Lower Bounds for Approximation Schemes for Closest String*

Marek Cygan[1], Daniel Lokshtanov[2], Marcin Pilipczuk[3], Michał Pilipczuk[4], and Saket Saurabh[5]

1 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
cygan@mimuw.edu.pl
2 **Department of Informatics, University of Bergen, Bergen, Norway**
daniello@ii.uib.no
3 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
malcin@mimuw.edu.pl
4 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
michal.pilipczuk@mimuw.edu.pl
5 **Department of Informatics, University of Bergen, Bergen, Norway; and Institute of Mathematical Sciences, Chennai, India**
saket.saurabh@ii.uib.no, saket@imsc.res.in

Abstract

In the Closest String problem one is given a family $\mathcal{S}$ of equal-length strings over some fixed alphabet, and the task is to find a string y that minimizes the maximum Hamming distance between y and a string from $\mathcal{S}$. While polynomial-time approximation schemes (PTASes) for this problem are known for a long time [Li et al.; J. ACM'02], no *efficient* polynomial-time approximation scheme (EPTAS) has been proposed so far. In this paper, we prove that the existence of an EPTAS for Closest String is in fact unlikely, as it would imply that FPT = W[1], a highly unexpected collapse in the hierarchy of parameterized complexity classes. Our proof also shows that the existence of a PTAS for Closest String with running time $f(\varepsilon)\cdot n^{o(1/\varepsilon)}$, for any computable function f, would contradict the Exponential Time Hypothesis.

1998 ACM Subject Classification F.2.2 Nonnumerical algorithms and problems

Keywords and phrases closest string, PTAS, efficient PTAS

Digital Object Identifier 10.4230/LIPIcs.SWAT.2016.12

1 Introduction

Closest String and Closest Substring are two computational problems motivated by questions in molecular biology connected to identifying functionally similar regions of DNA or RNA sequences, as well as by applications in coding theory. In Closest String we are given a family $\mathcal{S}$ of strings over some fixed alphabet Σ, each of length L. The task is to find one string $y \in \Sigma^L$ for which $\max_{x\in\mathcal{S}} \mathcal{H}(x, y)$ is minimum possible, where $\mathcal{H}(x, y)$ is

* M. Cygan and Ma. Pilipczuk have been supported by Polish National Science Centre grant DEC-2012/05/D/ST6/03214. Mi. Pilipczuk has been supported by Polish National Science Centre grant DEC-2013/11/D/ST6/03073 and by the Foundation for Polish Science via the START stipend programme. During the work on these results, Mi. Pilipczuk held a post-doc position at Warsaw Center of Mathematics and Computer Science. D. Lokshtanov is supported by the BeHard grant under the recruitment programme of the of Bergen Research Foundation. S. Saurabh is supported by PARAPPROX, ERC starting grant no. 306992.


© Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh;
licensed under Creative Commons License CC-BY

the *Hamming distance* between x and y, that is, the number of positions on which x and y have different letters. We will consider both the optimization variant of the problem where the said distance is to be minimized, and the decision variant where an upper bound d is given on the input, and the algorithm needs to decide whether there exists a string y with $\max_{x\in\mathcal{S}} \mathcal{H}(x,y) \leq d$. Closest Substring is a more general problem where the strings from the input family $\mathcal{S}$ all have length $m \geq L$, and we look for a string $y \in \Sigma^L$ that minimizes $\max_{x\in\mathcal{S}} \min_{x' \text{ substring of } x} \mathcal{H}(x',y)$. In other words, we look for y that can be fit as close as possible to a substring of length L of each of the input strings from $\mathcal{S}$.

Both Closest String and Closest Substring, as well as numerous variations on these problems, have been studied extensively from the point of view of approximation algorithms. Most importantly for us, for both of these problems there are classic results providing *polynomial-time approximation schemes* (*PTASes*): for every $\varepsilon > 0$, it is possible to approximate in polynomial time the optimum distance within a multiplicative factor of $(1+\varepsilon)$. The first PTASes for these problems were given by Li et al. [9], and they had running time bounded by $n^{\mathcal{O}(1/\varepsilon^4)}$. This was later improved by Andoni et al. [1] to $n^{\mathcal{O}(\frac{\log 1/\varepsilon}{\varepsilon^2})}$, and then by Ma and Sun [11] to $n^{\mathcal{O}(1/\varepsilon^2)}$, which constitutes the current frontier of knowledge. We refer to the works [3, 8, 7, 9, 11, 12] for a broad introduction to biological applications of Closest String, Closest Substring, and related problems, as well as pointers to relevant literature.

