Theoretical Computer Science 498 (2013) 107–114

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On approximating string selection problems with outliers ☆

Christina Boucher [a], Gad M. Landau [b,c], Avivit Levy [d,e,*], David Pritchard [f], Oren Weimann [b]

[a] Department of Computer Science, University of California, San Diego, USA
[b] Department of Computer Science, University of Haifa, Haifa 31905, Israel
[c] Polytechnic Institute of NYU, Brooklyn, NY 11201-3840, USA
[d] Shenkar College for Engineering and Design, Ramat-Gan 52526, Israel
[e] CRI, University of Haifa, Mount Carmel, Haifa 31905, Israel
[f] CEMC, University of Waterloo, Canada

ARTICLE INFO

Article history:
Received 9 September 2012
Received in revised form 6 March 2013
Accepted 30 May 2013
Communicated by M. Crochemore

Keywords:
String selection
String algorithms

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 the 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 that this problem has no polynomial-time approximation scheme (PTAS) unless NP has randomized polynomial-time algorithms, correcting a decade-old erroneous proof made previously in the literature. 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 $\mathsf{P} = \mathsf{NP}$. We also observe *Closest to k Strings* has no efficient PTAS (EPTAS) unless a parameterized complexity hierarchy collapses. 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 have 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. For example, the problem of finding transcription factor binding sites in biological data is abstractly viewed as the motif-recognition problem [22,33]. Transcription factors are proteins that bind to promoter regions in the genome and have the effect of regulating the expression of one or more genes. Hence, the region where a transcription factor binds is very well-conserved, and the problem of detecting such regions can be viewed as a stringology problem. In light of this observation, Lanctot et al. [22] 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 [10,16,22].

☆ A previous version of this paper appeared in CPM 2012.

* Corresponding author at: Shenkar College for Engineering and Design, Ramat-Gan 52526, Israel.
E-mail addresses: cboucher@eng.ucsd.edu (C. Boucher), landau@cs.haifa.ac.il (G.M. Landau), avivitlevy@shenkar.ac.il (A. Levy), daveagp@gmail.com (D. Pritchard), oren@cs.haifa.ac.il (O. Weimann).

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2013.05.037

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 – including the one previously discussed. 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.

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 [8]. Here, we prove results about the approximability of three *string selection problems with outliers*. For any two strings x and y of the 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 the three following NP optimization problems. Note that the decision versions of the first two are the same problem (i.e. the Closest String Problem).

Definition 1 *(Close to Most Strings).* (A.k.a. *Max Close String* [22,26].)
Input: n strings $S = \{s_1, \ldots, s_n\}$ of length ℓ over an alphabet Σ, and $d \in \mathbf{Z}_+$.
Solution: a string s of length ℓ.
Objective: maximize the number of strings s_i in S that satisfy $d(s, s_i) \leqslant d$.

Definition 2 *(Closest to k Strings).*
Input: n strings $S = \{s_1, \ldots, s_n\}$ of length ℓ over an alphabet Σ, and $k \in \mathbf{Z}_+$.
Solution: a string s of length ℓ 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 [14] that has received significant interest in parameterized complexity and approximability [3,4,17,22,24,27,34].

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

Definition 3 *(Most Strings with Few Bad Columns).*
Input: n strings $S = \{s_1, \ldots, s_n\}$ of length ℓ over an alphabet Σ, and $k \in \mathbf{Z}_+$.
Solution: a subset $S^* \subseteq S$ of strings such that the number $\{t \in [\ell] \mid \exists s_i^*,\ s_j^* \in S^*\colon\ s_i^*(t) \neq s_j^*(t)\}$ of bad columns is at most k.
Objective: maximize $|S^*|$.

A column t is *bad* when its entries are not-all-equal, among strings in S^*. As an example application, suppose we have a collection of DNA sequences from a heterogeneous population of two sub-groups: (1) a large collection of sequences that are identical except for k positions where mutations can occur, and (2) additional outliers. Then Most Strings with Few Bad Columns models the problem of separating the two groups. This problem also generalizes the problem of finding tandem repeats in a string [23].

