Journal of Computer and System Sciences **65**, 73–96 (2002)
doi:10.1006/jcss.2002.1823

Finding Similar Regions in Many Sequences[1]

Ming Li[2]

Department of Computer Science, University of California, Santa Barbara, California 93106
E-mail: mli@cs.ucsb.edu

Bin Ma[3]

Department of Computer Science, University of Western Ontario, London, Ontario N6A5B7, Canada
E-mail: bma@csd.uwo.ca

and

Lusheng Wang[4]

City University of Hong Kong, Kowloon, Hong Kong
E-mail: lwang@cs.cityu.edu.hk

Received July 1, 1999; revised July 1, 2001

Algorithms for finding similar, or highly conserved, regions in a group of sequences are at the core of many molecular biology problems. Assume that we are given n DNA sequences $s_1, ..., s_n$. The Consensus Patterns problem, which has been widely studied in bioinformatics research, in its simplest form, asks for a region of length L in each s_i, and a median string s of length L so that the total Hamming distance from s to these regions is minimized. We show that the problem is NP-hard and give a polynomial time approximation scheme (PTAS) for it. We then present an efficient approximation algorithm for the consensus pattern problem under the original relative entropy measure. As an interesting application of our analysis, we further obtain a PTAS for a restricted (but still NP-hard) version of the important consensus alignment problem allowing at most constant number of gaps, each of arbitrary length, in each sequence. © 2002 Elsevier Science (USA)

Key Words: approximation algorithms; consensus patterns; consensus alignment; computational biology.

[1] Some of the results in this paper have appeared as a part of an extended abstract presented in "Proc. 31st Symp. Theory of Computing, May 1999."

[2] Supported in part by the NSERC Research Grant OGP0046506, a CGAT grant, the E. W. R. Steacie Fellowship at the University of Waterloo and NSF-ITR Grant 0085801 at UCSB.

[3] Supported by NSERC research grant and a start-up grant from UWO.

[4] Fully supported by HK RGC Grants 9040444.

1. INTRODUCTION

Many problems in molecular biology involve finding similar regions common to each sequence in a given set of DNA, RNA, or protein sequences. These problems find applications in locating binding sites and finding conserved regions in unaligned sequences [7, 10, 20, 21]. One may also regard such problems as various generalizations of the common substring problem, allowing errors. Indeed, different ways of measuring errors give different problems with very different flavors in their complexity and algorithms to solve them. They are natural and fundamental problems in both molecular biology and computer science and require sophisticated ideas in designing and analyzing their algorithms. In this paper, we study the so-called Consensus Patterns problem. This problem has been widely studied, and heuristic algorithms for this problem have been implemented, in bioinformatics research [2, 3, 7, 9, 10 17, 18, 20, 21, 26]. We will show that the problem most likely does not have efficient soluticins by showing that it is NP- complete. We will then present efficient approximation algorithms under various measures. We will also generalize these ideas to study the well-known consensus multiple alignment problem and obtain a PTAS for a restricted version of the consensus multiple alignment problem.

Let s and s' be finite strings. If not otherwise mentioned, our strings are over the alphabet $\Sigma = \{1, 2, ..., A\}$, where A is usually 4 or 20 in practice. $|s|$ is the length of s. $s[i]$ is the ith character of s. Thus, $s = s[1]\, s[2] ... s[|s|]$. $d_E(s, s')$ denotes the edit distance between s and s'. When $|s| = |s'|$, $d_H(s, s')$ means the Hamming distance between s and s'.

We now define the problems that will be studied in this paper:

Consensus Patterns: Given a set $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ of sequences each of length m, and an integer L, find a median string s of length L and a substring t_i *(consensus patterns)* of length L from each s_i, minimizing $\sum_{i=1}^{n} d_H(s, t_i)$. We call this the H-cost.

Max Consensus Patterns: The set of instances of Max consensus patterns is the same as that of Consensus Patterns. For Max Consensus Patterns, find substrings $t_1, ..., t_n$, where each t_i is a string of length L of s_i and another substring s also of length L, such that the quantity $nL - \sum_{i=1}^{n} d_H(s, t_i)$ is minimized. We call this the M-cost.

General Consensus Patterns: The set of instances of General Consensus Patterns is the same as that of Consensus Patterns. For General Consensus Patterns, find substrings $t_1, ..., t_n$, where each t_i is a substring of length L of s_i, such that the quantity (*the average information content in each unit length of the consensus patterns*)

$$\frac{1}{L} \sum_{j=1}^{L} \sum_{a \in \Sigma} f_j(a) \log \frac{f_j(a)}{p(a)}$$

is maximized, where $f_j(a)$ denotes the frequency of letter a in the jth letters of t_i's, $p(a)$ denotes the frequency of letter a in the whole genome. We call this the I-cost.

Consensus Alignment: Given a set $\mathscr{S} = \{s_1, s_2, ..., s_n\}$ of strings each of length m, find a median sequence s minimizing $\sum_{i=1}^{n} d_E(s_i, s)$.

General Consensus Patterns is defined in various forms in [7, 10, 20, 21], in search of conserved regions or common sites in a set of unaligned biosequences. It is the central term to minimize in various objective functions in [7, 10, 20, 21]. The authors in [7, 10, 20, 21] gave heuristic or exponential time algorithms and developed working systems for this problem. Other related software and applications can be found in [2, 3, 9, 17, 18, 26]. Taking only the maximum term without the log factor in each c_j in General Consensus Patterns gives Consensus Patterns and its complement Max Consensus Patterns. The median strings in consensus patterns problem can be used as *motifs* in *repeated- motif* methods for multiple sequence alignment problems [6, 16, 19, 23–25] that repeatedly find motifs and recursively decompose the sequences into shorter sequences.

Another motivation for studying Consensus Patterns is that it is applicable to a restricted case of Consensus Alignment. Consensus Alignment is one of the most important problems in computational biology [6]. The problem is to find a median sequence minimizing the total edit distance between each given sequence and the median sequence. A multiple sequence alignment can be constructed based on the pairwise alignments between the given sequences and the median sequence. The best known approximation algorithm for consensus multiple alignment has performance ratio $2-o(1)$ [6]. A closely related problem, SP-alignment, has also been extensively studied recently. With much effort, the best-known performance ratio for SP-alignment has been improved from $2-\frac{2}{k}$ to $2-\frac{l}{k}$ for any constant l, where k is the number of the sequences [1, 5, 15]. The $2-o(1)$ barrier appears to be formidable. In a companion paper [12], we will study similar problem where the median string is required to be close to all sequences.

The paper is organized as follows. In Section 2, we show that Consensus Patterns is NP-hard. The problem resembles Consensus Alignment and thus a better than $2-o(1)$ ratio seems to be hard to achieve. Interestingly, in Section 3 we are able to design a PTAS for Consensus Patterns. And in Section 4, we give a PTAS for the Max Consensus Patterns problem. In Section 5, we present an efficient approximation algorithm for General Consensus Patterns. In Section 6, the ideas are then applied to a restricted version of Consensus Alignment, restricting the number of gaps in the pairwise alignment between any given sequence and the median sequence to be at most a constant. We call it Consensus c-Alignment. The problem is still very interesting since constant number of gaps may very well be good enough for some practical problems. We show that the Consensus c-Alignment problem remains NP-hard and give a PTAS for it.

2. CONSENSUS PATTERNS PROBLEMS IS NP-HARD

We prove Theorem 1 in this section. It is easy to see that Max Consensus Patterns is the complementary problem of Consensus Patterns. Therefore, if Theorem 1 is correct, we can also conclude that Max Consensus Patterns is NP-hard.

THEOREM 1. CONSENSUS PATTERNS *is NP-hard if the size of alphabet is four.*

Proof. The reduction is from Max Cut-3 that is NP- complete even if the degree of the node is at most 3 [4]. Let $G=(V,E)$ be a graph with degree bounded by 3, where $V=\{v_1, v_2, ..., v_n\}$. Define $\Sigma=\{0, 1, \sharp, *\}$. The letter $\sharp$ serves as a delimiter. For each $v_i \in V$, we construct a string $s_i = *^5 z_{i,1} z_{i,2} \cdots z_{i,n} D$, where $D=\sharp^6$ and

$$z_{i,j} = \begin{cases} D0*^3D1*^3 & \text{if } \; j \neq i,\, v_i \text{ and } v_j \text{ are adjacent} \\ D*^4D*^4 & \text{if } \; j \neq i,\, v_i \text{ and } v_j \text{ are not adjacent} \\ D1^4\, D0^4 & \text{if } \; j = i. \end{cases}$$

Observe that s_i is of length $20n+11$ and in general s_i has the form

$$*^5\, (Dx_{i,1}\; Dy_{i,1})(Dx_{i,2}\; Dy_{i,2}) \cdots (Dx_{i,n-1}\; Dy_{i,n-1})(Dx_{i,n}\; Dy_{i,n})\; D,$$

where $x_{i,j}$ is one of $0*^3$, $*^4$, and 1^4, and $y_{i,j}$ is one of $1*^3$, x^4, and 0^4.
Similarly, let $t_i = u_{i,1} u_{i,2} \cdots u_{i,n}\; D$, where

$$u_{i,j} = \begin{cases} D*^4\; D*^4 & \text{if } \; j \neq i \\ D1^4\; D0^4 & \text{if } \; j = i. \end{cases}$$

Define

$$X_0 = \{s_i \mid i = 1, 2, ..., n\},$$
$$X_1 = \{b_i *^5\, (D1^4\, D*^4)^{n-1}\, D1^4\, D,\, b_i *^5\, (D0^4\, D*^4)^{n-1}\, D0^4\, D \mid b_i = 0, 1, ..., 2^{\lceil \log n^2 \rceil} - 1,$$
$$\text{that is represented as a binary number of } \lceil \log n^2 \rceil \text{bits}\},$$
$$X_2 = \{c_i *^5 t_j \mid c_i = 00, 01, 10, 11 \text{ and } j = 1, 2, ..., n\}.$$

Here b_i and c_i at the beginning of every string in both X_1 and X_2 ensure that the strings in both X_1 and X_2 are distinct. Note that each s_i in X_0 is of length $20n+11$, each string in X_1 is of length $20n+1+\lceil \log n^2 \rceil$, and each string in X_2 is of length $20n+13$. To match the definition of the consensus patterns problem, one can add some $\sharp$'s at the left end of the strings in $X_0 \cup X_2$ so that every string in $X_0 \cup X_1 \cup X_2$ is of length $20n+1+\lceil \log n^2 \rceil$. Finally, L is defined to be $20n-4$. (The segment $*^5$ in each of the constructed strings is not necessary for the proof of this theorem. However, it is useful for the proof of Theorem 15.)

