Noname manuscript No.
(will be inserted by the editor)

Consensus Strings with Small Maximum Distance and Small Distance Sum

Laurent Bulteau · Markus L. Schmid

Received: date / Accepted: date

Abstract The parameterised complexity of various 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.

Keywords Consensus String Problems · Closest String · Closest Substring · Parameterised Complexity · Kernelisation

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

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

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

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 [17]), but fixed-parameter tractable for many variants (see [18]), including the parameterisation by d, which in biological applications can often be assumed to be small (see [13,19]). 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 with respect to some length-m substrings of the input strings. A parameterised complexity analysis (see [14,15,22]) 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 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 (see [6]).

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 [10,19,21,25], genetic probe design [19], antisense drug design [19,9], finding transcription factor binding sites in genomic data [27], determining an unbiased consensus of a protein family [3], and motif-recognition [19, 23,24]. The consensus string problems are a formalisation of these computational tasks 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 [7] for an overview of the parameterised complexity of consensus string problems). This work is a contribution to the latter branch of research. In the following, we motivate in more detail the research carried out in this paper.

From a theoretical point of view, these consensus string problems (as is usually the case for string problems) have a large number of quite natural and obvious numerical parameters, e. g., number of input strings, their lengths, alphabet size, the distance bounds and so on. Therefore, from a parameterised complexity point of view, they have a somewhat different nature than the typical graph problems, for which we have the obvious standard parameterisations (usually some size bound that is part of the input, e. g., the clique-size for Clique) or more complex structural parameters (like width-parameters as treewidth and so on); on the other hand, obvious numerical parameters, e. g., number of vertices or edges, are usually not interesting (with the degree of a graph being an exception). Consequently, for string problems, the challenge is to discover among the rather large number of different combinations of these obvious parameters those that yield fixed-parameter tractability; thus, obtaining a complete "map of fixed-parameter tractability" of the problem.

From a more practical point of view, we note that for string problems, which are usually motivated by tasks from computational biology, it is often

the case that it is known which parameters can be considered to be small in practical scenarios and which do not have this desirable property. This leads to parameters (or parameter combinations) that are more important than others. Consequently, the most pressing question is whether we can achieve fixed-parameter tractability for these "small" parameters. Furthermore, the knowledge of which parameters are important may guide an algorithmic engineering process, e. g., if we have achieved fixed-parameter tractability with respect to an important parameter, but the problem formalisation does not quite cover the practical scenario, we can search for modifications of the problem that maintain the fixed-parameter tractability and are still suitable for practical scenarios with small parameter values. For example, as explained in [6], in practical applications of consensus string problems it cannot always be avoided that the set of input strings includes a small number of strings that are quite different from all the others. In order to still get a solution, we would have to drastically increase the radius bound, which also leads to a solution that is undesirable from a practical point of view. Instead, it makes much more sense to directly cater for this presence of "outliers" by modifying the problem formulation accordingly. This is the motivation for the outlier-variant introduced in [6] (note that Closest String with Outliers parameterised by the radius bound and number of outliers is fixed-parameter tractable [6]). In particular, if we find a suitable solution string when some outliers are excluded, then it seems natural that the the initial decision of including these strings needs to be revised (as pointed out in [6], this is another motivation for the outlier-variant).

In this work, we propose a different modification, which leads to a generalisation of all the consensus string problems mentioned above: we consider the case where we have a radius bound *and* a distance sum bound at the same time. From a theoretical point of view, this leads to the question which of the fixed-parameter tractable cases of the variants with only one bound are still fixed-parameter tractable if we consider both bounds. However, we believe this problem can also be relevant from a practical point of view, since having both a radius bound and a distance sum bound allows for a finer tuning of the solutions (similar as the addition of outliers). We shall motivate this by an example.

Assume that by solving the outlier-variant for a set of strings, we have found out that our desired radius bound can only be met by declaring strings as outliers that should not be outliers (i. e., strings for which we know for certain that they should be included in the input set), or that a solution string cannot be found for a number of outliers that is small enough that the algorithm's running time is still acceptable. In this case, slightly increasing the radius bound seems inevitable, but it is still reasonable to require that this larger distance to the solution string should be used to its full capacity only by a small number of input strings. This requirement could be formulated by adding a distance sum bound that is significantly smaller than the number of input strings multiplied by the radius bound. It is also reasonable to think about allowing both, a small number of outliers that handles strings that

should not be in the input set at all and a distance sum bound that takes care of bounding the number of "high distance" input strings.

Next, we define more formally the consensus string problems considered in this paper and then explain in full detail the respective known results in the literature and our new contributions.

1.1 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 (*with respect to* 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 (*with respect to* v) is defined by $\mathsf{s_H}(v, S) = \sum_{u \in S} \mathsf{d_H}(v, u)$.[1]

Next, we state the consensus string problems to be investigated. The most basic one is (r, s)-Closest String (denoted by (r, s)-CloseStr in the following):

(r, s)-CloseStr	
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 $\mathsf{s_H}(s, S) \leq d_s$?

If we allow a given number of the input strings to be excluded and require the bounds d_r and d_s to be satisfied with respect to the remaining strings, we obtain the problem (r, s)-Closest String with Outliers (this will also be called the *outlier-variant* (of (r, s)-CloseStr) and will be denoted by (r, s)-CloseStr-wo):

(r, s)-CloseStr-wo	
Instance:	Multi-set $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^\ell$, $\ell \in \mathbb{N}$, and integers $d_r, d_s, t \in \mathbb{N}$.
Question:	Is there an $s \in \Sigma^\ell$ and $S' \subseteq S$ with $\lvert S' \rvert = k - t$ such that $\mathsf{r_H}(s, S') \leq d_r$ and $\mathsf{s_H}(s, S') \leq d_s$?

For the problem (r, s)-Closest Substring (which will also be called the

[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*.

substring-variant (of (r,s)-CloseStr) and is denoted by (r,s)-CloseSubstr), the input words can have different lengths and we are asking for a string that satisfies the bounds d_r and d_s with respect to some substrings of the input strings (that all have the same given length):

(r,s)-CloseSubstr	
Instance:	Multi-set $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^{\leq \ell}$, $\ell \in \mathbb{N}$, and integers $d_r, d_s, m \in \mathbb{N}$.
Question:	Is there an $s \in \Sigma^m$ and $S' = \{s'_i \mid s'_i \preceq s_i, 1 \leq i \leq k\} \subseteq \Sigma^m$ with $\mathsf{r}_\mathsf{H}(s, S') \leq d_r$ and $\mathsf{s}_\mathsf{H}(s, S') \leq d_s$?

See Figure 1 for an illustration of these problems. We next introduce some convenient terminology.

By the terms (r)-CloseStr and (s)-CloseStr, we denote the variants of (r,s)-CloseStr, where the only distance bound is d_r or d_s, respectively; we shall also call them the (r)- and (s)-*variant* of CloseStr, the *radius* and *distance sum variant* of CloseStr, or simply the *single-bound* variants if we refer to either of them; the problem (r,s)-CloseStr will sometimes be referred to as the *general variant*. We also use the term CloseStr (i. e., without the prefixes (r,s)-, (r)- or (s)-) whenever we generally refer to all (or any) of these different variants. Analogous terminology applies to the outlier-variant and the substring-variant.

1.2 Parameterised Complexity Theory

We assume the reader to be familiar with the basic concepts of (classical) complexity theory. Next, we shall briefly summarise the fundamentals of parameterised complexity (see also [12,16,8]).

A *parameterised problem* is a decision problem with instances (x, k), where x is the actual input and $k \in \mathbb{N}$ is the *parameter*. By FPT, we denote the class of *fixed-parameter tractable* problems, i. e., problems having an algorithm with running-time $\mathrm{O}(g(k) \cdot f(n))$, for a computable function g and polynomial f, where n is the size of the instance and k is the parameter. In order to argue about fixed-parameter *intractability*, we need the following kind of reductions. A (classical) many-one reduction R from a parameterised problem to another is an *fpt-reduction*, if the parameter of the target problem is bounded in terms of the parameter of the source problem, i. e., there is a recursive function $h\colon \mathbb{N} \to \mathbb{N}$ such that $R(x, k) = (x', k')$ implies $k' \leq h(k)$. Parameterised problems that are hard (with respect to fpt-reductions) for the class $\mathsf{W}[1]$ are not in FPT (under some complexity theoretical assumptions, see [12,16,8] for further details). If a parameterised problem is NP-hard when the parameter is fixed to a constant, then it is not in FPT, unless $\mathsf{NP} = \mathsf{P}$ (thus; providing even stronger evidence for fixed-parameter intractability than $\mathsf{W}[1]$-hardness).

A *kernelisation* for a parameterised problem P is an algorithm that transforms an instance (x, k) of P into a reduced instance (x', k') of P in time polynomial in $|x| + |k|$ such that

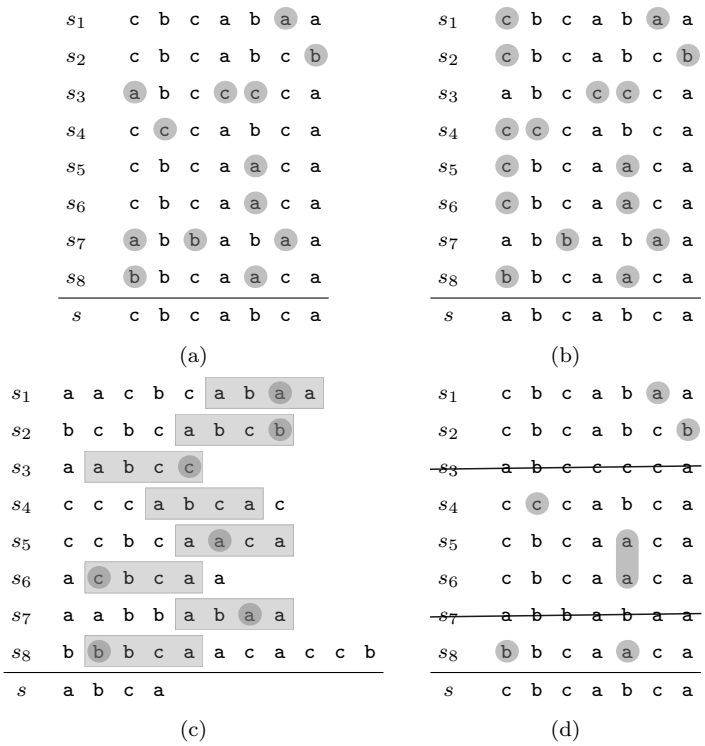

Fig. 1 Illustrations of instances and solution strings for different variants of consensus string problems (mismatches are highlighted by gray circles): (a) shows a CloseStr instance and a solution string with radius 3 and distance sum 13, (b) shows the same instance, but with a solution string with radius 2 and distance sum 16, (c) shows a CloseSubstr instance (with $m = 4$) and a solution string with radius 1 and distance sum 7 (the corresponding substrings are highlighted by gray rectangles), and (d) shows a CloseStr-wo instance (with $t = 2$) and a solution string with radius 2 and distance sum 7 (s_3 and s_7 are declared outliers).

