Improved Bounds on Sorting with Length-Weighted Reversals (Extended Abstract)

Michael A. Bender *† Dongdong Ge* Simai He* Haodong Hu* Ron Y. Pinter‡
Steven Skiena* Firas Swidan‡

Abstract

We study the sorting of integer permutations and sequences by length-weighted reversals under a wide class of cost functions, namely $f(l) = l^\alpha$, where l is the length of the reversed subsequence. We present tight or nearly tight upper and lower bounds for all $\alpha \geq 0$ for two important problems: finding a best worst-case behavior algorithm for minimum cost sorting of any sequence, and approximating the optimal reversal for a given sequence. Further, we give polynomial-time algorithms to determine the optimal reversal sequence for a restricted but interesting class of sequences and cost functions. Our results substantially improve and generalize previous bounds, and have direct application in computational biology to the field of comparative genomics.

1 Introduction

We consider the problem of sorting a given permutation π of the integers $1, \ldots, n$ by repeated reversals of contiguous subsequences of π. In our model, the cost of each reversal operation depends on the length of the reversed subsequence i.e., the number of elements in it, and the total cost of the sorting process is the sum of the individual reversal costs.

The problem of sorting by reversals arises in comparative genomics, where the elements of the permutation are genes and reversal (or inversion) *mutations* occur frequently in the evolution of chromosomes. The minimum-cost reversal distance [1] is a useful measure for reconstructing the evolutionary history of an organism because the most parsimonious series of reversals transforming one sequence to another corresponds to a likely evolutionary path between the two organisms. This analysis has been applied, for example, to drosophila [9, 20], plants [2, 15], viruses [10], and mammals [8, 17].

Traditionally [3, 13], such analysis assumes that each reversal has unit cost independent of the length of the fragment reversed. This assumption, however is not completely defensible biologically. To the contrary, the mechanics of genome reversal suggest that the probabilities of reversals are dependent on fragment length. Preliminary results on genome rearrangements that assign a length-dependent cost to reversal operations appear in [6], and indicate that length indeed plays an important role in biasing certain rearrangement patterns.

Two problems on reversal distance are of particular interest:

- *Existential Distance Determination* — here we seek to identify the minimum cost sufficient to sort *any* permutation of n elements. This goal is equivalent to computing the shortest path from π to the identity permutation in the *reversal graph*, namely the weighted graph where the vertices represent the length-n permutations and there is an edge (π_1, π_2) of weight $f(\ell)$ if there exists a single reversal of length ℓ transforming sequence π_1 to sequence π_2.

- *Pairwise Reversal Distance Determination* — here we consider the problem of approximating the minimum-cost reversal sequence for a given permutation of n elements. We seek an algorithm guaranteeing that the resulting reversal sequence costs no more than some slowly growing function of n times the cost of the optimal reversal sequence.

Pinter and Skiena [18] were the first to study the length-weighted model and to provide performance guarantees for the above problems for non-constant functions $f(x)$. They considered additive functions,

*Department of Computer Science, SUNY Stony Brook, Stony Brook, NY 11794-4400, USA. Email: {bender, dge, simaihe, skiena}@cs.sunysb.edu, huhd@math.sunysb.edu.

†Supported in part by Sandia National Laboratories and NSF Grants EIA-0112849 and CCR-0208670.

‡Department of Computer Science, Technion – Israel Institute of Technology, Haifa 32000, Israel. Email: {firas, pinter}@cs.technion.ac.il.

[1] The problems of sorting a given permutation by reversals and that of finding the reversal distance between two given permutations are equivalent: simply relabel the elements of the target permutation to be the identity and use the same relabeling for the source permutation.

where a function $f(x)$ is *additive* if $f(x) + f(y) = f(x+y)$, and proved an $O(n \log^2 n)$ distance bound for such functions and gave a heuristic for approximating reversal distance to an $O(\log^2 n)$ factor.

Results. In this paper, we generalize the reversal problems to a wide class of cost functions, namely $f(\ell) = \ell^\alpha$ for any $\alpha \geq 0$. This family models a wide variety of costs that are applicable to the study of genomes; computationally, it is general enough to include unit-weighted reversals ($\alpha = 0$), additive costs ($\alpha = 1$), *subadditive* costs where $f(x)+f(y) > f(x+y)$ ($\alpha \leq 1$), and *superadditive* costs where $f(x) + f(y) < f(x+y)$ ($\alpha \geq 1$). We give the first nontrivial lower bounds on the cost of sorting by reversal for any $\alpha \geq 0$.

We present the following results:

- We prove an $\Omega(n \lg n)$ lower bound on diameter for additive distance functions. This is the first non-trivial lower bound on the diameter of length-weighted reversals, and holds even in the restricted case of sorting a 0-1 sequence instead of permutations. This analysis, inspired by Knuth's "0-1 principle" [16], leads us to other interesting results.

- We prove tight or near-tight bounds on diameter for all $\alpha \geq 0$, as summarized in Table 1. These require the development of a new and interesting class of potential function arguments.

- We give heuristics yielding improved and non-trivial approximation ratios for all $\alpha \geq 0$, also summarized in Table 1. We note that different heuristics are needed to achieve performance bounds for each of the additive, sub-additive, and super-additive cases. Having a complete suite of provably good heuristics is particularly important for biological sequence analysis, as it permits experimentation to determine which cost function best matches the observed evolutionary data. We have implemented our heuristics, and are beginning a project to do such an experimental analysis.

- We give a polynomial-time algorithm for determining the exact cost of sorting a 0-1 sequence under an additive cost measure. This result is quite surprising, given the hardness of related problems, and opens interesting questions on the complexity of sorting 0-1 sequences under more general cost models as well as sorting permutations under additive distance measures. Further, we use this result to give an $O(\log n)$ approximation algorithm for linear cost functions, improving the $O(\log^2 n)$ result from [18].

