(Prefix) Reversal Distance for (Signed) Strings with Few Blocks or Small Alphabets[☆]

Laurent Bulteau[a,1], Guillaume Fertin[b,2], Christian Komusiewicz[c,2,3]

[a]*Laboratoire d'Informatique Gaspard Monge, CNRS UMR 8049, Université Paris-Est, 5 Bd Descartes 77454 Marne-la-Vallée, France*
[b]*Laboratoire d'Informatique de Nantes-Atlantique, UMR CNRS 6241, Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3, France*
[c]*Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany*

Abstract

We study the String Reversal Distance problem, an extension of the well-known Sorting by Reversals problem. String Reversal Distance takes two strings S and T built on an alphabet Σ as input, and asks for a minimum number of reversals to obtain T from S. We consider four variants: String Reversal Distance, String Prefix Reversal Distance (a constrained version of the previous problem, in which any reversal must include the first letter of the string), and the signed variants of these problems, namely Signed String Reversal Distance and Signed String Prefix Reversal Distance. We study algorithmic properties of these four problems, in connection with two parameters of the input strings: the number of *blocks* they contain (a block being a maximal substring such that all letters in the substring are equal), and the *alphabet size* $|\Sigma|$. Concerning the number of blocks, we show that the four problems are fixed-parameter tractable (FPT) when the considered parameter is the maximum number of blocks among the two input strings. Concerning the alphabet size, we first show that String Reversal Distance and String Prefix Reversal Distance are NP-hard even if the input strings are built on a binary alphabet $\Sigma = \{0, 1\}$, each 0-block has length at most two and each 1-block has length one. We also show that Signed String Reversal Distance and Signed String Prefix Reversal Distance are NP-hard even if the input strings have only *one* letter. Finally, when $|\Sigma| = O(1)$, we provide a singly-exponential algorithm that computes the exact distance between any pair of strings, for a large family of distances that we call *well-formed*, which

[☆]A preliminary version of this article appeared in Proceedings of the 25th Annual Symposium on Combinatorial Pattern Matching (CPM 2014) [3]

Email addresses: `laurent.bulteau@u-pem.fr` (Laurent Bulteau), `guillaume.fertin@univ-nantes.fr` (Guillaume Fertin), `christian.komusiewicz@uni-jena.de` (Christian Komusiewicz)

[1]Supported by the Alexander von Humboldt Foundation
[2]Supported by Procope grant number 28436VG
[3]Supported by a post-doctorial grant funded by the Région Pays de la Loire

includes the four distances we study here.

1. Introduction

Many problems studied in the realm of comparative genomics concern genome rearrangements, where the general goal is to better understand the evolutionary history of species, by realizing pairwise comparisons of their genomes. A genome is modeled as a linearly ordered sequence of elements, called genes, where each gene is represented by a unique symbol that identifies it; hence, genomes can also be seen as permutations. Given two genomes G_1 and G_2, and a set $\mathcal{S}$ of possible operations (rearrangements) that can be applied to these genomes, one classical problem consists in computing the smallest number of rearrangements that allows to obtain G_2, starting from G_1, and using only operations from $\mathcal{S}$. We refer the reader to [13] for an extensive survey on the subject.

One of the most studied, and historically one of the firstly described such rearrangement is the *reversal* [18], which consists in taking a contiguous subsequenceof a genome, reverse its order, and reincorporate it at the same location. This gave rise to the Sorting by Reversals (SBR) and Sorting by Signed Reversals (SBSR) problems. In the signed version of the problem, each element of the permutation is additionally labeled with a sign $+$ or $-$ and a reversal not only reverses the order, but also inverts the signs of all the elements involved in it. In terms of complexity, SBR has been proved to be NP-hard [5] and the best current approximation ratio is 1.375 [2]. In contrast, SBSR is polynomial [1]. Another variant consists in imposing the use of *prefix reversals* only, that is, each reversal must contain the first letter of the permutation it is applied to. The corresponding unsigned and signed problems are called SBPR and SBSPR, respectively. The SBPR problem has been recently shown to be NP-hard [4], and the best current approximation ratio is 2 [14]. The complexity of SBSPR, however, is still open, and a 2-approximation algorithm also exists [12]. The genome rearrangement problem has, for a long time, been studied in the case where each gene is considered to be unique, and thus, as mentioned above, where genomes are modeled by permutations. In biological applications, however, genomes of related species often contain many homologous genes. In this case, genomes cannot be modeled by permutations, but rather by (signed) strings [6]. Hence, a natural and biologically more relevant extension of SBR is the String Reversal Distance problem, formally defined, in its decision version, as follows:

> String Reversal Distance
> **Input:** Two unsigned strings S and T over alphabet Σ and an integer k.
> **Question:** Can S be transformed into T by applying at most k reversals?

We are also interested in the signed version of the above problem, which will be denoted Signed String Reversal Distance.

Signed String Reversal Distance
Input: Two signed strings S and T over alphabet Σ and an integer k.
Question: Can S be transformed into T by applying at most k reversals?

Now, if we allow prefix reversals only, the extension of SBPR to strings is defined as follows:

String Prefix Reversal Distance
Input: Two unsigned strings S and T over alphabet Σ and an integer k.
Question: Can S be transformed into T by applying at most k prefix reversals?

As above, Signed String Prefix Reversal Distance will denote the signed version of the problem.

Signed String Prefix Reversal Distance
Input: Two signed strings S and T over alphabet Σ and an integer k.
Question: Can S be transformed into T by applying at most k prefix reversals?

Any of these four problems is nontrivially posed only if S and T have the same content, that is, for each letter a in Σ, the number of occurrences of a in S is equal to the number of its occurrences in T. We call such pairs of strings *balanced* and throughout this work, we will always assume that S and T are balanced and of length n. Given two balanced strings S and T, $rd(S,T)$ ($srd(S,T)$) will denote the smallest k for which the answer to String Reversal Distance (Signed String Reversal Distance) is positive, which is exactly the (signed) reversal distance between S and T. Notations $prd(S,T)$ and $sprd(S,T)$ are similarly defined concerning problems String Prefix Reversal Distance and Signed String Prefix Reversal Distance, respectively. A *block* in a string S denotes a maximal substring s of S such that all letters in s are the same (and have the same sign, if strings are signed). For any instance (S,T) of any of the four problems presented above, we use $b_{\max}$ to denote the maximum of the number of blocks in S and T and $b_{\min}$ to denote its minimum. Unless stated otherwise, we assume that S has $b_{\max}$ blocks. Note that $n \geq b_{\max} \geq b_{\min} \geq |\Sigma|$. We are interested in *FPT* (for *Fixed-Parameter Tractable*) algorithms. These are algorithms running in time $f(k)\,\mathrm{poly}(n)$, where k is a suitable parameter and f is any computable function depending only on k.

In this paper, our goal is to study algorithmic and complexity issues for these four problems, in connection with two parameters of the input strings: the maximum number of blocks $b_{\max}$ they contain and the size of the alphabet Σ they are built on. We are particularly interested in the cases where $b_{\max}$ and/or $|\Sigma|$ is small.

Table 1: Overview of results for unsigned reversals. The results marked with $^{(*)}$ hold even when 0-blocks are of length at most 2 and 1-blocks are of length 1.

reversals		prefix reversals	
$\|\Sigma\| = 2$	general	$\|\Sigma\| = 2$	general
NP-h [10, 11] and Thm 7$^{(*)}$	NP-h [5, 10, 11]	NP-h [16] and Thm 7$^{(*)}$	NP-h [16, 4]
FPT in $b_{\max}$ (Thm 5)			

Table 2: Overview of results for signed reversals

signed reversals			signed prefix reversals		
$\|\Sigma\| = 1$	$\|\Sigma\| = 2$	general	$\|\Sigma\| = 1$	$\|\Sigma\| = 2$	general
NP-h (Thm. 8)	NP-h [17]		NP-h (Thm. 8)	NP-h [7]	
FPT in $b_{\max}$ (Thm. 6)					

Our results, together with some known results, are summarized in Tables 1 and 2. Our main algorithmic result is a fixed-parameter algorithm for the four problem variants and the parameter maximum number of blocks $b_{\max}$. This result relies mainly on diameter bounds that depend only on $b_{\max}$ and $|\Sigma|$ and which we provide in Section 2. We believe that these diameter bounds are also of independent interest. Then, in Section 3 we show the aforementioned algorithm for the parameter $b_{\max}$. This algorithm is based on a reduction to the problem of computing a maximum network flow. In Section 4 we describe a reduction from SBR (resp. SBPR) that yields several hardness results. First, it shows that for String Reversal Distance and String Prefix Reversal Distance we cannot make use of a bounded block length even if input strings are binary as both problems become NP-hard for the first nontrivial case, that is, if each 0-block has length at most two and each 1-block has length one. Second, it shows that Signed String Reversal Distance and Signed String Prefix Reversal Distance remain NP-hard even over unary alphabet. This strengthens the previous hardness results by Radcliffe et al. [17] and Chitturi [7] who showed hardness for binary alphabets. Finally, we show a simple algorithm that achieves a running time of $|\Sigma|^n \cdot \text{poly}(n)$ for many string distances including the ones under consideration in this work.