CLAIM 2. *The median string in an optimal solution of consensus patterns* $(X_0 \cup X_1 \cup X_2, 20n-4)$ *can be modified to be in the form*

$$(Dx_1\; D*^4)(Dx_2\; D*^4) \cdots (Dx_{n-1}\; D*^4)\; Dx_n\; D,$$

where each of the x_i*'s is either* 0^4 *or* 1^4.

Proof. This is due to the following reasons.

1. The strings in X_1 (there are at least $2n^2$ strings) force the median string to be of the form

$$(DY_1\ D^{*4})(DY_2\ D^{*4})\cdots(DY_{n-1}\ D^{*4})\ DY_n\ D,$$

where Y_i is a string containing four letters, each of which is either 0 or 1. First, we show that the median string cannot contain any $0/1$ bits at the left end corresponding to the $0/1$ bits from b_i's. Otherwise, at most half of the letters for the strings in X_1 will be matched at those bits. However, if the median does not contain any $0/1$ bits at the left end corresponding to the $0/1$ bits from b_i's (move to the right), then at least half of the $|X_1|$ letters will be matched at those corresponding bits. Though one may think that keeping some $0/1$ bits at the left end of the median string corresponding to the $0/1$ bits from bi's may benefit some strings in X_0 and X_2, from the construction, the total number of 0's and 1's in the strings in $X_0 \cup X_2$ is $O(n)$. Thus, the total number of extra matches is at most $O(n \times \lceil \log n^2 \rceil)$ if we keep $\lceil \log n^2 \rceil$ $0/1$ bits at the left end of the median string. However, if we keep some $0/1$ bits at the left end of the median string, then the median string must contain less $\sharp$'s than the specified form in Claim 2, since each D contains six $\sharp$'s, there is a segment $*^5$ right after b_i's, and $L = 20n - 4$. This leads to $O(|X_1|) = O(n^2)$ extra mismatches. Thus, the median string cannot contain any $0/1$ bit at the left end corresponding to the $0/1$ bits from b_i's. Moreover, if part of the segment $*^5$ is in the median string, without changing the cost we can delete them and add more $\sharp$'s at the right end of the median string.

2. The $4n$ strings in X_2 further force Y_i to be either 1^4 or 0^4. This is because in the segment $u_{i,i} = D1^4\ D0^4$ of t_i, either 1^4 matches Y_i or 0^4 matches Y_i. The four copies of $u_{i,i}$ in X_2 force Y_i to be either 1^4 or 0^4. ▮

From Claim 2 we derive the following observation.

Observation 3. Let s and s' be two strings of length L in the form specified in Claim 2. Let us write $X_1 \cup X_2$ as $\{w_1, w_2, \ldots, w_l\}$. Then the minima of the quantities $\sum_{i=1}^{l} d_H(q_i, s)$ and $\sum_{i=1}^{l} d_H(q_i, s')$, where each q_i ranges over the set of substrings of length L of w_i, are equal.

Observation 3 tells us that the choice of x_i (0^4 or 1^4) in the median string is irrelevant to the cost contributed by strings in $X_1 \cup X_2$. We use $c(X_1 \cup X_2)$ to denote the total cost contributed by the strings in $X_1 \cup X_2$. However, the choice of x_i is crucial to the cost contributed by the strings in X_0.

Suppose that there is a partition (V_0, V_1) of V, which cuts c edges. The median sequence can be constructed as

$$Dx_1\ D^{*4})(Dx_2\ D^{*4})\cdots(Dx_{n-1}\ D^{*4})(Dx_n\ D),$$

where each x_i is 1^4 if v_i is in V_1, and 0^4 otherwise.

Note that s_i contains a segment $z_{i,i} = D1^4\ D0^4$. The setting of x_i (0^4 or 1^4) will determine the cutting of s_i since for each s_i there are at most three 1's and three 0's in all $z_{i,j}$ for $i \neq j$ in s_i, and we have to match either 1^4 in $z_{i,i}$ to x_i if $x_i = 1^4$ or 0^4 in $z_{i,i}$ to x_i if $x_i = 0^4$. For each i, if x_i of the median string is 1^4 (i.e., $v_i \in V_1$), we have

```
######0***######1***######1111######0000######0***######1***######  s_i
######1111######***######1111######***######0000######            median
                                (a)

######0***######1***######1111######0000######0***######1***######   s_i
            ######0000######***######0000######***######1111######  median
                                (b)
```

FIG. 1. (a) v_i is in V_1. (b) v_i is in V_0.

to cut off $y_{i,n}$ and the right end delimiters of s_i as in Fig. 1a, and if x_i of the median string is 0^4 (i.e., $v_i \in V_0$), we have to cut off the left end delimiters of s_i and x_1 as in Fig. 1b. Note that each segment $z_{j,i} = D0^{*3}\,D1^{*3}$ for $j \neq i$ contributes cost by either 4 or 5. If $v_i \in V_1$, then for each v_j adjacent to v_i, the segment $z_{j,i}$ of s_j, which is of the form $D0^{*3}\,D1^{*3}$, will contribute 5 toward the cost if and only if $v_j \in V_1$, as in Fig. 4a. Similarly, if $v_i \in V_0$, then for each v_j adjacent to v_i, the segment $z_{j,i}$ of s_j will contribute 5 toward the cost if and only if $v_j \in V_0$, as in Fig. 4b. That is, all the edges that are not cut by the partition are counted once more here.

Let $c(v)$ denote the number of edges incident upon v that are cut by the partition. For each $v_i \in V$, s_i contributes $m_i - c(v_i)$ to the cost, where m_i is a number purely determined by n and the degree of v_i in G. Thus the total cost is

$$c(X_1 \cup X_2) + \sum_{i=1}^{n} [m_i - c(v_i)] = c(X_1 \cup X_2) + \sum_{i=1}^{n} m_i - \sum_{i=1}^{n} c(v_i)$$
$$= c(X_1 \cup X_2) + \sum_{i=1}^{n} m_i - 2c.$$

Conversely, given an optimal solution for the instance of the consensus patterns problem with cost $c(X_1 \cup X_2) + \sum_{i=1}^{n} m_i - 2c$, we can modify the solution to satisfy Claim 2. Then, one can easily construct a partition of G that cuts c edges by looking at the $0-1$ assignment to x_i's in the median string, i.e., if x_i is 0^4, then $v_i \in V_0$ and if x_i is 1^4, then $v_i \in V_1$. ∎

3. A PTAS FOR Consensus Patterns

We have shown that the Consensus Patterns problem is NP-hard. In this section, we present a polynomial time approximation scheme (PTAS) for the Consensus Patterns problem. This is the best one can hope for, assuming $NP \neq P$. Like many other approximation problems, while our algorithm is a simple greedy strategy, the analysis is quite interesting and intricate. The key idea is this: there are always a few "important" substrings, their consensus holds most of the "secrets" of the true optimal median string. If we simply do exhaustive search to find these few substrings, then the trivial optimal solution for these few substrings will do very well to approximate the real optimal solution.

To give our algorithm, we need the following definitions. Let $t_1, t_2, \ldots, t_k$ be k strings of length L. Overlaying them as a k by L matrix, we consider the letters column by column. The *majority letter* for the k letters in a column is the letter

```
Algorithm consensusPattern
Input  $n$ sequences $\{s_1, s_2, \ldots, s_n\} \subseteq \Sigma^m$, integers $L$ and $r$.
Output the median string and consensus patterns.
1. for every $r$ length-$L$ substrings $u_1, u_2, \ldots, u_r$ from $s_1, \ldots, s_n$ do
   (a) Set $u$ to be the column-wise majority string of $u_1, u_2, \ldots, u_r$;
   (b) for $i = 1, 2, \ldots, n$ do
           Set $v_i$ to be the substring of length $L$ from $s_i$ that is the
           closest to $u$.
   (c) Let
           $c(u) = \sum_{i=1}^{n} d_H(u, v_i)$.
2. Output the $u$ and the corresponding $v_1, v_2, \ldots, v_n$ that minimize
   $c(u)$ in step 1.
```

FIG. 2. PTAS for CONSENSUS PATTERNS.

which appears the most. A *column-wise majority string* for $t_1, t_2, \ldots, t_k$ is the string of L majority letters, one for each column.

Now we give our algorithm in Fig. 2. We will show that the algorithm is a PTAS for CONSENSUS PATTERNS.

THEOREM 4. *The algorithm consensusPattern is a PTAS for* CONSENSUS PATTERNS. *More precisely, with n input sequences, each of length m, for any $r \geqslant 3$, the algorithm outputs a solution with H-cost no more than*[5]

$$\left(1 + O\left(\sqrt{\frac{\log r}{r}}\right)\right) \times c_{opt} \tag{1}$$

in time $O((m-L+1)^{r+1} n^{r+1} L)$, where c_{opt} is the $\mathscr{H}$-cost of an optimal solution.

Proof. Step 1(a) takes $O(L)$ time, step 1(b) takes $O(n(m-L+1)\,L)$ time, and step 1(c) takes $O(nL)$ time. Step 1 is repeated for at most $(n(m-L+1))^r$ times. So, the time complexity of the algorithm is $O((m-L+1)^{r+1} n^{r+1} L)$.