α Value	Lower Bound	Upper Bounds		Approx. Ratio	
		Perm	0/1's	Perm	0/1's
$0 \leq \alpha < 1$	$\Omega(n)$	$O(n \log n)$	$\Theta(n)$	?	$O(1)$
$\alpha = 1$	$\Omega(n \log n)$	$O(n \log^2 n)$	$\Theta(n \log n)$	$O(\log n)$	1
$1 < \alpha < 2$	$\Omega(n^\alpha)$	$\Theta(n^\alpha)$	$\Theta(n^\alpha)$	$O(\log n)$	$O(1)$
$\alpha \geq 2$	$\Omega(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	2	1

Table 1: Sorting Bounds (Lower and Upper) and Approximation Ratios for 0-1 sequences and integer permutations.

Previous Work. The problem of computing the reversal distance between two permutations and its applications to comparative genomics have received extensive attention over the last decade. There are two variants of the problem: the *unsigned* case, in which we disregard the orientation of the elements throughout the reversal process, and the *signed* case, where the directions of the elements do matter. Both measures have merit in terms of the underlying biology; in this paper we focus on the unsigned case since low-cost single element reversals suffice to give an unsigned solution the proper signs.

For the case of unit-cost, unsigned reversals, the problem of computing the reversal distance has been shown to be NP-complete by Caprara [7]; our problem, in which the cost depends on the length of the subsequence being reversed, inherits hardness for $\alpha = 0$ from this result. Kececloglu and Sankoff [13] give approximation algorithms on reversal distance that guarantee a ratio at most 2 times optimal, which Bafna and Pevzner [3] improved to a factor of 7/4 approximation; recently, Berman et al. [5] reduced this factor even further, to 1.375. Kececloglu and Sankoff [14] report on the success of heuristics and search in determining the reversal distance for chromosomes.

In a celebrated result, Hannenhalli and Pevzner [11] gave a polynomial-time algorithm for the case of unit-cost, signed reversals. An elementary exposition of the Hannenhalli-Pevzner theory appears in [4]. Recently, Siepel [19] gave an efficient algorithm for constructing/enumerating *all* minimum-length reversal sequences. The huge number of such sequences implies that other criteria must be employed to have hope of reconstructing the true evolutionary history. Ajana et al. [1] developed algorithms for users of a (signed) reversal algorithm to choose one or several possible solutions based on different criteria, including additive reversal costs; this flexibility was shown to be useful for testing certain reversal hypotheses.

Minimum-cost unsigned reversal sorting has also been studied from the other end of the cost spectrum, under models where the cost increases so dramatically

with length that only length-2 reversals can be afforded. Thus each reversal simply transposes adjacent elements. Bubble sort and insertion sort [16] both sort any permutation π using exactly one transposition for each inversion in π, thus minimizing the number of reversals.

Outline. The rest of the paper is organized as follows. Section 2 presents upper and lower bounds for the problem of sorting any permutation and any sequence of 0's and 1's. Section 3 provides algorithms for sorting a specific permutation achieving the above mentioned approximation ratios. In Section 4 we show how to optimally sort a given 0-1 sequence when $\alpha = 1$ and also for $\alpha \geq 2$. We conclude in Section 5 with a summary and open problems. Omitted proofs will appear in the full version of the paper.

2 Existential Diameter Bounds

2.1 Upper Bounds on Diameter Pinter and Skiena [18] proved an $O(n \log^2 n)$ upper bound on diameter for linear cost reversals, based on a $O(n \log n)$ upper bound for sorting 0's and 1's. To sort a sequence of 0's and 1's, recursively sort the left and right halves, and then perform one more reversal across the median for a sorting cost of

$$T(n) \leq 2T(n/2) + O(n) = O(n \log n).$$

To sort a permutation π, divide the sequence around the median and recursively sorts both halves, as before. In order to divide around the median, treat the elements less than the median as 0's and the elements greater than the median as 1's and then sort. Thus, the cost to divide the elements around the median is $O(n \log n)$, yielding a sorting cost of

$$T(n) \leq 2T(n/2) + O(n \log n) = O(n \log^2 n).$$

We will show that this algorithm is in fact also optimal or almost optimal for *all* $\alpha \geq 0$, although different proofs are needed for large and small values of α.

First, we consider the case of $0 \leq \alpha < 1$. Consider the divide-and-conquer sorting algorithm given above. The recursion relation for sorting 0's and 1's becomes $B(n) \leq 2B(n/2) + n^\alpha \leq O(n)$ when $\alpha < 1$. For arbitrary numbers the recursion for the sorting costs becomes $A(n) \leq 2A(n/2) + \leq B(n) \leq O(n \log n)$ when $\alpha < 1$. Thus,

THEOREM 2.1. *When the cost function of reversals is $f(x) = x^\alpha$, for $0 \leq \alpha < 1$, 0's and 1's can be sorted with cost $O(n)$, and arbitrary numbers can be sorted with cost $O(n \log n)$.*

Now consider the case of $1 < \alpha \leq 2$. The recursion relation for sorting 0's and 1's (bits) becomes $B(n) \leq 2B(n/2) + n^\alpha \leq O(n^\alpha)$ when $\alpha > 1$. For sorting arbitrary numbers the recursion for the sorting costs becomes $A(n) \leq 2A(n/2) + B(n) \leq O(n^\alpha)$ when $\alpha > 1$. Thus,

THEOREM 2.2. *When the cost function of reversals is $f(x) = x^\alpha$, for $1 < \alpha \leq 2$, 0's and 1's and arbitrary numbers can both be sorted with cost $O(n^\alpha)$.*

2.2 Lower Bounds on Diameter

Lower Bound for Linear Costs We first give a lower bound on the cost of sorting by reversals with a linear cost function ($\alpha = 1$). This was an open problem in [18].

THEOREM 2.3. *The cost to sort n elements by reversals with a linear cost function is $\Omega(n \log n)$, even when all elements are 0's and 1's.*

Thus, our results are tight for sorting 0-1 sequences, but an $O(n \log n)$ multiplicative gap exits for sorting permutations.

