Consensus Strings with Small Maximum Distance and Small Distance Sum

Laurent Bulteau
Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-77454,
Marne-la-Vallée, France
laurent.bulteau@u-pem.fr

Markus L. Schmid
Fachbereich 4 – Abteilung Informatikwissenschaften, Universität Trier, 54286 Trier, Germany
mlschmid@mlschmid.de

Abstract

The parameterised complexity of consensus string problems (Closest String, Closest Substring, Closest String with Outliers) is investigated in a more general setting, i. e., with a bound on the maximum Hamming distance *and* a bound on the sum of Hamming distances between solution and input strings. We completely settle the parameterised complexity of these generalised variants of Closest String and Closest Substring, and partly for Closest String with Outliers; in addition, we answer some open questions from the literature regarding the classical problem variants with only one distance bound. Finally, we investigate the question of polynomial kernels and respective lower bounds.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness, Theory of computation → Fixed parameter tractability, Theory of computation → W hierarchy

Keywords and phrases Consensus String Problems, Closest String, Closest Substring, Parameterised Complexity, Kernelisation

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.1

1 Introduction

Consensus string problems have the following general form: given input strings $S = \{s_1, \ldots, s_k\}$ and a distance bound d, find a string s with distance at most d from the input strings. With the Hamming distance as the central distance measure for strings, there are two obvious types of distance between a single string and a set S of strings: the maximum distance between s and any string from S (called *radius*) and the sum of all distances between s and strings from S (called *distance sum*). The most basic consensus string problem is Closest String, where we get a set S of k length-ℓ strings and a bound d, and ask whether there exists a length-ℓ *solution string* s with radius at most d. This problem is NP-complete (see [16]), but fixed-parameter tractable for many variants (see [17]), including the parameterisation by d, which in biological applications can often be assumed to be small (see [12, 18]). A classical extension is Closest Substring, where the strings of S have length *at most* ℓ, the solution string must have a given length m and the radius bound d is w. r. t. some length-m substrings of the input strings. A parameterised complexity analysis (see [13, 14, 20]) has shown Closest Substring to be harder than Closest String. If we bound the distance sum instead of the radius, then Closest String collapses to a trivial problem, while Closest Substring, which is then called Consensus Patterns, remains

43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018).
Editors: Igor Potapov, Paul Spirakis, and James Worrell; Article No. 1; pp. 1:1–1:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

NP-complete. Closest String with Outliers is a recent extension, which is defined like Closest String, but with the possibility to ignore a given number of t input strings.

The main motivation for consensus string problems comes from the important task of finding similar regions in DNA or other protein sequences, which arises in many different contexts of computational biology, e. g., universal PCR primer design [9, 18, 19, 23], genetic probe design [18], antisense drug design [8, 18], finding transcription factor binding sites in genomic data [25], determining an unbiased consensus of a protein family [3], and motif-recognition [18,21,22]. The consensus string problems are a formalisation of this computational task and most variants of them are NP-hard. However, due to their high practical relevance, it is necessary to solve them despite their intractability, which has motivated the study of their approximability, on the one hand, but also their fixed-parameter tractability, on the other (see the survey [6] for an overview of the parameterised complexity of consensus string problems). This work is a contribution to the latter branch of research.

Problem Definition. Let Σ be a finite alphabet, Σ^* be the set of all strings over Σ, including the empty string ε and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. For $w \in \Sigma^*$, $|w|$ is the length of w and, for every i, $1 \leq i \leq |w|$, by $w[i]$, we refer to the symbol at position i of w. For every $n \in \mathbb{N} \cup \{0\}$, let $\Sigma^n = \{w \in \Sigma^* \mid |w| = n\}$ and $\Sigma^{\leq n} = \bigcup_{i=0}^{n} \Sigma^i$. By $\preceq$, we denote the *substring relation* over the set of strings, i. e., for $u, v \in \Sigma^*$, $u \preceq v$ if $v = xuy$, for some $x, y \in \Sigma^*$. We use the concatenation of sets of strings as usually defined, i. e., for $A, B \subseteq \Sigma^*$, $A \cdot B = \{uv \mid u \in A, v \in B\}$.

For strings $u, v \in \Sigma^*$ with $|u| = |v|$, $\mathsf{d_H}(u, v)$ is the *Hamming distance* between u and v. For a multi-set $S = \{u_i \mid 1 \leq i \leq n\} \subseteq \Sigma^\ell$ and a string $v \in \Sigma^\ell$, for some $\ell \in \mathbb{N}$, the *radius of S (w. r. t. v)* is defined by $\mathsf{r_H}(v, S) = \max\{\mathsf{d_H}(v, u) \mid u \in S\}$ and the *distance sum of S (w. r. t. v)* is defined by $\mathsf{s_H}(v, S) = \sum_{u \in S} \mathsf{d_H}(v, u)$.[1] Next, we state the problem (r, s)-Closest String in full detail, from which we then derive the other considered problems:

(r, s)-Closest String
Instance: Multi-set $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^\ell$, $\ell \in \mathbb{N}$, and integers $d_r, d_s \in \mathbb{N}$.
Question: Is there an $s \in \Sigma^\ell$ with $\mathsf{r_H}(s, S) \leq d_r$ and $\mathsf{s_H}(s, S) \leq d_s$?

For (r, s)-Closest Substring, we have $S \subseteq \Sigma^{\leq \ell}$ and an additional input integer $m \in \mathbb{N}$, and we ask whether there is a multi-set $S' = \{s'_i \mid s'_i \preceq s_i, 1 \leq i \leq k\} \subseteq \Sigma^m$ with $\mathsf{r_H}(s, S') \leq d_r$ and $\mathsf{s_H}(s, S') \leq d_s$. For (r, s)-Closest String with Outliers (or (r, s)-Closest String-wo for short) we have an additional input integer $t \in \mathbb{N}$, and we ask whether there is a multi-set $S' \subseteq S$ with $|S'| = k - t$ such that $\mathsf{r_H}(s, S') \leq d_r$ and $\mathsf{s_H}(s, S') \leq d_s$. We also call (r, s)-Closest String the *general variant* of Closest String, while (r)-Closest String and (s)-Closest String denote the variants, where the only distance bound is d_r or d_s, respectively; we shall also call them the *(r)-* and *(s)-variant* of Closest String. Analogous notation apply to the other consensus string problems. The problem names that are also commonly used in the literature translate into our terminology as follows: Closest String = (r)-Closest String, Closest Substring = (r)-Closest Substring, Consensus Patterns = (s)-Closest Substring and Closest String-wo = (r)-Closest String-wo.

[1] Note that we slightly abuse notation with respect to the subset relation: for a multi-set A and a set B, $A \subseteq B$ means that $A' \subseteq B$, where A' is the set obtained from A by deleting duplicates; for multi-sets A, B, $A \subseteq B$ is defined as usual. Moreover, whenever it is clear from the context that we talk about multi-sets, we also simply use the term *set*.

The motivation for our more general setting with respect to the bounds d_r and d_s is the following. While the distance measures of radius and distance sum are well-motivated, they have, if considered individually, also obvious deficiencies. In the distance sum variant, we may consider strings as close enough that are very close to some, but totally different to the other input strings. In the radius variant, on the other hand, we may consider strings as too different, even though they are very similar to all input strings except one, for which the bound is exceeded by only a small amount. Using an upper bound on the distance per each input string and an upper bound on the total sum of distances caters for these cases.[2]

For any problem K, by $K(p_1, p_2, \ldots)$, we denote the variant of K parameterised by the parameters $p_1, p_2, \ldots$. For unexplained concepts of parameterised complexity, we refer to the textbooks [7, 10, 15].

