datasets and the justification of selecting these particular four pairs of datasets for the alignments can be found in Supplementary Appendix.

For each dataset, the MS/MS spectra were used to identify peptides with the PEAKS 6 software (Zhang *et al.*, 2012). Parameters of the database search can be found in Supplementary Appendix. The peptides identified with false discovery rate $\le 1\%$ that matched only one feature in the LC–MS data were selected as the control set. The purpose of this control

Table 1. The number of features in different samples

Dataset	iPRG	Coon1.F3	Coon2.F4	Mann.1	Mann.2
Features	11 430	5879	5320	66 479	68 128

set of peptides was to find a subset of 'true' peptide feature matches between different datasets, which were used to evaluate different software's performance. Note that the peptide identification result was not used by any software as input, for the purposes discussed in Section 1.

Each of the compared software tools, SMFM, SMFM-g, msInspect, MZmine2 and MultiAlign, was used to produce the pairwise time alignment for iPRG versus Coon2.F4, Coon2.F4 versus Mann.1, Coon1.F3 versus Coon2.F4 and Mann.1 versus Mann.2, respectively. The m/z and retention time error tolerance of each software were set to be the same whenever possible. More specifically, Δ_t was set to be 5 min for the samples from different laboratories (iPRG versus Coon2.F4 and Coon2.F4 versus Mann.1) and 2 min for the ones from the same laboratory (Coon1.F3 versus Coo2.F4 and Mann.1 versus Mann.2). Other unique parameters of a software tool were set separately to achieve its own best performance:

(1) SMFM: $\Delta_m = 15$ ppm (part-per-million).
(2) SMFM-g: $\Delta_m = 15$ ppm, gap penalty $g(k) = 10k^2$.
(3) msInspect: spline mode, mass error tolerance = 15 ppm.
(4) MZmine2: RANSAC algorithm mode, m/z tolerance = 10 ppm (error tolerance 15 ppm crashed the software), retention time tolerance (before correction) = 50 min, number of RANSAN iterations = auto, minimal number of points = 20%, threshold value = 3 and same charge state was required.
(5) MultiAlign: mass tolerance = 15 ppm, and hybrid recalibration was selected.

The peptide features detected by msInspect from the LC–MS raw data were exported as the input of SMFM, SMFM-g and msInspect. MultiAlign and MZmine2 do not accept features detected by msInspect. Therefore, MultiAlign used the features detected by DeconTools (Slysz *et al.*, 2010), which was the preferred feature detection method of MultiAlign. MZmine2 used its own feature detection result.

The performance of each method was measured quantitatively with the average aligned time error of the true feature pairs. More specifically, for each pair of features $p = (m(p), t(p))$ and $p' = (m(p'), t(p'))$ that were from the two compared samples and shared the same peptide, the aligned time error was calculated as $|f(t(p)) - t(p')|$, where $f(\cdot)$ was the retention time alignment function calculated by each software tool. The average aligned time error and the percentage of correctly aligned 'true' feature pairs of each software applying on each pair of datasets are provided in Table 2. A feature pair is considered as correctly aligned if their aligned retention time difference is below the specified threshold in each experiment.

Although the five aforementioned software tools did not use the peptide identification deliberately, just for curiosity, the average aligned time errors obtained by a simple method (Polynomial-4) that used the peptide identification were also added in Table 2. By using the true feature pairs derived from

Table 2. The comparison of average aligned time errors (in seconds) and the percentages of correctly aligned feature pairs on true peptide features

Experiment	SMFM	SMFM-g	ms-Inspect	MZ-mine2	Multi-Align	Polynomial-4
iPRG-Coon2.F4	36.6 (100%)	35.2 (100%)	114.8 (87%)	55.0 (99%)	126.0 (92%)	30.3 (100%)
Coon2.F4-Mann.1	63.9 (82%)	62.1 (82%)	97.1 (71%)	65.7 (80%)	78.7 (73%)	66.2 (79%)
Coon1.F3-Coon2.F4	8.4 (100%)	7.4 (100%)	11.3 (96%)	21.8 (94%)	27.8 (89%)	8.2 (100%)
Mann.1-Mann.2	14.5 (90%)	13.0 (87%)	—	—	16.4 (81%)	15.7 (86%)