In the rest of the section, we prove Theorem 2.3. Our approach is to exhibit a difficult sorting instance. Specifically, we prove a lower bound of $\Omega(n \log n)$ on the cost to sort the length-n sequence $010101 \cdots 01$ by reversals. The proof follows a potential-function argument . Before the sorting begins, we match the i-th 0 with the i-th 1. Throughout the algorithm we keep this matching, and we let $d_i(t)$ be the current distance between the i-th 0 and i-th 1 after the t-th reversal. The potential function is

$$P(t) = \sum_i \log d_i(t).$$

When there is no ambiguity, we abbreviate $d_i(t)$ by d_i.

LEMMA 2.1. *The initial value of the potential function is 0, and the final value is $\Omega(n \log n)$.*

We now show how a reversal affects the value of d_i in the potential function by considering the i-th $(0, 1)$ pair.

OBSERVATION 2.1. *The distance d_i only changes when one element of the i-th* 0-1 *pair is* inside *the reversal and the other is* outside. *If the* 0-1 *pair is entirely* inside *or both* outside, *then d_i does not change.*

LEMMA 2.2. *A reversal of length k increases the potential $P(t)$ by at most $4k$.*

Proof. Suppose that for a reversal of length k, one of the elements is inside the reversal and the other one

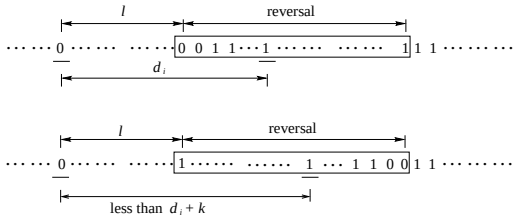

Figure 1: The sequence before and after one reversal.

is outside, so that d_i is affected by the reversal. The new distance between those two elements can increase at most $d_i + k$ because each element in the reversal is moved at most the distance k.

Assume by symmetry that the 0 is outside the reversed sequence and the 1 is inside. Suppose that the distance from the 0 to the beginning of the reversed sequence is ℓ; see Figure 1. Then the distance d_i for this pair is increased by at most $\log(k + d_i) - \log d_i = \log(1 + k/d_i) \leq \log(1 + k/\ell)$. The distance ℓ must be a natural number, and the same value of ℓ occurs at most twice in one reversal (the left and right sides of the reversed sequence).

Note that there are at most k such pairs whose distances change the value of potential function. Furthermore, a property of logarithms is that if $x \geq 1$, we have $\log(1 + x) - \log x = \log(1 + 1/x) \leq 1$, so $\log(1 + x) \leq 1 + \log x$.

Therefore, the value of the potential function increases by at most

$$\begin{aligned} 2\sum_{j=1}^{k/2} \log(1 + k/j) &\leq 2\sum_{j=1}^{k/2} (1 + \log(k/j)) \\ &\leq k + 2\sum_{j=1}^{k} \log(k/j) \\ &= k + 2\log(k^k/k!). \end{aligned}$$

Because for all $n \geq 1$, $(1 + 1/n)^n \leq e$, and $\frac{(n+1)^{n+1}/(n+1)!}{n^n/n!} = (1 + 1/n)^n$. We have $n^n/n! \leq e^n$ for all $n \geq 1$. So $\log(k^k/k!) \leq k \log e \leq \frac{3}{2}k$. Thus, the right-hand side of the inequality is less than or equal to $k + 3k = 4k$.

By combining Lemma 2.1, Lemma 2.2, we prove Theorem 2.3.

Lower Bound for $1 < \alpha < 2$

THEOREM 2.4. *There is a lower bound of $\Omega(n^\alpha)$ on the cost of sorting by reversals for cost function $f(x) = x^\alpha$, for $1 < \alpha \leq 2$, even when all the elements are 0's and 1's.*

We now prove Theorem 2.4. The same difficult sorting instance suffices. Specifically, we show that sorting the sequence $010101\cdots01$ of length n requires cost $\Omega(n^\alpha)$.

The proof follows a potential-function argument in the same spirit as that of Theorem 2.3. Before the sorting begins, we match the i-th 0 with the i-th 1. Throughout the algorithm we keep this matching and we let $d_i(t)$ be the current distance between the i-th 0 and i-th 1 after the t-th reversal. We define the potential function at time t to be

$$P(t) = \sum_i d_i^{\alpha-1}(t).$$

When there is no ambiguity, we abbreviate $d_i(t)$ by d_i.

LEMMA 2.3. *The initial value of the potential function is $\Theta(n)$. The final value of the potential function is $\Omega(n^\alpha)$.*

We now show how a reversal affects the value of the d_i's in the potential function.

LEMMA 2.4. *A reversal of length k increases the potential function by at most $2k^\alpha$.*

Proof. Suppose that d_i is affected by a given reversal of length k. Then one element of the i-th pairs is inside the reversal and the other one is outside. The new distance between those two elements can be increased to at most $d_i + k$ because the element in the reversed sequence can be moved at most a distance k.

Assume by symmetry that the 0 is outside the reversed sequence and the 1 is inside the reversed sequence. Suppose that the distance from the 0 to the beginning of the reversal sequence is ℓ. Then the contribution of the i-th pairs $d_i^{\alpha-1}$ to the potential function is increased by at most $(k + \ell)^{\alpha-1} - \ell^{\alpha-1}$.

Note that the function $x^{\alpha-2}$ is a decreasing function if $\alpha < 2$. By the Intermediate-Value Theorem, we obtain

$$(k+\ell)^{\alpha-1} - \ell^{\alpha-1} = (\alpha-1)k(\ell+\xi k)^{\alpha-2} \leq (\alpha-1)k\ell^{\alpha-2},$$

where ξ is a real number ranging from 0 to 1.

This distance ℓ must be a positive integer, and the same value of ℓ can occur at most twice, once for the left side and once for the right side of the reversed sequence. Note that there are at most k such pairs whose distances change. Furthermore, for $\alpha < 2$, $x^{\alpha-2}$ is a decreasing function about x.