Some Notations and a Structural Property. We start with some notations. We denote the i-th letter of a string S by $S[i]$. A reversal $\rho(i, j)$ in a string S of length n transforms

$$S[1] \cdots S[i-1] \boxed{S[i]S[i+1] \cdots S[j-1]S[j]} S[j+1] \cdots S[n]$$

into

$$S[1] \cdots S[i-1] \boxed{S[j]S[j-1] \cdots S[i+1]S[i]} S[j+1] \cdots S[n].$$

Applied on a signed string, a reversal additionally inverts the signs of the letters $S[i], \ldots, S[j]$. We denote the (signed) string that results from applying reversal ρ to S by $S \circ \rho$. We use $b(S)$ to denote the number of blocks of a string S. For any (signed) string S we denote by $\underline{S}$ the string that is obtained by the reversal $\rho(1, |S|)$. The following simple observation concerns the String Reversal Distance problem, and will be useful in a later part of this work.

Proposition 1. *For any two balanced unsigned strings S and T, there is a shortest sequence $\rho_1, \rho_2, \ldots, \rho_k$ of reversals from S to T such that for any $1 \leq \ell \leq k$, the start and endpoint of ρ_ℓ have different letters.*

Proof. Assume towards a contradiction that this is not the case. Then, there are two strings S and T, $S \neq T$, such that for any shortest sequence of reversals from S to T the first reversal $\rho(i,j)$ fulfills $S[i] = S[j]$. Pick among all such first reversals one in which $j - i$ is minimum. Clearly, $j - i > 2$ since otherwise, $S \circ \rho(i,j) = S$. Then, however, $\rho(i+1, j-1)$ is a reversal and since $S[i] = S[j]$ we have $S \circ \rho(i,j) = S \circ \rho(i+1, j-1)$. This contradicts the choice of i and j. $\square$

2. Upper Bounds on the Reversal Diameter

In this section, we present a series of results that will be essential for proving that our four problems are FPT in $b_{\max}$ (Theorems 5 and 6, Section 3). The following results are diameter upper bounds for our problems that depend only on $b_{\max}$ and $|\Sigma|$. On the one hand, these bounds are necessary for proving Theorems 5 and 6. On the other hand, we believe that they are of independent interest. We also point out that some of them may easily be improved: our main goal was not to find the best upper bounds, but sufficiently small ones so that our algorithm proves to be FPT. Our upper bounds complement previous diameter bounds which were presented for different types of reversal distances between strings [11, 8].

2.1. Considering Unsigned Reversals

We first show an upper bound on the number of reversals needed to reach an arbitrary "grouped" string, that is, any string with $|\Sigma|$ blocks. We note that related bounds were obtained by Christie and Irving [11] and Chitturi and Sudborough [8]. These bounds are incomparable to our bounds, because they are either restricted to binary alphabets or because the upper bounds may reach $\Omega(n)$ even for a constant number of blocks.

Lemma 1. *Let S be a string with b blocks over alphabet Σ. There exists a string S_{g} with $|\Sigma|$ blocks such that $rd(S, S_{\mathrm{g}}) \leq b - |\Sigma|$ and $prd(S, S_{\mathrm{g}}) \leq \min\{b - 1, 2(b - |\Sigma|)\}$.*

Proof. Let us first show the result for reversals. We apply the following greedy algorithm: while $b > |\Sigma|$, identify two distinct blocks that contain the same letter, say a (such blocks must exist because $b > |\Sigma|$). Apply the reversal that starts at the leftmost letter of the first block containing a and ends at the letter before the second block containing a. This reversal strictly decreases the number of blocks by 1, thus when S_{g} is reached, $b - |\Sigma|$ reversals have been applied.

For prefix reversals, we use a similar greedy strategy to show both upper bounds. First, we describe an algorithm that does not use more than $b - 1$ prefix reversals: if the first letter a appears only in this block, then reverse the complete string (which is a prefix reversal) and remove the last block of the resulting string from the instance (or similarly, apply the following only on the substring that excludes this block). The removal of the last block reduces $|\Sigma|$ by one since a appears only in this block. Since a string with unary alphabet has one block, we perform this type of prefix reversal at most $|\Sigma| - 1$ times. Note that this does not increase $b - |\Sigma|$ since b also decreases by one. If a appears in at least two blocks, then apply the prefix reversal whose endpoint is the rightmost letter before the second block that contains a. This prefix reversal reduces the number b of blocks by one and we will thus call such a reversal *efficient*. The overall number of prefix reversals that are applied until a grouped string is reached is thus at most $b - |\Sigma| + |\Sigma| - 1$.

Second, the $2(b - |\Sigma|)$ upper bound is obtained by iterating the following process until S_{g} is reached: if the first letter a appears in another block, apply an efficient prefix reversal as described in the previous algorithm. If not, let c be a letter that appears in at least two blocks in S (such a letter exists, otherwise we are done), and operate a prefix reversal that brings c in first position and does not break any other block. Altogether, $b - |\Sigma|$ efficient prefix reversals are used, and at most $b - |\Sigma|$ non-efficient ones. $\square$

We now use Lemma 1 to obtain an upper bound for the reversal distance between any strings. The approach is to transform each input string into some grouped string and then transform one grouped string into the other.

Theorem 1. *Let S and T be two balanced strings. Then*

- $rd(S,T) \leq b_{\max} + b_{\min} - |\Sigma| - 1$ *and*
- $prd(S,T) \leq \min\{b_{\max} + b_{\min} + \frac{18|\Sigma|}{11} + O(1), 2b_{\max} + 2b_{\min} - \frac{26|\Sigma|}{11} + O(1)\}$.

Proof. We first show the bound for reversals. By Lemma 1, there exists a grouped string S_{g} (resp. T_{g}) such that S (resp. T) can be transformed into S_{g} (resp. T_{g}) using at most $b(S) - |\Sigma|$ (resp. $b(T) - |\Sigma|$) reversals. Moreover, S_{g} and T_{g} can be transformed into each other by at most $|\Sigma| - 1$ reversals: Since each letter occurs only in one block, the problem reduces to finding the reversal distance of two permutations of length $|\Sigma|$, which is at most $|\Sigma| - 1$ [1]. The overall reversal distance between S and T thus is upper bounded by $b(S) - |\Sigma| + b(T) - |\Sigma| + |\Sigma| - 1 = b_{\max} + b_{\min} - |\Sigma| - 1$.

For prefix reversals, the proof is essentially the same: (i) reach a grouped string S_{g} from S and (ii) a grouped string T_{g} from T, and finish by (iii) obtaining T_{g} from S_{g}. Depending on what algorithm is used for achieving (i) and (ii) (see proof of Lemma 1), these two operations cost altogether either $b_{\max} + b_{\min} - 2$ or $2(b_{\max} + b_{\min} - 2|\Sigma|)$. Now, the number of prefix reversals in (iii) is upper-bounded by $\frac{18|\Sigma|}{11} + O(1)$ [9], which finishes the proof. $\square$

For the reversal case, we can also obtain a bound of the type $b_{\max} + O(|\Sigma|^2)$. Hence, if $b_{\min}$ is much larger than $|\Sigma|$, for example if $|\Sigma|$ is a small constant, this improves the above bound.

