



ACADEMIC
PRESS

Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Information and Computation 185 (2003) 41–55

Information
and
Computation

www.elsevier.com/locate/ic

Distinguishing string selection problems[☆]

J. Kevin Lanctot,^{a,*} Ming Li,^{a,1} Bin Ma,^b Shaojiu Wang,^c and Louxin Zhang^{d,2}

^a*Department of Computer Science, University of Waterloo, Waterloo, Ont., Canada N2L 3G1*

^b*Department of Computer Science, University of Western Ontario, London, Ont., Canada N6A 5B7*

^c*Pasteur Merieux Connaught Canada, 1755 Steeles Avenue West, Toronto, Ont., Canada M2R 3T4*

^d*Department of Mathematics, National University of Singapore, Singapore 117543, Singapore*

Received 19 August 1998; revised 14 June 2000

Abstract

This paper presents a collection of string algorithms that are at the core of several biological problems such as discovering potential drug targets, creating diagnostic probes, universal primers or unbiased consensus sequences. All these problems reduce to the task of finding a pattern that, with some error, occurs in one set of strings (Closest Substring Problem) and does not occur in another set (Farthest String Problem). In this paper, we break down the problem into several subproblems and prove the following results.

1. The following are all NP-Hard: the Farthest String Problem, the Closest Substring Problem, and the Closest String Problem of finding a string that is close to each string in a set.
2. There is a PTAS for the Farthest String Problem based on a linear programming relaxation technique.
3. There is a polynomial-time $\left(\frac{4}{3} + \epsilon\right)$ -approximation algorithm for the Closest String Problem for any small constant $\epsilon > 0$. Using this algorithm, we also provide an efficient heuristic algorithm for the Closest Substring Problem.
4. The problem of finding a string that is at least Hamming distance d from as many strings in a set as possible, cannot be approximated within n^ϵ in polynomial time for some fixed constant ϵ unless $NP = P$, where n is the number of strings in the set.

[☆] An extended abstract of this paper appeared in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms.

* Corresponding author.

E-mail addresses: jklanctot@wh.math.uwaterloo.ca (J.K. Lanctot), mli@math.uwaterloo.ca (M. Li), bma@csd.uwo.ca (B. Ma), jwang@ca.pmc-vacc.com (S. Wang), matzlx@nus.edu.sg (L. Zhang).

¹ Supported in part by the NSERC Research Grant OGP0046506, a CGAT grant, and the Steacie Fellowship.

² The work was done in Kent Ridge Digital Labs.

5. There is a polynomial-time 2-approximation for finding a string that is both the Closest Substring to one set, and the Farthest String from another set.

© 2003 Elsevier Science (USA). All rights reserved.

1. Introduction

With the wealth of genetic information being generated, the challenge of using this information beneficially presents a new series of problems to be solved. One set of problems is based on the idea of discovering and using genetic information that distinguishes one set of closely related species from another set of species. For example, one might want to create a drug that would kill several closely related pathogenic bacteria yet would be relatively harmless to humans. One approach is to look at genes that encode essential proteins to find a gene or part of a gene that is very similar among the set of related bacteria but different from that of humans. This distinguishing region could then become a potential target for drug design. It is the goal of this paper to formulate and systematically study the optimization problems that underpin this task.

1.1. DNA

To understand our approach, one must first understand the relationship between genetic material and protein. Deoxyribonucleic acid (DNA) is the genetic material for almost all organisms (except for some viruses). Its single-stranded form can be thought of as a string in the alphabet $\{A, C, G, T\}$ representing the four bases adenine, guanine, cytosine, and thymine respectively, and these bases encode the genetic information which is vital to the organization and functions within living organisms. To use this information, a cell *transcribes*, that is copies, a portion of the DNA into an intermediate compound called messenger ribonucleic acid (mRNA). Like DNA, mRNA can be thought of as a string but in a slightly different alphabet $\{A, C, G, U\}$, namely replacing T with U (for uracil). The cell then *translates* the information encoded in the mRNA into protein, which can be thought of as a string over the alphabet of 20 amino acids. Triplets of the mRNA sequence specify amino acids which are linked together to form a protein. Proteins are the basic functional units of the cell and are used, among other roles, to facilitate and control the complex chemical reactions that take place in the cell. So DNA encodes information by specifying the primary structure of proteins, the functional units of a cell.

1.2. Hamming distance

It is not just the role of DNA that motivates this paper, but also its structure. DNA often occurs in a double-stranded form, where the end of one strand lines up with the beginning of the other, each A on one strand pairs with a T on the other, and likewise each G on one strand pairs with a C on the other (Watson–Crick base pairing). When two strands bind together in this manner, they are said to hybridize, and if this matching is exact, one strand is called the reverse complement of the other.

Under certain conditions, exact Watson–Crick pairing is not required for this double-stranded form and these differences have been classified into various categories whose properties have been investigated. One category is substitutions [4,15] (also called mismatches), where a base in one strand is not

the Watson–Crick pair of the base on the other strand; another category is gaps [16] (also called bulges), where there is at least one extra base in one strand that is not paired with any bases in the other strand. The destabilizing effect of gaps and substitutions have been tabulated (at body temperature and in a 1 M NaCl solution [22]) and are quantified in terms of Gibbs free energy. For a substitution, the free energy, G_s^0 , is about 0.8 kcal/mol, and for a bulge G_b^0 , it is about 3.3 kcal/mol. Given two short sequences in equal concentrations that could either hybridize with one bulge or with one substitution (assuming that the rest of the base matches would be the same), then the ratio of substitutions to bulges is $e^{\Delta G^0/RT}$ [18] where T is the temperature and R is a constant, which means hybridizing with a substitution would be about 58 times more likely to occur than hybridizing with a gap. Because gaps are more destabilizing, if one is designing an oligomer (short strand of DNA) to bind tightly to another strand, one might use Hamming distance, which considers only substitutions, rather than edit distance, which considers both gaps and substitutions.

