Distinguishing String Selection Problems *

J. Kevin Lanctot[†] University of Waterloo
Ming Li[‡] University of Waterloo
Bin Ma[§] City U. of Hong Kong
Shaojiu Wang[¶] Pasteur Merieux Connaught Canada
Louxin Zhang[||] Nat. Univ. of Singapore

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 (the 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 Subtring 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 $(\frac{4}{3}+\epsilon)$-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.
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.

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

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

†Address: Dept. of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada. E-mail: jklanctot@neumann.uwaterloo.ca.

‡Supported in part by the NSERC Research Grant OGP0046506, a CGAT grant, and the Steacie Fellowship. Address: Dept. of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada. E-mail: mli@math.uwaterloo.ca.

§Address: Dept. of Computer Science, City University of Hong Kong, Hong Kong. E-mail: csbma@cityu.edu.hk.

¶Address: Pasteur Merieux Connaught Canada, 1755 Steeles Avenue West, Toronto, Ontario, M2R 3T4, Canada. E-mail: jwang@ca.pmc-vacc.com.

||Address: BioInformatics Center, NUS and Kent Ridge Digital Labs, 21 Heng Mui Keng Terrace, Singapore 119613. E-mail: lxzhang@bic.nus.edu.sg

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 things, 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 [BL+90, LF75] (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 [LKT90] (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 [TSF88]) and are quantified in terms of Gibbs free energy. For a substitution, the free energy, G_s^o, is about 0.8 kcal/mol, and for a bulge G_b^o, 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 $e^{\Delta G^o/RT}$ [MH90] 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 may 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 [FL97, GJL99], 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. 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.

2.1 Finding Targets for Potential New Drugs

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 a sequence fragment that is more conserved in all or most of the pathogens' sequences but not as conserved in the host's. Information encoded by this fragment then can be used for novel antibiotic development. For example, chemicals can be screened to identify those that only bind to the peptide encoded by the conserved regions of the bacterial DNA. Those chemicals can then be tested as potential broad-range antibiotics. Jiang *et al.* have also looked at this problem, however they use Gibbs sampling to identify the drug targets [JT+98].

2.2 Antisense Drug Design

Another possible strategy would be to use antisense drugs. These compounds are a new strategy in designing highly specific drugs that inhibit the production of the targeted disease-related proteins while not interfering with other proteins [CL93]. They work by impeding the translation of targeted genes by hybridizing to a portion of the mRNA corresponding to that gene after it has been transcribed from DNA but before it can be translated into a protein. By identifying a sequence fragment that distinguishes the pathogens from the host, the potential exists to create a drug that harms several pathogens with minimal effect on the host. Because the targeted species are more closely related to each other than to humans, it would be reasonable to expect to find sequences that are fairly similar in all the set, yet dissimilar to the host's essential genetic sequences.

2.3 Creating Diagnostic Probes for Bacterial Infection

Besides having a potential application in the treatment of disease, these algorithms can also be used for diagnosis, such as the task of creating diagnostic probes for bacterial infections [BL+97, MM90]. Given a dataset of DNA sequences from a group of closely related pathogenic bacteria, the task would be to find a substring that occurs in each of the bacterial sequences (with as few substitutions as possible) without occurring in the host's DNA [BL+97]. Probes are then designed to hybridize to these target sequences and they can be labeled, such as being made radioactive, for easy detection, so that their presence provides an indication that at least one bacterial species is present in the host.

2.4 Creating Universal PCR Primers

Polymerase chain reaction (PCR) primer design is another area related to the problems described in this paper. PCR is a laboratory technique for amplifying, that is creating many copies of, a portion of DNA [HMM96]. In order for this method to work, pairs of primers (or short strands of single-stranded DNA) must be created that hybridize to specific locations in relation to the area to be amplified. The Distinguishing String Selection Problem occurs when one is trying to design a single pair of primers to amplify several regions simultaneously. Here the task is to discover a pair of primers such that each will hybridize in the same region relative to the portions to be amplified and nowhere else. Several existing programs such as Primer [LB+91], PCRPROF [DR+93], and Primer Master [PH96] are available to design universal primers. The Primer paper does not state the algorithm by which the targets are selected [LB+91], PCRPROF assumes that the sequences have already been aligned by another program [DR+93], and Primer Master uses an unspecified search to find variable and conserved regions and then picks the primer candidates from the conserved regions [PH96]. None of the programs surveyed do an analysis of the complexity of their algorithms.

2.5 Creating Unbiased Consensus Sequences

A final 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 [BL+97] and this task is exactly our Closest String Problem.