1.1. Our contributions

A *polynomial-time approximation scheme* (PTAS) for an optimization 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 ϵ, 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 ϵ times a constant-degree polynomial in the input size. A decision problem lies in ZPP if it has a randomized algorithm that is always correct, and whose expected running time is polynomial. A well-known equivalent characterization of ZPP is, for any fixed $0 < p < 1$, that the algorithm *always* runs in polynomial time, outputs either the correct answer or no answer, and gives the correct answer with probability at least p for every input. We defer a brief description of the parameterized complexity classes W[1] and FPT to Section 1.2.

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 [26] 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. Theorem 2 of Ma [26] claims that the Closest to Most Strings problem is APX-hard

by using a simple reduction from the *Far from Most Strings* problem; however, the reduction is erroneous. To explain, it is helpful to define 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 $d(s, s_i) \geqslant d$ (rather than $\leqslant$). There is considerable experimental interest in heuristics for Far from Most Strings, mostly based on local search [31,12,13]. Far from Most Strings was introduced and studied by Lanctot et al. [22], 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 [21] to be inapproximable within a factor of $n/2^{\log^{3/4+\epsilon} n}$ (where n here is the number of nodes in the input graph), unless $\mathsf{NP} \subset \mathsf{BPTIME}(2^{\log^{O(1)} n})$.

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 ℓ has $d(s_i, s) \leqslant d$ if and only if the 0-1 complement $\bar{s}$ of s satisfies $d(s_i, \bar{s}) \geqslant \ell - d$. The crucial error in Ma's reduction [26] is a mis-citation of Lanctot et al. [22], assuming that their result held for binary alphabets. (One reason why the approach of Lanctot et al. [22] 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 $d = \ell$. Note that alphabet size is known to play a role for proving hardness, e.g. in [30,11].)

From the work of Lanctot et al. [22] and that of Ma [26] 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 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 at most some function that is independent of the instance size. The class W[1] is a superset of FPT closed under FPT reductions. A problem is W[1]-*hard* if any W[1] problem can be FPT-reduced to it, and W[1]-*complete* if it is both in W[1] and W[1]-hard. There are many natural W[1]-complete problems, like Maximum Clique parameterized by clique size. It is widely hypothesized that $\mathsf{FPT} \subsetneq \mathsf{W[1]}$, but unproven, analogous to $\mathsf{P} \subsetneq \mathsf{NP}$.

1.3. Previous work

The Closest String Problem can be viewed as a special case of the Close to Most Strings Problem where the number of outliers is equal to zero. This problem has been throughly studied and, therefore, there exist numerous results concerning its complexity and approximability [4,9,17,22,24–27,34,35]. It was shown NP-complete, even under the restriction that the alphabet is binary [14]. Lanctot et al. [22] gave a polynomial-time algorithm for the Closest String Problem that achieves a $\frac{4}{3} + o(1)$-approximation guarantee. Independently, Gąsieniec et al. [15] gave a $\frac{4}{3}$-approximation algorithm that uses a similar technique, which is based on a linear programming relaxation of an integer programming model of the problem. Using randomized rounding, Li et al. [24] proved the existence of a PTAS for this problem. The running time of the PTAS has since been improved by Andoni et al. [4], and Ma and Sun [27]. Currently, the PTAS with the best known running time is due to Ma and Sun, which runs in $O(n^{\Theta(\epsilon^{-2})})$-time.

Gramm et al. [17] demonstrated that the Closest String Problem is in FPT when parameterized by n, and when parameterized by d. Ma and Sun gave an $O(n|\Sigma|^{O(d)})$-time algorithm, which is a polynomial-time algorithm when $d = O(\log n)$ and Σ has constant size [27]; Chen et al. [9] and Zhao and Zhang [35] have improved upon the running time of this result. Boucher and Ma [8] considered parameterized versions of Closest to k Strings (under the name Closest String with Outliers). They note that restricting only $|\Sigma|$ does not make this problem tractable since it is NP-hard even when the alphabet is binary. However, if $|\Sigma|$ and ℓ are both parameters the problem becomes FPT, as we can enumerate and check all the $|\Sigma|^{\ell}$ possible center strings. As a result the problem is fixed-parameter tractable with the combined parameters $|\Sigma|$, ℓ, d and $n - k$. They also prove that it is imperative that $|\Sigma|$ be a parameter in order to obtain this tractability. See Tables 1 and 2 for a summary of the known results [8] (note that in [8], the notation of the parameters is different, i.e., their $n^* = n - k$, which is denoted as k in this paper and vice versa).

2. Approximation hardness of Close to Most Strings

In this section we prove the following.

