On The Closest String and Substring Problems*

Ming Li
Department of Computer Science
University of Waterloo
Waterloo, Ont. N2L 3G1, Canada
E-mail: mli@math.uwaterloo.ca

Bin Ma
Department of Computer Science
University of Western Ontario
London, Ont. N6A 5B7, Canada
E-mail: bma@csd.uwo.ca

Lusheng Wang
Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
E-mail: lwang@cs.cityu.edu.hk

Abstract

The problem of finding a center string that is 'close' to every given string arises in computational molecular biology and coding theory. This problem has two versions: the Closest String problem and the Closest Substring problem. Given a set of strings $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, each of length m, the Closest String problem is to find the smallest d and a string s of length m which is within Hamming distance d to each $s_i \in \mathcal{S}$. This problem comes from coding theory when we are looking for a code not too far away from a given set of codes. Closest Substring problem, with an additional input integer L, asks for the smallest d and a string s, of length L, which is within Hamming distance d away from a substring, of length L, of each s_i. This problem is much more elusive than the Closest String problem. The Closest Substring problem is formulated from applications in finding conserved regions, identifying genetic drug targets and generating genetic probes in molecular biology. Whether there are efficient approximation algorithms for both problems are major open questions in this area. We present two polynomial time approximation algorithms with approximation ratio $1+\epsilon$ for any small ϵ to settle both questions.

*Results in Section 2 and some preliminary results in sections 3 and 4 have been presented in *Proc. 31st ACM Symp. Theory of Computing*, May, 1999 [14]. The full solution of the Closest Substring problem in sections 3 and 4 were presented in *Proc. 11th Symp. Combinatorial Pattern Matching*, June, 2000, [17].

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 [24, 11, 10, 23], genetic drug target identification [13], designing genetic probes [13], universal PCR primer design [12, 4, 21, 13], and, outside computational biology, in coding theory [5, 6]. Such problems may be considered to be various generalizations of the common substring problem, allowing errors. Many measures have been proposed for finding such regions common to every given string. A popular and most fundamental measure is the Hamming distance. Moreover, two popular objective functions are used in these areas. One is the total sum of distances between the center string (common substring) and each of the given strings. The other is the maximum distance between the center string and a given string. In this paper, we focus on the second objective function for Hamming distance. The first objective function for Hamming distance and other measures, like the relative entropy measure used by Stormo and his coauthors [10], is treated in [15]. It requires completely different techniques.

Throughout the paper, we will use a fixed finite alphabet Σ. Let s and s' be finite strings over Σ. Let $d(s, s')$ denote the Hamming distance between s and s'. $|s|$ is the length of s. $s[i]$ is the i-th character of s. Thus, $s = s[1]s[2]\ldots s[|s|]$. The following are the problems we study in this paper:

Closest String: Given a set $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ of strings each of length m, find a center string s of length m minimizing d such that for every string $s_i \in \mathcal{S}$, $d(s, s_i) \leq d$.

Closest Substring: Given a set $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ of strings each of length m, and an integer L, find a center string s of length L minimizing d such that for each $s_i \in \mathcal{S}$ there is a length L substring t_i of s_i with $d(s, t_i) \leq d$.

Throughout this paper, we call the number d in the definitions of Closest String and Closest Substring the radius (or the cost) of the solution.

Closest String has been widely and independently studied in different contexts. In the context of coding theory it was shown to be NP-hard [5]. In DNA sequence related topics, [3] gave an exact algorithm when the distance d is a constant. [2, 6] gave near-optimal approximation algorithms only for large d (super-logarithmic in number of sequences); however the straightforward LP (linear programming) relaxation technique does not work when d is small because the randomized rounding procedure introduces large errors. This is exactly the reason why [6, 13] analyzed more involved approximation algorithms, and obtained the ratio $\frac{4}{3}$ approximation algorithms. Note that the small d is the key in applications such as genetic drug target search where we look for similar regions to which a complementary drug sequence would bind. It is a major open problem [5, 3, 2, 6, 13] to achieve the best approximation ratio for this problem. (Justifications for using Hamming distance can also be found in these references, especially [13].) We present a polynomial time approximation scheme (PTAS), settling the problem.

Closest Substring is a more general version of the Closest String problem. Obviously, it is also NP-hard. In applications such as drug target identification and genetic probe design, the radius d is usually small. Moreover, when the radius d is small, the center strings can also be used as *motifs* in *repeated-motif* methods for multiple sequence alignment problems [9, 20, 22, 25, 26, 28, 19], that repeatedly find motifs and recursively decompose the sequences into shorter sequences. The problem turns out to be much more elusive than Closest String. We extend the techniques developed for closest string here to design a PTAS for the Closest Substring problem when d is small, i.e., $d \le O(\log(nm))$. Using a *random sampling* technique, and combining our methods for Closest String, we then design a PTAS for Closest Substring, for all d.

It is perhaps cautious to point out that the approximation algorithms developed here may not be directly applicable to bioinformatics practice because of their high time complexity. Heuristics would help to improve the running time while losing some theoretical vigor. We have implemented one such system, COPIA [16].

Let us introduce more notations. Let s and t be strings of length m. A multiset $P = \{j_1, j_2, \ldots, j_k\}$ such that $1 \le j_1 \le j_2 \le \cdots \le j_k \le m$ is called a *position set*. By $s|_P$ we denote the string $s[j_1]\, s[j_2] \cdots s[j_k]$. We also write $d^P(s,t)$ to mean $d(s|_P, t|_P)$.

Before ending this section, we present two lemmas that will be used in both Section 2 and Section 4. Lemma 1 is commonly known as Chernoff's bounds ([18], Theorem 4.2 and 4.3):

Lemma 1 [18] *Let $X_1, X_2, \ldots, X_n$ be n independent random 0-1 variables, where X_i takes 1 with probability p_i, $0 < p_i < 1$. Let $X = \sum_{i=1}^{n} X_i$, and $\mu = E[X]$. Then for any $\delta > 0$,*

(1) $\mathbf{Pr}(X > (1+\delta)\mu) < \left[\frac{\mathbf{e}^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}$,

(2) $\mathbf{Pr}(X < (1-\delta)\mu) \le \exp\left(-\frac{1}{2}\mu\delta^2\right)$.

¿From Lemma 1, we can prove the following lemma:

Lemma 2 *Let X_i, $1 \le i \le n$, X and μ be defined as in Lemma 1. Then for any $0 < \epsilon \le 1$,*

(1) $\mathbf{Pr}(X > \mu + \epsilon n) < \exp\left(-\frac{1}{3}n\epsilon^2\right)$,

(2) $\mathbf{Pr}(X < \mu - \epsilon n) \le \exp\left(-\frac{1}{2}n\epsilon^2\right)$.