Therefore the value of the potential function increases by at most $2\sum_{\ell=1}^{k/2}(\alpha - 1)k\ell^{\alpha-2}$. We now bound

the sum by an integral and evaluate the integral:

$$2\sum_{\ell=1}^{k/2}(\alpha-1)k\ell^{\alpha-2} \leq 2k\int_{x=0}^{k/2}(\alpha-1)x^{\alpha-2}dx$$
$$= 2k(k/2)^{\alpha-1} = 2^{2-\alpha}k^{\alpha} \leq 2k^{\alpha}.$$

We obtain the following corollary directly by noting that the cost to reverse a sequence of length k is k^{α}.

COROLLARY 2.1. *If a given reversal increases the potential function by Δ, then the cost of the reversal is at least $\Delta/2$.*

Theorem 2.4 follows directly from Corollary 2.1.

Lower Bound for $0 < \alpha < 1$ Our results are tight for 0's and 1's, but there is a logarithmic gap for permutations:

THEOREM 2.5. *There is a lower bound of $\Omega(n)$ on the cost of sorting by reversals for cost function $f(x) = x^{\alpha}$, for $0 \leq \alpha < 1$, even when all the elements are 0's and 1's.*

Lower Bound for $\alpha \geq 2$ Sorting by reversals is straightforward when $\alpha \geq 2$, because the problem can be solved asymptotically optimally using bubble sort; it is never worth reversing sequences of any length other than 2.

We have the following theorem in the case of $\alpha \geq 2$:

THEOREM 2.6. *Sorting by reversals with the cost function $f(x) = x^{\alpha}$, $\alpha \geq 2$, has a tight cost bound of $\Theta(n^2)$.*

Bubble-Sort or Insert-Sort immediately gives an $O(n^2)$ upper bound on the sorting cost and the $\Omega(n^2)$ lower bound follows from a potential-function argument. The concrete proof appears in the full version of the paper.

In fact, we show in Section 3.3 that bubble sort is a 2-approximation to the optimal cost to sort an arbitrary sequence for $\alpha \geq 2$ and in Section 4 that bubble sort is *optimal* when $\alpha \geq 3$.

3 Approximating the Sorting Cost

We now consider the biologically important problem of approximating the optimal cost to sort a given sequence. That is, we study the pairwise reversal distance between two particular sequences, in contrast to the the existential diameter. To achieve good approximation bounds, we need different algorithms for the different ranges of α. This is in contrast to the approach in Section 2, where one divide-and-conquer algorithm achieves optimal or nearly optimal sorting bounds for all $0 < \alpha < 2$. However, the previous sorting bounds give little indication of the optimal cost to sort a given sequence.

3.1 Approximation Algorithms for $1 < \alpha < 2$

We cannot use the sorting algorithm from Section 2.1 for $1 < \alpha < 2$ because it does not deliver a good approximation ratio. To see why, consider the permutation $n, 1, 2, 3, \ldots, n-1$. The optimal solution ($n-1$ reversals of length 2) has cost $\Theta(n)$, whereas the sorting algorithm gives cost $\Theta(n^{\alpha})$. The moral is that an approximation for $\alpha > 1$ is totally different than for $\alpha = 1$, where sorting all out-of-order regions yields an $O(\log^2 n)$ approximation [18].

We begin by enumerating properties of the cost function $f(x) = x^{\alpha}$ when $1 < \alpha < 2$.

THEOREM 3.1. *Let S_1 and S_2 be disjoint subsequences of sequence S (i. e., the subsequences could interleave but have no common elements). Then for any reversal R in S, the cost of the reversal R is at least the sum of the cost of restricted reversals with cost function $f(x) = x^{\alpha}$ $(1 < \alpha < 2)$, that is,*

$$\mathcal{C}(R) \geq \mathcal{C}(R|S_1) + \mathcal{C}(R|S_2).$$

COROLLARY 3.1. *Let S_1 and S_2 be disjoint subsets of sequence S. Then the optimal cost to sort S is at least the sum of the optimal cost to sort S_1 and S_2 for cost function $f(x) = x^{\alpha}$ $(1 < \alpha < 2)$.*

3.1.1 Approximation Algorithm for 0-1 Sequences When $1 < \alpha < 2$ We now give a divide-and-conquer approximation algorithm for sorting 0-1 sequences, which we will later use as a subroutine for more general sequences.

- Suppose the sequence has k 0's, and therefore $n-k$ 1's. Split the sequence at position k. This means that the sequence has k elements to its left and $n-k$ elements to its right. Sort both left and right subsequences recursively.
- Now the sequence is $0\cdots01110001\cdots1$, where there are two blocks of 0's and two blocks of 1's. See Figure 2. Perform a reversal to switch the first block of 1's and second block of 0's, and the sequence is sorted.

This algorithm has the following performance:

THEOREM 3.2. *The approximation algorithm gives an $O(1)$-approximation for sorting 0-1 sequences with reversals when $1 < \alpha < 2$.*

To prove Theorem 3.2, we first give a lower bound using a potential-function argument. Then, we prove that the sorting cost for this algorithm is within a constant factor of the value of initial potential.

We now define a potential function $W(S)$ for any 0-1 sequence S.

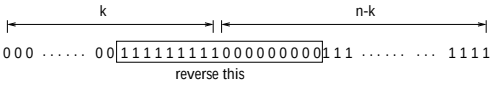

Figure 2: The sequence before and after one reversal.

DEFINITION 3.1. *For a 0-1 sequence S of length n, and any integer $1 \le i \le n$, define the number of wrong-side elements $w(i,S)$ according to position i to be the number of extra 1's in the first i elements in S plus the number of extra 0's in the last $n-i$ elements in S when compared with the sorted sequence.*

We define the potential function $W(S)$ as follows:

$$W(S) = \sum_{i=1}^{n} w(i,S)^{\alpha-1}.$$

Let $R(S)$ represent the sequence $R(S)$ after performing the reversal R.

LEMMA 3.1. *A reversal R of length r on sequence S increases the value of the potential function $W(S)$ by at most r^α, that is, $W(R(S)) - W(S) \le r^\alpha$.*

Because the cost for a reversal of length r is r^α and the value of $W(S)$ is 0 for the sorted sequence, we get the following corollary from the lemma above.