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

Table 1
A summary of the parameterized versions of Closest to k Strings problem with a single parameter with respect to a bounded or unbounded alphabet (i.e., $|\Sigma|$ is or isn't a parameter).

Parameters:	d	$n-k$	n	k	ℓ
Bounded alphabet:	open	W[1]-hard	FPT	open	FPT
Unbounded alphabet:	W[1]-hard	W[1]-hard	open	W[1]-hard	W[1]-hard

Table 2
A summary of the parameterized versions of Closest to k Strings problem with a set of parameters with respect to a bounded or unbounded alphabet (i.e., $|\Sigma|$ is or isn't a parameter).

Parameters:	$n-k,k$	d,k	ℓ,d	ℓ,k	$d,n-k$
Bounded alphabet:	FPT	open	FPT	FPT	FPT
Unbounded alphabet:	open	W[1]-hard	W[1]-hard	W[1]-hard	FPT

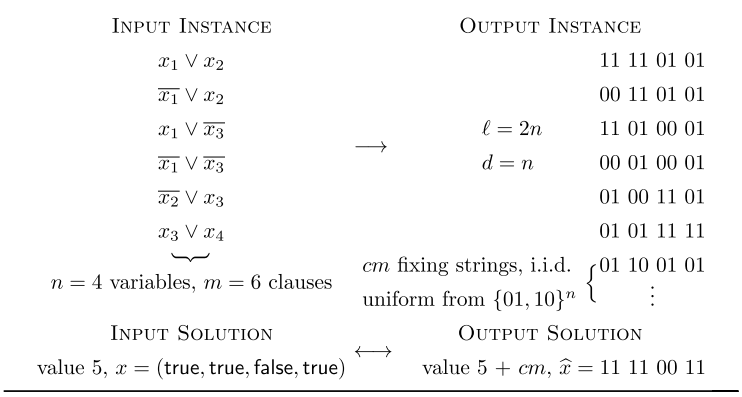

Fig. 1. Overview of the reduction used to prove Theorem 1.

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 [18] 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 $m \geqslant n$, 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 Fig. 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 ω_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 $\{01, 10\}^n$, 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 $\widehat{x}$ via

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

Notice that $\widehat{x}$ is at distance exactly $d = n$ from all of the fixing strings, and that $d(\widehat{x}, s_j) \leqslant n$ if and only if x satisfies clause ω_j. Hence, if x satisfies k clauses, the string $\widehat{x}$ 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 $s = \widehat{x}$ for some x. Then the claim will be complete via standard calculations.

Here is the precise statement of the structural property.

Lemma 1. *Fix $c \geqslant 20$. Let F be a set of cm strings selected uniformly and independently at random from $\{01, 10\}^n$ (with replacement), with $m \geqslant n$. Then with probability at least $1 - 0.9^n$, every string $s \in \{0,1\}^{2n} \setminus \{00,11\}^n$ 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 $f \in F$. Our first claim is the following:

Claim 1. *For any $s \in \{0,1\}^{2n} \setminus \{00,11\}^n$, if f is selected uniformly at random from $\{01,10\}^n$, then $\Pr[d(s,f) \geqslant n+1] \geqslant 1/4$.*

Proof. By hypothesis, for some i this s satisfies $s(2i-1) \neq s(2i)$, say $s(2i-1) = 0$ and $s(2i) = 1$ (the other case is symmetric). Let $\mathcal{E}$ denote the event $[f(2i-1) \neq s(2i-1)$ and $f(2i) \neq s(2i)]$. Since f is chosen uniformly at random from $\{01,10\}^n$, we have $\Pr[\mathcal{E}] = 1/2$.

Next we show that $\Pr[d(s,f) \geqslant n+1 \mid \mathcal{E}] \geqslant 1/2$. 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 $d(s,f) \mid \mathcal{E}$ is thus a shifted and scaled binomial distribution, in particular it is symmetric about $n+1$. So

$$\Pr\big[d(s,f) \geqslant n+1 \;\big|\; \mathcal{E}\big] = \Pr\big[d(s,f) \leqslant n+1 \;\big|\; \mathcal{E}\big]$$

and since these two probabilities' sum is at least 1, $\Pr[d(s,f) \geqslant n+1 \mid \mathcal{E}] \geqslant 1/2$ follows.