Now, we prove the performance ratio. Given an instance of CONSENSUS PATTERNS, we use s^* and $t_1, t_2, \ldots, t_n$ to denote the median string and consensus patterns in an optimal solution, respectively. The optimal solution ensures that the following two statements are true:

1. t_i is the length-L substring of s_i that is the closest to s^*.
2. s^* is the column-wise majority string of $t_1, t_2, \ldots, t_n$.

For any $1 \leqslant i_1, i_2, \ldots, i_r \leqslant n$, let $s_{i_1, i_2, \ldots, i_r}$ be a column-wise majority string of the r substrings $t_{i_1}, t_{i_2}, \ldots, t_{i_r}$. We want to approximate s^* by $s_{i_1, i_2, \ldots, i_r}$ for some $1 \leqslant i_1, i_2, \ldots, i_r \leqslant n$. Denote

[5] A better bound $(1 + \frac{4A-4}{\sqrt{e}(\sqrt{4r+1}-3)}) \times c_{opt}$ is given in Theorem 7.

$$c_{opt} = \sum_{i=1}^{n} d_H(s^*, t_i)$$

and

$$c_{i_1, i_2, \ldots, i_r} = \sum_{i=1}^{n} d_H(s_{i_1, i_2, \ldots, i_r}, t_i). \tag{2}$$

Let $i_1, i_2, \ldots, i_r$ be r independent and randomly chosen numbers from $\{1, 2, \ldots, n\}$. We will prove that

$$E[c_{i_1, i_2, \ldots, i_r}] \leqslant \left(1 + O\left(\sqrt{\frac{\log r}{r}}\right)\right) c_{opt}. \tag{3}$$

Let $\rho = (1 + O(\sqrt{\frac{\log r}{r}}))$. Inequality (3) says that the expected value of $c_{i_1, i_2, \ldots, i_r}$ is no more than ρc_{opt}. Therefore, if (3) is true, there must be one group of indices $1 \leqslant i'_1, i'_2, \ldots, i'_r \leqslant n$ such that $c_{i'_1, i'_2, \ldots, i'_r}$ is no more than ρc_{opt}.

Since we try all possible r substrings in step 1 of the algorithm, at some point, we will have $u_j = t_{i'_j}$ $(j = 1, 2, \ldots, r)$ and therefore $u = s_{i'_1, i'_2, \ldots, i'_r}$. By the definition of $c(u)$ in step 1(c) and the definition of $c_{i_1, i_2, \ldots, i_r}$ in (2), we have $c(u) \leqslant c_{i'_1, i'_2, \ldots, i'_r}$. Hence $c(u)$ is also no more than ρc_{opt}. That is, Algorithm consensusPattern finds a solution with H-cost no more than ρc_{opt}. Thus, to prove the theorem, we only have to prove (3).

For any character $a \in \Sigma$, let $h_j(a)$ be the number of i such that $1 \leqslant i \leqslant n$ and $t_i[j] = a$. Then for any string s of length L, we have

$$\sum_{i=1}^{n} d_H(t_i, s) = \sum_{j=1}^{L} (n - h_j(s[j])). \tag{4}$$

Thus, we have

$$c_{opt} = \sum_{i=1}^{n} d_H(s^*, t_i) = \sum_{j=1}^{L} [n - h_j(s^*[j])], \tag{5}$$

and

$$E[c_{i_1, i_2, \ldots, i_r}] = E\left[\sum_{j=1}^{L} (n - h_j(s_{i_1, i_2, \ldots, i_r}[j]))\right] = \sum_{j=1}^{L} E[(n - h_j(s_{i_1, i_2, \ldots, i_r}[j]))]. \tag{6}$$

From (5) and (6), to prove (3), it is sufficient to prove that for any $1 \leqslant j \leqslant L$,

$$E[n - h_j(s_{i_1, i_2, \ldots, i_r}[j])] \leqslant \rho \times (n - h_j(s^*[j])). \tag{7}$$

Substracting $n - h_j(s^*[j])$ from both sides of (7), (7) is equivalent to

$$E[h_j(s^*[j]) - h_j(s_{i_1, i_2, \ldots, i_r}[j])] \leqslant O\left(\sqrt{\frac{\log r}{r}}\right) \times (n - h_j(s^*[j])). \tag{8}$$

Since $s_{i_1, i_2, ..., i_r}[j]$ is the majority letter of $t_{i_1}[j]$, $t_{i_2}[j]$, ..., $t_{i_r}[j]$ and $s^*[j]$ is the majority letter of $t_1[j]$, $t_2[j]$, ..., $t_n[j]$, Inequality (7) can be proved by the following lemma.

Lemma 5. *Let $a_1, a_2, ..., a_n \in \Sigma$. For each $a \in \Sigma$, let $h(a)$ be the number of i such that $1 \leqslant i \leqslant n$ and $a_i = a$. Let $i_1, i_2, ..., i_r$ be r independent and randomly chosen numbers from $\{1, 2, ..., n\}$. Let a^r be a majority letter of $a_{i_1}, a_{i_2}, ..., a_{i_r}$ and a^* be a majority letter of $a_1, a_2, ..., a_n$. If $r \geqslant 3$, then*

$$E[h(a^*) - h(a^r)] \leqslant O\left(\sqrt{\frac{\log r}{r}}\right)(n - h(a^*)). \tag{9}$$

Proof. Let $m(a)$ be the number of j such that $1 \leqslant j \leqslant r$ and $a_{i_j} = a$. Let $p_a = \frac{h(a)}{n}$. Then for each j, the probability of $a_{i_j} = a$ is p_a. We prove the lemma in two cases:

Case. $h(a^*) \leqslant \frac{5n}{6}$

By Lemma 3 in [13], we know that for any $0 \leqslant \varepsilon \leqslant 1$,

$$\Pr(m(a) \geqslant rp_a + \varepsilon r) \leqslant \mathbf{e}^{-\varepsilon^2 r/3}, \tag{10}$$

and

$$\Pr(m(a) \leqslant rp_a - \varepsilon r) \leqslant \mathbf{e}^{-\varepsilon^2 r/2}. \tag{11}$$

Let $\delta = \sqrt{\frac{3 \log r}{r}}$. Define $\Sigma_1 = \{a \in \Sigma \mid h(a) \geqslant h(a^*) - 2\delta n\}$ and $\Sigma_2 = \Sigma - \Sigma_1$. Then for any letter $a \in \Sigma_2$, we have

$$p_a + \delta < p_{a^*} - \delta. \tag{12}$$

Intuitively, for any letter a in Σ_1, the difference between $h(a)$ and $h(a^*)$ is small so we can still be satisfied when $a^r = a$. While for any letter a in Σ_2, because $h(a)$ is much smaller than $h(a^*)$, the probability of $a^r = a$ is very small. So, this contributes very little to the left hand side of (9). We prove this formally as

$$\begin{aligned}
&E[h(a^*) - h(a^r)] \\
&\quad = \sum_{a \in \Sigma_1} \Pr(a^r = a) \times (h(a^*) - h(a)) + \sum_{a \in \Sigma_2} \Pr(a^r = a- \times (h(a^*) - h(a)) \\
&\quad \leqslant \left[\sum_{a \in \Sigma_1} \Pr(a^r = a) \times 2\delta n\right] + \left[\sum_{a \in \Sigma_2} \Pr(a^r = a) \times n\right] \\
&\quad \leqslant 2\delta n + n \times \sum_{a \in \Sigma_2} \Pr(m(a) \geqslant m(a^*)) \qquad (\text{since} \quad \Pr(a^r = a) < \Pr(m(a) \geqslant m(a^*))) \\
&\quad \leqslant 2\delta n + n \times \sum_{a \in \Sigma_2} [\Pr(m(a) \geqslant (p_a + \delta)\, r) + \Pr(m(a^*) \leqslant (p_{a^*} - \delta)\, r)] \quad (\text{from (12)}) \\
&\quad \leqslant 2\delta n + |\Sigma|\, n \times (\mathbf{e}^{-r\delta^2/3} + \mathbf{e}^{-r\delta^2/2}) \qquad (\text{from (10) and (11)}) \\
&\quad \leqslant O\left(\sqrt{\frac{\log r}{r}}\right) n = O\left(\sqrt{\frac{\log r}{r}}\right)(n - h(a^*)),
\end{aligned}$$

where the last equality is from $h(a^*) \leqslant \frac{5n}{6}$.

Case. $h(a^*) > \frac{5n}{6}$.

In this case, since letter a^* dominates $a_1, a_2, \ldots, a_n$, it is very unlikely that $m(a^*) < \frac{r}{2}$. So, the probability of $a^r \neq a^*$ is very small. Therefore, the left hand side of (9) is also very small. We prove this formally as follows.

