Consensus Optimizing Both Distance Sum and Radius

Amihood Amir[1], Gad M. Landau[⋆2], Joong Chae Na[⋆⋆3], Heejin Park[⋆⋆⋆4], Kunsoo Park[5], and Jeong Seop Sim[6]

[1] Bar-Ilan University, 52900 Ramat-Gan, Israel
[2] University of Haifa, Haifa 31905, Israel, and Polytechnic Institute of NYU, NY 11201-3840, USA
[3] Sejong University, Seoul 143-747, South Korea
[4] Hanyang University, Seoul 133-791, South Korea
[5] Seoul National University, Seoul 151-742, South Korea
[6] Inha University, Incheon 402-751, South Korea

Abstract. The consensus string problem is finding a representative string (consensus) of a given set $\mathbb{S}$ of strings. In this paper we deal with the consensus string problems optimizing both distance sum and radius, where the distance sum is the sum of (Hamming) distances from the strings in $\mathbb{S}$ to the consensus and the radius is the longest (Hamming) distance from the strings in $\mathbb{S}$ to the consensus. Although there have been results considering either distance sum or radius, there have been no results considering both as far as we know.

We present two algorithms to solve the consensus string problems optimizing both distance sum and radius for three strings. The first algorithm finds the optimal consensus string that minimizes both distance sum and radius, and the second algorithm finds the bounded consensus string such that, given constants s and r, the distance sum is at most s and the radius is at most r. Both algorithms take linear time.

1 Introduction

The multiple string comparison problem is one of fundamental research topics in computational biology and combinatorial pattern matching [1, 9, 10]. Finding

⋆ This work was partially supported by the Israel Science Foundation grant 35/05, the Israel-Korea Scientific Research Cooperation and Yahoo.

⋆⋆ Corresponding author, E-mail: jcna@sejong.ac.kr

⋆⋆⋆ This work was supported by the Korea Foundation for International Cooperation of Science & Technology(KICOS) through a grant provided by the Korean Ministry of Education, Science & Technology(MEST) in 2009 (No. K20717000007-09B0100-00710).

a representative string of a given set $\mathbb{S}$ of strings, called a *consensus string* (or *closest string* or *center string*), is a major problem in multiple string comparison, which is closely related to the motif recognition problem. Among the conditions that a string should satisfy to be accepted as a consensus, the two most important conditions are

1. to minimize the sum of (Hamming) distances from the strings in $\mathbb{S}$ to the consensus, and
2. to minimize the longest distance (or radius) from the strings in $\mathbb{S}$ to the consensus.

In this paper we deal with two related but different problems about finding a consensus string. The first one is finding an *optimal consensus string* minimizing both distance sum and radius as follows.

Problem 1. Optimal consensus
Given a set $\mathbb{S} = \{S_1, \ldots, S_k\}$ of k strings of length n, find a string X (if any) that minimizes both $\sum_{1 \le i \le k} d(X, S_i)$ and $\max_{1 \le i \le k} d(X, S_i)$ where $d(A, B)$ is the Hamming distance between strings A and B.

If such an optimal consensus string exists, it can be accepted as a consensus string of $\mathbb{S}$. However, sometimes such a string does not exist and a string satisfying loose conditions may be sought for as follows.

Problem 2. Bounded consensus
Given a set $\mathbb{S} = \{S_1, \ldots, S_k\}$ of k strings of length n and two positive integers s and r, find a string X (if any) satisfying both $\sum_{1 \le i \le k} d(X, S_i) \le s$ and $\max_{1 \le i \le k} d(X, S_i) \le r$.

Although minimizing both distance sum and radius from a consensus is important, researchers have only focused on finding a consensus minimizing either the distance sum or the radius. Minimizing the distance sum is rather easy. We can find a string X that minimizes the distance sum by selecting the character occurring most often in each position of the strings in $\mathbb{S}$. However, minimizing the radius is a hard problem in general. For general k, the problem of finding a string X such that $\max_{1 \le i \le k} d(X, S_i) \le r$ is NP-hard even when characters in strings are drawn from the binary alphabet [4]. Thus, attention has been restricted to approximation solutions [2, 5, 6, 11–14] and fixed-parameter solutions [7, 8, 14, 15].