One of the immediate questions stemming from the works of Li et al. [9], Andoni et al. [1], and Ma and Sun [11], is whether either for Closest String or Closest Substring one can also give an *efficient polynomial-time approximation scheme* (*EPTAS*), i.e., an approximation scheme that for every $\varepsilon > 0$ gives a $(1+\varepsilon)$-approximation algorithm with running time $f(\varepsilon)\cdot n^{\mathcal{O}(1)}$, for some computable function f. In other words, the degree of the polynomial should be independent of ε, whereas the exponential blow-up (inevitable due to NP-completeness) should happen only in the multiplicative constant standing in front of the running time. EPTASes are desirable from the point of view of applications, since they provide approximation algorithms that can be useful in practice already for relatively small values of ε, whereas running times of general PTASes are usually prohibitive.

For the more general Closest Substring problem, this question was answered negatively by Marx [12] using the techniques from parameterized complexity. More precisely, Marx considered various parameterizations of Closest Substring, and showed that when parameterized by d and $|\mathcal{S}|$, the problem remains W[1]-hard even for the binary alphabet. This means that the existence of a fixed-parameter algorithm with running time $f(d,|\mathcal{S}|)\cdot n^{\mathcal{O}(1)}$, where n is the total size of the input, would imply that FPT = W[1], a highly unexpected collapse in the parameterized complexity. This result shows that, under FPT $\neq$ W[1], also an EPTAS for Closest Substring can be excluded. Indeed, if such an EPTAS existed, then by setting any $\varepsilon < \frac{1}{d}$ one could in time $f(d)\cdot n^{\mathcal{O}(1)}$ distinguish instances with optimum distance value d from the ones with optimum distance value $d+1$, thus solving the decision variant in fixed-parameter tractable (FPT) time. Using more precise results about the parameterized hardness of the Clique problem, Marx [12] showed that, under the assumption of *Exponential Time Hypothesis* (*ETH*), which states that 3-SAT cannot be solved in time $\mathcal{O}(2^{\delta n})$ for some $\delta > 0$, one even cannot expect PTASes for Closest Substring with running time $f(\varepsilon)\cdot n^{o(\log(1/\varepsilon))}$ for any computable function f. We refer to a survey of Marx [13] for more examples of links between parameterized complexity and the design of approximation schemes.

The methodology used by Marx [12], which is the classic connection between parameterized complexity and EPTASes that dates back to the work of Bazgan [2] and of Cesati and

Trevisan [4], completely breaks down when applied to Closest String. This is because this problem actually does admit an FPT algorithm when parameterized by d. An algorithm with running time $d^d \cdot n^{\mathcal{O}(1)}$ was proposed by Gramm et al. [7]. Later, Ma and Sun [11] gave an algorithm with running time $2^{\mathcal{O}(d)} \cdot |\Sigma|^d \cdot n^{\mathcal{O}(1)}$, which is more efficient for constant-size alphabets. Both the algorithms of Gramm et al. and of Ma and Sun are known to be essentially optimal under ETH [10], and nowadays they constitute textbook examples of advanced branching techniques in parameterized complexity [5]. Therefore, in order to settle the question about the existence of an EPTAS for Closest String, one should look for a substantial refinement of the currently known techniques.

An approach for overcoming this issue was recently used by Boucher et al. [3], who attribute the original idea to Marx [13]. Boucher et al. considered a problem called Consensus Patterns, which is a variation of Closest Substring where the goal function is the total sum of Hamming distances between the center string and best-fitting substrings of the input strings, instead of the maximum among these distances. The problem admits a PTAS due to Li et al. [8], and was shown by Marx [12] to be fixed-parameter tractable when parameterized by the target distance d. Despite the latter result, Boucher et al. [3] managed to prove that the existence of an EPTAS for Consensus Patterns would imply that FPT = W[1]. The main idea is to provide a reduction from a W[1]-hard problem, such as Clique, where the output target distance d is not bounded by a function of the input parameter k (indeed, the existence of such a reduction would prove that FPT = W[1]), but the multiplicative gap between the optimum distances yielded for yes- and no-instances is $1 + \frac{1}{g(k)}$, for some computable function g. Even though the output parameter is unbounded in terms of k, an EPTAS for the problem could be still used to distinguish between output instances obtained from yes- and no-instances of Clique in FPT time, thus proving that FPT = W[1].

Our contribution

In this paper we provide a negative answer to the question about the existence of an EPTAS for Closest String by proving the following theorem.

