SIAM J. COMPUT.
Vol. 32, No. 4, pp. 1073–1090

GENETIC DESIGN OF DRUGS WITHOUT SIDE-EFFECTS*

XIAOTIE DENG†, GUOJUN LI‡, ZIMAO LI§, BIN MA¶, AND LUSHENG WANG†

Abstract. Consider two sets of strings, $\mathcal{B}$ (bad genes) and $\mathcal{G}$ (good genes), as well as two integers d_b and d_g ($d_b \leq d_g$). A frequently occurring problem in computational biology (and other fields) is to find a (distinguishing) substring s of length L that distinguishes the bad strings from good strings, i.e., such that for each string $s_i \in \mathcal{B}$ there exists a length-L substring t_i of s_i with $d(s, t_i) \leq d_b$ (close to bad strings), and for every substring u_i of length L of every string $g_i \in \mathcal{G}$, $d(s, u_i) \geq d_g$ (far from good strings).

We present a polynomial time approximation scheme to settle the problem; i.e., for any constant $\epsilon > 0$, the algorithm finds a string s of length L such that for every $s_i \in \mathcal{B}$ there is a length-L substring t_i of s_i with $d(t_i, s) \leq (1 + \epsilon)d_b$, and for every substring u_i of length L of every $g_i \in \mathcal{G}$, $d(u_i, s) \geq (1 - \epsilon)d_g$ if a solution to the original pair ($d_b \leq d_g$) exists. Since there is a polynomial number of such pairs (d_b, d_g), we can exhaust all the possibilities in polynomial time to find a good approximation required by the corresponding application problems.

Key words. approximation algorithms, computational molecular biology, distinguishing substring selection

AMS subject classifications. 68W25, 90C27, 92D20

DOI. 10.1137/S0097539701397825

1. Introduction.

Research effort in molecular biology, such as the human genome project, has been revealing the secret of our genetic composition, the long DNA sequences that can determine many aspects of life. Applications that use this information have posed new challenges to the design and analysis of efficient computational methods.

A frequently recurring problem in biological applications is to find one substring of length L that appears (with a few substitutions) at least once in each of a set of bad strings (such as bacterial sequences) and is not "close" to any substring of length L in *each* of another set of good strings (such as human and livestock sequences). The problem has various applications in molecular biology such as the design of universal PCR primers, identification of genetic drug targets, and design of genetic probes [8, 2, 12, 5]. In particular, the genetic drug target identification problem searches for a sequence of genes that is close to bad genes (the target) but far from all good genes (to avoid side-effects). Our study develops a polynomial time approximation scheme in both measures simultaneously.

*Received by the editors November 12, 2001; accepted for publication (in revised form) April 18, 2003; published electronically July 11, 2003.

http://www.siam.org/journals/sicomp/32-4/39782.html

†Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong (deng@cs.cityu.edu.hk, lwang@cs.cityu.edu.hk). The first author was supported by a grant from the National Natural Science Foundation of China and Research Grants Council of the HKSAR Joint Research Scheme (project N_CityU 102/01). The last author was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (project CityU 1047/01E).

‡School of Mathematics and System Sciences, Shandong University, Jinan 250100, P. R. China, and Institute of Software, Chinese Academy of Science, Beijing 100080, P. R. China (gjli@sdu.edu.cn).

§School of Computer Science and Technology, Shandong University, Jinan 250100, P. R. China (lizm@sdu.edu.cn). This author was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (project CityU 1047/01E).

¶Department of Computer Science, University of Western Ontario, London, ON N6A 5B7, Canada (bma@csd.uwo.ca).

1.1. The mathematical model. The *distinguishing substring selection* problem has as input two sets of strings, $\mathcal{B}$ and $\mathcal{G}$. It is required to find a substring of unspecified length (denoted by L) that is, informally, close to a substring of every string in $\mathcal{B}$ and far away from every length-L substring of strings in $\mathcal{G}$. Since $\mathcal{G}$ can be reconstructed to contain all substrings of length L in each of all the good strings, we can assume that every string in $\mathcal{G}$ has the same length L.

Therefore, here we formally define the problem as follows: Given a set $\mathcal{B} = \{s_1, s_2, \ldots, s_{n_1}\}$ of n_1 (bad) strings of length at least L, a set $\mathcal{G} = \{g_1, g_2, \ldots, g_{n_2}\}$ of n_2 (good) strings of length exactly L, and two integers d_b and d_g ($d_b \leq d_g$), the distinguishing substring selection problem (DSSP) is to find a string s such that for each string $s_i \in \mathcal{B}$ there exists a length-L substring t_i of s_i with $d(s, t_i) \leq d_b$, and for any string $g_i \in \mathcal{G}$, $d(s, g_i) \geq d_g$. Here $d(,)$ represents the Hamming distance between two strings. If all strings in $\mathcal{B}$ are also of the same length L, the problem is called the distinguishing string problem (DSP).

1.2. Previous results. The best previous known approximation ratio for DSSP (and DSP) is 2 [5]. In another direction, many simplified versions were proposed to make the task easier in terms of computation/approximation, conceding more general applicability.

The problem is NP-hard, even when only one objective is to be met [1, 5]. To approximate the center string that is far away from each of the good strings is not difficult and is shown to have a polynomial time approximation scheme by Lanctot et al. [5]. The problem of finding a center string s that is close to all the bad strings is nontrivial, and it has taken intensive investigation by several research groups [5, 3] to finally develop a polynomial time approximation scheme [6].

There are also several modifications and variations on this initial strategy, e.g., the *d-characteristic string* [4], the *far from most string* [5], and the formulation of Ben-Dor et al. [1].

The DSSP problem that contains two objective functions has remained elusive despite the above-mentioned intensive research effort. To meet the requirements of many application problems, however, we would need to have a solution for DSSP.

1.3. Our contribution. In this paper, we settle this difficult problem by presenting a polynomial time approximation scheme in both requirements. If a solution to the original problem exists, for any constant $\epsilon > 0$, our algorithm computes a string s of length L such that for every $s_i \in \mathcal{S}$ there is a length-L substring t_i of s_i with $d(t_i, s) \leq (1 + \epsilon)d_b$, and for every $g_i \in \mathcal{G}$, $d(g_i, s) \geq (1 - \epsilon)d_g$. Here d_b and d_g are two important parameters supplied by users to specify the distances from the distinguishing string s to the set of bad strings and the set of good strings. Our algorithm gives a solution that approximately meets the requirements, i.e., $(1 + \epsilon)d_b$ and $(1 - \epsilon)d_g$, if a solution to the original requirements, i.e., d_b and d_g, exists.

1.4. Sketch of our approach. Design and analysis of good algorithms for approximating multiple objective functions are not simple in general (see [11] for a general approach and related works).

The standard techniques for related center string problems follow a linear program approach combined with randomized rounding. That works for DSP and DSSP when the parameters are sufficiently large. The main difficulty for our problem lies in the case when objective function value is relatively small but still too large for enumeration methods to work.

To overcome this difficulty, we sample a constant number of strings and find the

positions at which the samples have the same letters (bases). We denote this set of positions by Q and the set of the rest by P. Our breakthrough starts with a key lemma that states that there is a set of samples for which there are y positions in Q such that, when we choose the y positions in Q carefully, choose the letters of the rest of $|Q|-y$ positions to be the same as the samples, and choose letters at positions in P by the linear program approach, then we can obtain the right approximate solution.

An interesting case is when $y < O(\log(n))$, i.e., y is small, but not small enough for a brute-force enumeration method to go through directly. A new method is designed to handle this case. Since similar situations occur in many combinatorial optimization problems, we expect that this idea may have wider application.

1.5. Organization of the paper. We focus on the polynomial time approximation scheme for DSP in sections 2 through 4. Section 2 introduces the related notation and the standard integer programming formulation, which works well when both d_b and d_g are large, i.e., at least $\Omega(L)$. Section 3 gives the key lemma. We discuss the methods for finding a good approximation of the set of y positions in Q in section 4.

In section 5, we show that the established algorithm for DSP can be extended to work for the general case, the DSSP. We conclude our work in section 6 with discussion and remarks.

2. Preliminaries. We consider two sets of strings: $\mathcal{G}$ (good strings) and $\mathcal{B}$ (bad strings). We call d_b the *upper* radius for bad strings ($\mathcal{B}$) and d_g the *lower* radius for good strings ($\mathcal{G}$). Let $n = n_1 + n_2$ be the total number of good and bad strings in $\mathcal{G} \cup \mathcal{B}$.

For the DSP, every good or bad string is of the same length L. The distance $d(x, y)$ of two strings x and y is their Hamming distance, i.e., the number of positions at which they differ from each other. We are to find the string x of length L such that

$$\begin{cases} d(s_i, x) \le d_b, \quad s_i \in \mathcal{B}, & i = 1, \ldots, n_1, \\ d(g_j, x) \ge d_g, \quad g_j \in \mathcal{G}, & j = 1, \ldots, n_2. \end{cases} \tag{1}$$

In this section, we present an approximation algorithm that works well for a special case of the DSP. The restriction is that L, d_b, and d_g are large; more specifically, $L \ge (4\log(n_1 + n_2))/\epsilon^2$, $d_g = \Omega(L)$, and $d_b = \Omega(L)$, where ϵ is the parameter controlling the performance ratio and n_1 and n_2 are the numbers of bad and good strings, respectively.