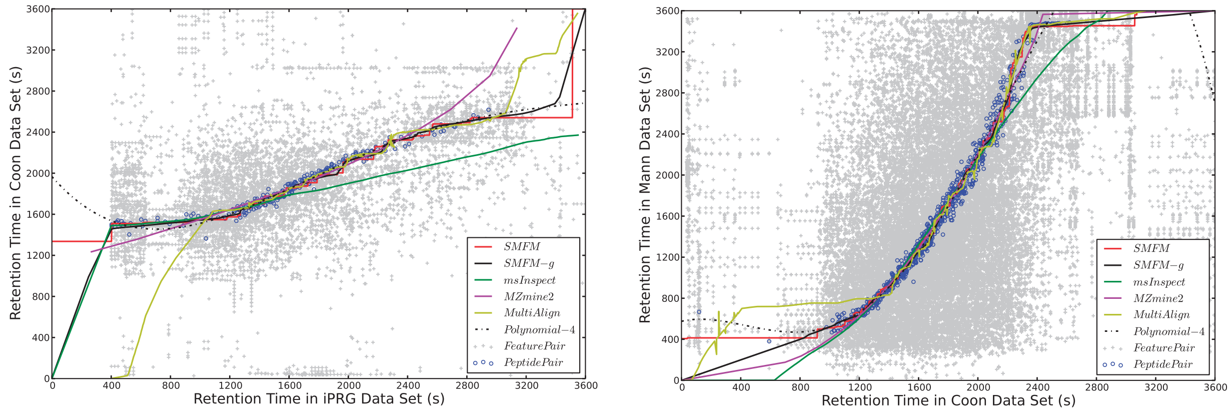

Fig. 1. Comparison of the feature matching software tools on datasets from different laboratories: iPRG versus Coon2.F4 (left) and Coon1.F3 versus Mann.1 (right). The x-axis denotes the retention time in the first sample, and the y-axis denotes the retention time in the second sample. A blue circle stands for a feature pair that is matched according to peptide identification, which is considered as the ground truth. A gray cross represents a possible feature pair matched purely by the precursor mass. The curves are produced by the compared algorithms without knowing the blue circles

the peptide identification, the Polynomial-4 method fitted a fourth degree polynomial as the time alignment function (the second and third degree were also tried, but the results were not as good as the fourth degree). Note that this was an unfair comparison because Polynomial-4 used additional information. Nevertheless, Table 2 showed that our new methods SMFM and SMFM-g also compared favorably with this polynomial fitting. This indicated that the time alignment function could not be fit accurately by a low-degree polynomial, and further justified the use of a monotonically increasing function instead of any specific simple function in our SMFM model. For the alignment of Mann.1 versus Mann.2, both msInspect and MZmine2 failed (msInspect crashed and MZmine2 returned no result). We suspected that it was due to the large data size of Mann's datasets (Table 1). Our new algorithms (SMFM and SMFM-g) finished successfully in <1 min with 560 MB of memory usage.

Figure 1 illustrates the relative performance of the six compared methods visually. The resulting time alignment from each software was plotted together with the 'true' peptide feature pairs (represented by blue circles). Retention time of both samples were scaled to 3600 s in the figure. All the possible feature pairs that had a mass difference <15 ppm were also plotted as gray crosses. Thus, intuitively, the software tools were using these gray crosses to compute a time alignment function. The better software's result should fit the trend of blue circles. Similar figures for the alignments between biological replicates, Coon1.F3 versus Coon2.F4 and Mann.1 versus Mann.2, were plotted in Supplementary Figure S5.

7 CONCLUSION AND DISCUSSION

The maximum feature matching problem (MFM) is formulated to match the peptide features in label-free peptide quantification. To our knowledge, this is the first combinatorial model for the problem. We show that the problem is NP-hard, and we provide a practical algorithm that has a performance guarantee for each instance. Experiments on real data demonstrate that the algorithm has a better performance comparing with other software in the literature.