2. Applications

The Hamming distance metric appears in several contexts. In particular, it is used in coding theory [7,9], and in several biological applications. The biological applications occur in two varieties: some require that a region of similarity be discovered, for example consensus sequences, and other applications use the reverse complement of that region, such as designing probes or primers [12,17,23]. Our algorithms report the region directly with the understanding that the reverse complement of the region can be easily calculated if required. With this in mind, several biological application will be discussed.

Distinguishing String Selection Problems have the potential to help out in drug target selection. Given a dataset of sequences of orthologous genes (the same gene from different species) from a group of closely related pathogens, and a host (such as humans or livestock), the goal would be to find an essential sequence that is more conserved in all or most of the pathogens but not as conserved in the hosts. The protein encoded by this fragment could become a target for novel antibiotic development. Jiang et al. [13] have looked at this problem, however, they use Gibbs sampling to identify the drug targets.

Another application of Distinguishing String Selection Problems is with consensus sequences. Given a collection of related sequences, a consensus sequence is a single sequence that best represents the collection. A challenge associated with creating consensus sequences is sample bias. For example, given a dataset of sequences of orthologous genes from many closely related species and a few more distantly related ones, the resulting consensus sequence could be biased towards sequences from the over-represented species group. One proposed approach to deal with the bias is to create a consensus sequence by minimizing the maximum distance from any sequence rather than minimizing the total distance [5] and this task is exactly our Closest String Problem.

Finally, Distinguishing String Selection Problems may also find applications in creating diagnostic probes for bacterial infection and creating universal PCR primers (see [14] for details).

3. Problem formulation and results

For any two strings x and y of same length, we use $d(x, y)$ to denote the Hamming distance between them, which is defined as the number of mismatched positions. The previous applications all require finding a string subject to the following two constraints: (1) it must be close or similar to one set of

strings, \mathcal{S}_c and (2) it must be far or different from another set of strings \mathcal{S}_f . More formally these two problems are as follows:

Closest String Problem.

Instance: Given a set \mathcal{S}_c of strings of length n over an alphabet A .

Objective: Find a string x of length n over A minimizing d_c such that for every string s in \mathcal{S}_c , $d(x, s) \leq d_c$.

Farthest String Problem.

Instance: Given a set \mathcal{S}_f of strings of length n over an alphabet A .

Objective: Find a string x of length n over A maximizing d_f such that for any s in \mathcal{S}_f , $d(x, s) \geq d_f$.

Informally we will say that a string x binds to another string s if the Hamming distance $d(x, s) \leq d_c$. Another possibility that occurs in practice is that x must be close to or far from a *substring* of s . This observation leads to two more problems:

Closest Substring Problem.

Instance: Given a set \mathcal{S}_c of strings of length at least n over an alphabet A .

Objective: Find a string x of length n minimizing d_c such that for every string s in \mathcal{S}_c , $d(x, y) \leq d_c$ holds for some length- n substring y of s .

Farthest Substring Problem.

Instance: Given a set \mathcal{S}_f of strings of length at least n over an alphabet A .

Objective: Find a string x of length n maximizing d_f such that for every string s in \mathcal{S}_f , and every length- n substring y of s , $d(x, y) \geq d_f$.

An additional consideration is that sometimes it is impossible to find a string that is far or close to every member in a set, so the next best constraint is to be as far or close to as many members of that set as possible. These leads to two more problems:

Close to Most String Problem.

Instance: Given a set \mathcal{S}_c of strings of length n over an alphabet A and a threshold $k_c > 0$.

Objective: Find a string x of length n maximizing the number of strings s in \mathcal{S}_c satisfying the constraint that the Hamming distance $d(x, s) \leq k_c$.

Far From Most String Problem.

Instance: Given a set \mathcal{S}_f of strings of length n over an alphabet A and a threshold $k_f \geq 0$.

Objective: Find a string x of length n maximizing the number of strings s in \mathcal{S}_f satisfying the constraint that the Hamming distance $d(x, s) \geq k_f$.

We will say x cannot bind to s if $d(x, s) \geq k_f$. But the condition that a string x cannot hybridize to any part of a string y is equivalent to the property that it is far from any substring of y that has the same length as x . Hence the Farthest String Problem and the Farthest Substring Problem are identical from the point of view of complexity.

Finally, the Distinguishing String Selection Problem (DSSP) can be formalized as the following.

Distinguishing String Selection Problem.

Instance: Given two sets of strings \mathcal{S}_c and \mathcal{S}_f , all of length at least n , and two positive integers k_c and k_f .

Objective: Find a string x satisfying that for each string s_c in \mathcal{S}_c , there exists a length n substring y_c of s_c such that $d(x, y_c) \leq k_c$, and for any substring y_f of string s_f in \mathcal{S}_f , $d(x, y_f) \geq k_f$.

For simplicity in this paper, we state all the results using Hamming distance where weights in the mismatched positions are just one. All results still hold when the weights are in a positive interval. These results are summarized as follows. The Farthest String Problem, the Farthest Substring Problem and Closest Substring Problem are NP-hard. There is a simple polynomial-time 2-approximation algorithm for the Closest Substring Problem and there is a polynomial-time $(\frac{4}{3} + \epsilon)$ -approximation algorithm for the Closest String Problem for any small constant $\epsilon > 0$. As well, there is a polynomial time approximation scheme (PTAS) for the Farthest String Problem. Finally the Far from Most String Problem cannot be approximated with n^ϵ in polynomial time for some fixed constant ϵ unless NP=P and there is a polynomial-time 2-approximation for the Distinguishing String Selection Problem.

