# Reversal Distances for Strings with Few Blocks or Small Alphabets

Laurent Bulteau, Guillaume Fertin, Christian Komusiewicz

# Reversal Distances for Strings with Few Blocks or Small Alphabets

Laurent Bulteau[1⋆], Guillaume Fertin[2], and Christian Komusiewicz[2⋆⋆]

[1] Institut für Softwaretechnik und Theoretische Informatik, TU Berlin
{l.bulteau}@campus.tu-berlin.de
[2] Université de Nantes, LINA - UMR CNRS 6241, France.
{guillaume.fertin,christian.komusiewicz}@univ-nantes.fr

**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 $T$ 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 (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 maximal substring such that all letters in the substring are equal), and the alphabet size $\Sigma$. For instance, we show that Signed String Reversal Distance and Signed String Prefix Reversal Distance are NP-hard even if the input strings have only *one* letter.

## 1 Introduction

Many problems studied in the realm of comparative genomics concern genome rearrangements, in which, given two genomes $G_1$ and $G_2$ and a set $\mathcal{S}$ of operations (called rearrangements) on genomes, the question is to compute the smallest number of rearrangements that allows to obtain $G_2$, starting from $G_1$, and using only operations from $\mathcal{S}$ (see for instance [8] for an extensive survey). One of the most studied, and historically one of the firstly described [14] such rearrangement is the *reversal*, which consists in taking a contiguous sequence of a genome, reverse its order, and reincorporate it at the same location. This gave rise to the Sorting by Reversals (SBR) (resp. Sorting by Signed Reversals (SBSR)) problem, in which a genome is represented by a permutation (resp. signed permutation) whose elements are genes. In the signed version of the problem each position of the permutation is additionally labeled with a sign + or − and a reversal $\rho$ not only reverses the order, but also inverts the signs of all the elements involved in $\rho$. The main complexity results concerning

⋆ Supported by the Alexander von Humboldt Foundation
⋆⋆ Supported by a post-doctorial grant funded by the Région Pays de la Loire

these two problems are the following: SBR is NP-hard [4] and the best current approximation ratio is 1.375 [2], while SBSR is polynomial [1]. Another variant consists in using *prefix reversals* only, that is, each reversal must contain the first letter of the string it is applied to. The unsigned (resp. signed) corresponding problem is called SBPR (resp. SBSPR). The SBPR problem has been recently shown to be NP-hard [3], and the best current approximation ratio is 2 [9]. On the other hand, the complexity of SBSPR is still open.

In biological applications, however, genomes of related species often contain many homologous genes. In this case, the genomes cannot be modeled by permutations, but rather by (signed) strings [5]. Hence, a natural (and more biologically relevant) extension of SBR is the String Reversal Distance problem, formally defined (in its decision version) as follows:

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

Besides, Signed String Reversal Distance will denote the signed version of the above problem. If we allow prefix reversals only, the extension of SBPR to strings is defined as follows:

String Prefix Reversal Distance
**Input:** Two strings $S$ and $T$ over alphabet $\Sigma$ 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. Any of these four problems is only nontrivially posed if $S$ and $T$ have the same letter content, that is, for each letter the number of its occurrences in $S$ equals the number of its occurrences in $T$. We call such strings *balanced*. In the remainder of the paper, we assume that $S$ and $T$ are balanced. A *block* is a maximal substring such that all letters in the substring are equal (and have the same sign, if strings are signed). For any instance, 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 $n \geq b_{\max} \geq b_{\min} \geq |\Sigma|$.

In this paper, we 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 $\Sigma$.

*Known Results.* Computing the reversal distance between binary strings is NP-hard via reduction from 3-Partition [6, 7]. A second proof uses a reduction from Sorting by Reversals [13]. Computing the prefix reversal distance between binary strings is NP-hard [11]. Sorting a binary string $S$ (to $0^p1^{n-p}$) is solvable in polynomial time [6, 7]. This result was later generalized to the case of ternary strings [13]. Hurkens et al. [11] considered the problem of "grouping" a string by prefix reversals, that is, find a shortest sequence of reversals from

a string $S$ to any string that has $|\Sigma|$ blocks. Concerning the diameter, that is, the maximum distance between any length-$n$ strings, it was first shown that the reversal diameter for binary strings of length $n$ is $\lfloor n/2 \rfloor$ [6, 7]. Later, this result was generalized to fixed alphabets of arbitrary size. More precisely, the reversal diameter of strings of length $n$ is $n - \max_{a \in \Sigma} \#(a)$ where $\#(a)$ denotes the number of occurrences of letter $a$ in either input string. Signed String Reversal Distance has applications in the identification of orthologous genes across two genomes [5, 10, 12]. Signed String Reversal Distance is NP-hard if each letter occurs at most twice [5] and for binary signed strings [13].

