REVERSALS AND TRANSPOSITIONS OVER FINITE ALPHABETS[*]

A. J. RADCLIFFE[†], A. D. SCOTT[‡], AND E. L. WILMER[§]

Abstract. Extending results of Christie and Irving, we examine the action of reversals and transpositions on finite strings over an alphabet of size k. We show that determining reversal, transposition, or signed reversal distance between two strings over a finite alphabet is NP-hard, while for "dense" instances we give a polynomial-time approximation scheme. We also give a number of extremal results, as well as investigating the distance between random strings and the problem of sorting a string over a finite alphabet.

Key words. strings, sorting, genome comparison, reversals, transpositions, NP-complete problems, MAX-SNP hardness, approximation algorithms

AMS subject classifications. 68R05, 68R15, 68Q17

DOI. 10.1137/S0895480103433550

Introduction. As a result of interest in both modelling large-scale genome changes and fundamental questions on the combinatorics of sequences, rearrangement operations, including transpositions, reversals, and signed reversals, have recently been the focus of intense combinatorial, algorithmic, and complexity-theoretic study. These superficially similar sequence operations turn out to have significantly different properties. Most previous work has concentrated on applying sequence operations to permutations. However, the analysis of operations on strings over finite alphabets was raised by Pevzner and Waterman [26] and investigated by Christie and Irving [9]. The study of sequence operations on strings may also be of some practical interest; for a recent example, see, for instance, Skaletsky et al. [27] on the roles played by palindromes and repetitive segments in the Y-chromosome.

The operations under consideration all act on strings $\alpha = a_1 \cdots a_n$ of length $|\alpha| = n$. The *reversal* R_{ij}, where $i < j$, reverses the substring $a_i \cdots a_j$, so that

$$R_{ij}(a_1 \cdots a_n) = a_1 \cdots a_{i-1} a_j a_{j-1} \cdots a_i a_{j+1} \cdots a_n.$$

The *transposition* T_{ijk}, where $i < j < k$, exchanges the substrings $a_i \cdots a_j$ and $a_{j+1} \cdots a_k$, so

$$T_{ijk}(a_1 \cdots a_n) = a_1 \cdots a_{i-1} a_{j+1} \cdots a_k a_i \cdots a_j a_{k+1} \cdots a_n.$$

The *pancake flip* or *prefix reversal* P_i reverses the substring $a_1 \cdots a_i$, so

$$P_i(a_1 \cdots a_n) = a_i a_{i-1} \cdots a_1 a_{i+1} \cdots a_n.$$

Signed reversals work on strings where each character has an orientation: we use $\overline{a}$ to denote the opposite orientation of a, and note that $\overline{\overline{a}} = a$. The *signed reversal* S_{ij} is

---

[†]Department of Mathematics, University of Nebraska-Lincoln, Lincoln, NE 68588-0323 (jradclif@math.unl.edu). This author's research was supported by NSF grant DMS 9401351.

[‡]Department of Mathematics, University College London, Gower Street, London WC1E 6BT, UK (scott@math.ucl.ac.uk).

[§]Department of Mathematics, Oberlin College, Oberlin, OH 44074 (elizabeth.wilmer@oberlin.edu). This author's research was supported by an NSF-AWM Mentoring Travel Grant.

the same as R_{ij}, except that the reversed elements change orientation:

$$S_{ij}(a_1, \dots , a_n) = a_1 \cdots a_{i-1}\overline{a_j}\,\overline{a_{j-1}} \cdots \overline{a_i}a_{j+1} \cdots a_n.$$

As the collections of reversals, transpositions, and pancake flips each generate the symmetric group S_n and are closed under taking inverses, they therefore induce metrics d_{rev}, d_{tr}, d_{pf} on S_n, where $d_X(\alpha, \beta)$ is the minimum length of a sequence of operations of type X transforming α to β. Signed reversals generate the larger hyperoctahedral group of signed permutations and define a metric $d_{\overline{\mathrm{rev}}}$. All these metrics can be defined for strings over finite alphabets, provided we restrict ourselves to *compatible* pairs, namely pairs of strings that have the same number of occurrences of each symbol.

Extremal investigation of sequence operations has concentrated on the diameter of the symmetric (or, for signed reversals, hyperoctahedral) group. Bafna and Pevzner [2] showed that the reversal diameter of S_n is $n - 1$, while Meidanis, Walter, and Dias [25] showed that the signed reversal diameter of the group of signed permutations is at most $n + 1$. For transpositions, Bafna and Pevzner [3] showed that the diameter lies between $n/2 + 1$ and $3n/4$. Eriksson et al. [10] improved the upper bound to $\lfloor (2n - 2)/3 \rfloor$ for $n \geq 9$. For pancake flips, Gates and Papadimitriou [14] showed that the diameter lies between $17n/16$ and $(5n + 5)/3$; Heydari and Sudborough [19] improved the lower bound to $15n/14$. Christie and Irving [9] investigated these problems for the set of binary strings: they showed that the maximum reversal and transposition distances between two compatible binary strings of length n is $\lfloor n/2 \rfloor$, and noted that there does not appear to be an easy generalization of these results to strings over alphabets of size $k > 2$. In section 1, we prove such a generalization for reversal distance between strings over alphabets of size k; furthermore, we determine the diameter of every equivalence class of strings (under the relation of compatibility).

In section 2, we consider the distance between random strings. Two randomly chosen permutations are typically reversal distance $\Theta(n)$ apart [2]. We show that strings from a k-letter alphabet with fixed fractions of letters of each type are typically at reversal distance $\Theta(n/\log n)$. Our arguments extend to any other class of string operations with a bounded number of cutpoints at each step and a linear bound on diameter in the permutation case, such as transpositions or pancake flips.

The complexity of calculating the distance between two permutations depends on the type of operations used. Caprara [7] showed that determining reversal distance is NP-hard, while Berman and Karpinski [6] (see also [22]) showed that the problem is MAX-SNP hard. The signed reversal distance, by contrast, can be found in polynomial time: algorithms were given by Hannenhalli and Pevzner [17], Berman and Hannenhalli [4], and Kaplan, Shamir, and Tarjan [21]. The complexity of finding transposition distance between permutations remains open. For binary strings, Christie and Irving [9] showed that reversal distance remains NP-hard for binary strings, but left open the difficulty of finding transposition distance. In section 3, we show that signed reversal distance and transposition distance are both NP-hard for binary strings (and hence for strings over any finite alphabet). This is the first hardness result for transposition distance; together with the difficulty of signed reversal distance, it suggests that these problems may be harder over finite alphabets.

In section 4, we turn to the problem of approximating the distance between pairs of strings. Karpinski [22] showed that it is NP-hard to approximate the reversal distance between two permutations to within any factor less than $1237/1236$. A number

of authors have given approximation algorithms: Kececioglu and Sankoff [23] gave a 2-approximation algorithm, Bafna and Pevzner [2] gave a 1.75-approximation algorithm, Christie [8] gave a 1.5-approximation algorithm, and recently, Berman, Hannenhalli, and Karpinski [5] gave a 1.375-approximation algorithm. Bafna and Pevzner [3] have also given a 1.5-approximation algorithm for transposition distance. For strings over a finite alphabet, Pevzner and Waterman [26, Problem 4], raised the problem of finding an approximation algorithm for determining reversal distance. It follows from Karpinski's results [22] and the results of section 2 that it is NP-hard to approximate reversal distance between strings to within any factor better than 1237/1236. However, we show that for *dense* instances (pairs of strings at distance $\Omega(n)$) there is a polynomial-time approximation scheme. Similar results hold for approximating signed reversal, prefix reversal, or transposition distance between two strings, and we conjecture that analogous results should hold for calculating the distance between permutations.

For permutations, a sorting algorithm suffices to determine the distance between an arbitrary pair of strings—just relabel the entries of both so that one string is sorted. For strings over a finite alphabet this equivalence fails, and sorting is strictly a special case of finding distance. In section 5, we show that the number of reversals required to sort a ternary string can be found in polynomial time. We also give some elementary bounds on reversal sorting over an arbitrary finite alphabet; these restrict any instance of sorting to a finite range of values. We conjecture that, for fixed k, these problems can be solved for k-ary alphabets in polynomial time.

Notation. Our alphabet will generally be the set $[k] = \{1, 2, \ldots, k\}$. We consider strings over this alphabet, elements $\alpha \in [k]^*$. We write $|\alpha|$ for the length of a string. We write $\mathcal{L}(a_1, \ldots, a_k)$ for the set of strings of length n with exactly a_i occurrences of i for each i. (Note that the set of permutations can be thought of as $\mathcal{L}(1, 1, \ldots, 1)$.)

1. Reversal diameter for finite alphabets. Our approach to finding the reversal diameter of $\mathcal{L}(a_1, \ldots, a_k)$ is straightforward: we present an algorithm that turns one element of $\mathcal{L}(a_1, \ldots, a_k)$ into any other in at most the desired number of reversals, and we also present a pair of elements of $\mathcal{L}(a_1, \ldots, a_k)$ that are provably at least the desired number of reversals apart. In order to prove the lower bound, we introduce an invariant of strings, *tilt*, which is linear in a certain sense and which cannot be changed very much by a single reversal. Bafna and Pevzner [2] looked at properties of a difference graph to prove lower bounds on reversal distance between permutations, and Christie and Irving [9] gave algorithmic upper bounds on the reversal distance between pairs of binary strings. However, both the graph we use to compute tilt and the algorithm we give for our upper bound are different from earlier work.