For fixed-parameter solutions, Stojanovic [15] proposed a linear-time algorithm for $r = 1$. Gramm et al. [7, 8] proposed the first fixed-parameter algorithm running in $O(kn + kr^{r+1})$ time for finding a string X such that $\max_{1 \le i \le k} d(X, S_i) \le$

r. Ma and Sun [14] presented another algorithm running in $O(kn + kr(16|\Sigma|)^r)$ time, where Σ denotes the alphabet. Furthermore, there have been some algorithms for a small constant k. Gramm et al. [7] proposed a direct combinatorial algorithm for finding a string X that minimizes the radius for three strings. Sze et al. [16] showed a condition for the existence of a string whose radius is less than or equal to r. Boucher et al. [3] proposed an algorithm for finding a string X such that $\max_{1\leq i\leq 4} d(X, S_i) \leq r$ for four binary strings. For brief surveys on approximation solutions, readers are referred to [3, 14]. However, as far as we know, there have been no results on finding a consensus string minimizing both distance sum and radius.

In this paper we present the first algorithms to solve the consensus string problems minimizing both distance sum and radius for the set of three strings (i.e., when $k = 3$).

- We present an algorithm to solve the optimal consensus string problem (Problem 1). The algorithm finds a string X that minimizes both distance sum ($\sum_{1\leq i\leq 3} d(X, S_i)$) and radius ($\max_{1\leq i\leq 3} d(X, S_i)$) if such a string exists. Otherwise, the algorithm returns a string with the minimum distance sum among the strings whose radii are minimum. On top of the powerful functionalities of the algorithm, the algorithm is very efficient. It takes only $O(n)$ time to do all the computation above.
- We present an algorithm to solve the suboptimal consensus problem (Problem 2). The algorithm returns a string X (if any) satisfying both $\sum_{1\leq i\leq 3} d(X, S_i) \leq s$ and $\max_{1\leq i\leq 3} d(X, S_i) \leq r$ for given s and r. This algorithm runs in $O(n)$ time. In addition, the algorithm can be modified for faster execution if input strings are given in advance and r and s are given later. The modified algorithm computes the minimum of $\sum_{1\leq i\leq 3} d(X, S_i) + \max_{1\leq i\leq 3} d(X, S_i)$ for any string X. The minimum can be computed from the input strings even before r and s are given. Later, when r and s are given, the problem can be solved in $O(1)$ by using the minimum. This is very useful when r and s are given later or when several problems with different pairs of r and s are asked on the same input strings.

This paper is organized as follows. In Section 2, we give some definitions and notations. We present our algorithms for the consensus problems in Section 3. Finally we give concluding remarks in Section 4.

2 Preliminaries

For a string S, let $S[i]$ denote the ith character of S. For two strings X and Y, $d(X, Y)$ is defined as the Hamming distance between X and Y. Let $\mathbb{S} =$

	Type 0	Type 1	Type 2	Type 3	Type 4
S_1:	⊙ ⊙	× × × × ×	⊙ ⊙ ⊙	⊙ ⊙	× × ×
S_2:	⊙ ⊙	⊙ ⊙ ⊙ ⊙ ⊙	× × ×	⊙ ⊙	× × ×
S_3:	⊙ ⊙	⊙ ⊙ ⊙ ⊙ ⊙	⊙ ⊙ ⊙	× ×	× × ×
	$c_0 = 2$	$c_1 = 5$	$c_2 = 3$	$c_3 = 2$	$c_4 = 3$

Fig. 1. Types of columns in the alignment of 3 strings, where ⊙ and × represent match and mismatch characters at each position, respectively.

$\{S_1, \ldots, S_k\}$ be a set of k strings of equal length n. Given a string X, the *(consensus) radius* of X for $\mathbb{S}$, denoted by $R_{\mathbb{S}}(X)$, is defined as $\max_{1 \le p \le k} d(X, S_p)$ and the *(consensus) distance sum* of X for $\mathbb{S}$, denoted by $E_{\mathbb{S}}(X)$, is defined as $\sum_{1 \le p \le k} d(X, S_p)$. We omit the set notation $\mathbb{S}$ if not confusing. Then, Problem 1 is finding a string X that minimizes both $E(X)$ and $R(X)$, and Problem 2 is finding a string Y such that $E(Y) \le s$ and $R(Y) \le r$. We call a solution of Problem 1 an *optimal consensus string* and a solution of Problem 2 a *bounded consensus string*.