*Our Results.* 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. Then, in Section 3 we show the aforementioned algorithm for the parameter $b_{\max}$. In Section 4 we describe a reduction from SBR that yields several hardness results. First, it shows that Signed String Reversal Distance and Signed String Prefix Reversal Distance remain NP-hard over unary alphabet. This strengthens the previous hardness result by Radcliffe et al. [13] who showed hardness for binary alphabets. Second, 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 one and each 1-block has length at most two. 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.

*Preliminaries.* 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 $S[1] \cdots S[i-1]S[i]S[i+1] \cdots S[j-1]S[j]S[j+1] \cdots S[n]$ into $S[1] \cdots S[i-1]S[j]S[j-1] \cdots S[i+1]S[i]S[j+1] \cdots S[n]$. We denote the string that results from applying reversal $\rho$ to $S$ by $S \circ \rho$. We use $b(S)$ to denote the number of blocks of a string $S$. For a (signed) string $S$ we denote by $\underline{S}$ the string that is obtained by the reversal $\rho(1, |S|)$. The following simple observation will be useful in a later part of this work.

**Proposition 1.** *There is a shortest sequence $\rho_1, \rho_2, \ldots, \rho_\ell$ of reversals from $S$ to $T$ such that the start and endpoint of each $\rho_i$ have different letters.*

## 2 Upper Bounds on the Reversal Diameter

In this section, we upper-bound the reversal diameter of balanced strings based on the number of blocks in these strings. To this end, 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.

**Lemma 1.** *Let $S$ be a string with $b$ blocks over alphabet $\Sigma$. There exists a string $S_{\text{g}}$ with $|\Sigma|$ blocks such that $S$ can be transformed into $S_{\text{g}}$ by at most $b - |\Sigma|$ reversals and by at most $b$ prefix reversals.*

*Proof.* The claim is obviously true if $b = |\Sigma|$. Now, assume by induction that the claim holds for all strings $S'$ with $b' < b$ blocks. Since $b > |\Sigma|$ there is one letter $a$ that appears in two blocks. By applying the reversal to $S$ that starts at the leftmost letter of the first block containing $a$ and ends at the letter before the second block containing $a$, one obtains a string $S'$ with $b-1$ blocks. By induction, there is a grouped string $S_{\mathrm{g}}$ such that the reversal distance from $S'$ to $S_{\mathrm{g}}$ is at most $b-1$.

For prefix reversals, we use a similar greedy strategy. 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 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. The overall number of prefix reversals that are applied until a grouped string is reached is thus at most $b - |\Sigma| + |\Sigma| - 1$. □

We now use this upper bound 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.** *Two balanced strings $S$ and $T$ with $b_{max}$ and $b_{min}$ blocks, respectively, can be transformed into each other by at most $b_{max} + b_{min} - |\Sigma| - 1$ reversals and at most $b_{max} + b_{min} + 2|\Sigma| - 3$ prefix reversals.*

For the reversal case, we can also obtain a bound of the type $b_{\max} + O(|\Sigma|^2)$.

**Theorem 2.** *Two balanced strings $S$ and $T$ each with at most $b_{max}$ blocks can be transformed into each other by at most $b_{max} + |\Sigma|^2 - 2|\Sigma|$ reversals.*

*Proof.* If $b_{\max} < |\Sigma|^2 - |\Sigma| + 2$, then the claim is true as $d(S,T) \leq b_{\max} + |\Sigma|^2 - 2|\Sigma|$ 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. Let $S'$ be the resulting string. By the inductive hypothesis, $d(S',T) \leq b_{\max} - 1 + |\Sigma|^2 - 2|\Sigma|$. Since $d(S,S') = 1$, 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 letters, say $a$ and $b$, such that $U$ contains the substring $ab$ twice. This can be seen by considering a directed multigraph with the vertex set $\Sigma$ in which we add an edge $(u,v)$ for each substring $uv$, $u \neq v$ of $U$. Any block change corresponds to an edge in this graph. Now there are at least $|\Sigma|^2 - |\Sigma| + 1$ block changes in $U$. Hence, the multigraph has $|\Sigma|^2 - |\Sigma| + 1$

edges. Since a simple directed graph can have at most $|\Sigma|^2 - |\Sigma|$ edges, there is one pair of vertices $u$ and $v$, for which the edge $(u, v)$ is contained twice in this multigraph.