4. Related work

In addition to the work mentioned in Section 2, there are several problems related those studied in this paper. First, the hitting string problem is similar to our Closest String Problem. Given a set S of strings of length n over $\{0, 1, \star\}$, the hitting string problem is to find a string over $\{0, 1\}$ that has at least one match with each string in S [8]. Such a problem was proved to be NP-complete by Fagin [6]. This problem is a special case of the Closest String Problem in which the Hamming distance bound is $n - 1$ and the sought string lies over $\{0, 1\}$ rather than over the original alphabet $\{0, 1, \star\}$.

The complexity and approximability of finding maximum feasible subsystems of linear relations was studied by Amaldi and Kann [2]. Although Hamming distance conditions can be transformed into linear relations, their results do not imply our results. Actually, our NP-hardness results are stronger than some of theirs.

Designing DNA probes was studied by Ito et al. [11]. They formalized the problem as follows: given a set of strings S and a subset $T \subseteq S$, find a d -characteristic string of T under S , which is defined to be a substring occurring in all the strings in T and at least edit distance d away from any substrings of strings in $S - T$. Since a characteristic string is a common substring of strings in T , such a problem can be solved in polynomial time [11]. The Distinguishing String Selection Problem studied here is more difficult because whereas they require that all the strings in T contain a common substring, we only require that all strings in T contain a substring that is within a constant Hamming distance from our characteristic string.

Ben-Dor et al. [5] gave a different formulation: given a set of strings S and a subset $T \subseteq S$, using Hamming distance, d_H , find a t that maximizes k such that

$$\min_{v \in S-T} d_H(t, v) - \max_{v \in T} d_H(t, v) = k.$$

As well, this paper provides an approximation to the Closest String Problem which with probability less than ϵ they can get

$$d_o + \sqrt{3d_o \log \frac{|m|}{\epsilon}},$$

where d_o is the optimal distance and m is the number of strings. However, a small d is critical for our applications and the straightforward LP relaxation method as used in [5] does not work well for small d .

The Closest String Problem also occurs in coding theory, and has been proven NP-Complete for binary codes [7]. Again in the context of coding theory, Gąsieniec [9] independently claim a $\left(\frac{4}{3} + \epsilon\right)$ -approximation.

5. The complexity of Farthest String Problem

In this section, we prove the Farthest String Problem is NP-hard. This will be broken up into alphabet sizes greater than two, and alphabet sizes equal to two. We are especially interested in the cases when the alphabet is four (for DNA and RNA) and twenty (for protein).

Theorem 1. *The Farthest String Problem is NP-hard for strings over any alphabet A with $|A| > 2$.*

Proof. The proof of NP-hardness is approached by reducing the strong 3-SAT problem [8] (that is, it cannot consist of trivial clauses like $(\bar{x} \vee z \vee z)$) to the Farthest String Problem. First consider the case where the alphabet size is three, that is $|A| = 3$.

Let $I_{m,n}$ be an arbitrary instance of the strong 3-SAT problem with m clauses $\{C_1, C_2, \dots, C_m\}$ and n variables $\{v_1, v_2, \dots, v_n\}$. Construct $m + 9$ strings each of length $n + 2$. The first n characters of the first m strings will encode the clauses of $I_{m,n}$ and the rest will encode constraints to ensure that the answer is a valid solution to the 3-SAT problem. As well, the i th string will be referred to as s_i and the j th character of the i th string will be denoted s_{ij} . Formally, the first m strings be formulated as follows:

$$s_{ij} = \begin{cases} 0 & v_j \in C_i, \\ 1 & \bar{v}_j \in C_i, \\ \star & v_j \text{ does not appear in } C_i, \\ \star & j = n + 1, n + 2. \end{cases}$$

For the last nine strings, the first n characters will always be \star and the last two characters will be a different one of the nine possible strings of length two on three characters, namely $\{0, 1, \star\}$. In other words, they are $\star^9\{0, 1, \star\}^2$.

Now we need to shown that the instance $I_{m,n}$ is satisfiable if and only if there is a string x of length $n + 2$ on $\{0, 1, \star\}$ such that $d(x, s_i) \geq n$ for every $1 \leq i \leq m + 9$.

First, suppose $I_{m,n}$ is satisfied by an assignment $x \in \{0, 1\}^n$. Define $x00 = x_1x_2 \cdots x_n00$. For the first m strings, if x makes C_i true, then the following three points hold. There is at least one mismatch between $x00$ and s_i where a 1 mismatches a 0 in the first n characters. There are $n - 3$ mismatches in the first n symbols where either a zero or a one in $x00$ mismatches a \star in s_i . There are two more mismatches because the last two characters of $x00$ and s_i are different.

For the last nine strings, since x does not contain a \star anywhere, $d(x00, s_{m+j}) \geq n$ for every $1 \leq j \leq 9$. Hence, $d(x00, s_i) \geq n$ for all i such that $1 \leq i \leq n$.

Conversely, suppose there is a string $x = x_1x_2 \cdots x_{n+1}x_{n+2}$ such that $d(x, s_i) \geq n$ for every $1 \leq i \leq m + 9$. Then x does not contain any \star 's in the first n positions, otherwise it would match at least three characters in at least one of the last nine strings, making the distance between that string and x less than or equal to $n - 1$. Since x does not contain any \star 's in the first n positions, it induces an assignment to the variables v_i . Such an assignment is a satisfying assignment for $I_{m,n}$ since at least one variable in each clause is true because it mismatches.