- $|x'| + k \leq g(k)$ for some computable function g,
- (x, k) is a positive instance if and only if (x', k') is a positive instance.

For the sake of convenience, we also say that a parameterised problem has a *kernel* in order to denote that there is a kernelisation as defined above. If the kernlisation is such that the function g is a polynomial, then we say that the problem has a *polynomial kernel*. It is a well-known fact that a parameterised problem is fixed-parameter tractable if and only if it has a kernel. On the other hand, many fixed-parameter tractable problems do not seem to have a polynomial kernel.

Note that all these concepts from parameterised complexity theory naturally extend to problems that are parameterised by several parameters at the same time.

The natural parameters that arise in the context of the consensus string problems defined above are the following (we shall consistently use these parameter names throughout the remainder of the paper):

k	number of input strings
ℓ	length of input strings
d_r	radius bound
d_s	distance sum bound
$\|\Sigma\|$	alphabet size
m	substring length (substring-variant)
t	number of outliers (outlier-variant)
$k-t$	number of inliers (outlier-variant)

For some parameters $p_1, p_2, \ldots$, by (r,s)-CloseStr$(p_1, p_2, \ldots)$ we denote the problem (r,s)-CloseStr parameterised by the parameters $p_1, p_2, \ldots$, e. g., (r,s)-CloseStr$(|\Sigma|, \ell)$ is the problem (r,s)-CloseStr parameterised by the alphabet size and the length of the input strings. In particular, note that this problem is trivially in FPT, since enumerating all strings in Σ^ℓ and checking for each whether it is a solution string is an fpt-algorithm. Moreover, this variant does not seem to have a polynomial kernel (in fact, it can be shown that, under some complexity theoretical assumption, it does not have a polynomial kernel; see Section 5), while (r,s)-CloseStr(k, ℓ) trivially has a polynomial kernel (the original input is of size $\ell \times k$ and therefore a polynomial kernel).

We use analogous terminology for the substring and outlier-variants and also for the single-bound variants, e. g., (r)-CloseStr-wo(d_r, t). Note that we consider parameters t and $k-t$ only for the outlier-variants, parameter m only for the substring-variants, and parameters d_r and d_s only if they exist for the problem, e. g., d_r can only be a parameter for the general variants or the (r)-variants, but not for (s)-variants.

1.3 Known Results

Some of the single-bound variants of the consensus string problems have already been considered in the literature, but under different names. More precisely, the names Closest String and Closest Substring are common in the literature in order to denote the radius variants of CloseStr and CloseSubstr, while the common term Consensus Patterns usually refers to what we have defined as the distance sum variant of CloseSubstr (see, e. g., [18,14,15,17,22]); the term Closest String with Outliers is used in [6] (where the outlier-variant is also introduced for the first time) in order to denote the radius variant of CloseStr-wo.

All the consensus string problems are NP-hard, except the distance sum variant of CloseStr, which is trivial problem (choosing for every column a symbol with majority always yields an optimal solution string). The parameterised complexity (with respect to the above mentioned parameters) of

the radius variants of CloseStr are completely settled (see [17,18]): parameterising by any of the single parameters k, d_r and ℓ yields fixed-parameter tractability, while the problem remains NP-hard if $|\Sigma| = 2$. To the knowledge of the authors, the complexity of the variant (r,s)-CloseStr with both bounds has not yet been investigated in the literature (an exception is [1], where optimising both the radius and the distance sum has been considered for the special case $k = 3$).

With respect to the substring-variants, all parameterisations of the radius variant have been settled, while for the distance sum variant all parameterisations but the single parameter ℓ (or (m, ℓ), which, since we can assume $m \leq \ell$, is the same) have been settled (see [14,15,22]). These results show that, at least for the single-bound variants, CloseSubstr is a much harder problem than CloseStr. More precisely, fixed-parameter tractability of (r)-CloseSubstr can only be achieved if parameterised by ℓ (see [14]) or $(m, |\Sigma|)$ (which is trivial), while all other parameterisations are W[1]-hard. With respect to (s)-CloseSubstr, the only known fixed-parameter tractable cases are with respect to $(d_s, |\Sigma|)$ (see [22]) and $(m, |\Sigma|)$ (which is again trivial), and the case of parameter ℓ is open. However, it has been shown in [26] that if we consider the difference between the length of the input strings and the length of the solution string, i. e., $(\ell - m)$, as a parameter, then adding any of the additional parameters k, d_r, ℓ (which also make (r)-CloseStr fixed-parameter tractable) yields fixed-parameter tractability for (r)-CloseSubstr. On the other hand, for the distance sum variant, the special parameter $(\ell - m)$ only helps if additionally k or d_s is also a parameter, while the case $((\ell - m) = 4, |\Sigma| = 4)$ is even NP-hard and the parameterisation $((\ell - m), m)$ is the same as ℓ, which again leads to the open case mentioned above (see [26] for details). As for CloseStr, the complexity of (r,s)-CloseSubstr has not yet been investigated in the literature.

A parameterised complexity analysis of the radius variant of CloseStr-wo has been started more recently in [6], where it is shown that the problem is fixed-parameter tractable with respect to single parameter d_r and the parameters $(|\Sigma|, k)$, while it is W[1]-hard with respect to $(\ell, d_r, k - t)$. The (s)-variant or the general variant with both bounds has not yet been considered in the literature.

Questions of kernelisations for consensus string problems have been recently investigated in [2].

1.4 Our Contribution

The main contribution of this paper is to initiate the parameterised complexity analysis of the general variants (i. e., with both the radius and the distance bound) of the consensus string problems. In this regards, we are able to completely settle (i. e., proving either fixed-parameter tractability or W[1]-hardness for *all* parameterisations with respect to the parameters defined in Section 1.2) the problems (r,s)-CloseStr and (r,s)-CloseSubstr

(and their single-bound variants). Obviously, as indicated by the discussions of Section 1.3, a large part of this complete picture is already provided in the existing literature, namely almost all the single-bound variants. Moreover, some of the results for the general variants can be concluded with moderate effort from results on the single-bound variants. What required more effort was to close the gap that was left in the literature with respect to (s)-CloseSubstr (see Section 1.3) and to carry over the fixed-parameter tractability from (r)-CloseStr(k) to (r, s)-CloseStr(k).

With respect to the outlier-variant, we are able to settle some more open problems from the literature, but the fixed-parameter tractability of many parameterisations remain unsettled. Our main positive algorithmic result is that (r, s)-CloseStr-wo(d_r, t) (and therefore (r, s)-CloseStr(k)) is fixed-parameter tractable, which is achieved by a non-trivial extension of a branching algorithm from [18] for (r)-CloseStr(d_r). While the general branching strategy is analogous to the one of [18], taking care of the distance sum bound and of the outliers requires some new ideas and leads to a more involved algorithm with a more complicated proof of correctness. While this is interesting from a theoretical point of view, it is particularly interesting in the light of the discussions at the beginning of Section 1 about the practical relevance of parameters d_r and t. In addition to several other simpler fixed-parameter tractability result, we show, as the main negative result with respect to the outlier-variant, that (s)-CloseStr-wo is W[1]-hard if parameterised by $(d_s, \ell, k - t)$. In particular, this shows that unlike CloseStr, for which the radius variant is hard and the distance sum variant is trivial, the outlier-variant resembles the substring-variant where both single-bound variants are hard (note that the general hardness of the (s)-variant of CloseStr-wo was not known).

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)-CloseStr and (r)-CloseSubstr are proved. Some results from [2] about the single-bound variants directly carry over to the general variants; our main contribution is a cross-composition from (r)-CloseStr into (r)-CloseStr-wo, which rules out a polynomial kernel for (r, s)-CloseStr-wo($d_r, d_s, \ell, (k - t), |\Sigma|$).

1.5 Organisation of the Paper

In Section 2, we settle all parameterisations of the problem (r, s)-CloseStr. Then, in Section 3, we consider the general as well as the single-bound variants of the outlier-variant; this section also contains our main result, i. e., the branching algorithm for (r, s)-CloseStr-wo(d_r, t). The substring-variant will then be investigated in Section 4 and questions about kernelisations will be discussed in Section 5. Finally, in Section 6, we summarise and discuss our results and mention the most interesting open problems.

2 Closest String with Radius and Distance Sum Bound

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, $\mathsf{s}_\mathsf{H}(s,S) = \min\{\mathsf{s}_\mathsf{H}(s',S) \mid s' \in \Sigma^\ell\}$ 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 $\mathsf{r}_\mathsf{H}(s,S) = \min\{\mathsf{r}_\mathsf{H}(s',S) \mid s' \in \Sigma^\ell\}$ or $\mathsf{s}_\mathsf{H}(s,S) = \min\{\mathsf{s}_\mathsf{H}(s',S) \mid s' \in \Sigma^\ell\}$, respectively.

It is a well-known fact that (r)-CloseStr allows fpt-algorithms for any of the single parameters k, d_r or ℓ, and it is still NP-hard for $|\Sigma| = 2$ (see [18]). While the latter hardness result trivially carries over to (r,s)-CloseStr (by setting $d_s = k\,d_r$), we have to modify the fpt-algorithms for extending the fixed-parameter tractability results to (r,s)-CloseStr.

We start with parameter k, for which we can extend the ILP-approach that is used in [18] to show (r)-CloseStr(k) $\in$ FPT. Before we can formally do this, we need a few more definitions.

We say that $S \subseteq \Sigma^\ell$ is *normalised*, if $\Sigma = \{a_1, a_2, \ldots, a_k\}$, every column of S contains the symbols $\{a_1, a_2, \ldots, a_p\}$, for some p, $1 \leq p \leq k$, and the first occurrence of a_i, $1 \leq i \leq p-1$, occurs before the first occurrence of a_{i+1}. If S is normalised, any two isomorphic columns are equal (i. e., if two columns are not identical, then it is not possible to obtain one from the other by bijective renaming of the symbols). It can be easily seen that any (r,s)-CloseStr instance can be transformed into an equivalent one with normalised S (see [18]). For both normalised and non-normalised S, we use the term *column-types* to denote the different forms of columns, rather than the collection of all columns, i. e., the set of column types of S is $\{c \in \Sigma^k \mid c \text{ occurs as a column in } S\}$.