▶ **Theorem 1.1.** *The following assertions hold:*

- *Unless* FPT = W[1]*, there is no EPTAS for* Closest String *over binary alphabet.*
- *Unless ETH fails, there is no PTAS for* Closest String *over binary alphabet with running time* $f(\varepsilon) \cdot n^{o(1/\varepsilon)}$*, for any computable function* f*.*

Thus, one should not expect an EPTAS for Closest String, whereas for PTASes there is still a room for improvement between the running time of $n^{\mathcal{O}(1/\varepsilon^2)}$ given by Ma and Sun [11] and the lower bound of Theorem 1.1. It is worth noting that our $f(\varepsilon) \cdot n^{o(1/\varepsilon)}$ time lower bound for $(1+\varepsilon)$-approximating Closest String also holds for the more general Closest Substring problem. This yields a significantly stronger lower bound than the previous $f(\varepsilon) \cdot n^{o(\log(1/\varepsilon))}$ lower bound of Marx [12].

Our proof of Theorem 1.1 follows the methodology proposed Marx [13] and used by Boucher et al. [3] for Consensus Patterns. The following theorem, which is the main technical contribution of this work, states formally the properties of our reduction.

▶ **Theorem 1.2.** *There is an integer* c *and an algorithm that, given an instance* (G, k) *of* Clique*, works in time* $2^k \cdot n^{\mathcal{O}(1)}$ *and outputs an instance* $(\mathcal{S}, L, d)$ *of* Closest String *over alphabet* $\{0,1\}$ *with the following properties:*

- *If* G *contains a clique on* k *vertices, then there is a string* $w \in \{0,1\}^L$ *such that* $\mathcal{H}(w, x) \leq d$ *for each* $x \in \mathcal{S}$*.*

- *If G does not contain a clique on k vertices, then for each string $w \in \{0,1\}^L$ there is $x \in \mathcal{S}$ such that $\mathcal{H}(w,x) > (1+\frac{1}{ck}) \cdot d$.*

The statement of Theorem 1.2 is similar to the core of the hardness proof of Boucher et al. [3]. However, our reduction is completely different from the reduction of Boucher et al., because the causes of the computational hardness of Closest String and Consensus Patterns are quite orthogonal to each other. In Consensus Patterns the difficulty lies in *picking* the right substrings of the input strings. Once these substrings are known the center string is easily computed in polynomial time, since we are minimizing the sum of the Hamming distances. In Closest String there are no substrings to pick, we just have to find a center string for the given input strings. This is a computationally hard task because we are minimizing the maximum of the Hamming distances to the center, rather than the sum.

Theorem 1.1 follows immediately by combining Theorem 1.2 with the known parameterized hardness results for Clique, gathered in the following theorem, and setting $\varepsilon = \frac{1}{ck}$.

▶ **Theorem 1.3** (cf. Theorem 13.25 and Corollary 14.23 of [5]). *The following assertions hold:*

- *Unless* FPT = W[1], Clique *cannot be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for any computable function f.*
- *Unless ETH fails,* Clique *cannot be solved in time $f(k) \cdot n^{o(k)}$ for any computable function f.*

The main idea of the proof of Theorem 1.2 is to encode the n vertices of the given graph G as an "almost orthogonal" family $\mathcal{T}$ of strings from $\{0,1\}^\ell$, for some $\ell = \mathcal{O}(\log n)$. Strings from $\mathcal{T}$ are used as identifiers of vertices of G, and the fact that they are almost orthogonal means that the identifiers of two distinct vertices of G differ on approximately $\ell/2$ positions. On the other hand $\ell = \mathcal{O}(\log n)$, so the whole space of strings into which $V(G)$ is embedded has size polynomial in n. Using these properties, the reduction promised in Theorem 1.2 is designed by a careful construction.

Notation

By $\log p$ we denote the base-2 logarithm of p. For a positive integer n, we denote $[n] = \{1,2,\ldots,n\}$. The length of a string x is denoted by $|x|$. For an alphabet Σ and two equal-length strings x, y over Σ, the *Hamming distance* between x and y, denoted $\mathcal{H}(x,y)$, is the number of positions on which x and y have different letters. If $\Sigma = \{0,1\}$ is the binary alphabet, then the *Hamming weight* of a string x over Σ, denoted $\mathcal{H}(x)$, is the number of 1s in it. The *complement* of a string x over a binary alphabet, denoted $\overline{x}$, is obtained from x by replacing all 0s with 1s and vice versa. Note that if $|x| = |y| = n$, then $\mathcal{H}(x,y) = n - \mathcal{H}(\overline{x},y) = n - \mathcal{H}(x,\overline{y}) = \mathcal{H}(\overline{x},\overline{y})$.

2 Selection gadget

For the rest of this paper, we fix the following constants: $\rho = 1/100$, $\alpha = 1/10$, $\beta = 1/20$. Instead of giving a set of constraints for ρ, α, and β which are satisfied by a range or assignments we decided to use this particular numerical examples to make the proof easier to follow.