Theorem 2. *Let S and T be two balanced strings. Then $rd(S,T) \leq b_{\max} + |\Sigma|^2 - 2|\Sigma|$.*

Proof. If $b_{\max} < |\Sigma|^2 - |\Sigma| + 2$, then $b_{\min} < |\Sigma|^2 - |\Sigma| + 2$ and thus

$$rd(S,T) \leq b_{\max} + b_{\min} - |\Sigma| - 1 \leq b_{\max} + |\Sigma|^2 - 2|\Sigma|$$

by Theorem 1. Now, assume that $b_{\max} \geq |\Sigma|^2 - |\Sigma| + 2$ and that the claim holds for all pairs of strings with $b'_{max} < b_{\max}$. We show how to apply a constant number of reversals on S or on S and T that reduce the number of blocks sufficiently to obtain the bound.

Case 1: $b_{\max} > b_{\min}$. Assume that S has $b_{\max}$ blocks. Apply any reversal on S that reduces the number of blocks by one (note that this is always possible since $b_{\max} > b_{\min} \geq |\Sigma|$). Let S' be the resulting string. By the inductive hypothesis, $rd(S',T) \leq b_{\max} - 1 + |\Sigma|^2 - 2|\Sigma|$. Since $rd(S,S') = 1$, the claim holds in this case.

Case 2: $b_{\max} = b_{\min}$. Any string U with at least $|\Sigma|^2 - |\Sigma| + 2$ blocks has the following property: there are two distinct letters, say a and b, such that U contains substring ab twice. This can be seen by considering a directed multigraph with vertex set Σ, in which we add an arc (u,v) for each substring uv, $u \neq v$ of U. Any pair of neighboring blocks corresponds to an arc in this graph. Now there are at least $|\Sigma|^2 - |\Sigma| + 1$ pairs of neighboring blocks in U. Hence, the multigraph has at least $|\Sigma|^2 - |\Sigma| + 1$ arcs. Since a simple directed graph can have at most $|\Sigma|^2 - |\Sigma|$ arcs, there is at least one pair of vertices u and v, for which the arc (u,v) is contained twice in this multigraph.

By the above, S has two distinct letters, say a and b, such that there are i and $j > i + 1$ with $S[i] = a$, $S[i+1] = b$, $S[j] = a$ and $S[j+1] = b$. The reversal $\rho(i+1, j)$ produces a string S' with $b_{\max} - 2$ blocks. Similarly, there is some reversal that transforms T into a string T' with $b_{\max} - 2$ blocks. By the inductive hypothesis, $rd(S',T') \leq b_{\max} - 2 + |\Sigma|^2 - 2|\Sigma|$. Together with the two additional reversals on S and T, we obtain the bound on the number of reversals also in this case. $\square$

2.2. Considering Signed Reversals

In view of presenting an FPT algorithm for signed versions of String Reversal Distance (Theorem 6), we now give diameter bounds for these versions. These bounds are based on similar observations as the ones above. Herein, we make use of the following folklore fact; we give a short proof for sake of completeness.

Proposition 2. *Let S and T be two signed strings with reversal distance $srd(S,T) = 1$. Then S can be transformed into T by at most three prefix reversals.*

Proof. Let $S := S_1 S_2 S_3$ and $T := S_1 \underline{S_2} S_3$. Apply the following three prefix reversals. First, transform S into $S' := \underline{S_2}\,\underline{S_1}\,S_3$. Then, transform S' into $S'' := S_2 \underline{S_1} S_3$. Then, transform S'' into T by one further signed prefix reversal. $\square$

The first step for obtaining the diameter bounds is again to show an upper bound on a string grouping problem. If we try to directly obtain a grouped string with $|\Sigma|$ blocks, then $2(b_{\max} - |\Sigma|)$ reversals would suffice since we can always reduce the block number by one with at most two reversals. In the following, we describe a somewhat more efficient approach, where we quickly reach a string with $2|\Sigma|$ blocks before completing the grouping into only $|\Sigma|$ blocks.

Theorem 3. *Let S be a signed string with b blocks over alphabet Σ. There exists a signed string S_{g} with $|\Sigma|$ blocks such that $srd(S, S_{\mathrm{g}}) \leq \frac{3}{2}b - |\Sigma| + 1$ and $sprd(S, S_{\mathrm{g}}) \leq \frac{9}{2}b - 3|\Sigma| + 3$.*

Proof. The first step consists in transforming S into a string S' with at most $2|\Sigma|$ blocks. Note that if $b \leq 2|\Sigma|$, then we directly take $S' = S$. Otherwise, there is some letter a that occurs in at least three blocks. We show how to either apply one reversal that reduces the block number by one or three reversals that reduce the block number by two.

Case 1: a occurs with different signs. Consider two blocks that contain a where one consists of $-a$'s and the other one of a's. Performing a reversal that starts at the leftmost letter of the first block and ends at the rightmost letter before the second block reduces the number of blocks by one, since the reversal changes the sign of the first block.

Case 2: a occurs with one sign only. Assume without loss of generality that a occurs only with positive sign. Apply one reversal that only reverses the second block containing a. Then, apply the reversal that starts at the leftmost letter of the first block and ends at the rightmost letter before the second block. As described above, this reversal reduces the number of blocks by one. Afterwards, there are still two blocks containing a with different signs. Hence, Case 1 applies and we can perform one further reversal that reduces the number of blocks by one. Hence, by three reversals the number of blocks can be reduced by two.

Altogether, this shows that if $b > 2|\Sigma|$ is even, then $\frac{3}{2}(b - 2|\Sigma|)$ reversals suffice to get to S'. If b is odd, $\frac{3}{2}(b-1-2|\Sigma|)$ reversals are sufficient to obtain a string with $2|\Sigma| + 1$ blocks, from which another 2 reversals lead to S'. In either case we can obtain S' with at most $\frac{3}{2}b - 3|\Sigma| + 1$ reversals.

The second step consists in transforming S', which has $b' \leq 2|\Sigma|$ blocks, into S_{g} with $|\Sigma|$ blocks. This can be done in $2(b' - |\Sigma|)$ reversals: as long as a letter occurs in two different blocks, apply 2 reversals to join the two blocks into one, and thus decrease $(b' - |\Sigma|)$ by one. Note that $2(b' - |\Sigma|) \leq 2|\Sigma|$ and that $2(b' - |\Sigma|) = \frac{3}{2}b' + \frac{1}{2}b' - 2|\Sigma| \leq \frac{3}{2}b' - |\Sigma|$.

Summing the two steps for the case $b \geq 2|\Sigma|$, we get a bound of $(\frac{3}{2}b - 3|\Sigma| + 1) + 2|\Sigma| \leq \frac{3}{2}b - |\Sigma| + 1$. Using only the second step when $b = b' \leq 2|\Sigma|$, we directly get an upper bound of $2(b' - |\Sigma|) \leq \frac{3}{2}b - |\Sigma|$.

The bound for signed prefix reversals follows directly from Proposition 2. □

We can now use the same strategy as for the unsigned version: First transform each string into a grouped string and then transform the two grouped strings into each other.

Theorem 4. *Let S and T be two signed balanced strings. Then $srd(S,T) \leq \frac{3}{2}(b_{\max} + b_{\min}) - |\Sigma| + 3$ and $sprd(S,T) \leq \frac{9}{2}(b_{\max} + b_{\min}) - 3|\Sigma| + 9$.*

Proof. We first prove the upper bound in the case of signed reversals. The bound for prefix reversals then follows from Proposition 2. Suppose without loss of generality that $b(S) = b_{\max}$ and $b(T) = b_{\min}$. By Theorem 3, using at most $\frac{3}{2}b_{\max} - |\Sigma| + 1$ reversals we can transform S into a string S_{g} having at most $|\Sigma|$ blocks. Similarly, at most $\frac{3}{2}b_{\min} - |\Sigma| + 1$ reversals are necessary to transform T into a string T_{g} with at most $|\Sigma|$ blocks. Now, with an additional $|\Sigma| + 1$ reversals [15], we can transform S_{g} into T_{g}, which concludes the proof. □

3. Sorting Strings with Few Blocks

In this section, we show that the four problems String Reversal Distance, String Prefix Reversal Distance, Signed String Reversal Distance and Signed String Prefix Reversal Distance are fixed-parameter tractable with respect to parameter $b_{\max}$. We begin by showing the following theorem, which is only concerned with String Reversal Distance. Then, Theorem 6 extends this tractability result to the three other problems.

