On Approximating String Selection Problems with Outliers

Christina Boucher∗ Gad M. Landau†‡ Avivit Levy§¶ David Pritchard∥
Oren Weimann†

February 14, 2012

Abstract

Many problems in bioinformatics are about finding strings that approximately represent a collection of given strings. We look at more general problems where some input strings can be classified as outliers. The Close to Most Strings problem is, given a set $S$ of same-length strings, and a parameter $d$, find a string $x$ that maximizes the number of “non-outliers” within Hamming distance $d$ of $x$. We prove this problem has no PTAS unless $\text{ZPP} = \text{NP}$, correcting a decade-old mistake. The Most Strings with Few Bad Columns problem is to find a maximum-size subset of input strings so that the number of non-identical positions is at most $k$; we show it has no PTAS unless $\text{P} = \text{NP}$. We also observe Closest to $k$ Strings has no EPTAS unless $\text{W}[1] = \text{FPT}$. In sum, outliers help model problems associated with using biological data, but we show the problem of finding an approximate solution is computationally difficult.

1 Introduction

With the development of high-throughput next generation sequencing technologies, there has arisen large amounts of genomic data, and an increased need for novel ways to analyze this data. This has inspired numerous formulations of biological tasks as computational problems. In light of this observation, Lanctot et al. [15] initiated the study of distinguishing string selection problems, where we seek a representative string satisfying some distance constraints from each of the input strings. We will mostly have constraints in the form of an upper bound on the Hamming distance, but lower bounds on the Hamming distance, and substring distances, have also been considered [6, 11, 15].

Typically, the distance constraint must be satisfied for each of the input strings. However, biological sequence data is subject to frequent random mutations and errors, particularly in specific segments of the data; requiring that the solution fits the entire input data is problematic for many problems in bioinformatics. It would be preferable to find the similarity of a portion of the input strings, excluding a few bad reads that have been corrupted, rather than trying to fit the complete set of input and in doing so finding one that is distant from many or all of the strings.

∗Department of Computer Science and Engineering, University of California, San Diego
†Department of Computer Science, University of Haifa, Mount Carmel, Haifa 31905, Israel.
‡Polytechnic Institute of NYU, NY 11201-3840, USA.
§Shenkar College for Engineering and Design, Ramat-Gan, 52526, Israel
¶CRI, University of Haifa, Mount Carmel, Haifa 31905, Israel
∥CEMC, University of Waterloo, Canada
What if we are given a measure of goodness (e.g., distance) the representative must satisfy, and want to choose the largest subset of strings with such a representative? Conversely, what if we specify the subset size and seek a representative that is as good as possible? Some results are known in this area with respect to fixed-parameter tractability [5]. Here, we prove results about the approximability of three string selection problems with outliers. For any two strings $x$ and $y$ of same length, we denote the Hamming distance between them as $d(x, y)$, which is defined as the number of mismatched positions. Our main results are about three NP optimization problems.

**Definition 1.** Close to Most Strings (a.k.a. Max Close String [15, 20])

**Input:** $n$ strings $S = \{s_1, \ldots, s_n\}$ of length $\ell$ over an alphabet $\Sigma$, and $d \in \mathbb{Z}_+$.  
**Solution:** a string $s$ of length $\ell$.  
**Objective:** maximize the number of strings $s_i$ in $S$ that satisfy $d(s, s_i) \leq d$.

**Definition 2.** Closest to $k$ Strings

**Input:** $n$ strings $S = \{s_1, \ldots, s_n\}$ of length $\ell$ over an alphabet $\Sigma$, and $k \in \mathbb{Z}_+$.  
**Solution:** a string $s$ of length $\ell$ and a subset $S^*$ of $S$ of size $k$.  
**Objective:** minimize $\max\{d(s, s_i) \mid s_i \in S^*\}$.

In the special case $k = n$, Closest to $k$ Strings becomes Closest String — an NP-hard problem [9] that has received significant interest in parameterized complexity and approximability [1, 2, 10, 15, 17, 22, 23].

We also consider a problem where the “outliers” are considered to be positions (“columns”) rather than strings (“rows”). Let $s(j)$ indicate the $j$th character of string $s$.