In the proof we will set $\ell = C \cdot \lceil \log n \rceil$ for some large integer C divisible by 100; this will ensure that $\rho\ell$, $\alpha\ell$, and $\beta\ell$ are all integers. First, we prove that among binary strings of logarithmic length one can find a linearly-sized family of "almost orthogonal" strings of

balanced Hamming weight, which will be later used in the reduction to represent vertices of a Clique instance. The proof is by a simple greedy argument.

▶ **Lemma 2.1.** *There exist positive integers C and N, where C is divisible by 100, with the following property. Let $n > N$ be any integer, and let us denote $\ell = C \cdot \lceil \log n \rceil$. Then there exists a set $\mathcal{T} \subseteq \{0,1\}^\ell$ with the following properties:*
1. $|\mathcal{T}| = n$,
2. $\mathcal{H}(x) = \ell/2$ *for each* $x \in \mathcal{T}$, *and*
3. $(1/2 - \rho)\ell < \mathcal{H}(x,y) < (1/2+\rho)\ell$ *for each distinct* $x, y \in \mathcal{T}$.

Moreover, given n, $\mathcal{T}$ can be constructed in time polynomial in n.

Proof. Let $H_2(\cdot)$ denote the binary entropy, i.e., $H_2(p) = -p\log p - (1-p)\log(1-p)$ for $p \in (0,1)$. Since ℓ is some positive integer divisible by 100. Then it is well known that

$$\sum_{i=0}^{k} \binom{\ell}{i} \leq 2^{\ell \cdot H_2(k/\ell)} \tag{1}$$

for all integers k with $0 < k \leq \ell/2$; cf. [6, Lemma 16.19]. Let us denote

$$A = \sum_{i=0}^{(1/2-\rho)\ell} \binom{\ell}{i} + \sum_{i=(1/2+\rho)\ell}^{\ell} \binom{\ell}{i}.$$

Then from (1) it follows that

$$A \leq 2 \cdot 2^{\sigma\ell},$$

where $\sigma = H_2(1/2-\rho) < 1$.

Suppose now that $\ell = C \cdot \lceil \log n \rceil$ for some positive integers C and $n > 1$, where C is divisible by 100. Then

$$\begin{aligned} n(\ell+1)\cdot A &\leq 2n \cdot (C\lceil \log n \rceil + 1) \cdot 2^{\sigma \cdot C \lceil \log n \rceil} \\ &\leq 2 \cdot (2C+1) \cdot 2^{\sigma C} \cdot n \log n \cdot 2^{\sigma \cdot C \log n} \\ &\leq (4C+2) \cdot 2^{\sigma C} \cdot n^{\sigma C + 2}. \end{aligned}$$

Since $\sigma < 1$, we can choose C to be an integer divisible by 100 so that $\sigma C + 2 < C$. Then, we can choose N large enough so that

$$(4C+2) \cdot 2^{\sigma C} \cdot n^{\sigma C + 2} \leq n^C$$

for all integers $n > N$. Hence,

$$nA \leq \frac{n^C}{\ell + 1}. \tag{2}$$

We now verify that this choice of C, N satisfies the required properties.

Consider the following greedy procedure performed on $\{0,1\}^\ell$. Start with $\mathcal{T} = \emptyset$ and all strings of $\{0,1\}^\ell$ marked as unused. In consecutive rounds perform the following:
1. Pick any $x \in \{0,1\}^\ell$ with $\mathcal{H}(x) = \ell/2$ that was not yet marked as used, and add x to $\mathcal{T}$.
2. Mark every $y \in \{0,1\}^\ell$ with $\mathcal{H}(x,y) \leq (1/2-\rho)\ell$ or $\mathcal{H}(x,y) \geq (1/2+\rho)\ell$ as used.

It is clear that at each step of the procedure, the constructed family $\mathcal{T}$ satisfies properties (2) and (3) from the lemma statement. Hence, it suffices to prove that the procedure can be performed for at least n rounds.

Note that the number of strings marked as used at each round is at most A. On the other hand, if $\mathcal{D}$ is the set of strings from $\{0,1\}^\ell$ that have Hamming weight exactly $\ell/2$, then

$$|\mathcal{D}| \geq \frac{|\{0,1\}^\ell|}{\ell+1} = \frac{2^{C\lceil \log n\rceil}}{\ell+1} \geq \frac{n^C}{\ell+1}.$$

From (2) we infer that $|\mathcal{D}| \geq nA$. This means that the algorithm will be able to find an unmarked $x \in \mathcal{D}$ for at least n rounds, and hence to construct the family $\mathcal{T}$ with $|\mathcal{T}| = n$. It is easy to implement the algorithm in polynomial time using the fact that the size of $\{0,1\}^\ell$ is polynomial in n. ◀