This is the case for an alphabet of size three. For alphabet sizes p , where $p > 2$, $(p - 2)p^2$ extra strings must be added to the m strings, which can be grouped into $p - 2$ groups of p^2 characters, where each group consists of the first n characters being one of the characters in the alphabet other than $\{0, 1\}$, and the last two characters being every possible combination of two characters in the alphabet. Hence the theorem is proved. \square

The binary case of the Farthest String Problem requires careful encoding.

Theorem 2. *The Farthest String Problem is NP-hard even for binary strings.*

Proof. Again, the 3-SAT problem will be reduced to the Farthest String Problem. Let $I_{m,n}$ be an arbitrary instance of the 3-SAT problem with m clauses $\{C_1, C_2, \dots, C_m\}$ and n variables $\{v_1, v_2, \dots, v_n\}$. For each clause C_i , construct a string $s_i = s_{i1}s_{i2} \dots s_{in}$ of length $2n$ from the set $\{00, 01, 11\}^n$, where s_{ij} is in the following format:

$$s_{ij} = \begin{cases} 00 & v_j \in C_i, \\ 11 & \bar{v}_j \in C_i, \\ 01 & v_j \text{ does not appear in } C_i. \end{cases}$$

Let $p(x, i)$ represent the string $(10)^{i-1}x(10)^{n-i}$, and let $P_n(x)$ represent the set of n strings $\{p(x, i) \mid 1 \leq i \leq n\}$. As well, let $q(x, i)$ represent the string $(01)^{i-1}x(01)^{n-i}$ and let $Q_n(x)$ represent the set of strings $\{q(x, i) \mid 1 \leq i \leq n\}$. Then the corresponding instance of the Farthest String Problem is

$$S_{I_{m,n}} = P_n(00) \cup P_n(11) \cup P_n(01) \cup \{(10)^n\} \cup Q_n(00) \cup Q_n(11) \\ \cup Q_n(10) \cup \{(01)^n\} \cup \{s_i \mid 1 \leq i \leq m\}.$$

Clearly the instance $S_{I_{m,n}}$ is computable in polynomial time, and like the non-binary instance, the purpose of including the various $P_n()$, $Q_n()$, $(10)^n$, $(01)^n$ sets is to force the solution of $S_{I_{m,n}}$ to be a string in $\{00, 11\}^n$.

Now we show that the instance $I_{m,n}$ is satisfiable if and only if there exists a string x that is at least $n - 1$ away from every string in the set $S_{I_{m,n}}$.

First assume that there is a solution to the strong 3-SAT problem. Then a string $x = x_1x_2 \dots x_n$ over the alphabet $\{00, 11\}^n$ can be constructed so that there is at least one mismatch between x and each string s_i causing the Hamming distance to be at least $n - 1$, in the following manner:

$$x_i = \begin{cases} 11 & \text{if } v_j \text{ is true,} \\ 00 & \text{if } v_j \text{ is false.} \end{cases}$$

Since x is an element of $\{00, 11\}^n$, then by construction it is at least $n - 1$ away from any other string in $S_{I_{m,n}}$.

Next let x be a solution to the Farthest String Problem. In order to prove that x is in $\{00, 11\}^n$, assume the contrary. Factor x into the concatenation of pairs of binary symbols, that is, $x = x_1x_2 \dots x_n$ where $x_i \in \{00, 11, 10, 01\}$. Since x is not in $\{00, 11\}^n$, x is in the form $\{00, 11\}^* (10|01)\{01, 11, 10, 01\}^*$.

Let n_{10} and n_{01} denote the numbers of 10s and 01s in the factorization. Without loss of generality, assume that $n_{10} \leq n_{01}$. There are two cases to consider.

Case 1: If $0 < n_{10} \leq n_{01}$, then let the first 10 occur in the i th position in the factorization, that is, $x_i = 10$. Then $d(x, q(10, i)) = (n - n_{10} - n_{01}) + 2(n_{10} - 1) \leq n - 2$, which contradicts the requirement that x is at least $n - 1$ units away from all strings.

Case 2: If $n_{10} = 0$ and $n_{01} > 0$, then, $d(x, (01)^n) = n - n_{01} \geq n - 1$. Thus, $n_{01} = 1$. Let the i th position be a location where 01 does not occur. If $x_i = 11$ then $d(x, q(11, i)) = n - 2$, and if $x_i = 00$ then $d(x, q(00, i)) = n - 2$.

Hence x is a string in the alphabet $\{11, 00\}^n$ that is at least $n - 1$ away from any string in $\{s_i | 1 \leq i \leq n\}$, and this string can be used to find a satisfying truth assignment for the 3-SAT problem because a mismatch of 00 to 11 in each string will correspond to a satisfying assignment for at least one term in each clause. \square

6. PTAS for the Farthest String Problem

Although the Farthest String Problem is NP-hard, through the use of a linear programming relaxation technique, a PTAS for the Farthest String Problem can be presented. First a lower bound is needed.

Lemma 1. *Let \mathcal{S} be a set of m strings, $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$, each of length n over an alphabet A of size t . If $n > 6 \ln(2m/\beta)/\beta^2$ where β is a constant such that $0 < \beta \leq 1$, then there exists a string x such that the Hamming distance between s_i and x is at least $(1 - \beta) \frac{n(t-1)}{t}$ for every $s_i \in \mathcal{S}$.*

Proof. The theorem can be proved by a probabilistic argument. Let s_i be a string in \mathcal{S} . Given a random string $x \in A^n$, the expected value of the Hamming distance between s_i and x is $\frac{n(t-1)}{t}$. Thus, by a Chernoff bound [20],

$$\Pr \left[\left| d(x, s_i) - \frac{n(t-1)}{t} \right| > \beta \frac{n(t-1)}{t} \right] \leq \frac{2}{e^{0.38\beta^2 \frac{n(t-1)}{t}}}.$$

By requiring $n > 6 \ln(2m/\beta)/\beta^2$, and substituting this value for n in the above inequality, the results yield that the right hand side is bounded from above by β/m . Hence,

$$\Pr \left[\bigcup_{1 \leq i \leq m} \left(\left| d(x, s_i) - \frac{n(t-1)}{t} \right| > \beta \frac{n(t-1)}{t} \right) \right] < \beta.$$

Thus, there exists a string x such that $d(x, s_i) \geq (1 - \beta) \frac{n(t-1)}{t}$ for all strings s_i . \square

The next step is to set-up a formulation of the Farthest String Problem by integer programming. Let $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ be a set of m strings, each of length n over an alphabet $A = \{a_1, a_2, \dots, a_t\}$ of t symbols.