Theorem 5. String Reversal Distance *is fixed-parameter tractable, parameterized by $b_{\max}$. More precisely, it can be solved in time:*

- $(6b_{\max})^{4b_{\max}}\,\mathrm{poly}(n)$ *time on arbitrary strings ;*
- $(6b_{\max})^{2b_{\max}}\,\mathrm{poly}(n)$ *time whenever $|\Sigma| = O(1)$.*

The FPT algorithm we present for solving String Reversal Distance consists of two main steps: First, "guess" between which blocks each of the reversals takes place, then compute the precise endpoints of the reversals within each block. The guesses of the first step fix the structure of the reversals and, for any positive integer k, we will thus call a sequence of at most k of those guesses a *scaffold* (see Definition 2 and Figure 1). The guessing step is performed by a simple enumeration algorithm (the main difficulty consists in bounding the number of scaffolds that are considered), while the second step is achieved by computing a maximum flow on an auxiliary graph.

Step 1: Enumerating Scaffolds. The purpose of a scaffold is to describe the behavior of the blocks through a sequence of reversals. However, it does not describe precisely the position of each reversal endpoint within the blocks (since otherwise there would be too many cases to enumerate). We do, however,

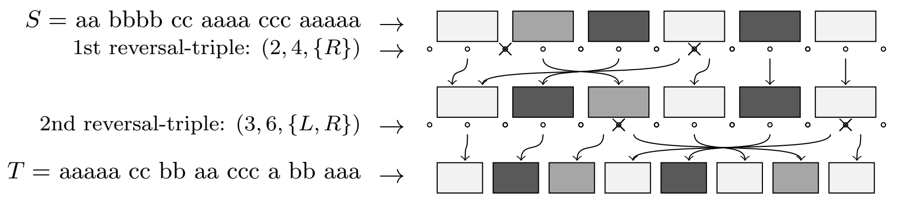

$S =$ aa bbbb cc aaaa ccc aaaaa $\rightarrow$
1st reversal-triple: $(2, 4, \{R\})$ $\rightarrow$
2nd reversal-triple: $(3, 6, \{L, R\})$ $\rightarrow$
$T =$ aaaaa cc bb aa ccc a bb aaa $\rightarrow$

Figure 1: Illustration of a scaffold. Strings S and T (left) are interpreted as a series of blocks (right, the colors depict the repeated letter of each block). The circles depict the possible reversal endpoints (either breaking a block, or between two consecutive blocks). The scaffold is defined by the reversal-triples, each of them being represented as two crosses. The arcs represent how blocks (or parts of block) move from one step to the other.

guess whether the startpoint of each reversal is the first position of a block and whether the endpoint of each reversal is the last position of a block. This notion is defined as follows.

Definition 1. *A reversal $\rho(i, j)$ is called* left-breaking *if $S[i - 1] = S[i]$ and* right-breaking *if $S[j] = S[j + 1]$.*

We can now give a formal definition of a scaffold.

Definition 2. *A* scaffold *$\mathcal{S}(S, T) = ((i_1, j_1, B_1), (i_2, j_2, B_2), \ldots, (i_k, j_k, B_k))$ is a tuple of triples, called* reversal-triples*, where $i_\ell, j_\ell \in \mathbb{N}$ and $B_\ell \subseteq \{L, R\}$, $1 \leq \ell \leq k$. A sequence of k reversals $\rho_1, \rho_2, \ldots, \rho_k$ from string S_1 to S_{k+1} with $S_i \circ \rho_i = S_{i+1}$* respects *a scaffold if for each ℓ, $1 \leq \ell \leq k$, we have that*

- *the startpoint of ρ_ℓ is in the i_ℓth block of S_ℓ,*
- *the endpoint of ρ_ℓ is in the j_ℓth block of S_ℓ,*
- *ρ_ℓ is left-breaking if and only if $L \in B_\ell$, and*
- *ρ_ℓ is right-breaking if and only if $R \in B_\ell$.*

The aim of Step 1 is simply to enumerate all possible scaffolds, so that in Step 2, the algorithm computes for each scaffold whether one can assign two positions to each reversal to obtain a sequence of reversals that respects the scaffold and transforms S into T.

In order to bound the running time of the algorithm, we need to bound the number of different scaffolds that are considered during this step. Intuitively, the number of scaffolds is a function of the number of blocks in each string (bounded by $O(b_{\max})$), and the *size* of each scaffold, i.e. the number of reversal-triples k.

By Theorems 1 and 2, we can assume that $k < b_{\max} + b_{\min} - |\Sigma| \leq 2b_{\max} - 2$ and $k \leq b_{\max} + |\Sigma|^2 - 2|\Sigma| < b_{\max} + |\Sigma|^2$ since otherwise the instance is a yes-instance. Hence, every scaffold that is respected by an optimal solution has at most $\max\{2b_{\max} - 2, b_{\max} + |\Sigma|^2\}$ reversal-triples. The algorithm branches for each such reversal-triple into the possible choices for i_ℓ and j_ℓ and whether the reversal shall be left- or right-breaking. By the above argument, it needs to perform at most $\max\{2b_{\max} - 2, b_{\max} + |\Sigma|^2\}$ branchings. Furthermore, the number of blocks in any “intermediate” string is bounded as shown below.

Lemma 2. *Let S' be a string such that there is an optimal sequence of reversals from S to T in which S' is one of the strings produced by this sequence. Then, $b(S') \leq \min\{2b_{\max} + b_{\min} - |\Sigma| - 1, 2b_{\max} + |\Sigma|^2 - 2|\Sigma|\}$ blocks.*

Proof. Assume towards a contradiction that $b(S') > 2b_{\max} + b_{\min} - |\Sigma| - 1$. More than $(b_{\max} + b_{\min} - |\Sigma| - 1)/2$ reversals are needed to transform S into S' since each reversal increases the number of blocks by at most two and the difference in the number of blocks between S and S' is more than $b_{\max} + b_{\min} - |\Sigma| - 1$. Similarly, more than $(b_{\max} + b_{\min} - |\Sigma| - 1)/2$ reversals are needed to transform S' into T. Hence, the number of reversals to transform S into T via S' is more than $b_{\max} + b_{\min} - |\Sigma| - 1$. However, by Theorem 1, this contradicts the choice of S'. The same argument applies to show the second bound of the lemma if we make use of Theorem 2. □

Now, the algorithm creates for increasing $k' \leq k$ all possible reversal scaffolds. By the above lemma, there are less than $3b_{\max}$ choices for each i_ℓ and j_ℓ. Hence, the overall number of reversal scaffolds that need to be considered is at most

$$(3b_{\max})^{2\cdot(2b_{\max}-2)} \cdot 4^{2b_{\max}-2} = O((6b_{\max})^{4b_{\max}})$$

in the case of arbitrary alphabets. For constant-size alphabets, we can use the bound on k given by Theorem 2 and thus the overall number of reversal scaffolds that need to be considered in this case is less than

$$(3b_{\max})^{2\cdot(b_{\max}+|\Sigma|^2)} \cdot 4^{b_{\max}+|\Sigma|^2} = (6b_{\max})^{2b_{\max}} \cdot \text{poly}(b_{\max}).$$