3 Formulation and Results

The previous applications can be formulated as the following set of problems using Hamming distance.

Closest String Problem
INSTANCE: Given a set $\mathcal{S}$ of strings of length n over an alphabet A.
OBJECTIVE: Find a string x of length n minimizing d such that for every string s_i, $d_H(x, s_i) \leq d$.

Closest Substring Problem
INSTANCE: Given a set $\mathcal{S}$ of strings of length at least n over an alphabet A.
OBJECTIVE: Find a string x of length n minimizing d such that for every string z in $\mathcal{S}$, $d_H(x, y) \leq d$ for some substring y (of length n) of z.

Max Close String Problem
INSTANCE: Given a set $\mathcal{S}$ of strings of length at least n over an alphabet A, and a threshold $d \geq 0$.
OBJECTIVE: Find a string x of length n maximizing the number of strings y in $\mathcal{S}$ satisfying the condition that for some substring z of y, $d_H(x, z) \leq d$.

Similarly, x cannot bind to y if $d_H(x, y) \geq d$ for some threshold $d > 0$. 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 problem of being far from a string, and far from any substring are identical from the point of view of complexity. Thus, the problem can be described as

Farthest String Problem
INSTANCE: Given a set $\mathcal{S}$ of strings of length n over an alphabet A and a threshold $d \geq 0$.
OBJECTIVE: Find a string x of length n such that for any $y \in \mathcal{S}$, $d_H(x, y) \geq d$.

Far from Most String Problem
INSTANCE: Given a set $\mathcal{S}$ of strings of length n over an alphabet A and a threshold $d \geq 0$.
OBJECTIVE: Find a string x of length n maximizing the number of strings y satisfying $d_H(x, y) \geq d$.

Finally, the Distinguishing String Selection Problem can be formalized as

Distinguishing String Selection Problem
INSTANCE: Given a set $\mathcal{B}$ of (bad) strings of length at least n, a set $\mathcal{G} \subseteq A^n$ of (good) strings and two thresholds d_b and d_g;
OBJECTIVE: Find a string x such that for each string $z \in \mathcal{B}$, $d_H(x, y) \leq d_b$ for some substring y (of length n) of z, and for any string $z \in \mathcal{G}$, $d_H(x, z) \geq d_g$.

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. Our 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 $\epsilon > 0$. Using these two algorithms, we also provide an efficient heuristic algorithm for the Closest Substring Problem. As well, there is a 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 Nearest 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 [GJ79]. Such a problem was proved to be NP-complete by Fagin [Fa74]. This problem is a special case of the Nearest 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 [AK95]. 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.* [IS+94]. 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 [IS+94]. 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.* [BL+97] 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

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

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 [BL+97] does not work well for small d's.

The Closest String Problem also occurs in coding theory, and has been proven NP-Complete for binary codes [FL97]. Again in the context of coding theory, Gąsieniec *et al.* [GJL99] independently claim a weaker $(\frac{4}{3}+\epsilon)$-approximation, with the requirement that d is large (super-logarithmic). Our $(\frac{4}{3}+\epsilon)$-approximation in this paper works for *all d*.

5 The Complexity of Farthest String Problem

In this section, we prove the Farthest String Problem is NP-hard. The proof for an alphabet size of two is relatively easy. 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. We reduce the 3-SAT problem to the Farthest String Problem.

To prove its NP-hardness, we reduce the 3-SAT problem to the Farthest String Problem. We only consider the case $|A| = 3$. Other cases are similar.

Given an instance I of the 3-SAT problem, we construct an instance of the Farthest String Problem as follows. Let I have k clauses $C_1, C_2, \cdots, C_k$ and contain n variables $x_1, x_2, \cdots, x_n$. Let $N = n+2$. For each clause C_i, we construct a string $s_i = s_{i1}s_{i2}\cdots s_{iN}$ of length N over $\{0, 1, \star\}$, where

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

We also set $\{s_{k+1}, s_{k+2}, \cdots, s_{k+9}\} = \star^n\{0,1,\star\}^2$.

Lemma 1 *The instance I is satisfiable if and only if there is a string s on $\{0, 1, \star\}$ such that $d_H(s, s_i) \geq n$ for every $i \leq k+9$.*

Proof. Suppose I is satisfied by an assignment $x_i = x'_i$, $i \leq n$. Define $s = x'_1x'_2\cdots x'_n00$. First, since s does not contain any $\star$, $d_H(s, s_{k+j}) \geq n$ for every $1 \leq j \leq 9$. For each s_i, $i \leq k$, if $x_l = x'_l$ makes C_i true, then the lth positions in s and s_i are a mismatch by definition. Hence, $d_H(s, s_i) \geq n$.