This is achieved via a standard method using integer linear programming. Define $\chi(s, i, k, a) = 0$ if $s_i[k] = a$, and $\chi(s, i, k, a) = 1$ if $s_i[k] \ne a$. Similarly, define $\chi(g, j, k, a) = 0$ if $g_j[k] = a$, and $\chi(g, j, k, a) = 1$ if $g_j[k] \ne a$. The problem becomes the following integer linear programming (LP) problem:

$$\begin{cases} \sum_{a \in \Sigma} x_{k,a} = 1, \quad k = 1, 2, \ldots, L, \\ \sum_{1 \le k \le L} \sum_{a \in \Sigma} \chi(s, i, k, a)\, x_{k,a} \le d_b, \quad i = 1, 2, \ldots, n_1, \\ \sum_{1 \le k \le L} \sum_{a \in \Sigma} \chi(g, j, k, a)\, x_{k,a} \ge d_g, \quad j = 1, 2, \ldots, n_2, \\ x_{k,a} \in \{0, 1\}, \quad a \in \Sigma, \ k = 1, 2, \ldots, L. \end{cases} \tag{2}$$

Let $\bar{x}_{k,a}$ be a solution for LP relaxation of (2). For each $0 \le k \le L$, with probability $\bar{x}_{k,a}$, we set $x_{k,a} = 1$ and set $x_{k,a'} = 0$ for any $a' \ne a$. We choose the random variables independently for different k. This results in an integer solution for (2) (and hence a solution for (1)) if d_b is replaced by $(1 + \epsilon)d_b$ and d_g by $(1 - \epsilon)d_g$, as shown in Theorem 2. Standard derandomization methods [10] transfer it to a

deterministic algorithm in Theorem 3. The following lemma is useful for the proof of Theorem 2.

LEMMA 1. *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 $0 < \delta \leq 1$, $\mathbf{Pr}(X > \mu + \delta n) < \exp(-\frac{1}{3}n\delta^2)$, and $\mathbf{Pr}(X < \mu - \delta n) \leq \exp(-\frac{1}{2}n\delta^2)$.*

THEOREM 2. *Let $\delta > 0$ be any constant. Suppose that $L \geq (4\log(n_1 + n_2))/\delta^2$, $d_g \geq c_g L$, and $d_b \geq c_b L$. There is a randomized algorithm that finds a solution, i.e., a string x of length L, with high probability such that for each string s_i in $\mathcal{B}$, $d(s_i, x) < (1 + \delta/c_b)d_b$, and for any string g_j in $\mathcal{G}$, $d(g_j, x) > (1 - \delta/c_g)d_g$.*

Proof. We can see that $\sum_{a\in\Sigma} \chi(s, i, k, a)\, x_{k,a}$ and $\sum_{a\in\Sigma} \chi(g, j, k, a)\, x_{k,a}$ take 1 or 0 randomly and independently for different k's. Thus we have that $d(s, i, x) = \sum_{1\leq k\leq L} \sum_{a\in\Sigma} \chi(s, i, k, a)\, x_{k,a}$ is a sum of L independent 0-1 random variables. Similarly, $d(g, j, x) = \sum_{1\leq k\leq L} \sum_{a\in\Sigma} \chi(g, j, k, a)\, x_{k,a}$ is also a sum of L independent 0-1 random variables. Moreover,

$$\begin{aligned} E[d(s, i, x)] &= \sum_{1\leq k\leq L} \sum_{a\in\Sigma} \chi(s, i, k, a)\, E[x_{k,a}] \\ &= \sum_{1\leq k\leq L} \sum_{a\in\Sigma} \chi(s, i, k, a)\, \bar{x}_{k,a} \\ &\leq \bar{d} \leq d_b, \end{aligned} \tag{3}$$

$$\begin{aligned} E[d(g, j, x)] &= \sum_{1\leq k\leq L} \sum_{a\in\Sigma} \chi(g, j, k, a)\, E[x_{k,a}] \\ &= \sum_{1\leq k\leq L} \sum_{a\in\Sigma} \chi(g, j, k, a)\, \bar{x}_{k,a} \\ &\geq d_g. \end{aligned} \tag{4}$$

Thus, for any fixed $\delta > 0$, using Lemma 1, we have

$$\mathbf{Pr}\left(d(s, i, x) > d_b + \delta L\right) < \exp\left(-\frac{1}{3}\delta^2 L\right),$$

$$\mathbf{Pr}\left(d(g, j, x) < d_g - \delta L\right) \leq \exp\left(-\frac{1}{2}\delta^2 L\right).$$

Considering all bad and good strings, respectively, we have

$$\mathbf{Pr}\left(d(s, i, x) > d_b + \delta L \text{ for at least one } s_i \in \mathcal{B}\right) < n_1 \times \exp\left(-\frac{1}{3}\delta^2 L\right)$$

and

$$\mathbf{Pr}\left(d(g, j, x) < d_g - \delta L \text{ for at least one } g_j \in \mathcal{G}\right) \leq n_2 \times \exp\left(-\frac{1}{2}\delta^2 L\right).$$

When $L \geq (4\log(n_1 + n_2))/\delta^2$, we get $n_1 \times \exp(-\frac{1}{3}\delta^2 L) \leq n_1^{-\frac{1}{3}}$ and $n_2 \times \exp(-\frac{1}{2}\delta^2 L) \leq n_2^{-1}$. For $d_b = \Omega(L)$ and $d_g = \Omega(L)$, there exist positive constants c_b and c_g such that $d_b \geq c_b L$ and $d_g \geq c_g L$. Thus we get a randomized algorithm to find a

solution x for (1) with probability at least $1-(n_1^{-\frac{1}{3}}+n_2^{-1})$ such that for $i=1,2,\ldots,n_1$, $d(s,i,x) \leq (1+\delta/c_b)d_b$, and for $j=1,2,\ldots,n_2$, $d(g,j,x) \geq (1-\delta/c_g)d_g$. □

THEOREM 3. *There exists a polynomial time approximation scheme (PTAS) for a DSP when $d_g = \Omega(L)$ and $d_b = \Omega(L)$. Specifically, when $d_g = c_g \times L$ and $d_b = c_b \times L$, the PTAS runs in $O(L \times |\Sigma|^{(4\log(n_1+n_2))/\delta^2})$ time and finds a solution, i.e., a string x of length L, such that for each string s_i in $\mathcal{B}$, $d(s_i,x) < (1+\epsilon)d_b$, and for any string g_j in $\mathcal{G}$, $d(g_j,x) > (1-\epsilon)d_g$, where $\delta = \epsilon \times \min\{c_g, c_b\}$.*

Proof. When $L \geq (4\log(n_1+n_2))/\delta^2$, the following derandomized algorithm is a PTAS.

Let $x_{j,a}$ be a fractional solution for (2) for $j=1,2,\ldots,L$. We can arbitrarily decompose the L positions into disjoint sets of positions $L_1, L_2, \ldots, L_k$ such that $|L_i| \geq (4\log(n_1+n_2))/\delta^2$ and $|L_i| < (8\log(n_1+n_2))/\delta^2$ for $i=1,2,\ldots,k$. Let $\mu_{s,l,i} = \sum_{1\leq j\leq |L_i|}\sum_{a\in\Sigma}\chi(s,l,j,a)x_{j,a}$ and $\mu_{g,l,i} = \sum_{1\leq j\leq |L_i|}\sum_{a\in\Sigma}\chi(g,l,j,a)x_{j,a}$. For each L_i, replacing $\{1,2,\ldots,L\}$ with L_i in (2), we know that there exists a string x_i of length $|L_i|$ such that for each string $s_l \in \mathcal{B}$,

$$d(s_l|_{L_i}, x_i) \leq \mu_{s,l,i} + \delta|L_i|,$$

and for each string $g_l \in \mathcal{G}$,

$$d(g_l|_{L_i}, x_i) \geq \mu_{g,l,i} - \delta|L_i|.$$

Thus, we can enumerate all strings of length $|L_i|$ in $|\Sigma|^{(4\log(n_1+n_2))/\delta^2}$ time to determine x_i. Concatenating all x_i's, we have a string x of length L such that for each string $s_l \in \mathcal{B}$,

$$d(s_l,x) \leq \sum_{i=1}^{k}(\mu_{s,l,i} + \delta|L_i|) = d_b + \delta L \leq d_b + \epsilon d_b,$$

and for each string $g_l \in \mathcal{G}$,

$$d(g_l,x) \geq \sum_{i=1}^{k}(\mu_{l,i} - \delta|L_i|) = d_g - \delta L \geq d_g - \epsilon d_g.$$

Thus, we obtain a desired string x in $O(L \times |\Sigma|^{(4\log(n_1+n_2))/\delta^2})$ time, which is polynomial in terms of the input size when $|\Sigma|$ is a constant.

When $L \leq (4\log(n_1+n_2))/\delta^2$, we can enumerate all possible strings of length L in $O(|\Sigma|^{(4\log(n_1+n_2))/\delta^2})$ time. Thus, we can get a desired solution. □

3. A key lemma. To obtain our PTAS algorithm, we need to introduce two parameters: an integer r and a positive number δ. The constant r is introduced in this section and depends purely on ϵ. The constant $\delta > 0$ will be introduced in the next section and depends on both $\epsilon > 0$ and r.

For any $r > 2$, let $1 \leq i_1, i_2, \ldots, i_r \leq n_1$ be r distinct numbers. Let $Q_{i_1,i_2,\ldots,i_r}$ be the set of positions at which $s_{i_1}, s_{i_2}, \ldots, s_{i_r}$ agree. Let s be a feasible solution of length L for the distinguishing string selection problem with the upper radius d_b for bad strings ($\mathcal{B}$) and lower radius d_g for good strings ($\mathcal{G}$). $P_{i_1,i_2,\ldots,i_r} = \{1,2,\ldots,L\} - Q_{i_1,i_2,\ldots,i_r}$ is the set of positions at which at least one pair of strings s_{i_j} and $s_{i_{j'}}$ differ. Then, at such positions, s differs from at least one of s_{i_j}. The number of positions at which s differs from one of s_{i_j} is no more than d_b. Therefore, the total number of

positions at which s differs from at least one of s_{i_j} $(j = 1, 2, \ldots, r)$ is no more than rd_b. It follows that $rd_b \geq |P_{i_1,i_2,\ldots,i_r}| = L - |Q_{i_1,i_2,\ldots,i_r}|$. That is, $|Q_{i_1,i_2,\ldots,i_r}| \geq L - rd_b$. In addition, we immediately obtain the following result, which is often applied late in the proof of our main result.