Given a graph G with vertex set $V \subset \mathbb{Z}^+$ and an edge-weight function $w : \binom{V}{2} \to \mathbb{Z}$, define the *tilt* of G to be

$$t(G) = \sum_{i \text{ odd}, \, j \text{ even}} w(\{i, j\}) \varepsilon_{ij}, \quad \text{where } \varepsilon_{ij} = \begin{cases} 1, & i < j, \\ -1, & i > j. \end{cases}$$

Tilt is linear in the following sense: when G and H are weighted graphs on the same vertex set G, let $G + H$ denote the weighted graph on V whose edge weight function is the sum of those of G and H. Then
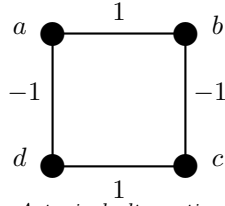
$$t(G + H) = t(G) + t(H).$$

FIG. 1. *A typical alternating square.*

An *alternating square* C on vertices $abcd$ is the weighted graph obtained from the closed walk $abcda$ by giving the edges ab and cd weight 1 and the edges bc and da weight -1 (when the edges are not distinct, the weights are summed; however, we exclude loops).

LEMMA 1. *If C is an alternating square, then $|t(C)| \leq 2$.*

Proof. Label C as shown in Figure 1. We argue by contradiction, first assuming that $t(C) \leq -3$; the other case can be argued symmetrically.

When $t(C) \leq -3$, at least 3 edges must contribute -1 to $t(C)$. Without loss of generality let them be ab, bc, and cd. When a is even,

- $b > a$ and b is odd, hence
- $c > b$ and c is even, hence
- $d > c$ and d is odd.

Thus the edge da contributes $+1$ and $t(C) = -2$. A similar argument applies when a is odd. □

Why is this relevant? Given a string $\alpha = \alpha_1 \cdots \alpha_n \in [k]^n$, define the associated weighted graph $G(\alpha)$ to have vertex set $[k]$ and edge weights

$$w(\{i, j\}) = |\{l : \{\alpha_l, \alpha_{l+1}\} = \{i, j\}\}| .$$

That is, $w(e)$ counts the number of times the edge e is used, in either direction, by the walk α. We ignore loops, however. When the reversal R_{ij} is applied to α, the transitions $\alpha_{i-1}\alpha_i$ and $\alpha_j\alpha_{j+1}$ are replaced by $\alpha_{i-1}\alpha_j$ and $\alpha_i\alpha_{j+1}$, while all other transitions remain unchanged. Thus

$$G(R_{ij}(\alpha)) = G(\alpha) + C,$$

where C is an alternating square on vertices $\alpha_i\alpha_{j+1}\alpha_j\alpha_{i-1}$. It follows that if β is obtained from α by a sequence of d reversals, then

$$G(\beta) = G(\alpha) + C_1 + C_2 + \cdots + C_d,$$

where $C_1, \ldots, C_d$ are alternating squares. Linearity of tilt and Lemma 1 now yield the following.

LEMMA 2. *When $\alpha, \beta \in \mathcal{L}(a_1, \ldots, a_k)$,*

$$d_{\mathrm{rev}}(\alpha, \beta) \geq \frac{1}{2} \left| t\left(G(\alpha) - G(\beta)\right) \right| .$$

We use this to determine the diameter of $\mathcal{L}(a_1, \ldots, a_k)$.

THEOREM 3. *The reversal diameter of $\mathcal{L}(a_1, \ldots, a_k) \subseteq [k]^n$ is $n - \max_i a_i$.*

Proof. Without loss of generality, we assume $a_1 \geq a_2 \geq \cdots \geq a_k$. For the upper bound, we give a procedure for successively modifying two strings $\alpha = \alpha_1 \cdots \alpha_n$ and $\beta = \beta_1 \cdots \beta_k$ in $\mathcal{L}(a_1, \ldots, a_k)$ until both are the same. Because reversals are
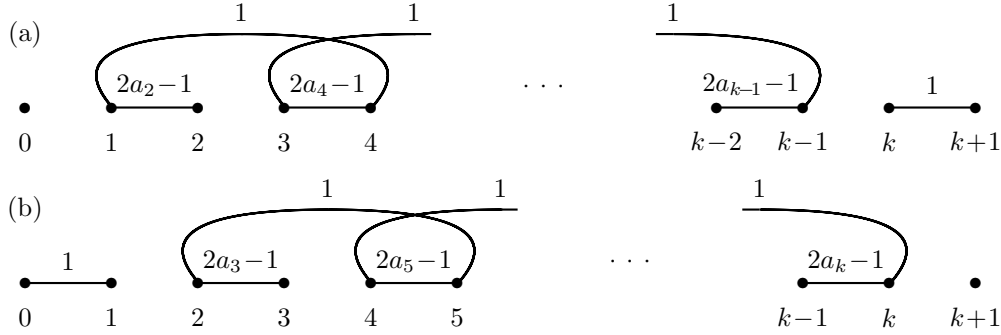
FIG. 2. *Multiplicities of edges joining vertices of opposite parity in* (a) $G(\alpha')$ *and* (b) $G(\beta')$ *(when k is odd).*

involutions on $[k]^n$, we can produce a sequence of the same total length carrying α to β.

Let $\alpha^{(0)} = \alpha$ and $\beta^{(0)} = \beta$. Given $\alpha^{(i)}$ and $\beta^{(i)}$, let j be the smallest index such that $\alpha_j^{(i)} \neq \beta_j^{(i)}$.

- If $\beta_j^{(i)} \neq 1$, pick a $j' > j$ such that $\alpha_{j'}^{(i)} = \beta_j^{(i)}$. Let $\alpha^{(i+1)} = R_{j,j'}(\alpha^{(i)})$ and let $\beta^{(i+1)} = \beta^{(i)}$.
- If $\beta_j^{(i)} = 1$ (and thus $\alpha_j^{(i)} \neq 1$), pick a $j' > j$ such that $\beta_{j'}^{(i)} = \alpha_j^{(i)}$. Let $\beta^{(i+1)} = R_{j,j'}(\beta^{(i)})$ and let $\alpha^{(i+1)} = \alpha^{(i)}$.

For each i, let $\gamma^{(i)}$ be the initial segment on which $\alpha^{(i)}$ and $\beta^{(i)}$ agree. Note that $|\gamma^{(i)}|$ is strictly increasing in i, and that furthermore $|\{j : \gamma_j^{(i)} \neq 1\}|$ is strictly increasing in i. This process must therefore stop after at most

$$|\{j : \alpha_j^{(0)} \neq 1\}| = n - a_1$$

steps.

For the lower bound, we give two strings $\alpha, \beta \in \mathcal{L}(a_1, \ldots, a_k)$ at distance at least $n - a_1$. We actually apply Lemma 2 to $\alpha' = 0\alpha(k+1)$ and $\beta' = 0\beta(k+1)$; since any sequence of reversals taking α to β also takes α' to β', $d_{\mathrm{rev}}(\alpha, \beta) \geq d_{\mathrm{rev}}(\alpha', \beta')$.

When k is odd, take

$$\alpha' = 0(21)^{a_2} 1^{a_1 - a_2} \cdots (k-1\ k-2)^{a_{k-1}} (k-2)^{a_{k-2} - a_{k-1}} k^{a_k} (k+1)$$

and

$$\beta' = 01^{a_1} (32)^{a_3} 2^{a_2 - a_3} \cdots (k\ k-1)^{a_k} (k-1)^{a_{k-1} - a_k} (k+1).$$

The edges joining vertices of opposite parity determine tilt; these are shown with their multiplicities in Figure 2 (recall that both ij and ji substrings contribute to $w(\{i, j\})$). As all relevant edges in $G(\alpha')$ increase from odd to even and all relevant

edges in $G(\beta')$ increase from even to odd,

$$
\begin{aligned}
t\left(G(\alpha') - G(\beta')\right) &= (2a_2 - 1) + 1 + (2a_4 - 1) + 1 + \cdots + (2a_{k-1} - 1) + 1 \\
&\quad + 1 + (2a_3 - 1) + 1 + (2a_5 - 1) + \cdots + 1 + (2a_k - 1) \\
&= 2\sum_{i=2}^{k} a_i = 2(n - a_1).
\end{aligned}
$$

Lemma 2 now gives the desired lower bound on $d_{\mathrm{rev}}(\alpha, \beta)$.

When k is even, the strings

$$
0(2\,1)^{a_2}1^{a_1 - a_2}\cdots(k\ \ k-1)^{a_k}(k-1)^{a_{k-1}-a_k}(k+1),
$$

$$
01^{a_1}(3\,2)^{a_3}2^{a_2 - a_3}\cdots(k-1\ \ k-2)^{a_{k-1}}(k-2)^{a_{k-2}-a_{k-1}}k^{a_k}(k+1)
$$

yield the same bound. ☐

Remark. As we noted earlier, permutations are simply $\mathcal{L}(1, 1, \ldots, 1)$. In this case, our argument gives a new proof that the reversal diameter of S_n is $n - 1$. If the symbols appearing in α and β are relabeled so that β is taken to $123\cdots n$, then α is taken to $315274\cdots$ (with ending depending on the parity of n). Bafna and Pevzner [2] showed that this permutation and its inverse are the only permutations at distance $n - 1$ from the identity.

