Published in final edited form as:

IEEE Trans Nanobioscience. 2017 March ; 16(2): 123–130. doi:10.1109/TNB.2017.2666780.

Filling a Protein Scaffold with a Reference

Letu Qingge,

Gianforte School of Computing, Montana State University, Bozeman, MT 59717, USA

Xiaowen Liu,

School of Informatics and Computing, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202, USA

Farong Zhong, and

College of Math, Physics and Information Engineering, Zhejiang Normal University, Jinhua, Zhejiang 324001, China

Binhai Zhu

Gianforte School of Computing, Montana State University, Bozeman, MT 59717, USA

Abstract

In mass spectrometry-based de novo protein sequencing, it is hard to complete the sequence of the whole protein. Motivated by this we study the (one-sided) problem of filling a protein *scaffold* $\mathcal{S}$ with some missing amino acids, given a sequence of contigs none of which is allowed to be altered, with respect to a complete reference protein $\mathcal{P}$ of length n , such that the BLOSUM62 score between $\mathcal{P}$ and the filled sequence $\mathcal{S}'$ is maximized. We show that this problem is polynomial-time solvable in $O(n^{26})$ time. We also consider the case when the contigs are not of high quality and they are concatenated into an (incomplete) sequence $\mathcal{I}$, where the missing amino acids can be inserted anywhere in $\mathcal{I}$ to obtain $\mathcal{S}'$, such that the BLOSUM62 score between $\mathcal{P}$ and $\mathcal{S}'$ is maximized. We show that this problem is polynomial-time solvable in $O(n^{22})$ time. Due to the high time complexity, both of these algorithms are impractical, we hence present several algorithms based on greedy and local search, trying to solve the problems practically. The empirical results, based on some antibody and mammalian proteins, show that the algorithms can fill protein scaffolds with high quality, provided that a good pair of scaffold and reference are given.

Keywords

protein sequencing; protein scaffold filling; BLO-SUM62; greedy search; local search; algorithms

I. Introduction

In mass spectrometry-based de novo protein sequencing, it is hard to complete the sequence of the whole protein [2]. An incomplete sequence contains one or several contigs, each is a protein segment. When the order of the contigs is known (e.g., ordered contigs reported by

top-down mass spectrometry-based de novo protein sequencing [13]), these contigs are called a *scaffold*. In many applications, it is more desirable to obtain complete protein sequences. We comment that a similar phenomena occurs in the sequencing of genomes, firstly noticed by Muñoz et al. [16], and has resulted in a series of interesting algorithmic studies [8], [9], [10], [11], [3] (also, see [19] for a recent survey).

Hence, a natural combinatorial problem is to fill the missing amino acids into scaffolds. As one must find a biologically meaningful way of filling scaffolds, it makes sense to use a complete homologous protein sequence as a reference. Here we consider two kinds of scaffolds. One kind is lists of contigs which are computed with a good confidence and should not be altered. Throughout this paper, we use $\mathcal{S}$ to denote such a scaffold, composed of contigs $C_1, C_2, \dots, C_m$. The other kind is sequences, which usually arise when contigs are computed without a very high confidence — in which case it would be unrealistic not to alter the contigs. We then simply concatenate the contigs into $\mathcal{S}$, the incomplete sequence. We try to carry out this idea to fill a scaffold (resp. sequence) $\mathcal{S}$ (resp. $\mathcal{S}$) to have $\mathcal{S}'$ (resp. $\mathcal{S}'$), such that the similarity between $\mathcal{S}'$ (resp. $\mathcal{S}'$) and a given (complete) protein $\mathcal{P}$ is maximized. These problems are called *Contig-Preserving Protein Scaffold Filling* (CP-PSF for short) and *Protein Scaffold Filling* (PSF for short) respectively.

The BLOSUM62 matrix [7] is the most popular similarity measure to align a pair of protein sequences. (The matrix can be found in the appendix.) We will use BLOSUM62 to measure the accuracy of our scaffold filling algorithms. However, for empirical results it is not very indicative to interpret BLOSUM62 scores for measuring the similarity between sequences. Therefore, we will use the number of *matched pairs* as a similarity measure (this can be considered a rounded BLOSUM62 matrix: if the score between two amino acids in BLOSUM62 is less than 4, then set the score as 0; otherwise, set the score as 1). Of course, when a space (denoted as a $\star$ in this paper) is used, with BLOSUM62 we get a negative score; and this is not reflected in the number of matched pairs.

Firstly, we show that the PSF problem is solvable in $O(n^{22})$ time. As this method is impractical, we make use of the standard Needleman-Wunsch algorithm to align $\mathcal{S}$ with $\mathcal{P}$, we then design two heuristic algorithms, based on greedy and local search, to insert the amino acids in $X = \mathcal{P} - \mathcal{S}$ into $\mathcal{P}$. We test our algorithm using some real datasets (i.e., 4 chains from two antibody proteins and 2 chains from two mammalian proteins). Secondly, for the CP-PSF problem of filling $\mathcal{S}$, we also present a polynomial time solution which takes $O(n^{26})$ time. We then design a heuristic method based on the Needleman-Wunsch algorithm, and implement two versions using the greedy and local search methods. We test our algorithms on the same datasets.

We comment that our problems are related to but different from the *one-sided* scaffold filling problem for genomes (with gene repetitions), the main difference is that the similarity measure between genomes is different from that between protein sequences. For protein sequences, the order of its amino acids is probably even more critical compared with genomes. Given a complete genome $\mathcal{G}$ and a genomic scaffold $\mathcal{H}$, the one-sided scaffold filling problem, i.e., filling $\mathcal{H}$ into $\mathcal{H}'$ such that the number of adjacencies between $\mathcal{G}$ and

$\mathcal{H}'$ is maximized, is NP-hard [8], [9] and constant-factor approximation algorithms are known [10], [19].

This paper is organized as follows. In Section 2, we give necessary definitions. In Section 3, we present an $O(n^2)$ time solution and two practical algorithms for PSF, and we then present some empirical results using some real datasets. In Section 4, we first show that CP-PSF can be solved in $O(n^2)$ time, and then we design two practical algorithms for CP-PSF. We finally present some empirical results using the same datasets. In Section 5, we conclude the paper.