Known Results. In contrast to graph problems, where interesting parameters are often hidden in the graph structure, string problems typically contain a variety of obvious, but nevertheless interesting parameters that could be exploited in terms of fixed-parameter tractability. For the consensus string problems these are the number of input strings k, their length ℓ, the radius bound d_r, the distance sum bound d_s, the alphabet size $|\Sigma|$, the substring length m (in case of (r,s)-Closest Substring), the number of *outliers* t and *inliers* $k - t$ (in case of (r,s)-Closest String-wo). This leads to a large number of different parameterisations, which justifies the hope for fixed-parameter tractable variants.

The parameterised complexity (w. r. t. the above mentioned parameters) of the radius as well as the distance sum variant of Closest String and Closest Substring has been settled by a sequence of papers (see [13, 14, 16, 17, 20] and, for a survey, [6]), except (s)-Closest Substring with respect to parameter ℓ, which has been neglected in these papers and mentioned as an open problem in [24], in which it is shown that the fixed-parameter tractability results from (r)-Closest String carry over to (r)-Closest Substring, if we additionally parameterise by $(\ell - m)$. The parameterised complexity analysis of the radius variant of Closest String with Outliers has been started more recently in [5] and, to the knowledge of the authors, the distance sum variant has not yet been considered.

The parameterised complexity of the general variants, where we have a bound on both the radius and the distance sum, has not yet been considered in the literature. While there are obvious reductions from the (r)- and (s)-variants to the general variant, these three variants describe, especially in the parameterised setting, rather different problems.

Our Contribution. In this work, we answer some open questions from the literature regarding the (r)- and (s)-variants of the consensus string problems, and we initiate the parameterised complexity analysis of the general variants.

We extend all the FPT-results from (r)-Closest String to the general variant; thus, we completely settle the fixed-parameter tractability of (r,s)-Closest String. While for some parameterisations, this is straightforward, the case of parameter d_r follows from a non-trivial extension of the known branching algorithm for (r)-Closest String(d_r) (see [17]).

For (r,s)-Closest Substring, we classify all parameterised variants as being in FPT or $\mathsf{W}[1]$-hard, which is done by answering the open question whether (s)-Closest Substring(ℓ) is in FPT (see [24]) in the negative (which also settles the parameterised complexity of (s)-Closest Substring) and by slightly adapting the existing FPT-algorithms.

[2] To the knowledge of the authors, optimising both the radius and the distance sum has been considered first in [1], where algorithms for the special case $k = 3$ are considered.

Regarding (r,s)-Closest String-wo, we solve an open question from [5] w. r. t. the radius variant, we show $\mathsf{W}[1]$-hardness for a strong parameterisation of the (s)-variant, we show fixed-parameter tractability for some parameter combinations of the general variant and, as our main result, we present an FPT-algorithm (for the general variant) for parameters d_r and t (which is the same algorithm that shows (r,s)-Closest String$(d_r) \in \mathsf{FPT}$ mentioned above). Many other cases are left open for further research.

Finally, we investigate the question whether the fixed-parameter tractable variants of the considered consensus string problems allow polynomial kernels; thus, continuing a line of work initiated by Basavaraju et al. [2], in which kernelisation lower bounds for (r)-Closest String and (r)-Closest Substring are proved. Our respective main result is a cross-composition from (r)-Closest String into (r)-Closest String-wo.

Due to space constraints, proofs for results marked with $(*)$ are omitted.

2 Closest String and Closest String-wo

In this section, we investigate (r,s)-Closest String and (r,s)-Closest String-wo (and their (r)- and (s)-variants) and we shall first give some useful definitions.

It will be convenient to treat a set $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^\ell$ as a $k \times \ell$ matrix with entries from Σ. By the term *column of* S, we refer to the transpose of a column of the matrix S, which is an element from Σ^k; thus, the introduced string notations apply, e. g., if c is the i^{th} column of S, then $c[j]$ corresponds to $s_j[i]$. A string $s \in \Sigma^\ell$ is a *majority string* (*for a set* $S \subseteq \Sigma^\ell$) if, for every i, $1 \leq i \leq \ell$, $s[i]$ is a symbol with majority in the i^{th} column of S. Obviously, $\mathsf{s_H}(s, S) = \min\{\mathsf{s_H}(s', S) \mid s' \in \Sigma^\ell\}$ if and only if s is a majority string for S. We call a string $s \in \Sigma^\ell$ *radius optimal* or *distance sum optimal* (*with respect to a set* $S \subseteq \Sigma^\ell$) if $\mathsf{r_H}(s, S) = \min\{\mathsf{r_H}(s', S) \mid s' \in \Sigma^\ell\}$ or $\mathsf{s_H}(s, S) = \min\{\mathsf{s_H}(s', S) \mid s' \in \Sigma^\ell\}$, respectively.

It is a well-known fact that (r)-Closest String allows FPT-algorithms for any of the single parameters k, d_r or ℓ, and it is still NP-hard for $|\Sigma| = 2$ (see [17]). While the latter hardness result trivially carries over to (r,s)-Closest String (by setting $d_s = k\,d_r$), we have to modify the FPT-algorithms for extending the fixed-parameter tractability results to (r,s)-Closest String. We start with parameter k, for which we can extend the ILP-approach that is used in [17] to show (r)-Closest String$(k) \in \mathsf{FPT}$.

▶ **Theorem 1 (*).** (r,s)-Closest String$(k) \in \mathsf{FPT}$.

Next, we consider the parameter d_r. For the (r)-variant of (r,s)-Closest String, the fixed-parameter tractability with respect to d_r is shown in [17] by a branching algorithm, which proved itself as rather versatile: it has successfully been extended in [5] to (r)-Closest String-wo(d_r, t) and in [24] to (r)-Closest Substring$(d_r, (\ell - m))$.

We propose an extension of the same branching algorithm, that allows for a bound d_s on the distance sum; thus, it works for (r,s)-Closest String(d_r). In fact, we prove in Theorem 7 an even stronger result, where we also extend the algorithm to exclude up to t outlier strings from the input set S, i. e., we extend it to the problem (r,s)-Closest String-wo(d_r, t). Since Theorem 3 can therefore be seen as a corollary of this result by taking $t = 0$, we only give an informal description of a direct approach that solves (r,s)-Closest String(d_r) (and refer to Theorem 7 for a formal proof).

The core idea is to apply the branching algorithm starting with the majority string for the input set S, instead of any random string from S. Then, as in [17], the algorithm would replace some characters of the current string with characters of the solution string. This way, it can be shown that the distance sum of the current string is always a lower bound of the

Table 1 Results for (r, s)-Closest String.

k	d_r	d_s	$\|\Sigma\|$	ℓ	Result	Note/Ref.
p	–	–	–	–	FPT	Thm. 1
–	**p**	–	–	–	FPT	Thm. 3
–	–	**p**	–	–	FPT	Cor. 4
–	–	–	2	–	NP-hard	from (r)-variant [16]
–	–	–	–	**p**	FPT	Cor. 4

distance sum of the optimal string, which allows to cut any branch where the distance sum goes beyond the threshold d_s. We prove the following lemma, which allows to bound the depth of the search tree (and shall also be used in the proof of Theorem 7 later on):

▶ **Lemma 2 (*).** *Let $S \subseteq \Sigma^\ell$, $s \in \Sigma^\ell$ such that $\mathsf{r_H}(s, S) \leq d_r$, and let s_m be a majority string for S. Then $\mathsf{d_H}(s_m, s) \leq 2d_r$.*

▶ **Theorem 3.** *(r, s)-Closest String$(d_r) \in$ FPT.*

Obviously, we can assume $d_r \leq \ell$ and we can further assume that every column of S contains at least two different symbols (all columns without this property could be removed), which implies $\mathsf{s_H}(s_i, S) \geq \ell$ for every $s \in \Sigma^\ell$; thus, we can assume $\ell \leq d_s$. Consequently, we obtain the following corollary:

▶ **Corollary 4.** *(r, s)-Closest String$(\ell) \in$ FPT, (r, s)-Closest String$(d_s) \in$ FPT.*

This completely settles the parameterised complexity of (r, s)-Closest String with respect to parameters k, d_r, d_s, $|\Sigma|$ and ℓ; recall that the (r)-variant is already settled, while the (s)-variant is trivial.