COROLLARY 3.2. *The potential $W(S)$ is a lower bound on the cost to sort the sequence S by reversals when $1 < \alpha < 2$.*

Now we prove that the algorithm sorts using cost $W(S)$, which yields an $O(1)$-approximation. We first prove a lemma about the number $w(i,S)$ of wrong-side elements.

LEMMA 3.2. *If S is a sequence of length n and i is an integer $1 \le i \le n$, and we add a 0 or 1 to right (respectively, left) end of sequence S to create a new sequence S', the value of $w(i,S)$ can only increase, i. e., $w(i,S') \ge w(i,S)$.*

COROLLARY 3.3. *If S is a sequence, S_L is the subsequence of the left k elements, and S_R is the subsequence of the right $n-k$ elements, then for any $1 \le i \le n$,*

$$w(i,S) \ge \begin{cases} w(i,S_L), & 1 \le i \le k, \\ w(i-k,S_R), & k < i \le n. \end{cases}$$

THEOREM 3.3. *For any sequence S, the cost of the algorithm to sort the S is $O(W(S))$.*

Proof. Recall that there are k 0's, so that we split the sequence S so that the lengths of S_L and S_R are k and $n-k$, respectively. Define cost $\mathcal{C}(S)$ to be the cost of this algorithm to sort sequence S.

Notice that in Step 2 we reverse $w(k,S)$ wrong-sided elements with respect to position k for a cost of $w(k,S)^\alpha$. Thus, we have the following recurrence for $\mathcal{C}(S)$:

$$\mathcal{C}(S) = \mathcal{C}(S_L) + \mathcal{C}(S_R) + w(k,S)^\alpha. \tag{3.1}$$

Now we want to prove that

$$W(S) - W(S_L) - W(S_R) \ge c w(k,S)^\alpha,$$

for some constant c. To do so, we define

$$\Delta(i) = \begin{cases} [w(i,S)]^{\alpha-1} - [w(i,S_L)]^{\alpha-1}, & i \le k, \\ [w(i,S)]^{\alpha-1} - [w(i-k,S_R)]^{\alpha-1}, & i > k. \end{cases}$$

From Corollary 3.3, we know that $\Delta(i) \ge 0$. Therefore,

$$W(S) - W(S_L) - W(S_R) = \sum_{i=1}^{n} \Delta(i) \ge \sum_{i=k-w(k,S)/4}^{k+w(k,S)/4} \Delta(i).$$

Because shifting the position i in the sequence to the left or right by 1 can change the number of wrong-sided elements by at most 2, for any $1 \le j \le w(k,S)/4$, $w(k-j,S) \ge w(k,S) - 2j$ and $w(k+j,S) \ge w(k,S) - 2j$.

Notice $w(k,S_L) = 0$ and $w(0,S_R) = 0$. For the same reason as above we know that for any $1 \le j \le w(k,S)/4$, $w(k-j,S_L) \le w(k,S_L) + 2j = 2j$ and $w(j,S_R) \le w(0,S_R) + 2j = 2j$.

Thus, we have

$$W(S) - W(S_L) - W(S_R) \ge \sum_{i=k-w(k,S)/4}^{k+w(k,S)/4} \Delta(i)$$

$$\begin{aligned} &\ge \sum_{j=0}^{w(k,S)/8} \left\{ [w(k,S) - 2j]^{\alpha-1} - (2j)^{\alpha-1} \right\} \\ &\ge \frac{w(k,S)}{8} \left[(3/4)^{\alpha-1} - (1/4)^{\alpha-1} \right] w(k,S)^{\alpha-1} \\ &= \frac{1}{8} \left[(3/4)^{\alpha-1} - (1/4)^{\alpha-1} \right] w(k,S)^\alpha. \end{aligned}$$

We do induction on the length of sequence S and the conjecture is $W(S) \ge cC(S)$, where

$$c = \min \left\{ \frac{1}{8} \left[(3/4)^{\alpha-1} - (1/4)^{\alpha-1} \right], \frac{1}{2} \right\}.$$

The base case is a sequence of length 2, in which case $W(S) = 2^{\alpha-1}$ and $C(S) = 2^\alpha$. If $W(S') \ge cC(S')$ for all S' of length at most k, then for a sequence S of length $k+1$, we know that S_L and S_R are of length at most k. So

$$\begin{aligned} W(S) &\ge W(S_L) + W(S_R) + c w(k,S)^{\alpha-1} \ge \\ &c[C(S_L) + C(_R') + w(k,S)^{\alpha-1}] = cC(S). \end{aligned}$$

Thus, for all sequence S, $W(S) \ge cC(S)$.

Thus, we establish Theorem 3.2.

3.1.2 Approximation Algorithm for Arbitrary Sequences When $1 < \alpha < 2$ We give the following approximation algorithm for sorting an arbitrary sequence S, which is a surprising enhancement of the sorting algorithm from Section 2.1. We add one intermediate step: after we divide the sequence S into two halves about the median, we recursively sort each half to return the elements to the same order as an S. Only then do we recursively sort each half. At first glance, this modification seems to increase the complexity, but, in fact the complexity is reduced enough to approximate the optimal sorting cost to within a logarithmic factor.

1. Treat the elements less than the median as 0 and those greater than the median as 1, and sort them as 0-1 sequence using the algorithm from Section 3.1.1.
2. Return the elements in each half to the original order. To do this, reverse the restricted permutation of Step 1 on both subsequences. From Theorem 3.1, this step is at most double the cost of Step 1.
3. Now there is a new left side sequence S'_L and new right side sequence S'_R, which are disjoint subsequences of the original sequence S. Sort S'_L and S'_R recursively.