From now on, we adopt the constants C, N given by Lemma 2.1 to the notation. Let us also fix $n > N$; then let $\ell = C \cdot \lceil \log n \rceil$ and $\mathcal{T}$ be the set of strings given by Lemma 2.1, which we shall call *selection strings*. We define the set of *forbidden strings* $\mathcal{F} = \mathcal{F}(\mathcal{T})$ as follows:

$$\mathcal{F} = \{y\colon y \in \{0,1\}^\ell \text{ and } \mathcal{H}(x,y) \leq (1-\alpha)\ell \text{ for all } x \in \mathcal{T}\}.$$

In other words, $\mathcal{F}$ comprises all the strings that are not almost diametrically opposite to some string from $\mathcal{T}$. The following lemma asserts the properties of $\mathcal{T}$ and $\mathcal{F}$ that we shall need later on.

▶ **Lemma 2.2.** *Suppose $u \in \{0,1\}^\ell$. Then the following assertions hold:*
1. *If $u \in \mathcal{T}$, then $\mathcal{H}(u,y) \leq (1-\alpha)\ell$ for each $y \in \mathcal{F}$.*
2. *If $\mathcal{H}(x,u) \geq \beta\ell$ for all $x \in \mathcal{T}$, then there exists $y \in \mathcal{F}$ such that $\mathcal{H}(u,y) \geq (1-\beta)\ell$.*

Proof. Property (1) follows directly from the definition of $\mathcal{F}$, so we proceed to the proof of (2).

Suppose $\mathcal{H}(u,x) \geq \beta\ell$ for all $x \in \mathcal{T}$. If $\overline{u} \in \mathcal{F}$, then we could take $y = \overline{u}$, so suppose that $\overline{u} \notin \mathcal{F}$. This means that there exists $x_0 \in \mathcal{T}$, for which $\mathcal{H}(x_0,\overline{u}) > (1-\alpha)\ell$; equivalently, $\mathcal{H}(\overline{x_0},\overline{u}) < \alpha\ell$. On the other hand, we have that $\mathcal{H}(x_0,u) \geq \beta\ell$, so also $\mathcal{H}(\overline{x_0},\overline{u}) \geq \beta\ell$. Construct y from $\overline{u}$ by taking any set of positions X of size $\beta\ell$ on which $\overline{u}$ and $\overline{x_0}$ have the same letters, and flipping the letters on these positions (replacing 0s with 1s and vice versa). Such a set of positions always exists because $\alpha + \beta < 1$. Then we have that $\mathcal{H}(\overline{x_0},y) = \mathcal{H}(\overline{x_0},\overline{u}) + \beta\ell$, which implies that

$$\alpha\ell = \beta\ell + \beta\ell \leq \mathcal{H}(\overline{x_0},y) < (\alpha+\beta)\ell.$$

We claim that $y \in \mathcal{F}$; suppose otherwise. Since $\mathcal{H}(\overline{x_0},y) \geq \alpha\ell$, then also $\mathcal{H}(x_0,y) \leq (1-\alpha)\ell$. As $y \notin \mathcal{F}$, there must exist some $x_1 \in \mathcal{T}$, $x_0 \neq x_1$, such that $\mathcal{H}(x_1,y) > (1-\alpha)\ell$; equivalently $\mathcal{H}(\overline{x_1},y) < \alpha\ell$. Hence, from the triangle inequality we infer that

$$\mathcal{H}(x_0,x_1) = \mathcal{H}(\overline{x_0},\overline{x_1}) \leq \mathcal{H}(\overline{x_0},y) + \mathcal{H}(y,\overline{x_1}) < (2\alpha+\beta)\ell.$$

This is a contradiction with the assumption that $\mathcal{H}(x_0,x_1) \geq (1/2-\rho)\ell$, which is implied by $x_0, x_1 \in \mathcal{T}$. Indeed, we have that $2\alpha+\beta = \frac{1}{4} < \frac{49}{100} = 1/2-\rho$.

Hence $y \in \mathcal{F}$. By definition we have that $\mathcal{H}(\overline{u},y) = \beta\ell$, which implies that $\mathcal{H}(u,y) = (1-\beta)\ell$. Thus, y satisfies the required properties. ◀

3 Main construction

In this section we provide the proof of Theorem 1.2. Let (G,k) be the input instance of Clique, let $n = |V(G)|$, and without loss of generality assume $k \leq n$. Let C, N be the

constants given by Lemma 2.1. We can assume that $n > N$, because otherwise the instance (G, k) can be solved in constant time. Let $\ell = C\lceil \log n \rceil$. We run the polynomial-time algorithm given by Lemma 2.1 that computes the set $\mathcal{T} \subseteq \{0,1\}^\ell$ of selection strings. Let $\mathcal{F} = \mathcal{F}(\mathcal{T})$ be the set of forbidden strings, as defined in Section 2. Note that $\mathcal{F}$ can be computed in polynomial time directly from the definition, due to $|\{0,1\}^\ell| = n^{\mathcal{O}(1)}$.