**Definition 3.** Most Strings with Few Bad Columns

**Input:** $n$ strings $S = \{s_1, \ldots, s_n\}$ of length $\ell$ over an alphabet $\Sigma$, and $k \in \mathbb{Z}_+$.  
**Solution:** a subset $S^* \subseteq S$ of strings such that the number $\{t \in [\ell] \mid \exists s_i^*, s_j^* \in S^*: s_i^*(t) \neq s_j^*(t)\}$ of bad columns is at most $k$.  
**Objective:** maximize $|S^*|$.

In other words, a column $t$ is bad when its entries are not-all-equal, among strings in $S^*$. The Most Strings with Few Bad Columns Problem generalizes the problem of finding tandem repeats in a string [16].

### 1.1 Our contributions

A PTAS for a minimization problem is an algorithm that takes an instance of the problem and a parameter $\epsilon > 0$ and, in time that is polynomial for any fixed $\epsilon$, produces a solution that is within a factor $1 + \epsilon$ of being optimal. An efficient PTAS (EPTAS) further restricts the running time to be some function of $\epsilon$ times a constant-degree polynomial in the input size. We present several results on the computational hardness of efficiently finding an approximate solution to the above optimization problems. Specifically, we show the following:

- The Close to Most Strings Problem has no PTAS, unless ZPP = NP (Theorem 1).
- The Most Strings with Few Bad Columns Problem has no PTAS, unless P = NP (Theorem 2).
- We observe that the known PTAS [20] for the Closest to $k$ Strings Problem cannot be improved to an EPTAS, unless W[1] = FPT.
Our first result corrects an error in prior literature. A problem is APX-hard if for some fixed \(\epsilon > 0\), finding a \((1+\epsilon)\)-approximation is NP-hard. A 2000 paper of Ma \([20]\) claims that the Close to Most Strings problem is APX-hard; however, the reduction is erroneous. To explain, it is helpful to define one more problem, Far from Most Strings, which is the same as Close to Most Strings except that we want to maximize the number of strings \(s_i\) in \(S\) that satisfy \(d(s, s_i) \geq d\) (rather than \(\leq\)).

There is considerable experimental interest in heuristics for Far from Most Strings, mostly based on local search \([19, 7, 8]\). Far From Most Strings was introduced and studied by Lanctot et al. \([15]\), and they (correctly) showed that for any fixed alphabet size greater than or equal to three, Far from Most Strings is at least as hard to approximate as Independent Set. Currently, Independent Set is known \([14]\) to be inapproximable within a factor of \(n/2^{\log^{3/4+\epsilon} n}\) unless \(\text{NP} \subseteq \text{BPTIME}(2^{\log^{O(1)} n})\).

The main idea in Ma’s approach was to consider a binary alphabet. In detail, the Far from Most Strings and Close to Most Strings Problem on alphabets \(\Sigma = \{0, 1\}\) are basically the same problem, since a string \(s\) of length \(\ell\) has distance at most \(d\) from \(s_i\) if and only if the complementary string \(\bar{s}\) has distance at least \(\ell - d\) from \(s_i\). The crucial error in \([20]\) is that Ma mis-cited \([15]\), assuming that their result worked on binary alphabets. (One reason why the approach of \([15]\) does not extend to binary alphabets in any obvious way is that the instances produced by their reduction satisfy \(d = \ell\), whereas Far from Most Strings is easy to solve when \(|\Sigma| = 2\) and \(d = \ell\).)

From \([15]\) and \([20]\) we cannot conclude anything about the hardness of Close to Most Strings, nor can we say anything about the hardness of Far from Most Strings when \(|\Sigma| = 2\). Our results close both of these gaps: the proof of Theorem 1 actually shows Close to Most Strings is hard over a binary alphabet, from which it follows that Far from Most Strings is, too. At the same time, the hardness that we are able to achieve is much more modest than the previous claim; we show only that there is no 1.001-approximation. We also require a randomized reduction. It is a very interesting open problem to determine whether this problem has any constant-factor approximation, even over a binary alphabet.