CLAIM 4. *Assuming that $d_g \geq d_b$, we have $rd_g \geq rd_b \geq L - |Q_{i_1,i_2,\ldots,i_r}| = |P_{i_1,i_2,\ldots,i_r}|$.*

Let $\mathcal{B} = \{s_1, s_2, \ldots, s_{n_1}\}$. 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 $\rho_k = \min_{1 \leq i_1,i_2,\ldots,i_k \leq n_1} p_{i_1,i_2,\ldots,i_k}/d_b$. The following claim is a variant of a claim in [6], but the proof is identical.

CLAIM 5. *For any k such that $2 \leq k \leq r$, where r is a constant, there are indices $1 \leq i_1, i_2, \ldots, i_r \leq n_1$ such that for any $1 \leq l \leq n_1$*

$$|\{j \in Q_{i_1,i_2,\ldots,i_r} \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \leq (\rho_k - \rho_{k+1})\, d_b.$$

Proof. Consider 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

$$\begin{aligned}
&|\{j \in Q_{i_1,i_2,\ldots,i_r} \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \\
&\leq |\{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \qquad (5)\\
&= |\{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] \neq s[j]\} \\
&\quad - \{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] = s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \\
&= |\{j \in Q_{i_1,i_2,\ldots,i_k} \mid s_{i_1}[j] \neq s[j]\} - \{j \in Q_{i_1,i_2,\ldots,i_k,l} \mid s_{i_1}[j] \neq s[j]\}| \\
&= p_{i_1,i_2,\ldots,i_k} - p_{i_1,i_2,\ldots,i_k,l} \qquad (6)\\
&\leq (\rho_k - \rho_{k+1})\, d_{opt},
\end{aligned}$$

where inequality (5) holds because $Q_{i_1,i_2,\ldots,i_r} \subseteq Q_{i_1,i_2,\ldots,i_k}$ and equality (6) holds because $Q_{i_1,i_2,\ldots,i_k,l} \subseteq Q_{i_1,i_2,\ldots,i_k}$. ∎

From Claim 5, we can immediately obtain the following proposition.

PROPOSITION 6. *For any constant r, there are indices $1 \leq i_1, i_2, \ldots, i_r \leq n_1$ such that for any $s_l \in \mathcal{B}$,*

$$|\{j \in Q_{i_1,i_2,\ldots,i_r} \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \leq \frac{1}{r-1}\, d_b.$$

Proof. Note that

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

Therefore, one of $(\rho_k - \rho_{k+1})$ is at most $\frac{1}{r-1}$. Thus, Claim 5 immediately implies that the lemma is true. ∎

The following corollary will also be useful later.

COROLLARY 7. *For any $Q \subseteq Q_{i_1,i_2,\ldots,i_r}$ (as in Proposition 6), we have that, for any $s_l \in \mathcal{B}$,*

$$|\{j \in Q \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \leq \frac{1}{r-1}\, d_b.$$

Moreover, let $P = \{1, 2, \ldots, L\} - Q$. Then, for any $s_l \in \mathcal{B}$,

$$d(s_l, s_{i_1}|Q) + d(s_l, s|P) \leq \frac{r}{r-1} d_b.$$

Proof. The first part is obvious by Proposition 6. For the second part, consider

$$A_k = \{j \in Q \mid s_{i_1}[j] \neq s_k[j] \text{ and } s_{i_1}[j] \neq s[j]\}$$

and

$$B_k = \{j \in Q \mid s_{i_1}[j] \neq s_k[j] \text{ and } s_{i_1}[j] = s[j]\}.$$

Then $d(s_k, s_{i_1}|Q) = |A_k| + |B_k|$. By the first part of the corollary, $|A_k|$ is no more than $\frac{1}{r-1}d_b$. By definition, $|B_k| \leq d(s_k, s|Q)$. Therefore, $|B_k| + d(s_k, s|P) \leq d(s_k, s|Q) + d(s_k, s|P) = d(s_k, s) \leq d_b$. The rest of the corollary follows. □

Informally, Proposition 6 implies that s_{i_1} is a good approximation of s at positions in $Q_{i_1,i_2,\ldots,i_r}$ for the bad strings; i.e., for any $Q \subseteq Q_{i_1,i_2,\ldots,i_r}$,

$$d(s_l, s_{i_1}|Q) \leq d(s_l, s|Q) + \frac{1}{r-1}\, d_b.$$

Before we present our key lemma, we need a boosting proposition that, when applied together with Corollary 7, obtains a better and better solution.

PROPOSITION 8. *Let $Q \subset Q_{i_1,i_2,\ldots,i_r}$ (as in Proposition 6). Consider the index $k : 1 \leq k \leq n_1$ and the number $y \geq 0$ such that, for any $s_l \in \mathcal{B}$, $d(s_l, s_{i_1}|Q) + d(s_l, s|P) \leq d(s_k, s_{i_1}|Q) + d(s_k, s|P) = \frac{r}{r-1}d_b - y$ (where $P = \{1, 2, \ldots, L\} - Q$). Then we have*

$$|\{j \in Q^k | s_{i_1}[j] \neq s[j]\}| \leq y,$$

where $Q^k = \{j \in Q : s_{i_1}[j] = s_k[j]\}$.

Proof. Dividing $d(s_k, s_{i_1}|Q)$ into two parts, we have

$$d(s_k, s_{i_1}|Q - Q^k) + d(s_k, s_{i_1}|Q^k) + d(s_k, s|P) = \frac{r}{r-1}d_b - y. \tag{7}$$

By Corollary 7 and with the further restriction that $s_l[j] = s[j]$, we have that, for any string $s_l \in \mathcal{B}$,

$$|\{j \in Q \mid s_{i_1}[j] \neq s_l[j] \text{ and } s_{i_1}[j] \neq s[j] \text{ and } s_l[j] = s[j]\}| \leq \frac{1}{r-1}\, d_b.$$

That is, for any string $s_l \in \mathcal{B}$,

$$|\{j \in (Q - Q^l) | s_l[j] = s[j]\}| \leq \frac{1}{r-1}\, d_b. \tag{8}$$

From (8), there exists β with $\beta \leq 1$ such that for any $s_l \in \mathcal{B}$, $|\{j \in (Q - Q^l) \mid s_l[j] = s[j]\}| \leq |\{j \in (Q - Q^k) \mid s_k[j] = s[j]\}| = \beta\frac{1}{r-1}\, d_b$. On the other hand, $|\{j \in (Q - Q^k) \mid s_k[j] \neq s_{i_1}[j]\}|$ is no more than $|\{j \in (Q - Q^k) \mid \}|$, which is the sum of $|\{j \in (Q - Q^k) \mid s_k[j] = s[j]\}|$ and $|\{j \in (Q - Q^k) \mid s_k[j] \neq s[j]\}|$. Combining those two formulae, we have

$$d(s_k, s|Q - Q^k) \geq d(s_k, s_{i_1}|Q - Q^k) - \beta\frac{1}{r-1}\, d_b. \tag{9}$$

Moreover, combining (7) and (9), we have

$$\begin{aligned} d(s_k, s|Q - Q^k) + \frac{\beta}{r-1}\, d_b + d(s_k, s_{i_1}|Q^k) + d(s_k, s|P) \\ \geq \frac{r}{r-1}d_b - y. \end{aligned} \tag{10}$$

Since

$$d(s_k, s|Q - Q^k) + d(s_k, s|Q^k) + d(s_k, s|P) \le d_b, \tag{11}$$

then (10) implies

$$\begin{aligned} d(s_k, s|Q^k) &\le d(s_k, s_{i_1}|Q^k) + \frac{\beta - 1}{r - 1} d_b + y \\ &= \frac{\beta - 1}{r - 1} d_b + y \le y. \end{aligned} \tag{12}$$

Here (12) is from the facts that $d(s_k, s_{i_1}|Q^k) = 0$ and $\beta \le 1$. Therefore,

$$|\{j \in Q^k | s_{i_1}[j] \ne s[j]\}| \le y. \tag{13}$$

This completes the proof. ∎

Here is our main lemma.

LEMMA 9. *For any constant r, there exist indices $1 \le i_1, i_2, \ldots, i_t \le n_1$, $r \le t \le 2r$, and a number $0 \le y \le d_b$ such that $|\{j \in Q_{i_1,i_2,\ldots,i_t} | s_{i_1}[j] \ne s[j]\}| \le y$. (Note that this implies $d(g, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) \ge d(g, s|Q_{i_1,i_2,\ldots,i_t}) - y$.) If we use s_{i_1} as the distinguishing string at positions in $Q_{i_1,i_2,\ldots,i_t}$, and s (the original distinguishing string) at the rest of the positions, then for any $s_l \in \mathcal{B}$, $d(s_l, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) + d(s_l, s|P_{i_1,i_2,\ldots,i_t}) \le \frac{r+1}{r-1} d_b - y$, where $P_{i_1,i_2,\ldots,i_t} = \{1, 2, \ldots, L\} - Q_{i_1,i_2,\ldots,i_t}$.*

Proof. We shall repeatedly apply Proposition 8 up to r times for the proof.