Theorem 1 (r,s)-CloseStr(k) $\in$ FPT.

Proof We extend the ILP-approach that has been used in [18] to show that (r)-CloseStr(k) $\in$ FPT. Let $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^\ell$ with columns t_i, $1 \leq i \leq \ell$, and $d_r, d_s \in \mathbb{N}$ be a (r,s)-CloseStr instance. Let S be normalised, let T be the set of column types and, for every $t \in T$, let $\psi_t = |\{i \mid 1 \leq i \leq \ell, t_i \text{ has type } t\}|$, i. e., the number of columns of type t. Note that $|T| \leq B(k) \leq k!$ (where $B(k)$ is the Bell number). We extend the ILP from [18] that has a solution if and only if the (r,s)-CloseStr instance has a solution. For every column type t and every $a \in \Sigma$, the variable $x_{t,a}$ stands for the number $|\{i \mid 1 \leq i \leq \ell, t_i \text{ has type } t, s[i] = a\}|$, where s is the hypothetical solution string. Intuitively speaking, the number $x_{t,a}$ says how often a column of type t is *paired* with an occurence of the symbol a in the solution string. The equations of the ILP are as follows:

$x_{t,a} \geq 0$, $t \in T$, $a \in \Sigma$	(the number of pairings is non-negative)
$\sum_{a \in \Sigma} x_{t,a} = \psi_t$, $t \in T$	(type t is paired as often as it occurs in S)
$\sum_{t \in T} \sum_{a \in \Sigma \setminus \{t[i]\}} x_{t,a} \leq d_r$, $1 \leq i \leq k$	(mismatches caused by each string bounded by d_r)
$\sum_{i=1}^{k} \sum_{t \in T} \sum_{a \in \Sigma \setminus \{t[i]\}} x_{t,a} \leq d_s$	(total number of mismatches bounded by d_s)

Since we can assume $|\Sigma| \leq k$, we have $k \cdot B(k)$ variables and the result follows from the fact that ILP parameterised by the number of variables is in FPT (see [20]). □

Next, we consider the parameter d_r. For the (r)-variant of CloseStr, the fixed-parameter tractability with respect to d_r is shown in [18] by a branching algorithm, which proved itself as rather versatile: it has successfully been extended in [6] to (r)-CloseStr-wo(d_r, t) and in [26] to (r)-CloseSubstr$(d_r, (\ell - m))$. We shall next briefly sketch this algorithm from [18].

Let $S = \{s_1, s_2, \ldots, s_k\} \subseteq \Sigma^\ell$, $d_r \in \mathbb{N}$ be an (r)-CloseStr instance and assume that there is a solution string s. If $s' = s_1$ is not a solution string, i. e., $\mathsf{r}_\mathsf{H}(s', S) \geq d_r + 1$, then there is some input string s_i with $\mathsf{d}_\mathsf{H}(s', s_i) \geq d_r + 1$. Moreover, every set $\{j \mid 1 \leq j \leq \ell, s_i[j] \neq s'[j]\}$ of cardinality $d_r + 1$ must contain at least one position j, such that $s_i[j] = s[j]$ (otherwise $\mathsf{d}_\mathsf{H}(s, s_i) \geq d_r + 1$, which is a contradiction). Obviously, in order to transform s' to the solution string, this position j of s' must be changed to the one of s_i. Consequently, arbitrarily choosing a set $\{j \mid 1 \leq j \leq \ell, s_i[j] \neq s'[j]\}$ of cardinality $d_r + 1$, branching over all these $d_r + 1$ positions and changing them in s' to the corresponding positions in s_i yields a branching algorithm for (r)-CloseStr (note that a branching depth of d_r is sufficient, since it must be possible to reach the solution string by changing at most d_r positions of s_1).

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)-CloseStr(d_r). In fact, we prove in Theorem 5 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)-CloseStr-wo(d_r, t). Since Theorem 2 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)-CloseStr(d_r) (and refer to Theorem 5 for a formal proof of correctness).

The main problem in extending the algorithm to the case of an additional bound d_s on the distance sum can be described as follows. If we start with some input string as the first candidate string and then carry out the branching as sketched above, then we have no guarantee that the resulting solution satisfies the distance sum bound d_s. On the other hand, if we start with some other candidate string that is somehow tailored to the distance sum bound, we lose

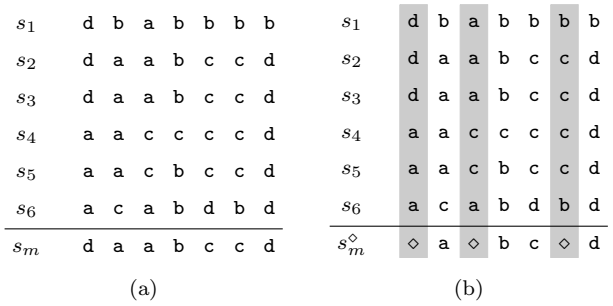

Fig. 2 (a) A matrix of strings and its majority string. (b) The same matrix of strings, its refined majority string and the disputed columns highlighted in grey.

the guarantee that a solution can be reached by a number of changes that only depends on d_r (which is trivially the case if we start with an input strings).

An obvious choice for a first candidate string for a branching algorithm that also takes the distance sum bound into consideration is a majority string (see Figure 2), since this is the "best" string with respect to the distance sum bound. Starting with this string, we can apply the same branching strategy in order to change it step by step into a string that satisfies the radius bound. However, this can only result in a valid fpt-algorithm (with respect to parameter d_r), if the branching depth can be bounded by a function in d_r, which is done by the following lemma (that we also need later for the proof of correctness of the algorithm for (r,s)-CloseStr-wo(d_r,t)).

Lemma 1 *Let $S \subseteq \Sigma^\ell$, $s \in \Sigma^\ell$ such that $\mathsf{r}_\mathsf{H}(s,S) \leq d_r$, and let s_m be a majority string for S. Then $\mathsf{d}_\mathsf{H}(s_m,s) \leq 2d_r$.*

Proof Let $\widehat{d} = \mathsf{d}_\mathsf{H}(s_m,s)$ and $k = |S|$. Let i, $1 \leq i \leq \ell$, with $s_m[i] \neq s[i]$ and let p be the number of occurrences of $s_m[i]$ in the i^{th} column of S. Obviously, if $p \geq \frac{k}{2}$, then $s[i]$ matches at most $k-p \leq \frac{k}{2}$ entries of the i^{th} column of S, and if $p < \frac{k}{2}$, then $s[i]$ matches at most p entries of the i^{th} column of S. Consequently, for every i, $1 \leq i \leq \ell$, if $s_m[i] \neq s[i]$, then $|\{j \mid 1 \leq j \leq k, s[i] \neq s_j[i]\}| \geq k/2$. Summing over all i, $1 \leq i \leq \ell$, this implies $\mathsf{s}_\mathsf{H}(s,S) \geq \widehat{d}\,\frac{k}{2}$. Since $\mathsf{r}_\mathsf{H}(s,S) \leq d_r$, we have $\mathsf{s}_\mathsf{H}(s,S) \leq kd_r$. Hence, $kd_r \geq \widehat{d}\,\frac{k}{2}$, that is $\widehat{d} \leq 2d_r$. □

A branching algorithm for (r,s)-CloseStr(d_r) can now be sketched as follows. We start with a majority string s_m and apply the branching as described above. The branching depth is bounded by $2d_r$ (due to Lemma 1) and we cut any branch where the distance sum goes beyond the threshold d_s. If there exists a solution that satisfies the d_r bound, then there must be a path in the branching tree in which all changes of single positions are necessary, and, since we started with a majority string, all unchanged positions have a symbol that causes the fewest additional mismatches (for a formal proof of correctness, we refer to Theorem 5).

k	d_r	d_s	$\lvert\Sigma\rvert$	ℓ	Result	Note/Ref.
p	–	–	–	–	FPT	Thm. 1
–	**p**	–	–	–	FPT	Thm. 2
–	–	**p**	–	–	FPT	Cor. 1
–	–	–	2	–	NP-hard	from (r)-variant [17]
–	–	–	–	**p**	FPT	Cor. 1

Table 1 Results for (r,s)-CloseStr.

Theorem 2 (r,s)-CloseStr$(d_r) \in$ FPT.

It only remains to take a look at the parameters ℓ and d_s, for which containment in FPT follows easily from known results. More precisely, 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 $\mathsf{s}_\mathsf{H}(s_i, S) \geq \ell$ for every $s \in \Sigma^\ell$; thus, we can assume $\ell \leq d_s$. Consequently, we obtain the following corollary:

Corollary 1

- (r,s)-CloseStr$(\ell) \in$ FPT.
- (r,s)-CloseStr$(d_s) \in$ FPT.

This completely settles the parameterised complexity of (r,s)-CloseStr with respect to parameters k, d_r, d_s, $|\Sigma|$ and ℓ (see Table 1 for an overview of the results). Recall that the (r)-variant is already settled, while the (s)-variant is trivial.

3 The Outlier-Variant

In this section, we investigate (r,s)-CloseStr-wo and their (r)- and (s)-variants. We first prove several fixed-parameter tractability results for the general variant and we consider the (r)- and (s)-variants later on.

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

Theorem 3 (r,s)-CloseStr-wo$(k) \in$ FPT.

Proof Let $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^\ell$, $\ell \in \mathbb{N}$, and integers $d_r, d_s, t \in \mathbb{N}$ be an (r,s)-CloseStr-wo instance. We note that $s \in \Sigma^\ell$ and $S' \subseteq \{s_1, s_2, \ldots, s_k\}$ with $|S'| = k - t$ is a solution of this instance if and only if s is a solution for the (r,s)-CloseStr instance S', d_r, d_s. Consequently, we can solve the (r,s)-CloseStr-wo instance by solving all the $\binom{k}{k-t}$ many (r,s)-CloseStr instances (S_1, d_r, d_s), (S_2, d_r, d_s), …, (S_m, d_r, d_s), where $m = \binom{k}{k-t}$ and the S_i, $1 \leq i \leq m$, are all subsets of $\{s_1, s_2, \ldots, s_k\}$ with cardinality $k - t$. Since (r,s)-CloseStr$(k) \in$ FPT (see Theorem 1), this yields an fpt-algorithm for (r,s)-CloseStr-wo(k). □

We next show that if the number $k - t$ of inliers exceeds d_s, then an (r,s)-CloseStr-wo instance becomes easily solvable; thus, $k-t$ can be bounded by d_s. If in addition t is also a parameter, this implies that k is bounded, so fixed-parameter tractability follows from Theorem 3.