Finally, unconditioning,

$$\Pr\big[d(s,f) \geqslant n+1\big] = \Pr\big[d(s,f) \geqslant n+1 \;\big|\; \mathcal{E}\big] \cdot \Pr[\mathcal{E}] \geqslant 1/2 \cdot 1/2 = 1/4. \qquad \square$$

Continuing with the proof of Lemma 1, we next use a Chernoff bound to reason about how a single s interacts with the entire collection F, and then will use a union bound to cover 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, $E[\sum_i X_i] \geqslant cm/4$. We will use a Chernoff bound of the following form:

Claim 2 *(Lower Chernoff bound). For any $\delta > 0$, if X is a sum of independent random variables that each only take on the values 0 and 1, then*

$$\Pr\big[X < (1-\delta)E[X]\big] < \exp\big(-E[X]\delta^2/2\big).$$

For a proof of this standard result, see for example, Theorem 4.2 in Motwani and Raghavan's book [32], p. 70. We will apply it to $X = \sum_i X_i$ and to δ chosen so that $(1-\delta)cm/4 = m$, i.e. $\delta = 1 - 4/c$. Then the Chernoff bound implies:

$$\begin{aligned}\Pr[X < m] &\leqslant \Pr\big[X < (1-\delta)E[X]\big] < \exp\big(-E[X]\delta^2/2\big) \\ &\leqslant \exp\big(-cm/4 \cdot (1-4/c)^2/2\big) = \exp\left(\frac{-(c-4)^2}{8c} m\right).\end{aligned}$$

This shows that every s is very unlikely to falsify Lemma 1. We may now take 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

$$\big(4^n - 2^n\big) \exp\left(\frac{-(c-4)^2}{8c} m\right) < 4^n \exp\left(\frac{-(c-4)^2}{8c} n\right) = \exp\left(\left(\ln 4 - \frac{(c-4)^2}{8c}\right) n\right),$$

where we used $m \geqslant n$ in the first inequality. Any large enough c makes this probability exponentially decreasing in n; it is straightforward to calculate that when $c = 20$ this is at most 0.9^n, as needed. $\square$

Now that the proof of Lemma 1 is complete, we proceed with the proof of Theorem 1. 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 $(1+\epsilon)$-approximation for this instance, where ϵ will be a small constant fixed later to satisfy two properties.

Let k^* be the maximum number 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 $s \notin \{00,11\}^n$ (i.e. not of the form $s = \widehat{x}$) is within distance d of at most cm of the $(c+1)m$ strings. Thus provided that ϵ is small enough to satisfy $1 + \epsilon < \frac{cm + m/2}{cm} = 1 + \frac{1}{2c}$, then s_A is of the form $\widehat{x}_A$ 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 ϵ so that

$$\frac{cm + k^*}{1+\epsilon} > cm + \frac{21}{22} k^* \tag{1}$$

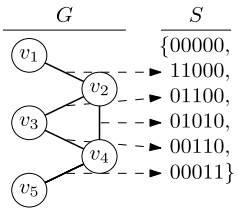

Fig. 2. Example of the reduction from an instance of Densest-k-Subgraph with G and $k = 3$ to an instance of Most Strings with Few Bad Columns with 6 strings of length 5.

then $\widehat{x}_A$ satisfies more than $\frac{21}{22}k^*$ clauses, which is NP-hard by Håstad's result. Using that $k^* \geqslant m/2$, 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. When the output s_A of the Close to Most Strings approximation algorithm satisfies $s_A \notin \{00, 11\}^n$ we output nothing. When $s_A \in \{00, 11\}^n$ we know for certain that $\widehat{x}_A$ is a 22/21-approximate solution for Max-2-SAT, as needed. □

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

Ma showed in [26] that the Closest to k Strings problem has a PTAS. 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 an FPTAS (running time polynomial in the input length and ϵ^{-1}) [5,19,28]. We observe that 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 [29], along with the fact that the decision version of Closest to k Strings is W[1]-hard when parameterized by d [8].

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 ϵ less than $(d + 1)/d$. Notice that 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 $d_{OPT} \leqslant d$, since $d_{OPT} \leqslant d_{ALG} \leqslant (1 + \epsilon)d_{OPT}$ and $d_{OPT}, d_{ALG} \in \mathbf{Z}$, we have $d_{OPT} = d_{ALG} \leqslant d$. Otherwise, $d_{ALG} \geqslant d_{OPT} > d$. So, we get an FPT algorithm just by comparing d_{ALG} to d.