2.1 (r, s)-Closest String-wo

We now turn to the problem (r, s)-Closest String-wo and we first prove several fixed-parameter tractability results for the general variant; in Sec. 2.2, we consider the (r)- and (s)-variants separately.

First, we note that solving an instance of (r, s)-Closest String-wo(k) can be reduced to solving $f(k)$ many (r, s)-Closest String(k)-instances, which, due to the fixed-parameter tractability of the latter problem, yields the fixed-parameter tractability of the former.

▶ **Theorem 5 (*).** *(r, s)-Closest String-wo$(k) \in$ FPT.*

If the number $k - t$ of inliers exceeds d_s, then an (r, s)-Closest String-wo-instance becomes easily solvable; thus, $k - t$ can be bounded by d_s, which yields the following result:

▶ **Theorem 6 (*).** *(r, s)-Closest String-wo$(d_s, t) \in$ FPT.*

The algorithm introduced in [17] to prove (r)-Closest String$(d_r) \in$ FPT has been extended in [5] with an additional branching that guesses whether a string s_j should be considered an outlier or not; thus, yielding fixed-parameter tractability of (r)-Closest String-wo(d_r, t). We present a non-trivial extension of this algorithm, with a carefully selected starting string, to obtain the fixed-parameter tractability of (r, s)-Closest String-wo(d_r, t) (and, as explained in Section 2, also of (r, s)-Closest String(d_r)):

▶ **Theorem 7.** *(r, s)-Closest String-wo$(d_r, t) \in$ FPT.*

Input:	$s_1 =$	dbaddcbcdbbdbb
$d_r = 5$	$s_2 =$	daaaacbcdccdbd
$d_s = 14$	$s_3 =$	daaddabcaccdbd
$t = 2$	$s_4 =$	aacdaccdcccabd
	$s_5 =$	aacdaabdaccadd
$D = 10$	$s_6 =$	acaaaabcddbadd

Step	S'	t		s'	d	$\mathsf{r_H}(s', S')$	action
1	$\{s_1, s_2, \ldots, s_6\}$	2		◇a◇◇◇◇b◇◇c◇◇◇d	20	13	$s[3] \leftarrow s_1[3]$
2	$\{s_1, s_2, \ldots, s_6\}$	2		◇aa◇◇◇b◇◇c◇◇◇d	19	12	$s[12] \leftarrow s_1[12]$
3	$\{s_1, s_2, \ldots, s_6\}$	2		◇aa◇◇◇b◇◇c◇d◇d	18	11	remove s_6
4	$\{s_1, s_2, \ldots, s_5\}$	1		◇aa◇◇◇b◇◇c◇d◇d	18	11	$s[6] \leftarrow s_1[6]$
5	$\{s_1, s_2, \ldots, s_5\}$	1		◇aa◇◇cb◇◇c◇d◇d	17	10	remove s_5
6	$\{s_1, \ldots, s_4\}$	0		◇aa◇◇cb◇◇c◇d◇d	17	10	
			$s'' =$	daadacbcdccdbd			$s[7] \leftarrow s_4[7]$
7	$\{s_1, \ldots, s_4\}$	0		◇aa◇◇cc◇◇c◇d◇d	16	10	
			$s'' =$	daadacccdccdbd			return S', s''

Figure 1 Example for Algorithm 1 on an instance of (r, s)-Closest String-wo. The shown steps correspond to one branch that yields a correct solution. The algorithm starts with the majority string where disputed characters are replaced by $\diamond$. At each step, the algorithm either inserts a character from an input string at maximal distance from s' (note that even non-disputed characters may be replaced), or removes one such string. When $t = 0$, it is checked whether the completion s'' of s' is a correct solution. At step 7, we return a solution with $\mathsf{r_H}(s'', S') = 5$ and $\mathsf{s_H}(s'', S') = 14$.

Proof. Let (S, d_s, d_r, t) be a positive instance of (r, s)-Closest String-wo(d_r, t) with $k \geq 5t$ (otherwise k can be considered as a parameter). A character x is *frequent in column* i if it has at least as many occurrences as the majority character minus t (thus, for any $S' \subseteq S$, $|S'| \geq |S| - t$, all majority characters for S' are frequent characters). A column i is *disputed* if it contains at least two frequent characters. Let D be the number of disputed columns.

Let (S^*, s^*) be a solution for this instance. In a disputed column i, no character occurs more than $\frac{k+t}{2}$ times, hence, among the $k - t$ strings of S^*, there are at least $(k - t) - \frac{k+t}{2} = \frac{k-3t}{2}$ mismatches at position i. The disputed columns thus introduce at least $D\frac{k-3t}{2}$ mismatches. Since the overall number of mismatches is upper-bounded by $d_r(k - t)$, we have $D \leq \frac{2d_r(k-t)}{k-3t} = 2d_r\left(1 + \frac{2t}{k-3t}\right)$, and, with $k \geq 5t$, the upper-bound $D \leq 4d_r$ follows.

We introduce a new character $\diamond \notin \Sigma$. A string $s' \in (\Sigma \cup \{\diamond\})^\ell$ is a *lower bound* for a solution s^*, if, for every i such that $s'[i] \neq s^*[i]$, either i is a disputed column and $s'[i] = \diamond$, or i is not disputed and $s'[i]$ is the majority character for column i of S^* (which is equal to the majority character for column i of S). Intuitively speaking, whenever a character $s'[i]$ differs from $s^*[i]$, it is the majority character of its column (except for disputed columns in which we use an “undecided” character $\diamond$). Finally, the *completion for* S' of a string $s' \in (\Sigma \cup \{\diamond\})^*$ is the string obtained by replacing each occurrence of $\diamond$ by a majority character of the corresponding column in S'.

We now prove that Algorithm 1 solves (r, s)-Closest String-wo in time at most $O^*((d_r + 1)^{6d_r} 2^{6d_r+t})$, using the following three claims (see Figure 1 for an example).

Claim 1. Any call to Solve Closest String-wo(S', t, s', d) always returns after a time $O^*((d_r + 1)^d 2^{d+t})$.