Theorem 4 (r,s)-CloseStr-wo$(d_s,t) \in \mathsf{FPT}$.

Proof Let $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^\ell$ and $d_r, d_s, t \in \mathbb{N}$ be an (r,s)-CloseStr-wo instance. If $d_s < k - t$, then every solution string s must satisfy $s = s_i$, for some i, $1 \leq i \leq k$. Moreover, if s_i is a solution string, then it is a solution string with respect to the set $S' \subseteq S$ containing the $k - t$ strings with the least Hamming-distance from s_i. Consequently, we can compute a solution in polynomial-time. If, on the other hand, $k - t \leq d_s$, then $k - t$ and t can be considered parameters; thus, k is a parameter and the result follows from (r,s)-CloseStr-wo$(k) \in \mathsf{FPT}$ (see Theorem 3). □

We now turn to the parameter d_r. As briefly mentioned in Section 2, the algorithm introduced in [18] to prove (r)-CloseStr$(d_r) \in \mathsf{FPT}$ has been extended in [6] 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)-CloseStr-wo(d_r,t). Moreover, we already sketched how the algorithm from [18] could be extended to (r,s)-CloseStr(d_r). Next, we combine these two approaches in a non-trivial way in order to obtain an fpt-algorithm for (r,s)-CloseStr-wo(d_r,t) (as explained in Section 2, this also provides a formal proof of Theorem 2).

The main problem about the general approach sketched in Section 2, i. e., starting with the majority string as a first candidate, is that whether a certain symbol is a majority symbol in a column depends on the choice of outliers. For example, both `a` and `d` are majority symbols of the first column of the matrix of Figure 2(a), but if $t = 2$ and s_1 and s_2 are declared as outliers, then, in case that the first symbol is not changed by the branching modifications, it is possible that `d` was a bad choice, since it causes more mismatches compared to `a` (with respect to the matrix from which the outliers are removed). In order to deal with this issue, we refine the concept of a majority string and tailor it to the outlier-variant.

Let (S, d_s, d_r, t) be an instance of (r,s)-CloseStr-wo(d_r,t) and let (S^*, s^*) be a solution for this instance. We say that a character x is *frequent in column* i if it has at least as many occurrences as a 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 $\diamond \notin \Sigma$ be a new symbol and let s_m be the majority string of S. The *refined majority* string $s_m^\diamond \in (\Sigma \cup \{\diamond\})^*$ (with respect to S and t) is defined by $s_m^\diamond[i] = s_m[i]$ if i is not a disputed column and $s_m^\diamond[i] = \diamond$ if i is a disputed column, for every i, $1 \leq i \leq \ell$ (see Figure 2(b) for an example).

More generally, 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 $\diamond$). In particular, note that the refined majority string is by definition a lower bound. A *completion for* $S' \subseteq S$ 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' (for example, in Figure 2(b), a possible completion for $\{s_3, s_4, s_5, s_6\}$ of the refined majority string $s_m^\diamond$ would be `aacbccd`).

The following lemma states that the number of disputed columns of S can be bounded in terms of d_r, which shall be a central building block of the following branching algorithm.

Lemma 2 *Let (S, d_s, d_r, t) be a positive instance of (r, s)-CloseStr-wo(d_r, t) with D disputed columns. If $k \geq 5t$, then $D \leq 4d_r$.*

Proof Let (S^*, s^*) be a solution for the instance (S, d_s, d_r, t). 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 are now ready to present the fpt-algorithm for (r, s)-CloseStr-wo(d_r, t) and prove its correctness (an illustration of the algorithm is provided by Figure 3).

ALGORITHM 1: Solve CSO

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 ∇

```
if t = 0 then
    s'' = completion of s' in S';
    if s_H(s'', S') ≤ d_s, and r_H(s'', S') ≤ d_r then return (S', s'');
    if d = 0 then return ∇;
Let s_j ∈ S' be such that d_H(s', s_j) is maximal;
if t > 0 then
    sol = Solve CSO(S' \ {s_j}, t − 1, s', d);
    if sol ≠ ∇ then return sol;
if d > 0 then
    Let I ⊆ {1, ..., ℓ} contain d_r + 1 indices i s. t. s'[i] ≠ s_j[i] (or all indices if
      d_H(s_j, s') ≤ d_r);
    for i ∈ I do
        s'' = s', s''[i] = s_j[i];
        sol = Solve CSO(S', t, s'', d − 1);
        if sol ≠ ∇ then return sol;
return ∇;
```

Theorem 5 (r,s)-CloseStr-wo$(d_r,t) \in \mathsf{FPT}$.

Proof Let (S,d_s,d_r,t) be an instance of (r,s)-CloseStr-wo(d_r,t). We assume that $k \geq 5t$, since for all other instances, k can be considered as a parameter and therefore they can be solved in fpt-time according to Theorem 3.

The algorithm is presented as Algorithm 1 and in the following, we denote it by Solve CSO. The algorithm is formulated in a recursive way and in any recursive call, it receives as input a set S' of the remaining input strings (i. e., the initial input strings with some outliers removed), a number t' that denotes how many outlier-choices are left, a current candidate string s' (over $\Sigma \cup \{\diamond\}$) and a number d denoting how many branching steps are left.

We first show that any recursive call to Solve CSO(S',t,s',d) returns after fpt-time with respect to d_r, d and t.

Claim 1: Any call to the algorithm Solve CSO(S',t',s',d) always returns after time $O^*((d_r+1)^d 2^{d+t'})$.

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 $O^*((d_r+1)(d_r+1)^{d-1}2^{d+t'-1} + (d_r+1)^d 2^{d+t'-1}) = O^*((d_r+1)^d 2^{d+t'})$. (Claim 1) □

In the following, we say that 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^* (in the sense defined above). 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 CSO 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 CSO(S',t',s',d). Assume it is valid, with witness (S^*,s^*). We prove the statement of the claim by considering several cases:

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.

This concludes the proof of the claim. (Claim 2) □

In Particular, Claim 2 implies that any valid call to Solve CSO 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.

Next, we show that starting the algorithm with parameters $S' = S$, $t' = t$, $s' = s^\diamond_m$ and $d = 2d_r + D$ (where D is the number of disputed columns) is a valid call.

Claim 3: Solve CSO$(S, t, s^\diamond_m, 2d_r + D)$ is a valid call.

Proof of Claim 3: Consider a solution (S^*, s^*). We need to check whether $\mathsf{d_H}(s^*, s^\diamond_m) \leq 2d_r + D$, and whether $s^\diamond_m$ is a lower bound of s^*. The latter follows by definition and has already been observed above. String s^* can be seen as a solution of (r, s)-CloseStr over S^*, d_r, d_s, thus, Lemma 1 implies that the distance between s^* and the majority string of S^* is at most $2d_r$. Hence there are at most $2d_r$ mismatches between $s^\diamond_m$ 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. (Claim 3) □

Finally, we note that, according to Lemma 2 (recall that we initially made the assumption $k \geq 5t$), $D \leq 4d_r$. Consequently, the above claims imply that calling Solve CSO with parameters $S, t, s^\diamond_m, 6d_r$ solves the (r, s)-CloseStr-wo instance in time $O^*((d_r + 1)^{6d_r} 2^{6d_r + t})$. □

Next, we consider the (r)- and (s)-variants of CloseStr-wo. With respect to (r)-CloseStr-wo, the fixed-parameter tractability with respect to k and

Input:
$d_r = 5$
$d_s = 14$
$t = 2$

$D = 10$

$s_1 =$ dbaddcbcdbbdbb
$s_2 =$ daaaacbcdccdbd
$s_3 =$ daaddabcaccdbd
$s_4 =$ aacdaccdcccabd
$s_5 =$ aacdaabdaccadd
$s_6 =$ 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''

Fig. 3 Example for Algorithm 1 on an instance of (r, s)-CloseStr-wo. The shown steps correspond to one branch that yields a correct solution. The algorithm starts with the refined majority string. 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 $\mathsf{s_H}(s'', S') = 14$.

$(|\Sigma|, d_r, k - t)$ are reported as open problems in [6]. Since Theorem 3 also applies to (r)-CloseStr-wo (by setting $d_s = kd_r$), the only open cases left for the (r)-variant are the following:

Question 1 What is the fixed-parameter tractability of (r)-CloseStr-wo with respect to $(|\Sigma|, k - t)$, $(|\Sigma|, d_r)$ and $(|\Sigma|, d_r, k - t)$?

We now turn to the (s)-variant of CloseStr-wo (which, to the knowledge of the authors, has not yet been considered in the literature). We recall that the (r)-variant of CloseStr is hard, while its (s)-variant is trivial. On the other hand, for the substring-variant we have a quite different situation, since both single-bound variants of CloseSubstr are hard. We shall see next that the outlier-variant resembles the substring-variant in this regard, i. e., both single-bound variants are hard (for the (r)-variant this is known [6], while for the (s)-variant this is established by the following theorem).

We use a reduction from the problem Multi-Coloured Clique (which is $\mathsf{W}[1]$-hard, see [11]). The problem Multi-Coloured Clique is identical to the standard parameterisation of Clique (i. e., we want to find a clique of a given size k_c, and k_c is also the parameter), but the input graph $G = (V, E)$ has a partition $V = V_1 \cup \ldots \cup V_{k_c}$, such that every V_i, $1 \leq i \leq k_c$, is an independent set (we denote the parameter by k_c to avoid confusion with the number of input strings k).