II. Preliminaries

We first present some necessary definitions.

We denote the set of 20 amino acids as $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$. A protein sequence $\mathcal{P}$ is a sequence over Σ . We also use $c(\mathcal{P})$ to denote the multiset of elements in $\mathcal{P}$. For example, $\mathcal{P} = \langle A, D, C, I, K, W, Y, C, I \rangle$ (or simply, $\mathcal{P} = \text{ADCIKWYCI}$) with $c(\mathcal{P}) = \{A, C, C, D, I, I, K, W, Y\}$. In bottom-up mass spectrometry based de novo protein sequencing, we first derive peptide sequences from tandem mass spectra of the target protein, then build up longer contigs by spectral or peptide assembly [1], [12]. For real datasets, the length of a peptide is typically between 5 and 40.

Given two protein sequences $\mathcal{P} = \langle p_1, p_2, \dots, p_n \rangle$ and $\mathcal{Q} = \langle q_1, q_2, \dots, q_m \rangle$, (p_i, q_j) forms a *matched pair* if $p_i = q_j$ i.e., p_i and q_j are the same amino acid. We use $\mathcal{P}[i]$ to represent p_i , $\mathcal{P}[i, j]$ to represent the substring $\langle p_i, \dots, p_j \rangle$ and $\mathcal{P}[i..]$ to represent the substring $\langle p_i, \dots, p_n \rangle$. The length of $\mathcal{P}$ is denoted as $|\mathcal{P}|$, which is n here. We use $m(\mathcal{P}, \mathcal{Q})$ to denote the corresponding number of matched pairs. Let T_1 and T_2 denote two sets of protein sequences of the same length, we use $m(T_1, T_2)$ to denote the maximum number of matched pairs $m(\mathcal{P}, \mathcal{Q})$, for any $\mathcal{P} \in T_1, \mathcal{Q} \in T_2$. This definition also holds when $\mathcal{P}$ and $\mathcal{Q}$ are aligned from initial protein sequences, possibly of different lengths, using the standard sequence alignment algorithm by Needleman and Wunsch [17], i.e., when $\mathcal{P}$ and $\mathcal{Q}$ contain $*$ (the gap, or space) letters. (For the ease of presentation we use $*$ instead of $-$, as the latter is used as subtraction as well.) Note that a $*$ does not form a matched pair with any amino acid.

BLOSUM62 (B62 for short), coming from BLOck SUBstitution Matrix [7], is based on comparisons of multiply (locally) aligned ungapped segments corresponding to the most highly conserved regions of proteins in the Blocks database [18]. The number of 62 means the alignment comparisons are based on first clustering ungapped sequences with 62% identity. BLOSUM62 is the default matrix for the standard protein-BLAST program. The integer values in BLOSUM62 vary from -4 to 11 , e.g., $B62[W, D] = -4$ and $B62[W, W] = 11$. It should be noted that a score -8 is applied for introducing a $*$, which is a restriction for using too many $*$'s. Throughout this paper, we use B62 to measure the similarity of protein sequences.

A *scaffold* $\mathcal{S}$ is a list of contigs $C_1, C_2, \dots, C_m$, where each contig is a sequence of amino acids. For example, $\mathcal{S} = \langle C_1, C_2, C_3 \rangle$, where $C_1 = \text{AEFGIA}$, $C_2 = \text{CDIKLNTVW}$, and

$C_3 = \text{PQAWYA}$. These contigs are usually computed with some weight constraint, say, the total weight of amino acids in C_1 is roughly $71+129+147+57+113+71=598$ Dalton. Hence, we should be more careful in inserting some missing amino acid into a contig. In fact, throughout this paper, we do not alter the amino acids in a given contig at all. Given a multiset X of amino acids, and a scaffold $\mathcal{S} = \langle C_1, \dots, C_m \rangle$, $\mathcal{S} + X$ is the set of all protein sequences obtained by inserting the amino acids in X in between C_i 's (i.e., C_i 's are not altered).

A *sequence scaffold* (or just sequence) $\mathcal{S}$ is an incomplete protein sequence, i.e., with some unknown missing amino acids. This sequence is obtained usually when the contigs are not of high quality, and we just concatenate C_i 's to obtain a sequence. We are allowed to insert missing amino acids anywhere in $\mathcal{S}$.

In practice, if the total mass of the target protein is known, which can be measured by top-down mass spectrometry, we would know the total mass of missing amino acids. In this case, one way for handling this is to enumerate the sets of amino acids which sum to a certain mass — corresponding to that of the missing amino acids. Of course, when this mass is decently large we could have an exponential number of such sets. In this paper, we only focus on a given set X of missing amino acids, which can be computed as the difference between the reference protein and given scaffold (or sequence). We do not consider the mass information for all the results in this paper — though we will discuss that at the end of the paper.

Given two protein sequences $\mathcal{P}, \mathcal{Q}$, we use $B62(\mathcal{P}, \mathcal{Q})$ to denote the maximum BLOSUM62 score when aligning them. Let $\mathcal{Z}$ be a set of protein sequences. Then $B62(\mathcal{P}, \mathcal{Z})$ is the maximum BLOSUM62 score when aligning $\mathcal{P}$ and any sequence $z \in \mathcal{Z}$. Given a protein sequence $\mathcal{S}$ and a multiset X of amino acids, we denote $\mathcal{S} + X$ as the set of all protein sequences obtained by filling all the amino acids in X into $\mathcal{S}$. We use $|X|$ to denote the size of the set X .

The contig-preserving protein scaffold filling (CP-PSF) problem is defined as follows.

Contig-Preserving Protein Scaffold Filling to Maximize the B62 Score (CP-PSF)

Input: a complete protein sequence $\mathcal{P}$, a protein scaffold $\mathcal{S} = \langle C_1, \dots, C_m \rangle$, and a multiset X of amino acids.