ALGORITHM 1: Solve Closest String-wo.

```
Input   : $S' \subseteq S$, $t \in \mathbb{N}$, $s' \in (\Sigma \cup \{\diamond\})^\ell$, $d \in \mathbb{N}$
Output : a pair $(S^*, s^*)$ or the symbol $\triangledown$
1  if  $t = 0$ then
2      $s''$ = completion of $s'$ in $S'$;
3      if $\mathsf{s_H}(s'', S') \leq d_s$, and $\mathsf{r_H}(s'', S') \leq d_r$ then return $(S', s'')$;
4      if $d = 0$ then return $\triangledown$;
5  Let $s_j \in S'$ be such that $\mathsf{d_H}(s', s_j)$ is maximal;
6  if $t > 0$ then
7      $sol$ = Solve Closest String-wo($S' \setminus \{s_j\}, t - 1, s', d$);
8      if $sol \neq \triangledown$ then return $sol$;
9  if $d > 0$ then
10     Let $I \subseteq \{1, \ldots, \ell\}$ contain $d_r + 1$ indices $i$ s. t. $s'[i] \neq s_j[i]$ (or all indices if $\mathsf{d_H}(s_j, s') \leq d_r$);
11     for $i \in I$ do
12         $s'' = s'$, $s''[i] = s_j[i]$;
13         $sol$ = Solve Closest String-wo($S', t, s'', d - 1$);
14         if $sol \neq \triangledown$ then return $sol$;
15 return $\triangledown$;
```

Proof of Claim 1. We prove this running time by induction: if $d = t = 0$, then the function returns in Line 3 or 4; thus, it returns after polynomial time. Otherwise, it performs at most $d_r + 1$ recursive calls with parameters $(d - 1, t)$, and one recursive call with parameters $(d, t - 1)$. By induction, the complexity of this step is $O^*((d_r + 1)(d_r + 1)^{d-1} 2^{d+t-1} + (d_r + 1)^d 2^{d+t-1}) = O^*((d_r + 1)^d 2^{d+t})$. ◀

A tuple (S', t', s', d) is *valid* if $|S'| - t' = |S| - t$, there exists an optimal solution (S^*, s^*) for which $S^* \subseteq S'$, $|S^*| = |S'| - t'$, $\mathsf{d_H}(s', s^*) \leq d$, and s' is a lower bound for s^*. A call of the algorithm is *valid* if its parameters form a valid tuple, its *witness* is the pair (S^*, s^*).

Claim 2. Any valid call to Solve Closest String-wo either directly returns a solution or performs at least one recursive valid call.

Proof of Claim 2. Let $S' \subseteq \Sigma^\ell$, $t' \geq 0$, $s' \in (\Sigma \cup \{\diamond\})^\ell$, and $d \geq 0$. Consider the call to Solve Closest String-wo(S', t', s', d). Assume it is valid, with witness (S^*, s^*).

Case 1. If $d = t' = 0$, then $s^* = s'$ and $S^* = S'$. The completion s'' of s' is exactly s', and since (S', s') is a solution, it satisfies the conditions of Line 3 and is returned on Line 3.

Case 2. If $t' = 0$ and $\forall s \in S' : \mathsf{d_H}(s, s') \leq d_r$. Then $S^* = S'$ and s' is a lower bound for s^*. Let s'' be the completion of s'. We show that $\mathsf{s_H}(s'', S') \leq \mathsf{s_H}(s^*, S') \leq d_s$. Indeed, consider any column i with $s''[i] \neq s^*[i]$. Either $s'[i] = \diamond$, in which case $s''[i]$ is the majority character for column i of S', or $s'[i] \neq \diamond$, in which case by the definition of lower bound, i is not a disputed column and $s'[i] = s''[i]$ contains the only frequent character of this column, which is the majority character for S'. In both cases, $s''[i]$ is a majority character for S' in any column where it differs from s^*; thus, it satisfies the upper-bound on the distance sum. Since it also satisfies the distance radius (by the case hypothesis: $\mathsf{d_H}(s, s'') \leq \mathsf{d_H}(s, s') \leq d_r$ for all $s \in S'$), it satisfies the conditions of Line 3; thus, solution (S', s'') is returned on Line 3.

In the following cases, we can thus assume that the algorithm reaches Line 5. Indeed, if it returns on Line 3 then it returns a solution, and if it returns on Line 4 then we have $d = t' = 0$, which is dealt in Case 1 above (the algorithm may not return on this line when it has a valid input). We can thus define s_j to be the string selected in Line 5.

Case 3. $s_j \in S' \setminus S^*$. Then in particular $t' > 0$; and since $S^* \subseteq S' \setminus \{s_j\}$, the recursive call in Line 7 is valid, with the same witness (S^*, s^*).

Case 4. $s_j \in S^*$, $d = 0$ and $t' > 0$. Then $s' = s^*$, let s'_j be any string of $S' \setminus S^*$, and $S^+ = S^* \setminus \{s'_j\} \cup \{s_j\}$. Then the pair (S^+, s^*) is a solution, since $\mathsf{d_H}(s^*, s'_j) \leq \mathsf{d_H}(s^*, s_j)$ by definition of s_j. Thus the recursive call on Line 7 is valid, with witness (S^+, s^*).

Case 5. $s_j \in S^*$, $d > 0$ and $\mathsf{d_H}(s_j, s') > d_r$. Consider the set I defined in Line 10. I has size $d_r + 1$, hence there exists $i_0 \in I$ such that $s_j[i_0] = s^*[i_0]$. Then the recursive call with parameters $(S', t, s'', d-1)$ in Line 13 with $i = i_0$ is valid with the same witness (S^*, s^*). Indeed, s'' is obtained from s' by setting $s''[i_0] = s^*[i_0] \neq s'[i_0]$, hence, all mismatches between s'' and s^* already exist between s' and s^*, which implies that s'' is still a lower bound for s^*. Moreover, $\mathsf{d_H}(s'', s^*) = \mathsf{d_H}(s', s^*) - 1 \leq d - 1$.

From now on, we can assume that $d > 0$ and $t' > 0$. Indeed, $d = 0$ is dealt with in cases 1, 3 and 4, and $t' = 0, d > 0$ is dealt with in cases 2 and 5. Moreover, with cases 3 and 5, we can assume that $s_j \in S^*$ and $\mathsf{d_H}(s_j, s') \leq d_r$ (i.e. $\mathsf{d_H}(s, s') \leq d_r$ for all $s \in S^*$).

Case 6. There exists i_0 such that $s_j[i_0] = s^*[i_0] \neq s'[i_0]$. Then again consider the set I defined in Line 10. Since $\mathsf{d_H}(s_j, s') \leq d_r$, we have $i_0 \in I$, and, with the same argument as in Case 5, there is a valid recursive call in Line 13 when $i = i_0$.

Case 7. For all i, $s_j[i] \neq s'[i] \Rightarrow s_j[i] \neq s^*[i]$. In this case no character from s_j can be used to improve our current solution, so the character switching procedure Line 13 will not improve the solution, but still s_j is part of our witness set S^*, so it is not clear a priori that we can remove s_j from our current solution, i.e. that the recursive call on Line 7 is valid.

We handle this situation as follows. Let s^+ be obtained from s' by filling the $\diamond$-positions of s' with the corresponding symbols of s^*. We now show that (S^*, s^+) is a solution. To this end, let $s \in S^*$. For every i, $1 \leq i \leq \ell$, if $s[i] \neq s^+[i]$, then either $s'[i] = \diamond$ or $s'[i] \in \Sigma$ with $s'[i] = s^+[i]$. In both cases, we have $s[i] \neq s'[i]$, which implies $\mathsf{d_H}(s, s^+) \leq \mathsf{d_H}(s, s') \leq d_r$, i.e., the radius is satisfied. Regarding the distance sum, we note that if $s^+[i] \neq s^*[i]$, then, since occurrences of $\diamond$ of s' have been replaced by the corresponding symbol from s^*, $s'[i] \neq \diamond$, which, by the definition of lower bound, implies that $s^+[i] = s'[i]$ is the majority character for column i of S^*. Consequently, $\sum_{s \in S^*} \mathsf{d_H}(s^+[i], s[i]) \leq \sum_{s \in S^*} \mathsf{d_H}(s^*[i], s[i])$, which implies $\mathsf{s_H}(s^+, S^*) \leq \mathsf{s_H}(s^*, S^*) \leq d_s$.