Let $G = (V_1 \cup \ldots \cup V_{k_c}, E)$ be a Multi-Coloured Clique instance. Without loss of generality, we assume that, for some $q \in \mathbb{N}$, $V_i = \{v_{i,1}, v_{i,2}, \ldots v_{i,q}\}$, $1 \leq i \leq k_c$, i. e., each vertex has an index depending on its colour-class and its

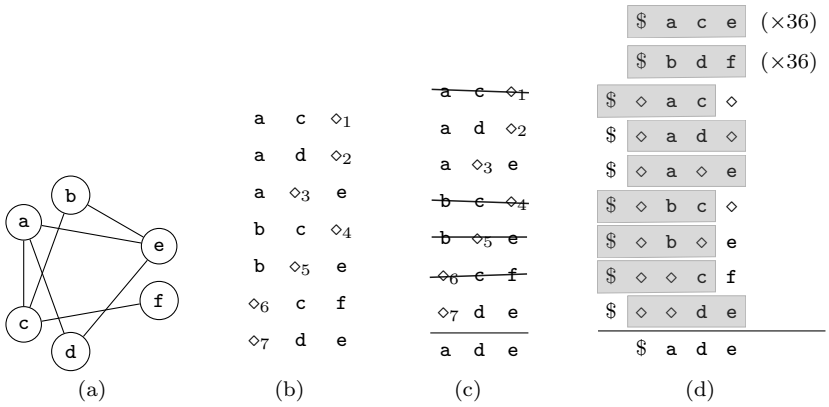

Fig. 4 Illustration of the reductions of Theorems 6 and 7: (a) is an example instance of Multi-Coloured Clique (with $k_c = 3$, $q = 2$ and colour classes $\{\mathtt{a},\mathtt{b}\}$, $\{\mathtt{c},\mathtt{d}\}$ and $\{\mathtt{e},\mathtt{f}\}$), (b) is the ($\mathsf{s}$)-CloseStr-wo instance obtained from the graph by the reduction of Theorem 6 (where $\Gamma = \{\diamond_i \mid 1 \leq i \leq 7\}$, $t = |E| - \binom{k_c}{2} = 7-3 = 4$, $d_s = \binom{k_c}{2}(k_c-2) = 3$), (c) is the same instance, but with the appropriate outliers crossed out and a solution string representing a k_c-clique, and (d) shows the (s)-CloseSubstr instance obtained from the graph by the reduction of Theorem 7 (where $m = k_c+1 = 4$, $d_s = (|E|(k_c+2)+1)qk_c + |E|k_c - \frac{k_c(k_c-1)}{2} = 234$; note that by definition, each of the the first $q = 2$ strings, i. e., the strings $\mathcal{V}_j$, is repeated $|E|(k_c+2)+1 = 36$ times), the appropriate substrings of length k_c+1 highlighted in grey and a solution string representing a k_c-clique.

rank within its colour-class. Let $\Sigma = V \cup \Gamma$, where Γ is some alphabet with $|\Gamma| = |E|(k_c-2)$. For every $e = (v_{i,j}, v_{i',j'}) \in E$, let $s_e \in \Sigma^{k_c}$ 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 $S = \{s_e \mid e \in E\}$, $t = |E| - \binom{k_c}{2}$ and $d_s = \binom{k_c}{2}(k_c-2)$. See Figure 4(a), (b) and (c) for an illustration of the reduction and the following proof.

Theorem 6 (s)-CloseStr-wo$(d_s, \ell, k-t)$ *is* $\mathsf{W}[1]$*-hard.*

Proof We prove that the reduction defined above is a parameterised reduction from Multi-Coloured Clique to (s)-CloseStr-wo$(d_s, \ell, k-t)$. To this end, let $G = (V_1 \cup \ldots \cup V_{k_c}, E)$ be a Multi-Coloured Clique instance and let S, t and d_s be obtained from G by the reduction. We first note that $\ell = k_c$, $d_s = \binom{k_c}{2}(k_c-2)$ and $k - t = \binom{k_c}{2}$, which shows that the parameters of the (s)-CloseStr-wo instance are all bounded by a function in k_c. It remains to prove that G has a clique of size k_c if and only if the (s)-CloseStr-wo instance has a solution.

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

Now let $s \in \Sigma^{k_c}$ and $S' \subseteq S$ with $|S'| = \binom{k_c}{2}$ be a solution for the (s)-CloseStr-wo instance S, t, d_s. If, for some $s'_1 \in S'$, $\mathsf{d_H}(s'_1, s) \geq k_c - 1$, then there is an $s'_2 \in S'$ with $\mathsf{d_H}(s'_2, s) \leq k_c - 3$. Thus, for some i, $1 \leq i \leq k_c$, $s[i] = s'_2[i]$ 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 $\mathsf{d_H}(s'_2, s) \leq k_c - 2$. By repeating such operations, we can transform s such that $\mathsf{d_H}(s', s) \leq k_c - 2$, $s' \in S'$. Next, assume that, for some i, $1 \leq i \leq k_c$, there is an $S'' \subseteq S'$ with $|S''| = k_c$ and, for every $s' \in S''$, $s[i] = s'[i]$. Since $\mathsf{d_H}(s', s) \leq k_c - 2$ for every $s' \in S''$, 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', $1 \leq i' \leq k_c$, and $i' \neq i$, which, by the structure of the strings of S, is a contradiction. Consequently, for every i, $1 \leq i \leq k_c$, s matches with at most $k_c - 1$ strings from S' at position i. Since there are at least $2\binom{k_c}{2} = k_c(k_c - 1)$ matches, we conclude that, for every i, $1 \leq i \leq k_c$, $s[i]$ matches exactly $k_c - 1$ times with the i^{th} position of a string from S'. Hence, $s[i] \in V_i$, $1 \leq i \leq k_c$, i. e., $s = v_{1,r_1} v_{2,r_2} \dots v_{k_c,r_{k_c}}$, for some $r_j \in \{1, 2, \dots, q\}$, $1 \leq j \leq k_c$. Let $K = \{v_{1,r_1}, v_{2,r_2}, \dots, v_{k_c,r_{k_c}}\}$. For every $s' \in S'$, by definition of the strings s_e, we have $\mathsf{d_H}(s, s') \geq k_c - 2$, combining with the lower-bound proved ealier, we conclude $\mathsf{d_H}(s, s') = k_c - 2$, for every $s' \in S$. Now let $e = (v_{i,j}, v_{i',j'}) \in E$ be such that $s_e \in S'$. From $\mathsf{d_H}(s, s_e) = k_c - 2$ its follows that $s[i] = v_{i,j}$ and $s[i'] = v_{i',j'}$, which implies $e \subseteq K$. Since $|S| = \binom{k_c}{2}$, there are $\binom{k_c}{2}$ edges connecting vertices from K; thus, K is a clique. □

We note that the reduction used in the proof of Theorem 6 can also be used in order to obtain a simpler proof for the W[1]-hardness of the problem (r)-CloseStr-wo$(d_r, \ell, k - t)$ shown in [6]. More precisely, this is achieved by simply setting $d_r = k_c - 2$ instead of $d_s = \binom{k_c}{2}(k_c - 2)$. On the other hand, the reduction used in [6] to show the W[1]-hardness of (r)-CloseStr-wo$(d_r, \ell, k - t)$ (which is from the Clique problem instead of Multi-Coloured Clique) does not work for the (s)-variant (more precisely, the reduction produces an instance the distance sum of which is not bounded in terms of the clique size).

These results, together with the known results from [6], settle a large number of parameterisations of the different outlier-variants of CloseStr. However, many cases are still open; see Table 2 for a summary. This is due to the fact that, unlike for CloseStr, not even the single-bound variants are completely settled, and there are more parameters to be considered. We shall discuss the most interesting respective open cases in Section 6.

4 The Substring-Variant

In this section, we consider the substring-variants of CloseStr, i. e., the different variants of the problem CloseSubstr. Similar to CloseStr, all parameterisations of the (r)-variant, and *almost* all parameterisations of the (s)-variant are already settled in the literature (while the variant with both bounds has not yet been considered in the literature). As has been done in

k	t	$\|\Sigma\|$	ℓ	d_r	d_s	$k-t$	Result	Note/Ref.
p	–	–	–	–	–	–	FPT	Thm. 3, Open Prob. in [6]
–	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. 5, and [6] for d_r-var.
–	**p**	–	–	–	**p**	–	FPT	Thm. 4
–	**p**	–	–	–	–	**p**	FPT	$k = t + (k-t)$
–	–	**p**	**p**	–	–	–	FPT	trivial
–	–	**p**	–	$\star$	$\star$	$\star$	Open	param. $\|\Sigma\|$ and some of $d_r, d_s, k-t$
–	–	–	**p**	**p**	**p**	**p**	W[1]-hard	even for d_r-var. [6] and d_s-var. (Thm. 6)

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

Section 2 for (r,s)-CloseStr, we are able to classify *all* parameterisations of (r,s)-CloseSubstr (and its single-bound variants) with respect to the parameters ℓ, k, m, d_r, d_s and $|\Sigma|$ into either fixed-parameter tractable or W[1]-hard (thus, also solving the case left open in the literature with respect to (s)-CloseSubstr).

With respect to the (s)-variant, the status of (s)-CloseSubstr(ℓ) is unknown, which is mentioned as open problem in [26]. We shall first close this gap by proving this parameterisation to be W[1]-hard.

We devise a reduction from Multi-Coloured Clique. Let $G = (V_1 \cup \ldots \cup V_{k_c}, E)$ be a Multi-Coloured Clique instance. We again assume that, for some $q \in \mathbb{N}$, $V_i = \{v_{i,1}, v_{i,2}, \ldots v_{i,q}\}$, $1 \leq i \leq k_c$, 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 $\mathcal{V}_j = \$v_{1,j}v_{2,j}\ldots v_{k_c,j}$. For every edge $e = (v_{i,j}, v_{i',j'})$ with $i < i'$, we define a string $\mathcal{E}_e = \$\diamond^i v_{i,j} \diamond^{i'-i-1} v_{i',j'} \diamond^{k_c-i'-1}$. Note that $\mathcal{E}_e = \$\diamond\mathcal{E}'_e$, where $|\mathcal{E}'_e| = k_c$, 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)-CloseSubstr instance is now defined as follows. Let S contain $N = |E|(k_c+2)+1$ occurrences of each $\mathcal{V}_j$, $1 \leq j \leq q$, and one occurrence of each $\mathcal{E}_e$, $e \in E$, and let $m = k_c + 1$. See Figure 4(a) and (d) for an illustration of the reduction.

For proving the correctness of the reduction, we first 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 3 *If $s \in \Sigma^{k+1}$ is distance sum optimal with respect to S, then $s \in \{\$\} \cdot V_1 \cdot V_2 \cdot \ldots \cdot V_k$.*

Proof We first note that a string $s \in \Sigma^{k_c+1}$ is a majority string of $\{\mathcal{V}_j \mid 1 \leq j \leq q\}$ if and only if $s \in \{\$\} \cdot V_1 \cdot V_2 \cdot \ldots \cdot V_{k_c}$. More precisely, the first column of $\{\mathcal{V}_j \mid 1 \leq j \leq q\}$ is $\$^q$ and, for every i, $2 \leq i \leq k_c + 1$, the i^{th} column of $\{\mathcal{V}_j \mid 1 \leq j \leq q\}$ contains every vertex from V_i exactly once, so every $v \in V_i$ has majority in column i.