Let $\beta > 0$ be any small number. If $n < 6 \ln(2m/\beta)/\beta^2$, perform an exhaustive search to find an optimal solution. Otherwise, set-up the following zero–one integer programming formulation of the problem. Let $s = s_1 s_2 \dots s_n \in \mathcal{S}$ and let $x = x_1 x_2 \dots x_n$ be an arbitrary string over the alphabet A . The Hamming distance between s and x can be calculated as follows:

$$d(s, x) = \sum_{i=1}^n \left(1 + \sum_{j=1}^t c_{ij} x_{ij} \right),$$

where $c_{ij} = 0$ if $s_i \neq t_j$ and -1 otherwise, $x_{ij} = 0$ if $x_i \neq t_j$ and 1 otherwise.

With this formulation of the Hamming distance, the Farthest String Problem is stated as the following zero–one integer program:

$$\begin{aligned} & \max d_f, \\ & \sum_{i=1}^n \sum_{j=1}^t c_{hij} x_{ij} + n \geq d_f \quad \forall 1 \leq h \leq m; \\ & \sum_{j=1}^t x_{ij} = 1 \quad \forall 1 \leq i \leq n; \\ & x_{ij} \in \{0, 1\} \quad \forall 1 \leq i \leq n, 1 \leq j \leq t, \end{aligned} \tag{1}$$

where $c_{hij} = 0$ if the i th character in the h th string is not t_j and -1 otherwise.

It is NP-hard to solve integer programs. Therefore, to solve IP (1) approximately, we first relax the integrality constraints on each x_{ij} by replacing them with the constraints $0 \leq \bar{x}_{ij} \leq 1$ and solve the resulting linear program. Let $\bar{x} = (\bar{x}_{ij})$ be the solution vector of this linear program and let the optimal objective value be \bar{d}_f .

Now apply randomized rounding [19] to restore integrality: for each i , independently set x_{ij} to be 1 with probability \bar{x}_{ij} . The random process produces a 0–1 solution with the objective value d_f satisfying

$$d_f > \bar{d}_f - \epsilon O\left(\sqrt{n \log m}\right),$$

where $\epsilon > 0$ is an arbitrary constant with high probability (related to ϵ).

Letting $d_{f,opt}$ be the unknown optimal solution for the Farthest String Problem, by Lemma 1,

$$d_{f,opt} > (1 - \beta) \frac{n(t - 1)}{t},$$

with high probability (related to β).

Consider the ratio between d_f and $d_{f,opt}$, for large n .

$$\frac{d_f}{d_{f,opt}} \geq \frac{\bar{d}_f}{d_{f,opt}} - \frac{\epsilon O\left(\sqrt{n \log m}\right)}{d_{f,opt}} \geq 1 - \frac{\epsilon t}{(1 - \beta)(t - 1)} O\left(\sqrt{(\log m)/n}\right) \geq 1 - \epsilon,$$

with high probability (related to both ϵ and β).

With this bound achieved, the above algorithm can result in a PTAS after de-randomization. One approach to de-randomizing would be with conditional probabilities (see [1] or [20]). Hence, we have proved the following result.

Theorem 3. *There is a PTAS for the Farthest String Problem.*

7. Hardness of approximating Far From Most String Problem

Given two optimization problems P and Q , a polynomial-time transformation f from P to Q is an L -reduction if there are constants α and β such that for every instance x of P (see [21]),

1. $opt_Q(f(x)) \leq \alpha \cdot opt_P(x)$,
2. for every solution y of $f(x)$ with objective value c_2 , a polynomial time solution y' of x with objective value c_1 can be found such that $|opt_P(x) - c_1| \leq \beta \cdot |opt_Q(f(x)) - c_2|$.

Theorem 4. *For strings over an alphabet A with $|A| = 3$, approximating the Far From Most String Problem within a factor n^ϵ is NP-hard for some fixed constant ϵ , where n is the number of strings.*

Proof. There is an identical L-reduction from the Independent Set problem to the Far From Most String (FFMS) problem. To see this, we will show that for a non-complete graph G of m edges, there is a corresponding FFMS instance (S_G, m) with threshold m such that G has an independent set of size k if and only if (S_G, m) has a solution of size k (i.e., there is a string which is distance m away from at least k strings).

Give a non-complete graph $G = (V, E)$, we construct an instance (S_G, d) of the Far From Most String Problem as follows. Let $V = \{v_1, v_2, \dots, v_n\}$ and m edges $E = \{e_1, e_2, \dots, e_m\}$. For each vertex v_i , construct a string $s_i = s_{i1}s_{i2} \cdots s_{im} \in \{0, 1, \star\}$ of length m . For each position $j \leq m$, define s_{ij} as

$$s_{ij} = \begin{cases} 0 & \text{if } e_j = (v_i, v_{i'}) \text{ for some } i' > i, \\ 1 & \text{if } e_j = (v_{i'}, v_i) \text{ for some } i' < i, \\ \star & \text{if } v_i \text{ is not an endpoint of } e_j. \end{cases}$$

Then, $S_G = \{s_1, s_2, \dots, s_n\}$ and the Hamming distance threshold k_c is m . The transformation from G to (S_G, m) is obviously computable in polynomial time.