Having defined a new solution string s^+ (with respect to S^*), we now prove that s^+ is also a solution string with respect to $S^+ = (S^* \setminus \{s_j\}) \cup \{s'_j\}$, where s'_j is any string of $S' \setminus S^*$. To this end, we prove that $\mathsf{d_H}(s'_j, s^+) \leq \mathsf{d_H}(s_j, s^+)$; together with the fact that $\mathsf{d_H}(s'_j, s') \leq d_r$, this implies that (S^+, s^+) is a solution. For two strings $s_1, s_2 \in \Sigma^\ell$, let $d_\diamond(s_1, s_2)$ be the number of mismatches between s_1 and s_2 at positions i such that $s'[i] = \diamond$, and $d_\Sigma(s_1, s_2)$ be the number of mismatches at other positions. Clearly $\mathsf{d_H}(s_1, s_2) = d_\diamond(s_1, s_2) + d_\Sigma(s_1, s_2)$. Comparing strings s_j and s'_j to s', we have $d_\diamond(s_j, s') = d_\diamond(s'_j, s')$ (both distances are equal to the number of occurrences of $\diamond$ in s'). Since $\mathsf{d_H}(s_j, s')$ is maximal, we have $d_\Sigma(s'_j, s') \leq d_\Sigma(s_j, s')$. Consider now s^+. Since s^+ is equal to s' in every non-$\diamond$ characters, we have $d_\Sigma(s'_j, s^+) \leq d_\Sigma(s_j, s^+)$. Finally, for any i such that $s'[i] = \diamond$, by hypothesis of this case we have $s_j[i] \neq s^*[i] = s^+[i]$, hence $d_\diamond(s_j, s^+)$ is equal to the number of occurrences of $\diamond$ in s', which is an upper bound for $d_\diamond(s'_j, s^+)$. Overall, $d(s'_j, s^+) \leq d(s_j, s^+)$, and (S^+, s^+) is a solution.

Thus, (S^+, s^+) is a solution such that $S^+ \subseteq S' \setminus \{s_j\}$, s' is a lower bound for s^+, and $\mathsf{d_H}(s', s^+) \leq d$, hence the recursive call in Line 7 is valid. ◀

It follows from the claim above that any valid call to Solve Closest String-wo returns a solution. Indeed, if it does not directly return a solution, then it receives a solution of a more constrained instance from a valid recursive call, which is returned on Line 8 or 14.

Claim 3. Let s' be the majority string for S where for every disputed column i, $s'[i] = \diamond$. Then Solve Closest String-wo$(S, t, s', 2d_r + D)$ is a valid call.

Proof of Claim 3. Consider a solution (S^*, s^*). We need to check whether $\mathsf{d_H}(s^*, s') \leq 2d_r + D$, and whether s' is a lower bound of s^*. The fact that s' is a lower bound follows from the definition, since $\diamond$ is selected in every disputed column, and the majority character is selected in the other columns. String s^* can be seen as a solution of (r, s)-Closest String over S^*, d_r, d_s, thus, we can use Lemma 2: the distance between s^* and the majority string of S^* is at most $2d_r$. Hence there are at most $2d_r$ mismatches between s' and s^* in non-disputed columns (since in those columns, the majority characters are identical in S and S^*). Adding the D mismatches from disputed columns, we get the $2d_r + D$ upper bound. ◀

◀

2.2 The (r)- and (s)-Variants of Closest String-wo

In [5], the fixed-parameter tractability of (r)-Closest String-wo w. r. t. parameter k and w. r. t. parameters $(|\Sigma|, d_r, k - t)$ are reported as open problems. Since Theorem 5 also applies to (r)-Closest String-wo, the only open cases left for the (r)-variant are the following:

▶ **Open Problem 8.** *What is the fixed-parameter tractability of* (r)-Closest String-wo *with respect to* $(|\Sigma|, k - t)$, $(|\Sigma|, d_r)$ *and* $(|\Sigma|, d_r, k - t)$*?*

Next, we consider the (s)-variant of Closest String-wo. We recall that replacing the radius bound by a bound on the distance sum turns (r)-Closest String into a trivial problem, while (s)-Closest Substring remains hard. The next result shows that Closest String-wo behaves like Closest Substring in this regard. For the proof, we use Multi-Coloured Clique (which is $\mathsf{W}[1]$-hard, see [11]), which is identical to the standard parameterisation of Clique, but the input graph $G = (V, E)$ has a partition $V = V_1 \cup \ldots \cup V_{\mathrm{k_c}}$, such that every V_i, $1 \leq i \leq \mathrm{k_c}$, is an independent set (we denote the parameter by $\mathrm{k_c}$ to avoid confusion with the number of input strings k).

▶ **Theorem 9.** (s)-Closest String-wo$(d_s, \ell, k - t)$ *is* $\mathsf{W}[1]$*-hard.*

Proof. Let $G = (V_1 \cup \ldots \cup V_{\mathrm{k_c}}, E)$ be a Multi-Coloured Clique-instance. We assume that, for some $q \in \mathbb{N}$, $V_i = \{v_{i,1}, v_{i,2}, \ldots v_{i,q}\}$, $1 \leq i \leq \mathrm{k_c}$, i. e., each vertex has an index depending on its colour-class and its rank within its colour-class. Let $\Sigma = V \cup \Gamma$, where Γ is some alphabet with $|\Gamma| = |E|(\mathrm{k_c} - 2)$. For every $e = (v_{i,j}, v_{i',j'}) \in E$, let $s_e \in \Sigma^{\mathrm{k_c}}$ with $s_e[i] = v_{i,j}$, $s_e[i'] = v_{i',j'}$ and all other non-defined positions are filled with symbols from Γ such that each $x \in \Gamma$ has exactly one occurrence in the strings s_e, $e \in E$. We set $S = \{s_e \mid e \in E\}$, $t = |E| - \binom{\mathrm{k_c}}{2}$ (i. e., the number of inliers is $\binom{\mathrm{k_c}}{2}$) and $d_s = \binom{\mathrm{k_c}}{2}(\mathrm{k_c} - 2)$.

Let K be a clique of G of size $\mathrm{k_c}$, let $s \in \Sigma^{\mathrm{k_c}}$ be defined by $\{s[i]\} = K \cap V_i$, $1 \leq i \leq \mathrm{k_c}$, and let $S' = \{s_e \mid e \subseteq K\}$. Since $\mathsf{d_H}(s, s') = \mathrm{k_c} - 2$, for every $s' \in S'$, $\mathsf{s_H}(s, S') = d_s$. Consequently, S' and s is a solution for the (s)-Closest String-wo-instance S, t, d_s.

Now let $s \in \Sigma^{\mathrm{k_c}}$ and $S' \subseteq S$ with $|S'| = \binom{\mathrm{k_c}}{2}$ be a solution for the (s)-Closest String-wo-instance S, t, d_s. If, for some $s'_1 \in S'$, $\mathsf{d_H}(s'_1, s) \geq \mathrm{k_c} - 1$, then there is an $s'_2 \in S'$ with $\mathsf{d_H}(s'_2, s) \leq \mathrm{k_c} - 3$. Thus, for some i, $1 \leq i \leq \mathrm{k_c}$, $s[i] = s'_2[i]$ and $s'_2[i] \in \Gamma$, which implies that replacing $s[i]$ by $s'_1[i]$ does not increase $\mathsf{s_H}(s, S')$. Moreover, after this modification, $\mathsf{d_H}(s'_1, s)$ has decreased by 1, while $\mathsf{d_H}(s'_2, s) \leq \mathrm{k_c} - 2$. By repeating such operations, we can transform s such that $\mathsf{d_H}(s', s) \leq \mathrm{k_c} - 2$, $s' \in S'$. Next, assume that, for some i, $1 \leq i \leq \mathrm{k_c}$, there is an $S'' \subseteq S'$ with $|S''| = \mathrm{k_c}$ and, for every $s' \in S''$, $s[i] = s'[i]$. Since $\mathsf{d_H}(s', s) \leq \mathrm{k_c} - 2$ for every

Table 2 Results for (r,s)-Closest String-wo, including (r)- and (s)-variants.