This algorithm has the following performance:

THEOREM 3.4. *The algorithm for sorting arbitrary sequences is an $O(\log n)$ approximation algorithm when $1 < \alpha < 2$.*

Proof. Let $\text{OPT}(S)$ be the optimal cost to sort sequence S. The cost for Step 1 is at most $O(\text{OPT}(S))$ by Theorem 3.3. The cost for Step 2 is at most the cost of Step 1, and hence is at most $O(\text{OPT}(S))$. This follows from Theorem 3.1, and because the inverse of a reversal is itself, and we are just doing the inverse restricted permutation on left and right subsequences. We also know from Theorem 3.1 that $\text{OPT}(S'_L) + \text{OPT}(S'_R) \leq \text{OPT}(S)$ because S'_L and S'_R are disjoint subsequences of the original sequence S. Thus, the cost of Step 3 is

$$\mathcal{C}(S) = \mathcal{C}(S_L) + \mathcal{C}(S_R) + \text{cost for Steps 1 and 2},$$

and with a simple induction we establish the $O(\log n)$ approximation.

3.1.3 Approximation Algorithm for Arbitrary Sequences When $\alpha = 1$

ALGORITHM 3.1. BetterRatioReversalSort(a, b, π)

- zerOneSort(a, b, π)
- reorder$\left(a, \left\lfloor \frac{b-a}{2} \right\rfloor, \pi\right)$
- reorder$\left(\left\lceil \frac{b-a}{2} \right\rceil, b, \pi\right)$
- BetterRatioReversalSort$\left(a, \left\lfloor \frac{b-a}{2} \right\rfloor, \pi\right)$
- BetterRatioReversalSort$\left(\left\lceil \frac{b-a}{2} \right\rceil, b, \pi\right)$

LEMMA 3.3. *Algorithm 3.1 has an approximation ratio in $O(\log n)$*

Proof. In each recursive round, zerOneSort (Algorithm 4.1) costs less than the optimal permutation sort, since sorting the permutation implies sorting the induced 0-1 series. The result follows because we perform $\theta(\log n)$ rounds of zerOneSort.

3.2 Approximation Algorithm for $0 \leq \alpha \leq 1$ We first give an $O(\log n)$-approximation algorithm for sorting sequences of 0's and 1's. A sequence composed of 0's and 1's can be viewed as composed of zero blocks (0's) and one blocks (1's). Without loss of generality, suppose the sequence is in this form: $z_1 w_2 z_2 \ldots w_m z_m$, where w_i and z_i represent the i-th one and zero block, respectively. By symmetry, all other cases can be reduced to this case. We have the following lower bound:

LEMMA 3.4. *A lower bound on $\text{OPT}(S)$ to sort a sequence $S = w_1 z_1 w_2 z_2 \ldots w_m z_m$ by reversals when $0 \leq \alpha \leq 1$ is*

$$V(S) = \tfrac{1}{2} \sum_{i=1}^{m} \left(w_i^\alpha + z_i^\alpha\right).$$

Now we give the approximation algorithm for 0-1 sequence based on divide-and-conquer:

1. Map each block of 0's or 1's to a single element 0 or 1 in a new sequence S', ignoring the block of 0's at the leftmost position and the block of 1's at the rightmost position if they exist.
2. Use the divide-and-conquer algorithm from Section 2 to determine the reversals to sort sequence S'.
3. Map back each elements in S' onto a block in S map each reversal back according to the same mapping.

Thus, we are just performing the standard divide-and-conquer algorithm from Section 2, but on the the blocks of 0's and 1's. The performance guarantees are based on the following structural lemma:

LEMMA 3.5. *In the above divide-and-conquer algorithm on the blocks, each element appears in at most $\log n$ reversals.*

THEOREM 3.5. *The divide-and-conquer algorithm on the 0-1 blocks is an $O(\log n)$-approximation when $0 < \alpha < 1$.*

3.2.1 $O(1)$ Approximation Algorithm for 0-1 Sequences When $0 \leq \alpha < 1$ We obtain an $O(1)$-approximation for the same problem by improving the splitting. The algorithm is as follows:

- If there are any 0-1 blocks of size at least $n/3$, perform a reversal of length at most n to move this block to the edge of the sequence (a 0 block moves to the front, a 1 block moves to the back). Then remove this block from the sequence S and sort the rest of sequence S' recursively.
- If there are no blocks of 0's or 1's of size at least $n/3$, then there exits a block edge at a distance at least $n/3$ from both ends. Split the whole sequence S at this edge to form left and right subsequences S_L and S_R. Sort S_L and S_R recursively, then perform a reversal of length at most $n = S_L + S_R$, and the sequence S is sorted.

This algorithm sorts any 0-1 sequence S with cost at most $\mathcal{C}(S)$.

THEOREM 3.6. *For a 0-1 sequence S of length n, if it is formed by blocks of 0's and 1's $B_1, B_2, \ldots, B_m$ of size $b_1, b_2, \ldots, b_m$, then this algorithm will sort this sequence S with cost at most $\mathcal{C}(S)$.*

3.3 Approximation Algorithm for $\alpha \geq 2$ As described in Section 4, bubble sort is optimal for sorting 0-1 sequences when $\alpha \geq 2$, a 2-approximation for sorting arbitrary sequences when $2 \leq \alpha < 3$, and optimal (see Section 4) when $\alpha \geq 3$.

4 Polynomial-Time Algorithms for 0-1 Sorting

In this section, we provide polynomial-time algorithms for sorting 0-1 series when $\alpha = 1$, $\alpha \geq 2$ and for sorting permutations when $\alpha \geq 3$. The main idea in all these cases is to give a sufficiently constrained property of an optimal solution, enabling it to be found in polynomial-time.

Given a reversal, consider the first and last blocks that it affects. If the values of these blocks are identical, that is both equal 0 or 1, we call the reversal *useless*. If one of those two blocks is contained in a bigger block, we call the reversal a *cutting* reversal. If the reversal affects more than two blocks, we call the reversal *complex*. We call a reversal that is not complex *simple*.

A reversal sequence $\rho_1, \ldots, \rho_n$ *separates* a 0-1 series S, if performing the reversals on the series joins all 0's and all 1's of S in two blocks. If the the separation is such that the 0's appear before the 1's, it will be called *positive separation orientation*. Otherwise it will be called *negative separation orientation*. Notice that positive separation is equivalent to sorting.