Let s' be an optimal solution for the instance (S_G, m) and let $S' = \{s \in S_G \mid d(s', s) \geq m\}$, where ‘ \geq ’ is actually ‘=’ since each string has length m . If s' contains a \star in some position j , then, s' can mismatch at most two strings at position j , which corresponds to the endpoints of the j th edge in G . Thus, $|S'| \leq 2$. Since G is not complete, G has an independent set of size 2. If s' does not contain any \star 's, then the vertices corresponding to strings in S' form an independent set for G . Hence, G has an independent set of size $|S'|$.

Conversely, given an independent set V' of G , we consider the strings that correspond to vertices in V' . Let S' denote the set of such strings. Since V' is independent, on each position, S' contains at most one string that has non-star symbol, that is either 0 or 1. Thus, we construct a string $s = s_1s_2 \cdots s_m$ by assigning 0 to s_i if 0 does not appear in the i th position of every string in S' , and 1 otherwise. It is not difficult to see that s is Hamming distance m far from each string in S' .

We have proved that there is an identical reduction from the Independent Set problem to the FFMS problem. From the hardness result on the Independent Set problem [3], our theorem follows in the case of $|A| = 3$. \square

Remark. The above construction has to be modified for $|A| \geq 4$. In particular, when $|A| = 4$, the construction is as follows. Consider only the graphs G that have an independent set containing at least $|A| = 4$ vertices. For such a graph $G = (V, E)$ with n vertices and m edges, construct a set S_4 of n strings from $S = \{s_1, s_2, \dots, s_n\}$ constructed in the above proof by substituting 0 with 0^n , 1 with 1^n , and the i th \star in each position (counting down from s_1 to s_n) with $\star^{i-1}\# \star^{n-i}$. Thus, each resulting string is of length nm . The relationship between G and (S_4, nm) is the same.

8. Approximation of the Closest String and Substring Problems

First, we have that the Closest String Problem is NP-hard.

Theorem 5. *The Closest String Problem is NP-hard.*

Proof. Reduce the Farthest String Problem to this problem. First consider the case of $|A| = 2$. Given an integer k_c and a set of m strings $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ each of length n , the following statements are all equivalent for $1 \leq i \leq m$. There is a string x such that $d(x, s_i) \geq k_c$ for every s_i . There are at least k_c mismatches between x and s_i . There are strictly less than $k_f = n - k_c$ mismatches between s_i and \bar{x} , the complement of x . For every s_i , $d(\bar{x}, s_i) \leq k_f$.

The binary case can be generalized to the case $|A| > 2$ in the following manner. Given an instance of the binary Closest Substring Problem, $I_{m,n}$ over the alphabet A_2 , use the non-binary algorithm to solve $I_{m,n}$ to the distance bound k_c . Then take that non-binary solution, convert each letter that is not in A_2 to one of the symbols in A_2 arbitrarily. This step will only improve the quality of the solution, yielding a binary solution within distance k_c . By this reduction, the case $|A| > 2$ is also NP-hard. \square

Since the Closest String Problem is a special case of the Closest Substring Problem, we also have the following.

Theorem 6. *The Closest Substring Problem is NP-hard.*

With the NP-hardness of these problems determined, the next step is to establish the hardness of approximation. First, we have the following.

Lemma 2. *Both the Closest String Problem and the Closest Substring Problem can be approximated within a ratio of two in polynomial time.*

Proof. Consider the following special case of the star alignment approximation algorithm [10]. Let $I_{n,m}$ be an instance of the Closest Substring Problem, consisting of a set $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ each of length at least n over an alphabet A . Let x be an optimal solution such that for each string $s_i \in \mathcal{S}$, including s_1 , there exists a substring p_i of s_i such that $d(x, p_i) \leq k_c$. For every other string in \mathcal{S} , say s_j , by the triangle inequality, there exists a substring p_j of s_j such that $d(p_1, p_j) \leq 2k_c$. To find this string p_1 , just try each substring of length n in s_1 .

Since the Closest String Problem is just a special case of the Substring Problem, the results apply to both. \square

To improve the above approximation, we first study the following special case:

Theorem 7. *The Closest String Problem can be approximated with $\frac{4}{3}(1 + \epsilon)$ in polynomial time for any small constant ϵ .*

Proof. Let $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ be a set of strings each of length n over an alphabet A . Without loss of generality, assume the following: $d(s_1, s_2) = k \geq d(s_i, s_j)$ for any $s_i, s_j \in \mathcal{S}$ and that the k mismatches between s_1 and s_2 occur at the first k positions. These two assumptions are valid because the characters and strings of \mathcal{S} could be permuted to this format, the problem solved, and then the characters permuted back to their original positions without affecting the validity of the solution.

To simplify things, consider the following notation. Given a string x of length n , let x' represent the first k characters and let x'' represent the rest of the string. For example, $s_i = s'_i s''_i$, where $s'_i =$

$s_1 s_2 \cdots s_k$ and $s_i'' = s_{i(k+1)} s_{i(k+2)} \cdots s_{in}$. To derive an approximation solution of the form $s s_1''$ for the problem, where s is an unknown string of length k , we derive an integer program as

$$\begin{aligned} \min d; \\ d(s s_1'', s_i) = d(s, s_i') + d(s_1'', s_i'') \leq d, \quad i = 1, \dots, m, \end{aligned} \quad (2)$$

where we have not written out the precise form of the integer program and also omitted the zero–one constraints.

Let the optimal objective value for IP (2) be $d_{1,opt}$. It is impossible to compute $d_{1,opt}$ efficiently. Therefore, approximate it by considering two cases. Let $\delta > 1$ be some fixed constant.