k	t	$\lvert\Sigma\rvert$	ℓ	d_r	d_s	$k-t$	Result	Note/Ref.
p	–	–	–	–	–	–	FPT	Thm. 5, Open Prob. in [5]
–	0	2	–	–	–	–	NP-hard	even for d_r-var., but P for d_s-var.
–	**p**	–	**p**	–	–	–	FPT	$d_r \leq \ell$
–	**p**	–	–	**p**	–	–	FPT	Thm. 7, and [5] for d_r-var.
–	**p**	–	–	–	**p**	–	FPT	Thm. 6
–	**p**	–	–	–	–	**p**	FPT	$k = t + (k-t)$
–	–	**p**	**p**	–	–	–	FPT	trivial
–	–	**p**	–	$\star$	$\star$	$\star$	Open	param. $\lvert\Sigma\rvert$ and some of $d_r, d_s, k-t$
–	–	–	**p**	**p**	**p**	**p**	W[1]-hard	even for d_r-var. [5] and d_s-var. (Thm. 9)

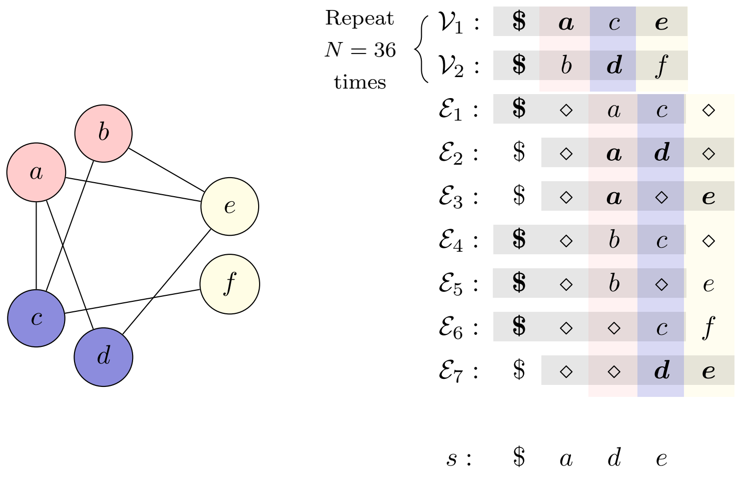
$s' \in S''$, pigeon-hole principle implies that there are $s'_1, s'_2 \in S''$ with $s'_1[i'] = s'_2[i'] = s[i']$, for some i', $1 \leq i' \leq \mathrm{k_c}$, and $i' \neq i$, which, by the structure of the strings of S, is a contradiction. Consequently, for every i, $1 \leq i \leq \mathrm{k_c}$, s matches with at most $\mathrm{k_c} - 1$ strings from S' at position i. Since there are at least $2\binom{\mathrm{k_c}}{2} = \mathrm{k_c}(\mathrm{k_c} - 1)$ matches, we conclude that, for every i, $1 \leq i \leq \mathrm{k_c}$, $s[i]$ matches exactly $k_c - 1$ times with the i^{th} position of a string from S'. Hence, $s[i] \in V_i$, $1 \leq i \leq \mathrm{k_c}$, i.e., $s = v_{1,r_1} v_{2,r_2} \ldots v_{\mathrm{k_c},r_{\mathrm{k_c}}}$, for some $r_j \in \{1, 2, \ldots, q\}$, $1 \leq j \leq \mathrm{k_c}$. Let $K = \{v_{1,r_1}, v_{2,r_2}, \ldots, v_{\mathrm{k_c},r_{\mathrm{k_c}}}\}$. For every $s' \in S'$, by definition of the strings s_e, we have $\mathsf{d_H}(s, s') \geq \mathrm{k_c} - 2$, combining with the lower-bound proved ealier, we conclude $\mathsf{d_H}(s, s') = \mathrm{k_c} - 2$, for every $s' \in S$. Now let $e = (v_{i,j}, v_{i',j'}) \in E$ be such that $s_e \in S'$. From $\mathsf{d_H}(s, s_e) = \mathrm{k_c} - 2$ its follows that $s[i] = v_{i,j}$ and $s[i'] = v_{i',j'}$, which implies $e \subseteq K$. Since $|S| = \binom{\mathrm{k_c}}{2}$, there are $\binom{\mathrm{k_c}}{2}$ edges connecting vertices from K; thus, K is a clique. ◀

Setting $d_r = \mathrm{k_c} - 2$ instead of $d_s = \binom{\mathrm{k_c}}{2}(\mathrm{k_c} - 2)$ in the reduction of Theorem 9 leads to a simpler proof for the W[1]-hardness of (r)-Closest String-wo$(d_r, \ell, k - t)$ shown in [5] (on the other hand, the reduction of [5] does not work for (s)-Closest String-wo$(d_s, \ell, k - t)$). The results obtained in this section are summarized in Table 2.

3 Closest Substring

In this section, we consider the problem (r,s)-Closest Substring and, as done in Section 2 for (r,s)-Closest String, we classify all parameterisations of (r,s)-Closest Substring (and its (r)- and (s)-variants) with respect to the parameters ℓ, k, m, d_r, d_s and $|\Sigma|$ into either fixed-parameter tractable or W[1]-hard. Of course, many of those questions are already solved in the literature, but, unlike for (r,s)-Closest String, not all cases of the (r)- and (s)-variants are settled, i.e., the status of (s)-Closest Substring(ℓ) is unknown, which is mentioned as open problem in [24]. We shall first close this gap by defining a reduction from Multi-Coloured Clique to (s)-Closest Substring.

Let $G = (V_1 \cup \ldots \cup V_{\mathrm{k_c}}, E)$ be a Multi-Coloured Clique-instance. We assume that, for some $q \in \mathbb{N}$, $V_i = \{v_{i,1}, v_{i,2}, \ldots v_{i,q}\}$, $1 \leq i \leq \mathrm{k_c}$, i.e., each vertex has an index depending on its colour-class and its rank within its colour-class. Let $\Sigma = V \cup \{\$, \diamond\}$. For every j, $1 \leq j \leq q$, we list all j^{th} elements of the colour-classes as a string $\mathcal{V}_j = \$ v_{1,j} v_{2,j} \ldots v_{\mathrm{k_c},j}$. For every edge $e = (v_{i,j}, v_{i',j'})$ with $i < i'$, we define a string $\mathcal{E}_e = \$ \diamond^i v_{i,j} \diamond^{i'-i-1} v_{i',j'} \diamond^{\mathrm{k_c} - i' - 1}$. Note that $\mathcal{E}_e = \$ \diamond \mathcal{E}'_e$, where $|\mathcal{E}'_e| = \mathrm{k_c}$, the positions i and i' of $\mathcal{E}'_e$ are $v_{i,j}$ and $v_{i',j'}$, respectively, and all remaining positions are $\diamond$. The (s)-Closest Substring-instance is now defined as follows. Let S contain $N = |E|(\mathrm{k_c} + 2) + 1$ occurrences of each $\mathcal{V}_j$, $1 \leq j \leq q$, and one occurrence of each $\mathcal{E}_e$, $e \in E$, and let $m = \mathrm{k_c} + 1$. We note that $\ell = \mathrm{k_c} + 2$. See Figure 2 for an example.