We do not have an analogue of Theorem 3 for transpositions.

PROBLEM 1. *What is the transposition diameter of $\mathcal{L}(a_1, \ldots, a_k)$?*

There are similar problems for signed reversals, prefix reversals, etc.

2. Distance between random strings. The diameter of S_n is $\Theta(n)$ for each of the families of transformations we are considering: reversals [24], transpositions [3, 10], pancake flips [14]. The distance between randomly chosen permutations can quickly be seen to be $\Theta(n)$ with high probability, since each family has a bounded number of cuts per transformation and two random permutations have only about Poisson(2) adjacencies in common.

For strings taken from a finite alphabet with a positive fraction of the string devoted to each letter, we have shown above that the diameter under reversals is linear. Cutpoint arguments clearly imply the same for the other families of operations. In this section we show that the distance between random strings σ_1, σ_2 in the same component of $[k]^n$ is typically much smaller: only $\Theta(n/\log n)$. First, both σ_1 and σ_2 are partitioned into substrings of length approximately $c \log n$. With high probability, most of the resulting pieces appear about the same number of times in σ_1 and σ_2. For each family of operations considered, these substrings can be arranged and any remaining letters aligned in $O(n/\log n)$ operations. Furthermore, with high probability σ_1 and σ_2 have no common substrings of length $C \log n$, where $C > c$, and thus at least $n/(C \log n)$ cuts must be made.

Most of this section examines the anatomy of pairs of random strings. Our conclusions about distances between typical pairs can be drawn for any collection of operations with boundedly many cutpoints per operation.

Let $\mathbf{p} = (p_1, \ldots, p_k)$ be a rational probability vector satisfying $p_1 \geq p_2 \geq \cdots \geq p_k$ and let $h_i = -\log p_i$. For $\alpha = \alpha_1 \cdots \alpha_m \in [k]^*$, let $h(\alpha) = \sum_{i=1}^{m} h_{\alpha_i}$ be the entropy of α. We also set $H = H(\mathbf{p}) = \sum_{i \in [k]} p_i h_i$, the entropy of $\mathbf{p}$.

Given a $c > 0$, define the *c-threshold set*, $\mathcal{A}_c$, to consist of all words $\alpha = \alpha_1 \ldots \alpha_m \in [k]^*$ such that $h(\alpha) \geq c \log n$, but $h(\alpha_1 \ldots \alpha_{m'}) < c \log n$ for $m' < m$. (Much of the notation in this section conceals dependence on n.)

The following are immediate from definitions.

(1) $$\text{For } \alpha \in \mathcal{A}_c, \quad c \log n \leq h(\alpha) < c \log n + h_k.$$

(2) $$\text{For } \alpha \in \mathcal{A}_c, \quad \frac{c \log n}{h_k} \leq |\alpha| \leq \left\lceil \frac{c \log n}{h_1} \right\rceil.$$

(3) $$n^c \leq |\mathcal{A}_c| < \frac{n^c}{p_k}.$$

We also note that

(4) $$\sum_{\alpha \in \mathcal{A}_c} e^{-h(\alpha)} = 1.$$

This simply states that for the random process $\alpha_1 \alpha_2 \ldots$, in which the α_i are chosen independently according to $\mathbf{p}$, the stopping time

$$T = \min\{m \,:\, h(\alpha_1 \alpha_2 \ldots \alpha_m) \geq c \log n\}$$

has $\Pr(T < \infty) = 1$.

The next lemma will be useful as we decompose strings into short pieces.

LEMMA 4. *Let* $L_c = \sum_{\alpha \in \mathcal{A}_c} |\alpha| e^{-h(\alpha)}$ *be the expected value of the length of a random* $\alpha \in \mathcal{A}_c$ *determined by successive i.i.d. choices of letters according to* $\mathbf{p}$. *Then*

$$L_c = \frac{c \log n}{H} \left(1 + o\left(1\right)\right).$$

Proof. Consider characters $\alpha_1, \alpha_2, \ldots$ chosen independently from $[k]$ according to $\mathbf{p}$. For each i, $E[h_{\alpha_i}] = H$; let $\sigma^2 = \text{Var}[h_{\alpha_i}]$.

Let $\alpha^- = \alpha_1 \cdots \alpha_{m^-}$ and $\alpha^+ = \alpha_1 \cdots \alpha_{m^+}$ be the initial strings of lengths $m^- = \left\lceil \frac{c \log n}{H} - (\log n)^{2/3} \right\rceil$ and $m^+ = \left\lceil \frac{c \log n}{H} + (\log n)^{2/3} \right\rceil$, respectively, and let $\alpha = \alpha_1 \ldots \alpha_{|\alpha|} \in \mathcal{A}_c$. The definition of $\mathcal{A}_c$ and Chebyshev's inequality now give

(5)
$$\begin{aligned}
\Pr\left[|\alpha| \leq m^-\right] &= \Pr\left[h(\alpha^-) \geq c \log n\right] \\
&= \Pr\left[h(X^-) - Hm^- > H(\log n)^{2/3} + O(1)\right] \\
&\leq \frac{\sigma^2 m^-}{(H(\log n)^{2/3} + O(1))^2} = O((\log n)^{-1/3}).
\end{aligned}$$

Similarly,

(6)
$$\begin{aligned}
\Pr\left[|\alpha| > m^+\right] &= \Pr\left[h(X^+) < c \log n\right] \\
&= \Pr\left[h(X^+) - Hm^+ < -H(\log n)^{2/3} + O(1)\right] \\
&\leq \frac{\sigma^2 m^+}{(H(\log n)^{2/3} + O(1))^2} = O((\log n)^{-1/3}).
\end{aligned}$$

Since (2), (5), and (6) show that $|\alpha|$ is within $O((\log n)^{2/3})$ of $c \log n / H$ with probability at least $1 - O((\log n)^{-1/3})$ and is always within a bounded factor of $c \log n$, we have

$$L_n = \left(1 - O((\log n)^{-1/3})\right) \left(\frac{c \log n}{H} + O((\log n)^{2/3})\right) + O\left(\frac{\log n}{(\log n)^{1/3}}\right)$$

$$= \frac{c \log n}{H} (1 + o(1)) . \quad \square$$

In what follows, we always let $n \to \infty$ in such a way that $p_1 n, \ldots, p_k n$ are all integral.

THEOREM 5. *Fix $\epsilon > 0$. When σ_1 and σ_2 are chosen independently and uniformly from $\mathcal{L}(p_1 n, \ldots, p_k n)$, then, with probability approaching 1 as $n \to \infty$, σ_1 and σ_2 can be broken into identical collections of at most $(1 + \epsilon)(\frac{H n}{\log n})$ substrings.*

Proof. We first consider two strings, ρ_1 and ρ_2, each consisting of n letters chosen independently from $[k]$ according to $\mathbf{p}$. We show that ρ_1 and ρ_2 can be broken into words of approximately equal probability in such a way that both strings have approximately equal numbers of each type of word. We then modify ρ_1 and ρ_2 slightly to obtain σ_1 and σ_2, each uniformly distributed in $\mathcal{L}(a_1, \ldots, a_k)$; as these modifications do not affect very many pieces, all discrepancies can be broken into singletons.

Fix a $\delta \in (0, 1)$ such that $\frac{1}{1-\delta} < 1 + \epsilon$, and call $\alpha \in [k]^*$ *substantial* when $\alpha \in \mathcal{A}_{1-\delta}$. Each ρ_j, $j = 1, 2$, can be broken uniquely into disjoint substantial words, starting from the left and proceeding down the string. We call those words $\beta_1^j, \beta_2^j, \ldots$. (We can regard each ρ_j as the initial segment of length n from an infinite string chosen according to $\mathbf{p}$, and we extract the β_i^j from this infinite string. Thus β_i^j is defined for all i.) Let

$$N = \frac{n}{L_{1-\delta}} - \sqrt{n}.$$

We claim that, with high probability, each ρ_j contains at least N substantial words, while the first N substantial words of each ρ_j cover at least $N - \sqrt{n}(\log n)^2$ characters. Chebyshev's inequality, together with (2), gives

$$\Pr\left[|\beta_1^j| + \cdots + |\beta_N^j| > n\right] \leq \Pr\left[|\beta_1^j| + \cdots + |\beta_N^j| - N L_{1-\delta} > L_{1-\delta}\sqrt{n}\right]$$

$$\leq \frac{N(O((\log n)^2))}{n L_{1-\delta}^2} = O((\log n)^{-1}) = o(1)$$

and

$$\Pr\left[|\beta_1^j| + \cdots + |\beta_N^j| < n - \sqrt{n}(\log n)^2\right]$$

$$\leq \Pr\left[|\beta_1^j| + \cdots + |\beta_N^j| - N L_{1-\delta} < -\sqrt{n}(\log n)^2(1 + o(1))\right]$$

$$\leq \frac{O(N(\log n)^2)}{n(\log n)^4} = O((\log n)^{-3}) = o(1).$$

For each $\alpha \in \mathcal{A}_{1-\delta}$, let $N_{\alpha,j}$ be the number of pieces of type α among the first N substantial words of ρ_j. We use the following Chernoff-type inequality (see Janson, Łuczak, and Rucinski [20, p. 26]: if $X \sim \text{binomial}(n, p)$, then for $t > 0$,

$$\Pr[|X - EX| > t] \leq 2 \exp\left(-\frac{t^2}{np + \frac{t}{3}}\right).$$

Since $N_{\alpha,j}$ is binomial$(N, e^{-h(\alpha)})$,

$$\Pr[|N_{\alpha,j} - E(N_{\alpha,j})| \geq n^{\delta/2} \log n]$$

$$\leq 2 \exp\left(-\frac{n^\delta (\log n)^2}{2\left(\frac{Hn(1+o(1))}{(1-\delta)\log n}\left(\frac{1}{n^{1-\delta}}\right) + \frac{n^{\delta/2}\log n}{3}\right)}\right)$$

$$= 2 \exp\left(-\Omega\left((\log n)^3\right)\right) = o\left(\frac{1}{n^{1-\delta}}\right).$$