Proof. (1) Let $\delta = \frac{\epsilon n}{\mu}$. By Lemma 1,

$$\mathbf{Pr}(X > \mu + \epsilon n) < \left[\frac{\mathbf{e}^{\frac{\epsilon \mathbf{n}}{\mu}}}{(1+\frac{\epsilon n}{\mu})^{(1+\frac{\epsilon n}{\mu})}}\right]^{\mu} = \left[\frac{\mathbf{e}}{(1+\frac{\epsilon n}{\mu})^{(1+\frac{\mu}{\epsilon n})}}\right]^{\epsilon n} \leq \left[\frac{\mathbf{e}}{(1+\epsilon)^{1+\frac{1}{\epsilon}}}\right]^{\epsilon n},$$

where the last inequality is because $\mu \leq n$ and that $(1+x)^{(1+\frac{1}{x})}$ is increasing for $x \geq 0$. One can verify that for $0 < \epsilon \leq 1$, $\frac{\mathbf{e}}{(1+\epsilon)^{1+\frac{1}{\epsilon}}} \leq \exp\left(-\frac{\epsilon}{3}\right)$. Therefore, (1) is proved.

(2) Let $\delta = \frac{\epsilon n}{\mu}$. By Lemma 1, (2) is proved. □

2 Approximating Closest String

In this section, we give a PTAS for Closest String. Let $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ be a set of n strings, each of length m. We note that a direct application of LP relaxation in [2] to all the m positions of the strings does not work when the optimal radius is small. Rather we extend an idea in [13] to do LP relaxation only to a fraction of the m positions.

We first describe the basic idea behind our approach intuitively. Given a subset of r strings from $\mathcal{S}$. Line them up. Consider the characters where they all agree. Intuitively, there is a high likelihood that the optimal solution should also agree with these characters on corresponding positions. Indeed, this will only slightly worsen the solution. Lemma 3 shows that this is true for at least one subset of r strings. Thus all we need to do is to optimize on the positions (characters) where the r strings do not agree, by LP relaxation and randomized rounding.

Let r be an integer. Roughly, our algorithm will be running in time $n^{O(r)}$ and with approximation ratio $1 + O(\frac{1}{r})$. So for each fixed r, our algorithm is a polynomial time algorithm. As r grows, our approximation becomes better, but the time complexity also grows.

Let $P = \{j_1, j_2, \ldots, j_k\}$ be a set and $1 \leq j_1 \leq j_2 \leq \cdots \leq j_k \leq m$. Let s be a string of length m. Recall, we write $s|_P$ to denote the string $s[j_1]\, s[j_2] \cdots s[j_k]$.

Let s be the optimal solution with radius d_{opt}. For any $s_i \in \mathcal{S}$, if we use s_i as the approximation of s, then $d(s_i, s_j) \leq 2d_{opt}$ for any $s_j \in \mathcal{S}$. Thus, this *center replacing* algorithm has performance ratio 2. Now, we generalize the ratio-2 algorithm by considering k strings s_{i_1}, s_{i_2}, $\ldots$, s_{i_k} in $\mathcal{S}$ at a time. Let $Q_{i_1,i_2,\ldots,i_k}$ be the set of positions where $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$ agree. We can show that there exist indices i_1, i_2, $\ldots$, i_r such that the unique characters at positions in $Q_{i_1,i_2,\ldots,i_r}$ form a good approximation of an optimal center string s, i.e., for any $s_l \in \mathcal{S}$,

$$d(s_l|_{Q_{i_1,i_2,\ldots,i_r}}, s_{i_1}|_{Q_{i_1,i_2,\ldots,i_r}}) - d(s_l|_{Q_{i_1,i_2,\ldots,i_r}}, s|_{Q_{i_1,i_2,\ldots,i_r}}) \leq \frac{1}{2r-1} d_{opt}. \quad (1)$$

Let $P_{i_1,i_2,\ldots,i_k} = \{1, 2, \ldots, m\} - Q_{i_1,i_2,\ldots,i_k}$. Then the size of $P_{i_1,i_2,\ldots,i_k}$ is reduced so that either the LP-relaxization approach or an enumeration approach can work at the positions in $P_{i_1,i_2,\ldots,i_k}$.

To show (1), we do not directly compare s_{i_1} and s at positions in $Q_{i_1,i_2,\ldots,i_r}$. Instead, we consider the relaxed sets $J(l) = \{j \in Q_{i_1,i_2,\ldots,i_r} \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}$ for all $s_l \in \mathcal{S}$. We show that for each $s_l \in \mathcal{S}$, the size of $J(l)$ is small, i.e., $|J(l)| \leq \frac{1}{2r-1} d_{opt}$.

For any $2 \leq k < r$, and $1 \leq i_1, i_2, \ldots, i_k \leq n$, let $p_{i_1,i_2,\ldots,i_k}$ be the number of mismatches between s_{i_1} and s at the positions in $Q_{i_1,i_2,\ldots,i_k}$. Let

$$\rho_k = \min_{1 \leq i_1, i_2, \ldots, i_k \leq n} p_{i_1,i_2,\ldots,i_k} / d_{opt}.$$

Increasing k, the number of positions where s_{i_j} all agree decreases, i.e., the size of $Q_{i_1,i_2,\ldots,i_k}$ decreases. By definition, ρ_k decreases, too.

Our strategy is to show that (1) $|J(l)| \leq (\rho_k - \rho_{k+1}) d_{opt}$ for any $s_l \in \mathcal{S}$ and any $2 \leq k \leq r$; and (2) one of $\rho_2 - \rho_3$, $\rho_3 - \rho_4$, $\ldots$, $\rho_r - \rho_{r+1}$ is at most $\frac{1}{2r-1}$.

Let $\rho_0 = \max_{1 \leq i,j \leq n} d(s_i, s_j)/d_{opt}$. By the triangle inequality, $\rho \leq 2$ in worst case. The following lemma shows that (1) holds when the center replacing algorithm does not give a high quality approximation, i.e., $\rho_0 > 1 + \frac{1}{2r-1}$. (Otherwise, we can simply use the center replacing algorithm.) This lemma is key to our approximation algorithm.

Lemma 3 *For any constant r, $2 \leq r < n$, if $\rho_0 > 1 + \frac{1}{2r-1}$, then there are indices $1 \leq i_1, i_2, \ldots, i_r \leq n$ such that for any $1 \leq l \leq n$,*

$$d(s_l|_{Q_{i_1,i_2,\ldots,i_r}}, s_{i_1}|_{Q_{i_1,i_2,\ldots,i_r}}) - d(s_l|_{Q_{i_1,i_2,\ldots,i_r}}, s|_{Q_{i_1,i_2,\ldots,i_r}}) \leq \frac{1}{2r-1} d_{opt}.$$

Proof. First, we show the following claim.