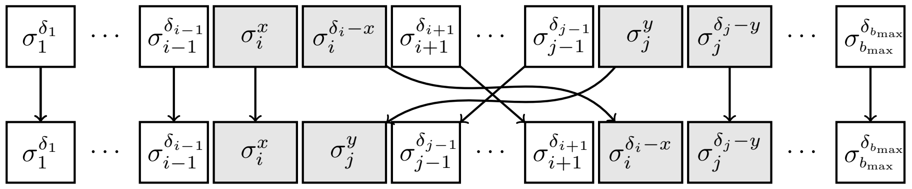
Consider one such scaffold, assume there is a sequence of reversals that respects the scaffold, and for any $1 \leq \ell \leq k'$, let $S_{\ell+1} := S_\ell \circ \rho_\ell$ denote the string obtained after the ℓth reversal. We show that the number and order of blocks of each S_ℓ is completely fixed by the reversal scaffold. First, assume that S has $b_{\max}$ blocks and consider $S_1 := S$. For any $1 \leq i \leq b_{\max}$, let δ_i denote the number of letters in the ith block of S_1 and let σ_i denote the letter of the ith block. Furthermore, assume that i_1 is in the ith block of S_1 and j_1 is in the jth block of S_1. Then this reversal transforms the string

$$S_1 = (\sigma_1)^{\delta_1} \cdots (\sigma_i)^{\delta_i} (\sigma_{i+1})^{\delta_{i+1}} \cdots (\sigma_{j-1})^{\delta_{j-1}} (\sigma_j)^{\delta_j} \cdots (\sigma_{b_{\max}})^{\delta_{b_{\max}}}$$

into the following string (where x and y represent the number of elements to the left of the cut in the ith and jth blocks, see Figure 2):

$$S_2 = (\sigma_1)^{\delta_1} \cdots (\sigma_i)^{x} (\sigma_j)^{y} (\sigma_{j-1})^{\delta_{j-1}} \cdots (\sigma_{i+1})^{\delta_{i+1}} (\sigma_i)^{\delta_i - x} (\sigma_j)^{\delta_j - y} \cdots (\sigma_{b_{\max}})^{\delta_{b_{\max}}}.$$

Recall that the scaffold fixes whether the reversal is left-breaking and whether it is right-breaking. In other words, it is known whether $x = 0$ or $x > 0$ and whether $y < \delta_j$ or $y = \delta_j$. Consequently, it is fixed whether the letter preceding the endpoint of the reversal in S_2 is σ_i or whether this letter is σ_{i+1}.



$\sigma_1^{\delta_1}$
$\sigma_{i-1}^{\delta_{i-1}}$
σ_i^x
$\sigma_i^{\delta_i-x}$
$\sigma_{i+1}^{\delta_{i+1}}$
$\sigma_{j-1}^{\delta_{j-1}}$
σ_j^y
$\sigma_j^{\delta_j-y}$
$\sigma_{b_{\max}}^{\delta_{b_{\max}}}$

Figure 2: A reversal that starts at the $(x+1)$th position of block i and ends at the yth position of block j. If $x = 0$, then the block with content σ_i^x is empty (similarly for $y = \delta_j$ and the block with $\sigma_j^{\delta_j-y}$). Note that in this case the blocks with content $\sigma_{i-1}^{\delta_{i-1}}$ and $\sigma_j^{\delta_y}$ are merged into one block if $\sigma_{i-1} = \sigma_j$.

Similarly, it is fixed whether the letter succeeding the startpoint of the reversal in S_2 is σ_j or whether it is σ_{j-1}. Therefore, we know whether the borders of the reversal are startpoints or endpoints of new blocks in S_2 or whether they are "merged" with old blocks. Consequently, the number of blocks in S_2 and the letter for each block in S_2 is known. This is similarly true for $S_3 = S_2 \circ \rho_2$ up until $S_{k'+1} = S_{k'} \circ \rho_{k'}$. Hence, the number of blocks, their order, and the letter that each block contains is fixed in $S_{k'+1}$. Thus, if the number of blocks in T is different from the number of blocks in $S_{k'+1}$ or if the letter of the ith block in T is different from the letter from the ith block in $S_{k'+1}$, then we can discard the reversal scaffold. Consequently, it now remains to check whether the reversal scaffold can produce blocks of the correct size.

Step 2: Computing block sizes. One possible way of checking whether blocks of the correct size can indeed be produced would be to introduce a variable for the length of each block in each S_i, $1 \leq i \leq k'$, and then introduce equations that model the dependencies between the blocks. For example if the reversal from S_i to S_{i+1} appears after the first block, then the lengths of the first blocks should be equal. Since the number of blocks of each S_i and k' are bounded in functions of $b_{\max}$ this would yield an integer linear program whose number of variables depends only on $b_{\max}$, which implies fixed-parameter tractability with respect to $b_{\max}$. In the following, we describe a more efficient approach that is based on computing maximum value flows. For each considered reversal scaffold we create one flow network $G = (V, A, c, s, t)$, where $c : A \to \mathbb{N}$ denotes the capacity function on the arcs of G, and s (resp. t) is the source (resp. target) vertex. The flow network G contains $O((b_{\max})^2)$ vertices and arcs, and is constructed as follows (see also Figure 3).

Create the source s and the sink t. For each block i in each intermediate string S_ℓ, $1 \leq \ell \leq k'+1$, add one vertex v_ℓ^i; we use $V_\ell := \{v_\ell^i \mid 1 \leq i \leq b(S_\ell)\}$ to denote the vertex set corresponding to S_ℓ. Now, add arcs and capacities as follows. For each v_1^i add the arc (s, v_1^i). Set the capacity of $c(s, v_1^i)$ to be exactly the length of the ith block in S. For each $\ell \leq k'$ introduce arcs between the vertices corresponding to blocks of S_ℓ to those representing blocks of $S_{\ell+1}$ as follows.

Assume that the reversal ρ is fixed to start within the ith block of S_ℓ and end in the jth block of S_ℓ. Furthermore, let β denote the difference between the number of blocks in $S_{\ell+1}$ and in S_ℓ. Then, add the following arcs, with unbounded capacity, to G:

- for all $i' < i$ add the arc $(v_\ell^{i'}, v_{\ell+1}^{i'})$
- for all $i' > j$ add the arc $(v_\ell^{i'}, v_{\ell+1}^{\beta+i'})$
- if ρ is left-breaking:
  - add the arc $(v_\ell^i, v_{\ell+1}^i)$,
  - for each i' with $i \le i' \le j$ add the arcs $(v_\ell^{i'}, v_{\ell+1}^{i+1+j-i'})$;
- if ρ is not left-breaking:
  - if the endpoint of ρ and the $(i-1)$th block in S_ℓ have the same letter, then add for each i' with $i \le i' \le j$ the arcs $(v_\ell^{i'}, v_{\ell+1}^{i-1+j-i'})$,
  - if they have different letters, then add for each i' with $i \le i' \le j$ the arcs $(v_\ell^{i'}, v_{\ell+1}^{i+j-i'})$;
- if ρ is right-breaking, add the arc $(v_\ell^j, v_{\ell+1}^{j+\beta})$.

Note that for the case that ρ is left-breaking, we assume by Proposition 1 that the ith and jth block in S_ℓ have different letters and thus the endpoint of the reversal creates a new block in $S_{\ell+1}$. Moreover, for the right side of ρ we do not check explicitly whether the startpoint of ρ and the successor of its endpoint have the same letter, since this fact is completely determined when we know β (which can be directly deduced from the scaffold) and whether a block is "created" or "lost" at the left side of the reversal.

The construction is completed by adding for each $v_{k'+1}^i$, the arc $(v_{k'+1}^i, t)$ and setting the capacity $c(v_{k'+1}^i, t)$ to be exactly the length of the ith block in T.

Lemma 3. *Let S and T be balanced strings of length n, and let $G = (V, A, c, s, t)$ be a flow network constructed from a reversal scaffold from S to T. Then there is a sequence of reversals that transforms S into T and respects this scaffold if and only if G admits a flow of value n.*

Proof. Assume that there is such a sequence of reversals. Then the flow of value n is as follows. For each arc from s to the vertices v_1^i the flow value is exactly the capacity of the arc. Note that this means that the incoming flow at each vertex v_1^i is exactly the length of the corresponding block in S_1.