1.2 Brief Description of Parameterized Complexity

Some parameterized complexity concepts will arise in later sections, so we give a birds-eye view of this area. With respect to a parameter \(k\), a decision algorithm with running time \(f(k)n^{O(1)}\) (where \(n\) is the input length) is called fixed parameter tractable (FPT); the class \(\text{FPT}\) contains all parameterized problems with FPT algorithms. The corresponding reduction notion between two parameterized problems is an FPT reduction, which is FPT, and also increases the parameter by some function that is independent of the instance size. The class \(\text{W}[1]\) is a superset of \(\text{FPT}\) closed under FPT-reductions. A problem is \(\text{W}[1]\)-hard if any \(\text{W}[1]\) problem can be FPT-reduced to it, and \(\text{W}[1]\)-complete if it is both in \(\text{W}[1]\) and \(\text{W}[1]\)-hard. There are many natural \(\text{W}[1]\)-complete problems, like Maximum Clique parameterized by clique size. It is widely hypothesized that \(\text{FPT} \subsetneq \text{W}[1]\), but unproven, analogous to \(P \subsetneq \text{NP}\).

2 Approximation Hardness of Close to Most Strings

**Theorem 1.** For some \(\epsilon > 0\), if there is a polynomial-time \((1+\epsilon)\)-approximation algorithm for the Close to Most Strings Problem, then \(\text{ZPP} = \text{NP}\).

**Proof.** We use a reduction from the Max-2-SAT Problem, which is to determine for a given 2-CNF formula, an assignment that satisfies the maximum number of clauses. Let \(X = \{x_1, \ldots, x_n\}\) be
a set of Boolean variables. In 2-CNF, each clause is a disjunction of two literals, each of which is either $x_i$ or $\overline{x}_i$ for some $i$. Håstad [12] showed it is NP-hard to compute a $22/21$-approximately optimal solution to Max-2-SAT, and this is the starting point for our proof. We will assume that $m \geq n$, i.e. the number of clauses is greater than or equal to the number of variables, which is without loss of generality since otherwise some variable appears in at most one clause and the instance can be reduced.

We give a schematic overview of our reduction in Figure 1. The reduction will be randomized. It takes as input an instance of Max-2-SAT with $m$ clauses and $n$ variables. The reduction’s output is an instance of Close to Most Strings with $cm + m$ strings of length $2n$ for some constant $c$, and the distance parameter of the instance is $d = n$. Of these strings, $cm$ will be “fixing” strings to enforce a certain structure in near-optimal solutions, and the remaining $m$ strings are defined from the clauses as follows. Given a 2-clause $\omega_j$ over the variables in $X$, we define the corresponding string $s_j = s_j(1) \ldots s_j(2n)$ as follows:

$$s_j(2i-1)s_j(2i) = \begin{cases} 00 & \text{if } \omega_j \text{ contains the literal } \overline{x}_i, \\ 11 & \text{if } \omega_j \text{ contains the literal } x_i, \\ 01 & \text{otherwise.} \end{cases}$$

The fixing strings will all be elements of $\{0,1\}^n$, selected independently and uniformly at random.

We now give a high-level explanation of the proof. For every variable assignment vector $x$ define a string $\hat{x}$ via

$$\hat{x}(2i-1)\hat{x}(2i) = \begin{cases} 11 & \text{if } x_i \text{ is true,} \\ 00 & \text{if } x_i \text{ is false.} \end{cases}$$

Notice that $\hat{x}$ is at distance exactly $d = n$ from all of the fixing strings, and that $d(\hat{x}, s_j) \leq n$ if and only if $x$ satisfies clause $\omega_j$. Hence, if $x$ satisfies $k$ clauses, the string $\hat{x}$ is within distance $d$ of $cm + k$ out of the $cm + m$ total strings. We will show conversely that with high probability, for all strings $s$ within distance $d$ of $cm$ of the strings, we have $s \in \{00,11\}^n$. Using this crucial structural...
claim, it follows that any sufficiently good approximation algorithm for Close to Most Strings must output s such that s = ̂x for some x. Then the claim will be complete via standard calculations.

Here is the precise statement of the structural property.

**Lemma 1.** For c ≥ 20, the following holds. Let F be a set of cm strings selected uniformly and independently at random from \{01, 10\}^n (with replacement), with m ≥ n. Then with probability at least 1 − 0.9^n, every string s ∈ \{0, 1\}^n \{00, 11\}^n has distance greater than n from at least m strings in F.