Claim 4 *For any k such that $2 \leq k \leq r$, where r is a constant, there are indices $1 \leq i_1, i_2, \ldots, i_r \leq n$ such that for any $s_l \in \mathcal{S}$,*

$$|J(l)| \leq (\rho_k - \rho_{k+1})\, d_{opt}.$$

Proof. Consider the indices $1 \leq i_1, i_2, \ldots, i_k \leq n$ such that $p_{i_1,i_2,\ldots,i_k} = \rho_k d_{opt}$. Then for any $1 \leq i_{k+1}, i_{k+2}, \ldots, i_r \leq n$ and $1 \leq l \leq n$, we have

$$\begin{aligned}
& |J(l)| = |\{j \in Q_{i_1,i_2,\ldots,i_r} \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \\
\leq\ & |\{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \qquad (2) \\
=\ & |\{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] \neq s[j]\} - \{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] = s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \\
=\ & |\{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] \neq s[j]\} - \{j \in Q_{i_1,i_2,\ldots,i_k,l} \mid s_{i_1}[j] \neq s[j]\}| \\
=\ & |\{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] \neq s[j]\}| - |\{j \in Q_{i_1,i_2,\ldots,i_k,l} \mid s_{i_1}[j] \neq s[j]\}| \qquad (3) \\
=\ & p_{i_1,i_2,\ldots,i_k} - p_{i_1,i_2,\ldots,i_k,l} \\
\leq\ & (\rho_k - \rho_{k+1})\, d_{opt},
\end{aligned}$$

where Inequality (2) comes from the fact that $Q_{i_1,i_2,\ldots,i_r} \subseteq Q_{i_1,i_2,\ldots,i_k}$ and Equality (3) is because $\{j \in Q_{i_1,i_2,\ldots,i_k,l} \mid s_{i_1}[j] \neq s[j]\} \subseteq \{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] \neq s[j]\}$. $\Box$

We now give an upper bounded for $\min\{\rho_2 - \rho_3, \rho_3 - \rho_4, \ldots, \rho_r - \rho_{r+1}\}$. Consider the sum of the $r-1$ terms,

$$(\rho_2 - \rho_3) + (\rho_3 - \rho_4) + \ldots + (\rho_r - \rho_{r+1}) = \rho_2 - \rho_{r+1} \leq \rho_2 \leq 1.$$

Thus, one of $\rho_2 - \rho_3$, $\rho_3 - \rho_4$, $\ldots$, $\rho_r - \rho_{r+1}$ is at most $\frac{1}{r-1}$. We can give a better bound by considering the weighted average of r terms $\rho_0 - 1$, $\rho_2 - \rho_3$, $\rho_3 - \rho_4$, $\ldots$, $\rho_r - \rho_{r+1}$.

Claim 5 *For $2 \leq r < n$, $\min\{\rho_0 - 1, \rho_2 - \rho_3, \rho_3 - \rho_4, \ldots, \rho_r - \rho_{r+1}\} \leq \frac{1}{2r-1}$.*

Proof. Consider $1 \leq i, j \leq n$ such that $d(s_i, s_j) = \rho_0 d_{opt}$. Then among the positions where s_i mismatches s_j, for at least one of the two strings, say, s_i, the number of mismatches between s_i and s is at least $\frac{\rho_0}{2} d_{opt}$. Thus, among the positions where s_i matches s_j, the number of mismatches between s_i and s is at most $(1 - \frac{\rho_0}{2})d_{opt}$. Therefore, $\rho_2 \leq 1 - \frac{\rho_0}{2}$. So,

$$\frac{\frac{1}{2}(\rho_0 - 1) + (\rho_2 - \rho_3) + (\rho_3 - \rho_4) + \cdots + (\rho_r - \rho_{r+1})}{\frac{1}{2} + r - 1} \leq \frac{\frac{1}{2}\rho_0 + \rho_2 - \frac{1}{2}}{r - \frac{1}{2}} \leq \frac{1}{2r-1}.$$

Thus, at least one of $\rho_0 - 1$, $\rho_2 - \rho_3$, $\rho_3 - \rho_4$, $\ldots$, $\rho_r - \rho_{r+1}$ is less than or equal to $\frac{1}{2r-1}$. $\Box$

If $\rho_0 > 1 + \frac{1}{2r-1}$, then from Claim 5, there must be a $2 \leq k \leq r$ such that $\rho_k - \rho_{k+1} \leq \frac{1}{2r-1}$. ¿From Claim 4, $|J(l)| \leq \frac{1}{2r-1} d_{opt}$. Hence, there are at most $\frac{1}{2r-1} d_{opt}$ positions in $Q_{i_1,i_2,\ldots,i_r}$ where s_l differs from s_{i_1} while agrees with s. The lemma is proved. $\Box$

Lemma 3 gives a way to find a good approximation of an optimal center string s at positions in $Q_{i_1,i_2,\ldots,i_r}$ for some i_1, i_2, $\ldots$, i_r. Lemma 7 shows how to use this partial approximation solution from Lemma 3 to construct a good center string at all L positions.

Let $P_{i_1,i_2,\ldots,i_k} = \{1, 2, \ldots, m\} - Q_{i_1,i_2,\ldots,i_k}$. We have

Claim 6 *$|P_{i_1,i_2,\ldots,i_k}| \leq k d_{opt}$ and $|Q_{i_1,i_2,\ldots,i_k}| \geq m - k d_{opt}$.*

Proof. Let q be a position where some of s_{i_1}, s_{i_2}, $\ldots$, s_{i_k} do not agree. Then there exists some s_{i_j} such that $s_{i_j}[q] \neq s[q]$. Since $d(s_{i_j}, s) \leq d_{opt}$, each s_{i_j} contributes at most d_{opt} positions in $P_{i_1,i_2,\ldots,i_k}$. Thus, $|P_{i_1,i_2,\ldots,i_k}| \leq k d_{opt}$. By definition, $|Q_{i_1,i_2,\ldots,i_k}| \geq m - k d_{opt}$. $\Box$

For the positions in $P_{i_1,i_2,\ldots,i_r} = \{1, 2, \ldots, m\} - Q_{i_1,i_2,\ldots,i_r}$, we use ideas in [13] and employ the following two strategies: (1) if $|P_{i_1,i_2,\ldots,i_r}|$ is large, i.e., $d_{opt} > \Omega(\log m)$, we use the LP relaxation to approximate s, since $|P_{i_1,i_2,\ldots,i_k}| \leq k d_{opt}$ and thus the conditions for applying the method are satisfied; (2) if $|P_{i_1,i_2,\ldots,i_r}|$ is small, i.e., $d_{opt} \leq O(\log m)$, we can enumerate $|\Sigma|^{|P_{i_1,i_2,\ldots,i_r}|}$ possibilities to approximate s.