Hence, we obtain the following observation.

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

This implies that the PTAS given by Ma [26] cannot be improved to an EPTAS.

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 end-points in U.

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). We assume that $k > 2$ since $k \leqslant 2$ produces trivial cases.

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 $\mathbf{0}$. This completes the description of the reduction; note it only takes polynomial time. See Fig. 2 for an illustration of this reduction. We will use the following lemma in the proof of the subsequent Claim 3.

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, $|T'| \geqslant |T|$, and $\mathbf{0} \in T'$.*

Proof. Assume that $\mathbf{0} \notin T$, otherwise the lemma trivially follows. Also, assume $W = T \cup \{\mathbf{0}\}$ 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 $\mathbf{0}$, 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, $T' = T \setminus s \cup \{\mathbf{0}\}$ satisfies the lemma: compared with T it is bad in column v but not bad in column w. □

Claim 3. *Let α be the optimal value for the Densest-k-Subgraph instance. Then the optimal value β for the new Most Strings with Few Bad Columns instance is $\beta = \alpha + 1$.*

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

For the reverse direction, take a subset T of β strings that have at most k bad columns. We can assume without loss of generality that the string $\mathbf{0}$ is in T, as shown by Lemma 2. Using Lemma 2, we simply reverse the above reduction to show $\alpha \geqslant \beta - 1$. 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 $\mathbf{0} \in S^*$ — 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 $\chi(uv) \in S^* \setminus \{\mathbf{0}\}$ 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. This ends the proof of Claim 3. □

This reduction yields our result:

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

Proof. Khot [20] 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

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

where we used $OPT \geqslant 1$ 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 $O(|V|^{1/4+\epsilon})$ [6].

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 some open problems related to our results:

- The instances produced by the proof of Theorem 1 have $d = \ell/2$. Are the instances with $d < \ell/2$ still hard?
- Is there a constant-factor approximation for Close to Most Strings (even over a binary alphabet)?
- Is there a constant-factor approximation for Most Strings with Few Bad Columns (even over a binary alphabet)?
- Does Closest to k Strings have an EPTAS when the alphabet is binary? The reduction used in Section 3 needs an arbitrarily large alphabet. A more important problem is, does there exist an EPTAS for the Closest String Problem? Since the Closest String problem is FPT with respect to d [17], the standard technique used in Section 3 cannot be used naively. The method presented by Boucher et al. [7] for proving the non-existence of an EPTAS may be applicable in this context.
- The approximability of other string selection problems with outliers warrants investigation. The *Sum Closest String with Outliers* problem takes a set S of the same-length strings as input, and seeks a maximum-size $S^* \subseteq S$ subject to a given upper bound on $\sum_{s_i \in S^*} d(s, s_i)$. This problem is closely related to the cycle detection, correction and approximate periodicity problems of [1,2]. What is the approximability of this problem?

Acknowledgements

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. We thank Dr. Daniel Lokshtanov, Christine Lo, and the referees for their insights and comments. This work was supported by Natural Sciences and Engineering Research Council of Canada Post Doctoral Fellowship program, the Gerald Schwartz and Heather Reisman Foundation, the Israel Science Foundation grant (347/09), the National Science Foundation Award (0904246), and Grant Number 2008217 from the United States–Israel Binational Science Foundation (BSF) and DFG.