For the purpose of estimation, we examine $r - m(a^*)$. $r - m(a^*)$ can be considered as a sum of r independent $0-1$ variables, each takes 1 with probability $1 - p_{a^*}$. By Chernoff's bound [14, Theorem 4.2], for any $\beta > 0$,

$$\Pr(r - m(a^*) > (1+\beta)(1-p_{a^*})\, r) < \left(\frac{\mathbf{e}^{\beta}}{(1+\beta)^{(1+\beta)}}\right)^{(1-p_{a^*})\, r}. \tag{13}$$

Let $x = \frac{n - h(a^*)}{n} = 1 - p_{a^*}$. (13) becomes

$$\Pr(r - m(a^*) \geqslant (1+\beta)\, xr) \leqslant \left(\frac{\mathbf{e}^{\beta}}{(1+\beta)^{(1+\beta)}}\right)^{xr}. \tag{14}$$

Let $(1+\beta) = \frac{1}{2x}$. Inequality (14) becomes

$$\Pr\left(r - m(a^*) \geqslant \frac{r}{2}\right) \leqslant \left(\frac{(2\mathbf{e}x)^{1/(2x)}}{\mathbf{e}}\right)^{xr} \leqslant \left(\frac{\sqrt{2\mathbf{e}x}}{\mathbf{e}^{x}}\right)^{r}. \tag{15}$$

Since $h(a^*) > \frac{5n}{6}$, $x = \frac{n - h(a^*)}{n} < \frac{1}{6}$. Therefore, $\sqrt{2\mathbf{e}x} < 0.91$ and the following Inequalities are straightforward,

$$\begin{aligned}\left(\frac{\sqrt{2\mathbf{e}x}}{\mathbf{e}^{x}}\right)^{r} &\leqslant (\sqrt{2\mathbf{e}x}\,)^{r} \\ &\leqslant (\sqrt{2\mathbf{e}x}\,)^{r-2} \times 2\mathbf{e}x \\ &\leqslant 0.91^{r-2} \times 2\mathbf{e}x \\ &= O\left(\sqrt{\frac{\log r}{r}}\right) x.\end{aligned} \tag{16}$$

Therefore, in Case 2, we still have

$$\begin{aligned}E[h(a^*) - h(a^r)] &\leqslant \Pr(a^r \neq a^*) \times n \\ &\leqslant \Pr\left(r - m(a^*) \geqslant \frac{r}{2}\right) \times n \\ &\leqslant O\left(\sqrt{\frac{\log r}{r}}\right)(n - h(a^*)),\end{aligned} \tag{17}$$

where the last Inequality is from (15), (16), and the definition of x.

So, in both cases, the lemma holds. ∎

From Lemma 5, Inequality (8) and (7) are true. Therefore, Inequality (3) is true and this proves the theorem. ∎

By using a less intuitive combinatorial method, we can prove the following lemma which is slightly stronger than Lemma 5:

LEMMA 6. *Let $h(\cdot)$, a^*, a^r, and r be defined in Lemma 5. Then*

$$E[h(a^*)-h(a^r)] \leqslant \frac{4A-4}{\sqrt{e}\,(\sqrt{4r+1}-3)}\,(n-h(a^*)). \tag{18}$$

Lemma 6 leads to a stronger version of Theorem 4.

THEOREM 7. *The performance ratio of algorithm consensusPattern is*

$$\left(1+\frac{4A-4}{\sqrt{e}\,(\sqrt{4r+1}-3)}\right)\times c_{opt}.$$

We will use this better ratio in the rest of this paper, while the proof of Lemma 6 is put in Appendix A.

4. A PTAS FOR MAX CONSENSUS PATTERNS

MAX CONSENSUS PATTERNS is the complement of CONSENSUS PATTERNS. It is easy to see that M- cost is at least nL/A and at most nL, where A is the alphabet size. Thus, we can easily prove that the algorithm consensusPattern also gives a PTAS for MAX CONSENSUS PATTERNS.

THEOREM 8. *Given an instance of* MAX CONSENSUS PATTERNS, *suppose its optimal M-cost is c_{mopt}, then for any $r \geqslant 2$, the algorithm consensusPattern outputs a solution with M-cost at least $(1-\frac{4(A-1)^2}{\sqrt{e}\,(\sqrt{4r+1}-3)})\,c_{mopt}$ in time $O((m-L+1)^{r+1}n^{r+1}L)$, where A is the alphabet size.*

Proof. Let c_{hopt} be the optimal H-cost of CONSENSUS PATTERNS and c_{mopt} the optimal M-cost of MAX CONSENSUS PATTERNS. c_{halg} and c_{malg} denote the costs of the solution produced by the algorithm consensusPattern for CONSENSUS PATTERNS and MAX CONSENSUS PATTERNS, respectively. It is easy to see that

$$c_{mopt}+c_{hopt}=nL \qquad \text{and} \qquad c_{malg}+c_{halg}=nL.$$

Moreover, it is easy to see that $c_{mopt} \geqslant nL/A$. Thus,

$$\begin{aligned} c_{mopt}-c_{malg} &= c_{halg}-c_{hopt} \\ &\leqslant \frac{4A-4}{\sqrt{e}\,(\sqrt{4r+1}-3)}\,c_{hopt} \qquad \text{(from Theorem 7)} \\ &= \frac{4A-4}{\sqrt{e}\,(\sqrt{4r+1}-3)}\,(nL-c_{mopt}) \\ &\leqslant \frac{4A-4}{\sqrt{e}\,(\sqrt{4r+1}-3)}\,(A-1)\,c_{mopt}. \qquad \blacksquare \end{aligned}$$

```
Algorithm generalPatterns
Input   $n$ sequences $\{s_1, s_2, \ldots, s_n\} \subseteq \Sigma^m$, integer $L$, $r$, real number
        $p(a)$ for each $a \in \Sigma$ such that $\sum_{a \in \Sigma} p(a) = 1$.
Output  the consensus patterns
1. for every $r$ length-$L$ substrings $u_1, u_2, \ldots, u_r$ (allowing repeats)
   of $s_1, \ldots, s_n$, do
    (a) for $1 \leq j \leq L$ and $a \in \Sigma$ do
            Set $f_j^*(a)$ to be the frequency of letter $a$ in $j$-th column of
            $u_1, \ldots, u_r$. Set $\bar{f}_j(a) = \max\{f_j^*(a), (\frac{\log r}{r})^{\frac{1}{3}}\}$.
    (b) for $i = 1, 2, \ldots, n$ do
            Set $t_i'$ to be the substring of length $L$ in $s_i$ such that
            $\sum_{j=1}^{L} \log \frac{\bar{f}_j(t_i'[j])}{p(t_i'[j])}$ is maximized.
    (c) Compute the I-cost for $t_1', t_2', \ldots, t_n'$.
2. Output the substrings $t_1', t_2', \ldots, t_n'$ in Step 1(c) that maximize the
   I-cost.
```

FIG. 3. Algorithm for the General Consensus Patterns.

Theorem 8 does not hold for $r = 1, 2$. However, the following theorem shows that algorithm consensusPattern has good performance ratio even when $r = 1$. The proof of the theorem is put in Appendix B.

Theorem 9. *When $r = 1$, algorithm consensusPattern has performance ratio $\frac{\sqrt{A}+1}{2}$ for* Max Consensus Patterns, *where A is the size of the alphabet.*

5. APPROXIMATING General Consensus Patterns

In the algorithm for General Consensus Pattern (Fig. 3), again we pick r substrings from the given n strings $s_1, s_2, \ldots, s_n$ and use these r substrings to profile an optimal solution. Then we search for a substring t_i' conforming the profile the most from each s_i. We then prove that starting from at least one group of r substrings, the obtained $t_1', t_2', \ldots, t_n'$ area good suboptimal solution.

Next we briefly introduce the method we use to profile the optimal solution. Suppose $t_1, t_2, \ldots, t_n$ are the substrings in an optimal solution, and $f_j(a)$ is the frequency of letter a in $t_1[j], t_2[j], \ldots, t_n[j]$. Let $u_1, u_2, \ldots, u_r$ be randomly chosen from $t_1, t_2, \ldots, t_n$. We overlay them and denote the frequency of letter a in $u_1[j], u_2[j], \ldots, u_r[j]$ by $f^*_j[a]$. We can expect that at least one group of $u_1, u_2, \ldots, u_r$ is such that $f^*_j(a)$ approximates $f_j(a)$ well for $1 \leqslant j \leqslant L$ and $a \in \Sigma$. However, we still have two barriers to use $f^*_j(a)$ as a profile. First, $\log f^*_j(a)$ does not approximate $\log f_j(a)$ well when $f^*_j(a)$ is near zero. Second, we do not know $t_1, t_2, \ldots, t_n$ from which $u_1, u_2, \ldots, u_r$ are chosen. The first barrier can be solved by using a modified function $\bar{f}_j(a) = \max\{f^*_j(a), (\frac{\log r}{r})^{1/3}\}$ instead of $f^*_j(a)$. The second one can be solved by trying every r length-L substrings $u_1, u_2, \ldots, u_r$ of $s_1, s_2, \ldots, s_n$.

The detailed algorithm is given in Fig. 5. The performance guarantee of the algorithm is proved in Theorem 10.

THEOREM 10. *Let c_{opt} be the maximum I-cost of an optimal soultion. Then Algorithm generalPatterns outputs a solution with I-cost at least $c_{opt} - O((\frac{\log r}{r})^{1/3})$.*

Proof. Let $f_j(a)$ be the frequency of letter a in $t_1[j], t_2[j], \dots, t_n[j]$, where $t_1, t_2, \dots, t_n$ are the substrings in an optimal solution. Let $u_1, u_2, \dots, u_r$ and $\bar{f}_j(a)$ be defined in the algorithm. We first prove the following lemma, which suggests us to profile the optimal solution with $\bar{f}_j(a)$.