Question: maximize the B62 score $B62(\mathcal{P}, \mathcal{S} + X)$.

For most of the practical instances we could assume that X is given as $X = c(\mathcal{P}) - \cup_i c(C_i)$. Note that we have no restriction on the length of $\mathcal{P}$ and the filled sequences, as with B62 we could use $\star$'s.

When the scaffold is a sequence $\mathcal{S}$, the problem can be simplified as follows.

Protein Scaffold Filling to Maximize the B62 Score (PSF)

Input: a complete protein sequence $\mathcal{P}$ with $|\mathcal{P}|=n$, an incomplete protein sequences $\mathcal{I}$ and a multiset of amino acids X .

Question: maximize the B62 score $B62(\mathcal{P}, \mathcal{I} + X)$.

Note that, again, for most of the practical instances we could assume that X is given as $X = c(\mathcal{P}) - c(\mathcal{I})$.

As solutions for PSF could be used as subroutines for CP-PSF, in the next section we first show that the PSF problem is polynomially solvable by giving a complex dynamic programming solution which runs in $O(n^2)$ time. We then present two practical methods and show some empirical results.

III. Algorithms and Empirical Results for PSF

A. PSF is in P

We present a dynamic programming solution for solving the PSF problem, given $\mathcal{P}$, $\mathcal{I}$ and X , with $|\mathcal{P}|=n$ and $|\mathcal{I}|=m$. The objective is to insert X into $\mathcal{I}$ to obtain $\mathcal{I}'$ such that $B62(\mathcal{P}', \mathcal{I}')$ is maximized. Note that, to obtain the maximum score $B62(\mathcal{P}', \mathcal{I}')$, it is possible that some $*$'s are inserted into $\mathcal{P}$ to eventually have $\mathcal{P}'$.

The detailed idea of our algorithm is as follows. In an optimal solution, when scanning $\mathcal{P}$ from left to right, there are 5 cases: (1) $\mathcal{P}[i]$ is aligned to some amino acid $\mathcal{I}[j]$, (2) $\mathcal{P}[i]$ is aligned to an amino acid in X (inserted at $\mathcal{I}[j]$), (3) $\mathcal{P}[i]$ is aligned to a $*$ inserted at $\mathcal{I}[j]$, (4) $\mathcal{P}[i]$ is inserted with a $*$ and is aligned to some amino acid $\mathcal{I}[j]$, and (5) $\mathcal{P}[i]$ is inserted with a $*$ and is aligned to an amino acid in X (inserted at $\mathcal{I}[j]$). To denote the set X (or any of its subset), we use a 20-dimension vector $X = \langle a_1, a_2, \dots, a_{20} \rangle$, where a_k is the number of amino acid k ($k = 1..20$). Define $|X| = a_1 + a_2 + \dots + a_{20}$. It is clear that the number of distinct subsets of X is bounded by

$$\begin{aligned} & (a_1+1) \times (a_2+1) \times \dots \times (a_{20}+1) \\ & \leq \left(\frac{a_1+a_2+\dots+a_{20}+20}{20} \right)^{20} \\ & = \left(\frac{|X|+20}{20} \right)^{20} \\ & = O(n^{20}). \end{aligned}$$

Let t be a subset of X , represented as a 20-dimension vector. Define $\mathcal{S}[i, j, t]$ as the maximum B62 score when aligning $\mathcal{P}[1..i]$ with $\mathcal{I}[1..j]$ where $t \subseteq X$ has been inserted into $\mathcal{I}[1..j]$ (to achieve the score). Let X_k be one of the (non-empty) k -th amino acid in X , for $k = 1..20$. The recurrence relation for updating $\mathcal{S}[i, j, t]$ is as follows.

$$\mathcal{S}[i+1, j+1, t] = \max \begin{cases} \mathcal{S}[i, j, t] + B62(\mathcal{P}[i+1], \mathcal{J}[j+1]), // \mathcal{P}[i+1] \text{ matches } \mathcal{J}[j+1], \text{ case1} \\ \max_k \{ \mathcal{S}[i, j, t - X_k] + B62(\mathcal{P}[i+1], X_k) \}, // \mathcal{P}[i+1] \text{ matches } X_k \text{ (inserted at } \mathcal{J}[j+1]), \text{ case2} \\ \mathcal{S}[i+1, j, t] + B62(\mathcal{P}[i+1], \star), // \mathcal{P}[i+1] \text{ matches } \star \text{ (inserted at } \mathcal{J}[j+1]), \text{ case3} \\ \mathcal{S}[i, j+1, t] + B62(\star, \mathcal{J}[j+1]), // \star \text{ (inserted at } \mathcal{P}[i+1]) \text{ matches } \mathcal{J}[j+1], \text{ case4} \\ \mathcal{S}[i, j, t - X_k] + B62(\star, X_k), // \star \text{ (inserted at } \mathcal{P}[i+1]) \text{ matches } X_k \text{ (inserted at } \mathcal{J}[j+1]), \text{ case5} \end{cases}$$

The base cases, $\mathcal{S}[i, 0, t]$'s, can be computed by a greedy method in $O(n^2)$ time and space.

This is because we have $O(n^2)$ subset t 's of X . The optimal solution value is $\mathcal{S}[n, m, X]$.

Similar to the Needleman-Wunsch algorithm, we could reconstruct $\mathcal{P}'$ and $\mathcal{J}'$ given

$\mathcal{S}[i, j, t]$. The pseudo-code is given as follows.