Conversely, suppose there is a string $s = x_1x_2\cdots x_{n+1}x_{n+2}$ such that $d_H(s, s_i) \geq n$ for every $i \leq k+9$. Then s does not contain any $\star$ in the first n positions. Otherwise, let $s_{k+l} = \star^n x_{n+1}x_{n+2}$

for some $l \leq 9$. Then it is not difficult to see that $d_H(s, s_{k+l}) \leq N - 3 < n$. This is a contradiction. Since s does not contain any $\star$ in the first n positions, it induces an assignment of variables x_i's. Such an assignment is a true assignment for I. $\square$

By Lemma 1, the Farthest String Problem is NP-hard. $\square$

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. We again reduce the 3-SAT problem to this problem. Let I be an instance of the 3-SAT problem with k clauses C_i $(1 \leq i \leq k)$ and n variables x_j $(1 \leq j \leq n)$. For each clause C_i, we construct a string $s_i = s_{i1}s_{i2}\cdots s_{i(2n-1)}s_{i(2n)}$ of length $2n$ over $\{0,1\}$, where $s_{i(2j-1)}s_{i(2j)} = 00$ if $x_j \in C_i$, 11 if $\overline{x}_j \in C_i$, and 01 otherwise.

We also let

$$t_0 = (01)^n,\ t_i = (01)^{i-1}10(01)^{n-i},\ 1 \leq i \leq n,$$

and let

$$p_0 = (10)^n,\ p_i = (10)^{i-1}01(10)^{n-i},\ 1 \leq i \leq n.$$

Finally, let, for i from 1 to n,

$$q_i' = (10)^{i-1}00(10)^{n-i},\ q_i'' = (01)^{i-1}00(01)^{n-i};$$

$$u_i' = (10)^{i-1}11(10)^{n-i},\ u_i'' = (01)^{i-1}11(01)^{n-i}.$$

The corresponding instance of the Farthest String Problem is

$$S_I = \{s_i\} \cup \{t_i\} \cup \{p_i\} \cup \{q_i', q_i''\} \cup \{u_i', u_i''\}$$

with the Hamming distance bound is $n-1$. Obviously, the instance $(S_I, n-1)$ is computable in polynomial time. The purpose of including t_i and p_i is to force that the solution to the instance $(S_I, n-1)$ will be a string in $(00,11)^n$.

Lemma 2 *If a string s is at least Hamming distance $n-1$ away from every string t_i, p_i, q_i', q_i'', u_i' and u_i'', then $s \in (00,11)^n$.*

Proof. Let $s \notin (00,11)^n$. Then, $s \in \{00,11\}^\star(10|01)\{01,11,10,01\}^\star$. We factor s into a product as

$$s = s_1 s_2 \cdots s_n,$$

where $s_i \in \{00,11,10,01\}$. Let n_{10} and n_{01} denote the numbers of 10's and 01's in the factorization. Without loss of generality, we assume that $n_{10} \leq n_{01}$.

If $0 < n_{10} \leq n_{01}$, then we assume that the first 10 appears in the ith position in the factorization, i.e. $s_i = 10$. Then $d_H(s, t_i) = (n - n_{10} - n_{01}) + 2(n_{10} - 1) \leq n - 2$ and this is a contradiction.

If $n_{10} = 0$ and $n_{01} > 0$, then, $d_H(s, t_0) = n - n_{10} \geq n - 1$, Thus, $n_{01} = 1$. Without loss of generality, we assume that the first 11 appears in the ith position (the case 00 is similar), then $d_H(s, u_i'') \leq n - 2$, a contradiction. $\square$

By the lemma, any solution to $(S_I, n-1)$ must be a string in $\{11,00\}^n$. Further, it is not difficult to see that any such a solution to $(S_I, n-1)$ induces a truth assignment to x_i satisfying every clause in I. $\square$

6 PTAS for the Farthest String Problem

In this section, we use a linear programming relaxation technique to obtain polynomial time approximation schemes (PTASs) for the Farthest String problem.

6.1 Binary case

First, we shall design a PTAS for the Farthest String Problem in the binary case. This will then be generalized to the non-binary case in Section 6.3. To this end, we first prove a lower bound result.