LEMMA 11. *There are r integers $1 \leqslant i_1, i_2, \dots, i_r \leqslant n$, such that when $u_j = t_{i_j}$ in Algorithm generalPatterns,*

$$\frac{1}{L}\sum_{j=1}^{L}\sum_{a\in\Sigma} f_j(a)\log\frac{\bar{f}_j(a)}{p(a)} \geqslant \frac{1}{L}\sum_{j=1}^{L}\sum_{a\in\Sigma} f_j(a)\log\frac{f_j(a)}{p(a)} - O\left(\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right). \tag{19}$$

Proof. Let $i_1, i_2, \dots, i_r$ be r independently and randomly chosen numbers from $\{1, 2, \dots, n\}$ and $u_k = t_{i_k}$ for $1 \leqslant k \leqslant r$. Then in Algorithm generalPattern, $f^*_j(a)\, r$ is the number of a's in $t_{i_1}[j], t_{i_2}[j], \dots, t_{i_r}[j]$. Since $i_1, i_2, \dots, i_r$ are randomly chosen, $t_{i_k}[j] = a$ with probability $f_j(a)$ for $1 \leqslant k \leqslant r$. So we have $E[f^*_j(a)\, r] = f_j(a)\, r$. Moreover, by Chernoff's bound [14, Theorem 4.3], for any $0 < \delta < 1$,

$$\Pr(f^*_j(a)\, r < (1-\delta)\, f_j(a)\, r) < \mathrm{e}^{-f_j(a)\, r\delta^2/2}. \tag{20}$$

Meanwhile, $\bar{f}_j(a)$'s are also random variables. To prove the lemma, it is sufficient to prove that

$$E\left[\frac{1}{L}\sum_{j=1}^{L}\sum_{a\in\Sigma}\left(f_j(a)\log\frac{f_j(a)}{p(a)} - f_j(a)\log\frac{\bar{f}_j(a)}{p(a)}\right)\right] \leqslant O\left(\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right). \tag{21}$$

Since

$$f_j(a)\log\frac{f_j(a)}{p(a)} - f_j(a)\log\frac{\bar{f}_j(a)}{p(a)} = f_j(a)\log\frac{f_j(a)/p(a)}{\bar{f}_j(a)/p(a)} = f_j(a)\log\frac{f_j(a)}{\bar{f}_j(a)},$$

to prove Inequality (21), we only need to prove that for any $1 \leqslant j \leqslant L$,

$$E\left[\sum_{a\in\Sigma} f_j(a)\log\frac{f_j(a)}{\bar{f}_j(a)}\right] \leqslant O\left(\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right). \tag{22}$$

Let $\Sigma_1 = \{a \in \Sigma \mid fg_j(a) \geqslant (\frac{\log r}{r})^{1/3}\}$ and $\Sigma_2 = \Sigma - \Sigma_1$. We estimate $E\,[\log\frac{f_j(a)}{\bar{f}_j(a)}]$ for $a \in \Sigma_1$ and $a \in \Sigma_2$ separately.

Let $a \in \Sigma_2$. Then $f_j(a) < (\frac{\log r}{r})^{1/3}$. By the definition of $\bar{f}_j(a)$, we have $\bar{f}_j(a) \geqslant (\frac{\log r}{r})^{1/3} \geqslant f_j(a)$. Therefore,

$$E\left[\log\frac{f_j(a)}{\bar{f}_j(a)}\right] \leqslant 0 \qquad \text{for any} \quad a \in \Sigma_2. \tag{23}$$

Let $a \in \Sigma_1$. Setting $\delta = (\frac{\log r}{r})^{1/3}$ and substituting $f^*_j(a)$ with $(\frac{\log r}{r})^{1/3}$ in Formula (20), we get

$$\Pr\left(f^*_j(a) < \left(1 - \left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right) f_j(a)\right) < \mathrm{e}^{-(\log r)/2} = \frac{1}{\sqrt{r}}. \tag{24}$$

Let $p_0 = \Pr(\bar{f}_j(a) < (1 - (\frac{\log r}{r})^{\frac{1}{3}})\, f_j(a))$. Combining (24) with the fact that $\bar{f}_j(a) \geqslant f^*_j(a)$, we know that $p_0 < \frac{1}{\sqrt{r}}$. Finally, we have

$$\begin{aligned}
E\left[\log \frac{f_j(a)}{\bar{f}_j(a)}\right] &\leqslant p_0 \times \max\left\{\log \frac{f_j(a)}{\bar{f}_j(a)} \mid \bar{f}_j(a) < \left(1 - \left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right) f_j(a)\right\} \\
&\quad + (1 - p_0) \times \max\left\{\log \frac{f_j(a)}{\bar{f}_j(a)} \mid \bar{f}_j(a) \geqslant \left(1 - \left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right) f_j(a)\right\} \\
&\leqslant p_0 \times \log \frac{1}{\left(\frac{\log r}{r}\right)^{\frac{1}{3}}} + (1 - p_0) \times \left(-\log\left(1 - \left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right)\right) \\
&\leqslant \frac{1}{\sqrt{r}} \times \frac{1}{3} \times \log\left(\frac{r}{\log r}\right) - \log\left(1 - \left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right) \\
&\leqslant O\left(\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right) \qquad \text{for any} \quad a \in \Sigma_1.
\end{aligned} \tag{25}$$

Combining (25) and (23), we know that

$$E\left[\sum_{a \in \Sigma} f_j(a) \log \frac{f_j(a)}{\bar{f}_j(a)}\right] \leqslant \sum_{a \in \Sigma} f_j(a)\, E\left[\log \frac{f_j(a)}{\bar{f}_j(a)}\right] \leqslant O\left(\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right).$$

That is, (22) holds. Therefore, there is at least one group of $1 \leqslant i_1, i_2, \ldots, i_r \leqslant n$ satisfying (19). The lemma is proved. ▌

Let $i_1, i_2, \ldots, i_r$ be the integers satisfying Lemma 11. Let $u_j = t_{i_j}$ and $t'_1, t'_2, \ldots, t'_n$ be in Step 1(b) of the algorithm. Let $f'_j(a)$ be the frequency of letter a in the jth column of $t'_1, t'_2, \ldots, t'_n$. Then the I-cost of $t'_1, t'_2, \ldots, t'_n$ is $\frac{1}{L} \sum_{j=1}^{L} \sum_{a \in \Sigma} f'_j(a) \log f'_j(a)/p(a)$. In order to use Lemma 11, we need to establish the relation between $\frac{1}{L} \sum_{j=1}^{L} \sum_{a \in \Sigma} f'_j(a) \log f'_j(a)/p(a)$ and $\frac{1}{L} \sum_{j=1}^{L} \sum_{a \in \Sigma} f_j(a) \log \bar{f}_j(a)/p(a)$. Clearly, the relation is established by the following two claims:

Claim 12.

$$\sum_{a \in \Sigma} f'_j(a) \log \frac{f'_j(a)}{p(a)} \geqslant \sum_{a \in \Sigma} f'_j(a) \log \frac{\bar{f}_j(a)}{p(a)} - O\left(\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right). \tag{26}$$

Claim 13.

$$\sum_{j=1}^{L} \sum_{a \in \Sigma} f'_j(a) \log \frac{\bar{f}_j(a)}{p(a)} \geqslant \sum_{j=1}^{L} \sum_{a \in \Sigma} f_j(a) \log \frac{\bar{f}_j(a)}{p(a)}. \tag{27}$$

Proof of Claim 12. (26) is equivalent to

$$\sum_{a \in \Sigma} f'_j(a) \log \frac{f'_j(a)}{\bar{f}_j(a)} \geqslant -O\left(\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right). \tag{28}$$

Since $\log\,(1+|\Sigma|(\frac{\log r}{r})^{\frac{1}{3}})$, it is sufficient to prove that

$$\sum_{a \in \Sigma} \bar{f}'_j(a) \log \frac{f'_j(a)}{\bar{f}'_j(a)} \geqslant -\log\left(1+|\Sigma|\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right). \tag{29}$$

Let $f = \sum_{a \in \Sigma} \bar{f}_j(a)$. Then $\sum_{a \in \Sigma} \bar{f}_j(a)/f = 1$. By the entropy theory,

$$\sum_{a \in \Sigma} f'_j(a) \log \frac{f'_j(a)}{\bar{f}_j(a)/f} \geqslant 0.$$

That is,

$$\sum_{a \in \Sigma} f'_j(a)\left(\log \frac{f'_j(a)}{\bar{f}_j(a)} + \log f\right) \geqslant 0.$$

So,

$$\sum_{a \in \Sigma} f'_j(a) \log \frac{f'_j(a)}{\bar{f}_j(a)} \geqslant -\sum_{a \in \Sigma} f'_j(a) \log f = -\log f.$$

By the definition of f and $\bar{f}_j(a)$, $1 \leqslant f \leqslant 1+|\Sigma|\,(\frac{\log r}{r})^{1/3}$. Thus Formula (29) is correct; hence the claim. ▮

Proof of Claim 13. Note that,

$$\sum_{i=1}^{n} \log \frac{\bar{f}_j(t'_i[j])}{p(t'_i[j])} = \sum_{a \in \Sigma} \sum_{\substack{t'_i[j]=a \\ 1 \leqslant i \leqslant n}} \log \frac{\bar{f}_j(t'_i[j])}{p(t'_i[j])} = \sum_{a \in \Sigma} (f'_j(a)\, n) \log \frac{\bar{f}_j(a)}{p(a)}. \tag{30}$$

Thus,

$$\sum_{j=1}^{L} \sum_{a \in \Sigma} f'_j(a) \log \frac{\bar{f}_j(a)}{p(a)} = \frac{1}{n} \sum_{j=1}^{L} \sum_{i=1}^{n} \log \frac{\bar{f}_j(t'_i[j])}{p(t'_i[j])} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{L} \log \frac{\bar{f}_j(t'_i[j])}{p(t'_i[j])}.$$

For the same reason, we have

$$\sum_{j=1}^{L} \sum_{a \in \Sigma} f_j(a) \log \frac{\bar{f}_j(a)}{p(a)} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{L} \log \frac{\bar{f}_j(t_i[j])}{p(t_i[j])}.$$

From the choice of t'_i in the algorithm,

$$\sum_{i=1}^{n} \sum_{j=1}^{L} \log \frac{\bar{f}_j(t'_i[j])}{p(t'_i[j])} \geqslant \sum_{i=1}^{n} \sum_{j=1}^{L} \log \frac{\bar{f}_j(t_i[j])}{p(t_i[j])}. \tag{31}$$