**Proof.** To explain the proof more simply, fix s and consider a particular f ∈ F. By hypothesis, this s satisfies s(2i − 1) ≠ s(2i), say s(2i − 1) = 0 and s(2i) = 1 (the other case is symmetric). Since f is chosen uniformly at random from \{01, 10\}^n, the event \mathcal{E} where f(2i − 1) = 1 and f(2i) = 0 has Pr[\mathcal{E}] = 1/2. A short calculation which we postpone momentarily shows that Pr[d(s, f) ≥ n + 1 | \mathcal{E}] ≥ 1/2. So unconditioning, Pr[d(s, f) ≥ n + 1] = Pr[d(s, f) ≥ n + 1 | \mathcal{E}] · Pr[\mathcal{E}] ≥ 1/4.

Let us verify now that Pr[d(s, f) ≥ n + 1 | \mathcal{E}] ≥ 1/2. Observe that d(s, f) is a sum of n independent random variables d(s(2j − 1)s(2j), f(2j − 1)f(2j)) for j from 1 to n; conditioning on \mathcal{E} just fixes one of these variables at 2. The remaining ones are either always 1 (if s(2j − 1) = s(2j)), or a uniformly random element of \{0, 2\}. The conditioned random variable d(s, f) | \mathcal{E} is thus a shifted and scaled binomial distribution, in particular it is symmetric about n + 1. So Pr[d(s, f) ≥ n + 1 | \mathcal{E}] = Pr[d(s, f) ≤ n + 1 | \mathcal{E}] and since these two probabilities’ sum is at least 1, Pr[d(s, f) ≥ n + 1 | \mathcal{E}] ≥ 1/2 follows.

We will continue reasoning about this fixed s, and use a Chernoff bound to get large enough probability to work for all possible s. Let F = \{f_1, \ldots, f_{cm}\} and let X_i be an indicator variable for the event that d(f_i, s) > n. We have argued that each X_i is 1 with probability at least 1/4. Therefore, \mathbb{E}[\sum_i X_i] ≥ cm/4. We will use a Chernoff bound of the following form:

**Claim 1 (Lower Chernoff bound, [21]).** If X is a sum of independent 0-1 random variables, then we have

\[ \Pr[X < (1 - \delta)\mathbb{E}[X]] < \exp(-\mathbb{E}[X]\delta^2/2). \]

Choose δ so that (1 − δ)cm/4 = m, i.e. δ = 1 − 4/c. Then Pr[X < m] ≤ Pr[X < (1 − δ)\mathbb{E}[X]] < \exp(-cm/4 · (1 − 4/c)^2/2) = \exp\left(-\frac{(c-4)^2}{8c}m\right). By a union bound over all 4^n − 2^n possible choices of s, the probability that a random choice of F admits any bad s is at most

\[(4^n − 2^n) \exp\left(-\frac{(c-4)^2}{8c}m\right) < 4^n \exp\left(-\frac{(c-4)^2}{8c}n\right) = \exp\left((\ln 4 - \frac{(c-4)^2}{8c})n\right).\]

Any large enough c makes this exponentially decreasing in n; it is straightforward to calculate that when c = 20 this is at most 0.9^n, as needed. □

Now let us complete the overall proof; fix c = 20. Given a Max-2-SAT instance, we run the randomized reduction above to get an instance of Close to Most Strings. Let s_A be a (1 + ϵ)-approximation for this instance, where ϵ will be a small constant fixed later to satisfy two properties.

Let k^* be the number maximum of satisfiable clauses in the Max-2-SAT instance. As an important technicality, note that k^* is lower-bounded by m/2, since the expected number of clauses satisfied by a random assignment is at least m/2, by linearity of expectation. So the optimal solution to the Close to Most Strings instance has value at least cm + m/2.

First we want to use the structural lemma (Lemma 1). Assume for now the bad event with probability 0.9^n does not happen; so every s /∈ \{00, 11\}^n (i.e. not of the form s = ̂x) is within
distance $d$ of at most $cm$ of the $(c+1)m$ strings. Thus provided that $\epsilon$ is small enough to satisfy $1 + \epsilon < \frac{cm + m/2}{cm} = 1 + \frac{1}{2c}$, then $s_A$ is of the form $\hat{x}_A$ for some $x_A$.