Lemma 7 *Let $\mathcal{S} = \{s_1, s_2, \ldots s_n\}$, where $|s_i| = m$ for $i = 1, \ldots, n$. Assume that s is the optimal solution of* Closest String *and $\max_{1 \le i \le n} d(s_i, s) = d_{opt}$. Given a string s' and a position set Q of size $m - O(d_{opt})$ such that for any $i = 1, \ldots, n$,*

$$d(s_i|_Q, s'|_Q) - d(s_i|_Q, s|_Q) \le \rho\, d_{opt}, \tag{4}$$

where $0 \le \rho \le 1$, one can obtain a solution with radius at most $(1+\rho+\epsilon)d_{opt}$ in polynomial time for any fixed $\epsilon \ge 0$. The $\epsilon\, d_{opt}$ is considered as the error of the solution.

Proof. Let $P = \{1, 2, \ldots, m\} - Q$. Then, for any two strings t and t' of length m, we have $d(t|_P, t'|_P) + d(t|_Q, t'|_Q) = d(t, t')$. Thus for any $i = 1, 2, \ldots, n$,

$$\begin{aligned} d(s_i|_P, s|_P) &= d(s_i, s) - d(s_i|_Q, s|_Q) \\ &\le d(s_i, s) - (d(s_i|_Q, s'|_Q) - \rho\, d_{opt}) \\ &\le (1+\rho)\, d_{opt} - d(s_i|_Q, s'|_Q). \end{aligned}$$

Therefore, the following optimization problem

$$\begin{cases} \min \ \ d; \\ d(s_i|_P, y) \le d - d(s_i|_Q, s'|_Q), \ \ i = 1, \cdots, n; |y| = |P|, \end{cases} \tag{5}$$

has a solution $y = s|_P$ with cost $d \le (1+\rho)d_{opt}$. Note that y is a variable that represents a string of length $|P|$. Suppose the optimal solution for the optimization problem (5) is such that the cost is d_0. Then

$$d_0 \le (1+\rho)d_{opt}. \tag{6}$$

Now we solve (5) approximately. Let π be a permutation on $1, 2, \ldots, m$. For any length m string t, we use t^π to denote the string $t[\pi(1)]t[\pi(2)]\ldots t[\pi(m)]$. Clearly, s is an optimal center string for $\{s_1, s_2, \ldots, s_n\}$ if and only if s^π is an optimal center string for $\{s_1^\pi, s_2^\pi, \ldots, s_n^\pi\}$ with the same cost d. Therefore, without loss of generality, we assume that $P = \{1, 2, \ldots, |P|\}$. Similar to [2, 13], we use a 0-1 variable $y_{j,a}$ to indicate whether $y[j] = a$, where $1 \le j \le |P|$ and $a \in \Sigma$. Denote $\chi(s_i[j], a) = 0$ if $s_i[j] = a$ and 1 if $s_i[j] \ne a$. Then (5) can be formulated to an equivalent 0-1 optimization problem as follows:

$$\begin{cases} \min \ \ d; \\ \sum_{a \in \Sigma} y_{j,a} = 1, \ \ j = 1, 2, \ldots, |P|, \\ \sum_{1 \le j \le |P|} \sum_{a \in \Sigma} \chi(s_i[j], a)\, y_{j,a} \le d - d(s_i|_Q, s'|_Q), \ \ i = 1, 2, \ldots, n. \end{cases} \tag{7}$$

Solve (7) by linear programming to get a fractional solution $y_{j,a} = \bar{y}_{j,a}$ $(1 \le j \le |P|, a \in \Sigma)$ with cost $\bar{d}$. Clearly $\bar{d} \le d_0$. For each $1 \le j \le |P|$, independently, with probability $\bar{y}_{j,a}$, set $y'_{j,a} = 1$ and $y'_{j,a'} = 0$ for any $a' \in \Sigma - \{a\}$. Then we get a solution $y_{j,a} = y'_{j,a}$ $(1 \le j \le m$, $a \in \Sigma)$ for the 0-1 optimization problem (7), hence a solution $y = y'$ for (5).

For each $1 \le j \le |P|$, the above random rounding ensures that only one $a \in \Sigma$ is such that $y'_{j,a} = 1$. Therefore, $\sum_{a\in\Sigma} \chi(s_i[j], a)\, y'_{j,a}$ takes 1 or 0 randomly. Since the rounding is independent for different j's, $\sum_{a\in\Sigma} \chi(s_i[j], a)\, y'_{j,a}$ are independent 0-1 random variables for $1 \le j \le |P|$. Thus, for any $1 \le i \le n$, $d(s_i|_P, y') = \sum_{1\le j\le|P|} \sum_{a\in\Sigma} \chi(s_i[j], a)\, y'_{j,a}$ is a sum of $|P|$ independent 0-1 random variables, and

$$\begin{aligned} E[d(s_i|_P, y')] &= \sum_{1\le j\le|P|} \sum_{a\in\Sigma} \chi(s_i[j], a)\, E[y'_{j,a}] \\ &= \sum_{1\le j\le|P|} \sum_{a\in\Sigma} \chi(s_i[j], a)\, \bar{y}_{j,a} \\ &\le \bar{d} - d(s_i|_Q, s'|_Q) \le d_0 - d(s_i|_Q, s'|_Q). \end{aligned} \tag{8}$$

Therefore, for any fixed $\epsilon' > 0$, by Lemma 2,

$$\mathbf{Pr}\left(d(s_i|_P, y') \ge d_0 + \epsilon'|P| - d(s_i|_Q, s'|_Q)\right) \le \exp\left(-\frac{1}{3}\epsilon'^2|P|\right).$$

Considering all of the n strings, we have

$$\mathbf{Pr}\left(d(s_i|_P, y') \ge d_0 + \epsilon'|P| - d(s_i|_Q, s'|_Q) \text{ for some } 1 \le i \le n\right) \le n \times \exp\left(-\frac{1}{3}\epsilon'^2|P|\right). \tag{9}$$

If $|P| \ge (4\ln n)/\epsilon'^2$, then, $n \times \exp\left(-\frac{1}{3}\epsilon'^2|P|\right) \le n^{-\frac{1}{3}}$. Thus we obtain a randomized algorithm to find a solution for (5) with cost at most $d_0 + \epsilon'|P|$ with probability at least $1 - n^{-\frac{1}{3}}$.

For binary strings, Raghavan gave a method to derandomize the above randomized algorithm in [27]. Also see [18]. Here we give a method to derandomize the algorithm for any constant size alphabet.

Let $y_{j,a} = \bar{y}_{j,a}$ be a fractional solution for (7) for $j \in P$. We can arbitrarily decompose P into disjoint sets P_1, P_2, …, P_k such that $|P_i| \ge (4\ln n)/\epsilon'^2$ and $|P_i| < (8\ln n)/\epsilon'^2$ for $i = 1, 2, \ldots, k$. Let $\mu_{l,i} = \sum_{1\le j\le|P_i|} \sum_{a\in\Sigma} \chi(s_l[j], a)\, y_{j,a}$. For each P_i, replacing P with P_i in (9), we know that there exists a string x_i of length $|P_i|$ such that for each $s_l \in \mathcal{S}$,

$$d(s_l|_{P_i}, x_i) \le \mu_{l,i} + \epsilon'|P_i|.$$

Thus, we can simply enumerate all string of length $|P_i|$ to find out x_i in polynomial time for any constant size alphabet. Concatenating all x_i's, we have a string x of length $|P|$ such that for each string $s_l \in \mathcal{S}$,

$$d(s_l|_P, x) \le \sum_{i=1}^{k} (\mu_{l,i} + \epsilon'|P_i|) = d_0 - d(s_l|_Q, s'|_Q) + \epsilon'|P|.$$

Thus, we obtain a desired string x in polynomial time.