We start with $Q = Q_{i_1,i_2,\ldots,i_r}$. By Corollary 7, for any $s_l \in \mathcal{B}$, $d(s_l, s_{i_1}|Q) - |\{j \in Q \,|\, s_{i_1}[j] \ne s_l[j] \text{ and } s_{i_1}[j] = s_l[j]\}| \le \frac{1}{r-1} d_b$. Notice that $|\{j \in Q \,|\, s_{i_1}[j] \ne s_l[j] \text{ and } s_{i_1}[j] = s_l[j]\}| \le d(s_l, s \,|\, Q)$. We get

$$d(s_l, s_{i_1}|Q) + d(s_l, s|P)| \le \frac{1}{r - 1} d_b + d(s_l, s) \le \frac{r}{r - 1} d_b.$$

Therefore, there exist an index k (denoted by i_{r+1}) and number z^0 ($\frac{r}{r-1} d_b \ge z^0 \ge 0$) such that for any $s_l \in \mathcal{B}$,

$$d(s_l, s_{i_1}|Q) + d(s_l, s|P)| \le d(s_{i_{r+1}}, s_{i_1}|Q) + d(s_{i_{r+1}}, s|P)| = \frac{1}{r - 1} d_b - z^0.$$

By Proposition 8, $|\{j \in Q_{i_1,i_2,\ldots,i_r,i_{r+1}} | s_{i_1}[j] \ne s[j]\}| \le z^0$. Now apply Corollary 7 again to the index set $Q' = Q_{i_1,i_2,\ldots,i_{r+1}}$. Then, for any $s_l \in \mathcal{B}$, $d(s_l, s_{i_1}|Q_{i_1,i_2,\ldots,i_{r+1}}) + d(s_l, s|\{1, \ldots, L\} - Q_{i_1,i_2,\ldots,i_{r+1}}) \le \frac{r}{r-1} d_b$. Again, choose k' (and denote it by i_{r+2} later) and $y' \ge 0$ such that

$$d(s_l, s_{i_1}|Q_{i_1,i_2,\ldots,i_{r+1}}) + d(s_l, s|\{1, \ldots, L\} - Q_{i_1,i_2,\ldots,i_{r+1}}) \le \frac{r}{r - 1} d_b - y'$$

and such that equality holds for $l = k'$.

If $y' \ge z^0 - \frac{1}{r-1} d_b$, then our lemma follows with $t = r + 1$ and $y = z^0$. Therefore, we need to consider only the case $y' \le z^0 - \frac{1}{r-1} d_b$. Notice that if $z^0 < \frac{1}{r-1} d_b$, we will reach a contradiction here. We should also have $z^0 \ge \frac{1}{r-1} d_b$, or the lemma holds.

Define $z^1 = y'$. The conditions of Proposition 8 hold with $y = z^1$ and $Q = Q_{i_1,i_2,\ldots,i_{r+1}}$. Therefore, we are able to repeat the above process. Notice that $z^1 \le z^0 - \frac{1}{r-1} d_b$, and it is no smaller than $\frac{1}{r-1} d_b$. The same holds each time we repeat the

process. Therefore, the process can only be repeated up to r times before we obtain the result as claimed. □

In Lemma 9, we decompose the L positions into two parts, $Q_{i_1,i_2,\ldots,i_t}$ and $P_{i_1,i_2,\ldots,i_t}$. In $P_{i_1,i_2,\ldots,i_t}$, either the linear programming approach in section 2 or a brute-force enumeration method can go through since $|P_{i_1,i_2,\ldots,i_t}| \leq rd_b$. Details will be given in the next section. Thus, we can get a good approximation solution at positions in $P_{i_1,i_2,\ldots,i_t}$.

By Lemma 9, at positions in $Q_{i_1,i_2,\ldots,i_t}$ there are at most y positions where the letters for a feasible (optimal) solution are different from the letters in s_{i_1}. This implies that $d(g, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) \geq d(g, s|Q_{i_1,i_2,\ldots,i_t}) - y$. We may need to carefully choose y positions in $Q_{i_1,i_2,\ldots,i_t}$ in s_{i_1} to change to s'_{i_1} so that the condition $d(g, s'_{i_1}|Q_{i_1,i_2,\ldots,i_t}) \geq d(g, s|Q_{i_1,i_2,\ldots,i_t})$ is satisfied (with negligible error) for each $g \in \mathcal{G}$.

On the other hand, again by Lemma 9, if any y positions chosen in $Q_{i_1,i_2,\ldots,i_t}$ are changed to obtain a string s'_{i_1}, i.e, $d(s_{i_1}, s'_{i_1}) = y$, then $d(s_l, s|P_{i_1,i_2,\ldots,i_t}) + d(s_l, s'_{i_1}|Q_{i_1,i_2,\ldots,i_t}) \leq (1+\frac{1}{r-1})d_b$. That is, the total error produced for every string in $\mathcal{B}$ will be at most $\frac{1}{r-1}d_b$ no matter which y positions in $Q_{i_1,i_2,\ldots,i_t}$ are chosen. Therefore, to choose y positions in $Q_{i_1,i_2,\ldots,i_t}$, we can simply consider the good strings in $\mathcal{G}$ and ignore the strings in $\mathcal{B}$. This dramatically reduces the difficulty of the problem.

The main contribution of Lemma 9 is to transform the original DSP into the problem of how to choose y positions in $Q_{i_1,i_2,\ldots,i_t}$ to modify the string s_{i_1} into a string s' such that $d(g, s'_{i_1}|Q_{i_1,i_2,\ldots,i_t}) \geq d(g, s|Q_{i_1,i_2,\ldots,i_t})$. Though we do not know the value of y, we can enumerate all y: $y = 0, 1, \ldots, d_b$ in the algorithm. We will elaborate on this in the next section.

4. Choice of a distinguishing string. In general, we aim at constructing an approximate solution s'_{i_1} that differs from s_{i_1} at y positions (up to negligible error) in $Q_{i_1,i_2,\ldots,i_t}$. Though we do not know the value of y, we can enumerate all y: $y = 0, 1, \ldots, d_b$ in the algorithm. However, if $d_g \geq (r-1)y$, we can simply use s_{i_1} at all positions in $Q_{i_1,i_2,\ldots,i_t}$. For good strings $g \in \mathcal{G}$, the error at positions in $Q_{i_1,i_2,\ldots,i_t}$ will be $\frac{1}{r-1}d_g$, which will be good enough for a PTAS. For bad strings $s_l \in \mathcal{B}$, the error created by making s_{i_1} the approximate solution at positions in $Q_{i_1,i_2,\ldots,i_t}$ will be at most $\frac{2}{r-1}d_b$ (again, good enough for a PTAS) by Lemma 9. Therefore, we can assume that $d_g < (r-1)y$.

For any string $g_i \in \mathcal{G}$, if $d(g_i, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) \geq (r-1)y$, then the selected y positions in $Q_{i_1,i_2,\ldots,i_t}$ will cause error by at most $\frac{1}{r-1}d_g$. In fact, let s'_{i_1} be the string obtained from s_{i_1} by changing y positions in $Q_{i_1,i_2,\ldots,i_t}$. Then $d(g_i, s'_{i_1}|Q_{i_1,i_2,\ldots,i_t}) \geq (r-2)y$. Recalling the assumption that $d_g < (r-1)y$, we get for any $g_i \in \mathcal{G}$,

$$d(g_i, s'_{i_1}) \geq d(g_i, s'_{i_1}|Q_{i_1,i_2,\ldots,i_t}) \geq \frac{r-2}{r-1}d_g = \left(1 - \frac{1}{r-1}\right)d_g.$$

Therefore, we have to consider only the strings $g_i \in \mathcal{G}$ with the restriction that $d(g_i, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) < (r-1)y$. (This condition can be verified in polynomial time for each $g \in \mathcal{G}$.) In summary, without loss of generality, we have the following.

Assumption 1. For every string $g_i \in \mathcal{G}$, $d(g_i, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) < (r-1)y$ and $d_g \leq (r-1)y$.

In the remaining part of the section, several different methods will be used to carefully select y positions in $Q_{i_1,i_2,\ldots,i_t}$, each dealing with one of the three cases. Let $L' = |Q_{i_1,i_2,\ldots,i_t}|$. The complete algorithm is given in Figure 1.

In addition to the constant integer r, we need to introduce another positive number $\delta_0 > 0$. We should first choose r to be sufficiently large but remain a constant.

Algorithm gdistString

Input $\mathcal{B} = \{s_1, s_2, \ldots, s_{n_1}\}$ and $\mathcal{G} = \{g_1, g_2, \ldots, g_{n_2}\}$.
Output a center string s of length L.

1. **if** Case 1 holds, **then** use the method in section 2 to solve the problem.
2. **for** each $2r$-element subset $\{s_{i_1}, s_{i_2}, \ldots, s_{i_{2r}}\}$ of the n_1 input strings in $\mathcal{B}$ **do**
   (a) $Q = \{1 \le j \le L \mid s_{i_1}[j] = s_{i_2}[j] = \cdots = s_{i_t}[j]\}$, $P = \{1, 2, \ldots, L\} - Q$.
   (b) For the positions in $Q_{i_1,i_2,\ldots,i_t}$, use the methods described in Cases 2 and 3 to obtain a string of length $|Q_{i_1,i_2,\ldots,i_t}|$. Notice that the algorithm described in Case 1 produces directly a string s' of length L which is a PTAS.
   (c) Solve the optimization problem at the positions in $|P|$ using either an LP formulation (randomized rounding approach) (when $|P| \ge O(\log n)$) or an exhaustive search method (when $|P| \le O(\log n)$) to get an approximate solution x of length $|P|$.
3. Output the best solution obtained above.

FIG. 1. *Algorithm for DSP.*

Then we should choose δ_0 to be sufficiently small to achieve the required bound $(1+\epsilon)$ (and $(1-\epsilon)$) for PTAS. Notice that, since δ_0 is chosen after r is fixed, it may be a function of r. However, since r is a constant, we can make δ_0 sufficiently small (as a function of r) and at the same time δ_0 remains a constant. Therefore, in some of the asymptotic notation (Ω and big-O) used in this section, the constant may be a function of r. However, since we should set the value of r to a sufficiently large constant first, both r and δ_0 are constant when we later set the value of $\delta_0 > 0$ sufficiently small.