Now sum over $\alpha \in \mathcal{A}_{1-\delta}$ and $j = 1,2$. Equation (3) implies

$$\Pr\left[\max_{\alpha,i} |N_{\alpha,i} - E(N_{\alpha,i})| < n^{\delta/2}\log n\right] \to 1 \text{ as } n \to \infty.$$

Thus, we can with high probability match up all except

$$n^{\delta/2}\log n |\mathcal{A}_{1-\delta}| = O\left(n^{1-\delta/2}\log n\right)$$

of the first N substantial words in ρ_1 with (distinct) counterparts among the first N substantial words of ρ_2.

We now modify ρ_1 and ρ_2 to obtain uniformly distributed elements σ_1 and σ_2 of $\mathcal{L}(a_1, \ldots, a_k)$. For $i \in [k]$, let $N_{i,j}$ denote the number of i's in ρ_j. Since $N_{i,j}$ is a binomial(n, p_i) random variable, Chebyshev gives

$$\Pr\left[|N_{i,j} - p_i n| > \sqrt{n}\log n\right] \leq \frac{p_i(1-p_i)n}{(\log n)^2 n} = O((\log n)^{-2}),$$

so

$$\Pr\left[\max_{i,j}|N_{i,j} - p_i n| < \sqrt{n}\log n\right] \to 1 \text{ as } n \to \infty.$$

To generate σ_1 and σ_2, we must reallocate some sites containing overrepresented characters to currently underrepresented ones. Take as many sites as necessary uniformly from each overrepresented letter, and fill the entire collection of sites thus selected with an assignment of the appropriate multiset of characters uniformly chosen from the possible assignments. With high probability, we need only change $O(\sqrt{n}\log n)$ characters—and thus will break at most that many of the substantial-word matches we built between ρ_1 and ρ_2.

Now break all unmatched substantial words (including those past position N that were never considered, those among the first N that we tried to match but failed, and those whose matches were broken by character modifications) in both σ_1 and σ_2 into single characters. Let N^* be the resulting number of fragments in each string. With high probability as $n \to \infty$,

$$N^* = N + O\left(\sqrt{n}(\log n)^2 + n^{1-\delta/2}(\log n)^2 + \sqrt{n}(\log n)^2\right)$$

$$= \frac{Hn}{(1-\delta)\log n}(1 + o(1)) \leq (1+\epsilon)Hn$$

for n sufficiently large. □

THEOREM 6. *Fix $\delta > 0$. Choose σ_1 and σ_2 independently and uniformly from $\mathcal{L}(p_1 n, \ldots, p_k n)$. For $i = 1, 2$, let $\mathcal{S}_i$ be the multiset of all substrings of σ_i that belong to $\mathcal{A}_{1+\delta}$. Then*

$$\Pr[|\mathcal{S}_1 \cap \mathcal{S}_2| \geq n^{1-\delta}(\log n)] \to 0 \ \ as \ n \to \infty.$$

Proof. For $\alpha \in [k]^*$, let q_α be the probability that α occurs as an initial substring of a string σ chosen uniformly from $\mathcal{L}(p_1 n, \ldots, p_k n)$. We can generate such a σ by sampling without replacement from a bag containing $p_i n$ copies of i. From this it is easy to see that, for c fixed and $\alpha \in \mathcal{A}_c$,

(7)
$$q_\alpha \leq \prod_{i=1}^{|\alpha|} \left(p_{\alpha_i} \left(\frac{n}{n - |\alpha|} \right) \right) = e^{-h(\alpha)}(1 + o(1)) \leq n^{-c}(1 + o(1)).$$

(The $o(1)$ estimate follows from (3).) Thus the probability that α appears as a substring of σ starting at any given position is also at most $n^{-c}(1 + o(1))$.

Since there are only $n - O(\log n)$ possible starting positions in σ_1 for a substring of weight at least $1 + \delta$, we trivially have $|\mathcal{S}_1| \leq n$. Similarly, there are $n - O(\log n)$ possible starting positions in σ_2 for a substring in $\mathcal{A}_{1+\delta}$. Let N be the number of locations i for which the corresponding substring is an element of $\mathcal{S}_1$; clearly $N \geq |\mathcal{S}_1 \cap \mathcal{S}_2|$. By (7), for any given $\mathcal{S}_1$,

$$E[N|\mathcal{S}_1] \leq n \sum_{\alpha \in \mathcal{S}_1} q_\alpha \leq n^{1-\delta}(1 + o(1)),$$

so

$$E[N] \leq n^{1-\delta}(1 + o(1)),$$

and Markov's inequality gives

(8)
$$\Pr[N > n^{1-\delta} \log n] = o(1). \qquad \square$$

The following theorem combines the previous results to give bounds on the distance between random strings.

THEOREM 7. *Fix $\epsilon > 0$, and choose σ_1 and σ_2 uniformly and independently from $\mathcal{L}(p_1 n, \ldots, p_k n)$. Then each of the following statements holds with probability approaching 1 as $n \to \infty$:*

(i) $\frac{1-\epsilon}{2}\left(\frac{Hn}{\log n}\right) \leq d_{\mathrm{rev}}(\sigma_1, \sigma_2) \leq (1 + \epsilon)\left(\frac{Hn}{\log n}\right)$.

(ii) $\frac{1-\epsilon}{3}\left(\frac{Hn}{\log n}\right) \leq d_{\mathrm{tr}}(\sigma_1, \sigma_2) \leq \frac{2(1+\epsilon)}{3}\left(\frac{Hn}{\log n}\right)$.

(iii) $(1 - \epsilon)\left(\frac{Hn}{\log n}\right) \leq d_{\mathrm{pf}}(\sigma_1, \sigma_2) \leq 2(1 + \epsilon)\left(\frac{Hn}{\log n}\right)$.

Proof. For the upper bounds, we need only apply results on the diameter of S_n under the various operations to the decomposition of σ_1 and σ_2 into at most $N^* = (1 + \epsilon)\frac{Hn}{\log n}$ identical pieces that Theorem 5 provides with high probability. Meidanis, Walter, and Dias [24] show that the signed reversal diameter of S_{N^*} is at most $N^* + 1$, while Eriksson et al. [10] bounded the transposition diameter of S_{N^*} by $\lfloor (2N^* - 2)/3 \rfloor$ for $N^* > 9$, and Gates and Papadimitriou [14] showed that the signed pancake-flipping diameter of S_{N^*} is at most $2N^* + 3$.

The lower bounds are nearly as simple. Fix a $\delta > 0$ such that $1 - \epsilon < \frac{1}{1+\delta}$. We call a word $\alpha \in \mathcal{A}_{1+\delta}$ *unusual*, while a word $\alpha \in \mathcal{A}_3$ is termed *implausible*. Equations (3)

and (7) guarantee that

$$\Pr[\sigma_1 \text{ and } \sigma_2 \text{ share an implausible substring}] \leq n^2 \left(\frac{n^3}{p_k}\right) \left(\frac{(1+o(1))}{n^3}\right)^2$$
$$= O(n^{-1}).$$

Equation (2) implies that, with probability $1 - O(1/n)$, all common substrings of σ_1 and σ_2 have length at most $\lceil \frac{3\log n}{h_1} \rceil$. If we (temporarily) define the *weight* of a word $\alpha \in [k]^*$ to be $h(\alpha)/\log n$, then we can compute as follows. By Theorem 6, there are, with high probability, fewer than $n^{1-\delta}\log n$ sites in each string to start common unusual substrings. Equation (8) now implies that with high probability at most $n^{1-\delta}(\log n)\lceil \frac{3\log n}{h_1} \rceil = o(n)$ characters are contained in common substrings of weight greater than $1 + \delta$ (and their combined weight is also $o(n)$). Any sequence of operations taking σ_1 to σ_2 must cut the remaining characters into words of weight less than $1+\delta$. The entire string, without the unusual common substrings, has weight $(H - o(1))n/\log n$, and so there must be at least $(1 + o(1))\frac{Hn}{(1+\delta)\log n}$ cuts. A single reversal makes at most two cuts, a single transposition makes at most three cuts, and a single pancake flip makes at most one cut, giving the results above.   ☐

   We conjecture that, in each case, the expected value of $d(\sigma_1, \sigma_2)/(n/\log n)$ tends to a constant; we also leave open the further problem of determining this constant.