Algorithm

Scaffold-Filling($\mathcal{P}, \mathcal{J}$) // $X = c(\mathcal{P}) - c(\mathcal{J})$

```

0   Compute  $\mathcal{S}[i, j, t]$ 's following the recurrence relation
1    $\mathcal{P}' \leftarrow ""$ 
2    $\mathcal{J}' \leftarrow ""$ 
3    $n \leftarrow |\mathcal{P}|$ 
4    $m \leftarrow |\mathcal{J}|$ 
5    $t \leftarrow X$ 
6   while ( $n > 0$  or  $m > 0$ )
7   {
8       if ( $n > 0$  and  $m > 0$ )
9           and  $\mathcal{S}[n, m, t] = \mathcal{S}[n-1, m-1, t] + B62(\mathcal{P}[n], \mathcal{J}[m])$ 
10          {
11               $\mathcal{P}' \leftarrow \mathcal{P}[n] + \mathcal{P}', \mathcal{J}' \leftarrow \mathcal{J}[m] + \mathcal{J}'$ 
12               $n \leftarrow n-1, m \leftarrow m-1$ 
13          }
14       else if ( $n > 0$  and  $m = 0$ )
15           and  $\mathcal{S}[n, m, t] = \mathcal{S}[n-1, m, t - X_k] + B62(\mathcal{P}[n], X_k)$ 
16          {
17               $\mathcal{P}' \leftarrow \mathcal{P}[n] + \mathcal{P}', \mathcal{J}' \leftarrow X_k + \mathcal{J}'$ 
18               $n \leftarrow n-1, t \leftarrow t - X_k$ 
19          }
20       else if ( $m > 0$  and  $n = 0$ )
21           and  $\mathcal{S}[n, m, t] = \mathcal{S}[n, m-1, t] + B62(\star, \mathcal{J}[m])$ 
22          {
23               $\mathcal{P}' \leftarrow \mathcal{P}', \mathcal{J}' \leftarrow \mathcal{J}[m] + \mathcal{J}'$ 
24               $m \leftarrow m-1$ 
25          }
26   }
```

```

14   else if ( $n > 0$  and  $\mathcal{S}[n, m, t] = \mathcal{S}[n - 1, m, t] + B62(\mathcal{P}[n], \star)$ )
15   {
16        $\mathcal{P}' \leftarrow \mathcal{P}[n] + \mathcal{P}', \mathcal{J}' \leftarrow \star + \mathcal{J}'$ 
17        $n \leftarrow n - 1$ 
18   }
19   else if ( $m > 0$  and  $\mathcal{S}[n, m, t] = \mathcal{S}[n, m - 1, t] + B62(\star, \mathcal{J}[m])$ )
20   {
21        $\mathcal{P}' \leftarrow \star + \mathcal{P}', \mathcal{J}' \leftarrow \mathcal{J}[m] + \mathcal{J}'$ 
22        $m \leftarrow m - 1$ 
23   }
24   else if ( $m > 0$  and  $\mathcal{S}[n, m, t] = \mathcal{S}[n, m - 1, t - X_k] + B62(\star, X_k)$ )
25   {
26        $\mathcal{P}' \leftarrow \star + \mathcal{P}', \mathcal{J}' \leftarrow X_k + \mathcal{J}'$ 
27        $m \leftarrow m - 1$ 
28        $t \leftarrow t - X_k$ 
29   }
30   }

```

As the number of subsets of X is bounded by $O(n^{20})$, it is obvious that the algorithm takes $O(n^{22})$ time and space. We thus have the following theorem.

Theorem 1—PSF can be solved in $O(n^{22})$ time and space.

Even though PSF is polynomially solvable, the high running time makes the solution practically infeasible to implement. We next present two practical methods.

B. Practical Algorithms for PSF

In the following, we assume that a complete protein sequence $\mathcal{P}$ is always given. Our first algorithm is called *Align+Greedy*. We first align $\mathcal{J}$ to $\mathcal{P}$ using the standard Needleman-Wunsch algorithm, and then we insert the missing amino acids at the $\star$ positions in the aligned $\mathcal{J}$ with a greedy method (i.e., according to the maximum of their B62 scores).

Algorithm 1

Align+Greedy($\mathcal{P}, \mathcal{J}, X$)

```

1   Align  $\mathcal{J}$  with  $\mathcal{P}$  to obtain  $\mathcal{J}_1$  with the maximum B62 score.
2   Find a  $\star$  position in the aligned  $\mathcal{J}_1$  such that inserting an amino acid in  $X$  would incur the maximum B62 score among all  $\star$  positions.

```

```

3   Repeat Step 2 until all elements in  $X$  are inserted into  $\mathcal{J}_1$ .
4   Return the filled  $\mathcal{J}_1$  as  $\mathcal{J}'$ , with the total alignment score between  $\mathcal{J}'$  and  $\mathcal{P}$  being  $b62(\mathcal{P}, \mathcal{J}', X)$ .

```

An improved version of our first algorithm is based on a local search method. The idea is that if Algorithm 1 does not give us the best return, there must be a way to locally update the solution to have a better result. The algorithm is as follows.

Algorithm 2

LocalSearch($\mathcal{P}, \mathcal{J}, X$)

```

1   Compute  $b62(\mathcal{P}, \mathcal{J}, X)$  using Algorithm 1. Assign  $U \leftarrow b62(\mathcal{P}, \mathcal{J}, X)$ .
2   Insert an amino acid  $x \in X$  into a position  $i$  of  $\mathcal{J}$  to obtain  $\mathcal{J}(i, x)$  such that the score  $b62(\mathcal{P}, \mathcal{J}, (i, x), X - \{x\})$  is maximum, for all  $x \in X$  and for all  $i \in [0, |\mathcal{J}|]$ .
3   Run Algorithm 1 to obtain  $b62(\mathcal{P}, \mathcal{J} + x, X - \{x\})$ . If  $b62(\mathcal{P}, \mathcal{J} + x, X - \{x\}) \leq U$ , then return the solution  $\mathcal{J}''$  incurring  $U$ ; else update  $U \leftarrow b62(\mathcal{P}, \mathcal{J}, (i, x), X - \{x\})$ ,  $\mathcal{J} \leftarrow \mathcal{J}(i, x)$ ,  $X \leftarrow X - \{x\}$ , and repeat Step 2.