$\mathcal{V}_1$:	**$**	***a***	c	***e***	
$\mathcal{V}_2$:	**$**	b	***d***	f	
$\mathcal{E}_1$:	**$**	$\diamond$	a	c	$\diamond$
$\mathcal{E}_2$:	$	$\diamond$	***a***	***d***	$\diamond$
$\mathcal{E}_3$:	$	$\diamond$	***a***	$\diamond$	***e***
$\mathcal{E}_4$:	**$**	$\diamond$	b	c	$\diamond$
$\mathcal{E}_5$:	**$**	$\diamond$	b	$\diamond$	e
$\mathcal{E}_6$:	**$**	$\diamond$	$\diamond$	c	f
$\mathcal{E}_7$:	$	$\diamond$	$\diamond$	***d***	***e***
s :	$	a	d	e	

Figure 2 Illustration of the parameterized reduction from a Multi-Coloured Clique-instance to (s)-Closest Substring. The colour-classes of the graph are $V_1 = \{a, b\}$ (red), $V_2 = \{c, d\}$ (blue) and $V_1 = \{e, f\}$ (yellow), the occurrences of symbols from V in the strings $\mathcal{V}_j$ and $\mathcal{E}_i$ are coloured according to their colour-classes. The string $s = \$ade$ is an optimal solution with respect to the substrings emphasised with grey background (positions producing a match are in bold). Note that vertices $\{a, d, e\}$ form a clique in G.

In the following, we extend the notation of *radius optimal* and *distance sum optimal* to sets $S \subseteq \Sigma^{\leq \ell}$ and strings $s \in \Sigma^m$ in the natural way by taking all sets S' of length-m substrings of the string in S into account. The next lemma shows that distance sum optimal strings (with respect to S and m) are basically lists of vertices from each colour-class.

▶ **Lemma 10 (*).** *If $s \in \Sigma^{k+1}$ is distance sum optimal w. r. t. S, then $s \in \{\$\} \cdot V_1 \cdot V_2 \cdot \ldots \cdot V_k$.*

Now let s be distance sum optimal with respect to S and m. From Lemma 10, we can conclude that $s = \$ v_{1,r_1} v_{2,r_2} \ldots v_{k_c,r_{k_c}}$, for some $r_j \in \{1, 2, \ldots, q\}$, $1 \leq j \leq k_c$. Let K be the corresponding set of vertices, i. e., $K = \{v_{1,r_1}, v_{2,r_2}, \ldots, v_{k_c,r_{k_c}}\}$.

▶ **Lemma 11 (*).** *Let $e \in E$. The optimal distance between s and a length-$(\mathrm{k_c} + 1)$ substring of $\mathcal{E}_e$ is $\mathrm{k_c} - 1$ if $e \subseteq K$, and $\mathrm{k_c}$ otherwise.*

Using the lemmas from above, we can now show the correctness of the reduction.

▶ **Theorem 12.** *(s)-Closest Substring(ℓ, m) is $\mathsf{W}[1]$-hard.*

Proof. Let $s \in \Sigma^{\mathrm{k_c}+1}$ be distance sum optimal with respect to S and m, and let K be the corresponding set of vertices. We first note that the total distance from s to the N copies of the strings $\mathcal{V}_j$, $1 \leq j \leq q$, is exactly $Nq\,\mathrm{k_c}$. According to Lemma 11, for every $e \in E$, the optimal distance sum between s and the respective substring of $\mathcal{E}_e$ is $\mathrm{k_c} - 1$ if $e \subseteq K$, and $\mathrm{k_c}$ otherwise. Hence, the total distance sum from s to the respective substrings of $\mathcal{E}_e$, $e \in E$, is $|E|\,\mathrm{k_c} - r$, where $r = \{e \in E \mid e \subseteq K\}$, and the total distance sum between s and S is therefore $Nq\,\mathrm{k_c} + |E|\,\mathrm{k_c} - r$. This implies that the distance sum between s and S is $Nq\,\mathrm{k_c} + |E|\,\mathrm{k_c} - \frac{\mathrm{k_c}(\mathrm{k_c}-1)}{2}$ if and only if $r = \frac{\mathrm{k_c}(\mathrm{k_c}-1)}{2}$ if and only if K is a clique of size $\mathrm{k_c}$. Consequently, the above reduction, with the addition of $d_s = Nq\,\mathrm{k_c} + |E|\,\mathrm{k_c} - \frac{\mathrm{k_c}(\mathrm{k_c}-1)}{2}$, is a parameterised reduction from Multi-Coloured Clique to (s)-Closest Substring(ℓ, m). ◀

Table 3 Results for (s)-Closest Substring.

ℓ	k	m	d_s	$\|\Sigma\|$	Result	Reference
–	–	**p**	–	**p**	FPT	trivial
p	–	–	–	**p**	FPT	[24]
p	**p**	–	–	–	FPT	[24]
p	–	–	**p**	–	FPT	[24]
–	–	–	**p**	**p**	FPT	[20]
–	**p**	–	–	**2**	W[1]-hard	[14]
–	**p**	**p**	**p**	–	W[1]-hard	[14]
p	–	**p**	–	–	W[1]-hard	Thm. 12

Table 4 Results for (r, s)-Closest Substring.

ℓ	k	m	d_r	d_s	$\|\Sigma\|$	Result	Reference
–	–	**p**	–	–	**p**	FPT	Thm. 14
p	**p**	–	–	–	–	FPT	Thm. 14
p	–	–	–	**p**	–	FPT	Thm. 14
p	–	–	–	–	**p**	FPT	Thm. 14
p	–	**p**	**p**	–	–	W[1]-hard	Cor. 13, Open Prob. in [24]
–	**p**	–	**p**	**p**	**p**	W[1]-hard	[20]
–	**p**	**p**	**p**	**p**	–	W[1]-hard	[14]

As illustrated by Table 3, Theorem 12 together with known results from the literature completely settle the parameterised complexity of (s)-Closest Substring.

Moving on to the problem (r, s)-Closest Substring, we first observe that reducing (s)-Closest Substring to (r, s)-Closest Substring by setting $d_r = m$ is a parameterised reduction from (s)-Closest Substring(ℓ, m) to (r, s)-Closest Substring(ℓ, m, d_r), which implies the following corollary:

▶ **Corollary 13.** *(r, s)-Closest Substring(ℓ, m, d_r) is W[1]-hard.*

Next, we consider several fixed-parameter tractable variants of (r, s)-Closest Substring.

▶ **Theorem 14 (*).** *(r, s)-Closest Substring(x) ∈ FPT, for every $x \in \{(m, |\Sigma|), (\ell, k), (\ell, |\Sigma|), (\ell, d_s)\}$.*

It remains to observe that all remaining parameterisations of (r, s)-Closest Substring are W[1]-hard. More precisely, it is known that (r)-Closest Substring is W[1]-hard for parameterisations $(k, d_r, |\Sigma|)$ (see [20]) and (k, m, d_r) (see [14]). Hence, the obvious reduction from (r)-Closest Substring to (r, s)-Closest Substring, i. e., setting $d_s = k\, d_r$, shows that (r, s)-Closest Substring is W[1]-hard for parameterisations $(k, d_r, d_s, |\Sigma|)$ and (k, m, d_r, d_s). As can be checked with the help of Table 4, this now classifies all parameterised variants of (r, s)-Closest Substring.

4 Kernelisation

Neither (r)-Closest String$(d_r, \ell, |\Sigma|)$ nor (r)-Closest Substring(k, m, d_r) admit polynomial kernels unless coNP ⊆ NP/Poly (see [2]), and (r)-Closest String(k, d_r) has a kernel of size $O(k^2 d_r \log k)$ (see [17]). From these results, we can conclude the following:

▶ **Proposition 15** (*).

- (r,s)-Closest String$(d_r, \ell, |\Sigma|)$ *has no polynomial kernel unless* $\mathsf{coNP} \subseteq \mathsf{NP/Poly}$*.*
- (r,s)-Closest String(k, d_r) *has a kernel of size* $\mathrm{O}(k^2 d_r \log k)$*.*
- (r,s)-Closest String(d_s) *has a kernel of size* $\mathrm{O}((d_s)^3 \log d_s)$*.*

This only leaves the case open, where only k (or k and $|\Sigma|$, which, due to the dependency $|\Sigma| \leq k$ (see [17]), is the same question) is a parameter (regarding this case, note that for (r)-Closest String(k) no combinatorial kernel or combinatorial FPT-algorithm is known).