Case 1. $L' \le (r-1)^2 y$ and $y \ge (4\log(n_1+n_2))/{\delta_0}^2$. In this case, it follows that $d_b \ge y \ge (8\log(n_1+n_2))/{\delta_0}^2$. Moreover, we should prove that $d_b \ge c_b L$ and $d_g \ge c_g L$, where $c_b = c_g = \min\{\frac{1}{4r}, \frac{1}{2(r-1)^2}\}$.

Because $L' \ge y$ and $L' \le (r-1)^2 y$, it follows that $y = eL'$ for some $e : \frac{1}{(r-1)^2} \le e \le 1$. Since $d_b \ge y$ and $d_b \le d_g \le (r-1)y$ (from Assumption 1), we have $d_b = xy$ for $1 \le x \le r-1$. Thus, $d_b = xy = xeL'$ and $\frac{1}{(r-1)^2} \le xe \le r-1$. Note that $L = L' + |P|$ and $|P| \le td_b$ by Claim 4. We consider two subcases. (a) $|P| \le 0.5L$. Then $L' \ge 0.5L$. Thus $d_b = xeL' \ge 0.5xeL$. Since $d_b \le L$, we have $d_b = c'L$ for some $c' : \frac{1}{2(r-1)^2} \le c' \le 0.5(r-1)$. (b) $|P| > 0.5L$. Then by Claim 4, $d_b \ge |P|/t \ge 0.5L/t \ge L/4r$. Recall that we also assume that $d_g \ge d_b$ in the problem setting. It follows that $d_g \ge d_b > c_g L$.

Since $d_b \ge c_b L$ and $d_g \ge c_g L$, we can directly use the method described in section 2.

Case 2. $L' > (r-1)^2 y$ and $y \ge (4\log(n_1+n_2))/{\delta_0}^2$. By Assumption 1, $d(g_i, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) < (r-1)y$. Thus, for any $g_i \in \mathcal{G}$, s_{i_1} and g_i agree at most of the positions (at least $(r-1)(r-2)$ out of $(r-1)^2$ positions) in $Q_{i_1,i_2,\ldots,i_t}$. Intuitively, if we randomly select y positions in $Q_{i_1,i_2,\ldots,i_t}$ and change them into any different letters from s_{i_1} to get y_{i_1}, then y_{i_1} and g_i should differ at most of the y positions. Therefore, we can apply the following simple randomized algorithm:

1. Arbitrarily divide the L' positions into y sets of positions $Y_1, Y_2, \ldots, Y_y$ such that $|Y_1| = |Y_2| = \cdots = |Y_{y-1}| = \lfloor \frac{L'}{y} \rfloor$.

2. For each Y_i, independently randomly select a position in Y_i.
3. Let $Y = \{j_1, j_2, \ldots, j_y\}$ be the y selected positions in step 2, and y_{i_1} be a string of length $|Q_{i_1,i_2,\ldots,i_t}|$ obtained from s_{i_1} by changing the letters at these positions in Y (to any different letters) such that $d(y_{i_1}, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) = y$.

This algorithm gives us the needed solution for Case 2. We have the following proposition.

PROPOSITION 10. *The randomized algorithm for Case* 2 *produces a center string* y_{i_1} *of length* $|Q_{i_1,i_2,\ldots,i_t}|$ *with high probability such that*

(i) *for any* $s_i \in \mathcal{B}$, $d(y_{i_1}, s_i|Q_{i_1,i_2,\ldots,i_t}) + d(s, s_i|P_{i_1,i_2,\ldots,i_t}) \leq \frac{r+1}{r-1} d_b$, *and*

(ii) *for any* $g_j \in \mathcal{G}$, $d(y_{i_1}, g_j|Q_{i_1,i_2,\ldots,i_t}) + d(s, g_j|P_{i_1,i_2,\ldots,i_t}) \geq (1 - \frac{2}{r-1} - 2\delta_0) d_g$, *where* s *is a feasible (optimal) distinguishing string.*

Proof. (i) arises directly from Lemma 9.

Now, we show that (ii) is true for the case where $\lfloor \frac{L'}{y} \rfloor = \frac{L'}{y}$. (The general case is left to interested readers.) Consider a string $g \in \mathcal{G}$. Let Z_i be the set of positions j in Y_i with $s_{i_1}[j] = g[j]$. Let X_i be a 0/1 random variable. $X_i = 1$ indicates that j_i is in Z_i. The probability that $X_i = 1$ is $\frac{|Z_i|}{|Y_i|}$. Then we get

$$\mu = E\left[\sum_{i=1}^{y} X_i\right] = \sum_{i=1}^{y} \frac{|Z_i|}{|Y_i|} = \frac{y}{L'} \sum_{i=1}^{y} |Z_i| \geq \frac{y(L' - (r-1)y)}{L'}.$$

The last inequality is by the fact from Assumption 1 that $d(g, s_{i_1}|Q_{i_1,i_2,\ldots,i_t}) < (r-1)y$. Since $L' > (r-1)^2 y$, $\mu \geq \frac{y(r-2)}{r-1}$. By Lemma 1,

$$\mathbf{Pr}\left(\sum_{i=1}^{y} X_i < \mu - \delta_0 y\right) < \exp\left(-\frac{1}{2}\delta_0^2 y\right) \leq n_2^{-2}.$$

The last inequality is from the fact that $y \geq 4\log(n_1 + n_2)/\delta_0^2$.

Let y_{i_1} be the string obtained from s_{i_1} by changing the letters at y positions using the randomized algorithm. Then for the string $g \in \mathcal{G}$,

$$d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) + \left(y - \sum_{i=1}^{y} X_i\right) - d(s_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) \geq \sum_{i=1}^{y} X_i.$$

The above inequality is due to the facts that (1) s_{i_1} and y_{i_1} differ at the y selected positions, (2) y_{i_1} and g differ at every position i with $X_i = 1$, and (3) y_{i_1} and g may agree at the rest of the $y - \sum_{i=1}^{y} X_i$ positions. Thus, we have

$$\frac{d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) - d(s_{i_1}, g_l|Q_{i_1,i_2,\ldots,i_t}) + y}{2} \geq \sum_{i=1}^{y} X_i.$$

Therefore, we get

$$\begin{aligned} & \mathbf{Pr}\left(\frac{d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) - d(s_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) + y}{2} < \mu - \delta_0 y\right) \\ \leq \quad & \mathbf{Pr}\left(\sum_{i=1}^{y} X_i < \mu - \delta_0 y\right) < \exp\left(-\frac{1}{2} y \delta_0^2\right) \leq n_2^{-2}. \end{aligned}$$

Substituting $\mu \geq \frac{y(r-2)}{r-1}$ into the above inequality, and after some calculation, we have

$$\mathbf{Pr}\left(d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) - d(s_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) \leq \frac{y(r-3)}{r-1} - 2\delta_0 y\right) \leq n_2^{-2}.$$

From Lemma 9, $d(s_i, g|Q_{i_1,i_2,\ldots,i_t}) \geq d(s, g||Q_{i_1,i_2,\ldots,i_t}) - y$, so we get

$$\mathbf{Pr}\left(d(s, g|Q_{i_1,i_2,\ldots,i_t}) - d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) > 2\delta_0 y + \frac{2y}{r-1}\right) \leq n_2^{-2}.$$

Considering all good strings, we get

$$\mathbf{Pr}\left(d(s, g|Q_{i_1,i_2,\ldots,i_t}) - d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) > 2\delta_0 y + \frac{2y}{r-1} \text{ for any } g \in \mathcal{G}\right) \leq n_2^{-1}.$$

Therefore, with high probability we get that

$$d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) \geq d(s, g|Q_{i_1,i_2,\ldots,i_t}) - \left(\frac{2}{r-1} + 2\delta_0\right) d_g,$$

that is,

$$\begin{aligned} & d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) + d(s, g|P_{i_1,i_2,\ldots,i_t}) \\ \geq \quad & d(s, g_l) - \left(\tfrac{2}{r-1} + 2\delta_0\right) d_g \geq d_g - \left(\tfrac{2}{r-1} + 2\delta_0\right) d_g = \left(1 - \tfrac{2}{r-1} - 2\delta_0\right) d_g. \qquad \square \end{aligned}$$

Now, we give a derandomized algorithm for Case 2.

LEMMA 11. *There is a deterministic algorithm that runs in time equal to* $O(d_b^2(n_1+n_2)^{\log(2(r-1)^2e)8/\delta_0^2})$ *and finds a string* y_{i_1} *from* s_{i_1} *such that for any* $g \in \mathcal{G}$,

$$d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_{2r}}) - d(s_{i_1}, g|Q_{i_1,i_2,\ldots,i_{2r}}) \geq \frac{r-3}{r-1} y - 2\delta_0 y.$$

Proof. Proposition 10 holds when $L' = (r-1)^2 y$, since in that case $\mu = \frac{y(r-2)}{r-1}$. Therefore, if $L' > (r-1)^2 y$, we can arbitrarily select a subset $Q \subset Q_{i_1,i_2,\ldots,i_t}$ of size $(r-1)^2 y$ and try to select y positions in Q.

Next, if y is big enough, i.e., $y = (4\log(n_1+n_2))/{\delta_0}^2$, then Proposition 10 holds. That is, a good approximation solution, i.e., a set of y positions, exists in Q. We can try all possible ways to choose y positions in Q. The time required is $C^y_{(r-1)^2 y}$. Note that

$$C_m^y = \frac{m(m-1)\cdots(m-y+1)}{y!} \leq \frac{m^y}{y!} \leq \left(\frac{m}{y}\right)^y e^y.$$

We have

$$C^y_{(r-1)^2 y} \leq ((r-1)^2)^y e^y = (n_1+n_2)^{\log((r-1)^2 e)4/\delta_0^2}.$$

Thus, the time required is at most $O((n_1+n_2)^{\log((r-1)^2 e)4/\delta_0^2})$ if $y = (4\log(n_1 + n_2))/{\delta_0}^2$.