Lemma 3 *Given a set $\mathcal{S}$ of m strings each of length n over over the alphabet $\{0,1\}$. If $n > 10\ln m/\beta^2$ where $\beta < 1$ is any constant number, then there exists a string s such that the Hamming distance between s_i and s is at least $(1-\beta)\frac{n}{2}$ for every $s_i \in \mathcal{S}$.*

Proof. We prove it by a simple probabilistic argument. Let $s \in \{0,1\}^n$. Given a random string $t \in \{0,1\}^n$, the expectation value of the Hamming distance between s and t is $\frac{n}{2}$. Thus, by a Chernoff bound [MNR95],

$$\Pr(|d_H(s,t) - n/2| > \tfrac{\beta n}{2}) \leq \tfrac{2}{e^{0.19\beta^2 n}}$$

Since $n > 10\ln m/\beta^2$, $\Pr(\cup_{i\in I}|d_H(s_i,t) - n/2| > \frac{\beta n}{2}) < o(1)$. Thus, there exists a string s such that $d_H(s,s_i) \geq (1-\beta)n/2$ for all strings s_i. □

Let $s = s_1 s_2 \cdots s_n$ and $x = x_1 x_2 \cdots x_n$. The Hamming distance $d_H(s,x) = x'_1 + x'_2 + \cdots + x'_n$, where $x'_i = x_i$ if $s_i = 0$, $1 - x_i$ otherwise. Therefore, the Farthest String Problem can be abstracted as a zero-one integer linear program:

$$\begin{array}{lll} max\ \ D & & \\ \sum_{j=1}^{n} a_{ij}x_j \geq C_i + D & \forall i \in I; & \\ x_{ij} = 0,1 & \forall i,j; & \end{array} \qquad (1)$$

where $a_{ij} = 1, 0, -1$ and C_i is an integer derived from the ith strings.

We first relax the integrality constraints on x_j and solve the resulting linear program. Let $\overline{x}$ be the solution vector of the resulting linear program and the objective function have value $\overline{D}$. We now apply randomized rounding to restore integrality: for each i, independently set x_i to 1 with probability $\overline{x}_i$ and to 0 with probability $1 - \overline{x}_i$. This rounding procedure gives us an assignment x that makes the objective function D larger than $\overline{D} - \epsilon O(n^{2/3})$ with high probability. Since $D_{opt} \geq (1-\beta)\frac{n}{2}$, we have

$$\frac{D}{D_{opt}} \geq \frac{\overline{D} - \epsilon O(n^{2/3})}{D_{opt}} \geq 1 - \epsilon O(n^{-1/3}) \geq 1 - \epsilon.$$

6.2 Derandomization

Derandomizing the algorithm involves derandomizing the randomized rounding. One approach is using the method of conditional probabilities (for example, see [ASE9] or [MR95]). Here we omit the details. Thus, we have

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

Proof. Let $\epsilon > 0$ be any small number. If $n > 10\ln m/\beta^2$, the above algorithm output a solution with distance bound at least $(1-\epsilon)d_{opt}$. Otherwise, we have $2^n \leq m^{10/\beta^2}$. Hence, we can do an exhaustive search for finding an optimal solution. □

6.3 The general case

Now we give a PTAS for the Farthest String Problem for the general case. Recall that we are most interested in the 4-letter and 20-letter cases. Similar to Lemma 3, we can prove

Lemma 4 *If $n > 10 \ln m/\beta^2$ for any $\beta < 1$. Let A be an alphabet of k letters. Then, there exists a string s such that the Hamming distance between s_i and s is at least $(1-\beta)\frac{(k-1)n}{k}$ for every i.*

We use the same technique to get a PTAS. However, the zero-one integer program for the general case is more complicated. Let $A = \{a_1, a_2, \cdots, a_k\}$. Given two strings $s = s_1 s_2 \cdots s_n$, $x = x_1 x_2 \cdots x_n$, where $s_i, x_i \in A$ for every i. The Hamming distance $d_H(s, x)$ can be calculated as follows. For each bit position i $(1 \leq i \leq n)$, we introduce k variables x_{ij}, $1 \leq j \leq k$, compute $d_H(s, t)$ as

$$d_H(s,t) = \sum_{1 \leq i \leq n} f_i(x_{i1}, x_{i2}, \cdots, x_{ik}),$$

where $f_i(x_{i1}, x_{i2}, \cdots, x_{ik}) = \sum_{j \neq m} x_{ij} + 1 - x_{im}$ if $s_i = a_m$ and $x_{ij} = 0$ for $j \neq l$ and $x_{il} = 1$ if $x_i = a_l$. Therefore, the Farthest String Problem can be formulated as:

$$\begin{array}{ll} max\ \ D & \\ \sum_{i=1}^{k} \sum_{j=1}^{n} a_{lij} x_{ij} + n \geq D & \forall l \in I; \\ \sum_{j=1}^{k} x_{ij} = 1; & \forall i; \\ x_{ij} = 0, 1 & \forall i, j; \end{array} \tag{2}$$