Thus, the claim is proved. ▮

Thus, from Lemma 11, Claim 12, and Claim 13, we know that there is a set of $u_1, u_2, ..., u_r$ such that

$$\frac{1}{L}\sum_{j=1}^{L}\sum_{a\in\Sigma} f'_j(a)\log\frac{f'_j(a)}{p(a)} \geqslant \frac{1}{L}\sum_{j=1}^{L}\sum_{a\in\Sigma} f_j(a)\log\frac{f_j(a)}{p(a)} - O\left(\frac{\log r}{r}\right)^{\frac{1}{3}}$$
$$= c_{opt} - O\left(\frac{\log r}{r}\right)^{\frac{1}{3}}.$$

Since we try every possibility of $u_1, u_2, ..., u_r$ in the algorithm, the theorem is true. ∎

Note that when $p(a)$'s are equal for all letters in Σ, then to maximize the score

$$\frac{1}{L}\sum_{j=1}^{L}\sum_{a\in\Sigma} f_j(a)\log\frac{f_j(a)}{p(a)}$$

is equivalent to maximize the score

$$\sum_{j=1}^{L}\sum_{a\in\Sigma} h_j(a)\log h_j(a),$$

where $h_j(a)$ is the number of the appearances of letter a as the jth letter of the patterns. For this special case, we have the following corollary:

COROLLARY 14. *With input $p(a)=1/|\Sigma|$ for every $a\in\Sigma$, Algorithm general-Patterns is a PTAS to* GENERAL CONSENSUS PATTERNS *with score $\sum_{j=1}^{L}\sum_{a\in\Sigma} h_j(a)\log h_j(a)$ to be maximized.*

Proof. Suppose $t_1, t_2, ..., t_n$ are the consensus patterns that maximize the score $\sum_{j=1}^{L}\sum_{a\in\Sigma} h_j(a)\log h_j(a)$. Note that the proof of Theorem 10 does not need to assume that $p(a)$ is the frequency of letter a in all the strings, except that $\sum_{a\in\Sigma} p(a)=1$. So, by Theorem 10, we know that the algorithm outputs consensus patterns t'_i's such that

$$\frac{1}{L}\sum_{j=1}^{L}\sum_{a\in\Sigma}\frac{h'_j(a)}{n}\log\frac{h'_j(a)/n}{1/|\Sigma|} \geqslant \frac{1}{L}\sum_{j=1}^{L}\sum_{a\in\Sigma}\frac{h_j(a)}{n}\log\frac{h_j(a)/n}{1/|\Sigma|} - O\left(\frac{\log r}{r}\right)^{\frac{1}{3}},$$

where $h'_j(a)$ is the number of the appearances of letter a as the jth letter of t'_i's. Therefore,

$$\sum_{j=1}^{L}\sum_{a\in\Sigma} h'_j(a)\log h'_j(a) \geqslant \sum_{j=1}^{L}\sum_{a\in\Sigma} h_j(a)\log h_j(a) - O\left(\bar{f}_j(a)\,\frac{\log r}{r}\right)^{\frac{1}{3}} nL. \qquad (32)$$

It is easy to verify that when $n \geqslant \mathbf{e}\,|\Sigma|$, then

$$\sum_{j=1}^{L}\sum_{a\in\Sigma} h_j(a)\log h_j(a) \geqslant nL.$$

Thus, by Formula (32), we know that

$$\sum_{j=1}^{L} \sum_{a \in \Sigma} h'_j(a) \log h'_j(a) \geqslant \left(1 - O\left(\frac{\log r}{r}\right)^{\frac{1}{3}}\right) \sum_{j=1}^{L} \sum_{a \in \Sigma} h_j(a) \log h_j(a).$$

Thus, we proved the corollary. ▮

Remark. Instead of looking for one pattern from every string, all above theorems (Theorem 4, 7–10, and Corollary 14) generalize to the case when we are looking for k distinct patterns from the n given strings, allowing several patterns to come from the same string and some strings may contribute no pattern.

6. CONSENSUS ALIGNMENT WITH CONSTANT GAPS

Consensus alignment is a very important model of the multiple sequence alignment problem [6]. It is well known that the consensus multiple alignment problem is NP-hard [22]. The best current known approximation algorithm has the performance ratio $2-o(1)$ [6]. For pairwise alignment, gap penalties are imposed to reduce the number of gaps in the alignment in literatures. An interesting variant is to allow no more than c gaps in each of the two sequences for a constant c. We call this the *c-alignment*. Accordingly, the consensus multiple alignment has a modified version, the *multiple c-alignment*, that is to find a median sequence s for a set of sequences $\{s_1, s_2, ..., s_n\}$ minimizing $\sum_{i=1}^{n} d_c(s_i, s)$, where $d_c(s_i, s)$ is the c-alignment cost of s_i and s. We consider the simplest scoring scheme: a match costs 0 and a mismatch costs 1.

Once the median sequence is obtained, one can construct a multiple alignment for the n given sequences based on the pairwise c-alignments of s_i and s. (See [6] for reference on how to build a multiple alignment from pairwise alignments.) It should be emphasized that in the multiple alignment constructed above, each sequence can have unbounded number of gaps, though there are at most c gaps in the pairwise alignments between given sequences s_i's and the median sequence s. We show that consensus c-alignment remains NP-hard and give a PTAS for it.

THEOREM 15. *Consensus c-alignment is NP-hard if the size of alphabet is four.*

Proof. The reduction is again from Max Cut-3. The constructed sequences are the same as in the proof of Theorem 1. Here we do not have to add ♯'s at the left ends of strings in $X_0 \cup X_2$ such that every sequence is of the same length since for the multiple sequence alignment problem the lengths of the given sequences can be different.

Similar to Claim 2, we have

CLAIM 16. *The median sequence in an optimal solution of consensus c-alignment* $(X_0 \cup X_1 \cup X_2)$ *should be in the form*

$$b^{*5}\,(Dx_1\, D^{*4})(Dx_2\, D^{*4}) \cdots (Dx_{n-1}\, d^{*4})\, Dx_n\, D,$$

```
 *****######0***######1***######1111######0000######0***######1***######  s_i
b*****######1111######****######1111######****######0000######            median
                                   (a)

 *****######0***######1***######1111######0000######0***######1***######  s_i
b*****          ######0000######****######0000######****######1111######  median
                                   (b)
```

FIG. 4. (a) v_i is in V_1. (b) v_i is in V_0.

where b is any binary number of $\lceil \log n^2 \rceil$ *bits; there are n blocks of* x_i*'s which are either* 0^{k+1} *or* 1^{k+1}.

By Claim 16, the cost $c(X_1 \cup X_2)$ contributed by the sequences in $X_1 \cup X_2$ is irrelevant to the selections of x_i's.

Suppose that there is a partition (V_0, V_1) of V, which cuts c edges. The median sequence can be constructed as

$$b^{*5}\,(Dx_1\, D^{*4})(Dx_2\, D^{*4}) \cdots (Dx_{n-1}\, D^{*4})(Dx_n\, D),$$

where there are n blocks of x_i, x_i is 1^4 if v_i is in V_1, and 0^4 otherwise.

For each i, if x_i of the median sequence is 1^4 (i.e., $v_i \in V_1$), we align s_i with the median sequence as in Fig. 4a, i.e., the right end delimiters of s_i and $y_{i,n}$ are matched with spaces, and if x_i of the median sequence is 0^{k+1} (i.e., $v_i \in V_0$), we align s_i with the median sequence as in Fig. 4b; i.e., the left end delimiters of s_i and x_1 are matched with spaces. Note that each segment $z_{j,i} = D0^{*k}\, D1^{*k}$ for $j \neq i$ contributes cost by either 4 or 5. If $v_i \in V_1$, then for each v_j adjacent to v_i, the segment $z_{j,i}$ of s_j, which is of the form $D0^{*3}\, D1^{*3}$, will contribute 5 toward the cost if and only if $v_j \in V_1$, as in Fig. 4a. Similarly, if $v_i \in V_0$, then for each v_j adjacent to v_i, the segment $z_{j,i}$ of s_j will contribute 5 toward the cost if and only if $v_j \in V_0$, as in Fig. 4b. That is, all the edges that are not cut by the partition are counted once more here.

Let $c(v)$ denote the number of edges incident upon v that are cut by the partition. For each $v_i \in V$, s_i contributes $m_i - c(v_i)$ to the cost, where m_i is a number purely determined by n and the degree of v_i in G. thus the total cost of the alignment is $c(X_1 \cup X_2) + \sum_{i=1}^{n} m_i - 2c$.

Conversely, if there is an alignment of cost $c(X_1 \cup X_2) + \sum_{i=1}^{n} m_i - 2c$, one can easily construct a partition of G that cuts c edges by looking at the $0-1$ assignment to x_i's in the median sequence. ▌

As an application of our algorithm consensusPattern (and its analysis), Fig. 5 describes an algorithm which outputs a median sequence s with total cost less than $1+\varepsilon$ times the minimum cost of the consensus c-alignment.

THEOREM 17. *Algorithm consensusAlign outputs a median sequences such that*

$$\sum_{i=1}^{n} d_E(s_i, s) \leqslant \left(1 + \frac{4A}{\sqrt{e}\,(\sqrt{4r+1}-3)}\right) \sum_{i=1}^{n} d_c(s_i, s_{opt})$$

in $O((crm^2)^{cr^2+2}\, n^{r+1}/m^2)$ *time, where* $A = |\Sigma|$, *r is the parameter used in the algorithm, m is the length of the given sequences, and* s_{opt} *is the optimal median sequence.*

```
Algorithm consensusAlign
Input  $\mathcal{S} = \{s_1, \ldots, s_n\}$, each $s_i$ has length $m$.
Output a median sequence $s$ of length $m$.
1. for any $s_{i_1}, s_{i_2}, \ldots, s_{i_r} \in \mathcal{S}$ do
      for any possible alignment of $s_{i_1}, s_{i_2}, \ldots, s_{i_r}$ such that the number of gaps inserted in every sequence is no more than $cr$ and the length of each gap is no more than $crm$ do
         (a) Find the column-wise majority letters to form the median sequence $s$;
         (b) Calculate $\sum_{i=1}^{n} d_E(s, s_i)$.
2. Output an $s$ such that $\sum_{i=1}^{n} d_E(s, s_i)$ is minimized among all $s$'s obtained in step 1(a).
```

FIG. 5. PTAS for consensus c-alignment.