Given a reversal sequence $\rho_1, \ldots, \rho_n$ acting on a 0-1 series $S = s_1, s_2, \ldots, s_m$ where $s_i \in \{0, 1\}$ denote the number of reversals in which element s_i takes part by $n(s_i)$. This will be called the *reversal count* of s_i.

If a subseries $s_i, \ldots, s_j$ of S is never cut by a reversal sequence $\rho_1, \ldots, \rho_n$, one can define the reversal count of the subseries, denoting it by $n(s_i, \ldots, s_j)$, as the number of reversals in which the subseries takes part.

4.1 Sorting 0-1 series when $\alpha = 1$ We will show that for linear cost functions $f(x) = ax, a > 0$ an optimal reversal sequence contains neither useless nor cutting reversals, and that an optimal reversal sequence containing no complex reversals exists. The following equation relating the reversal counts to the reversal sequence cost will be useful for the proofs.

$$\sum_{i=1}^{n} f(|\rho_i|) = \sum_{j=1}^{m} f(n(s_j)) \tag{4.2}$$

LEMMA 4.1. *A reversal sequence containing a useless reversal cannot be optimal.*

Proof. Consider a useless reversal affecting elements $0^{i_1}1^{i_2}\ldots1^{i_{k-1}}0^{i_k}$ of a 0-1 series. Assume that $i_1 \geq i_k$. Consider a modified reversal affecting the elements $0^{i_1-i_k}1^{i_2}\ldots1^{i_{k-1}}$. The modified reversal has a lower cost, where the 0-1 series that it produces is identical to the 0-1 series that the original useless reversal produced. Therefore, the original reversal sequence can not be optimal. The case $i_1 < i_k$ or if the affected elements start and end with a 1-block is handled similarly.

LEMMA 4.2. *A reversal sequence containing a cutting reversal can not be optimal.*

Proof. WLOG, assume a sorting reversal sequence $\rho_1, \ldots, \rho_n$ contains no useless reversals. Consider the last cutting reversal ρ_j. Let 0^i be a block that the reversal cut, such that it affects x 0's of the block, $0 < x < i$. Notice that $n(0^x), n(0^{i-x})$ are well defined since ρ_j is the last cutting reversal in the reversal sequence.

If $n(0^{i-x}) \leq n(0^x)$, then excluding 0^x from the cutting reversal, and including it in all reversals in which 0^{i-x} takes part, i.e. moving the block 0^i as a unit through the reversals affecting 0^{i-x}, yields a reversal sequence having a cost less than or equal to the original reversal sequence- by Equation 4.2. However, the modified reversal, that is the cutting reversal after excluding 0^x from it, is a useless reversal. Therefore by Lemma 4.1 the modified sequence, and hence the original sequence, cannot be optimal. The case $n(0^{i-x}) > n(0^x)$, or if the reversal cuts a 1-block, is handled similarly.

Proving that an optimal reversal sequence exists that contains no complex reversals requires substantial case analysis. We sketch the proof below.

LEMMA 4.3. *There exists an optimal reversal sequence containing no complex reversals.*

Proof. By induction on n, the number of 0-1 blocks in a 0-1 series.

The claim is obvious for $n = 2$. Suppose it is true for n. We need to prove it for $n + 2$. Consider an optimal reversal sequence. By Lemmas 4.1 and 4.2, the sequence contains neither useless nor cutting reversals. Since the first reversal, denote it ρ_1, is neither useless nor cutting, the 0-1 series after performing ρ_1 contains n blocks. According to the induction assumption, we can assume that all other reversals are simple.

If ρ_1 is simple, we are done. Otherwise, ρ_1 is complex. One can prove the following.

OBSERVATION 4.1. *WLOG, $\rho_i, i > 1$ are not commutative* [2] *to ρ_1.*

Assuming that ρ_1 is complex and is not commutative to any other reversal, denote the subseries that ρ_1 affects with s_2, denote the subseries to its left with s_1 and the one to its right with s_3 [3]. Denote the last reversal in the sequence with ρ_l[4]. Consider the following modifications in the reversal sequence: perform the reversals $\rho_2, \ldots, \rho_l$ restricted to s_2. Notice that this will cause s_2 to be negatively separated. Perform the reversals $\rho_2, \ldots, \rho_{l-1}$ restricted to s_1, s_3. This separates s_1, s_3 as well, since ρ_1, ρ_l do not affect the separation state of s_1, s_3.

Considering the separation's orientation of s_1, s_3 one can prove the following.

OBSERVATION 4.2. *The separation orientation of s_1, s_3 is opposed to that of s_2, or the separation orientation of one of s_1, s_3 is opposed to s_2 while the orientation of the other is identical to s_2.*

We will present the prove of the first case. The prove of the second uses similar ideas.

OBSERVATION 4.3. *If the separation orientation of s_1, s_3 is opposed to that of s_2, the 0-1 series has a pattern of the form $01 \cdots 10 \cdots 01$ after performing the restricted reversals.*

From a counting point of view the following holds.

OBSERVATION 4.4. *If the separation orientation of s_1, s_3 is opposed to that of s_2, the reversal counts of the elements of s_2, the 0's of s_3 and the 1's of s_1 after performing the restricted reversals are smaller than the reversal counts of the same elements in the original reversal sequence.*

Performing a last reversal on the blocks $1 \cdots 10 \cdots 0$ will separate the series- Observation 4.3. The cost of the modified reversal sequence, that is complex free, is not greater than the original one, by Observation 4.4.

A reversal sequence having no useless, cutting or complex reversals, will be called a *good* sequence. Given a good sequence, we want to characterize the counting pattern of the affected 0-1 series. This characterization enables us to find an optimal reversal sequence using dynamic programming.