   3. Complexity. The complexity of sorting permutations by reversals or transpositions has been extensively studied. Kececioglu and Sankoff [23] conjectured that sorting reversals by permutations (MIN-SBR) is NP-hard, and this was proved by Caprara [7]. Berman and Karpinski [6] showed that sorting by reversals is MAX-SNP hard; Karpinski [22] showed that for any $\epsilon > 0$ it is NP-hard to approximate reversal distance within a factor $1237/1236 - \epsilon$. A number of authors have given approximation algorithms: Kececioglu and Sankoff [23] gave a 2-approximation algorithm, Bafna and Pevzner [2] gave a 1.75-approximation algorithm, Christie [8] gave a 1.5-approximation algorithm, and recently, Berman, Hannenhalli, and Karpinski [5] gave a 1.375-approximation algorithm.

   The problem of sorting *signed* permutations by reversals turns out to be polynomial time, as was shown by Hannenhalli and Pevzner [17]. Faster algorithms were found by Berman and Hannenhalli [4] and by Kaplan, Shamir, and Tarjan [21].

   The complexity of sorting by transpositions remains unknown, although Bafna and Pevzner [3] have given a 1.5-approximation algorithm.

   Christie and Irving [9] considered the complexity of reversal distance and transposition distance for strings over finite alphabets. They showed that reversal distance is NP-hard for binary strings, although a binary string can be sorted in polynomial time. They also showed that binary strings can be sorted by transpositions in polynomial time, but left open the complexity of transposition distance.

   We begin this section by giving another proof of Christie and Irving's [9] result that reversal distance is NP-hard for strings over binary alphabets; this of course implies that determining reversal distance is NP-hard for any finite alphabet. Using a similar argument we also show that, surprisingly, signed reversal distance is also NP-hard for signed strings over finite alphabets. We then prove that sorting by transpositions is NP-hard for binary strings.

   THEOREM 8. *Reversal distance is NP-hard for binary strings.*

   Proof. We give a reduction from sorting permutations by reversals. Given a

permutation $\pi = \pi(1)\cdots\pi(n)$, we define the string $\lambda(\pi)$ by

$$\lambda(\pi) = (10^{\pi(1)}1)^{2n}\cdots(10^{\pi(n)}1)^{2n}.$$

We call the substrings $(10^{\pi(i)}1)^{2n}$ the *blocks* of $\lambda(\pi)$. Each block consists of $2n$ *sub-blocks*, each of the form $10^{\pi(i)}1$. Clearly, $\lambda(\pi)$ can be constructed from π in polynomial time.

Given permutations π_1 and π_2, it is easy to see that

$$d_{\mathrm{rev}}\left(\lambda(\pi_1),\lambda(\pi_2)\right) \le d_{\mathrm{rev}}\left(\pi_1,\pi_2\right),$$

since a sequence of reversals mapping π_1 to π_2 maps to a sequence of reversals on the corresponding sequence of blocks $(10^{\pi_1(j)}1)^{2n}$ in $\lambda(\pi_1)$ (note that each block is invariant under reversals).

Now let $t = d_{\mathrm{rev}}\left(\lambda(\pi_1),\lambda(\pi_2)\right)$. If $t < d_{\mathrm{rev}}\left(\pi_1,\pi_2\right)$, then consider a sequence of t reversals taking $\lambda(\pi_1)$ to $\lambda(\pi_2)$. Since the reversal diameter of S_n is less than n, we have $t < n$. Now consider a block $(10^{\pi_1(i)}1)^{2n}$. This contains $2n$ subblocks: since the t reversals cut the string in at most $2t < 2n$ places, there must be one subblock I_i that does not get cut. It follows that I_i must get mapped to a segment of the block $(10^{\pi_1(i)}1)^{2n} = (10^{\pi_2(i')}1)^{2n}$, where $i' = \pi_2^{-1}\pi_1(i)$.

Thus the segments $I_1,\dots,I_n$, which occur in order in $\lambda(\pi_1)$, are rearranged by the sequence of t reversals to occur in $\lambda(\pi_2)$ in the order $I_{\pi_1^{-1}\pi_2(1)},\dots,I_{\pi_1^{-1}\pi_2(n)}$. Considering the action of the reversals just on the segments $I_1,\dots,I_n$ implies that there exists a sequence of t reversals which rearranges id to $\pi_1^{-1}\pi_2$. Since $d_{\mathrm{rev}}\left(id,\pi_1^{-1}\pi_2\right) = d_{\mathrm{rev}}\left(\pi_1,\pi_2\right)$, this is a contradiction.

We therefore have $d_{\mathrm{rev}}\left(\lambda(\pi_1),\lambda(\pi_2)\right) = d_{\mathrm{rev}}\left(\pi_1,\pi_2\right)$, and so we have a reduction from reversal distance for permutations to reversal distance for binary strings. □

As noted above, signed permutations can be sorted in polynomial time [17, 4, 21]. By contrast, over finite alphabets, the problem of finding signed reversal distance is NP-hard.

THEOREM 9. *Signed reversal distance is NP-hard for binary strings.*

Proof. As in the previous theorem, we reduce from MIN-SBR. Given a permutation $\pi = \pi(1)\cdots\pi(n)$, we encode π by

$$\lambda(\pi) = (10^{\pi(1)}1\overline{1}\,\overline{0}^{\pi(1)}\overline{1})^{2n}\cdots(10^{\pi(n)}1\overline{1}\,\overline{0}^{\pi(n)}\overline{1})^{2n}.$$

Since each block is invariant under reversal,

$$d_{\overline{\mathrm{rev}}}\left(\lambda(\pi_1),\lambda(\pi_2)\right) \le d_{\mathrm{rev}}\left(\pi_1,\pi_2\right).$$

Arguing as before, we deduce that

$$d_{\overline{\mathrm{rev}}}\left(\lambda(\pi_1),\lambda(\pi_2)\right) = d_{\mathrm{rev}}\left(\pi_1,\pi_2\right).$$

Thus signed reversal distance is NP-hard. □

As we have seen, signed reversal distance is NP-hard for strings with repeated symbols. We show now that the difficulty remains even if we allow only two occurrences of each symbol.

THEOREM 10. *Signed reversal distance is NP-hard for strings in which there is at most one positive and one negative occurrence of each symbol.*

Proof. We prove this by reduction from MIN-SBR. We start by mapping each instance $\pi(1)\cdots\pi(n)$ to the string $S = \pi(1)\overline{\pi(1)}\cdots\pi(n)\overline{\pi(n)}$. The signed distance

from this to $T = 1\bar{1} \cdots n\bar{n}$ is clearly at most the reversal distance from $\pi(1) \cdots \pi(n)$ to $1 \cdots n$ (note that $i\bar{i}$ is reversal-invariant). On the other hand, any sequence of signed reversals taking S to T yields a corresponding sequence of reversals taking $\pi(1) \cdots \pi(n)$ to $1 \cdots n$ by restricting attention to the (unsigned) action of the signed reversals on the elements of S that initially have positive orientation and then ignoring signs. The two distances are therefore equal and so it is NP-hard to determine signed reversal distance.   □

We remark that if only $O(\log n / \log \log n)$ repeats in total are allowed, then signed reversal distance is solvable in polynomial time, since we can systematically examine all possible pairings between symbols of the same type; on the other hand, it is NP-hard to determine signed reversal distance with $\Omega(n^\epsilon)$ repeats, since we can encode instances of MIN-SBR of size n^ϵ using the methods of Theorem 10 and then pad out with additional variables occurring once each and in the same order at the end of each string.

THEOREM 11. *Transposition distance is NP-hard for binary strings.*

Proof. We prove the result first for strings over the alphabet $\{0, 1, 2, 3\}$ by reduction from the NP-hard problem 3-PARTITION [13], which we quote here.

INSTANCE: Positive integers n and N and positive integers $a_1, \ldots, a_{3n}$, with $N/4 < a_i < N/2$ for every i.

QUESTION: Is there a partition of the a_i into n (multi)sets of size 3, each summing to N?

The problem 3-PARTITION is strongly NP-hard [13]: that is, there is a polynomial $p(n)$ such that it is still NP-hard when all the a_i are at most $p(n)$. Our reduction is polynomially bounded for instances of this type.