```

We use two kinds of datasets to test the effectiveness of our algorithms, and they are from antibody proteins and mammalian proteins respectively. Our antibody dataset is based on MabCampath (or Alemtuzumab) and Humira (or Adalimumab), which are two similar antibody proteins. Both of them contain a light chain and a heavy chain, the lengths for them are 214 and 449 for MabCampath, and 214 and 453 for Humira respectively. The pairwise alignments of the two light chains and two heavy chains display 91.1% and 86.6% identity respectively. For each protein sequence we compute a set of peptides from bottom up tandem mass spectra using PEAKS [14], [15], which is a de novo peptide sequencing software tool. Then we simply select a maximal set of disjoint peptides for each protein sequence. For the light chain of MabCampath (M-L for short): we have two (disjoint) peptides of lengths 12 and 19. For the heavy chain of MabCampath (M-H for short): we have eight (disjoint) peptides of lengths 9, 7, 13, 12, 14, 15, 12 and 19. For the heavy chain of Humira (H-H for short): we have six (disjoint) peptides of lengths 7, 7, 9, 9, 10 and 8. For the light chain of Humira (H-L for short), PEAKS is not able to obtain any peptides of decent quality. So we will only use Humira-L for reference purpose. Due to that the amino acids *I* and *L* have the same mass, in the peptides and all our comparisons, all *I*'s have been converted to *L*'s.

Our mammalian dataset is based on two homologous protein sequences, which are carbonic anhydrase 2 (CAH2 for short) for *Bos taurus* and Human respectively. The length of these CAH2 proteins are 258 and 260 respectively. The number of aligned pairs between these two CAH2 proteins is 206, indicating an identity close to 80%. All the datasets can be found in www.cs.montana.edu/~qingge.letu/ResearchData.html. Our code was written in Matlab and Java.

C. Empirical Results for Antibody Proteins

Let $x \in \{M-H, H-H, M-L\}$ and let the corresponding references of x be $H-H$, $M-H$, and $H-L$ respectively. For each x , we use Algorithm 1 to compute the filled sequence $\mathcal{S}'_x$, and we use Algorithm 2 to compute the filled sequence $\mathcal{S}''_x$. As the B62 scores are not very indicative, in the following we use the number of matched pairs resulting from the computed B62 scores. The empirical results are shown in the following two tables. Note that in most cases, Algorithm 2 produces slightly better results. Of course, that should be considered as normal. In all the tables, R , T , ℓ_R , ℓ_T represent the reference protein sequence, target protein sequence and their lengths respectively; moreover, the first column indicates the scaffold represented by x . Recall that $m(-, -)$ represents the *matched pairs* between two sequences.

D. Empirical Results for Mammalian Proteins

We use CAH2 of *Bos taurus* (CB for short) as a target sequence and simply select different disjoint peptides as our scaffold from its bottom-up spectra, which can be generated by PEAKS. Our idea is to use CAH2 of Human (CH for short) as a reference to insert the missing amino acids into the incomplete scaffold to obtain a new complete sequence. Then we try to measure the similarity between the target sequence and a newly computed complete sequence. For the similarity measure, we again use the number of matched pairs resulting from the computed B62 scores instead of directly using the B62 scores of its own.

We have three groups of datasets which consist of distinct numbers of different disjoint peptides. The first group contains two disjoint peptides of lengths 15 and 11. The second group contains three disjoint peptides of lengths 9, 9 and 7. The third group contains four disjoint peptides of lengths 15, 9, 8 and 9. Due to the fact that the amino acids I and L have the same mass, all I 's have been changed to L in the peptides. We represent the three groups as CB2, CB3 and CB4 respectively.

Let $x, \bar{x} \in \{CB2, CB3, CB4\}$ and let the corresponding references all be CH. we use Algorithm 1 to compute the filled sequence $\mathcal{S}'_x$, and we use Algorithm 2 to compute the filled sequence $\mathcal{S}''_x$. Again, recall that ℓ_R , ℓ_T represent the length of the reference and target protein sequence respectively. The empirical results are shown in the following two tables.

IV. ALGORITHMS FOR EMPIRICAL RESULTS FOR CP-PSF

A. CP-PSF is Polynomially Solvable

In this section, we first show that CP-PSF is also solvable in polynomial time. As the time complexity of this algorithm is too high (hence infeasible for implementation), we just sketch a solution without necessarily trying to obtain the best running time.

Define $B[i, j, k, \ell, \mathcal{A}, \mathcal{B}]$ as the maximum B62 score when the last element of C_{ℓ_1} is aligned with $\mathcal{P}[i]$, and C_{ℓ} is aligned with $\mathcal{P}[i..k]$, where $\mathcal{A}, \mathcal{B}$ is the set of amino acids inserted in between C_{ℓ_1} and C_{ℓ} . Let $INS(\mathcal{P}[i+1..j-1], t[i, j, k, \ell])$ be the maximum B62 score obtained by inserting the amino acids in $\mathcal{A}, \mathcal{B}$ to align with the substring $\mathcal{P}[i+1..j-1]$. Apparently, $INS(-, -)$ can be computed using (the base cases of) the

dynamic programming algorithm summarized in Theorem 3.1, which takes at most $O(n^{22})$ time and space. Note that $*$'s could be inserted in $\mathcal{P}$ and $\mathcal{S}$, while the amino acids in X can only be inserted before and after each of the m contigs in $\mathcal{S}$. The optimal solution is $\max_{i,j,k} \{B[i, j, k, m, t[i, j, k, m]] + \text{INS}(\mathcal{P}[k+1..n], X - t[i, j, k, m])\}$, which might not be unique. As there are $O(mn^3) = O(n^4)$ cells for the first 4 entries in the table $B[i, j, k, \ell -]$, the time complexity of this algorithm is $O(n^4) \times O(n^{22}) = O(n^{26})$. We then have the following theorem.

Theorem 2—The Contig-Preserving PSF can be solved in $O(n^{26})$ time and space.

Unfortunately, the time complexity of this algorithm is too high (even after some possible improvement). Hence, we first design some practical algorithms and then show some empirical results for the CP-PSF problem using these practical algorithms.