We represent a 0-1 series $1^{w_1}, 0^{w_2}, \ldots, 0^{w_{2l}}$ by the lengths of each block, that is we let $w_1, w_2, \ldots, w_{2l}$ denote a 0-1 series. Given a good reversal sequence sorting $w_1, w_2, \ldots, w_{2l}$, let c_i denote the number of reversals in which w_i takes part.

LEMMA 4.4. *Given a 0-1 series $w = w_1, w_2, \ldots, w_{2l}$, and a good reversal sorting sequence, there exist i, j such that w_i corresponds to a 1-block, w_j corresponds to a 0-block, $c_i = c_j = 1$ and $j > i$.*

Lemma 4.4 enables us to search for an optimum efficiently.

ALGORITHM 4.1. zerOneSort

Given a 0-1 series $w_1, w_2, \ldots, w_{2l}$, for each i, j such that $j > i$, w_j corresponds to a 0-block and i corresponds to a 1-block, calculate the following, and take the minimum over all results.

- Separate positively and optimally segments $w_1, \ldots, w_{i-1}$ and $w_{j+1}, \ldots, w_{2l}$.
- Separate negatively and optimally segment $w_i, \ldots, w_j$.
- Perform a last sorting reversal.
- The sum of all the above costs will give the minimum cost of sorting w under the condition $c_i = c_j = 1$.

LEMMA 4.5. *The time complexity of zerOneSort is in $O(n^3)$. The space complexity of zerOneSort is in $O(n^2)$.*

THEOREM 4.1. *Algorithm zerOneSort sorts a 0-1 series optimally, given that the cost function is linear.*

[2] One can prove that two reversals are commutative iff one contains the other, or they are disjoint.

[3] s_1, s_3 might be empty.

[4] One can prove that $l = (n + 2)/2$.

5 Conclusions

We have presented a comprehensive analysis of the problem of sorting sequences with reversals for a wide class of cost functions relevant to comparative genomics. Many interesting open problems remain beyond the relatively minor gaps left between our upper and lower bounds. It would be interesting to develop algorithms for even more general cost functions, particularly cost functions defined by a small number of size classes, e.g. one price for 'short' reversals and another price for 'long' reversals. These are of interest both combinatorially and biologically. Another interesting open question is whether 0-1 sequences can be sorted in polynomial time for *all* values of α.

References

[1] Y. Ajana, J. Lefebvre, E. Tillier, and N. El-Mabrouk. Exploring the set of all minimal sequences of reversals — an application to test the replication-directed reversal hypothesis. In *Second Int. Workshop on Algorithms in Bioinformatics (WABI'02)*, volume 2452, pages 300–315. Springer-Verlag Lecture Notes in Computer Science, 2002.

[2] V. Bafna and P. Pevzner. Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history of X chromosome. *Molecular Biology and Evolution*, 1994.

[3] V. Bafna and P. Pevzner. Genome rearrangements and sorting by reversals. *SIAM J. Computing*, 25:272–289, 1996.

[4] A. Bergeron. A very elementary presentation of the hannenhalli-pevzner theory. In *Proc. 12th Symp. Combinatorial Pattern Matching (CPM)*, volume 2089, pages 106–117. Springer-Verlag Lecture Notes in Computer Science, 2001.

[5] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375–approximation algorithm for sorting by reversals. In *10th European Symposium on Algorithms*, pages 200–210. Springer-Verlag Lecture Notes in Computer Science, 2002.

[6] M. Blanchette, T. Kunisawa, and D. Sankoff. Parametric genome rearrangement. *Gene*, 172:GC:11–17, 1996.

[7] A. Caprara. Sorting by reversals is difficult. In *Proc. First Annual International Conference on Computational Molecular Biology (RECOMB'97)*, pages 75–83. ACM Press, 1997.

[8] M. Davisson. X-linked genetic homologies between mouse and man. *Genomics*, 1:213–227, 1987.

[9] T. Dobzhansky and A.H.Sturtevant. Inversions in the chromosomes of *drosophila pseudoobscura*. *Genetics*, 23:28–64, 1938.

[10] S. Hannenhalli, C. Chappey, E. Koonin, and P. Pevzner. Scenarios for genome rearrangements: Herpesvirus evolution as a test case. In *Proc. of 3rd Intl. Conference on Bioinformatics and Complex Genome Analysis*, 1994.

[11] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, 46:1–27, 1999.

[12] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the 8th Annual Symposium on Discrete Algorithms(SODA)*, pages 344–351. ACM Press, 1997. Also in Proc. RECOMB 97, page 163.

[13] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for the inversion distance between two permutations. In *Proc. of 4th Ann. Symp. on Combinatorial Pattern Matching*, Lecture Notes in Computer Science 684, pages 87–105. Springer Verlag, 1993.

[14] J. Kececioglu and D. Sankoff. Efficient bounds for oriented chromosome inversion distance. In *Proc. of 5th Ann. Symp. on Combinatorial Pattern Matching*, pages 307–325. Springer-Verlag LNCS 807, 1994.

[15] E. B. Knox, S. R. Downie, and J. D. Palmer. Chloroplast genome rearrangements and evolution of giant lobelias from herbaceous ancestors. *Mol. Biol. Evol.*, 10:414–430, 1993.

[16] D. Knuth. *The Art of Computer Programming, Vol. III: Sorting and Searching*. Addison-Wesley, Reading, MA, 1973.

[17] J. H. Nadeau and B. A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proc. Natl. Acad. Sci. USA*, 81:814–818, 1984.

[18] R. Pinter and S. Skiena. Sorting with length-weighted reversals. In *Proc. 13th International Conference on Genome Informatics (GIW 2002)*, pages 173–182. Universal Academic Press, 2002.

[19] A. Siepel. An algorithm to find all sorting reversals. In *Proc. Sixth Annual International Conference on Computational Molecular Biology (RECOMB'02)*, pages 281–290. ACM Press, 2002.

[20] A. H. Sturtevant and T.Dobzhansky. Inversions in the third chromosome of wild races of *drosophila pseudoobscura*, and their use in the study of the history of the species. *Proc. Nat. Acad. Sci.*, 22:448–450, 1936.