Given an instance of 3-PARTITION with weights $a_1, \ldots, a_{3n}$ bounded by $p(n)$, consider the strings

$$S = 2^{n+1} 30^{a_1} 130^{a_2} 13 \cdots 30^{a_{3n}} 13$$

and

$$T = (20^N 1^3)^n 23^{3n+1}.$$

Note that if this instance of 3-PARTITION is solvable, then $d_{\mathrm{tr}}(S, T) \leq 3n$, since we can successively generate each segment $20^N 1^3 2$ of T by moving three intervals $0^{a_i} 1$ between an adjacent pair of 2's. On the other hand, suppose that $d_{\mathrm{tr}}(S, T) \leq 3n$. Note that, among the adjacencies in S, the sequence must destroy n adjacencies of form 22, $3n$ adjacencies of form 30, $3n$ adjacencies of form 13, and $2n$ adjacencies of form 01, a total of $9n$ adjacencies. It follows that there must be exactly $3n$ transpositions in the sequence, and furthermore that no transposition cuts an adjacency 00. The blocks 0^{a_i} of zeros in S are therefore preserved in T and so constitute a solution to 3-partition.

We have shown that $d_{\mathrm{tr}}(S, T) = 3n$ if and only if our instance of 3-PARTITION has a solution. Since the lengths of S and T are bounded by a polynomial in n, it follows that it is NP-hard to determine transposition distance over alphabets of size 4.

To show that transposition distance is NP-hard for binary strings, we use an encoding similar to that in Theorem 8. Given a string $\epsilon = \epsilon_1 \cdots \epsilon_n$ with $\epsilon \in \{0, 1, 2, 3\}$, we encode it as

$$\lambda(\epsilon) = (10^{\epsilon_1 + 1} 1)^{3n} \cdots (10^{\epsilon_n + 1} 1)^{3n}.$$

For strings ϵ and ϵ' of length n, since $d_{\mathrm{tr}}(\epsilon, \epsilon') \leq n - 1$ and each transposition cuts the string in three places, arguing as before, we find that

$$d_{\mathrm{tr}}(\lambda(\epsilon), \lambda(\epsilon')) = d_{\mathrm{tr}}(\epsilon, \epsilon').$$

Composing these two reductions, both of which are polynomially computable for instances whose components are of polynomially bounded magnitude, sends instances of 3-PARTITION to instances of transposition distance for binary strings. We conclude that transposition distance for binary strings is NP-hard.   □

Let us note that the reductions from MIN-SBR employed in Theorems 8 and 9 are distance-preserving. Since MIN-SBR is NP-hard to approximate to within any factor better than 1237/1236 [22], it follows that reversal distance and signed reversal distance for binary strings are also NP-hard to approximate to within better than 1237/1236. We conjecture that, for some $\epsilon > 0$, it is NP-hard to approximate transposition distance for binary strings to within a factor better than $1 + \epsilon$.

4. An approximation algorithm for dense instances. For many NP-hard approximation problems, it is also NP-hard to find an approximate solution that is correct to within a small multiplicative factor. However, it is sometimes easier to handle *dense* cases of these problems. For instance, although there is an algorithm that approximates Max Cut to within a factor 1.138 [15], it is NP-hard to approximate to within a factor better than 17/16 [18]. On the other hand, for dense instances of Max Cut (instances G with $\Omega(|G|^2)$ edges or minimum degree $\Omega(|G|)$), it is possible to find a polynomial-time approximation scheme [11, 12]. Similar results exist for a number of other NP-hard problems (see, for instance, [1, 22]).

Our aim in this section is to describe a polynomial-time approximation scheme for dense instances of reversal distance for strings over a finite alphabet. This requires us to define a notion of "density" for instances of reversal distance: for $c > 0$, we say that an instance (σ, τ) with $|\sigma| = |\tau| = n$ is *c-dense* if $d_{\mathrm{rev}}(\sigma, \tau) \geq cn$. We show below that, for any fixed k and any $\epsilon > 0$, there is a linear time algorithm that approximates reversal distance between k-ary strings of length n to within an additive error ϵn. It follows that, for fixed k, and any $c > 0$ and $\epsilon > 0$, there is a linear time algorithm that approximates reversal distance of c-dense instances to within a factor $1 + \epsilon$.

We first prove a simple lemma concerning the effect of deletions on reversal distance.

LEMMA 12. *Suppose that σ and τ are compatible strings and that σ' and τ' are compatible strings obtained by deleting m elements from each string. Then*

$$|d_{\mathrm{rev}}(\sigma', \tau') - d_{\mathrm{rev}}(\sigma, \tau)| \leq 2m.$$

Proof. To see that

$$d_{\mathrm{rev}}(\sigma, \tau) \leq d_{\mathrm{rev}}(\sigma', \tau') + 2m,$$

consider an optimal sequence of reversals taking σ' to τ'. These same reversals can be applied to σ and τ, always placing cuts that occur in sites where letters have been deleted to the left of those letters. The two-reversal sequence shown below suffices to move an individual letter to a new location without changing the rest of the string:

$$A|xB|C \rightarrow A|\overline{B}|xC \rightarrow ABxC.$$

Thus we can correct the reinserted letters with at most $2m$ additional reversals. A similar argument shows that

$$d_{\mathrm{rev}}(\sigma', \tau') \leq d_{\mathrm{rev}}(\sigma, \tau) + 2m.   □$$

The existence of a polynomial-time approximation scheme follows immediately from the following theorem.

THEOREM 13. *For each fixed k and every $\epsilon > 0$ there is a linear time algorithm that approximates reversal distance between k-ary strings of length n to within an additive error of ϵn and outputs a sequence of reversals achieving this bound.*

Proof. The main idea of the proof is to break up the problem instance into "good" subinstances of a finite number of types, each of which can be sorted optimally. The subinstances can then be recombined at small cost.

Given $\epsilon < 1/10$, let $K = \lceil 100/\epsilon \rceil$ and let $\alpha_1, \dots, \alpha_L$ be an enumeration of the $L = k^K$ k-ary strings of length K. Let $f : [L] \to \mathbb{R}^k$ count the number of each digit present in the strings α_i: thus $(f(i))_j$ is equal to the number of times j occurs in α_i. We now build a mapping $\tilde{f} : \mathbb{R}^L \to \mathbb{R}^k$ by setting $\tilde{f}(a_1, \dots, a_L) = \sum_{i=1}^{L} a_i f(i)$. Thus if we break a string X into segments of length K, obtaining a_i segments of type α_i for each i, then $\tilde{f}(a_1, \dots, a_L)$ counts the number of occurrences of each character in the original string X.

For $d \geq 1$, we write U_d for the unit simplex in $\mathbb{R}^d$ given by the convex hull of the origin and the d standard unit basis vectors. Let $D = \{x_1, \dots, x_M\} \subseteq U_k$ satisfy the following two conditions:

(i) D is an $\epsilon/4$-net in U_k. (That is, every point in U_k is distance less than $\epsilon/4$ from some $x_i \in D$.)

(ii) All the coordinates of each $x_i \in D$ are rational.

For $i = 1, \dots, M$, let $X_i = f^{-1}(x_i)$ be the affine subspace in $\mathbb{R}^L$ that maps to x_i and let E_i be an $\epsilon/4K$-net in $X_i \cap U_L$, where we choose E_i so that each point has all coordinates rational.

By deleting at most $(\epsilon/4)n$ elements from the string σ, we obtain a string σ' of length $n' < n$ such that, for some i, σ' has $(x_i)_j n'$ occurrences of the digit j for $1 \leq j \leq k$. Delete the same collection of characters from τ to obtain τ'. Then σ' and τ' are compatible strings; furthermore, we may assume that both σ' and τ' have length a multiple of K.

Now break each of σ' and τ' into segments of length K. Deleting at most $\epsilon n'/4K$ segments from each of σ' and τ', we may assume that we have strings σ'' and τ'', each broken into $n'' \geq (1 - \epsilon)n'/K$ segments of length K such that σ'' has $n'' a_j$ copies of α_j and τ'' has $n'' b_j$ copies of α_j for each j, where a and b both belong to E_i. By the definition of E_i, σ'' and τ'' are compatible. Furthermore, by Lemma 12, $|d_{\mathrm{rev}}(\sigma'', \tau'') - d_{\mathrm{rev}}(\sigma, \tau)| < \epsilon n/2$.

It is therefore sufficient to solve the following problem to within an additive constant ϵn.

- INPUT: Two elements a and b of E_i and two compatible strings σ and τ such that
  - $|\sigma| = |\tau| = n$, where $K|n$;
  - when broken into segments of length K, σ falls into $a_i n/K$ copies of S_i for each i;
  - when broken into words of length K, τ falls into $b_i n/K$ copies of S_i for each i.
- OUPUT: An optimal sequence of reversals taking σ to τ.

This breaks up into a constant number of separate problems, one for each choice of $a, b \in E_i$. We show that each of these problems has a good approximation algorithm.

Fix i and let $a, b \in E_i$. Let n_0 be an integer such that all entries of $n_0 a$ and $n_0 b$ are integers. Note that we can rearrange the n/K segments of σ and τ into any order

with at most $4n/K \leq \epsilon n/25$ reversals. We can therefore assume that the segments of σ and τ are in any order we choose, with cost at most $\epsilon n/25$.