Consider the alignment of a string X and the strings in $\mathbb{S}$. Because the Hamming distance allows only substitutions, $X[i]$ is aligned with $S_p[i]$'s ($1 \le p \le k$). Thus, $\mathbb{S}$ can be regarded as a $k \times n$ character matrix, where the ith column consists of the ith characters of the k strings. For each column, we call the *majority* the character occurring most often and the *minority* the character occurring most seldom.

If we only consider the distance sum, that is, we want to find a string X with the minimum distance sum, X can be found easily by choosing the majority in each column. However, the problem of finding a string Y such that $R(Y) \le r$ is NP-hard even when restricted to a binary alphabet [4]. Thus, Problems 1 and 2 are also NP-hard in general.

3 Consensus string for three strings

In this section we consider the consensus string problems for $\mathbb{S} = \{S_1, S_2, S_3\}$. We first describe an algorithm for computing a string X with the minimum radius and show that X computed by the algorithm also minimizes the distance sum (Problem 1). Then, we show how to compute a bounded consensus string Y from X (Problem 2).

3.1 String with the minimum radius

Consider the alignment of the three strings S_1, S_2, and S_3. The column in every position i is divided into the following five types. See Figure 1.

- Type 0: $S_1[i] = S_2[i] = S_3[i]$ (all matches).
- Type 1: $S_1[i] \neq S_2[i] = S_3[i]$ ($S_1[i]$ is the minority).
- Type 2: $S_2[i] \neq S_1[i] = S_3[i]$ ($S_2[i]$ is the minority).
- Type 3: $S_3[i] \neq S_1[i] = S_2[i]$ ($S_3[i]$ is the minority).
- Type 4: $S_1[i] \neq S_2[i]$, $S_2[i] \neq S_3[i]$, and $S_3[i] \neq S_1[i]$ (all mismatches).

Let c_j $(0 \leq j \leq 4)$ denote the number of columns for type j. Without loss of generality, we assume that $c_1 \geq c_2 \geq c_3$.

Let E_{min} be the smallest sum of Hamming distances of S_1, S_2, S_3 from any string X', i.e., $E_{min} = min_{X'} \sum_{1 \leq p \leq 3} d(X', S_p)$. Obviously, the minimum distance sum $E_{min} = c_1 + c_2 + c_3 + 2c_4$. Let R_{min} be the smallest max of Hamming distances of S_1, S_2, S_3 from any string X'. i.e., $R_{min} = min_{X'} \max_{1 \leq p \leq 3} d(X', S_p)$. The following lemma gives a lower bound for the minimum radius R_{min}.

Lemma 1. *$R_{min} \geq \max(L_1, L_2)$, where $L_1 = (c_1 + c_2 + c_4)/2$ and $L_2 = (c_1 + c_2 + c_3 + 2c_4)/3$.*

Proof: First, R_{min} is greater than or equal to half of the distance between two farthest strings (i.e., S_1 and S_2) by Hamming distance. That is, $R_{min} \geq (c_1 + c_2 + c_4)/2 = L_1$. Moreover, $R_{min} \geq E_{min}/3 = L_2$. Indeed if there exists a string X' such that $R(X') < E_{min}/3$, then $E(X') < E_{min}$, which is a contradiction. □

Remark. Because R_{min} is an integer, L_1 is exactly $\lceil (c_1 + c_2 + c_4)/2 \rceil$ and L_2 is $\lceil (c_1 + c_2 + c_3 + 2c_4)/3 \rceil$. Throughout the paper, the ceiling function or the floor function should be applied to all fractional expressions including L_1 and L_2. For simplicity, however, we assume that values of all fractional expressions are integers.

By comparing the two lower bounds L_1 and L_2, we get the following.

Corollary 1. *$R_{min} \geq L_2$ if $c_1 + c_2 \leq 2c_3 + c_4$, and $R_{min} \geq L_1$ otherwise.*

Thus, if an algorithm computes a string whose radius is L_2 when $c_1 + c_2 \leq 2c_3 + c_4$ and L_1 when $c_1 + c_2 > 2c_3 + c_4$, the algorithm always computes a string with the minimum radius.

Now we describe how to compute a string X with the minimum radius and show that the radius of X is $\max(L_1, L_2)$. Basically, we select one of $S_1[i]$, $S_2[i]$, and $S_3[i]$ in each position i. We always select the majority in every column of

type 0. In columns of other types, we select characters in the following way. We have two cases.