We now present the construction of the output instance $(\mathcal{S}, L, d)$ of Closest String. Set $L = k\ell + \gamma\ell$, where $\gamma = \rho + \alpha = \frac{11}{100}$, and partition the set $[L]$ of positions in strings of length L into $k+1$ *blocks*:

- k blocks B_i for $i \in [k]$ of length ℓ each, where $B_i = \{(i-1)\ell + 1, (i-1)\ell + 2, \ldots, i\ell\}$;
- special *balancing block* Γ of length $\gamma\ell$, where $\Gamma = \{k\ell + 1, k\ell + 2, \ldots, L\}$.

For $w \in \{0,1\}^L$ and a contiguous subset of positions X, by $w[X]$ we denote the substring of w formed by positions from X.

Let us first discuss the intuition. The choice the solution string makes on consecutive blocks B_i will encode a selection of a k-tuple of vertices in G. Vertices of G will be mapped one-to-one to strings from $\mathcal{T}$. The family of constraint strings $\mathcal{S}$ will consist of two subfamilies $\mathcal{S}_{\mathrm{sel}}$ and $\mathcal{S}_{\mathrm{adj}}$ with the following roles:

- Strings from $\mathcal{S}_{\mathrm{sel}}$ ensure that on each block B_i, the solution picks a substring that is close to some element of $\mathcal{T}$. The selection of this element encodes the choice of the ith vertex from the k-tuple.
- Strings from $\mathcal{S}_{\mathrm{adj}}$ verify that vertices of the chosen k-tuple are pairwise different and adjacent, and hence they form a clique.

A small technical caveat is that for strings from $\mathcal{S}_{\mathrm{sel}}$ and from $\mathcal{S}_{\mathrm{adj}}$, the intended Hamming distance from the solution string will be slightly different. The role of the balancing block Γ is to equalize this distance by a simple additional construction.

We proceed to the formal description. Since $|V(G)| = |\mathcal{T}|$, let $\iota \colon V(G) \to \mathcal{T}$ be an arbitrary bijection.

The family $\mathcal{S}_{\mathrm{sel}}$ consists of strings $a(i, y, \phi, z)$, for all $i \in [k]$, $y \in \mathcal{F}$, ϕ being a function from $[k] \setminus \{i\}$ to $\{0,1\}$, and z being a binary string of length $\gamma\ell$. String $a(i, y, \phi, z)$ is constructed as follows:

- On block B_i put the string y.
- For each $j \in [k] \setminus \{i\}$, on block B_j put a string consisting of ℓ zeroes if $\phi(j) = 0$, and a string consisting of ℓ ones if $\phi(j) = 1$.
- On balancing block Γ put the string z.

Thus, $|\mathcal{S}_{\mathrm{sel}}| = k \cdot |\mathcal{F}| \cdot 2^{k-1} \cdot 2^{\gamma\ell} \leq 2^k \cdot n^{\mathcal{O}(1)}$; here and in some later estimates we use that $k \leq n$. Also, $\mathcal{S}_{\mathrm{sel}}$ can be constructed in time $2^k \cdot n^{\mathcal{O}(1)}$ directly from the definition.

The family $\mathcal{S}_{\mathrm{adj}}$ consists of strings $b(i, j, (u, v), \psi)$, for all $i, j \in [k]$ with $i < j$, (u, v) being an ordered pair of vertices of G that are either equal or non-adjacent, and ψ being a function from $[k] \setminus \{i, j\}$ to $\{0,1\}$. String $b(i, j, (u, v), \psi)$ is constructed as follows:

- On block B_i put the string $\overline{\iota(u)}$.
- On block B_j put the string $\overline{\iota(v)}$.
- On block B_q, for $q \in [k] \setminus \{i, j\}$, put a string consisting of ℓ zeroes if $\psi(q) = 0$, and a string consisting of ℓ ones if $\psi(q) = 1$.
- On balancing block Γ put a string consisting of $\gamma\ell$ zeroes.

Thus, $|\mathcal{S}_{\mathrm{adj}}| \leq \binom{k}{2} \cdot n^2 \cdot 2^{k-2} \leq 2^k \cdot n^{\mathcal{O}(1)}$. Again, $\mathcal{S}_{\mathrm{adj}}$ can be constructed in time $2^k \cdot n^{\mathcal{O}(1)}$ directly from the definition.

Set $\mathcal{S} = \mathcal{S}_{\mathrm{sel}} \cup \mathcal{S}_{\mathrm{adj}}$ and $d = (k/2 + 1/2 + \rho) \cdot \ell$. This concludes the construction. Its correctness will be verified in two lemmas that mirror the properties listed in Theorem 1.2.