For $j \geq 1$, let A_j be the collection of strings of length $n_0 jK$ constructed from $n_0 j a_l$ copies of α_l for each l, in any order; similarly, let B_j be the set of strings with $n_0 j b_l$ copies of α_l for each l, again in any order. Let

$$d_j = \min\{d_{\mathrm{rev}}(\alpha, \beta) : \alpha \in A_j, b \in B_j\}.$$

Clearly, for $j, j' \geq 1$,

$$(9) \qquad\qquad\qquad\qquad d_{j+j'} \leq d_j + d_{j'},$$

since strings from A_j and $A_{j'}$ can be concatenated to form strings in $A_{j+j'}$. Thus d_j is subadditive and therefore d_j/j tends to a limit r_∞. Let j_0 be large enough so that $|d_j/j - r_\infty| \leq \epsilon/4$ for $j \geq j_0$, and let $\alpha^{(0)} \in A_{j_0}$, $\beta^{(0)} \in B_{j_0}$ be strings with $d_{\mathrm{rev}}\left(\alpha^{(0)}, \beta^{(0)}\right) = d_{j_0} \leq j_0 r_\infty + \epsilon j_0/4$.

Now for $m \geq 1$, given strings $\alpha \in A_m$ and $\beta \in B_m$, we can rearrange α in blocks of size K to give α' with $\lfloor m/j_0 \rfloor$ copies of $\alpha^{(0)}$ and at most a constant number of additional segments; similarly, we can rearrange β to get β' with $\lfloor m/j_0 \rfloor$ copies of $\beta^{(0)}$ and at most a constant number of additional segments. Each of these rearrangements takes at most $n/K \leq \epsilon n/100$ reversals. Furthermore,

$$d_{\mathrm{rev}}(\alpha', \beta') \leq \left\lfloor \frac{m}{j_0} \right\rfloor d_{\mathrm{rev}}\left(\alpha^{(0)}, \beta^{(0)}\right) + O(1) = \frac{m d_{j_0}}{j_0} + O(1),$$

and we can write down an explicit sequence of reversals taking α to β in this time. Finally, note that

$$d_{\mathrm{rev}}(\alpha, \beta) \geq m r_\infty \geq \frac{m d_{j_0}}{j_0} - \frac{\epsilon m}{4} \geq \frac{m d_{j_0}}{j_0} - \frac{\epsilon n}{K},$$

so our approximation is correct to within ϵn.   □

The following corollary is an immediate consequence of the theorem.

COROLLARY 14. *For every fixed k and $c > 0$, there is a polynomial-time approximation scheme for c-dense instances of reversal distance for k-ary strings.*

Note that a similar argument gives a polynomial-time approximation scheme for dense instances of transposition distance over finite alphabets. The same approach also works for prefix reversals, except that (9) is replaced by

$$d_{j+j'} \leq d_j + d_{j'} + O(1),$$

as we work on the concatenation AB by first working on A, then reversing the entire string and working on B. This implies that d_j/j approaches some limit r_∞ (for instance, as an easy corollary of a result of Hammersley [16]), and the rest of the argument goes through with minor modification.

Given these results for strings over finite alphabets, it is natural to ask what happens for permutations. Recall that, for permutations, it suffices to consider MIN-SBR, the problem of sorting, for which each instance is a single permutation. We say that a permutation σ of length n is c-dense if there are at least cn integers i with $1 \leq i < n$ such that $|\sigma(i) - \sigma(i+1)| > 1$. It follows that c-dense strings necessarily require $\Omega(n)$ reversals to sort. We conjecture that dense permutations exhibit the same behavior as dense pairs of strings.

CONJECTURE 1. *For every $c > 0$, there is a polynomial-time approximation scheme for c-dense instances of sorting permutations by reversals.*

We also conjecture that similar statements hold for sorting permutations by prefix reversals and sorting permutations by transpositions.

5. Sorting strings over finite alphabets. Any algorithm for determining the number of reversals necessary to sort a permutation suffices to determine the reversal distance between an arbitrary pair of permutations: the distance between π_1 and π_2 is the same as that between $\pi_1\pi_2^{-1}$ and *id*. For strings from a finite alphabet, there is no such correspondence. There is, however, some hope that this special case of the distance problem might be easier than the full one. Indeed, Christie and Irving [9] show that the number of reversals required to sort a binary string σ is one less than the number of 0-blocks in the string 0σ, which can of course easily be determined in polynomial time. (An *i-block* is a maximal nonempty substring consisting entirely of copies of the character i.) We give below a similarly simple criterion for determining the number of reversals required to sort a ternary string, along with some elementary bounds over an arbitrary finite alphabet.

For the remainder of this section, we generally prepend a 0 and append a $k - 1$ to all strings in $\{0, 1, \ldots, k - 1\}^*$. These added characters are not allowed to move, but are included when we count blocks. We also generally replace each i-block with a single copy of the character i in example strings. The string resulting from applying both operations to σ is called the *standard form* of σ. Note that the number of reversals required to sort the standard form of σ is identical to the number required to sort σ.

We call a reversal *optimal* if it reduces the number of blocks by 2. Every optimal reversal is of the form $\ldots a|b \ldots a|b \ldots$. Conversely, whenever the string contains a *repeated transition*—that is, some substring containing two distinct characters occurs more than twice in the string—an optimal reversal is possible.

It will be convenient to categorize transitions between pairs of consecutive characters by the *unordered* pair of characters involved. There are then three types of transitions: 01/10, 02/20, and 12/21. For example, 01212101202 contains three 01/10 transitions, two 02/20 transitions, and five 12/21 transitions.

Define a 02-*block* to be a maximal substring consisting only of 0's and 2's and containing at least one of each. Call a 02-block *odd* if it contains an odd number of blocks of 0's and 2's; otherwise, call it *even*.

Example. The standard form of 01220000022221112111020 is 012021210202, which has two 02-blocks; the first is odd and the second is even.

LEMMA 15. *Let σ be a ternary string whose standard form has b blocks. If b is odd (even), then σ has an even (odd) number of* 02/20 *transitions, while the numbers of* 01/10 *and* 12/21 *transitions are both odd (even).*

Proof. Consider the standard form of σ to be a walk on the vertices $\{0, 1, 2\}$. Since the walk begins at 0 and ends at 2, the vertices 0 and 2 have odd degree, while vertex 1 has even degree. Thus the numbers of 01/10 and 12/21 transitions are of the same parity, which is opposite to that of both the number of 02/20 transitions and the total number of transitions.   ☐

We say that a string σ satisfies the *matching odd block condition* if σ has at least one 02-block, all 02-blocks of S are odd, and all 02-blocks of S have the same initial character.

THEOREM 16. *Let σ be a ternary string containing all 3 possible characters whose standard form has b blocks. Then the minimal number of reversals required to sort σ*

is $\left\lceil \frac{b-3}{2} \right\rceil$, unless σ satisfies the matching odd block condition, in which case sorting σ requires $\left\lceil \frac{b-3}{2} \right\rceil + 1$ reversals.

Proof. We note first that each reversal can reduce the number of blocks by at most 2. Thus $\left\lceil \frac{b-3}{2} \right\rceil$ is necessarily a lower bound for the number of reversals required to sort σ.

When b is even, Lemma 15 ensures that σ does not satisfy the matching odd block condition. It is not difficult to sort directly in this case. As long as there are repeated transitions, perform optimal reversals—which do not change the parity of the block number. Since there are only 6 possible transitions over the 3-letter alphabet, the resulting string σ' has at most 6 transitions. Let b' be the number of blocks in σ'. As $3 \le b' \le 7$ and b' is even, it is true that $b' = 4$ or $b' = 6$. The only possible structures for b' are shown below, along with an optimal reversal sorting of each:

$$0|10|2 \to 0012,$$

$$0|21|2 \to 0122,$$

$$0|10|212 \to 001|21|2 \to 001122,$$

$$0|1210|2 \to 001|21|2 \to 001122,$$

$$0|210|12 \to 001|21|2 \to 001122.$$

Hence we can sort σ in $\frac{b-b'}{2} + \left\lceil \frac{b'-3}{2} \right\rceil = \left\lceil \frac{b-3}{2} \right\rceil$ reversals, matching the elementary lower bound.

Now assume b is odd, but σ does not satisfy the matching odd block condition. We proceed in order through each of the following steps.

- Use optimal reversals internal to single 02-blocks to reduce any 02-blocks with 4 or more blocks to either 2 or 3 sub-blocks:

$$\cdots 0|20|2 \cdots \to \cdots 0022 \cdots .$$

  These reversals do not change the parities of the 02-blocks. After this stage, all remaining 02 blocks have one of the following forms: 02, 20, 202, or 020.

- Because σ did not originally satisfy the matching odd block condition and we have not changed the parities or types of any 02-blocks, either there are odd 02-blocks of both types or there are even 02-blocks. In the case that there are odd 02-blocks with different initial characters, we reduce them via an optimal reversal to two even 02-blocks:

$$\cdots 02|0 \cdots 2|02 \cdots \to \cdots 022 \cdots 002 \cdots .$$

  After this step, we may assume that there are even 02-blocks present.

- An even 02-block and an odd 02-block can be reduced with an optimal reversal to a single even 02-block:

$$\cdots 2|02 \cdots 2|0 \cdots \to \cdots 22 \cdots 200 \cdots$$

  or

$$\cdots 20|2 \cdots 0|2 \cdots \to \cdots 200 \cdots 22 \cdots .$$

  We repeat this step until all remaining odd 02-blocks are eliminated.