By the above, $S$ has two 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, $d(S', T') = b_{\max} - 2 + |\Sigma|^2 - 2|\Sigma|$. Together with the two additional reversals on $S$ and $T$ we obtain the bound on the number of reversals also in this case. □

## 3 An Algorithm for Strings with Few Blocks

We now show how to solve String Reversal Distance in $(b_{\max})^{O(b_{\max})} \cdot \text{poly}(n)$ time. Our algorithm consists of two main steps: first, "guess" between which blocks each of the reversals takes place. These guesses fix the structure of the reversals and we will thus call a sequence of at most $k$ of those guesses a *scaffold*. A scaffold would completely describe a sequence of reversals if one would additionally specify the precise positions at which the reversal starts or ends in each of the blocks. Since the blocks can be very long compared to the number of blocks, we can not branch into all possible cases for these positions. However, we guess whether the startpoint of the reversal is the first position of a block and whether the endpoint of the 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 *$((i_1, j_1, B_1), (i_2, j_2, B_2), \ldots, (i_k, j_k, B_k))$ 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 $S_i \rho_i = S_{i+1}$* respects *a scaffold if for each $\ell$, $1 \leq \ell \leq k$ we have that*

- *the startpoint of $\rho_\ell$ is in the $i_\ell$-th block of $S_\ell$,*
- *the endpoint of $\rho_\ell$ is in the $j_\ell$-th block of $S_\ell$,*
- *$\rho_\ell$ is left-breaking if and only if $L \in B_\ell$, and*
- *$\rho_\ell$ is right-breaking if and only if $R \in B_\ell$.*

For each possible scaffold, the algorithm now aims to compute whether one can assign two numbers to each reversal to obtain a sequence of reversals that respects the scaffold and transforms $S$ into $T$. This is done by computing a maximum flow on an auxiliary graph.

First, we bound the number of different scaffolds that need to be considered. 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 $2b_{\max} - 2$ reversal-triples. The algorithm branches for each such reversal-triple into the possible choices for $i_\ell$ and $j_\ell$ and whether the reversal shall be left- or right-breaking. By the above lemma, it needs to perform at most $2b_{\max} - 2$ 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, $S'$ has at most $\min\{2b_{max} + b_{min} - |\Sigma| - 1, 2b_{max} + |\Sigma|^2 - 2|\Sigma|\}$ blocks.*

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 $i_\ell$ and $j_\ell$. 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

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

Consider one such scaffold, assume there is a sequence of reversals that respects the scaffold, and let $S_i := S_{i-1} \circ \rho_i$, $i \leq k'+1$, denote the string obtained after the $i$-th reversal. We show that the number and order of blocks of each $S_i$ is completely fixed by the reversal scaffold. First, consider $S_1 := S$ and let $\delta_\ell$ denote the number of letters in the $\ell$-th block of $S_1$, let $\sigma_i$ denote the letter of the $\ell$-th block and assume that $S$ has $b_{\max}$ blocks. Furthermore, assume that $i_1$ is in the $i$-th block of $S_1$ and $j_1$ is in the $j$-th 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 $i$-th and $j$-th blocks):

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

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 $y < \delta_i$ or $y = \delta_i$. Consequently, it is fixed whether the letter preceding the endpoint of the reversal in $S_2$ is $\sigma_i$ or whether this letter is $\sigma_{i-1}$. Similarly, it is fixed whether the letter succeeding the startpoint of the reversal in $S_2$ is $\sigma_j$ or whether it is $\sigma_{j+1}$. Therefore, we know whether the borders of the reversal are start 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 $i$-th block in $T$ is different from the letter from the $i$-th block in $S_{k'+1}$, then we can discard the reversal scaffold. Thus, it now remains to check whether the reversal scaffold can produce blocks of the correct size.

One possible way of checking whether this is indeed true would be to introduce a variable for the length of each block in each $S_i$ and then introduce equations that model the dependencies between the blocks. For instance 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 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 with $O((b_{\max})^2)$ vertices as follows.