Next we finish the typical calculations in a proof of APX-hardness. We know that $s_A$ is within distance $d$ of at least $(cm + k^*)/(1 + \epsilon)$ strings. If we can pick $\epsilon$ so that $cm + k^* > cm + 21 \frac{1}{22} k^*$, then $\hat{x}_A$ satisfies more than $21 \frac{1}{22} k^*$ clauses, which is NP-hard by Håstad’s result. Using that $k^* \geq m/2$, it is easy to verify that (1) holds for all $\epsilon < 1/(21 + 44c)$.

Finally, we confirm that the randomized algorithm for Max-2-SAT coming from the reduction is ZPP-style, i.e. Las Vegas style. As long as the output $s_A$ of the Close to Most Strings approximation algorithm satisfies $s_A \notin \{00, 11\}^n$ we re-create the reduction again using fresh random bits and re-run the approximation algorithm. But once $s_A \in \{00, 11\}^n$ we know for certain that $\hat{x}_A$ is a $22/21$-approximate solution for Max-2-SAT, as needed. The expected number of trials is at most $1/(1 - 0.9^n) = O(1)$.

3 Non-existence of an EPTAS for Closest to $k$ Strings

Ma showed in [20] that the Closest to $k$ Strings problem has a PTAS, which contrasts with the APX-hardness we obtain for the other problems in this paper. A natural question that comes up after a PTAS is obtained, is whether the running time can be improved to an EPTAS, or even further to a FPTAS (running time polynomial in the input length and $\epsilon^{-1}$). We observe there does not exist an EPTAS for Closest to $k$ Strings when the alphabet is unbounded, unless W[1] = FPT. To see this, we use a well-known fact relating fixed-parameter algorithms to the notion of an EPTAS, e.g. see [18], along with the fact that the decision version of Closest to $k$ Strings is W[1]-hard when parameterized by $d$ [5].

In detail, suppose for the sake of contradiction that we had an EPTAS for Closest to $k$ Strings, i.e. that one could obtain a $(1 + \epsilon)$-approximation in time $f(\epsilon)s^{O(1)}$ where $s$ is the input size. It is enough to prove that there is an FPT algorithm for the decision version of Closest to $k$ Strings, with parameter $d$. Given an instance of this parameterized problem we need only call the EPTAS with any $\epsilon$ less than $(d + 1)/d$; notice the resulting algorithm takes FPT time with respect to $d$. To analyze this, let $d_{ALG}$ be the distance value of the solution produced by the EPTAS algorithm, and $d_{OPT}$ be the optimal distance value. If $d_{OPT} \leq d$, since $d_{OPT} \leq d_{ALG} \leq (1 + \epsilon)d_{OPT}$ and $d_{OPT}, d_{ALG} \in \mathbb{Z}$, we have $d_{OPT} = d_{ALG} \leq d$. Otherwise, $d_{ALG} \geq d_{OPT} > d$. So, we get an FPT algorithm just by comparing $d_{ALG}$ to $d$.

Observation 1. Closest to $k$ Strings has no EPTAS unless W[1] = FPT.

4 APX-Hardness of Most Strings with Few Bad Columns

In this section, we prove that the Most Strings with Few Bad Columns Problem is APX-hard, even in binary alphabets. To do this we reduce from the Densest-k-Subgraph Problem: given a graph $G = (V, E)$ and a parameter $k$, find a subset $U \subseteq V$ with $|U| = k$ such that $|E[U]|$ is maximized — here $E[U]$ denotes the induced edges for $U$, meaning the set of all edges with both endpoints in $U$. 
Figure 2: Example of the reduction from an instance of Densest-$k$-Subgraph with $G$ and $k = 3$ to an instance of Most Strings with Few Bad Columns with 6 strings of length 5.

Our reduction will be approximation-preserving up to an additive $+1$ term. Given an instance $(G = (V, E), k)$ of Densest-$k$-Subgraph, we will generate an instance of Most Strings with Few Bad Columns with $|E| + 1$ strings, each of length $|V|$, and with the same values for the two parameters $k$ (size of subgraph, maximum number of bad columns).

Let us define the set $S$ of strings generated by the reduction; to do this, index $V = \{v_1, v_2, \ldots\}$.

For each edge $e = v_i v_j \in E$, let that edge’s 0-1 incidence vector $\chi(e)$ be the 0-1 string with 1s in positions $i$ and $j$ and 0 elsewhere; we put $\chi(e)$ into $S$. Finally, we put one more string into $S$, namely the all-zero string 0. This completes the description of the reduction; note it only takes polynomial time. See Figure 2 for an illustration of this reduction.