Case 1. $k = \max_{i \neq j} d(s_i, s_j) \leq \frac{6 \ln(\delta m)}{\epsilon^2}$.

We do a simple exhaustive search for finding an optimal solution s_x for IP (2). We could try all the $|A|^k$ strings of length k over A . Then each chosen string can be checked in $O(mk)$ time. The total time is $O(|A|^k mk)$, a polynomial in terms of m .

Case 2. $k > \frac{6 \ln(\delta m)}{\epsilon^2}$.

We first solve the linear relaxation of IP (2). Let \bar{d} be the optimal objective value of the relaxation and let \bar{x}_i be its solution. Clearly, $d_{1,opt} \geq \bar{d}$. All \bar{x}_i 's are fractional values and therefore may not constitute a feasible solution to IP (2). We therefore round these fractional values to 0's and 1's to obtain a feasible solution for (2). The rounding process is similar to that found in Lemma 1 and Theorem 3. Such a rounding process results in a random string s_x . Consider a fixed string $s_i \in \mathcal{S}$. The expected value of the distance between $s_x s_1''$ and s_i satisfies the following:

$$E(d(s_x s_1'', s_i)) \leq \bar{d} \leq d_{1,opt}.$$

Let d_{opt} denote the optimal solution of the Closest String problem for \mathcal{S} . Then, $d_{opt} \geq \frac{k}{2} > \frac{3 \ln(\delta m)}{\epsilon^2}$,

$$\epsilon > \sqrt{\frac{3 \ln(\delta m)}{d_{opt}}}.$$

Using Hoeffding's bound (see, e.g. [20]), we get

$$\Pr[d(s_x s_1'', s_i) < d_{1,opt} + \epsilon d_{opt}] \leq \frac{1}{\delta m}.$$

Summing over every string S_i , we have

$$\Pr[d(s_x s_1'', s_i) < d_{1,opt} + \epsilon d_{opt} \text{ for all } i] \leq \frac{1}{\delta}.$$

Therefore, with high probability,

$$d(s_x s_1'', s_i) \leq d_{1,opt} + \epsilon d_{opt}$$

for every string s_i .

To derived a polynomial-time approximation for (2), we derandomized this randomized algorithm using techniques of conditional probabilities (for example, see [1] and [20]). Here we omit the details.

In summary, our approximation algorithm consists of two steps: (1) Solve IP (2) approximately or do a exhaustive search as described in Case 1 in the proof. Let s_x be a solution with distance bound $d_1 \leq d_{1,opt} + \epsilon d_{opt}$. (2) Output one of $s_1, s_x s_1''$ which reaches the bound $\min(d(s_1, s_2), d_1)$.

Now we start to analyze the algorithm. Let s_{opt} be a solution string with the optimal distance bound d_{opt} of the instance S . If $d(s_1, s_2) \leq \frac{4}{3}d_{opt}$, then, obviously, the solution given by our algorithm is a string with distance bound at most $\frac{4}{3}d_{opt}$.

Otherwise, $d(s'_1, s'_2) = d(s_1, s_2) > \frac{4}{3}d_{opt}$. By the triangle inequality,

$$d(s'_1, s'_{opt}) + (s'_2, s'_{opt}) \geq d(s'_1, s'_2) > \frac{4}{3}d_{opt}.$$

Thus, $d(s'_1, s'_{opt}) > \frac{2}{3}d_{opt}$ or $(s'_2, s'_{opt}) > \frac{2}{3}d_{opt}$. If the first holds, then $d(s''_1, s''_{opt}) \leq \frac{1}{3}d_{opt}$ and so for any $i \leq n$, by triangle inequality,

$$\begin{aligned} d(s'_{opt}s''_1, s_i) &= d(s_{opt}, s_i) - d(s''_{opt}, s''_i) + d(s''_1, s''_i) \\ &\leq d_{opt} + d(s''_1, s''_i) - d(s''_{opt}, s''_i) \\ &\leq d_{opt} + d(s''_1, s''_{opt}) \\ &\leq \frac{4}{3}d_{opt}. \end{aligned}$$

Similarly, if $d(s'_2, s'_{opt}) > \frac{2}{3}d_{opt}$, the above inequality still holds because $s''_1 = s''_2$. Thus, we have that the optimal objectal value of IP (2) is less than or equal to $\frac{4}{3}d_{opt}$, i.e., $d_{1,opt} \leq \frac{4}{3}d_{opt}$. This implies that

$$d(s_x s''_1, s_i) \leq d_{1,opt} + \epsilon d_{opt} \leq \frac{4}{3}(1 + \epsilon)d_{opt}.$$

This implies that in any case, our algorithm outputs a solution with distance bound at most $(1 + \epsilon)\frac{4}{3}d_{opt}$. □

Theorem 7 can be used to design an efficient heuristic for the Closest Substring Problem. Let $\mathcal{S} = \{s_1, s_2, \dots, s_m\}$ be a set of strings where the length of each string is at least n . Without loss of generality, assume s_1 is either shorter than, or the same length as any other string in \mathcal{S} . The heuristic takes two steps.

1. For each p_i , a substring of s_1 , and for each s_j , a string in \mathcal{S} such that $j \neq 1$, find a substring of s_j with length n that is closest to p_i , call it q_j , such that

$$\max_{p_i, s_j} d(p_i, q_j)$$

is minimized.

2. Apply the algorithm in Theorem 7 to improve the solution.

9. Distinguishing String Selection Problem

Let $(A, n, \mathcal{S}_c, k_c, \mathcal{S}_f, k_f)$ be an instance of the Distinguishing String Selection Problem (DSSP). The DSSP requires a string x of length n such that for each $s_c \in \mathcal{S}_c$, $d(x, p) \leq k_c$ for some substring p of s_c , and for each $q \in \mathcal{S}_f$, $d(x, q) \geq k_f$. By setting either $k_c = n$ or $k_f = 0$, respectively, both the Farthest String and Closest Substring Problems are special cases of the DSSP. Hence the DSSP is NP-hard.