▶ **Lemma 3.1.** *If G contains a clique on k vertices, then there exists a string $w \in \{0,1\}^L$ such that $\mathcal{H}(w,x) \leq d$ for each $x \in \mathcal{S}$.*

Proof. Let $\{c_1, c_2, \ldots, c_k\}$ be a k-clique in G. Construct w by putting $\iota(c_i)$ on block B_i, for each $i \in [k]$, and zeroes on all the positions of the balancing block Γ.

First, take any string $a = a(i, y, \phi, z) \in \mathcal{S}_{\mathrm{sel}}$. Since $\iota(c_i) \in \mathcal{T}$ and $y \in \mathcal{F}$, by Lemma 2.2(1) we infer that $\mathcal{H}(w[B_i], a[B_i]) = \mathcal{H}(\iota(c_i), y) \leq (1-\alpha)\ell$. For each $j \in [k] \setminus \{i\}$, since $\mathcal{H}(\iota(c_j)) = \ell/2$ due to $\iota(c_j) \in \mathcal{T}$, we have that $\mathcal{H}(w[B_j], a[B_j]) = \mathcal{H}(\iota(c_j), a[B_j]) = \ell/2$, regardless of the value of $\phi(j)$. Finally, obviously $\mathcal{H}(w[\Gamma], a[\Gamma]) \leq |\Gamma| = \gamma\ell$. Hence

$$\mathcal{H}(w,a) \leq (1-\alpha)\ell + (k-1)\ell/2 + \gamma\ell = d.$$

Second, take any string $b = b(i, j, (u, v), \psi) \in \mathcal{S}_{\mathrm{adj}}$. Since c_i and c_j are different and adjacent, whereas u and v are either equal or non-adjacent, we have $(c_i, c_j) \neq (u, v)$. Without loss of generality suppose that $c_i \neq u$; the second case will be symmetric. Then $\mathcal{H}(w[B_i], b[B_i]) = \mathcal{H}(\iota(c_i), \overline{\iota(u)}) \leq (1/2+\rho)\ell$, due to property (3) of Lemma 2.1. Obviously, $\mathcal{H}(w[B_j], b[B_j]) \leq |B_j| \leq \ell$. Finally, for every $q \in [k] \setminus \{i, j\}$ we have that $\mathcal{H}(\iota(c_q)) = \ell/2$, and hence $\mathcal{H}(w[B_q], b[B_q]) = \mathcal{H}(\iota(c_q), b[B_q]) = \ell/2$, regardless of the value of $\psi(q)$. Strings w and b match on positions of Γ, so $\mathcal{H}(w[\Gamma], b[\Gamma]) = 0$. Summarizing,

$$\mathcal{H}(w,b) \leq (1/2+\rho)\ell + \ell + (k-2)\ell/2 = d. \qquad \blacktriangleleft$$

▶ **Lemma 3.2.** *If there is a string $w \in \{0,1\}^L$ such that $\mathcal{H}(w,x) < d + \beta\ell$ for each $x \in \mathcal{S}$, then G contains a clique on k vertices.*

Proof. We first prove that on each block B_i, w is close to selecting an element of $\mathcal{T}$.

▶ **Claim 3.3.** *For each $i \in [k]$ there exists a unique $x_i \in \mathcal{T}$ such that $\mathcal{H}(w[B_i], x_i) < \beta\ell$.*

Proof. Uniqueness follows directly from property (3) of Lemma 2.1 and the triangle inequality, so it suffices to prove existence.

Let $u = w[B_i]$. For the sake of contradiction, suppose $\mathcal{H}(u,x) \geq \beta\ell$ for each $x \in \mathcal{T}$. From Lemma 2.2(2) we infer that there exists $y \in \mathcal{F}$ such that $\mathcal{H}(u,y) \geq (1-\beta)\ell$. Let us take $\phi\colon [k]\setminus\{i\} \to \{0,1\}$ defined as follows: $\phi(j) = 0$ if in w the majority of positions of B_j contain a one, and $\phi(j) = 1$ otherwise. Also, define $z = \overline{w[\Gamma]}$. Consider string $a = a(i, y, \phi, z) \in \mathcal{S}_{\mathrm{sel}}$. Then, it follows that

- $\mathcal{H}(w[B_i], a[B_i]) = \mathcal{H}(u,y) \geq (1-\beta)\ell$;
- $\mathcal{H}(w[B_j], a[B_j]) \geq \ell/2$ for each $j \in [k] \setminus \{i\}$;
- $\mathcal{H}(w[\Gamma], a[\Gamma]) = \mathcal{H}(w[\Gamma], \overline{w[\Gamma]}) = |\Gamma| = \gamma\ell$.