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

Assume that the reversal $\rho$ is fixed to start within the $i$-th block of $S_\ell$ and end in the $j$-th block of $S_\ell$. Furthermore, let $\beta$ denote the difference between the number of blocks in $S_{\ell+1}$ and in $S_\ell$. Then, add the following edges, with unbounded capacity, to $G$:

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

Note that for the case that $\rho$ is left-breaking, we assume by Proposition 1 that the $i$-th and $j$-th block in $S_\ell$ have different letters and thus the endpoint of the reversal creates a new block in $S_{\ell+1}$. Note that for the right side of $\rho$ we do not check explicitly whether the startpoint of $\rho$ and the successor of its endpoint have the same letter, since this fact is completely determined when we know $\beta$ (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 $v^i_{k'+1}$, the edge $(v^i_{k'+1}, t)$ and setting the capacity $c(v^i_{k'+1}, t)$ to be exactly the length of the $i$-th block in $T$.

**Lemma 3.** *Let $N = (V, E, c, s, t)$ be a flow network constructed from a reversal scaffold as described above. Then there is a sequence of reversals that transforms $S$ into $T$ and respects the reversal scaffold if and only if $N$ admits a flow of value $n$.*

**Theorem 3.** String Reversal Distance *can be solved in* $(6b_{max})^{4b_{max}} \operatorname{poly}(n)$ *time on arbitrary strings and in* $(6b_{max})^{2b_{max}} \operatorname{poly}(n)$ *time if* $|\Sigma|$ *is constant.*

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, the instance with $S := 011100100$ and $T := 110001001$ is a counterexample. An optimal solution contains exactly two reversals as shown by the example $011100100 \to 011\underline{100100} = 011001001 \to \underline{011}001001 = 110001001$. This solution creates an intermediate string with six blocks but the input strings have only five blocks. There is also no solution that has less than six blocks in an intermediate string which can be shown by a case distinction. The algorithm can be adapted to work for the other problem variants as well.

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

## 4 Reversals on Strings with Small Alphabet

*Hardness Results for Restricted Cases.* 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. For reversal problems the reduction is from Sorting by Reversals, for prefix reversal problems the reduction is from Sorting by Prefix Reversals. Recall that both problems are NP-hard [3, 4].

Given an instance of SBR or SBPR, replace each permutation as follows.

**Construction 1: Signed (Prefix) Reversals.** For each integer $i > 0$, let $S_i := +a(-a)^{i+1}(+a)^{i+1} - a$. Note that each $S_i$ has length $2i + 4$.

**Construction 2: Unsigned (Prefix) Reversals.** For each $i > 0$, let $S_i := 01001(01)^{i+1}0010$. Note that each $S_i$ has length $2i + 11$.

In both constructions, let $S(\pi) := (S_{\pi(1)})^{2n}(S_{\pi(2)})^{2n} \cdots (S_{\pi(n)})^{2n}$ for any permutation $\pi$. Given an input to SBR or SBPR with permutation $\pi$, the reduction creates the two strings $S(I_n)$ and $S(\pi)$ as defined above ($I_n$ denotes the identity permutation of length $n$).

The following observations apply to both constructions. First, the reversed string of $S_i$ is $S_i$ itself. Further, for any $i \neq j$, $S_i$ is not a substring of $S_j$, and the longest suffix of $S_i$ which is also prefix of $S_j$ has length two in Construction

1 (it is $+a-a$) and length six in Construction 2 (it is 010010). In both cases, this 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 $S_{j_1} \cdot S_{j_2} \cdot \cdots \cdot S_{j_\ell}$, then $i \in \{j_1, \ldots, j_\ell\}$. Moreover, if $S_{j_1} \cdot S_{j_2} \cdot \cdots \cdot S_{j_\ell}$ contains substrings $S_{i_1}$, $S_{i_2}$,..., $S_{i_h}$ in this order, then $(i_1, i_2, \ldots, i_h)$ is a subsequence of $(j_1, j_2, \ldots, j_\ell)$.

**Theorem 5.** Signed String [Prefix] Reversal Distance *is NP-hard even when* $|\Sigma| = 1$. String [Prefix] Reversal Distance *is 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.* Let $S(I_n)$ and $S(\pi)$ be constructed from a permutation $\pi$ as described above. Let $k_S$ be the [prefix] reversal distance from $S(\pi)$ to $S(I_n)$, and $k_\pi$ be the [prefix] reversal distance from $\pi$ to $I_n$. We show that $k_\pi = k_S$.