I. When $c_1 + c_2 \leq 2c_3 + c_4$, i.e., $L_1 \leq L_2$.

Let $c_{41} = (c_4 + 2c_1 - c_2 - c_3)/3$, $c_{42} = (c_4 + 2c_2 - c_1 - c_3)/3$, and $c_{43} = (c_4 + 2c_3 - c_1 - c_2)/3$. Obviously, $c_{41} + c_{42} + c_{43} = c_4$. Then, we compute a string X in the following way.

- In every column of Types 0-3, select the majority.
- In columns of type 4, select c_{41} characters of S_1, c_{42} characters of S_2, and c_{43} characters of S_3.

Now, we prove (1) that c_{41}, c_{42}, and c_{43} are nonnegative and (2) that the string X is a string with the minimum radius by showing that its radius is L_2.

- c_{41}, c_{42}, and c_{43} are nonnegative.
  - c_{43} is nonnegative by the condition $c_1 + c_2 \leq 2c_3 + c_4$.
  - c_{42} is nonnegative if inequality $c_1 + c_3 \leq 2c_2 + c_4$ is satisfied. Since we assume that $c_3 \leq c_2$, $c_1 + c_3 \leq c_1 + c_2$ and $2c_3 + c_4 \leq 2c_2 + c_4$, and thus $c_1 + c_3 \leq 2c_2 + c_4$ by the condition $c_1 + c_2 \leq 2c_3 + c_4$.
  - The proof that c_{41} is nonnegative is similar to the proof that c_{42} is nonnegative.
- The radius of X is L_2.
  The distances of strings S_1, S_2, and S_3 from X are as follows:
  - $d(S_1, X) = c_1 + c_{42} + c_{43} = c_1 + (c_4 + 2c_2 - c_1 - c_3)/3 + (c_4 + 2c_3 - c_1 - c_2)/3 = (c_1 + c_2 + c_3 + 2c_4)/3 = L_2$.
  - $d(S_2, X) = c_2 + c_{41} + c_{43} = c_2 + (c_4 + 2c_1 - c_2 - c_3)/3 + (c_4 + 2c_3 - c_1 - c_2)/3 = (c_1 + c_2 + c_3 + 2c_4)/3 = L_2$.
  - $d(S_2, X) = c_3 + c_{41} + c_{42} = c_3 + (c_4 + 2c_1 - c_2 - c_3)/3 + (c_4 + 2c_2 - c_1 - c_3)/3 = (c_1 + c_2 + c_3 + 2c_4)/3 = L_2$.

  Since $d(S_1, X) = d(S_2, X) = d(S_3, X) = L_2$, the radius (i.e. $\max_{1 \leq p \leq 3} d(X, S_p)$) is L_2.

II. When $c_1 + c_2 > 2c_3 + c_4$, i.e., $L_1 > L_2$.

We separate this case into two subcases $c_1 - c_2 < c_4$ and $c_1 - c_2 \geq c_4$.

(a) *When $c_1 - c_2 \leq c_4$.*

Let $c_{41} = (c_4 + c_1 - c_2)/2$, $c_{42} = (c_4 - c_1 + c_2)/2$, and $c_{43} = 0$. Obviously, $c_{41} + c_{42} + c_{43} = c_4$. Then, we compute a string X in the following way.

- In every column of Types 0-3, select the majority.
- In columns of type 4, select c_{41} characters of S_1, c_{42} characters of S_2, and c_{43} characters of S_3.

Now, we prove (1) that c_{41} and c_{42} are nonnegative and (2) that the string X is a string with the minimum radius by showing that its radius is L_1.

- c_{41}, c_{42}, and c_{43} are nonnegative.
  - c_{41} is nonnegative by the assumption $c_1 \geq c_2$.
  - c_{42} is nonnegative by the condition $c_1 - c_2 \leq c_4$.
- The radius of X is L_1.
  The distances of strings S_1, S_2, and S_3 from X are as follows:
  - $d(S_1, X) = c_1 + c_{42} + c_{43} = c_1 + (c_4 - c_1 + c_2)/2 = (c_4 + c_1 + c_2)/2 = L_1$.
  - $d(S_2, X) = c_2 + c_{41} + c_{43} = c_2 + (c_4 + c_1 - c_2)/2 = (c_4 + c_1 + c_2)/2 = L_1$.
  - $d(S_3, X) = c_3 + c_{41} + c_{42} = c_3 + c_4 < L_1$. (One can show $L_1 - (c_3 + c_4) > 0$ using the condition $c_1 + c_2 > 2c_3 + c_4$.)