When k_c is a small constant, a solution can be found by exhaustive search. This is because, if s_c is the shortest string in \mathcal{S}_c , there are at most

$$O\left((|s_c| - n + 1)n^{k_c}(|A| - 1)^{k_b}\right)$$

candidates and each candidate can be tested to determine if it is a solution in time that is polynomial in terms of the size of the instance. When k_c is relatively large, the following approximation result is obtained.

Theorem 8. *Let $(A, n, \mathcal{S}_c, k_c, \mathcal{S}_f, k_f)$ be an instance of the Distinguishing String Selection Problem. If there is a solution for the instance, an approximate solution x can be obtained in polynomial time such that for each $s_c \in \mathcal{S}_c$, $d(x, p) \leq 2k_c$ for some substring p of s_c , and for each $s_f \in \mathcal{S}_f$, $d(x, s_f) \geq k_f - k_c$.*

Proof. Suppose x_o is a solution, and x is a substring of a sequence in \mathcal{S}_c such that $d(x_o, x) \leq k_c$. For any $s_c \in \mathcal{S}_c$, there is a substring p of s_c such that by the triangle inequality $d(x, p) \leq 2k_c$. For any $s_f \in \mathcal{S}_f$, since $d(x_o, s_f) \geq k_f$, then

$$d(x, s_f) \geq d(x_o, s_f) - d(x_o, x) \geq k_f - k_c.$$

Hence the theorem is proved. \square

Acknowledgments

We would thank Forbes Burkowski, Tao Jiang, Paul Kearney, Ian Munro and Lusheng Wang for discussions on this research and especially Todd Wareham for helpful comments and providing us with some important references. We are also grateful to the referees and the SODA'99 program committee for pointing out errors and important Refs. [5,7,9] and helpful suggestions.

References

- [1] N. Alon, J. Spencer, P. Erdős, *The Probabilistic Method*, Wiley Interscience, New York, 1992.
- [2] E. Amaldi, V. Kann, The complexity and approximability of finding maximum feasible subsystems of linear relations, *Theoret. Comput. Sci.* 147 (1995) 181–210.
- [3] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and intractability of approximation problems, *Proceedings/Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, 1992, pp. 13–22.
- [4] T. Brown, G.A. Leonard, E.D. Booth, G. Kneale, Influence of pH on the conformation and stability of mismatch base-pairs in DNA, *J. Mol. Biol.* 212 (1990) 437–440.
- [5] A. Ben-Dor, G. Lancia, J. Perone, R. Ravi, Banishing bias from consensus sequences, *Combinatorial Pattern Matching*, 8th Annual Symposium, 1997, pp. 247–261.
- [6] R. Fagin, Generalized first-order spectra and polynomial time recognizable sets, in: R. Karp (Ed.), *Complexity of Computation*, American Mathematical Society, Providence, 1974, pp. 43–73.
- [7] M. Frances, A. Litman, On covering problems of codes, *Theor. Comput. Syst.* 30 (1997) 113–119.
- [8] M. Garey, D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

- [9] L. Gaśieniec, J. Jansson, A. Lingas, Efficient approximation algorithms for the Hamming center problem, in: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, pp. 905–907.
- [10] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, Cambridge, 1997.
- [11] M. Ito, K. Shimizu, M. Nakanishi, A. Hashimoto, Polynomial-time algorithms for computing characteristic strings, in: Proceedings of the 5th Symposium on Combinatorial Pattern Matching, 1994, pp. 274–288.
- [12] E.M. Hillis, C. Moritz, B.K. Mable, Molecular Systematics, 2nd ed., Sinauer Associates Inc., Sunderland, 1996.
- [13] T. Jiang, C. Trendall, S. Wang, T. Wareham, X. Zhang, Drug target identification using Gibbs sampling techniques, in: Pacific Symposium on Biocomputing, 2000, pp. 389–400.
- [14] J.K. Lanctot, M. Li, B. Ma, S. Wang, L. Zhang, Distinguishing String Selection Problems (Extended Abstract), in: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999, pp. 633–642.
- [15] A.J. Lomant, J.R. Fresco, Structural and energetic consequences of noncomplementary base oppositions in nucleic acid helices, *Nucleic Acid Res. Mol. Biol.* 15 (1975) 185–218.
- [16] C.E. Longfellow, R. Kierzek, D.H. Turner, Thermodynamic and spectroscopic study of bulge loops in oligoribonucleotides, *Biochemistry* 29 (1990) 278–285.
- [17] A. Macario, E. Macario, Gene Probes for Bacteria, Academic Press, San Diego, 1990.
- [18] C.K. Mathews, K.E. van Holde, Biochemistry, Benjamin/Cummings Publishing Company, Redwood City, 1990.
- [19] R. Motwani, J. Naor, P. Raghavan, Randomized approximation algorithms in combinatorial optimization, in: D.S. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing Company, Boston, 1995, pp. 447–481.
- [20] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
- [21] C.H. Papadimitriou, M. Yannakakis, Optimization, approximation and complexity classes, *JCSS* 43 (1991) 425–440.
- [22] D.H. Turner, N. Sugimoto, S.M. Freier, RNA structure prediction, *Annu. Rev. Biophys. Chem.* 17 (1988) 167–192.
- [23] J.G. Wetmur, DNA probes: Applications of the principles of nucleic acid hybridization, *Crit. Rev. Biochem. Mol. Biol.* 26 (1991) 227–259.