Proof. It is easy to see that the number of the possible alignments for fixed r sequences in step 1 is no more than $(m^{cr}(crm)^{cr})^r = (crm^2)^{cr^2}$. Steps 1(a) and 1(b) can be done in $O(c^2r^2m^2n)$ time and there are n^r groups of r sequences. Thus, the time complexity of the algorithm is $O((crm^2)^{cr^2+2}\, n^{r+1}/m^2)$.

Now we derive the performance ratio of the algorithm. Let s_{opt} be an optimal median sequence for the given n sequences $s_1, s_2, \ldots, s_n$. Consider the multiple alignment $\mathcal{M}$ for the n sequences obtained from the pairwise c-alignments of s_i's and s_{opt}. Let L be the length of the alignment $\mathcal{M}$. Treating the spaces in the multiple alignment $\mathcal{M}$ as new letters, we get n strings $s'_1, s'_2, \ldots, s'_n$ of length L over alphabet $\Sigma \cup \{space\}$ from $\mathcal{M}$, corresponding to $s_1, s_2, \ldots, s_n$. Accordingly, denote s'_{opt} as the string of length L obtained from s_{opt} by padding spaces according to the multiple alignment $\mathcal{M}$.

In the proof of Theorem 4, we know that there are $1 \leqslant i_1, i_2, \ldots, i_r \leqslant n$, such that if s^* is a column-wise majority string of $s'_{i_1}, s'_{i_2}, \ldots, s'_{i_r}$, then

$$\begin{aligned}\sum_{i=1}^{n} d_H(s^*, s'_i) &\leqslant \left(1 + \frac{4A}{\sqrt{e}\,(\sqrt{4r+1}-3)}\right) \sum_{i=1}^{n} d_H(s'_{opt}, s'_i)\\ &= \left(1 + \frac{4A}{\sqrt{e}\,(\sqrt{4r+1}-3)}\right) \sum_{i=1}^{n} d_c(s_{opt}, s_i).\end{aligned}$$

It is easy to see that the induced multiple alignment of $s_{i_1}, s_{i_2}, \ldots, s_{i_r}$ from $\mathcal{M}$ has at most cr gaps inserted in every sequence and each gap is of length no more than crm. Hence, algorithm consensusAlign tries this alignment once in step 1. Thus, it finds a median sequence s with

$$\begin{aligned}\sum_{i=1}^{n} d_E(s, s_i) &\leqslant \sum_{i=1}^{n} d_H(s^*, s'_i)\\ &\leqslant \left(1 + \frac{4A}{\sqrt{e}\,(\sqrt{4r+1}-3)}\right) \sum_{i=1}^{n} d_c(s_{opt}, s_i). \quad \blacksquare\end{aligned}$$

7. CONCLUDING REMARKS

One main problem of our algorithms is the efficiency. For practical purposes, we have implemented our algorithms with some heuristic strategies in a software tool COPIA [11]. The software is accessable at http:/dna.cs.ucsb.edu/copia/copia_submit.html.

APPENDIX A—PROOF OF LEMMA 6

To obtain better approximation ratio in terms of r. We first need the following technical lemma.

LEMMA 18. *Let* $g(x, y)=\frac{1}{1-x}(x-y)(1-x-y+2\sqrt{xy})^r$. *If* $r \geqslant 3$, $0 \leqslant y < x$ *and* $x+y \leqslant 1$, *then* $g(x, y) < \frac{4}{\sqrt{e}\,(\sqrt{4r+1}-3)}$.

Proof. It is easy to verify that if $x+y \leqslant 1$, then the following equations

$$\begin{cases} \sqrt{u}-\sqrt{v}=\sqrt{x}-\sqrt{y} \\ u+v=1 \end{cases}$$

has a solution $u=x_0$, $v_{=y_0}$ such that $x_0 \geqslant x$ and $y_0 \geqslant y$. Since

$$g(x, y)=\frac{1}{1-x}(\sqrt{x}+\sqrt{y})(\sqrt{x}-\sqrt{y})(1-(\sqrt{x}-\sqrt{y})^2)^r,$$

we have

$$g(x, y) \leqslant g(x_0, y_0)=g(x_0, 1-x_0). \tag{33}$$

Let $f(x)=g(x, 1-x)=\frac{(2x-1)(2\sqrt{x(1-x)})^r}{1-x}$. Now we show that when $0<x \leqslant 1$,

$$f(x) \leqslant \frac{4}{\sqrt{e}\,(\sqrt{4r+1}-3)}. \tag{34}$$

Since

$$f'(x)=\frac{1}{2}\frac{(-4rx^2+4rx+2x-r)(2\sqrt{x(1-x)})^r}{(1-x)^2\,x}$$

by solving $f'(x)=0$, we get four possible points where $f(x)$ may take its maximum value:

$$x=0,\ 1,\ \frac{1+2r+\sqrt{1+4r}}{4r}, \qquad \text{or} \qquad \frac{1+2r-\sqrt{1+4r}}{4r}.$$

Let $x_1=\frac{1+2r+\sqrt{1+4r}}{4r}$. It is easy to verify that $f(x)$ takes its maximum value when $x=x_1$. That is,

$$
\begin{aligned}
f(x) &\leqslant \frac{2x-1}{1-x_1}\left(2\sqrt{x_1(1-x_1)}\,\right)^r \\
&= \frac{4}{\sqrt{4r+1}-3}\left(1-\left(\frac{1+\sqrt{1+4r}}{2r}\right)^2\right)^{r/2} \\
&\leqslant \frac{4}{\sqrt{4r+1}-3}\left(1-\frac{1}{r}\right)^{r/2} < \frac{4}{\sqrt{4r+1}-3}\times\frac{1}{\sqrt{e}}
\end{aligned}
$$

Thus, we have proved Inequality (34), and from Formula (33), the lemma follows. ▮

Proof of Lemma 6. To simplify the proof, we first introduce two index sets, $\mathscr{I}_a$ and $\mathscr{L}_a$. For every $a \in \Sigma = \{1, ..., A\}$, let l_a denote the number of a's in an r-element set, and let

$$
\begin{aligned}
\mathscr{I}_a &= \{(i_1, i_2, ..., i_r) \mid a \text{ is a majority of } a_{i_1}, a_{i_2}, ..., a_{i_r}\} \\
\mathscr{L}_a &= \{(l_1, l_2, ..., l_A) \mid l_1 + l_2 + \cdots + l_A = r \\
&\qquad \text{and } l_b \leqslant l_a \text{ for any } b \in \Sigma\}.
\end{aligned}
$$

Let $x_a = \frac{h(a)}{n}$. Recall that $\Sigma = \{1, 2, ..., A\}$. Then the left part of Inequality (18) is

$$
\begin{aligned}
n^{-r} &\sum_{1 \leqslant i_1, i_2, ..., i_r \leqslant n} [h(a^*) - h(a_{(i_1, i_2, ..., i_r)})] \\
&\leqslant n^{-r} \sum_{a=1}^{A} \sum_{(i_1, i_2, ..., i_r) \in \mathscr{I}_a} [h(a^*) - h(a)] \\
&= n^{-r} \sum_{a=1}^{A} [h(a^*) - h(a)]\, |\mathscr{I}_a|. \qquad (35)
\end{aligned}
$$

Thus, to upper bound the left part of Inequality (18), we need to upper bound $|\mathscr{I}_a|$.

Let $i_1, i_2, ..., i_r) \in \mathscr{I}_a$. For each $b \in \Sigma$, let l_b be the number of j such that $1 \leqslant j \leqslant r$ and $a_{i_j} = b$. By the definition of $\mathscr{I}_a$, we have $l_b \leqslant l_a$ for any $b \in \Sigma$. That is, $(l_1, l_2, ..., l_A) \in \mathscr{L}_a$. Conversly, if $l_1, l_2, ..., l_A \in \mathscr{L}_a$, then many $(i_1, i_2, ..., i_r)$ such that there are l_b indices $1 \leqslant j \leqslant r$ satisfying $a_{i_j} = b$ is in $\mathscr{I}_a$. the number of such $(i_1, i_2, ..., i_r)$ is $\frac{r!}{l_1! l_2! ... l_A!}(h(1))^{l_1}(h(2))^{l_2} \cdots (h(A))^{l_A}$. Therefore, we can bound the size of set $\mathscr{I}_a$ as follows.

$$
\begin{aligned}
|\mathscr{I}_a| &= \sum_{(l_1, l_2, ..., l_A) \in \mathscr{L}_a} \frac{r!}{l_1! l_2! \cdots l_A!}(h(1))^{l_1}(h(2))^{l_2} \cdots (h(A))^{l_A} \\
&= n^r \sum_{(l_1, l_2, ..., l_A) \in \mathscr{L}_a} \frac{r!}{l_1! l_2! \cdots l_A!} x_1^{l_1} x_2^{l_2} \cdots x_A^{l_A}. \qquad (36)
\end{aligned}
$$