B. Practical Algorithms for CP-PSF

We design a heuristic algorithm for CP-PSF, based on the Needleman-Wunsch algorithm. We scan from left to right in $\mathcal{P}$ to find the best alignment for $C_1, C_2, \dots, C_m$. Each C_i is aligned with (an unaligned part of) $\mathcal{P}$ using the Needleman-Wunsch algorithm. In the worst case, that would take $O(mn \max_i |C_i|)$ time. We then repeatedly replace a $*$, which does not lie in any C_i , with an amino acid in X which would incur the maximum B62 score.

Algorithm 3

ContigAlign+Greedy($\mathcal{P}, \mathcal{S}, X$)

- 1 Align $\mathcal{S}$ with $\mathcal{P}$ to obtain $\mathcal{S}_1$ with the maximum B62 score, using the Needleman-Wunsch algorithm.
- 2 Find a $*$ position in the aligned $\mathcal{S}_1$ (which does lie in any contig) such that inserting an amino acid in X would incur the maximum B62 score among all such $*$ positions.
- 3 Repeat Step 2 until all elements in X are inserted into $\mathcal{S}_1$.
- 4 Return the filled $\mathcal{S}_1$ as $\mathcal{S}'$, with the total alignment score between $\mathcal{S}'$ and $\mathcal{P}$ being $b62(\mathcal{P}, \mathcal{S}', X)$.

Similar to the idea in Section 3.2, we could use a local search idea to try to improve Algorithm 3. Here, we could augment the contigs by appending some amino acids at its two ends, but the initial contigs are never altered.

Algorithm 4

ContigLocalSearch($\mathcal{P}, \mathcal{S}, X$)

- 1 Compute $b62(\mathcal{P}, \mathcal{S}, X)$ using Algorithm 3.
Assign $U \leftarrow b62(\mathcal{P}, \mathcal{S}, X)$.
- 2 Insert an amino acid $x \in X$ at the beginning (resp. end) of contig $S_i \in \mathcal{S}$ to obtain $S_i + x$ such that the score $b62(\mathcal{P}, \mathcal{S} - \{S_i\} \cup \{S_i + x\}, X - \{x\})$ is the maximum, for all $x \in X$ and all $S_i \in \mathcal{S}$.

3 Run Algorithm 3 to obtain $b62(\mathcal{P}, \mathcal{S} - \{S_i\} \cup \{S_i+x\}, X - \{x\})$. If $b62(\mathcal{P}, \mathcal{S} - \{S_i\} \cup \{S_i+x\}, X - \{x\}) \leq U$, then return the solution $\mathcal{S}''$ incurring U ; else update $U \leftarrow b62(\mathcal{P}, \mathcal{S} - \{S_i\} \cup \{S_i+x\}, X - \{x\})$, $\mathcal{S} \leftarrow \mathcal{S} - \{S_i\} \cup \{S_i+x\}$, $X \leftarrow X - \{x\}$ and repeat Step 2.

C. Empirical Results for Antibody Proteins

Similar to Section III.C, let $x \in \{M-H, H-H, M-L\}$ and let the corresponding references of x be H-H, M-H, and H-L respectively. For each x , we use Algorithm 3 to compute the filled sequence $\mathcal{S}'_x$ and we use Algorithm 4 to compute the filled sequence $\mathcal{S}''_x$. Instead of directly using the B62 scores, we again use the number of matched pairs resulting from the computed B62 scores. The empirical results are shown in the following two tables. Note that in all cases, compared with Algorithm 3, with respect to a reference Algorithm 4 produces slightly better results. This can be seen in the first two columns in Table 5 and Table 6. However, with respect to the corresponding target, it is not always the case that Algorithm 4 performs better than Algorithm 3 (though the difference is small). This can be checked in the last two columns of Table 5 and Table 6 (and when $x = \text{MabCampath-H}$).

D. Empirical Results for Mammalian Proteins

Similar to Section III.D, $x \in \{CB2, CB3, CB4\}$ and let the corresponding references all be CH. We use Algorithm 3 to compute the filled sequence $\mathcal{S}'_x$ and we use Algorithm 4 to compute the filled sequence $\mathcal{S}''_x$. The empirical results are shown in the following two tables. It can be seen that Algorithm 4 performs at least as good as Algorithm 3, although the improvement is small.

V. CONCLUSIONS AND DISCUSSIONS

In this paper, we study the protein scaffold filling problem when a reference protein is given and we solve the two corresponding versions in $O(n^{22})$ and $O(n^{26})$ time, which are both impractical, respectively. We then design two practical methods, for each version, and obtain some empirical results using some datasets from four antibody protein sequences and mammalian proteins. Our empirical results are very promising: as long as the right reference protein and a high-quality scaffold are given, the algorithms can fill the scaffold with 73.6%–91% accuracy. However, to make the results practically useful a lot still need to be done. We discuss possible directions for further research as follows.

1. Is it possible to incorporate the structure similarity information so that the filled scaffolds are more likely to appear in nature? To the best of our knowledge, a lot of scaffolds are obtained using (both bottom-up and/or top-down) mass spectrometry and in these cases the structure information are never utilized.
2. The success of the above methods are very much dependent on selecting the right references. (Without a reasonable reference our algorithms cannot return good results.) In practice, this could be done by homology search in a database of protein sequences. (Of course, for some antibody proteins which are hard to find

in databases, this could be done through a different method, for example, using mass spectra.) The actual empirical testing on using protein databases will be an interesting direction for future research.

3. In practice, with top-down mass spectrometry, we might know the length (or weight) of the target protein sequence. In this case, we might only need to insert a subset $X' \subset X$ of the missing amino acids (of certain weight). The current greedy-based method do not seem to work for this scenario. This will also be another interesting direction for future research.
4. Finally, the running times of our (heuristic) algorithms are decent with the current datasets (each takes at most 20 seconds). In fact, we believe that it will not take more than a few minutes even when the proteins in consideration have a couple of thousand residues (though we do not have such a dataset yet). However, when combined with any one of the previous three options, the running time could then increase significantly. Special care should be taken in this respect.

Acknowledgments

This research was partially supported by NSF of China under grant 61628207 and by the Opening Fund of Top Key Discipline of Computer Software and Theory in Zhejiang Provincial Colleges at Zhejiang Normal University. Xiaowen Liu was supported by the National Institute of General Medical Sciences, NIH through grant R01GM118470. We also thank anonymous reviewers for several useful comments.