Consequently,

$$\mathcal{H}(w,a) \geq (1-\beta)\ell + (k-1)\ell/2 + \gamma\ell = d + \beta\ell.$$

This is a contradiction with the assumption that $\mathcal{H}(w,x) < d + \beta\ell$ for each $x \in \mathcal{S}$. $\blacktriangleleft$

For each $i \in [k]$, let $c_i = \iota^{-1}(x_i)$.

▶ **Claim 3.4.** *For all $i, j \in [k]$ with $i < j$, vertices c_i and c_j are different and adjacent.*

Proof. For the sake of contradiction, suppose c_i and c_j are either equal or non-adjacent. Define $\psi\colon [k] \setminus \{i,j\} \to \{0,1\}$ as follows: $\psi(q) = 0$ if in w the majority of positions of B_q contain a one, and $\psi(q) = 1$ otherwise. Then, for (c_i, c_j) we have constructed string $b = b(i, j, (c_i, c_j), \psi) \in \mathcal{S}_{\mathrm{adj}}$. Observe now that

- $\mathcal{H}(w[B_i], b[B_i]) = \mathcal{H}(w[B_i], \overline{x_i}) > (1-\beta)\ell$, since $\mathcal{H}(w[B_i], x_i) < \beta\ell$;
- Similarly, $\mathcal{H}(w[B_j], b[B_j]) > (1-\beta)\ell$;
- $\mathcal{H}(w[B_q], b[B_q]) \geq \ell/2$ for each $q \in [k] \setminus \{i, j\}$;
- $\mathcal{H}(w[\Gamma], b[\Gamma]) \geq 0$.

Consequently,

$$\mathcal{H}(w, b) \geq 2(1-\beta)\ell + (k-2)\ell/2 = (k/2 + 1 - 2\beta)\ell > d + \beta\ell.$$

This is a contradiction with the assumption that $\mathcal{H}(w, x) < d + \beta\ell$ for each $x \in \mathcal{S}$. ◀

Claim 3.4 asserts that, indeed, $\{c_1, c_2, \ldots, c_k\}$ is a k-clique in G. ◀

Lemmas 3.1 and 3.2 conclude the proof of Theorem 1.2, where c can be taken to be any constant larger than $\frac{d}{\beta\ell \cdot k} \leq \frac{2}{\beta} = 40$.

4 Conclusions

In this paper we have proved that Closest String does not have an EPTAS under the assumption of FPT $\neq$ W[1]. Moreover, under the stronger assumption of the Exponential Time Hypothesis, one can also exclude PTASes with running time $f(\varepsilon) \cdot n^{o(1/\varepsilon)}$, for any computable function f. However, the fastest currently known approximation scheme for Closest String has running time $n^{\mathcal{O}(1/\varepsilon^2)}$ [11]. This leaves a significant gap between the known upper and lower bounds. Despite efforts, we were unable to close this gap, and hence we leave it as an open problem.

References

1. Alexandr Andoni, Piotr Indyk, and Mihai Pătraşcu. On the optimality of the dimensionality reduction method. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 449–458. IEEE Computer Society, 2006.
2. Cristina Bazgan. *Schémas d'approximation et complexité paramétrée*. PhD thesis, Université Paris Sud, 1995. In French.
3. Christina Boucher, Christine Lo, and Daniel Lokshtanov. Consensus Patterns (probably) has no EPTAS. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2015. Full version available at `http://www.ii.uib.no/~daniello/papers/ConsensusPatterns.pdf`.
4. Marco Cesati and Luca Trevisan. On the efficiency of polynomial time approximation schemes. *Inf. Process. Lett.*, 64(4):165–171, 1997.
5. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. URL: `http://dx.doi.org/10.1007/978-3-319-21275-3`, `doi:10.1007/978-3-319-21275-3`.
6. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.
7. Jens Gramm, Rolf Niedermeier, and Peter Rossmanith. Fixed-parameter algorithms for Closest String and related problems. *Algorithmica*, 37(1):25–42, 2003.
8. Ming Li, Bin Ma, and Lusheng Wang. Finding similar regions in many sequences. *J. Comput. Syst. Sci.*, 65(1):73–96, 2002.

9 Ming Li, Bin Ma, and Lusheng Wang. On the Closest String and Substring problems. *J. ACM*, 49(2):157–171, 2002.

10 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 760–776. SIAM, 2011.

11 Bin Ma and Xiaoming Sun. More efficient algorithms for Closest String and Substring problems. *SIAM J. Comput.*, 39(4):1432–1443, 2009.

12 Dániel Marx. Closest substring problems with small distances. *SIAM J. Comput.*, 38(4):1382–1410, 2008.

13 Dániel Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008.