We solve the linear program relaxation of (2), in which the integrality constraint is replaced by the constraints $0 \leq x_{ij} \leq 1$. Let $\overline{x} = (\overline{x}_{ij})$ be the solution of the linear program and let $\overline{D}$ be the value of the objective function for this solution. Let D_{opt} be the optimal value of the objective function in (2). Obviously, $D_{opt} \leq \overline{D}$. Since there are equality constraints on x_{ij}, $1 \leq j \leq k$, we do the following randomized rounding: independently for each i, set x'_{ij} to 1 and x'_{il}, $l \neq j$ to 0 with probability $\overline{x}_{ij}$. Thus, for each i, $\Pr(x'_{ij} = 1) = \overline{x}_{ij}$.

Since the solution of the linear program satisfy its constraints, we have for each l,

$$\sum_{i=1}^{k} \sum_{j=1}^{n} a_{lij} \overline{x}_{ij} + n \geq \overline{D}.$$

Let

$$D' = \sum_{i=1}^{k} \sum_{j=1}^{n} a_{lij} x'_{ij} + n.$$

Note that x'_{ij} is a Poisson trial with probability $\overline{x}_{ij}$. Further, x_{ij}'s are each independent of x'_{lj} for $i \neq l$. Therefore, D' is the sum of independent Poisson trials satisfying

$$E(D') = \sum_{i=1}^{k} \sum_{j=1}^{n} a_{lij} E(x'_{ij}) + n < \overline{D}.$$

Therefore, with high probability,

$$D' > \overline{D} - \epsilon O(n^{2/3}).$$

Since $D_{opt} > (1-\beta)\frac{(k-1)n}{k}$, this gives a PTAS after derandomization. Therefore, Theorem 3 can be generalized into the following result.

Theorem 4 *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 constant α and β such that for every instance x of P,

(1) $opt_Q(f(x)) \leq \alpha \cdot opt_P(x)$;

(2) for every solution y of $f(x)$ with objective value c_2, we can find in polynomial time a solution y' of x with objective value c_1 such that $|opt_P(x) - c_1| \leq \beta \cdot |opt_G(f(x)) - c_2|$ (see [PY91]).

Theorem 5 *For strings over an alphabet A with $|A| \geq 3$, approximating Far from Most String Problem within a factor n^ϵ is NP-hard for some fixed constant ϵ.*

Sketch of Proof. We first consider the case of $|A| = 3$. Given a graph $G = (V, E)$, we construct an instance (S_G, d) of the Far from Most String Problem as follows. Let $V = \{v_1, v_2, \cdots, v_n\}$ and $E = \{e_1, e_2, \cdots, e_m\}$. For each vertex v_i, we construct a string $s_i = s_{i1}s_{i2}\cdots s_{im} \in \{0, 1, \star\}$ of length m. For each position $j \leq m$, we 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, $S_G = \{s_1, s_2, \cdots, s_n\}$ and the Hamming distance bound is $d = m$. Obviously, the transformation from G to (S_G, m) is computable in polynomial time.

Without loss of generality, we may assume that G is not a complete graph. Let s' be an optimal solution for the instance (S_G, m) and let $S' = \{s \in S_G \mid d_H(s', s) \geq m\}$. If s' contains a $\star$ in some position j, then, s' can mismatch at most 2 strings at position j, which corresponds to the endpoints of the jth edge in G. Thus, $|S'| \leq 2$. Since G is not complete, the optimal solution V' for G has at least two vertices. Thus, $opt_I(G) \geq opt_H((S_G, m))$. If s' does not contain any $\star$'s, then, the vertices corresponding to strings in S' consists of an independent set for G. Hence, we still have that $opt_I(G) \geq opt_H((S_G, m))$.

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 letter 0 or 1. Thus, we can easily construct a string that is Hamming distance m far from each string in S'. Hence, we actually have $opt_I(G) = opt_H((S_G, m))$. The above argument also implies that given an approximation S'' of the problem (S_G, m), we can easily get an independent set V'' such that $|opt_I(G) - |V''|\,| \leq |opt_H((S_G, m)) - |S''|\,|$. Therefore, the Independent Set problem is L-reduced to the From From Most String Problem. From the hardness result on the Independent Set problem [AL+92], our theorem follows in the case of $|A| = 3$.