First, we show $k_S \leq k_\pi$. Consider any sequence of $k_\pi$ [prefix] reversals sorting $\pi$, we show that there exists a corresponding sequence of $k_\pi$ [prefix] reversals sorting $S(\pi)$. For any [prefix] reversal $\rho$ of the sub-permutation of $\pi$ 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 $k_\pi$ [prefix] 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 [prefix] 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 $i$ an *untouched* copy of $S_{\pi[i]}$. For each [prefix] reversal $\rho_r$, there exists $i$ and $j$ such that $\rho_r$ reverses a string containing the $i$th to the $j$th untouched copies (ordered from left to right). Let $\rho'_r = (1, \ldots, i-1, j, j-1, \ldots, i+1, i, j+1, \ldots, n)$. Note that if $\rho_r$ is a prefix reversal, then $i = 1$ and $\rho'_r$ is also a prefix reversal.

Let $\tau = \rho'_{k_S} \circ \ldots \circ \rho'_1 \circ \pi$, 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. By definition of $S(I_n)$ and using the property of strings $S_i$, sequence $(1, \ldots, 1, 2, \ldots, 2, \ldots, n, \ldots, n)$ contains $\tau[1]$, $\tau[2]$, ..., $\tau[n]$ as a subsequence. Since $\tau$ is a permutation, $\tau$ is the identity $I_n$. Thus, the sequence of $k_S$ [prefix] reversals $\rho'_1$,..., $\rho'_{k_S}$ transforms $\pi$ into the identity, and $k_\pi \leq k_S$. □

*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. Call a string distance $d$ *well-formed* if it has the following properties: 1) For each string $S$ of length $n$, the set containing exactly the strings $T$ with $d(S,T) = 1$ can be computed in poly$(n)$ time. 2) All strings $S$ and $T$ with $d(S,T) = 1$ are balanced. 3) 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 6.** *Let $d$ be a well-formed string distance, and let $S$ and $T$ be two strings of length $n$ over alphabet $\Sigma$. Then $d(S,T)$ can be computed in $|\Sigma|^n \cdot$* poly$(n)$ *time.*

## 5 Conclusion

Our work leads to several open questions. Is the reversal diameter for strings $b_{\max}-1$? If yes, this would generalize the upper bound of $n-1$ on the diameter for the reversal distance between permutations. 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 $n \leq \text{poly}(b_{\max})$? Finally, can we solve String Reversal Distance in time $O((|\Sigma|-\epsilon)^n)$? In particular, can we solve String Reversal Distance on binary strings in $O(c^n)$ time for $c < 2$?

## References


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

[2] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proc. 10th ESA*, volume 2461 of *LNCS*, pages 200–210. Springer, 2002.

[3] L. Bulteau, G. Fertin, and I. Rusu. Pancake flipping is hard. In *Proc. 37th MFCS*, volume 7464 of *LNCS*, pages 247–258. Springer, 2012.

[4] A. Caprara. Sorting by reversals is difficult. In *Proc. 1st RECOMB*, pages 75–83, 1997.

[5] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM T. Comput. Bi.*, 2(4):302–315, 2005.

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

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

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

[9] J. Fischer and S. W. Ginzinger. A 2-approximation algorithm for sorting by prefix reversals. In *Proc. 13th ESA*, volume 3669 of *LNCS*, pages 415–425. Springer, 2005.

[10] Z. Fu, X. Chen, V. Vacic, P. Nan, Y. Zhong, and T. Jiang. MSOAR: A high-throughput ortholog assignment system based on genome rearrangement. *J. Comput. Biol.*, 14(9):1160–1175, 2007.

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

[12] T. Jiang. Some algorithmic challenges in genome-wide ortholog assignment. *J. Comput. Sci. Technol.*, 25(1):42–52, 2010.

[13] A. Radcliffe, A. Scott, and E. Wilmer. Reversals and transpositions over finite alphabets. *SIAM J. Discrete Math.*, 19(1):224, 2006.

[14] G. Watterson, W. Ewens, T. Hall, and A. Morgan. The chromosome inversion problem. *J. Theor. Biol.*, 99(1):1 – 7, 1982.