APPENDIX

Table 9

The BLOSUM62 score matrix.

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	9																			
S	-1	4																		
T	-1	1	5																	
P	-3	-1	-1	7																
A	0	1	0	-1	4															
G	-3	0	-2	-2	0	6														
N	-3	1	0	-2	-2	0	6													
D	-3	0	-1	-1	-2	-1	1	6												
E	-4	0	-1	-1	-1	-2	0	2	5											
Q	-3	0	-1	-1	-1	-2	0	0	2	5										
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8									
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5								
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5							
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5						
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4					
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4				
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4			

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6		
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7	
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11

Biographies

Letu Qingge Letu Qingge is currently a PhD student at Gianforte School of Computing, Montana State University, USA. His research interests are computational biology and algorithms.

Xiaowen Liu Xiaowen Liu is currently an assistant professor at School of Informatics and Computing, IUPUI, USA. He obtained his Ph.D. in Computer Science from City University of Hong Kong in 2008. Between 2008 and 2012, he had been a postdoctoral fellow at University of Western Ontario, University of Waterloo and UC-San Diego. Dr. Liu’s research interests are bioinformatics, computational biology and algorithms. He has published over 40 papers in these areas.

Farong Zhong Farong Zhong is currently a professor in College of Mathematics, Physics and Information Engineering, Zhejiang Normal University, China. He obtained his Ph.D. in Computer Science from Shanghai JiaoTong University, China in 2005. Dr. Zhong’s research

interests are formal methods, algorithms and theory of computation. He has published over 40 papers in these areas.

Binhai Zhu Binhai Zhu is currently a professor in computer science at Montana State University, USA. He obtained his Ph.D. in Computer Science from McGill University, Canada, in 1994. He was a post-doctoral research associate at Los Alamos National Laboratory, USA from 1994 to 1996. From 1996 to 2000, he was an assistant professor at City University of Hong Kong. He has been at Montana State University since 2000 (associate professor until 2006, professor since 2006). Professor Zhu's research interests are geometric computing, biological/geometric modeling, bioinformatics and combinatorial optimization. He has published over 170 papers in these areas.

References

1. Bandeira N, Pham V, Pevzner P, Arnott D, Lill J. Beyond Edman Degradation: Automated De novo Protein Sequencing of Monoclonal Antibodies. *Nat Biotechnol.* 2008; 26(12):1336–1338. [PubMed: 19060866]
2. Bandeira N, Tang H, Bafna V, Pevzner P. Shotgun protein sequencing by tandem mass spectra assembly. *Anal Chem.* 2004; 76:7221–7233. [PubMed: 15595863]
3. Bulteau L, Carrieri AP, Dondi R. Fixed-parameter algorithms for scaffold filling. *Theo Comput Sci.* 2015; 568:72–83.
4. Downey, R., Fellows, M. *Parameterized Complexity.* Springer-Verlag; 1999.
5. Flum, J., Grohe, M. *Parameterized Complexity Theory.* Springer-Verlag; 2006.
6. Garey, MR., Johnson, DS. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H Freeman; 1979.
7. Henikoff S, Henikoff J. Amino acid substitution matrices from protein blocks. *PNAS.* 1992; 89(22): 10915–10919. [PubMed: 1438297]
8. Jiang, H., Zhong, F., Zhu, B. *Proc 22nd Annual Combinatorial Pattern Matching Symposium (CPM'11), LNCS 6661.* Palermo, Italy: Jun 27–29. 2011 Filling scaffolds with gene repetitions: maximizing the number of adjacencies; p. 55-64.
9. Jiang H, Zheng C, Sankoff D, Zhu B. Scaffold filling under the breakpoint and related distances. *IEEE/ACM Transactions on Computational Biology and Bioinformatics.* Jul-Aug;2012 9(4):1220–1229. [PubMed: 22529329]
10. Liu N, Jiang H, Zhu D, Zhu B. An improved approximation algorithm for scaffold filling to maximize the common adjacencies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics.* Jul-Aug;2013 10(4):905–913. [PubMed: 24334385]
11. Liu N, Zhu D, Jiang H, Zhu B. A 1.5-approximation algorithm for two-sided scaffold filling. *Algorithmica.* 2016; 74(1):91–116.
12. Liu X, Han Y, Yuen D, Ma B. Automated protein (re)sequencing with MS/MS and a homologous database yields almost full coverage and accuracy. *Bioinformatics.* 2009; 25:2174–2180. [PubMed: 19535534]

13. Liu X, Dekker L, Wu S, Vanduijn M, Luider T, Tolic N, Kou Q, Dvorkin M, Alexandrova S, Vyatkina K, Pasa-Tolic L, Pevzner P. De Novo protein sequencing by combining top-down and bottom-up tandem mass spectra. *J Proteome Res.* 2014; 13:3241–3248. [PubMed: 24874765]
14. Ma B, Zhang K, Hendrie C, Liang C, Li M, Doherty-Kirby A, Lajoie G. PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry. *Rapid Communications in Mass Spectrometry.* 2003; 17(20):2337–42. [PubMed: 14558135]
15. Ma B, Zhang K, Liang C. An effective algorithm for peptide de novo sequencing from MS/MS spectra. *J Comp Sys Sci.* 2005; 70(3):418–430.
16. Muñoz A, Zheng C, Zhu Q, Albert V, Rounsley S, Sankoff D. Scaffold filling, contig fusion and gene order comparison. *BMC Bioinformatics.* 2010; 11:304. [PubMed: 20525342]
17. Needleman S, Wunsch C. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Molecular Biology.* 1970; 48(3):443–453.
18. Petrokovski S, Henikoff J, Henikoff S. The Blocks database - a system for protein classification. *Nucl Acids Res.* 1996; 24(1):197–200. [PubMed: 8594578]
19. Zhu B. Genomic scaffold filling: A progress report. *Proc 10th Intl Frontiers of Algorithmics Workshop (FAW'16), LNCS 9711.* 2016:8–16.