The above construction has to be modified in the case of $|A| \geq 4$. For example, in the case of $|A| = 4$, the construction can be modified as follows. We only consider graphs G that have an independent set containing at least $|A| = 4$ vertices. For such a graph $G = (V, E)$, we construct a set S_4 of n strings from $S = \{s_1, s_2, \cdots, s_n\}$ constructed in above proof by substituting 0 with 0^n, 1 with 1^n, and the ith $\star$ 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 $n|E|$. The relationship between G and $(S_4, n|E|)$ are the same: $opt_I(G) = opt_H((S_4, n|E|))$; for any solution x' of $(S_4, n|E|)$, let S' consist of all strings s' satisfying $d_H(x, s') \leq n|E|$. Then, all vertices corresponding to strings $s' \in S'$ form an independent set V' of G which satisfies that

$$|opt_I(G) - |V'|\,| \leq |opt_H((S_4, n|E|)) - |S'|\,|.$$

8 Approximation of the Closest String and Substring Problems

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

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

Proof.

We first consider the case of $|A| = 2$. Given an integer d and a set of n strings $\mathcal{S} = \{s_1, s_2, \ldots s_n\}$ each of length m, the following statements are all equivalent for $1 \leq i \leq n$. There is a string t such that $d_H(t, s_i) \geq d$. There are strictly less than $m - d$ mismatches between $\bar{t}$, the complement of t, and s_i. $d_H(\bar{t}, s_i) \leq m - d - 1$.

The case $|A| > 2$ can be reduced to the binary case as follows. Given an instance of the binary Closest Substring Problem, we can use the non-binary subroutine (if it is not NPC) to solve it to within distance d. Then take that non-binary solution, convert each letter that's not in $\{0, 1\}$ to $\{0, 1\}$ arbitrarily — this will only improve the quality of the solution. Hence this is also a binary solution within distance d. So the non-binary case is also NP complete.

□

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

Theorem 7 *The Closest Substring Problem is NP-hard.*

Given an instance S of the Closest Substring Problem, in which each string is of length at least n, let x be an optimal solution such that for each string $s_i \in S$, $d_H(x, y_i) \leq d_{opt}$ for some substring y_i of s_i. Then, $d_H(y_1, y_i) \leq 2d_{opt}$. Since y_1 is a substring of s_1, a substring z of s_1 can be obtained in polynomial time (in terms of $|S| = \sum |s_i|$) which satisfies the property that $d_H(z, z_i) \leq 2d_{opt}$ for some substring z_i of each s_i. This gives a simple polynomial-time approximation with ratio 2.

Theorem 8 *The Closest Substring Problem can be approximated within ratio 2 in polynomial time.*

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

Theorem 9 *The Closest String Problem can be approximated with $\frac{4}{3}(1+\epsilon)$ in polynomial time for any small constant ϵ.*

Proof.

Let $\mathcal{S}$ have m strings each of length n and let $s_i = s_{i1}s_{i2}\cdots s_{in}$, $s_{ij} \in A$ for each $s_i \in S$ and $1 \leq i \leq n$. We may assume that $d_H(s_1, s_2) \geq d_H(s_i, s_j)$ for any $i, j \leq m$ by reordering s_i's if necessary. Let $k = d_H(s_1, s_2)$. Without loss of generality, we may assume that k mismatches between s_1 and s_2 occur on the first k positions, i.e. $s_{1j} \neq s_{2j}$ if $1 \leq j \leq k$ and $s_{1j} = s_{2j}$ if $k < j \leq n$. This is because for any permutation π on $\{1, \cdots, n\}$, S has an optimal solution $x = x_1x_2\cdots x_n$ with the minimum distance bound d_{opt} if and only if $\pi(S) = \{s_{i\pi(1)}s_{i\pi(2)}\cdots s_{i\pi(n)} \mid 1 \leq i \leq m\}$ has the optimal solution $\pi(x) = x_{\pi(1)}x_{\pi(2)}\cdots x_{\pi(n)}$ with the same minimum bound d_{opt}. Finally, let $s'_i = s_{i1}s_{i2}\cdots s_{ik}$ and $s''_i = s_{i(k+1)}s_{i(k+2)}\cdots s_{in}$ for each $i \leq m$. We derive an integer program from s'_i as

$$\begin{aligned} &\min \ d; \\ &d_H(s, s'_i) \leq d - d_H(s''_1, s''_i), \ \ i = 1, \cdots, m. \end{aligned} \tag{3}$$

where s is an unknown string of length k. Note that we have not written out the precise form of the integer program and also omitted the zero-one constraints. Let the optimal bound for IP (3) be $d_{1,opt}$. It is impossible to compute $d_{1,opt}$ efficiently. Therefore, we approximate it by considering two cases. Let $\delta > 1$ be some fixed constant.