Although recognizing the contribution and the need of the *ad hoc* software in bioinformatics research, we advocate that, whenever possible, the bioinformatics problem should have a clear combinatorial definition. This old practice in algorithmic research can help reduce the risk of overfitting the training data in the process of seeking for a better algorithm. It also helps predict the performance of an algorithm before running the software associated with it. As experienced by many, running a published but undocumented and un-maintained software package can be challenging. For example, during the preparation of this manuscript, 10 published software programs have been tried. However, only 4 of the 10 produced reasonable output. Among the four, the results of msInspect, MZmine2 and MultiAlign were included in the experimental result, whereas another one, OpenMS (Sturm *et al*., 2008), used a linear model that clearly could not fit our testing data and, therefore, was not included. Problems encountered by other six programs included crashing, out of memory and reporting error in the middle of the execution. As such, there is a non-negligible risk to compare algorithms with their software implementations. A buggy implementation can inadvertently affect the real performance of an algorithm.

Although a feature p is defined by a 2-tuple $(m(p), t(p))$ in this article, more information about a peptide feature retrieved from the LC–MS data can be added easily. One only needs to replace $m(p)$ with a vector that includes the other information, and in $w(\delta_m, \delta_t)$ replace δ_m by the difference of the two vectors of the two compared features. For example, the intensity distribution over the isotope peaks and over the retention time can be used to measure the similarity (or matching quality) of two matched features. With this adjustment, the NP-hardness and algorithmic results presented in this article remain the same.

In developing bioinformatics tools, researchers aim to find the 'real' biological solutions from the input data. However, as the real solution is unknown when the software is used in practice, the optimization goal defined in the bioinformatics problem cannot depend on the real solution, but it is at best an approximation to the *property* of the real solution. We have demonstrated that by clearly defining such optimization goal as a simple function of the input and the algorithm's output, not only does the biological problem become a pure combinatorial problem that algorithmic researchers can work on, the performance of the algorithm also compares favorably with the state-of-the-art *ad hoc* software packages.

In fact, clearly defining the optimization goal is helpful even in the *ad hoc* solutions. For example, many have proposed to alternatively find a time alignment and a set of matched features by using each other as the input. But there is no guarantee that such iteration will converge or can improve the result. However, if the optimization goal is a simple function of the output, one can then evaluate the current solution with the optimization goal at each step of the iteration, and it stops when the score stops growing.

ACKNOWLEDGEMENT

The authors thank Dr Zefeng Zhang for his useful discussion.

Funding: The authors were supported by an NSERC discovery (RGPIN 238748-2011) grant and a starter grant (to B.M.) at University of Waterloo.

Conflict of Interest: none declared.

REFERENCES

Askenazi,M. *et al*. (2011) iPRG 2011: a study on the identification of electron transfer dissociation (ETD) mass spectra. *J. Biomol. Tech*., **22** (**Suppl**), S20.

Bellew,M. *et al*. (2006) A suite of algorithms for the comprehensive analysis of complex protein mixtures using high-resolution LC-MS. *Bioinformatics*, **22**, 1902–1909.

Bylund,D. *et al*. (2002) Chromatographic alignment by warping and dynamic programming as a pre-processing tool for parafac modelling of liquid chromatography–mass spectrometry data. *J. Chromatogr. A*, **961**, 237–244.

Cappadona,S. *et al*. (2012) Current challenges in software solutions for mass spectrometry-based quantitative proteomics. *Amino Acids*, **43**, 1087–1108.

Felinger,A. (1998) *Data Analysis and Signal Processing in Chromatography*. Elsevier, San Diego, CA.

Fischer,B. *et al*. (2006) Semi-supervised LC/MS alignment for differential proteomics. *Bioinformatics*, **22**, e132–e140.

Fredman,M.L. and Tarjan,R.E. (1987) Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, **34**, 596–615.

Heck,A.J. and Krijgsveld,J. (2004) Mass spectrometry-based quantitative proteomics. *Expert Rev. Proteomics*, **1**, 317–326.