If $|P| < (4\ln n)/\epsilon'^2$, $|\Sigma|^{|P|} < n^{(4\ln|\Sigma|)/\epsilon'^2}$ is polynomial in n. So, we can enumerate all strings in $\Sigma^{|P|}$ to find an optimal solution for (5) in polynomial time. Thus, in both cases, we can obtain a solution $y = y_0$ for the optimization problem (5) with cost at most $d_0 + \epsilon'|P|$ in polynomial time. Since $|P| = O(d_{opt})$, $|P| \leq c \times d_{opt}$ for a constant c. Let $\epsilon' = \frac{\epsilon}{c}$. Let s^* be the string such that $s^*|_Q = s'|_Q$ and $s^*|_P = y_0$. ¿From (5),

$$\begin{aligned} d(s_i, s^*) &= d(s_i|_P, s^*|_P) + d(s_i|_Q, s^*|_Q) \\ &= d(s_i|_P, y_0) + d(s_i|_Q, s'|_Q) \\ &\leq d_0 + \epsilon'|P| \leq (1+\rho)d_{opt} + \epsilon d_{opt}, \end{aligned}$$

where the last inequality is from (6). This proves the lemma. □

Now we describe the complete algorithm in Figure 1.

Algorithm closestString

Input $s_1, s_2, \ldots, s_n \in \Sigma^m$, an integer $r \geq 2$ and a small number $\epsilon > 0$.

Output a center string $s \in \Sigma^m$.

1. **for** each r-element subset $\{s_{i_1}, s_{i_2}, \ldots, s_{i_r}\}$ of the n input strings **do**
   (a) $Q = \{1 \leq j \leq m \mid s_{i_1}[j] = s_{i_2}[j] = \ldots = s_{i_r}[j]\}$, $P = \{1, 2, \ldots, m\} - Q$.
   (b) Let $s' = s_{i_1}$. Solve the optimization problem defined by (5) as described in the proof of Lemma 7 to get an approximate solution $y = y_0$ within error $\epsilon\, d_{opt}$.
   (c) Let u be a string such that $u|_Q = s_{i_1}|_Q$ and $u|_P = y_0$. Calculate the radius of the solution with u as the center string.
2. **for** $i = 1, 2, \ldots, n$ **do**
   calculate the radius of the solution with s_i as the center string.
3. Output the best solution of the above two steps.

Figure 1: Algorithm for CLOSEST STRING

Theorem 8 *The algorithm closestString is a PTAS for* CLOSEST STRING.

Proof. Given an instance of CLOSEST STRING, suppose s is an optimal solution and the optimal radius is d_{opt}, i.e. $\max_{i=1,\ldots,n} d(s, s_i) = d_{opt}$. Let P be defined as in step 1(a) of Algorithm closestString. Since for every position in P, at least one of the r strings $s_{i_1}, s_{i_2}, \ldots, s_{i_r}$ conflicts with the optimal center string s, we have $|P| \leq r \times d_{opt}$. As far as r is a constant, step 1(b) can be done in polynomial time by Lemma 7. Obviously the other steps of Algorithm closestString runs in polynomial time, with r a constant. In fact, it can be easily verified that the algorithm runs in time $O\left((nm)^r\, n^{O\left(\log|\Sigma| \times r^2/\epsilon^2\right)}\right)$.

If $\rho_0 - 1 \leq \frac{1}{2r-1}$, then by the definition of ρ_0, the algorithm finds a solution with radius at most $\rho_0 d_{opt} \leq (1 + \frac{1}{2r-1}) d_{opt}$ in step 2.

If $\rho_0 > 1 + \frac{1}{2r-1}$, then from Lemma 3 and Lemma 7, the algorithm finds a solution with radius at most $(1 + \frac{1}{2r-1} + \epsilon) d_{opt}$ in step 1.

Therefore, the performance ratio of Algorithm closestString is $(1 + \frac{1}{2r-1} + \epsilon) d_{opt}$. This proves the theorem. $\square$

3 Approximating CLOSEST SUBSTRING when d is small

In some applications such as drug target identification, genetic probe design, the radius d is often small. As a direct application of Lemma 3, we now present a PTAS for CLOSEST STRING when the radius d is small, i.e., $d \leq O(\log(nm))$. Again, we focus on the construction of the center string.

Suppose $\langle \mathcal{S}, L \rangle$ is the given instance, where $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$. Let s be an optimal center string and t_i be the substring from s_i which is the closest to s $(1 \leq i \leq n)$.

We first describe the intuition behind the proof. The basic idea is that for any fixed constant $r > 0$, by trying all the choices of r substrings from $\mathcal{S}$, we can assume that we know the t_{i_1}, t_{i_2}, ..., t_{i_r} that satisfy Lemma 3 by replacing s_i by t_i and s_{i_j} by t_{i_j}. Since $d \leq O(\log(nm))$, the r substrings t_{i_1}, t_{i_2}, ..., t_{i_r} disagree at at most $O(\log(nm))$ positions. Therefore, we can keep the characters at the positions where t_{i_1}, t_{i_2}, ..., t_{i_r} all agree, and try all possibilities for the rest of the positions. From Lemma 3, we will find a good approximation to s. The complete algorithm is described in Figure 2:

Theorem 9 *Algorithm smallSubstring is a PTAS for* CLOSEST SUBSTRING *when the radius d is small, i.e., $d \leq O(\log(nm))$.*

Proof. Obviously, the size of P in Step 1 is at most $O(r \times \log(nm))$. Step 1 takes $O\left((mn)^r \times \Sigma^{O(r \times \log(nm))} \times mnL\right) = O\left((nm)^{r+1} \times (nm)^{O(r \times \log|\Sigma|)}\right) = O\left((nm)^{O(r \times \log|\Sigma|)}\right)$ time. Other steps take less than that time. Thus, the total time required is $O\left((nm)^{O(r \times \log|\Sigma|)}\right)$, which is polynomial in terms of input size for any constant r.

¿From Lemma 3, the performance ratio of the algorithm is $1 + \frac{1}{2r-1}$. $\square$

4 A PTAS for CLOSEST SUBSTRING

In this section, we further extend the algorithm for CLOSEST STRING to a PTAS for CLOSEST SUBSTRING, making use of a *random sampling* strategy. Note that Algorithm smallSubstring runs in exponential time for general radius d and the algorithm closestString does not work for CLOSEST SUBSTRING since we do not know how to construct an optimization

Algorithm smallSubstring

Input $s_1, s_2, \ldots, s_n \in \Sigma^m$, an integer L and an integer $r > 0$.