Now let $s \in \Sigma^{k_c+1}$ be such that s is not a majority string for $\{\mathcal{V}_j \mid 1 \leq j \leq q\}$, which implies that s is not distance sum optimal with respect to

$\{\mathcal{V}_j \mid 1 \leq j \leq q\}$. Since every $\mathcal{V}_j$, $1 \leq j \leq q$, has $N = |E|(k_c+2)+1$ occurrences in S, any majority string for $\{\mathcal{V}_j \mid 1 \leq j \leq q\}$, in comparison with s, causes at least $|E|(k_c+2)+1$ fewer mismatches with respect to all occurrences of the strings $\{\mathcal{V}_j \mid 1 \leq j \leq q\}$. Since the total number of symbols of the remaining strings in $\{\mathcal{E}_e \mid e \in E\}$ is $|E|(k_c+2)$, this cannot be compensated, which means that a majority string for $\{\mathcal{V}_j \mid 1 \leq j \leq q\}$ has lower distance sum than s and therefore s is not distance sum optimal with respect to S and m. □

Now let s be distance sum optimal with respect to S and m. From Lemma 3, we can conclude that $s = \$v_{1,r_1}v_{2,r_2}\dots v_{k_c,r_{k_c}}$, for some $r_j \in \{1,2,\dots,q\}$, $1 \leq j \leq k_c$. Let K be the corresponding set of vertices induced by s, i. e., $K = \{v_{1,r_1}, v_{2,r_2}, \dots, v_{k_c,r_{k_c}}\}$.

Lemma 4 *Let $e \in E$. The optimal distance between s and a length-(k_c+1) substring of $\mathcal{E}_e$ is $k_c - 1$ if $e \subseteq K$, and k_c otherwise.*

Proof We first recall that $s[1] = \$$, $s[i+1] \in V_i$, $1 \leq i \leq k_c$, and s has no occurrences of $\diamond$. Now let $e = (v_{i,j}, v_{i',j'})$. The string $\mathcal{E}_e$ has two length-m substrings: $\mathcal{E}_e[1..k_c+1]$ and $\mathcal{E}_e[2..k_c+2]$. The string $\mathcal{E}_e[1..k_c+1]$ starts with \$, has $v_{i,j}$ and $v_{i',j'}$ at positions $i+2$ and $i'+2$ (if $i'+2 \leq k_c+1$), respectively, and $\diamond$ at every other position. Consequently, due to the structure of s, the strings s and $\mathcal{E}_e[1..k_c+1]$ only match at position 1, which implies $\mathsf{d_H}(s, \mathcal{E}_e[1..k_c+1]) = k_c$.

On the other hand, the string $\mathcal{E}_e[2..k_c+2]$ starts with $\diamond$, has $v_{i,j}$ and $v_{i',j'}$ at positions $i+1$ and $i'+1$, respectively, and $\diamond$ at every other position. Consequently, the only possible matching positions between s and $\mathcal{E}_e[2..k_c+2]$ are $i+1$ and $i'+1$. We note that both of these positions match if and only if $s[i+1] = v_{i,j}$ and $s[i'+1] = v_{i',j'}$, which means that $\mathsf{d_H}(s, \mathcal{E}_e[2..k_c+2]) = k_c-1$ if and only if $e \subseteq K$. If only one or none of these positions match, then $\mathsf{d_H}(s, \mathcal{E}_e[2..k_c+2]) \geq k_c$. □

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

Theorem 7 *(s)-CloseSubstr(ℓ, m) is* W[1]*-hard.*

Proof We first note that $\ell = k_c+2$ and $m = k_c+1$; thus, the parameters are bounded by a function in k_c.

Let $s \in \Sigma^{k_c+1}$ 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 Nqk_c. According to Lemma 4, for every $e \in E$, the optimal distance sum between s and the respective substring of $\mathcal{E}_e$ is k_c-1 if $e \subseteq K$, and k_c otherwise. Hence, the total distance sum from s to the respective substrings of $\mathcal{E}_e$, $e \in E$, is $|E|k_c - r$, where $r = \{e \in E \mid e \subseteq K\}$, and the total distance sum between s and S is therefore $Nqk_c + |E|k_c - r$. This implies that the distance sum between s and S is $Nqk_c + |E|k_c - \frac{k_c(k_c-1)}{2}$ if and only if $r = \frac{k_c(k_c-1)}{2}$ if and only if K is a clique of size k_c. Consequently, the above reduction, with the addition of $d_s = Nqk_c +$

ℓ	k	m	d_s	$\lvert\Sigma\rvert$	Result	Reference
–	–	**p**	–	**p**	FPT	trivial
p	–	–	–	**p**	FPT	[26]
p	**p**	–	–	–	FPT	[26]
p	–	–	**p**	–	FPT	[26]
–	–	–	**p**	**p**	FPT	[22]
–	**p**	–	–	**2**	W[1]-hard	[15]
–	**p**	**p**	**p**	–	W[1]-hard	[15]
p	–	**p**	–	–	W[1]-hard	Thm. 7

Table 3 Results for (s)-CloseSubstr.

$|E|k_c - \frac{k_c(k_c-1)}{2}$, is a parameterised reduction from Multi-Coloured Clique to (s)-CloseSubstr(ℓ, m). □

Theorem 7 together with known results from the literature completely settle the parameterised complexity of (s)-CloseSubstr. See Table 3 for an illustration.[2]

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

Corollary 2 *(r, s)-CloseSubstr(ℓ, m, d_r) is W[1]-hard.*

Next, we consider several fixed-parameter tractable variants of the problem (r, s)-CloseSubstr. To this end, we first observe the following obvious parameter dependencies:

Remark 1 For (r, s)-CloseSubstr, without loss of generality, we can assume the following dependencies between the parameters:

1. $m \leq \ell$,
2. $|\Sigma| \leq \ell\, k$,
3. $d_s < k$ implies that every solution string is identical to some length-m substring of some input string.

Theorem 8 *All of the following problems are in FPT:*

- *(r, s)-CloseSubstr$(m, |\Sigma|)$,*
- *(r, s)-CloseSubstr(ℓ, k),*
- *(r, s)-CloseSubstr$(\ell, |\Sigma|)$,*
- *(r, s)-CloseSubstr(ℓ, d_s).*

Proof The problem (r, s)-CloseSubstr$(m, |\Sigma|)$ is obviously in FPT, since there are only $|\Sigma|^m$ possible candidates for solution strings. With Points 1

[2] For a corresponding table of the already known results for (r)-CloseSubstr, see, e. g., [26, Table 1].

ℓ	k	m	d_r	d_s	$\lvert\Sigma\rvert$	Result	Reference
–	–	**p**	–	–	**p**	FPT	Thm. 8
p	**p**	–	–	–	–	FPT	Thm. 8
p	–	–	–	**p**	–	FPT	Thm. 8
p	–	–	–	–	**p**	FPT	Thm. 8
p	–	**p**	**p**	–	–	W[1]-hard	Cor. 2, Open Prob. in [26]
–	**p**	–	**p**	**p**	**p**	W[1]-hard	[22]
–	**p**	**p**	**p**	**p**	–	W[1]-hard	[15]

Table 4 Results for (r,s)-CloseSubstr.

and 2 of Remark 1, this directly implies (r,s)-CloseSubstr$(\ell,k) \in \mathsf{FPT}$ and (r,s)-CloseSubstr$(\ell,|\Sigma|) \in \mathsf{FPT}$.

Finally, we consider (r,s)-CloseSubstr(ℓ,d_s). If $k \leq d_s$, then ℓ and k are parameters, a variant for which containment in FPT is already shown. If, on the other hand, $d_s < k$, then, by Point 3 of Remark 1, it is sufficient to check all substrings of the input strings s_i, $1 \leq i \leq k$, which can be done in polynomial-time. □

From these new results presented in this section and the existing results for the single-bound variants, it follows that all remaining parameterisations of (r,s)-CloseSubstr are W[1]-hard. More precisely, it is known that the problem (r)-CloseSubstr is W[1]-hard for parameterisations $(k,d_r,|\Sigma|)$ (see [22]) and (k,m,d_r) (see [15]). Hence, the obvious reduction, i. e., setting $d_s = k\,d_r$, shows that (r,s)-CloseSubstr is W[1]-hard for the 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)-CloseSubstr.

5 Kernelisation

Before we can discuss kernelisations results for the consensus string problems, we need a few more preliminary concepts and results. First, we recall the concept of a polynomial parameter transformation from [5]. A *polynomial parameter transformation* from a parameterised problem P_1 to a parameterised problem P_2 is a polynomial-time computable function f that maps P_1 instances to P_2 instances and a polynomial p, such that, for every P_1 instance (x,k) with $f(x,k) = (x',k')$, we have

- (x,k) is a positive instance if and only if (x',k') is a positive instance,
- $k' \leq p(k)$.

Theorem 9 ([5]) *Let P_1 and P_2 be parameterised problems, and let $\widehat{P_1}$ and $\widehat{P_2}$ be the corresponding classical problems derived from P_1 and P_2. Moreover, assume that $\widehat{P_1}$ is* NP*-complete, $\widehat{P_2} \in$ NP and there is a polynomial parameter transformation from P_1 to P_2. If P_2 has a polynomial kernel, then P_1 has a polynomial kernel.*

While Theorem 9 allows to carry over the existence of a polynomial kernel from one problem to another, it also allows to show that a problem most likely does not have a polynomial kernel. More precisely, if a parameterised problem P has no polynomial kernel (with respect to some complexity theoretical assumption) and there is a polynomial parameter transformation from P to some parameterised problem P', then also P' has no polynomial kernel (with respect to the same assumption).

Next, we recall the concept of a polynomial equivalence relation from [4]. Let R be an equivalence relation over Δ^*. The relation R is a *polynomial equivalence relation* if the following conditions are satisfied:

- For given $x, y \in \Delta^*$, we can decide whether x and y are R-equivalent in polynomial time.
- For every finite $X \subseteq \Delta^*$ the relation R partitions S into a number of classes that is polynomially bounded in $\max\{|x| \mid x \in X\}$.

Finally, we recall the concept of a cross composition from [4]. Let $K \subseteq \Delta^*$ be a problem (interpreted as a language), let R be a polynomial equivalence relation on Δ^* and let $P \subseteq (\Delta^* \times \mathbb{N})$ be a parameterised problem. A *cross-composition* from K into P (with respect to R) is an algorithm that, given instances $x_1, x_2, \ldots, x_q \in \Delta^*$ of K that belong to the same R-equivalence class, takes time polynomial in $\sum_{i=1}^{q} |x_i|$ and produces an instance $(y, k) \in \Delta^* \times \mathbb{N}$ such that the following holds:

- k is polynomial bounded in $\max\{|x_i| + \log q \mid 1 \leq i \leq q\}$.
- (y, k) is a positive P instance if and only if at least one x_i, $1 \leq i \leq q$, is a positive K instance.