For each reversal ρ from S_ℓ to $S_{\ell+1}$, $1 \le \ell \le k$, we attribute the flow as follows. Assume that ρ starts at the xth letter of block i and ends at the yth letter of block j. Clearly, for vertices v_ℓ^i with outdegree one the value of the outgoing flow will be exactly the value of the ingoing flow. There are at most two vertices with outdegree two and these correspond exactly to the ith (v_ℓ^i) and jth block (v_ℓ^j) of S_ℓ. If v_ℓ^i has outdegree two, then one of its outneighbors

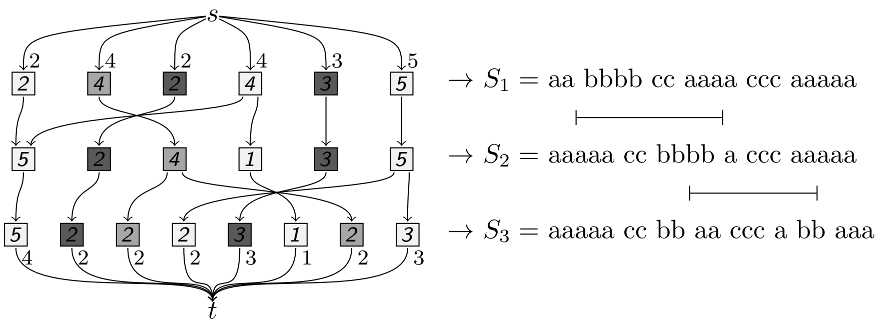

s
t
→ S₁ = aa bbbb cc aaaa ccc aaaaa
→ S₂ = aaaaa cc bbbb a ccc aaaaa
→ S₃ = aaaaa cc bb aa ccc a bb aaa

Figure 3: The flow network G constructed from the input strings and the scaffold given in Figure 1. The nodes (left) are the blocks in the scaffold (using the same drawing convention), with additional source s and sink t. The arcs represent how letters can move from one block to another. The capacities of the arcs are the lengths of the blocks in the original strings S and T. An italic number in each node represents the flow through this node in a maximum flow from s to t, which is interpreted (right) as the length of the corresponding block in string S_i. The sequence of reversals can be deduced from the strings (horizontal lines).

is $v^i_{\ell+1}$. Exactly $x-1$ units of the incoming flow are sent via $(v^i_\ell, v^i_{\ell+1})$ and the rest is sent via the other outgoing arc. If v^j_ℓ has outdegree two, then the flow is split in a similar manner: Writing δ_j for the length of v^j_ℓ, $\delta_j - y$ units of flow are sent via the arc between v^j_ℓ and $v^p_{\ell+1}$ where $p := b(S_{\ell+1}) - b(S_\ell) + j$ is defined such that the number of successor blocks of block j in S_ℓ is the same as the number of successor blocks of block p in $S_{\ell+1}$. The remaining flow is sent along the other outgoing arc of v^j_ℓ. Finally, the flow sent from the vertices corresponding to $S_{k'+1}$ is uniquely determined since each of these vertices has outdegree one.

We now show by induction that the above definition of flow is indeed valid. Then this direction of the claim follows since the flow has overall value of n. Assume by induction, that for S_ℓ, the value of flow incoming at each v^i_ℓ is equal to the length of the ith block of S_ℓ. We show that the same holds for $S_{\ell+1}$ (observe that the statement obviously holds for S_1). First, note that by the inductive assumption the amount of flow along the arcs from V_ℓ to $V_{\ell+1}$ is nonnegative. Second, for each vertex $v^i_{\ell+1}$ that has indegree one the claim holds: If the in-neighbor of $v^i_{\ell+1}$ has outdegree one, then there is a direct correspondence between the blocks. Thus, their size is the same which also holds for the value of incoming flow. If the in-neighbor of $v^i_{\ell+1}$ has outdegree two, then the block corresponding to the in-neighbor of $v^i_{\ell+1}$ has been split by the reversal from S_ℓ to $S_{\ell+1}$. Since $v^i_{\ell+1}$ has indegree one, it corresponds to a new block. The size of this block and the value of the incoming flow are equal. Finally, vertices $v^i_{\ell+1}$ with indegree two in $V_{\ell+1}$ correspond to blocks in $S_{\ell+1}$ that are merges of two blocks in S_ℓ. The size of these blocks is exactly the amount of flow sent along the corresponding arcs and thus the flow incoming in $v^i_{\ell+1}$ is equal to the sum of lengths of these two blocks and thus to the length of the ith block in $S_{\ell+1}$.

Summarizing, the amount of incoming flow at each vertex equals exactly the length of the corresponding block. This proves the forward direction of the claim.

For the reverse direction the same arguments apply in reverse to show that if the sum of the incoming flow at vertex set V_ℓ is exactly n, then there is a series of $\ell - 1$ reversals that transform S into a string that has exactly the blocks corresponding to the vertex set V_ℓ and for which each block has exactly the same length as the amount of flow incoming at the corresponding vertex. Now, the final argument to show correctness is that the string corresponding to $V_{k'+1}$ and whose block lengths are equal to the amount of flow incoming at each vertex is precisely T. Clearly, the order and letter content of the blocks is correct, since this was checked before building the flow network. Moreover, the amount of flow entering $V_{k'+1}$ is exactly n and the vertex capacities of the outgoing arcs sum up to n. Hence, the amount of flow incoming at each vertex is equal to the capacity of the outgoing arc. This capacity is by construction the length of the corresponding block in T. □

We now can terminate the proof of Theorem 5. By the above discussion, the algorithm constructs for unbounded alphabets $O((6b_{\max})^{4b_{\max}})$ many reversal scaffolds and $(6b_{\max})^{2b_{\max}}\,\text{poly}(n)$ reversal scaffolds in the case of constant size-alphabets. For each such reversal scaffold, the algorithm constructs in polynomial time the flow network G. By Lemma 3 it is sufficient to solve the maximum value flow problem on this network to decide whether there exists a sequence of reversals that respects the scaffold. This can be done in poly(n) time.

One possible approach to improve the above result would be to show that there is always an optimal sequence such that the number of blocks of every intermediate string never exceeds $b_{\max}$. However, this is not true, as stated by the following proposition.

Proposition 3. *For any integer $b \geq 5$, there exists a pair of balanced strings S and T such that $b_{\max} \geq b$, and such that any optimal reversal scenario goes through an intermediate string having strictly more than $b_{\max}$ blocks.*

Proof. Let $k \geq 2$ be any integer, and let $S := 011(10)^k$ and $T := 110(01)^k$. We thus have $b_{\max} = b(S) = b(T) = 2k + 1$. It can be easily seen that $rd(S, T) = 2$, since one reversal is not enough to obtain T from S, and since we can reach T from S using two reversals, for example as follows: $S = 011(10)^k \rightarrow 011\underline{(10)^k} = 011(01)^k \rightarrow \underline{011}(01)^k = 110(01)^k = T$. Moreover, no optimal solution goes through an intermediate string with less than $2k + 2$ blocks. Indeed, every optimal reversal in S must either start at the first or end at the last position, since both positions have a wrong letter, and since it is impossible to "fix" both positions in one reversal because they are both '0'. If the first reversal starts at the first position and does not increase the number of blocks, then it is either $\underline{0111}0(10)^{k-1}$ or $\underline{011(10)^i1}0(10)^{k-i-1}$, where $1 \leq i \leq k - 1$ (recall that reversals with both endpoints being the same letter do not need to be considered, by Proposition 1). Both reversals yield strings that cannot be transformed into T with one reversal. Similarly, if the reversal ends in the last position and does not increase the number of blocks, then it is either $011(10)^i\underline{(10)^{k-i}}$ with $1 \leq i \leq k-1$ or $0\underline{11(10)^k}$. Again, both reversals yield strings that cannot be transformed into T with one reversal. $\square$

Proposition 3 shows that asking for an optimal scenario where no string exceeds $b_{\max}$ blocks is too restrictive. However, it remains open whether relaxing the constraint to a maximum of $b_{\max} + O(1)$ blocks would be feasible.

The FPT algorithm we have presented above, and which led to Theorem 5, can be adapted to work for Signed String Reversal Distance, String Prefix Reversal Distance and Signed String Prefix Reversal Distance, as stated in the following theorem.

Theorem 6. String Prefix Reversal Distance *can be solved in* $(6b_{\max})^{4b_{\max}} \cdot \mathrm{poly}(n)$ *time;* Signed String Reversal Distance *and* Signed String Prefix Reversal Distance *can be solved in* $(b_{\max})^{O(b_{\max})} \cdot \mathrm{poly}(n)$ *time.*

We briefly sketch how the above FPT algorithm (leading to Theorem 5) can be adapted to these three problems. For String Prefix Reversal Distance, we have $k < 4b_{\max}$, as a consequence of Theorem 1. Further, every prefix reversal can increase or decrease the number of blocks by at most one. Hence, we obtain the following bound.