Output a center string $s \in \Sigma^L$.

1. **for** each r-element subset $\{t_{i_1}, t_{i_2}, \ldots, t_{i_r}\}$, where t_{i_j} is a substring of length L from s_{i_j} **do**
   (a) $Q = \{1 \le j \le L \mid t_{i_1}[j] = t_{i_2}[j] = \ldots = t_{i_r}[j]\}$, $P = \{1, 2, \ldots, L\} - Q$.
   (b) **for** every $x \in \Sigma^{|P|}$ **do**
   let t be the string such that $t|_Q = t_{i_1}|_Q$ and $t|_P = x|_P$; compute the radius of the solution with t as the center string.
2. **for** every length L substring u from any given sequence **do**
   compute the radius of the solution with u as the center string.
3. select a center string that achieves the best result in Step 1 and Step 2; output the best solution of the above two steps.

Figure 2: Algorithm for Closest Substring when d is small

problem similar to (5) — The construction of (5) requires us to know all the n strings (substrings) in an optimal solution of Closest String (Closest Substring). Thus, the choice of a "good" substring from every string s_i is the only obstacle on the way to the solution. For each $s_i \in \mathcal{S}$, there are $O(m)$ ways to choose t_i. Thus, we have $O(m^n)$ possible combinations, which is exponential in terms of n. We use random sampling to handle this.

Now let us outline the main ideas. Let $\langle \mathcal{S} = \{s_1, s_2, \ldots, s_n\}, L\rangle$ be an instance of Closest Substring, where each s_i is of length m, for $1 \le i \le n$. Suppose that s, of length L, is its optimal solution and t_i is a length L substring of s_i which is the closest to s ($1 \le i \le n$). Let $d_{opt} = \max_{i=1}^{n} d(s, t_i)$. The key step of our algorithm is to construct a function $f(u)$ for u being a length L substring from any $s_i \in \mathcal{S}$, such that $f(u)$ approximates $d(s, u)$ very well. Therefore, we can find an "approximation" to each t_i by minimizing $f(u)$ for u being a substring from s_i ($1 \le i \le n$).

More specifically, for any fixed $r > 0$, by trying all the choices of r substrings from $\mathcal{S}$, we can assume that $t_{i_1}, t_{i_2}, \ldots, t_{i_r}$ are the r substrings that satisfy Lemma 3 by replacing s_i by t_i and s_{i_j} by t_{i_j}. Let Q be the set of positions where $t_{i_1}, t_{i_2}, \ldots, t_{i_r}$ agree and $P = \{1, 2, \ldots, L\} - Q$. By Lemma 3, $t_{i_1}|_Q$ is a good approximation to $s|_Q$. We want to approximate $s|_P$ by the solution of y of the following optimization problem (10), where t'_i is a substring of s_i and is up to us to choose.

$$\begin{cases} \min \;\; d; \\ d(t'_i|_P, y) \le d - d(t'_i|_Q, t_{i_1}|_Q), \;\; i = 1, \cdots, n; |y| = |P|. \end{cases} \tag{10}$$

Note that t_i''s are not variables of problem (10), but rather they define the constraints. We need to choose them properly to construct the right problem (10) to solve. The ideal choice is $t_i' = t_i$, *i.e.*, t_i' is the closest to s among all substrings of s_i. However, we only approximately know the corresponding characters of s in Q and know nothing about those positions of s in P so far. So, we randomly pick $O(\log(mn))$ positions from P. Suppose the multiset of these random positions is R. (Here we use multiset, since each time we have to independently choose a position from P without changing the distribution. Thus, we can estimate the expected value of the $|R|$ independent random variables and apply the Chernoff bound and Lemma 2 in our proof.) By trying all length $|R|$ strings, we can assume that we know $s|_R$. Then for each $1 \le i \le n$, we find the substring t_i' from s_i such that $f(t_i') = d(s|_R, t_i'|_R) \times \frac{|P|}{|R|} + d(t_{i_1}|_Q, t_i'|_Q)$ is minimized. Then t_i' potentially belongs to the substrings which are the closest to s.

Then we solve (10) approximately by the method provided in the proof of Lemma 7 to get a solution $y = y_0$ and combine y_0 at the positions in P and t_{i_1} at the positions in Q. The resulting string should be a good approximation to s. The detailed algorithm (Algorithm closestSubstring) is given in Figure 3. We prove Theorem 10 in the rest of the section.

Theorem 10 *Algorithm closestSubstring is a PTAS for the closest substring problem.*

Proof. Let s be an optimal center string and t_i be the length-L substring of s_i that is the closest to s ($1 \le i \le n$). Let $d_{opt} = \max_{1\le i\le n} d(s, t_i)$. Let ϵ be any small positive number and $r \ge 2$ be any fixed integer. Let $\rho_0 = \max_{1\le i,j\le n} d(t_i, t_j)/d_{opt}$. If $\rho_0 \le 1 + \frac{1}{2r-1}$, then clearly we can find a solution s' within ratio ρ_0 in step 2. So, we assume that $\rho_0 \ge 1 + \frac{1}{2r-1}$ from now on.

Fact 1 There exists a group of substrings $t_{i_1}, t_{i_2}, \ldots, t_{i_r}$ chosen in step 1 of Algorithm closestSubstring, such that for any $1 \le l \le n$, $d(t_l|_Q, t_{i_1}|_Q) - d(s_l|_Q, s|_Q) \le \frac{1}{2r-1} d_{opt}$, where Q is defined in step 1(a) of the algorithm.

Proof. The fact follows from Lemma 3 directly. □

Obviously, the algorithm takes x as $s|_R$ at some point in step 1(c). Let $x = s|_R$ and $t_{i_1}, t_{i_2}, \ldots, t_{i_r}$ satisfy Fact 1. Let P and Q be defined as in step 1(a), and t_i' be defined as in step 1(c)(i). Let s^* be a string such that $s^*|_P = s|_P$ and $s^*|_Q = t_{i_1}|_Q$. Then we claim:

Fact 2 With high probability, $d(s^*, t_i') \le d(s^*, t_i) + 2\epsilon|P|$ for all $1 \le i \le n$.

Proof. Recall, for any position multiset T, we denote $d^T(t_1, t_2) = d(t_1|_T, t_2|_T)$ for any two strings t_1 and t_2. Let $\rho = \frac{|P|}{|R|}$. Consider any length L substring t' of s_i satisfying

$$d(s^*, t') \ge d(s^*, t_i) + 2\epsilon|P|. \tag{11}$$

Let $f(u) = \rho d^R(s^*, u) + d^Q(t_{i_1}, u)$ for any length L string u. The fact $f(t') \le f(t_i)$ implies either $f(t') \le d(s^*, t') - \epsilon|P|$ or $f(t_i) \ge d(s^*, t_i) + \epsilon|P|$. Thus, we have the following

Algorithm closestSubstring

Input $s_1, s_2, \ldots, s_n \in \Sigma^m$, an integer $1 \leq L \leq m$, an integer $r \geq 2$ and a small number $\epsilon > 0$.