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

We do a simple exhaustive search for finding an optimal solution s_x for IP (3). 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 (3). Let $\overline{d}$ be the value of the objective function of the relaxation and let $\overline{x}_i$ be its solution. Clearly, $d_{1,opt} \geq \overline{d}$. All $\overline{x}_i$'s are fractional values and therefore may not constitute a feasible solution to IP (3). We therefore "round" these fractional values to 0's and 1's to obtain a feasible solution for (3). The rounding process is similar to that found in Lemmas 3 and 4. Such a rounding process results in a random string s_x. Consider some fixed sequence $s_i \in S$. The expected value of the distance between s_x and s_i satisfies the following:

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

Since $d_{opt} \geq \frac{k}{2} > \frac{3 \ln(\delta m)}{\epsilon^2}$,

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

Using Hoeffding's bound (See, e.g., [MR95]), we get

$$Pr(d(s_x, s_i') > d_{1,opt} + \epsilon d_{opt} - d_H(s_1'', s_i'')) \leq \frac{1}{\delta m}.$$

Summing over every string S_i, we have

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

Therefore, with high probability,

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

for every i.

To derived a polynomial-time approximation for (3), we derandomized this randomized algorithm using techniques of conditional probabilities (for example, see [ASE9] and [MR95]). Here we omit the details.

In summary, our approximation algorithm consists of two steps: 1) Solve IP (3) 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 $\min(d_H(s_1, s_2), d_1)$.

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 $d_H(s_1, s_2) \leq \frac{4}{3} d_{opt}$, then, obviously, the solution given by our algorithm is a string with distance bound at most $\frac{4}{3} d_{opt}$.

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

$$d_H(s_1', s_{opt}') + d_H(s_2', s_{opt}') \geq \frac{4}{3} d_{opt}.$$

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

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

Similarly, if $d_H(s'_2, s'_{opt}) > \frac{2}{3}d_{opt}$, the above inequality still holds because $s''_1 = s''_2$. Thus, we have that $d_{1,opt} \leq \frac{4}{3}d_{opt}$. This implies

$$d_H(x_1, s_i) \leq d_{1,opt} + \epsilon d_{opt} - d_H(s''_1, s''_i) \leq \frac{4}{3}(1+\epsilon)d_{opt} - d_H(s''_1, s''_i).$$

Thus,

$$d_H(x_1 s''_1, s_i) = d_H(x_1, s'_i) + d_H(s''_1, s''_i) \leq \frac{4}{3}(1+\epsilon)d_{opt}.$$

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

Theorem 9 can be used to design an efficient heuristic algorithm for the Closest Substring Problem. The algorithm consists of two phases. Let $S = \{s_i \mid i \leq n\}$ be the input string set in which each string has length no less than n. First, we find substring y_i with length n of s_i such that $\max_{i,j} d_H(y_i, y_j)$ reaches the minimum over all such substrings. Then, we apply the algorithm in Theorem 9 to get a better solution x.

9 Distinguishing String Selection Problem

Since both the Farthest and Closest Substring Problems are its special cases, the Distinguishing String Selection Problem is NP-hard. Let $(S_b, S_g, n, A, d_b, d_g)$ be an instance of the problem. We are required to find a string x of length n such that for each $s \in S_b$, $d_H(x, z) \leq d_b$ for some substring z of s, and for each $t \in S_g$, $d_H(x, t) \geq d_g$. When d_b is a small constant, a solution can be searched by brute force. This is because that there are at most $O(\max_{s\in S_b} |s|n^{d_b}(|A|-1)^{d_b})$ candidates and for each candidate, whether it is a solution can be verified in polynomial time in term of the size of the instance. When d_b is relatively large, we have the following approximation result.

Theorem 10 *Let $(S_b, S_g, n, A, d_b, d_g)$ 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 $s \in S_b$, $d_H(x, y) \leq 2d_b$ for some substring y of s, and $d_H(x, s') \geq d_g - d_b$.*

Sketch of Proof. Suppose x is a feasible solution, and x' is a substring of a sequence in S_b such that $d_H(x, x') \leq d_b$. For any $s \in S_b$, there is a substring y of s such that $d_H(x, y) \leq d_b$, therefore $d_H(x', y) \leq 2d_b$. For any $s \in S_g$, since $d_H(x, s) \geq d_g$, $d_H(x', s) \geq d_(x, s) - d_(x, x') \geq d_g - d_b$.

10 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 SODA'99 program committee and the referees for pointing out errors and important references [BL+97, FL97, GJL99].

References

[ASE9] Alon, N., Spencer, J., and Erdős, P. (1992), "The Probabilistic Method," Wiley Interscience, New York.

[AK95] Amaldi, E., and Kann, V. (1995), The complexity and approximability of finding maximum feasible subsystems of linear relations, *Theoret. Comput. Science* **147**, 181-210.