Lemma 4. *Let S' be a string such that there is an optimal sequence of prefix reversals from S to T in which S' is one of the strings produced by this sequence. Then $b(S') \leq 3b_{\max}$.*

Proof. Assume towards a contradiction that $b(S') > 3b_{\max}$. As each prefix reversal increases or decreases the number of blocks by at most one, it takes at least $2b_{\max}$ prefix reversals to "reach" S' from S and at least $2b_{\max}$ prefix reversals to "reach" T from S'. This contradicts the fact that in an optimal sequence of prefix reversals less than $4b_{\max}$ prefix reversals are necessary. $\square$

We can now bound the number of scaffolds that we have to consider. Clearly, the starting point i_ℓ of each reversal ρ_ℓ is 1, so we have to guess only the endpoint among at most $4b_{\max}$ blocks. Similarly, we only need to guess whether the prefix reversal is right-breaking. Hence, the number of scaffolds that needs to be considered is at most $(3{b_{\max}}^{4b_{\max}}) \cdot 2^{4b_{\max}}$. For each scaffold we again create a flow network in a way similar to the one described for String Reversal Distance, except that now Proposition 1 does not hold anymore, which makes the definitions of the arcs a bit more involved. However, the flow network remains of order and size $O((b_{\max})^2)$, which is sufficient for the FPT property to hold. Thus, altogether, we obtain the claimed running time.

For signed reversals and prefix reversals, we can assume $k = O(b_{\max})$ by Theorem 4. Since each signed reversal can decrease or increase the number of blocks by at most two and each signed prefix reversal by at most one, this bounds the number of blocks in any intermediate string between S and T to $O(b_{\max})$. Hence, the number of scaffolds to consider is ${b_{\max}}^{O(b_{\max})}$. For each scaffold we again create a flow network in a way similar to the unsigned cases. The only difference is that now we have to take signs into account when deciding whether a reversal "merges" two blocks or not (we omit the details). Altogether, we obtain the claimed running time.

4. Reversals on Strings with Small Alphabet

In this section, we study the complexity of our four problems in the case where the alphabet size is fixed. We start by giving new NP-hardness results, then we provide an exact singly-exponential algorithm for constant size alphabets and a generic type of distance measures on strings, that includes the four distances we are interested in.

Hardness Results for Restricted Cases. Here, we describe two reductions, one for the signed case and one for the unsigned case, that show hardness of both the reversal and prefix reversal problems in restricted cases. We start with the unsigned case.

Theorem 7. String Reversal Distance *and* String Prefix Reversal Distance *are NP-hard, even when restricted to binary strings where all 0-blocks have length at most 2 and all 1-blocks have length 1.*

Proof. We first show the result for String Reversal Distance, which we obtain by reducing from the NP-hard Sorting by Reversals (SBR) problem [5]. Given an instance of SBR, that is, a permutation π of elements in $[1; n]$, we construct the string $S(\pi)$, built on the binary alphabet $\Sigma = \{0, 1\}$, as follows. For each $1 \leq i \leq n$, we let $S_i := 01001(01)^{i+1}0010$, and define $S(\pi)$ as follows: $S(\pi) := (S_{\pi(1)})^{2n}(S_{\pi(2)})^{2n} \cdots (S_{\pi(n)})^{2n}$. Given an instance π (of length n) of SBR, we reduce to String Reversal Distance by first creating the two strings $S(I_n)$ and $S(\pi)$ as described above, where I_n denotes the identity permutation of length n. Now, let us describe several crucial properties of our reduction. First, for any $1 \leq i \leq n$, S_i is of length $2i + 11$ and is reversal invariant. Further, for any $1 \leq i \neq j \leq n$, S_i is not a substring of S_j, and the longest suffix of S_i which is also prefix of S_j has length 6 (it is 010010), which is strictly smaller than half the length of S_i and S_j. Hence, if, for any integers $i, j_1, \ldots, j_\ell$, S_i is a substring of $S_{j_1} \cdot S_{j_2} \cdot \ldots \cdot S_{j_\ell}$, then $i \in \{j_1, j_2 \ldots j_\ell\}$. Consequently, if $S_{j_1} \cdot S_{j_2} \cdot \ldots \cdot S_{j_\ell}$ contains substrings $S_{i_1}, S_{i_2} \ldots S_{i_h}$ in this order, then $(i_1, i_2 \ldots i_h)$ is a subsequence of $(j_1, j_2 \ldots j_\ell)$.

Now, let us show that our reduction preserves the distance. More precisely, if we let $k_S = rd(S(\pi), S(I_n))$ and k_π be the reversal distance from π to I_n, then we claim that $k_\pi = k_S$. First, we show $k_S \leq k_\pi$. Consider any sequence of k_π reversals sorting π, we show that there exists a corresponding sequence of k_π reversals sorting $S(\pi)$. For any reversal $\rho(i, j)$ of the sub-permutation of π ranging from $\pi[i]$ to $\pi[j]$, we reverse the substring of $S(\pi)$ ranging from the first $S_{\pi[i]}$ to the last $S_{\pi[j]}$: the resulting string is $S(\rho \circ \pi)$. In the end, we obtain $S(I_n)$ after k_π reversals.

We now show that $k_\pi \leq k_S$. Note that $k_S \leq k_\pi$ implies that $k_S < n$. Consider a sequence $\rho_1, \ldots, \rho_{k_S}$ of reversals sorting $S(\pi)$. For each $1 \leq i \leq n$, among the $2n$ copies of $S_{\pi[i]}$ in $S(\pi)$, at least one does not contain an endpoint of any reversal: we thus assign to each $1 \leq i \leq n$ an *untouched* copy of $S_{\pi[i]}$. Keeping track of how untouched copies are sorted, we can create a sequence of reversals sorting π as follows. For each reversal ρ_r, there exist i_r and j_r such that ρ_r reverses a string containing the i_rth to the j_rth untouched copies (ordered from left to right). Define the corresponding reversal over $\{1, \ldots, n\}$ as $\rho'_r = \rho(i_r, j_r)$. Let $\tau = \pi \circ \rho'_1 \ldots \circ \rho'_{k_S}$. Then the final string $S(I_n)$ contains the untouched copies of $S(\tau[1])$, $S(\tau[2])$, ..., $S(\tau[n])$ as substrings in this order. Recall that $S(I_n) = (S_1)^{2n}(S_2)^{2n} \cdots (S_n)^{2n}$, hence, using the property of strings S_i, sequence $(1, \ldots, 1, 2, \ldots, 2, \ldots, n, \ldots, n)$, where each number is repeated $2n$ times, contains $(\tau[1], \tau[2], \ldots, \tau[n])$ as a subsequence. Since τ is a permutation, τ is the identity I_n. Thus, the sequence of k_S reversals $\rho'_1, \ldots, \rho'_{k_S}$ transforms π into the identity, and $k_\pi \leq k_S$. This concludes the proof that String Reversal Distance is NP-hard even on binary alphabets.

Concerning the String Prefix Reversal Distance problem, the same result can be obtained, using a reduction and arguments almost identical to those above. The reduction is from the NP-hard Sorting by Prefix Reversals (SBPR) problem [4], and the fact that distances are preserved is shown similarly, except that the term "prefix reversal" should replace "reversal", and that ρ_r being a prefix reversal, then $i = 1$ and ρ'_r is also a prefix reversal. $\square$

Concerning the two signed problems, namely Signed String Reversal Distance and Signed String Prefix Reversal Distance, we have a similar result as Theorem 7.

Theorem 8. Signed String Reversal Distance *and* Signed String Prefix Reversal Distance *are NP-hard even restricted to signed unary strings, that is,* $|\Sigma| = 1$.

Proof. As in the previous theorem, the reduction for Signed String Reversal Distance is from Sorting by Reversals (SBR), and the reduction for Signed String Prefix Reversal Distance is from Sorting by Prefix Reversals (SBPR). Given a permutation π of elements in $[1; n]$, we construct the string $S(\pi)$, built from a unique letter a, as follows. For each integer $1 \leq i \leq n$, we let $S_i := +a(-a)^{i+1}(+a)^{i+1} - a$, and we let $S(\pi) = (S_{\pi(1)})^{2n}(S_{\pi(2)})^{2n} \cdots (S_{\pi(n)})^{2n}$. Given an instance π (of length n) of SBR (resp. SBPR), we reduce to Signed String Reversal Distance (resp. Signed String Prefix Reversal Distance) by creating the two strings $S(I_n)$ and $S(\pi)$ as described above. Note that each S_i is reversal invariant, and has length $2i + 4$. Moreover, for any $1 \leq i \neq j \leq n$, the longest suffix of S_i which is also prefix of S_j has length 2 (it is $+a - a$), which is strictly smaller than half the length of S_i and S_j. Hence, all the arguments developed in proof of Theorem 7 to show that the distances are preserved in the reduction still hold. $\square$

An Algorithm for Small Alphabets. So far, none of the known exact algorithms for String Reversal Distance or Signed String Reversal Distance achieves a singly-exponential running time of $2^{O(n)}$. We show that such a running time can be achieved for constant size alphabets and a generic type of distance measures on strings.