▶ **Proposition 16** (*).

- (r,s)-Closest Substring$(k, m, d_r, d_s, |\Sigma|)$ *has no polynomial kernel unless* $\mathsf{coNP} \subseteq \mathsf{NP/Poly}$*.*
- (r,s)-Closest Substring(ℓ, k) *and* (r,s)-Closest Substring(ℓ, d_s) *have kernels of size* $\mathrm{O}(\ell k)$ *and* $\mathrm{O}(\ell d_s)$*, respectively.*

This almost settles the (r,s)-variant, for which only the parameterisation $(\ell, |\Sigma|)$ is open. For the (r)-variant, the parameterisations ℓ, (ℓ, d_r) and $(\ell, |\Sigma|)$, and for the (s)-variant, the parameterisations $(m, |\Sigma|)$ and $(d_s, |\Sigma|)$ are open.

For (r)-Closest String-wo no kernelisation lower bounds are known so far. However, the following can be concluded from [2]:

▶ **Proposition 17** (*). *(r)-Closest String-wo$(d_r, \ell, t, |\Sigma|)$ has no polynomial kernel unless* $\mathsf{coNP} \subseteq \mathsf{NP/Poly}$*.*

By a cross-composition[3] from (r)-Closest String into (r)-Closest String-wo, we can rule out a polynomial kernel for the parameterisation $(d_r, d_s, \ell, (k-t), |\Sigma|)$.

To this end, we define a polynomial equivalence relation $\sim$ over the (r)-Closest String-instances as follows. For $j \in \{1, 2\}$, let $S_j = \{s_{j,i} \mid 1 \leq i \leq k_j\} \subseteq \Sigma^{\ell_j}$ and $d_{r,j} \in \mathbb{N}$. Then $(S_1, d_{r,1}) \sim (S_2, d_{r,2})$ if $k_1 = k_2$, $\ell_1 = \ell_2$ and $d_{r,1} = d_{r,2}$. Now let $(S_1, d_r), (S_2, d_r), \ldots, (S_q, d_r)$ be $\sim$-equivalent (r)-Closest String-instances, where, for the sake of convenience, $S_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,k}\} \subseteq \Sigma^{\ell}$, $1 \leq i \leq q$. For every i, $1 \leq i \leq q$, let B_i denote the binary representation of i with exactly $\lceil \log(q) \rceil$ bits, and let $C_i = (B_i)^{2d_r+1}$. Moreover, for every i, $1 \leq i \leq q$, let $S'_i = \{s'_{i,1}, s'_{i,2}, \ldots, s'_{i,k}\}$, where, for every j, $1 \leq j \leq k$, $s'_{i,j} = s_{i,j} C_i$. Finally, let the (r,s)-Closest String-wo-instance be (S', d'_r, d'_s, t) with $S' = \bigcup_{i=1}^{q} S'_i$, $d'_r = d_r$, $d'_s = k d_r$ and $t = (q-1)k$.

▶ **Theorem 18** (*). *(r,s)-Closest String-wo$(d_r, d_s, \ell, (k-t), |\Sigma|)$ does not admit a polynomial kernel unless* $\mathsf{coNP} \subseteq \mathsf{NP/Poly}$*.*

5 Conclusions

The parameterised complexity of the (r)-, (s)- and general variant of Closest String and Closest Substring with respect to $\ell, k, m, d_r, d_s, |\Sigma|$ is now completely settled. For (r,s)-Closest Substring, where positive results are less abundant, it might be worthwhile to identify other parameters that yield fixed-parameter tractability. For (r,s)-Closest String, it should be pointed out that the FPT-algorithms with respect to k are based on ILP and are most likely practically not relevant; direct combinatorial FPT-algorithms are still unknown. For the outlier variant of (r,s)-Closest String, many cases are left open, most prominently, the ones with $|\Sigma|$ as parameter, and we expect those to be challenging. Moreover, for several FPT-variants, the existence of polynomial kernels is not yet answered.

[3] For the technique of cross-composition, see Bodlaender et al. [4].

References

1 A. Amir, G. M. Landau, J. C. Na, H. Park, K. Park, and J. S. Sim. Efficient algorithms for consensus string problems minimizing both distance sum and radius. *Theoretical Computer Science*, 412:5239–5246, 2011.

2 M. Basavaraju, F. Panolan, A. Rai, M. S. Ramanujan, and S. Saurabh. On the kernelization complexity of string problems. In *Proc. 20th International Conference on Computing and Combinatorics, COCOON 2014*, volume 8591 of *LNCS*, pages 141–153, 2014.

3 A. Ben-Dor, G. Lancia, R. Ravi, and J. Perone. Banishing bias from consensus sequences. In *Proc. 8th Annual Symposium on Combinatorial Pattern Matching, CPM 1997*, volume 1264 of *LNCS*, pages 247–261, 1997.

4 H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal of Discrete Mathematics*, 28(1):277–305, 2014.

5 C. Boucher and B. Ma. Closest string with outliers. *BMC Bioinformatics*, 12:S55, 2011.

6 L. Bulteau, F. Hüffner, C. Komusiewicz, and R. Niedermeier. Multivariate algorithmics for NP-hard string problems. *Bulletin of the EATCS*, 114:31–73, 2014.

7 M. Cygan, F. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

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

9 J. Dopazo, A. Rodríguez, J. Sáiz, and F. Sobrino. Design of primers for PCR amplification of highly variable genomes. *Computer Applications in the Biosciences*, 9(2):123–125, 1993.

10 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

11 Rodney G Downey and Michael Ralph Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

12 P. A. Evans, A. Smith, and H. T. Wareham. The parameterized complexity of p-center approximate substring problems. Technical Report TR01-149, Faculty of Computer Science, University of New Brunswick, Canada, 2001.

13 P. A. Evans, A. D. Smith, and H. T. Wareham. On the complexity of finding common approximate substrings. *Theoretical Computer Science*, 306:407–430, 2003.

14 M. R. Fellows, J. Gramm, and R. Niedermeier. On the parameterized intractability of motif search problems. *Combinatorica*, 26:141–167, 2006.

15 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

16 M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30:113–119, 1997.

17 J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37:25–42, 2003.

18 J. K. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. *Information and Computation*, 185:41–55, 2003.

19 K. Lucas, M. Busch, S. Mössinger, and J. A. Thompson. An improved microcomputer program for finding gene- or gene family-specific oligonucleotides suitable as primers for polymerase chain reactions or as probes. *Computer Applications in the Biosciences*, 7(4):525–529, 1991.

20 D. Marx. Closest substring problems with small distances. *SIAM Journal on Computing*, 38:1382–1410, 2008.

21 G. Pavesi, G. Mauri, and G. Pesole. An algorithm for finding signals of unknown length in DNA sequences. *Bioinformatics*, 17:S207–S214, 2001.

22 P. Pevzner and S. Sze. Combinatorial approaches to finding subtle signals in DNA strings. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, ISMB 2000*, pages 269–278, 2000.

23 V. Proutski and E. C. Holmes. Primer master: a new program for the design and analysis of PCR primers. *Computer Applications in the Biosciences*, 12(3):253–255, 1996.

24 Markus L. Schmid. Finding consensus strings with small length difference between input and solution strings. *ACM Transactions on Computation Theory*, 9(3):13:1–13:18, 2017.

25 M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Régnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu. Assessing computational tools for the discovery of transcription factor binding sites. *Nature Biotechnology*, 23(1):137–144, 2005.