[AL+92] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. (1992), Proof verification and intractability of approximation problems, *in* "Proceedings / Symposium on Foundations of Computer Science" IEEE Computer Society Press, Los Alamitos, pp. 13-22.

[BL+90] Brown, T., Leonard, G. A., Booth, E. D., and Kneale, G. (1990), Influence of pH on the Conformation and Stability of Mismatch Base-pairs in DNA, *J. Mol. Biol.* **212**, 437-440.

[BL+97] Ben-Dor, A., Lancia, G., Perone, J., and Ravi R. (1997), Banishing Bias from Consensus Sequences, *in* "Combinatorial Pattern Matching, 8th Annual Symposium," pp. 247-261.

[CL93] Crooke, S., and Lebleu, B. (editors) (1993), "Antisense Research and Applications," CRC Press Inc, Boca Raton.

[DR+93] Dopazo, J., Rodríguez, A., Sáiz, J. C., and Sobrino, F. (1993), Design of primers for PCR amplification of highly variable genomes, *CABIOS* **9**, 123-125.

[Fa74] Fagin, R. (1974), Generalized first-order spectra and polynomial time recognizable sets, *in* "Complexity of Computation" (R. Karp, Ed.), pp. 43-73, America Mathematical Society, Providence.

[FL97] Frances, M., and Litman, A. (1997) On Covering Problems of Codes, *Theory of Computing Systems* **30**, 113-119.

[GJ79] Garey, M., and Johnson, D. (1979), "Computers and Intractability: a guide to the theory of NP-completeness," W.H. Freeman, San Francisco.

[GJL99] Gąsieniec, L., Jansson, J., and Lingas, A., (1999) Efficient Approximation Algorithms for the Hamming Center Problem, to appear in "Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms."

[IS+94] Ito, M., Shimizu, K., Nakanishi, M., and Hashimoto, A. (1994), Polynomial-time algorithms for computing characteristic strings, *in* "Proceedings, 5th Symposium on Combinatorial Pattern Matching," pp. 274-288.

[HMM96] Hillis, E. M., Moritz, C., and Mable, B. K. (1996), "Molecular Systematics," 2nd ed., Sinauer Associates Inc., Sunderland.

[JT+98] Jiang, T., Trendall, C., Wang, S., Wareham, T., and Zhang, X. (1998) Drug target identification using Gibbs sampling techniques, submitted for publication.

[LF75] Lomant, A. J., and Fresco, J. R. (1975), Structural and Energetic Consequences of Non-complementary Base Oppositions in Nucleic Acid Helices *Nucleic Acid Res. and Mol. Bio.* **15**, 185-218.

[LKT90] Longfellow, C. E., Kierzek, R., and Turner, D. H.(1990), Thermodynamic and Spectroscopic Study of Bulge Loops in Oligoribonucleotides, *Biochemistry* **29**, 278-285.

[LB+91] Lucas, K., Busch, M., Mössinger, S., and Thompson, J. A. (1991), An improved microcomputer program for finding gene- or gene family-specific oligonucleotides suitable as primers for polymerase chain reactions or as probes, *CABIOS* **7**, 525-529.

[MM90] Macario, A., and Macario, E. (1990), "Gene Probes for Bacteria," Academic Press, San Diego.

[MH90] Mathews, C. K., and van Holde, K.E. (1990), "Biochemistry," Benjamin/Cummings Publishing Company, Inc., Redwood City.

[MNR95] Motwani, R., Naor, J., and Raghavan, P., (1995), Randomized Approximation Algorithms in Combinatorial Optimization *Approximation Algorithms for NP-Hard Problems* (D. S. Hochbaum, Ed.), pp. 447-481, PWS Publishing Company, Boston.

[MR95] Motwani, R., and Raghavan, P. (1995), "Randomized Algorithms," Cambridge University Press, Cambridge.

[PY91] Papadimitriou, C. H., and Yannakakis, M. (1991), Optimization, approximation and complexity classes, *JCSS* **43**, 425-440.

[PH96] Proutski, V., and Holme, E. C. (1996), Primer Master: a new program for the design and analysis of PCR primers, *CABIOS* **12**, 253-255.

[TSF88] Turner, D. H., Sugimoto, N., and Freier, S. M. (1988), RNA Structure Prediction, *Ann. Rev. Biophys. Chem.* **17**, 167-192.

[We91] Wetmur, J. G. (1991), DNA Probes: Applications of the Principles of Nucleic Acid Hybridization, *Critical Rev. in Biochem. and Mol. Bio.* **26**, 227-259.