When $y > (4\log(n_1+n_2))/{\delta_0}^2$, we cannot directly choose y positions from Q since the time complexity is higher. Let $y = x \times (4\log(n_1+n_2))/{\delta_0}^2 + z$, where $x \geq 1$ is an integer and $z < (4\log(n_1+n_2))/{\delta_0}^2$. We divide Q into x disjoint subsets $Q_1, Q_2, \ldots, Q_x$ such that $|Q_i| = (r-1)^y/x$ for $i = 1, 2, \ldots, x-1$ and $Q_x = Q - Q_1 - Q_2 - \cdots - Q_{x-1}$. Proposition 10 ensures that each Q_i contains a good approximation solution, i.e., a set of $(4\log(n_1+n_2))/{\delta_0}^2$ positions, such that y_1 and g differ at most of the positions (at least $\frac{(r-2)}{r-1}\%$). Thus, we can try all possible ways to choose the right $(4\log(n_1+n_2))/{\delta_0}^2$ positions from each Q_i. (We have to choose

$(4\log(n_1+n_2))/{\delta_0}^2 + z$ positions from Q_x.) Therefore, the time required for each Q_i is $O((n_1+n_2)^{\log(2(r-1)^2e)8/\delta_0^2})$. Since there are at most y Q_i's and we have to try different values of y, the total time required is $O(d_b^2(n_1+n_2)^{\log(2(r-1)^2e)8/\delta_0^2})$. □

Case 3. $y < (4\log(n_1+n_2))/{\delta_0}^2$. For any q good strings $g_{j_1}, g_{j_2}, \ldots, g_{j_q}$, we define

$$R_{j_1,j_2,\ldots,j_q} = \{j \in Q_{i_1,i_2,\ldots,i_t} \mid s_{i_1}[j] \neq g_{j_l}[j] \text{ for at least one } l \in \{1,2,\ldots,q\}\}$$

and

$$\bar{R}_{j_1,j_2,\ldots,j_q} = Q_{i_1,i_2,\ldots,i_t} - R_{j_1,j_2,\ldots,j_q}.$$

From the definition of $R_{j_1,j_2,\ldots,j_q}$ and $\bar{R}_{j_1,j_2,\ldots,j_q}$, we have that for any integers p, q $(p<q)$,

$$R_{j_1,j_2,\ldots,j_p} \subseteq R_{j_1,j_2,\ldots,j_q}, \qquad \bar{R}_{j_1,j_2,\ldots,j_q} \subseteq \bar{R}_{j_1,j_2,\ldots,j_p}.$$

Let $y_1 = d(s_{i_1}, s|R_{j_1,j_2,\ldots,j_r})$ and $y_2 = d(s_{i_1}, s|\bar{R}_{j_1,j_2,\ldots,j_r})$. Then $y_1 + y_2 \leq y$.

The following lemma is important in dealing with Case 3.

LEMMA 12. *There exist* r *indices* $1 \leq j_1, j_2, \ldots, j_r \leq n_2$ *such that for any* $g \in \mathcal{G}$,

$$|\{j \in \bar{R}_{j_1,j_2,\ldots,j_r} \mid s_{i_1}[j] \neq g[j] \text{ and } s_{i_1}[j] \neq s[j]\}| \leq \frac{y}{r}.$$

Proof. Take $g_{j_1} \in \mathcal{G}$, and let

$$U_{j_1} = \{j \in Q_{i_1,i_2,\ldots,i_t} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g_{j_1}[j]\}$$

such that $|U_{j_1}|$ is as large as possible. Clearly, $U_{j_1} \subseteq R_{j_1}$.

If $|U_{j_1}| \leq \frac{y}{r}$, let $j_2, \ldots, j_r$ be any other indices and $g \in \mathcal{G}$ be any good string; it follows that

$$\begin{array}{ll} & |\{j \in \bar{R}_{j_1,j_2,\ldots,j_r} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g[j]\}| \\ \leq & |\{j \in Q_{j_1,j_2,\ldots,j_t} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g_{j_1}[j]\}| \\ = & |U_{j_1}| \leq \frac{y}{r}. \end{array}$$

Thus we may assume that $|U_{j_1}| > \frac{y}{r}$. Take $g_{j_2} \in \mathcal{G}$, and let

$$U_{j_1,j_2} = \{j \in Q_{i_1,i_2,\ldots,i_t} - U_{j_1} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g_{j_2}[j]\}$$

such that $|U_{j_1,j_2}|$ is as large as possible. Clearly, $U_{j_1,j_2} \subseteq R_{j_1,j_2}$.

If $|U_{j_1,j_2}| \leq \frac{y}{r}$, let $j_3, \ldots, j_r$ be any other indices and $g \in \mathcal{G}$ be any good string; it follows that

$$\begin{array}{ll} & |\{j \in \bar{R}_{j_1,j_2,\ldots,j_r} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g[j]\}| \\ = & |\{j \in Q_{1_1,i_2,\ldots,i_t} - R_{j_1,j_2,\ldots,j_r} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g[j]\}| \\ \leq & |\{j \in Q_{1_1,i_2,\ldots,i_t} - U_{j_1} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g_{j_2}[j]\}| \\ = & |U_{j_1,j_2}| \leq \frac{y}{r}. \end{array}$$

The inequality comes from the fact $U_{j_1} \subseteq R_{j_1} \subseteq R_{j_1,j_2,\ldots,j_r}$ and the choice of g_{j_2}. So we have to consider only $|U_{j_1,j_2}| > \frac{y}{r}$ since otherwise we are done.

In general, we can get a list of sets of positions

$$U_{j_1}, U_{j_1,j_2}, \ldots, U_{j_1,j_2,\ldots,j_q},$$

where $U_{j_1,j_2,\ldots,j_q} = \{j \in Q_{1_1,i_2,\ldots,i_t} - U_{j_1,j_2,\ldots,j_{q-1}} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g_{j_q}[j]\}$ and $|U_{j_1,j_2,\ldots,j_q}|$ is as large as possible such that

- $U_{j_1,j_2,\ldots,j_s} \cap U_{j_1,j_2,\ldots,j_t} = \emptyset$, $1 \leq s \neq t \leq q$,
- $|U_{j_1,j_2,\ldots,j_p}| > \frac{y}{r}$, $p = 1, \ldots, q$,
- q is as large as possible.

Note that $U_{j_1} \cup U_{j_1,j_2} \cup \cdots \cup U_{j_1,j_2,\ldots,j_q} \subseteq \{j \in Q_{i_1,i_2,\ldots,i_t} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g_{j_l}[j] \text{ for at least one } l \in \{1,2,\ldots,q\}\} \subseteq \{j \in Q_{i_1,i_2,\ldots,i_t} \mid s_{i_1}[j] \neq s[j]\}$. Therefore, we get

$$\frac{qy}{r} < |U_{j_1} \cup U_{j_1,j_2} \cup \cdots \cup U_{j_1,j_2,\ldots,j_q}| \leq |\{j \in Q_{i_1,i_2,\ldots,i_t} \mid s_{i_1}[j] \neq s[j]\}| \leq y,$$

and thus $q < r$.

Finally, take $g_{j_{q+1}} \in \mathcal{G}$, and let

$$U_{j_1,j_2,\ldots,j_{q+1}} = \{j \in Q_{i_1,i_2,\ldots,i_t} - U_{j_1,j_2,\ldots,j_q} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g_{j_{q+1}}[j]\}$$

such that $|U_{j_1,j_2,\ldots,j_{q+1}}|$ is as large as possible. Clearly, $U_{j_1,j_2,\ldots,j_{q+1}} \subseteq R_{j_1,j_2,\ldots,j_{q+1}}$.

By the maximality of q, $|U_{j_1,j_2,\ldots,j_{q+1}}| \leq \frac{y}{r}$, let $j_{q+2}, \ldots, j_r$ be any other indices and $g \in \mathcal{G}$ be any good string; it follows that

$$\begin{array}{rl} & |\{j \in \bar{R}_{j_1,j_2,\ldots,j_r} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g[j]\}| \\ = & |\{j \in Q_{i_1,i_2,\ldots,i_t} - R_{j_1,j_2,\ldots,j_r} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g[j]\}| \\ \leq & |\{j \in Q_{i_1,i_2,\ldots,i_t} - U_{j_1,j_2,\ldots,j_q} \mid s_{i_1}[j] \neq s[j] \text{ and } s_{i_1}[j] \neq g_{j_{q+1}}[j]\}| \\ = & |U_{j_1,j_2,\ldots,j_{q+1}}| \leq \frac{y}{r}. \end{array}$$

Lemma 12 is now proved. □

Now, we consider the two parts $R_{j_1,j_2,\ldots,j_r}$ and $\bar{R}_{j_1,j_2,\ldots,j_r}$, respectively.

Assumption 1 and the definition of $R_{j_1,j_2,\ldots,j_r}$ ensure that

$$\begin{array}{rcl} |R_{j_1,j_2,\ldots,j_r}| & = & |\{j \in Q_{1_1,i_2,\ldots,i_t} \mid s_{i_1}[j] \neq g_{j_l}[j] \text{ for at least one } l \in \{1,2,\ldots,r\}\}| \\ & \leq & \sum_{l=1}^{r} |\{j \in Q_{1_1,i_2,\ldots,i_t} \mid s_{i_1}[j] \neq g_{j_l}[j]\}| \\ & = & \sum_{l=1}^{r} d(s_{i_1}, g_{j_l}|Q_{1_1,i_2,\ldots,i_t}) \leq r(r-1)y < r^2 y. \end{array}$$

Keep in mind that we are dealing with the case $y \leq (4\log(n_1+n_2))/{\delta_0}^2$. Thus, we have $|R_{j_1,j_2,\ldots,j_r}| \leq (4\log(n_1+n_2))/{\delta_0}^2$. We can try all possible ways to achieve the optimal distinguishing string s at the positions in $R_{j_1,j_2,\ldots,j_r}$. The time required is

$$\sum_{y_1=0}^{y} C_{r^2 y}^{y_1} \leq 2^{r^2 y} \leq O((n_1+n_2)^{4r^2/\delta_0^2}).$$

Thus we can assume that we know the optimal distinguishing string s at the positions in $R_{j_1,j_2,\ldots,j_r}$.