Jaitly,N. *et al.* (2006) Robust algorithm for alignment of liquid chromatography-mass spectrometry analyses in an accurate mass and time tag data analysis pipeline. *Anal. Chem.*, **78**, 7397–7409.

Kirchner,M. *et al.* (2007) amsrpm: robust point matching for retention time alignment of LC/MS data with R. *J. Stat. Softw.*, **18**, 1–12.

LaMarche,B.L. *et al.* (2013) Multialign: a multiple LC-MS analysis tool for targeted omics analysis. *BMC bioinformatics*, **14**, 49.

Lange,E. *et al.* (2007) A geometric approach for the alignment of liquid chromatography–mass spectrometry data. *Bioinformatics*, **23**, i273–i281.

Lange,E. *et al.* (2008) Critical assessment of alignment procedures for LC-MS proteomics and metabolomics measurements. *BMC bioinformatics*, **9**, 375.

Li,X.J. *et al.* (2005) A software suite for the generation and comparison of peptide arrays from sets of data collected by liquid chromatography-mass spectrometry. *Mol. Cell. Proteomics*, **4**, 1328–1340.

Mucha,M. and Sankowski,P. (2004) Maximum matchings via gaussian elimination. In: *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '04, pp. 248–255.

Mueller,L.N. *et al.* (2007) SuperHirn–a novel tool for high resolution LC-MS-based peptide/protein profiling. *Proteomics*, **7**, 3470–3480.

Nagaraj,N. *et al.* (2012) System-wide perturbation analysis with nearly complete coverage of the yeast proteome by single-shot ultra HPLC runs on a bench top orbitrap. *Mol. Cell. Proteomics*, **11**, M111.013722.

Pluskal,T. *et al.* (2010) MZmine 2: modular framework for processing, visualizing, and analyzing mass spectrometry-based molecular profile data. *BMC bioinformatics*, **11**, 395.

Podwojski,K. *et al.* (2009) Retention time alignment algorithms for LC/MS data must consider non-linear shifts. *Bioinformatics*, **25**, 758–764.

Radulovic,D. *et al.* (2004) Informatics platform for global proteomic profiling and biomarker discovery using liquid chromatography-tandem mass spectrometry. *Mol. Cell. Proteomics*, **3**, 984–997.

Silva,J.C. *et al.* (2005) Quantitative proteomic analysis by accurate mass retention time pairs. *Anal. Chem.*, **77**, 2187–2200.

Slysz,G.W. *et al.* (2010) The decontools framework: an application programming interface enabling flexibility in accurate mass and time tag workflows for proteomics and metabolomics. In: *Proceedings of 58th ASMS Conference on Mass Spectrometry and Allied Topics*. Salt Lake City.

Sturm,M. *et al.* (2008) OpenMS–an open-source software framework for mass spectrometry. *BMC bioinformatics*, **9**, 163.

Swaney,D.L. *et al.* (2008) Decision tree–driven tandem mass spectrometry for shotgun proteomics. *Nat. Methods*, **5**, 959–964.

Timms,J.F. and Cutillas,P.R. (2010) Overview of quantitative LC-MS techniques for proteomics and activitomics. *LC-MS/MS in Proteomics: Methods in Molecular Biology*, **658**, 19–45.

Tsou,C.C. *et al.* (2010) IDEAL-Q, an automated tool for label-free quantitation analysis using an efficient peptide alignment approach and spectral data validation. *Mol. Cell. Proteomics*, **9**, 131–144.

Vandenbogaert,M. *et al.* (2008) Alignment of LC-MS images, with applications to biomarker discovery and protein identification. *Proteomics*, **8**, 650–672.

Zhang,J. *et al.* (2009) Review of peak detection algorithms in liquid-chromatography-mass spectrometry. *Curr. Genomics*, **10**, 388.

Zhang,J. *et al.* (2012) PEAKS DB: de novo sequencing assisted database search for sensitive and accurate peptide identification. *Mol. Cell. Proteomics*, **11**, M111.01058.