Claim 2. Let $\alpha$ be the optimal value for the Densest-$k$-Subgraph instance. Then the optimal value $\beta$ for the new Most Strings with Few Bad Columns instance is $\beta = \alpha + 1$.

Proof. First we show the easy direction, that $\beta \geq \alpha + 1$. Consider the optimal $U$ for Densest-$k$-Subgraph, so that $|E[U]| = \alpha$ and $|U| = k$. Define a subset $T$ of $S$ by $T = \{0\} \cup \{\chi(e) \mid e \in F\}$. Then the strings in $T$ are all zero on any index corresponding to a node outside of $V$; the only bad columns are those corresponding to nodes in $V$, of which there are only $k$. So $\beta \geq |T| = \alpha + 1$.

For the reverse direction, take a subset $T$ of $\beta$ strings that have at most $k$ bad columns. We can assume without loss of generality that the string 0 is in $T$, as the following structural lemma shows.

Lemma 2. Let $T \subseteq S$ be a subset of strings with at most $k$ bad columns. Then there is a subset $T'$ of $S$ with at most $k$ bad columns, $|T'| \geq |T|$, and 0 $\in T'$.

Assume for the moment that the lemma is true. Then we simply reverse the above reduction to show $\alpha \geq \beta - 1$. Take an optimal set $S^*$ of strings with $|S^*| = \beta$ and such that $S^*$ has at most $k$ bad columns. By Lemma 2 we may assume 0 $\in S^*$ — this implies that the set $J$ of all non-bad columns for $S^*$ satisfies $s(j) = 0$ for all $s \in S^*, j \in J$. Thus, each $\chi(uv) \in S^* \setminus \{0\}$ has both of its 1s appearing at positions in $[\ell] \setminus J$, or equivalently each such $uv$ is an element of $E[V \setminus J]$. So $V \setminus J$ is the required solution for Densest-$k$-Subgraph, and it has at least $\beta - 1$ induced edges.

Proof of Lemma 2. Assume that 0 $\not\in T$, otherwise the lemma trivially follows. Also, assume $W = T \cup \{0\}$ has more than $k$ bad columns, otherwise we can take $T' = W$. Thus there must be a column
that is not bad for $T$ but that becomes bad when adding $0$. I.e. $T$ has a column that is entirely 1s. It follows that, viewed in the original graph setting, there exists a vertex $v$ that is an end-point of all the edges corresponding to $T$. Pick any such edge arbitrarily, i.e. suppose $s = \chi(vw) \in T$. Since the input graph is simple, in column $w$, all entries of $T$ are 0 except for $\chi(vw)$. Hence, $T' = T \setminus s \cup 0$ satisfies the lemma: compared with $T$ it is bad in column $v$ but not bad in column $w$. 

This ends the proof of Claim 2.

This reduction yields our result:

**Theorem 2.** The Most Strings with Few Bad Columns Problem is NP-hard, and APX-hard.

**Proof.** Khot [13] showed that the Densest-$k$-Subgraph Problem is APX-hard. We need only to argue that our reduction can transform a PTAS for Most Strings with Few Bad Columns into a PTAS for Densest-$k$-Subgraph. Indeed, if we had a $(1 + \delta)$-approximation algorithm for Most Strings with Few Bad Columns, then we get an algorithm for Densest-$k$-Subgraph that always returns a solution of value at least

$$(OPT + 1)/(1 + \delta) - 1 = (OPT - \delta)/(1 + \delta) \geq OPT(1 - \delta)/(1 + \delta) = OPT/(1 + O(\delta))$$

where we used $OPT \geq 1$ in the middle inequality.

While we ruled out a PTAS, it would also be out of the reach of current technology to obtain a constant or polylogarithmic factor for Most Strings with Few Bad Columns, because the best known approximation factor for the Densest-$k$-Subgraph Problem is $O(|V|^{1/4+\epsilon})$ [4].

## 5 Conclusions and Open Problems

Our results demonstrate that while outliers help model the problems associated with using biological data, such problems are computationally intractable to approximate. Here are the main open problems related to our results:

- Is there a constant-factor approximation for either Close to Most Strings or Most Strings with Few Bad Columns (even over a binary alphabet)?