Now, let us focus on the positions in $\bar{R}_{j_1,j_2,\ldots,j_r}$. For any $g \in \mathcal{G}$, define $Set(g, s_{i_1})$ be the set of positions in $\bar{R}_{j_1,j_2,\ldots,j_r}$, where g and s_{i_1} do not agree. From Lemma 12, we know that there exists a set of y_2 positions $K_{y_2} = \{k_j \in \bar{R}_{j_1,j_2,\ldots,j_r} \mid s_{i_1}[k_j] \neq s[k_j]\}$ such that for any $g \in \mathcal{G}$, $|K_{y_2} \cap Set(g, s_{i_1})| \leq \frac{y}{r}$. Though we do not know the exact value of y_2, again we can guess it in $O(y)$ time. Thus, we can assume that y_2 is known.

Let $s_{i_1}(K_{y_2})$ be the string obtained from s_{i_1} by changing the letters at the positions in K_{y_2} to any different letters. From Lemma 12, we can see that for any $g \in \mathcal{G}$,

$$d(g, s_{i_1}(K_{y_2})|\bar{R}_{j_1,j_2,\ldots,j_r}) \geq d(g, s|\bar{R}_{j_1,j_2,\ldots,j_r}) - \frac{1}{r}y. \tag{14}$$

Now, the only remaining task is to select y_2 positions in $\bar{R}_{j_1,j_2,\ldots,j_r}$ to approximate the set K_{y_2}. Recall that $|\{j \in \bar{R}_{j_1,j_2,\ldots,j_r} \mid s_{i_1}[j] \neq s[j]\}| = y_2$. From this we have that for every good string $g \in \mathcal{G}$,

$$d(g, s_{i_1}|\bar{R}_{j_1,j_2,\ldots,j_r}) - d(g, s|\bar{R}_{j_1,j_2,\ldots,j_r}) \geq -y_2. \tag{15}$$

Next we are going to show that we can find a set K'_{y_2} of y_2 positions in $\bar{R}_{j_1,j_2,\ldots,j_r}$ in polynomial time such that for any $g \in \mathcal{G}$, $|K'_{y_2} \cap Set(g, s_{i_1})| \leq \frac{y}{r}$. If this is true, by changing the letters of s_{i_1} on the positions in K'_{y_2}, the distance between $s_{i_1}(K'_{y_2})$ and g has to increase at least $y_2 - \frac{y}{r}$. To do this, let $m \leq |\bar{R}_{j_1,j_2,\ldots,j_r}|$ be an integer and $\bar{R}_{j_1,j_2,\ldots,j_r}(m) \subseteq \bar{R}_{j_1,j_2,\ldots,j_r}$ be any subset of $\bar{R}_{j_1,j_2,\ldots,j_r}$ with m elements.

The following two lemmas are the keys to solving the problem.

LEMMA 13. *If $m \geq 2^{r^2} \times y \times (n+1)^{\frac{r}{y}} + \frac{y}{r}$, then there exists a set $K'_{y_2} \subseteq \bar{R}_{j_1,j_2,\ldots,j_r}(m)$ of y_2 positions such that for any $g \in \mathcal{G}$,*

$$|K'_{y_2} \cap Set(g, s_{i_1})| \leq \frac{y}{r}.$$

Proof. Consider a string g in $\mathcal{G}$. Let $K''_{y_2} \subseteq \bar{R}_{j_1,j_2,\ldots,j_r}(m)$ denote a set of y_2 positions. By Assumption 1, that $d(s_{i_1}, g|Q_{i_1,i_2,\ldots,i_t}) < ry$, for any fixed $g \in \mathcal{G}$, the number of different K''_{y_2}'s such that

$$|K''_{y_2} \cap Set(g, s_{i_1})| \geq \frac{y}{r}$$

is at most

$$\begin{aligned} C_{m-ry}^{y_2-\frac{y}{r}} C_{ry}^{\frac{y}{r}} + C_{m-ry}^{y_2-\frac{y}{r}-1} C_{ry}^{\frac{y}{r}+1} + \cdots + 1 &\leq C_{m-ry}^{y_2-\frac{y}{r}} (C_{ry}^{\frac{y}{r}} + C_{ry}^{\frac{y}{r}+1} + \cdots + 1) \\ &\leq C_{m-ry}^{y_2-\frac{y}{r}} 2^{ry}. \end{aligned} \tag{16}$$

There are at most $n_2 \leq n_2 + n_1 = n$ strings in $\mathcal{G}$. Thus, if there is no $K''_{y_2} \subseteq \bar{R}_{j_1,j_2,\ldots,j_r}(m)$ such that

$$|K''_{y_2} \cap Set(g, s_{i_1})| \leq \frac{y}{r},$$

then

$$\begin{aligned} n_2 &\geq \frac{C_m^{y_2}}{2^{ry} C_{m-ry}^{y_2-\frac{y}{r}}} \\ &= \frac{m(m-1)\cdots(m-y_2+1)(y_2-\frac{y}{r})!}{y_2!(m-ry)(m-ry-1)\cdots(m-ry-y_2+\frac{y}{r}+1)2^{ry}} \\ &\geq \frac{m(m-1)\cdots(m-\frac{y}{r}+1)}{y_2(y_2-1)\cdots(y_2-\frac{y}{r}+1)2^{ry}} \\ &\geq \frac{(m-\frac{y}{r}+1)^{\frac{y}{r}}}{y^{\frac{y}{r}} \cdot 2^{ry}} > \frac{(m-\frac{y}{r})^{\frac{y}{r}}}{y^{\frac{y}{r}} 2^{ry}} = \frac{(m-\frac{y}{r})^{\frac{y}{r}}}{y^{\frac{y}{r}} (2^{r^2})^{\frac{y}{r}}} \\ &= \left(\frac{m-\frac{y}{r}}{2^{r^2} y}\right)^{\frac{y}{r}} \geq n+1. \end{aligned}$$

The last inequality is from the fact that $m \geq 2^{r^2} \times y \times (n+1)^{\frac{r}{y}} + \frac{y}{r}$. This is a contradiction. Thus, we can conclude that the lemma holds. ∎

Lemma 13 says that a good solution exists at the positions in $\bar{R}_{j_1,j_2,\ldots,j_r}(m)$ if $m = 2^{r^2} \times y \times (n+1)^{\frac{r}{y}} + \frac{y}{r}$. On the other hand, the following lemma shows that $m = 2^{r^2} \times y \times (n+1)^{\frac{r}{y}} + \frac{y}{r}$ is not too big so that we can find such a good solution in polynomial time by looking at all $C_m^{y_2}$ possible choices.

LEMMA 14. *If* $m = 2^{r^2} \times y \times (n+1)^{\frac{r}{y}} + \frac{y}{r}$ *and* $y \le (4\log(n_1+n_2))/{\delta_0}^2$, *then* C_m^y *is* $O((n_1+n_2)^{(4r^2+4+4\log e)/\delta_0^2+r}$.

Proof.

$$C_m^y = \frac{m(m-1)\cdots(m-y+1)}{y!} \le \frac{m^y}{y!}$$

$$\le \left(\frac{m}{y}\right)^y e^y \le \left(\frac{2(m-\frac{y}{r})}{y}\right)^y e^y \tag{17}$$

$$= \left(\frac{2(m-\frac{y}{r})}{2^{r^2}y}\right)^y (2^{r^2})^y e^y = 2^y (n+1)^r (2^{r^2})^y e^y. \tag{18}$$

Since $y \le (4\log(n_1+n_2))/{\delta_0}^2$, from (18), C_m^y is $O((n_1+n_2)^{(4r^2+4+4\log e)/\delta_0^2+r}$. ∎

LEMMA 15. *For Case* 3, *the time required to select* y *positions in* $Q_{i_1,i_2,\ldots,i+2r}$ *is* $O(rLd_b^2(n_1+n_2)^{(4r^2+4+4\log e)/\delta_0^2+2r})$.

Proof. The time to deal with $R_{j_1,j_2,\ldots,j_r}$ is smaller than that of $\bar{R}_{j_1,j_2,\ldots,j_r}$. It requires $O((n_1+n_2)^{(4r^2+4+4\log e)/\delta_0^2+r})$ time to deal with an $\bar{R}_{j_1,j_2,\ldots,j_r}$. Obtaining an $\bar{R}_{j_1,j_2,\ldots,j_r}$ needs $O(n_2^r Lr)$ time. We also have to guess y and y_2 (thus y_1) in d_b^2 time. Thus, the total time required to select y positions in $Q_{i_1,i_2,\ldots,i+2r}$ is $O(rLd_b^2(n_1+n_2)^{(4r^2+4+4\log e)/\delta_0^2+2r})$. ∎

THEOREM 16. *There is a PTAS for the DSP. The PTAS runs in* $O(L \times |\Sigma|^{O(\log(n_1+n_2)(r-1)^6)})$ *time and finds a distinguishing string* s *such that for every* $s_i \in \mathcal{B}$, $d(s_i,s) \le (1+\frac{1}{r-1})d_b$, *and for every* $g_i \in \mathcal{G}$, $d(g_i,s) \ge (1-\frac{1}{r-1})d_g$ *if a solution to the original pair* $(d_b \le d_g)$ *exists.*

Proof. According to Theorem 3, step 1 in Algorithm gdistString (Figure 1) requires $O(L \times |\Sigma|^{4\log(n_1+n_2)/\delta^2})$ time, where $\delta = \epsilon \min\{\frac{1}{4r}, \frac{1}{2(r-1)^2}\}$, and the performance ratios are $(1+\epsilon)$ and $(1-\epsilon)$ for bad strings and good strings, respectively.