  Thus the radius of X is L_1.

(b) *When* $c_1 - c_2 > c_4$.

Let $c_{11} = (c_1 + c_2 + c_4)/2$ (nonnegative trivially) and $c_{12} = (c_1 - c_2 - c_4)/2$ (nonnegative due to $c_1 - c_2 > c_4$). Then, we compute a string X in the following way.

- In every column of Types 0, 2, and 3, select the majority.
- In columns of type 1, select c_{11} majority characters and c_{12} minority characters (i.e. characters of S_1).
- In every column of type 4, select the character of S_1.

Now, we prove that the string X is a string with the minimum radius by showing that its radius is L_1.

- $d(S_1, X) = c_{11} = (c_1 + c_2 + c_4)/2 = L_1$.
- $d(S_2, X) = c_{12} + c_2 + c_4 = (c_1 - c_2 - c_4)/2 + c_2 + c_4 = (c_1 + c_2 + c_4)/2 = L_1$.
- $d(S_3, X) = c_{12} + c_3 + c_4 = (c_1 - c_2 - c_4)/2 + c_3 + c_4 = (c_1 - c_2 + 2c_3 + c_4)/2 \leq L_1$. (One can show $L_1 - (c_1 - c_2 + 2c_3 + c_4)/2 \geq 0$ using the assumption $c_2 \geq c_3$.)

Thus, the radius of X is L_1.

Conclusively, the algorithm computes a string with the minimum radius.

Lemma 2. *Given the string set* $\mathbb{S}$*, a string with the minimum radius for* $\mathbb{S}$ *can be found in* $O(n)$ *time.*

Proof: We have already shown that the radius of string X computed by the algorithm is minimum. Consider the time complexity. The types of columns and c_i $(0 \leq i \leq 4)$ can be determined by scanning three strings once. Furthermore, other computations can be done in constant time and character selections in every column can be done by scanning the strings once. Thus, the algorithm takes $O(n)$ time. □

3.2 Optimal consensus string

Consider the relation between the radius and the distance sum. In cases I and II (a), X is a string with the minimum distance sum as well as with the minimum radius because we select the majority in every column. In case II (b), however, X is not a string with the minimum distance sum. We can decrease the distance sum by reducing the number of minority selections in columns of type 1. If so, however, the radius increases as much as the distance sum decreases. The following lemma shows the relation between the radius and the distance sum in case II (b).

Lemma 3. *In case of II (b), $R(Z) + E(Z) \geq R_{min} + E_{min} + M$ for any string Z, where $M = (c_1 - c_2 - c_4)/2$.*

Proof: Recall that $R_{min} = L_1 = (c_1 + c_2 + c_4)/2$ and $E_{min} = c_1 + c_2 + c_3 + 2c_4$. Let Z be a string such that $E(Z) = E_{min} + t$, where t is the number of minority selections in all columns when constructing Z. Then, we prove this lemma by showing that $R(Z) \geq R_{min} + M - t = c_1 - t$. Let m_j $(1 \leq j \leq 3)$ be the number of minority selections (i.e., characters of S_j) in columns of type j when constructing Z. Obviously, $m_1 + m_2 + m_3 = t$. Let m_{4j} $(1 \leq j \leq 3)$ be the number of characters of S_j selected in columns of type 4 when constructing Z. Then,

$$\begin{aligned} d(S_1, Z) &= c_1 - m_1 + m_2 + m_3 + m_{42} + m_{43} \\ &= c_1 - t + 2m_2 + 2m_3 + m_{42} + m_{43} \text{ (using } m_1 = t - m_2 - m_3) \\ &\geq c_1 - t. \end{aligned}$$

Thus, $R(Z) = \max(d(Z, S_1), d(Z, S_2), d(Z, S_3)) \geq c_1 - t$. □

Corollary 2. *The string X computed by the above algorithm is a string that minimizes $R(X) + E(X)$.*

Lemma 4. *The above algorithm computes an optimal consensus string if exists.*

Proof: We have already shown that the radius of string X is minimum. In cases I and II (a), X is also a string with the minimum distance sum. In case II (b), there is no optimal consensus string by Lemma 3 because $c_1 - c_2 - c_4 > 0$. □