- Does there exist an EPTAS for the Closest String Problem? Since the Closest String Problem is FPT with respect to $d$ [10], the standard technique used in Section 3 cannot be used naively.

- Does there exist an EPTAS for the Closest to $k$ Strings Problem over a bounded-size or binary alphabet? The reduction used in Section 3 needs an arbitrarily large alphabet.

**Acknowledgments**

The authors would like to thank Dr. Bin Ma for mentioning the error in his inapproximability proof and encouraging us to work on a correction, and Dr. Daniel Lokshtanov and Christine Lo for their insights and comments.
References

[1] A. Amir, H. Paryenty, and L. Roditty. Approximations and partial solutions for the consensus
sequence problem. In Proc. of the 18th SPIRE, pages 168–173, 2011.

[2] A. Andoni, P. Indyk, and M. Patrascu. On the optimality of the dimensionality reduction
method. In Proc. of the 47th FOCS, pages 449–456, 2006.

[3] S. Arora, Polynomial Time Approximation Schemes for Euclidean Travelling Salesman and
other Geometric Problems. J. ACM, 45, 5:753–782, 1998.

[4] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, A. Vijayaraghavan. Detecting high log-
densities: an $O(n^{1/4})$ approximation for densest $k$-subgraph. In Proc. of the 42nd STOC,
pages 201–210, 2010.

[5] C. Boucher and B. Ma. CLOSEST STRING WITH OUTLIERS. BMC Bioinformatics, 12(Suppl
1):S55, 2011.

[6] X. Deng, G. Li, Z. Li, B. Ma, and L. Wang. Genetic design of drugs without side-effects. SIAM
Journal on Computing, 32(4):1073–1090, 2003.

[7] P. Festa. On some optimization problems in molecular biology. Mathematical Biosciences,
207(2):219–234, 2007.

[8] P. Festa and P. Pardalos. Efficient solutions for the far from most string problem. Annals of
Operations Research, published Online First, December 2011.

[9] M. Frances and A. Litman. On covering problems of codes. Theoretical Computer Science,
30(2):113–119, 1997.

[10] J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for CLOSEST
STRING and related problems. Algorithmica, 37(1):25–42, 2003.

[11] J. Gramm, J. Guo, and R. Niedermeier. On Exact and Approximation Algorithms for Distingu-
ishing Substring Selection. In Proc. FST, 195–209, 2003.

[12] J Håstad. Some optimal inapproximability results. Journal of the ACM, 48(4):798–859, 2001.

[13] S. Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique.
SIAM Journal on Computing, 36(4):1025–1071, 2006.

[14] S. Khot and A. K. Ponnuswami. Better Inapproximability Results for MaxClique, Chromatic
Number and Min-3Lin-Deletion. In Proc. of the 33rd ICALP, pages 226–237, 2006.

[15] J.K. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems.
Information and Computation, pages 41–55, 2003. Preliminary version appeared in Proc. 10th
SODA, pages 41-55, 1999.

[16] G.M. Landau, J.P. Schmidt, and D. Sokol. An algorithm for approximate tandem repeats.
Journal of Computational Biology, 8(1): 1–18, 2001.
[17] M. Li, B. Ma, and L. Wang. Finding similar regions in many strings. *Journal of Computer and System Sciences*, 65(1):73–96, 2002.

[18] D. Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1): 60–78, 2008.

[19] C. N. Meneses and C. A. S. Oliveira and P. M. Pardalos. Optimization techniques for string selection and comparison problems in genomics. *IEEE Engineering in Medicine and Biology Magazine*, 24(3):81–87, 2005.

[20] B. Ma. A polynomial time approximation scheme for the closest substring problem. In *Proc. of the 11th CPM*, pages 99–107, 2000.

[21] R. Motwani and P. Raghavan. *Randomized Algorithms*, Cambridge University Press, 2000.

[22] B. Ma and X. Sun. More efficient algorithms for closest string and substring problems. *SIAM Journal on Computing*, 39:1432–1443, 2009.

[23] L. Wang and B. Zhu. Efficient algorithms for the closest string and distinguishing string selection problems. In *Proc. of the 3rd FAW*, pages 261–270, 2009.