Table 1

Empirical results for the three computed $\mathcal{J}_x'$ using Algorithm 1, with respect to the antibody dataset. M-H (resp. M-L) represents the heavy (resp. light) chain of MabCampath, and H-H represents the heavy chain of Humira.

	$\mathbf{m}(\mathcal{J}_x', R)$	$\mathbf{m}(\mathcal{J}_x', R)/\ell_R$	$\mathbf{m}(\mathcal{J}_x', T)$	$\mathbf{m}(\mathcal{J}_x', T)/\ell_T$
M-H	406	406/453	355	355/449
		or, 89.62%		or, 79.06 %
H-H	442	442/449	385	385/453
		or, 98.44%		or, 84.99 %
M-L	214	214/214	195	195/214
		or, 100%		or, 91.12%

Table 2

Empirical results for the three computed $\mathcal{J}_x''$ using Algorithm 2, with respect to the antibody dataset. M-H (resp. M-L) represents the heavy (resp. light) chain of MabCampath, and H-H represents the heavy chain of Humira.

	$m(\mathcal{J}_x'', R)$	$m(\mathcal{J}_x'', R)/\ell_R$	$m(\mathcal{J}_x'', T)$	$m(\mathcal{J}_x'', T)/\ell_T$
M-H	415	415/453	370	370/449
		or, 91.61%		or, 82.41 %
H-H	442	442/449	385	385/453
		or, 98.44%		or, 84.99 %
M-L	214	214/214	195	195/214
		or, 100%		or, 91.12%

Table 3

Empirical results for the three computed $\mathcal{S}'_x$ using Algorithm 1, with respect to the mammalian dataset. $CBi(i = 2, 3, 4)$ represents the i peptides of CAH2 of *Bos taurus*. CH represents CAH2 of Human, which is the reference.

	$m(\mathcal{S}'_x, R)$	$m(\mathcal{S}'_x, R)/\ell_R$	$m(\mathcal{S}'_x, T)$	$m(\mathcal{S}'_x, T)/\ell_T$
CB2	258	258/260	206	206/258
		or, 99.23%		or, 79.84 %
CB3	252	252/260	204	204/258
		or, 96.92%		or, 79.07 %
CB4	241	241/260	194	194/258
		or, 79.07%		or, 75.19%

Table 4

Empirical results for the three computed $\mathcal{J}_x''$ using Algorithm 2, with respect to the mammalian dataset. $\text{CB}i(i = 2, 3, 4)$ represents the i peptides of CAH2 of *Bos taurus*. CH represents CAH2 of Human, which is the reference.

	$m(\mathcal{J}_x'', R)$	$m(\mathcal{J}_x'', R)/\ell_R$	$m(\mathcal{J}_x'', T)$	$m(\mathcal{J}_x'', T)/\ell_T$
CB2	258	258/260	206	206/258
		or, 99.23%		or, 79.84 %
CB3	254	254/260	203	203/258
		or, 97.69%		or, 78.68%
CB4	247	247/260	197	197/258
		or, 95.00%		or, 76.36%

Table 5

Empirical results for the three computed $\mathcal{S}'_x$ using Algorithm 3, with respect to the antibody dataset. M-H (resp. M-L) represents the heavy (resp. light) chain of MabCampath, and H-H represents the heavy chain of Humira.

	$m(\mathcal{S}'_x, R)$	$m(\mathcal{S}'_x, R)/\ell_R$	$m(\mathcal{S}'_x, T)$	$m(\mathcal{S}'_x, T)/\ell_T$
M-H	369	369/453	347	347/449
		or, 81.46%		or, 77.28%
H-H	421	421/449	363	363/453
		or, 93.76%		or, 80.13 %
M-L	214	214/214	195	195/214
		or, 100%		or, 91.12%

Table 6

Empirical results for the three computed $\mathcal{S}_x''$ using Algorithm 4, with respect to the antibody dataset. M-H (resp. M-L) represents the heavy (resp. light) chain of MabCampath, and H-H represents the heavy chain of Humira.

	$m(\mathcal{S}_x'', R)$	$m(\mathcal{S}_x'', R)/\ell_R$	$m(\mathcal{S}_x'', T)$	$m(\mathcal{S}_x'', T)/\ell_T$
M-H	377	377/453	345	345/449
		or, 83.22%		or, 76.84 %
H-H	422	422/449	364	364/453
		or, 93.97%		or, 80.35 %
M-L	214	214/214	195	195/214
		or, 100%		or, 91.12%

Table 7

Empirical results for the three computed $\mathcal{S}'_x$ using Algorithm 3., with respect to the mammalian dataset. CBi ($i = 2, 3, 4$) represents the i peptides of CAH2 of *Bos taurus*. CH represents CAH2 of Human, which is the reference.

	$m(\mathcal{S}'_x, R)$	$m(\mathcal{S}'_x, R)/\ell_R$	$m(\mathcal{S}'_x, T)$	$m(\mathcal{S}'_x, T)/\ell_T$
CB2	248	248/260	198	198/258
		or, 95.38%		or, 76.74 %
CB3	241	241/260	194	194/258
		or, 92.69%		or, 75.19 %
CB4	235	235/260	190	190/258
		or, 90.38%		or, 73.64%

Table 8

Empirical results for the three computed $\mathcal{S}_x''$ using Algorithm 4, with respect to the mammalian dataset. $CBi(i = 2, 3, 4)$ represents the i peptides of CAH2 of *Bos taurus*. CH represents CAH2 of Human, which is the reference.

	$m(\mathcal{S}_x'', R)$	$m(\mathcal{S}_x'', R)/\ell_R$	$m(\mathcal{S}_x'', T)$	$m(\mathcal{S}_x'', T)/\ell_T$
CB2	248	248/260	198	198/258
		or, 95.38%		or, 76.74 %
CB3	242	242/260	194	194/258
		or, 93.08%		or, 75.19 %
CB4	237	237/260	194	194/258
		or, 91.15%		or, 75.19%