Output the center string s.

1. **for** each r-element subset $\{t_{i_1}, t_{i_2}, \ldots, t_{i_r}\}$, where t_{i_j} is a substring of length L from s_{i_j} **do**
   (a) $Q = \{1 \leq j \leq L \,|\, t_{i_1}[j] = t_{i_2}[j] = \ldots = t_{i_r}[j]\}$, $P = \{1, 2, \ldots, L\} - Q$.
   (b) Let R be a multiset containing $\lceil \frac{4}{\epsilon^2} \log(nm) \rceil$ uniformly random positions from P.
   (c) **for** every string x of length $|R|$ **do**
      (i) **for** i from 1 to n **do**
         Let t_i' be a length L substring of s_i minimizing $f(t_i') = d(x, t_i'|_R) \times \frac{|P|}{|R|} + d(t_{i_1}|_Q, t_i'|_Q)$.
      (ii) Using the method provided in the proof of Lemma 7, solve the optimization problem defined by (10) approximately to get a solution $y = y_0$ within error $\epsilon |P|$.
      (iii) Let s' be the string such that $s'|_P = y_0$ and $s'|_Q = t_{i_1}|_Q$. Let $c = \max_{i=1}^n \min_{\{t_i \text{ is a substring of } s_i\}} d(s', t_i)$.
2. **for** every length-L substring s' of s_1 **do**
   Let $c = \max_{i=1}^n \min_{\{t_i \text{ is a substring of } s_i\}} d(s', t_i)$.
3. Output the s' with minimum c in step 1(c)(iii) and step 2.

Figure 3: The PTAS for the closest substring problem.

inequality:

$$
\begin{aligned}
& \mathbf{Pr}\left(f(t') \leq f(t_i)\right) \\
\leq \;& \mathbf{Pr}\left(f(t') \leq d(s^*, t') - \epsilon|P|\right) + \mathbf{Pr}\left(f(t_i) \geq d(s^*, t_i) + \epsilon|P|\right). \qquad (12)
\end{aligned}
$$

It is easy to see that $d^R(s^*, t')$ is the sum of $|R|$ independent random 0-1 variables $\sum_{i=1}^{|R|} X_i$, where $X_i = 1$ indicates a mismatch between s^* and t' at the i-th position in R. Let $\mu = E[d^R(s^*, t')]$. Obviously, $\mu = d^P(s^*, t')/\rho$. Therefore, by Lemma 2 (2),

$$
\begin{aligned}
& \mathbf{Pr}\left(f(t') \leq d(s^*, t') - \epsilon|P|\right) \\
= \;& \mathbf{Pr}\left(\rho\, d^R(s^*, t') + d^Q(s^*, t') \leq d(s^*, t') - \epsilon|P|\right) \\
= \;& \mathbf{Pr}\left(\rho\, d^R(s^*, t') \leq d^P(s^*, t') - \epsilon|P|\right) \\
= \;& \mathbf{Pr}\left(d^R(s^*, t') \leq \mu - \epsilon|R|\right)
\end{aligned}
$$

$$\leq \exp\left(-\frac{1}{2}\epsilon^2|R|\right) \leq (nm)^{-2}, \tag{13}$$

where the last inequality is due to the setting $|R| = \lceil \frac{4}{\epsilon^2}\log(nm)\rceil$ in step 1(b) of the algorithm. Similarly, using Lemma 2 (1) we have

$$\mathbf{Pr}\left(f(t_i) \geq d(s^*, t_i) + \epsilon|P|\right) \leq (nm)^{-\frac{4}{3}}. \tag{14}$$

Combining (12), (13), and (14), we know that for any t' that satisfies (11),

$$\mathbf{Pr}\left(f(t') \leq f(t_i)\right) \leq 2\,(nm)^{-\frac{4}{3}}. \tag{15}$$

For any fixed $1 \leq i \leq n$, there are less than m substrings t' that satisfy (11). Thus, from (15) and the definition of t_i',

$$\mathbf{Pr}\left(d(s^*, t_i') \geq d(s^*, t_i) + 2\epsilon|P|\right) \leq 2\,n^{-\frac{4}{3}}m^{-\frac{1}{3}}. \tag{16}$$

Summing up all $i \in \{1, 2, \ldots, n\}$, we know that with probability at least $1 - 2\,(nm)^{-\frac{1}{3}}$, $d(s^*, t_i') \leq d(s^*, t_i) + 2\epsilon|P|$ for all $i = 1, 2, \ldots, n$. □

¿From Fact 1, $d(s^*, t_i) = d^P(s, t_i) + d^Q(t_{i_1}, t_i) \leq d(s, t_i) + \frac{1}{2r-1}\, d_{opt}$. Combining with Fact 2 and $|P| \leq r\, d_{opt}$, we get

$$d(s^*, t_i') \leq (1 + \frac{1}{2r-1} + 2\epsilon\, r)d_{opt}. \tag{17}$$

By the definition of s^*, $s|_P$ is a solution for the optimization problem defined by (10) such that $d \leq (1 + \frac{1}{2r-1} + 2\epsilon\, r)d_{opt}$. We can solve the optimization problem (10) within error $\epsilon\,|P| \leq \epsilon\, r d_{opt}$ by the method in the proof of Lemma 7. Let $y = y_0$ be the approximate solution of the optimization problem in (10). Then by (10), for any $1 \leq i \leq n$,

$$d(t_i'|_P, y_0) \leq (1 + \frac{1}{2r-1} + 2\epsilon\, r)d_{opt} - d(t_i'|_Q, t_{i_1}|_Q) + \epsilon\, r d_{opt}. \tag{18}$$

Let s' be defined in step 1(c)(iii), then by (18),

$$\begin{aligned} d(s', t_i') &= d(y_0, t_i'|_P) + d(t_{i_1}|_Q, t_i'|_Q) \\ &\leq (1 + \frac{1}{2r-1} + 2\epsilon\, r)d_{opt} + \epsilon\, r d_{opt} \\ &\leq (1 + \frac{1}{2r-1} + 3\epsilon\, r)d_{opt}. \end{aligned}$$

For any small constant $\delta > 0$, by setting $r = \frac{1}{\delta} + \frac{1}{2}$ and $\epsilon = \frac{\delta^2}{6}$, we assure that $\frac{1}{2r-1} + 3\epsilon r \leq \delta$. Therefore, with high probability, Algorithm closestSubstring outputs in polynomial time a center string s' such that $d(t_i', s')$ is no more than $(1+\delta)d_{opt}$ for every $1 \leq i \leq n$, where t_i' is a substring of s_i. It can be easily verified that the running time of the algorithm is $O\left((n^2m)^{O\left((\log|\Sigma|)/\delta^4\right)}\right)$.