Lemma 5. *There is no optimal consensus string if and only if both* $c_1 + c_2 > 2c_3 + c_4$ *and* $c_1 - c_2 > c_4$ *(case II (b)).*

Lemma 6. *If there is no optimal consensus string, the above algorithm computes a string* X *whose distance sum is smallest among all strings with the minimum radius.*

Proof: The radius of string X is minimum. By Corollary 2, the distance sum of X is smallest among strings whose radius is $R(X)$. □

3.3 Bounded consensus string

We show how to compute a bounded consensus string Y from X (Problem 2). In cases I and II (a), a solution is easy. Because $R(X) = R_{min}$ and $E(X) = E_{min}$, X is a bounded consensus string if $R(X) \leq r$ and $E(X) \leq s$, and there is no bounded consensus string otherwise. Consider case II (b). If $R(X) > r$, $E(X) > s$, or $R(X) + E(X) > r + s$ (by Corollary 2), there is no bounded consensus string. Otherwise, we can find a bounded consensus string by decreasing the number of minority selections when constructing X.

Lemma 7. *A bounded consensus string can be found in* $O(n)$ *time if exists.*

Lemma 8. *Let* $M = (c_1 - c_2 - c_4)/2$ *if* $c_1 + c_2 > 2c_3 + c_4$ *and* $c_1 - c_2 > c_4$ *(case II (b)), and* $M = 0$ *otherwise. Then, there exists a bounded consensus string if and only if* $R_{min} \leq r$, $E_{min} \leq s$, *and* $R_{min} + E_{min} + M \leq r + s$,

By Lemmas 2, 4 and 8, we get the following theorem.

Theorem 1. *Problems 1 and 2 for three strings can be solved in* $O(n)$ *time.*

4 Concluding Remarks

We considered the consensus string problem optimizing both distance sum and radius, and proposed a linear-time algorithm for three strings. Moreover, we studied the conditions for which there exists an optimal consensus string or a bounded consensus string for three strings. It remains an open problem to find a consensus string for $k \geq 4$ strings. Another open problem is to find a consensus string when strings are compared by the edit distance. This problem doesn't look easy even for three strings.

References

1. S. Altschul and D. Lipman. Trees, stars, and multiple sequence alignment. *SIAM Journal on Applied Mathematics*, 49:197–209, 1989.
2. A. Ben-Dor, G. Lancia, J. Perone, and R. Ravi. Banishing bias from consensus sequences. In *Proceedings of the 8th Symposium on Combinatorial Pattern Matching*, pages 247–261, 1997.
3. C. Boucher, D. Brown, and S. Durocher. On the structure of small motif recognition instances. In *Proceedings of the 15th Symposium on String Processing and Information Retrieval*, pages 269–281, 2008.
4. M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.
5. L. Gasieniec, J. Jansson, and A. Lingas. Efficient approximation algorithms for the Hamming center problem. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 905–906, 1999.
6. L. Gasieniec, J. Jansson, and A. Lingas. Approximation algorithms for Hamming clustering problems. *Journal of Discrete Algorithms*, 2(2):289–301, 2004.
7. J. Gramm, R. Niedermeier, and P. Rossmanith. Exact solutions for closest string and related problems. In *Proceedings of the 12th International Symposium on Algorithms and Computation*, pages 441–453, 2001.
8. J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.
9. D. Gusfield. *Algorithms on Strings, Tree, and Sequences*. Cambridge University Press, Cambridge, 1997.
10. R.M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 278–285, 1993.
11. K. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms*, pages 633–642, 1999.
12. M. Li, B. Ma, and L. Wang. Finding similar regions in many strings. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, pages 473–482, 1999.
13. M. Li, B. Ma, and L. Wang. On the closest string and substring problems. *Journal of the ACM*, 49(2):157–171, 2002.
14. B. Ma and X. Sun. More efficient algorithms for closest string and substring problems. In *Proceedings of the 12th Annual International Conference on Research in Computational Molecular Biology*, pages 396–409, 2008.
15. N. Stojanovic, P. Berman, D. Gumucio, R. Hardison, and W. Miller. A linear-time algorithm for the 1-mismatch problem. In *Proceedings of the 5th International Workshop on Algorithms and Data Structures*, pages 126–135, 1997.
16. S. Sze, S. Lu, and J. Chen. Integrating sample-driven and pattern-driven approaches in motif finding. In *Proceedings of the 4th Workshop on Algorithms in Bioinformatics*, pages 438–449, 2004.