The purpose of cross-decompositions is demonstrated by the following theorem:

Theorem 10 ([4]) *If there is a cross-composition of an* NP*-hard problem into a parameterised problem P, then P does not have a polynomial kernel, unless* coNP $\subseteq$ NP/Poly.

Note that coNP $\subseteq$ NP/Poly implies a collapse of the polynomial hierarchy and is considered unlikely. We are now ready to present the kernelisation results, which shall be proved by applying the framework provided above.

For the (d_r)-variants of CloseStr and CloseSubstr, the question whether the fixed-parameter tractable variants have a polynomial kernel has already been investigated in [2]. We restate the results relevant for us:

Theorem 11 ([2]) *The parameterised problems* (r)-CloseStr$(d_r, \ell, |\Sigma|)$ *and* (r)-CloseSubstr$(k, m, d_r, |\Sigma|)$ *do not admit a polynomial kernel unless* coNP $\subseteq$ NP/Poly.

From these results (in addition to some simple observations and other known results), we can directly conclude some results about polynomial kernels with respect to (r, s)-CloseStr and (r, s)-CloseSubstr.

Proposition 1

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

Proof Transforming an (r)-CloseStr instance into an (r,s)-CloseStr instance by setting $d_s = kd_r$ is a polynomial parameter transformation from the problem (r)-CloseStr$(d_r, \ell, |\Sigma|)$ to (r,s)-CloseStr$(d_r, \ell, |\Sigma|)$; thus, Theorem 11 implies the first statement of the proposition.

The $\mathrm{O}(k^2 d_r \log k)$ kernel for (r)-CloseStr(k, d_r) (see [18]) is also a kernel for (r,s)-CloseStr(k, d_r); which proves the second statement.

If $k > d_s$, then the only possible solution strings are the input strings, which can be checked in polynomial-time (and the instance can accordingly be reduced to a trivial positive or negative kernel of constant size). Thus, we can assume that $k \leq d_s$. Moreover, if $d_r > d_s$, then we can set $d_r = d_s$, which results in an equivalent instance, since $\mathsf{s}_\mathsf{H}(S, s) \geq \mathsf{r}_\mathsf{H}(S, s)$ for any set of strings S. Thus, we can assume that $d_r \leq d_s$. Consequently, it follows from the second statement that we can construct a kernel of size $\mathrm{O}(k^2 d_r \log k) = \mathrm{O}(d_s^3 \log d_s)$. This proves the third statement. □

With respect to (r,s)-CloseStr, this leaves the case open where only k (or k and $|\Sigma|$, which, due to the dependency $|\Sigma| \leq k$ (see [18]), is the same question) is a parameter (regarding this case, note that for (r)-CloseStr(k) no combinatorial kernel or combinatorial fpt-algorithm is known).

Question 2 Does (r,s)-CloseStr(k) allow a polynomial kernel?

Next, we take a look at kernelisation questions for (r,s)-CloseSubstr.

Proposition 2

- (r,s)-CloseSubstr$(k, m, d_r, d_s, |\Sigma|)$ *has no polynomial kernel unless* coNP $\subseteq$ NP/Poly.
- (r,s)-CloseSubstr(ℓ, k) *has a kernel of size* $\mathrm{O}(\ell k)$.
- (r,s)-CloseSubstr(ℓ, d_s) *has a kernel of size* $\mathrm{O}(\ell d_s)$.

Proof Transforming an (r)-CloseSubstr instance into a (r,s)-CloseSubstr instance by setting $d_s = kd_r$ is a polynomial parameter transformation from (r)-CloseSubstr$(k, m, d_r, |\Sigma|)$ to (r,s)-CloseSubstr(k, m, d_r, d_s, Σ); thus, Theorem 11 implies the first statement of the proposition.

Since we can assume that $d_r, d_s \leq \ell k$, any (r,s)-CloseSubstr(ℓ, k) instance has size $\mathrm{O}(\ell k)$, which proves the second point.

If $d_s < k$, then the solution string must be a substring of an input string (see Remark 1), which can be checked in polynomial-time (and the instance can accordingly be reduced to a trivial positive or negative kernel of constant size). If, on the other hand, $k \leq d_s$, then the instance has size $\mathrm{O}(\ell d_s)$. □

With respect to the different variants of CloseSubstr, the following cases are left open.

Question 3 Which of the following problems allow a polynomial kernel?

- (r,s)-CloseSubstr$(\ell, |\Sigma|)$,
- (r)-CloseSubstr(ℓ),
- (r)-CloseSubstr(ℓ, d_r),
- (r)-CloseSubstr$(\ell, |\Sigma|)$,
- (s)-CloseSubstr$(m, |\Sigma|)$,
- (r)-CloseSubstr$(d_s, |\Sigma|)$.

For the outlier-variant, no kernelisation lower bounds are known so far. However, the following can be concluded from [2].

Proposition 3 *The following problems have no polynomial kernel unless* coNP $\subseteq$ NP/Poly.

- (r)-CloseStr-wo$(d_r, \ell, t, |\Sigma|)$.
- (r,s)-CloseStr-wo$(d_r, \ell, t, |\Sigma|)$.

Proof Reducing (r)-CloseStr to (r)-CloseStr-wo by setting $t = 0$ is a polynomial parameter transformation from the problem (r)-CloseStr$(d_r, \ell, |\Sigma|)$ to (r)-CloseStr-wo$(d_r, \ell, t, |\Sigma|)$; thus, Theorem 11 implies the first statement of the proposition.

Transforming an (r)-CloseStr instance into an (r,s)-CloseStr instance by setting $d_s = kd_r$ is a polynomial parameter transformation from the problem (r)-CloseStr-wo$(d_r, \ell, t, |\Sigma|)$ to (r,s)-CloseStr-wo$(d_r, \ell, t, |\Sigma|)$; thus, the second statement of the proposition follows.

□

As our main contribution to the question of kernelisation hardness of consensus string problems, we present a cross-composition from (r)-CloseStr into (r)-CloseStr-wo, which allows us to rule out a polynomial kernel for the parameterisation $(d_r, d_s, \ell, (k-t), |\Sigma|)$ of (r)-CloseStr-wo.

Theorem 12 (r,s)-CloseStr-wo$(d_r, d_s, \ell, (k-t), |\Sigma|)$ *does not admit a polynomial kernel unless* coNP $\subseteq$ NP/Poly.

Proof We prove the result by a cross-composition of (r)-CloseStr (over the alphabet $\Sigma = \{0,1\}$) into (r)-CloseStr-wo$(d_r, d_s, \ell, (k-t), |\Sigma|)$ (note that (r)-CloseStr is NP-complete for binary alphabets [17]). We first recall that a (r)-CloseStr instance is a tuple (S, d_r) with $S = \{s_i \mid 1 \leq i \leq k\} \subseteq \Sigma^\ell$ for some $\ell \in \mathbb{N}$, and $d_r \in \mathbb{N}$; in the following, we denote its total size by $|(S, d_r)|$. We note that $|(S, d_r)| = \mathrm{O}(k\ell \log(|\Sigma|) + \log(d_r)) = \mathrm{O}(k\ell)$ (since we assume $|\Sigma| = 2$ and $d_r \leq \ell$).

We define an equivalence relation $\sim$ over the set of (r)-CloseStr 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 $d_{r,j} \in \mathbb{N}$ be two (r)-CloseStr instances. Then $(S_1, d_{r,1}) \sim (S_2, d_{r,2})$ if $k_1 = k_2$, $\ell_1 = \ell_2$ and $d_{r,1} = d_{r,2}$. For any two instances $(S_1, d_{r,1})$ and $(S_2, d_{r,2})$, it can be checked in time polynomial in $|(S_1, d_{r,1})| + |(S_2, d_{r,2})|$ whether or not $(S_1, d_{r,1}) \sim (S_2, d_{r,2})$. Let X be a finite set of (r)-CloseStr instances with

$\widehat{k}$, $\widehat{\ell}$ and $\widehat{d_r}$ being the largest number of strings, lengths of strings and radius bound that occur in any instances of X (note that these parameters can occur in different instances). Obviously, the number of equivalence classes of X (with respect to relation $\sim$) is bounded by $(\widehat{k}\,\widehat{\ell}\,\widehat{d_r})$. Moreover, each of $\widehat{k}$, $\widehat{\ell}$ and $\widehat{d_r}$ is bounded by $\max\{|x| \mid x \in X\}$ (note that $\widehat{d_r}$ is bounded by $\max\{|x| \mid x \in X\}$ since we can assume $d_r \leq \ell$ for all instances). This implies that $\sim$ partitions X into at most $(\max\{|(S, d_r)| \mid (S, d_r) \in X\})^{\mathrm{O}(1)}$ equivalence classes. Consequently, $\sim$ is a polynomial equivalence relation.

Now let $(S_1, d_r), (S_2, d_r), \ldots, (S_q, d_r)$ be $\sim$-equivalent (r)-CloseStr 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 $C_i = (B_i)^{2d_r+1}$ (i. e., C_i is the $(2d_r + 1)$-fold repetition of the binary string B_i). 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)-CloseStr-wo instance be (S', d'_r, d'_s, t) with $S' = \bigcup_{i=1}^{q} S'_i$, $d'_r = d_r$, $d'_s = kd_r$ and $t = (q-1)k$. Note that (S', d'_r, d'_s, t) is a valid (r, s)-CloseStr-wo instance with $k' = qk$ input strings that are all of the same length $\ell' = \ell + (2d_r + 1)\lceil \log(q) \rceil$. This construction can clearly be computed in polynomial time and, in order to show that it is a correct cross-composition of (r)-CloseStr into (r, s)-CloseStr-wo$(d'_r, d'_s, \ell', (k' - t), |\Sigma|)$, we have to prove the following claims.

Claim 1: Each of the parameters d'_r, d'_s, ℓ', $(k' - t)$ and $|\Sigma|$ are bounded by a polynomial in $\max\{|(S_i, d_r)| \mid 1 \leq i \leq q\} + \log(q)$.

Proof of Claim 1: We first note that $\max\{|(S_i, d_r)| \mid 1 \leq i \leq q\} + \log(q) = k\ell + \log q$ and recall that $|\Sigma| = 2$, $d'_r = d_r \leq \ell$ and $d'_s = kd_r \leq k\ell$. Moreover, $(k' - t) = qk - (q - 1)k = k$ and $\ell' = \ell + (2d_r + 1)\lceil \log(q) \rceil = \mathrm{O}(\ell \log(q))$. (Claim 1) □

Claim 2: (S', d'_r, d'_s, t) is a positive (r, s)-CloseStr-wo$(d'_r, d'_s, \ell', (k' - t), |\Sigma|)$ instance if and only if there is an i, $1 \leq i \leq q$, such that (S_i, d_r) is a positive (r)-CloseStr instance.