The algorithm can be derandomized by standard methods. For instance, instead of randomly and independently choosing $O(\log\ (nm))$ positions from P, we can pick the vertices encountered on a random walk of length $O(\log\ (nm))$ on a constant degree expander [7]. Obviously, the number of such random walks on a constant degree expander is polynomial in terms of nm. Thus, by enumerating all random walks of length $O(\log\ (nm))$, we have a polynomial time deterministic algorithm. (Also see [1].) $\square$

5 Discussions

We have designed polynomial time approximation schemes for both the closest string and closest substring problems. A more general problem called *distinguishing string selection* problem is to search for a string of length L which is "close" (as an approximate substring) to some sequences (of harmful germs) and "far" from some other sequences (of humans). See [13] for a formal definition. The problem is originated from genetic drug target identification, and universal PCR primer design [12, 4, 21]. A trivial ratio-2 approximation was given in [13] and the first nontrivial algorithm with approximation ratio $2 - \frac{2}{2|\Sigma|+1}$ was given in [14]. An open problem is whether a polynomial time approximation scheme exists for this problem.

Acknowledgements

We would like to thank Tao Jiang, Kevin Lanctot, Joe Wang, and Louxin Zhang for discussions and suggestions on related topics.

Ming Li was supported in part by the NSERC Research Grant OGP0046506, a CGAT grant, a CITO grant, the E.W.R. Steacie Fellowship, and the NSF/ITR Grant ACI00-85801. Bin Ma was supported in part by the NSERC Research Grant OGP0046506 and the HK RGC Grant 9040444. Lusheng Wang was fully supported by the HK RGC Grant 9040444.

References

[1] S. Arora, D. Karger, M. Karpinski, Polynomial time approximation schemes for dense instances of NP-hard problems, *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, pp. 284-293, 1995.

[2] A. Ben-Dor, G. Lancia, J. Perone, and R. Ravi, Banishing bias from consensus sequences, *Proc. 8th Ann. Combinatorial Pattern Matching Conf.*, pp. 247-261, 1997.

[3] P. Berman, D. Gumucio, R. Hardison, W. Miller, and N. Stojanovic, A linear-time algorithm for the 1-mismatch problem, *Workshops on Algorithms and Data Structures*, pp. 126-135, 1997.

[4] J. Dopazo, A. Rodríguez, J. C. Sáiz, and F. Sobrino, Design of primers for PCR amplification of highly variable genomes, *CABIOS*, 9, pp. 123-125, 1993.

[5] M. Frances, A. Litman, On covering problems of codes, *Theor. Comput. Syst.*, 30, 113-119, 1997.

[6] L. Gąsieniec, J. Jansson, and A. Lingas, Efficient approximation algorithms for the Hamming center problem, *Proc. 10th ACM-SIAM Symp. on Discrete Algorithms*, pp. S905-S906, 1999.

[7] D. Gillman, A chernoff bound for random walks on expanders, *Proceedings of the* 34^{th} *FOCS*, pp. 680-691, 1993.

[8] D. Gusfield, Efficient methods for multiple sequence alignment with guaranteed error bounds, *Bull. Math. Biol.*, 30, pp. 141-154, 1993.

[9] D. Gusfield, *Algorithms on Strings, Trees, and Sequences*, Cambridge Univ. Press, 1997.

[10] G. Hertz and G. Stormo, Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps, *Proc. 3rd Int'l Conf. Bioinformatics and Genome Research*, pp. 201-216, 1995.

[11] C. Lawrence and A. Reilly, An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences, *Proteins*, 7, pp. 41-51, 1990.

[12] K. Lucas, M. Busch, S. Mössinger and J.A. Thompson, An improved microcomputer program for finding gene- or gene family-specific oligonucleotides suitable as primers for polymerase chain reactions or as probes, *CABIOS*, 7, pp. 525-529, 1991.

[13] K. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang, Distinguishing string selection problems, *Proc. 10th ACM-SIAM Symp. on Discrete Algorithms*, pp. 633-642, 1999. Also to appear in *Information and Computation.*

[14] M. Li, B. Ma, and L. Wang, Finding Similar Regions in Many Strings, *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, pp. 473-482, Atlanta, 1999.

[15] M. Li, B. Ma, and L. Wang, Finding Similar Regions in Many Sequences, *J. Comput. Syst. Sci.* special issue for *Thirty-first Annual ACM Symposium on Theory of Computing*, to appear.

[16] C. Liang, M. Li, B. Ma, COPIA – a Consensus Pattern Identification and Analysis software system. Manuscript, 2001. Software available at: http://dna.cs.ucsb.edu/copia/copia_submit.html.

[17] B. Ma, A polynomial time approximation scheme for the closest substring problem, *Proc. 11th Annual Symposium on Combinatorial Pattern Matching*, pp. 99-107, Montreal, (2000).

[18] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge Univ. Press, 1995.

[19] P.A. Pevzner, *Computational molecular biology – an algorithmic approach.* The MIT Press, 2000.

[20] J. Posfai, A. Bhagwat, G. Posfai, and R. Roberts, Predictive motifs derived from cytosine methyltransferases, *Nucl. Acids Res.*, 17, pp. 2421-2435, 1989.

[21] V. Proutski and E. C. Holme, Primer Master: a new program for the design and analysis of PCR primers, *CABIOS*, 12, pp. 253-255, 1996.

[22] G. D. Schuler, S. F. Altschul, and D. J. Lipman, A workbench for multiple alignment construction and analysis, *Proteins: Structure, Function and Genetics*, 9, pp. 180-190, 1991.

[23] G. Stormo, Consensus patterns in DNA, in R.F. Doolittle (ed.), *Molecular evolution: computer analysis of protein and nucleic acid sequences*, *Methods in Enzymology*, 183, pp. 211-221, 1990.

[24] G. Stormo and G.W. Hartzell III, Identifying protein-binding sites from unaligned DNA fragments. *Proc. Natl. Acad. Sci. USA*, 88, pp. 5699-5703, 1991.

[25] M. Waterman, Multiple sequence alignment by consensus, *Nucl. Acids Res.*, 14, pp. 9095-9102, 1986.

[26] M. Waterman, R. Arratia and D. Galas, Pattern recognition in several sequences: consensus and alignment, *Bull. Math. Biol.*, 46, pp. 515-527, 1984.

[27] P. Raghavan, Probabilistic construction of deterministic algorithms: Approximate packing integer programs, *Journal of Computer and System Sciences*, 37(2), pp. 130-143, 1988.

[28] M. Waterman and M. Perlwitz, Line geometries for sequence comparisons, *Bull. Math. Biol.*, 46, pp. 567-577, 1984.