- As long as there are at least three even 02-blocks, some pair must match and an optimal reversal can eliminate both.
- We now have at most two even 02-blocks remaining. By Lemma 15, the number of even 02-blocks must be even. If the last two match, they can be eliminated in a single optimal reversal. If not, we are in one of the cases

$$0 \cdots 02 \cdots 20 \cdots 2 \quad \text{or} \quad 0 \cdots 20 \cdots 02 \cdots 2.$$

In the first case, the final 2-block must be preceded by a 1-block (since we have eliminated all other 02 transitions); in the second, the initial 0-block must be followed by a 1-block. Once we fill in those and the 1-blocks that must border the even 02-blocks, it becomes clear that for either case there is an optimal reversal that reverses one of the even 02-blocks, after which both 02-blocks can be eliminated:

$$0 \cdots 02 \cdots 1|201 \cdots 1|2 \qquad \text{or} \qquad 0|1 \cdots 120|1 \cdots 02 \cdots 2.$$

Once all 02/20-transitions have been eliminated, there are only 01/10- and 12/21-transitions. By Lemma 15, there is an odd number of each. Apply optimal reversals until the number of each is reduced to one. Since the string starts with 0 and ends with 2, the remaining transitions must be 01 and 12, and the string is sorted.

Finally, assume that σ satisfies the matching odd block condition. An easy case analysis shows that any string resulting from the application of an optimal reversal to σ also satisfies the matching odd block condition, and so cannot be completely sorted. Thus at least one nonoptimal reversal must be used when sorting σ, and the number of reversals required to do so is strictly greater than $\frac{b-3}{2}$.

In order to sort σ, first apply optimal reversals until a string σ' containing no repeated transitions is obtained and let b' be the number of blocks of σ'. We know that the string σ' still satisfies the matching odd block condition and thus is not sorted, so $4 \leq b' \leq 7$. We also know the number of 02/20 transitions is even, so by Lemma 15 the numbers of 01/10 and 12/21 transitions are both odd, and $b' = 7$ is impossible. Thus, $b' = 5$. The only possible types of 5-block strings are shown below, each with an optimal reversal sorting:

$$0|120|2 \to 00|21|2 \to 00122,$$

$$0|20|12 \to 00|21|2 \to 00122.$$

We have sorted σ in $\left\lceil \frac{b-5}{2} \right\rceil + 2 = \left\lceil \frac{b-3}{2} \right\rceil + 1$ reversals. $\quad \square$

Remark. In the b odd case, naive choices of optimal reversals can get one into trouble. For instance, the string 021021202 does not satisfy the matching odd block condition and so can be sorted in 3 reversals. Applying the optimal reversal 0[210]21202 results in 001221202, which does satisfy the matching odd block condition—and thus itself requires 3 reversals.

What about strings from larger alphabets? We can combine earlier observations to determine the number of reversals required to sort a string from a k-ary alphabet up to a finite error (whose magnitude depends on k).

THEOREM 17. *Let σ be a k-ary string whose standard form contains all k letters and has b blocks, and let t be the number of reversals required to sort σ. Then*

$$\left\lceil \frac{b-k}{2} \right\rceil \leq t \leq \left\lceil \frac{b-k}{2} \right\rceil + \frac{k(k-1)}{4} + 1.$$

Proof. The lower bound is clear; each reversal can reduce the number of blocks by at most 2. For the upper bound, we first use optimal reversals to reduce to a string σ' with r blocks and no repeated transitions; r must have the same parity as b. By Theorem 3, σ' can be sorted in at most $r - \lceil \frac{r}{k} \rceil$ steps. We've shown that

$$t \leq \frac{b-r}{2} + \left\lceil \frac{r-k}{2} \right\rceil + \left(r - \left\lceil \frac{r}{k} \right\rceil - \left\lceil \frac{r-k}{2} \right\rceil \right)$$
$$\leq \left\lceil \frac{b-k}{2} \right\rceil + \left(\frac{1}{2} - \frac{1}{k} \right) r + \frac{k}{2}.$$

Now substitute $r \leq \binom{k}{2} + 1$ to obtain the desired inequality.   □

CONJECTURE 2. *For each fixed k, the number of reversals required to sort k-ary strings can be determined in time polynomial in string length.*

It also seems plausible that there are polynomial-time algorithms for determining the number of operations required to sort k-ary strings using transpositions or pancake flips.

REFERENCES

[1] S. ARORA, D. KARGER, AND M. KARPINSKI, *Polynomial time approximation schemes for dense instances of NP-hard problems*, J. Comput. System Sci., 58 (1999), pp. 193–210.

[2] V. BAFNA AND P. A. PEVZNER, *Genome rearrangements and sorting by reversals*, SIAM J. Comput., 25 (1996), pp. 272–289.

[3] V. BAFNA AND P. A. PEVZNER, *Sorting by transpositions*, SIAM J. Discrete Math., 11 (1998), pp. 224–240.

[4] P. BERMAN AND S. HANNENHALLI, *Fast sorting by reversal*, in Combinatorial Pattern Matching (Laguna Beach, CA, 1996), Springer, Berlin, 1996, pp. 168–185.

[5] P. BERMAN, S. HANNENHALLI, AND M. KARPINSKI, *1.375-Approximation Algorithm for Sorting by Reversals*, Technical report, DIMACS TR:2001-41, Piscataway, NJ, 2001.

[6] P. BERMAN AND M. KARPINSKI, *On some tighter inapproximability results (extended abstract)*, in Automata, Languages and Programming (Prague, 1999), Springer, Berlin, 1999, pp. 200–209.

[7] A. CAPRARA, *Sorting by reversals is difficult*, in Proceedings of the First International Conference on Computational Molecular Biology, ACM Press, New York, 1997, pp. 75–83.

[8] D. A. CHRISTIE, *A 3/2-approximation algorithm for sorting by reversals*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 1998), ACM, New York, 1998, pp. 244–252.

[9] D. A. CHRISTIE AND R. W. IRVING, *Sorting strings by reversals and by transpositions*, SIAM J. Discrete Math., 14 (2001), pp. 193–206.

[10] H. ERIKSSON, K. ERIKSSON, J. KARLANDER, L. SVENSSON, AND J. WÄSTLUND, *Sorting a bridge hand*, Discrete Math., 241 (2001), pp. 289–300.

[11] W. FERNANDEZ DE LA VEGA, *Max-cut has a randomized approximation scheme in dense graphs*, Random Structures Algorithms, 8 (1996), pp. 187–198.

[12] W. FERNANDEZ DE LA VEGA AND M. KARPINSKI, *Polynomial time approximation of dense weighted instances of MAX-CUT*, Random Structures Algorithms, 16 (2000), pp. 314–332.

[13] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability*, W. H. Freeman, San Francisco, 1979.

[14] W. H. GATES AND C. H. PAPADIMITRIOU, *Bounds for sorting by prefix reversal*, Discrete Math., 27 (1979), pp. 47–57.

[15] M. X. GOEMANS AND D. P. WILLIAMSON, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM., 42 (1995), pp. 1115–1145.

[16] J. M. HAMMERSLEY, *Generalization of the fundamental theorem on sub-additive functions*, Proc. Cambridge Philos. Soc., 58 (1962), pp. 235–238.

[17] S. HANNENHALLI AND P. A. PEVZNER, *Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals*, J. ACM, 46 (1999), pp. 1–27.

[18] J. HASTAD, *Some optimal inapproximability results*, in STOC '97 (El Paso, TX), ACM, New York, 1999, pp. 1–10.

[19] M. H. HEYDARI AND I. H. SUDBOROUGH, *On the diameter of the pancake network*, J. Algorithms, 25 (1997), pp. 67–94.

[20] S. JANSON, T. ŁUCZAK, AND A. RUCINSKI, *Random Graphs*, Wiley-Interscience, New York, 2000.

[21] H. KAPLAN, R. SHAMIR, AND R. E. TARJAN, *A faster and simpler algorithm for sorting signed permutations by reversals*, SIAM J. Comput., 29 (2000), pp. 880–892.

[22] M. KARPINSKI, *Polynomial time approximation schemes for some dense instances of NP-hard optimization problems*, Algorithmica, 30 (2001), pp. 386–397.

[23] J. KECECIOGLU AND D. SANKOFF, *Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement*, Algorithmica, 13 (1995), pp. 180–210.

[24] J. MEIDANIS, M. E. M. T. WALTER, AND Z. DIAS, *A Lower Bound on the Reversal and Transposition Diameter*, Technical report IC-00-16, Institute of Computing, University of Campinas, Campinas, Brazil, 2000.

[25] J. MEIDANIS, M. E. M. T. WALTER, AND Z. DIAS, *Reversal Distance of Signed Circular Chromosomes*, Technical report IC-00-23, Institute of Computing, University of Campinas, Campinas, Brazil, 2000.

[26] P. A. PEVZNER AND M. S. WATERMAN, *Open combinatorial problems in computational molecular biology*, in Third Israel Symposium on the Theory of Computing and Systems (Tel Aviv, 1995), IEEE Comput. Soc. Press, Los Alamitos, CA, 1995, pp. 158–173.

[27] H. SKALETSKY ET AL., *The male-specific region of the human Y chromosome is a mosaic of discrete sequence classes*, Nature, 423 (2003), pp. 825–837.