Proof of Claim 2: We first prove the *only if* direction and let (S', d'_r, d'_s, t) be a positive (r, s)-CloseStr-wo$(d'_r, d'_s, \ell', (k' - t), |\Sigma|)$ instance. This means that there is a set $A \subseteq S'$ with $|A| = (k' - t) = k$ and a string s such that $\mathsf{r}_\mathsf{H}(s, A) \leq d'_r = d_r$. For the sake of concreteness, let $A = \{s'_{i_1,j_1}, s'_{i_2,j_2}, \ldots, s'_{i_k,j_k}\}$. If, for some p, p', $1 \leq p < p' \leq q$, $i_p \neq i_{p'}$, then $\mathsf{d}_\mathsf{H}(s_{i_p,j_p}, s_{i_{p'},j_{p'}}) \geq 2d'_r + 1$, since $\mathsf{d}_\mathsf{H}(B_{i_p}, B_{i_{p'}}) \geq 1$ and s_{i_p,j_p}, $s_{i_{p'},j_{p'}}$ have suffixes C_{i_p}, $C_{i_{p'}}$, respectively. Consequently, $\mathsf{r}_\mathsf{H}(s', A) > d'_r$, for every $s' \in \Sigma^{\ell'}$, which is a contradiction. Hence, $i_1 = i_2 = \ldots = i_k$, which implies that $A = S'_i$, for some i, $1 \leq i \leq q$. Since all strings in S'_i have the same length-$((2d_r + 1)\lceil \log(q) \rceil)$ suffix C_i and since we obtain the strings of S_i if we remove this common suffix, we conclude that $\mathsf{r}_\mathsf{H}(s[1..\ell], S_i) \leq \mathsf{r}_\mathsf{H}(s, S'_i) \leq d'_r = d_r$, which implies that (S_i, d_r) is a positive (r)-CloseStr instance.

In order to prove the *if* direction, let, for some i, $1 \leq i \leq q$, (S_i, d_r) be a positive (r)-CloseStr instance, i. e., there is an $s \in \Sigma^\ell$ with $\mathsf{r}_\mathsf{H}(s, S_i) \leq d_r$.

Hence, $\mathsf{r}_\mathsf{H}(sC_i, S'_i) \leq d_r$, which, in particular, implies that $\mathsf{s}_\mathsf{H}(sC_i, S'_i) \leq kd_r = d'_s$. Consequently, (S', d'_r, d'_s, t) is a positive (r,s)-CloseStr-wo$(d'_r, d'_s, \ell', (k'-t), |\Sigma|)$ instance with respect to inlier-set $S'_i \subseteq S'$ and solution string sC_i. (Claim 2) □

This concludes the proof. □

6 Conclusions

In this section, we discuss our results and state the most interesting open problems that are left for further research.

Our main positive algorithmic result is the branching algorithm from Theorem 5. It demonstrates that the fixed parameter tractability of CloseStr with respect to the practically most relevant parameter d_r is robust in the sense that we can afford to also exclude outliers (as long as their number is also treated as a parameter) *and* add a distance sum bound (see also the motivation for such a problem variant at the beginning of Section 1).

Moreover, we provide a complete "fixed-parameter tractability map" for the problems CloseStr and CloseSubstr, i. e., their general variants and their single-bound variants, for all possible combinations of the parameters k, ℓ, d_r, d_s, $|\Sigma|$ and m. This is done by complementing the existing work with respect to the single-bound variants and by adapting these results to the general variant.

In this regard, (r,s)-CloseStr shows the same positive tractability results as the (r)-variant, i. e., it is fixed-parameter tractable for all single parameters except $|\Sigma|$. With respect to the substring-variant, our results demonstrate that adding a distance sum bound may increase the complexity. For example, while parameter ℓ is sufficient for fixed-parameter tractable with respect to the radius variant, we get a $\mathsf{W}[1]$-hard problem if we add a distance sum bound, even if we additionally take m and d_r as parameters. In order to maintain fixed-parameter tractability, we would have to also treat the distance sum bound as a parameter, or to take k as a parameter. In general, for the general or single-bound variants of the substring-variant, things do not look good fpt-wise.

The only questions left open with respect to CloseStr and CloseSubstr are about whether the fixed parameter tractable variants allow polynomial kernels. More precisely, it is unknown whether the general or the radius variant of CloseStr(k) allows a polynomial kernel and for the substring-variant several cases are open, which are summarised in Question 3. With respect to (r)-CloseStr(k), there is a more important question still open: the only fixed-parameter tractability results rely on integer linear programming and a combinatorial fpt-algorithm is still unknown. This has already been reported in [18] and is explicitly stated as an open problem in the survey [7]. In particular, we stress the fact that several of our fixed-parameter tractability results extend or directly use the ILP approach from [18] and therefore have the same issue;

namely, this is the case for (r,s)-CloseStr(k) and (r,s)-CloseStr-wo(k) and their single-bound variants.

With respect to the outlier-variant, our "fixed-parameter tractability map" is still rather incomplete (see also Table 2), both in the sense that for several parameterisations fixed-parameter tractability is unknown and for some fixed-parameter tractability variants it is unknown whether they allow polynomial kernels. The existing results show that, for fixed-parameter tractability, the single parameter k is sufficient (although based on ILP) and parameterising by the number of outliers t is also enough if at least one of the parameter ℓ, d_r, d_s or $k-t$ is taken into consideration as well. Unfortunately, t and $|\Sigma|$ is not enough and, surprisingly, for any other combination containing $|\Sigma|$ (except the trivial one $(|\Sigma|,\ell)$), we were not able to prove fixed-parameter tractability or W[1]-hardness. This is wort pointing out, since the parameter $|\Sigma|$ is rather important due to the fact that in practical scenarios it can often be assumed to be of rather small constant size. Consequently, the most important open question with respect to the outlier-variant is whether fixed-parameter tractability can be achieved by coupling $|\Sigma|$ with any of the parameters d_r, d_s or $k-t$ (see Question 1).

References

1. Amir, A., Landau, G.M., Na, J.C., Park, H., Park, K., Sim, J.S.: Efficient algorithms for consensus string problems minimizing both distance sum and radius. Theoretical Computer Science **412**, 5239–5246 (2011)
2. Basavaraju, M., Panolan, F., Rai, A., Ramanujan, M.S., Saurabh, S.: On the kernelization complexity of string problems. Theor. Comput. Sci. **730**, 21–31 (2018)
3. Ben-Dor, A., Lancia, G., Ravi, R., Perone, J.: Banishing bias from consensus sequences. In: Proc. 8th Annual Symposium on Combinatorial Pattern Matching, CPM 1997, *LNCS*, vol. 1264, pp. 247–261 (1997)
4. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernelization lower bounds by cross-composition. SIAM Journal of Discrete Mathematics **28**(1), 277–305 (2014)
5. Bodlaender, H.L., Thomassé, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. Theor. Comput. Sci. **412**(35), 4570–4578 (2011)
6. Boucher, C., Ma, B.: Closest string with outliers. BMC Bioinformatics **12**, S55 (2011)
7. Bulteau, L., Hüffner, F., Komusiewicz, C., Niedermeier, R.: Multivariate algorithmics for NP-hard string problems. Bulletin of the EATCS **114**, 31–73 (2014)
8. Cygan, M., Fomin, F., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)
9. Deng, X., Li, G., Li, Z., Ma, B., Wang, L.: Genetic design of drugs without side-effects. SIAM Journal of Computing **32**(4), 1073–1090 (2003)
10. Dopazo, J., Rodrguez, A., Siz, J., Sobrino, F.: Design of primers for PCR amplification of highly variable genomes. Computer Applications in the Biosciences **9**(2), 123–125 (1993)
11. Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer Science & Business Media (2012)
12. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer (2013)
13. Evans, P.A., Smith, A., Wareham, H.T.: The parameterized complexity of p-center approximate substring problems. Technical Report TR01-149, Faculty of Computer Science, University of New Brunswick, Canada (2001)
14. Evans, P.A., Smith, A.D., Wareham, H.T.: On the complexity of finding common approximate substrings. Theoretical Computer Science **306**, 407–430 (2003)

15. Fellows, M.R., Gramm, J., Niedermeier, R.: On the parameterized intractability of motif search problems. Combinatorica **26**, 141–167 (2006)
16. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
17. Frances, M., Litman, A.: On covering problems of codes. Theory of Computing Systems **30**, 113–119 (1997)
18. Gramm, J., Niedermeier, R., Rossmanith, P.: Fixed-parameter algorithms for closest string and related problems. Algorithmica **37**, 25–42 (2003)
19. Lanctot, J.K., Li, M., Ma, B., Wang, S., Zhang, L.: Distinguishing string selection problems. Information and Computation **185**, 41–55 (2003)
20. Lenstra, H.W.: Integer programming with a fixed number of variables. Mathematics of Operations Research **8**(4), 538–548 (1983)
21. Lucas, K., Busch, M., Mssinger, S., Thompson, J.A.: 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)
22. Marx, D.: Closest substring problems with small distances. SIAM Journal on Computing **38**, 1382–1410 (2008)
23. Pavesi, G., Mauri, G., Pesole, G.: An algorithm for finding signals of unknown length in DNA sequences. Bioinformatics **17**, S207–S214 (2001)
24. Pevzner, P., Sze, S.: Combinatorial approaches to finding subtle signals in DNA strings. In: Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, ISMB 2000, pp. 269–278 (2000)
25. Proutski, V., Holmes, E.C.: Primer master: a new program for the design and analysis of PCR primers. Computer Applications in the Biosciences **12**(3), 253–255 (1996)
26. Schmid, M.L.: Finding consensus strings with small length difference between input and solution strings. TOCT **9**(3), 13:1–13:18 (2017)
27. Tompa, M., Li, N., Bailey, T.L., Church, G.M., Moor, B.D., Eskin, E., Favorov, A.V., Frith, M.C., Fu, Y., Kent, W.J., Makeev, V.J., Mironov, A.A., Noble, W.S., Pavesi, G., Pesole, G., Rgnier, M., Simonis, N., Sinha, S., Thijs, G., van Helden, J., Vandenbogaert, M., Weng, Z., Workman, C., Ye, C., Zhu, Z.: Assessing computational tools for the discovery of transcription factor binding sites. Nature Biotechnology **23**(1), 137–144 (2005)