For any $(l_1, l_2, ..., l_A) \in \mathscr{L}_a$, since $l_{a^*} \leqslant l_a$, and $x_{a^*} \geqslant x_a$, we have $x_a^{l_a} x_{a^*}^{l_{a^*}} \leqslant (\sqrt{x_a x_{a^*}}\,)^{l_a} (\sqrt{x_a x_{a^*}}\,)^{l_{a^*}}$. Thus, by setting $y_a = y_{a^*} = \sqrt{x_a x_{a^*}}$, and $y_1 = x_i$ for $i \neq a$ and $i \neq a^*$, we know that

$$\begin{aligned}
\sum_{(l_1, l_2, \ldots, l_A) \in \mathscr{L}_a} & \frac{r!}{l_1! l_2! \cdots l_A!} x_1^{l_1} x_2^{l_2} \cdots x_A^{l_A} \\
&\leqslant \sum_{(l_1, l_2, \ldots, l_A) \in \mathscr{L}_a} \frac{r!}{l_1! l_2! \cdots l_A!} y_1^{l_1} y_2^{l_2} \cdots y_A^{l_A} \\
&\leqslant \sum_{l_1 + l_2 + \cdots + l_A = r} \frac{r!}{l_1! l_2! \cdots l_A!} y_1^{l_1} y_2^{l_2} \cdots y_A^{L_A} \\
&= (y_1 + y_2 + \cdots + y_A)^r \\
&= (1 - x_{a^*} - x_a + 2\sqrt{x_{a^*} x_a}\,)^r.
\end{aligned}$$

By Formula (36),

$$|\mathscr{I}_a| \leqslant n^r (1 - x_{a^*} - x_a + 2\sqrt{x_{a^*} x_a}\,)^r. \tag{37}$$

Now we can upper bound the left part of Inequality (18). Consider Formula (35). If $a = a^*$, then $[h(a^*) - h(a)]\,|\mathscr{I}_a| = 0$. If $a \neq a^*$, from Inequality (37),

$$\begin{aligned}
[h(a^*) - h(a)]\,|\mathscr{I}_a| &\leqslant n^{r+1} (x_{a^*} - x_a)(1 - x_{a^*} - x_a + 2\sqrt{x_{a^*} x_a}\,)^r \\
&< \frac{4}{\sqrt{e}\,(\sqrt{4r+1} - 3)} (1 - x_{a^*})\, n^{r+1} \qquad (38) \\
&= \frac{4}{\sqrt{e}\,(\sqrt{4r+1} - 3)} (n - h(a^*))\, n^r, \qquad (39)
\end{aligned}$$

where Inequality (38) is by Lemma 18. Combining Formula (35) and (39), the proof is complete. ▮

APPENDIX B

Proof of Theorem 9. Suppose the score of the optimal solution is c_{opt}, and $s_1, s_2, \ldots, s_n$ are the consensus patterns in an optimal solution. Let s^* be a column-wise majority of $s_1, s_2, \ldots, s_n$. For any character $a \in \Sigma$, let $h_j(a)$ be the number of s_i's such that $s_i[j] = a$. Then for any string s_i, the score of the solution obtained from s_i at step 1(b) is at least $\sum_{j=1}^{L} h_j(s_i[j])$. To prove the theorem, it is sufficient to prove that there is an i such that

$$\sum_{j=1}^{L} h_j(s_i[j]) \geqslant \frac{2}{\sqrt{A+1}} c_{opt}.$$

Therefore, it is sufficient to prove that

$$\sum_{i=1}^{n} \sum_{j=1}^{L} h_j(s_i[j]) \geqslant \frac{2}{\sqrt{A+1}} n c_{opt}.$$

$$\sum_{i=1}^{n} h_j(s_i[j]) \geqslant \frac{2}{\sqrt{A+1}} n h_j(s^*[j]). \tag{40}$$

For any, $a, b \in \Sigma$, let $\chi(a, b) = 0$ if $a \neq b$ and 1 if $a = b$. Then

$$\begin{aligned}\sum_{i=1}^{n} h_j(s_i[j]) &= \sum_{i=1}^{n} \sum_{k=1}^{n} \chi(s_i[j], s_k[j]) \\ &= \sum_{a=1}^{A} \sum_{i=1}^{n} \sum_{k=1}^{n} \chi(s_i[j], a) \cdot \chi(s_k[j], a) \\ &= \sum_{a=1}^{A} (h_j(a))^2. \end{aligned} \tag{41}$$

Because $f(x) = x^2$ is a convex function, and $\sum_{a \neq s^*[j]} h_j(a) = n - h_j(s^*[j])$, we know that

$$\sum_{a \neq s^*[j]} (h_j(a))^2 \geqslant \frac{(n - h_j(s^*[j]))^2}{A-1}.$$

Combining with Formula (41), we have

$$\sum_{i=1}^{n} h_j(s_i[j]) \geqslant (h_j(s^*[j]))^2 + \frac{(n - h_j(s^*[j]))^2}{A-1}. \tag{42}$$

Moreover, we have

Claim 19. For any x such that $0 \leqslant x \leqslant n$, $x^2 + \frac{(n-x)^2}{A-1} \geqslant \frac{2}{\sqrt{A}+1} nx$.

Proof. For any $0 \leqslant \xi \leqslant 1$, we have

$$\xi + \frac{1}{A-1}(1-\xi)\left(\frac{1}{\xi} - 1\right) = \frac{1}{A-1}\left(\frac{1}{\xi} + A\xi - 2\right) \geqslant \frac{1}{A-1}(2\sqrt{A} - 2) = \frac{2}{\sqrt{A}+1}.$$

Let $\xi = \frac{x}{n}$, and multiplying nx to both sides of the above inequality, the claim is proved. ▮

Formula (40), thus the theorem, follows from Claim 19 and Formula (42). ▮

ACKNOWLEDGMENTS

We thank Paul Kearney, Tao Jiang, Kevin Lanctot, Joe Wang, Zhiwei Wang, and Louxin Zhang for discussions and suggestions on related topics. We also thank two anonymous referees for their great efforts in helping us to improve the presentation.

REFERENCES

1. V. Bafna, E. Lawler, and P. Pevzner, Approximation algorithms for multiple sequence alignment, *in* "Proc. 8th Ann. Combinatorial Pattern Matching Conf. (CPM'94)," pp. 43–53, 1994.
2. Q. Chan, G. Hertz, and G. Stormo, Matrix search 1.0: a computer program that scans DNA sequences for transcriptional elements using a database of weight matrices, *Comput. Appl. Biosci.* (1995), 563–566.

3. Y. M. Fraenkel, Y. Mandel, D. Friedberg, and H. Margalit, Identification of common motifs in unaligned DNA sequences: application to Escherichia coli Lrp regulon, *Comput. Appl. Biosci.* (1995), 379–387.
4. M. Garey and D. Johnson, "Computers and Intractability, a Guild to the Theory of NP-Completeness," Freeman, New York, 1979.
5. D. Gusfield, Efficient methods for multiple sequence alignment with guaranteed error bounds, *Bull. Math. Biol.* **30** (1993), 141–154.
6. D. Gusfield, "Algorithms on Strings, Trees, and Sequences," Cambridge Univ. Press, Cambridge, UK, 1997.
7. G. Hertz and G. Stormo, Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps, *in* "Proc. 3rd Int'l Conf. Bioinformatics and Genome Research" (Lim and Cantor, Eds.) pp. 201–216, World Scientific, Singapore, 1995.
8. R. Karp, Reducibility among combinatorial problems, *in* "Complexity of Computer Computation" (R. E. Miller and J. W. Thatcher, Eds.) pp. 85–103, Plenum Press, New York, 1972.
9. Y. V. Kondrakhin, A. E. Kel, N. A. Kolchanov, A. G. Romashchenko, and L. Milanesi, Eukaryotic promoter recognition by binding sites for transcription factors, *Comput. Appl. Biosci.* (1995), 477–488.
10. C. Lawrence and A. Reilly, An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences, *Proteins* **7** (1990), 41–51.
11. C. Liang, M. Li, and B. Ma, COPIA: A new software tool for finding consensus patterns in unaligned sequences, manuscript, 2001.
12. M. Li, B. Ma, and L. Wang, On the closest string and substring problems, to appear in *Journal of the ACM*, 2002.
13. B. Ma, A polynomial time approximation scheme for the closest substring problem, *in* "CPM '2000," Lecture Notes in Computer Science, Vol. 1848, pp. 99–107, Springer-Verlag, Berlin/New York, 2000.
14. R. Motwani and P. Raghavan, "Randomized Algorithms," Cambridge Univ. Press, Cambridge, UK, 1995.
15. P. Pevzner, Multiple alignment, communication cost, and graph matching, *SIAM J. Appl. Math.* **52** (1992), 1763–1779.
16. J. Posfai, A. Bhagwat, G. Posfai, and R. Roberts, Predictive motifs derived from cytosine methyltransferases, *Nucl. Acids Res.* **17** (1989), 2421–2435.
17. D. S. Prestridge, SIGNAL SCAN 4.0: additional databases and sequence formats, *Comput. Appl. Biosci.* (1996), 157–160.
18. M. A. Roytberg, A search for common patterns in many sequences, *Comput. Appl. Biosci.* (1992), 57–64.
19. G. D. Schuler, S. F. Altschul, and D. J. Lipman, A workbench for multiple alignment construction and analysis, *Proteins: Structure, Function Genetics* **9** (1991), 180–190.
20. G. Stormo, Consensus patterns in DNA, *in* "Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences, Methods in Enzymology" (R. F. Doolittle, Ed.), Vol. 183, pp. 211–221, 1990.
21. G. Stormo and G. W. Hartzell III, Identifying protein-binding sites from unaligned DNA fragments, *Proc. Natl. Acad. Sci. USA* **88** (1991), 5699–5703.
22. L. Wang and T. Jiang, On the complexity of multiple sequence alignment, *J. Comp. Biol.* **1** (1994), 337–348.
23. M. Waterman, Multiple sequence alignment by consensus, *Nucl. Acids Res.* **14** (1986), 9095–9102.
24. M. Waterman, R. Arratia, and D. Galas, Pattern recognition in several sequences: consensus and alignment, *Bull. Math. Biol.* **46** (1984), 515–527.
25. M. Waterman and M. Perlwitz, Line geometries for sequence comparisons, *Bull. Math. Biol.* **46** (1984), 567–577.
26. F. Wolfertstetter, K. Frech, G. Herrmann, and T. Werner, *Comput. Appl. Biosci.* (1996), 71–80.