Definition 3. *A string distance d is* well-formed *if it is a metric and:*

- *For each string S of length n, the set containing exactly the strings T with $d(S,T) = 1$ can be computed in* $\operatorname{poly}(n)$ *time.*
- *All strings S and T with $d(S,T) = 1$ have the same length and the same alphabet.*
- *For any two strings S and T with $d(S,T) = k$ there exists a string S' with $d(S,S') = 1$ and $d(S',T) = k-1$.*

Theorem 9. *Let d be a well-formed string distance, and let S and T be two strings of length n over alphabet Σ. Then $d(S,T)$ can be computed in $|\Sigma|^n \cdot$* $\operatorname{poly}(n)$ *time.*

Proof. Let $k := d(S,T)$. The definition of being well-formed guarantees that there is a sequence of strings $S = S_1, S_2 \ldots S_{k-1}, S_k = T$ such that $d(S_i, S_{i+1}) = 1$, $1 \leq i < k$. Note that, also by the well-formed definition, each S_i has length n and has the same alphabet Σ as S. We now describe an algorithm that finds this sequence or one of equal length.

Construct a graph G as follows. For each string A of length n over alphabet Σ create a vertex v_A. Then, for each vertex v_A, construct the set $N_A := \{B \mid d(A,B) = 1\}$ of strings within distance one of A in polynomial time and, for each $B \in N_A$ insert the edge $\{v_A, v_B\}$ into G. Now compute a shortest path between v_S and v_T in G. This path has length k: Clearly it has length at most k, since the sequence of S_i's described above is a path from S to T in G. Furthermore, any path $(v_{A_1}, v_{A_2}, \ldots v_{A_{\ell-1}}, v_{A_\ell})$ of length ℓ in G between S and T, means that $d(A_1, A_\ell) \leq \ell$. This can be shown by induction on the length of the path: Obviously, the claim holds for paths of length one. Now assume that the claim holds for paths of length $\ell - 1$. Then it also holds for paths of length ℓ, since $d(A_2, A_\ell) \leq \ell - 1$, $d(A_1, A_2) = 1$, and $d(A_1, A_\ell) \leq d(A_1, A_2) + d(A_2, A_\ell)$ since d is a metric by definition.

The running time follows from the fact that a shortest path in an unweighted graph G can be computed in $O(|V| + |E|)$ time via breadth-first search and the fact that G has $|\Sigma|^n$ vertices and $|\Sigma|^n \operatorname{poly}(n)$ edges. □

5. Conclusion

In this paper, we have studied the string reversal distance problem, under its four variants depending on the nature of reversals (prefix or not), and the way strings are modeled (signed or not). Our main goal was to provide new algorithmic results when the maximum number of blocks $b_{\max}$ of the input strings and/or the size of the alphabet Σ on which they are built is small. We showed that the four problems are fixed-parameter tractable with respect to $b_{\max}$. In the process, we showed a series of upper bounds on the diameter of the studied distances, which only depend on $b_{\max}$ and $|\Sigma|$. We also proved NP-hardness results for the four problems, even in very restricted cases, thus narrowing the gaps between polynomial classes of instances and NP-hard ones. We finally showed an exact algorithm that achieves a running time of $|\Sigma|^n \cdot \text{poly}(n)$ for many string distances, including the ones under consideration in this article.

Our work leads to several open questions, which we deliver here in arbitrary order. First, is the reversal diameter for strings $b_{\max} - 1$? If so, this would generalize the upper bound of $n-1$ on the diameter for the reversal distance between permutations [1]. If not, is an upper bound of $b_{\max} + O(|\Sigma|)$ achievable? Further, does String Reversal Distance admit a polynomial kernel for $b_{\max}$, that is, can it be reduced in polynomial time to an equivalent instance of String Reversal Distance with $n \leq \text{poly}(b_{\max})$? Also, can we solve String Reversal Distance in time $O((|\Sigma| - \epsilon)^n)$, for some strictly positive ϵ? In particular, can we solve String Reversal Distance on binary strings in $O(c^n)$ time for $c < 2$? Finally, we repeat the open question posed in [17], and whose answer was conjectured to be true: Can the reversal sorting problem (that is, computing the distance to a given grouped string) be solved in polynomial time for every fixed alphabet size?

References

[1] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[2] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In R. H. Möhring and R. Raman, editors, *Proceedings of the 10th Annual European Symposium on Algorithms (ESA '12)*, volume 2461 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2002.

[3] L. Bulteau, G. Fertin, and C. Komusiewicz. Reversal distances for strings with few blocks or small alphabets. In A. S. Kulikov, S. O. Kuznetsov, and P. A. Pevzner, editors, *Proceedings of the 25th Annual Symposium on Combinatorial Pattern Matching (CPM '14)*, volume 8486 of *Lecture Notes in Computer Science*, pages 50–59. Springer, 2014.

[4] L. Bulteau, G. Fertin, and I. Rusu. Pancake flipping is hard. *Journal of Computer and System Sciences*, 81(8):1556–1574, 2015.

[5] A. Caprara. Sorting by reversals is difficult. In *Proceedings of the 1st Annual International Conference on Research in Computational Molecular Biology (RECOMB '97)*, pages 75–83, 1997.

[6] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM*

Transactions on Computational Biology and Bioinformatics, 2(4):302–315, 2005.

[7] B. Chitturi. A note on complexity of genetic mutations. *Discrete Mathematics, Algorithms and Applications*, 3(3):269–286, 2011.

[8] B. Chitturi and I. H. Sudborough. Prefix reversals on strings. In H. R. Arabnia, Q. Tran, R. Chang, M. He, A. Marsh, A. M. G. Solo, and J. Y. Yang, editors, *International Conference on Bioinformatics & Computational Biology, BIOCOMP 2010, July 12-15, 2010, Las Vegas Nevada, USA, 2 Volumes*, pages 591–598. CSREA Press, 2010.

[9] B. Chitturi, W. Fahle, Z. Meng, L. Morales, C. O. Shields, I. H. Sudborough, and W. Voit. An $(18/11)n$ upper bound for sorting by prefix reversals. *Theoretical Computer Science*, 410(36):3372–3390, 2009.

[10] D. A. Christie. *Genome Rearrangement Problems.* PhD thesis, University of Glasgow, 1998.

[11] D. A. Christie and R. W. Irving. Sorting strings by reversals and by transpositions. *SIAM Journal on Discrete Mathematics*, 14(2):193–206, 2001.

[12] D. S. Cohen and M. Blum. on the problem of sorting burnt pancakes. *Discrete Applied Mathematics*, 61(2):105–120, 1995.

[13] G. Fertin, A. Labarre, I. Rusu, E. Tannier, and S. Vialette. *Combinatorics of Genome Rearrangements.* Computational Molecular Biology. MIT Press, 2009.

[14] J. Fischer and S. W. Ginzinger. A 2-approximation algorithm for sorting by prefix reversals. In G. S. Brodal and S. Leonardi, editors, *Proceedings of the 13th Annual European Symposium on Algorithms (ESA '05)*, volume 3669 of *Lecture Notes in Computer Science*, pages 415–425. Springer, 2005.

[15] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):1–27, 1999.

[16] C. A. J. Hurkens, L. van Iersel, J. Keijsper, S. Kelk, L. Stougie, and J. Tromp. Prefix reversals on binary and ternary strings. *SIAM Journal on Discrete Mathematics*, 21(3):592–611, 2007.

[17] A. Radcliffe, A. Scott, and E. Wilmer. Reversals and transpositions over finite alphabets. *SIAM Journal on Discrete Mathematics*, 19(1):224, 2006.

[18] G. Watterson, W. Ewens, T. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theoretical Biology*, 99(1):1 – 7, 1982.