References

[1] A. Amir, E. Eisenberg, A. Levy, Approximate periodicity, in: Proc. of the 21st ISAAC, 2010, pp. 25–36.
[2] A. Amir, E. Eisenberg, A. Levy, E. Porat, N. Shapira, Cycle detection and correction, in: Proc. of 37th ICALP, 2010, pp. 43–54.
[3] A. Amir, H. Paryenty, L. Roditty, Approximations and partial solutions for the consensus sequence problem, in: Proc. of the 18th SPIRE, 2011, pp. 168–173.
[4] A. Andoni, P. Indyk, M. Patrascu, On the optimality of the dimensionality reduction method, in: Proc. of the 47th FOCS, 2006, pp. 449–456.
[5] S. Arora, Polynomial time approximation schemes for Euclidean travelling salesman and other geometric problems, J. ACM 45 (5) (1998) 753–782.
[6] 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, 2010, pp. 201–210.
[7] C. Boucher, C. Lo, D. Lokshantov, Outlier detection for DNA fragment assembly, arXiv:1111.0376.
[8] C. Boucher, B. Ma, Closest string with outliers, BMC Bioinformatics 12 (Suppl. 1) (2011) S55.
[9] Z.-Z. Chen, B. Ma, L. Wang, A three-string approach to the closest string problem, J. Comput. System Sci. 78 (1) (2012) 164–178.
[10] X. Deng, G. Li, Z. Li, B. Ma, L. Wang, Genetic design of drugs without side-effects, SIAM J. Comput. 32 (4) (2003) 1073–1090.
[11] M.R. Fellows, J. Gramm, R. Niedermeier, On the parameterized intractability of motif search problems, Combinatorica 26 (2) (2006) 141–167.
[12] P. Festa, On some optimization problems in molecular biology, Math. Biosci. 207 (2) (2007) 219–234.
[13] P. Festa, P. Pardalos, Efficient solutions for the far from most string problem, Ann. Oper. Res. 196 (1) (2012) 663–682.
[14] M. Frances, A. Litman, On covering problems of codes, Theory Comput. Syst. 30 (2) (1997) 113–119.
[15] L. Gąsieniec, J. Jansson, A. Lingas, Efficient approximation algorithms for the Hamming center problem, in: Proc. of the 10th SODA, 1999, pp. 905–906.
[16] J. Gramm, J. Guo, R. Niedermeier, On exact and approximation algorithms for distinguishing substring selection, Theory Comput. Syst. 39 (4) (2006) 545–560.
[17] J. Gramm, R. Niedermeier, P. Rossmanith, Fixed-parameter algorithms for CLOSEST STRING and related problems, Algorithmica 37 (1) (2003) 25–42.
[18] J. Håstad, Some optimal inapproximability results, J. ACM 48 (4) (2001) 798–859.
[19] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, R.E. Stearns, NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs, J. Algorithms 26 (2) (1998) 238–274.
[20] S. Khot, Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique, SIAM J. Comput. 36 (4) (2006) 1025–1071.
[21] S. Khot, A.K. Ponnuswami, Better inapproximability results for MaxClique, Chromatic Number and Min-3Lin-Deletion, in: Proc. of the 33rd ICALP, 2006, pp. 226–237.
[22] J.K. Lanctot, M. Li, B. Ma, S. Wang, L. Zhang, Distinguishing string selection problems, Inform. and Comput. 185 (2003) 41–55.
[23] G.M. Landau, J.P. Schmidt, D. Sokol, An algorithm for approximate tandem repeats, J. Comput. Biol. 8 (1) (2001) 1–18.
[24] M. Li, B. Ma, L. Wang, Finding similar regions in many strings, J. Comput. System Sci. 65 (1) (2002) 73–96.
[25] D. Lokshtanov, D. Marx, S. Saurabh, Slightly superexponential parameterized problems, in: Proc. of the 16th SODA, 2011, pp. 760–776.
[26] B. Ma, A polynomial time approximation scheme for the closest substring problem, in: Proc. of the 11th CPM, 2000, pp. 99–107.
[27] B. Ma, X. Sun, More efficient algorithms for closest string and substring problems, SIAM J. Comput. 39 (2009) 1432–1443.
[28] D. Marx, Efficient approximation schemes for geometric problems?, in: Proc. of 13th ESA, vol. 51 (1), 2005, pp. 448–459.
[29] D. Marx, Parameterized complexity and approximation algorithms, Comput. J. 51 (1) (2008) 60–78.
[30] D. Marx, Closest substring problems with small distances, SIAM J. Comput. 38 (4) (2008) 1382–1410.
[31] C.N. Meneses, C.A.S. Oliveira, P.M. Pardalos, Optimization techniques for string selection and comparison problems in genomics, IEEE Eng. Med. Biol. Mag. 24 (3) (2005) 81–87.
[32] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 2000.
[33] P. Pevzner, S. Sze, Combinatorial approaches to finding subtle signals in DNA strings, in: Proc. of the 8th ISMB, 2000, pp. 269–278.
[34] L. Wang, B. Zhu, Efficient algorithms for the closest string and distinguishing string selection problems, in: Proc. of the 3rd FAW, 2009, pp. 261–270.
[35] R. Zhao, N. Zhang, A more efficient closest string algorithm, in: Proc. of the 2nd BICoB, 2010, pp. 210–215.