Consider step 2 of Algorithm gdistString. We discuss Cases 2 and 3 separately.

Case 2. Lemma 11 shows that for a fixed $Q_{i_1,i_2,\ldots,i_{2r}}$ we can find a string y_{i_1} from s_{i_1} such that for each $g \in \mathcal{G}$ we have

$$d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_{2r}}) - d(s_{i_1}, g|Q_{i_1,i_2,\ldots,i_{2r}}) \ge \frac{r-3}{r-1}y - 2\delta_0 y$$

in $O(d_b^2(n_1+n_2)^{\log((r-1)^2e)8/\delta_0^2})$ time.

For the positions in $P_{i_1,i_2,\ldots,i_{2r}}$, we can use the LP approach to solve the following inequalities:

$$\begin{cases} \sum_{a\in\Sigma} x_{k,a} = 1, \quad k = 1,2,\ldots,|P_{i_1,i_2,\ldots,i_{2r}}|, \\ \sum_{k\in P_{i_1,i_2,\ldots,i_{2r}}} \sum_{a\in\Sigma} \chi(s,i,k,a)\, x_{k,a} \le \frac{r+1}{r-1} d_b - d(s_{i_1}, s_i|Q_{i_1,i_2,\ldots,i_{2r}}) - y, \\ \qquad i = 1,2,\ldots,n_1, \\ \sum_{k\in P_{i_1,i_2,\ldots,i_{2r}}} \sum_{a\in\Sigma} \chi(g,j,k,a)\, x_{k,a} \ge d_g - d(s_{i_1}, g_j|Q_{i_1,i_2,\ldots,i_{2r}}) - y, \\ \qquad j = 1,2,\ldots,n_2, \\ x_{k,a} \in \{0,1\}, \quad a \in \Sigma, \ k \in P_{i_1,i_2,\ldots,i_{2r}}. \end{cases} \tag{19}$$

Since $|P_{i_1,i_2,\ldots,i_{2r}}| \le 2rd_b$, from Theorem 3, the run time required is $O(2rd_b^2 \times |\Sigma|^{4\log(n_1+n_2)\times/\delta^2})$ for $\delta = \epsilon/(2r)$ since we have to guess the value of y. Since we

have to try all $Q_{i_1,i_2,\ldots,i_{2r}}$'s and the values of d_b and d_g, the total time required is $O(L^4 n_1^{2r}[(n_1+n_2)^{\log((r-1)^2 e)8/\delta_0^2} + r \times |\Sigma|^{4\log(n_1+n_2)\times/\delta^2}])$. The total errors are $(1+\epsilon+\frac{2}{r-1})d_b$ and $(1-\epsilon-\frac{2}{r}-2\delta_0)d_g$.

Case 3. Lemmas 13 and 15 show that we can find a string y_{i_1} in $O(rLy^2(n_1+n_2)^{(4r^2+4+4\log e)/\delta_0^2+2r})$ time such that for any $g \in \mathcal{G}$ we have $d(y_{i_1}, g|Q_{i_1,i_2,\ldots,i_{2r}}) \geq d(s, g|Q_{i_1,i_2,\ldots,i_{2r}}) - \frac{2}{r}$.

For positions in $P_{i_1,i_2,\ldots,i_{2r}}$, we do the same thing as in Case 2. The time required is $O(rd_b^2 \times |\Sigma|^{4\log(n_1+n_2)\times/\delta^2})$ for $\delta = \epsilon/(2r)$. Thus, the total time required is $O(rd^4 n_1^{2r}[L(n_1+n_2)^{(4r^2+4+4\log e)/\delta_0^2+2r} + |\Sigma|^{4\log(n_1+n_2)\times/\delta^2}])$. The total errors are $(1+\epsilon+\frac{2}{r-1})d_b$ and $(1-\epsilon-\frac{2}{r})d_g$ in this case.

Case 1 dominates the time complexity when the errors are small. For example, if $e = \frac{1}{r-1}$, then the algorithm finds a distinguishing string s such that for every $s_i \in \mathcal{B}$, $d(s_i, s) \leq (1+\frac{1}{r-1})d_b$; for every $g_i \in \mathcal{G}$, $d(g_i, s) \geq (1-\frac{1}{r-1})d_g$; and the running time is $O(L \times |\Sigma|^{O(\log(n_1+n_2)(r-1)^6)})$. □

5. The DSSP. In this section, we present the algorithm for the DSSP. The idea is to combine the sampling technique in [9] with the algorithm for the DSP. The difficulty here is that for each $s_i \in \mathcal{B}$, we do not know the substring t_i of s_i. The sampling approach in [9] is as follows: For any fixed $r > 0$, by trying all the choices of r substrings of length L from $\mathcal{B}$, we can assume that $t_{i_1}, t_{i_2}, \ldots, t_{i_r}$ are the r substrings of length L that satisfy Lemma 13 by replacing s_l with t_l and s_{i_j} with t_{i_j}. Let Q be the set of positions at which $t_{i_1}, t_{i_2}, \ldots, t_{i_r}$ agree and $P = \{1, 2, \ldots, L\} - Q$. By Lemma 13, $t_{i_1}|Q$ is a good approximation to $s|Q$. However, we do not know the letters at positions in P. Thus, we randomly pick $O(\log(mn))$ positions from P, where m is the length of bad strings. Suppose that the multiset of these random positions is R. By trying all length-$|R|$ strings, we can assume that we know $s|R$. Then for each $1 \leq i \leq n_1$ we find the substring t_i' from s_i such that $f(t_i') = d(s, t_i'|R) \times \frac{|P|}{|R|} + d(t_{i_1}, t_i'|Q)$ is minimized. Let s be the optimal distinguishing string. Then t_i denotes the substring of s_i that is closest to s. Let s^* be a string such that $s^*|P = s|P$ and $s^*|Q = t_{i_1}|Q$. Then [9] shows the following.

Fact 1. With probability $1 - ((nm)^{-2} + (nm)^{-\frac{4}{3}})$, $d(s^*, t_i') \leq d(s^*, t_i) + 2\epsilon|P|$ for all $1 \leq i \leq n$.

After obtaining a t_i' of s_i for every $s_i \in \mathcal{B}$, we have the DSP, which can be solved by using the algorithms developed in section 3.

THEOREM 17. *There is a polynomial time approximation scheme that takes $\epsilon > 0$ as part of the input and computes a center string s of length L such that for every $s_i \in \mathcal{B}$ there is a length-L substring t_i of s_i with $d(s_i, s) \leq (1+\epsilon)d_b$, and for every $g_i \in \mathcal{G}$, $d(g_i, s) \geq (1-\epsilon)d_g$ if a solution exists.*

Proof. Let s' be the solution obtained by our algorithm. Lemma 9 and Fact 1 ensure that for each $s_i \in \mathcal{B}$, $s(t_i', s') \leq \frac{r}{r-1}d_b + \epsilon'|P|$, where $\frac{1}{r-1}d_b$ is the error at the positions in Q and $\epsilon'|P|$ ($\epsilon' = 2\epsilon + \epsilon''$) is the total error at the positions in P produced by both the random sampling method and the algorithm for the DSP. Lemma 9, together with sections 3 and 4, ensures that for each $g \in \mathcal{G}$, $d(g, s') \geq (1-\epsilon)d_g$, where ϵ is the error rate introduced by the algorithm for the DSP.

The randomized algorithm can be derandomized by the standard methods; [7] gives a short and clear explanation. □

6. Discussion and remarks. Our work concludes the search for provably good algorithms for the DSSP by presenting a PTAS. Some techniques have been developed

here dealing with values of parameters smaller than $\log(n)$ but larger than $O(1)$. They may have other applications to similar problems.

REFERENCES

[1] A. Ben-Dor, G. Lancia, J. Perone, and R. Ravi, *Banishing bias from consensus sequences*, in Proceedings of the 8th Annual Combinatorial Pattern Matching Conference, Aarhus, Denmark, 1997, pp. 247–261.

[2] J. Dopazo, A. Rodríguez, J. C. Sáiz, and F. Sobrino, *Design of primers for PCR amplification of highly variable genomes*, Comput. Appl. Biosci., 9 (1993), pp. 123–125.

[3] L. Gąsieniec, J. Jansson, and A. Lingas, *Efficient approximation algorithms for the Hamming center problem*, in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99), Baltimore, MD, 1999, SIAM, Philadelphia, 1999, pp. S905–S906.

[4] M. Ito, K. Shimizu, M. Nakanishi, and A. Hashimoto, *Polynomial-time algorithms for computing characteristic strings*, in Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching, Asilomar, CA, 1994, pp. 274–288.

[5] J. K. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang, *Distinguishing string selection problems*, in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99), Baltimore, MD, 1999, SIAM, Philadelphia, 1999, pp. 633–642.

[6] M. Li, B. Ma, and L. Wang, *Finding similar regions in many strings*, in Proceedings of the 31st ACM Symposium on Theory of Computing, Atlanta, GA, 1999, pp. 473–482.

[7] M. Li, B. Ma, and L. Wang, *On the closest string and substring problems*, J. Assoc. Comput. Mach., 49 (2002), pp. 157–171.

[8] 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*, Comput. Appl. Biosci., 7 (1991), pp. 525–529.

[9] B. Ma, *A polynomial time approximation scheme for the closest substring problem*, in Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, Montreal, Canada, 2000, pp. 99–107.

[10] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.

[11] C.H. Papadimitriou and M. Yannakakis, *On the approximability of trade-offs and optimal access of web sources*, in Proceedings of the 41st Annual Symposium on Foundations of Computer Science, Redondo Beach, CA, 2000, IEEE, pp. 86–92.

[12] V. Proutski and E. C. Holme, *Primer master: A new program for the design and analysis of PCR primers*, Comput. Appl. Biosci., 12 (1996), pp. 253–255.