Sorting by Restricted-Length-Weighted Reversals

Thach Cam Nguyen[1]*, Hieu Trung Ngo[2], and Nguyen Bao Nguyen[2]

[1] *Department of Mathematics, and* [2] *School of Computing, National University of Singapore, Singapore 117543.*

Classical sorting by reversals uses the unit-cost model, that is, each reversal consumes an equal cost. This model limits the biological meaning of sorting by reversal. Bender and his colleagues extended it by assigning a cost function $f(l) = l^\alpha$ for all $\alpha \geq 0$, where l is the length of the reversed subsequence. In this paper, we extend their results by considering a model in which long reversals are prohibited. Using the same cost function above for permitted reversals, we present tight or nearly tight bounds for the worst-case cost of sorting by reversals. Then we develop algorithms to approximate the optimal cost to sort a given 0/1 sequence as well as a given permutation. Our proposed problems are more biologically meaningful and more algorithmically general and challenging than the problem considered by Bender *et al.* Furthermore, our bounds are tight and nearly tight, whereas our algorithms provide good approximation ratios compared to the optimal cost to sort $0/1$ sequences or permutations by reversals.

Key words: reversal, sorting, length-weighted, permutation

Introduction

Biological motivation

Given the gene order of a chromosome, a reversal takes a segment and reverses the order of the sequences in it. This is one of the most important mutations at the chromosome level. Indeed, reversal is believed to be the most common in these mutations (*1*). For instance, this fact has been reported on bacteria (*2*), plants (*3*), and fruit fly (*4*).

The minimum-cost reversal distance thus becomes a useful measure for reconstructing the evolutionary history of organisms, because the most parsimonious series of reversals transforming one sequence to another is likely to be the evolutionary path between two organisms. However, most of the algorithmic works to date on the problem of sorting by reversals have used the unit-cost model, that is, each reversal consumes a unit cost. This model implicitly assumes simple evolutionary paths in which reversal mutations of different lengths are considered equally likely to happen. It is therefore not very biologically meaningful. Instead, the mechanics of genome reversals suggest that the probabilities of reversals depend on the fragment lengths (*5*). Furthermore, not all reversals can occur in the evolutionary path. The reversals whose lengths are greater than some limits should be forbidden because they affect the organisms seriously and destroy all the specific characteristics of their genomes.

Motivated by this characteristic of reversals, many works assigned length-sensitive costs to reversal operations and indicated that length indeed plays an important role in biasing certain rearrangement patterns. In this work, we generalize previous results on the problems of sorting by length-weighted reversals and investigate more general variants by imposing the condition that the reversals acting on the fragments longer than a certain length are prohibited.

Problem definition

Since we can always relabel genes so that the resulting permutation is the identity permutation, the problem of finding reversal distance is equivalent to that of finding the most economic series to transform a permutation into the identity one, and it is often regarded as "sorting by reversals". Our problem, sorting by restricted-length-weighted reversals, consists of three kinds of input: a 0/1 sequence or permutation S, a cost function f on the length of the reversals, and a positive integer k. The aim is to find a minimum-cost series of reversals whose lengths do not exceed k to sort S.

In this paper, we consider a wide class of cost func-

*** Corresponding author.**
E-mail: matnct@nus.edu.sg

tions, namely $f(l) = l^\alpha$ for all $\alpha \geq 0$, where l is the length of the reversed segment. Denoting the minimum cost to sort S by SBRLR (S, k, α), we address the following two problems:

1. Determining the *diameter* of sorting by restricted-length-weighted reversals, that is, finding the maximum cost C_l such that there is *one* permutation S whose cost to sort by restricted-length-weighted reversals is at least C_l, and the minimum cost C_u such that any permutation can be sorted with cost at most C_u.

2. Approximating SBRLR (S, k, α).

Related work

There is a huge number of literature on sorting by reversals. The most exciting results were due to Kececioglu and Sankoff (*6*), Hannenhalli and Pevzner (*7*), and Caprara (*8*).

Kececioglu and Sankoff (*6*) gave a 2-approximation algorithm, which is the first performance guaranteed approach for sorting by reversals. Bafna and Pevzner (*9*) then developed the notion of *breakpoint graph* of permutation and gave a better approximation algorithm.

Based on this notion, Hannenhalli and Pevzner (*7*) stated a novel dual theorem and gave the first polynomial time algorithm to sort a signed permutation, known as the HP's algorithm. This algorithm took $O(n^2)$ time to calculate the reversal distance between the genomes and $O(n^4)$ time to find the optimal series of reversals to sort a permutation.

On the other hand, Caprara (*8*) proved that sorting by reversals is NP-Hard for the general (unsigned) case. This result changed the focus of the study of sorting by reversals to the signed case. Since then, many simplifications and improvements of the HP's algorithm have been developed.

Berman and Hannenhalli (*10*) gave an algorithm that sorted a permutation in $O\big(n^2\alpha(n)\big)$ time where α is the inverse Ackerman function. Kaplan *et al* (*11*) then simplified the concept of hurdles and gave a simple $O(n^2)$ algorithm. Bader *et al* (*12*) gave an optimal $O(n)$ algorithm to calculate the reversal distance between genomes. Recently, Kaplan and Verbin (*13*) gave an interesting randomized algorithm, which Tannier and Sagot (*14*) derandomized to get a deterministic one, to sort by reversals in $O(n\sqrt{n \log n})$ time.

The lengths of reversals were first taken into account by Chen and Skiena (*15*), in which only reversals whose lengths are some constant k are allowed. They gave an algorithm to sort all circular n-permutations using $O(n^2/k + kn)$ k-reversals while proving that there exists permutations requiring $\Omega(n^2/k^2 + n)$ k-reversals to sort. Continuing on this track, Pinter and Skiena (*16*) studied the linear cost model, which has an upper bound of $O(n \log^2 n)$ on the cost of sorting any n-element permutation and a guaranteed approximation ratio of $O(\log^2 n)$ times the optimal cost.

Recently, Bender *et al* (*17*) presented tight and nearly tight lower and upper bounds for a wide class of cost models $f(l) = l^\alpha$ where $\alpha \geq 0$. They also gave good approximation algorithms for sorting linear permutations. Swidan *et al* (*18*) extended the results for the cases of signed and circular permutations. Table 1 summarizes the results obtained in Bender *et al*. Note that the lower and upper bounds shown in the table are true not only for linear permutations, but also for circular permutations. However, the approximation ratios are only for linear permutations.

Table 1 The Results Obtained in Bender *et al* (*17*)

α value	Lower bounds	Upper bounds		Approximation ratio	
		Permutation	0/1 sequence	Permutation	0/1 sequence
$0 \leq \alpha < 1$	$\Omega(n)$	$O(n \log n)$	$\Theta(n)$		$O(1)$
$\alpha = 1$	$\Omega(n \log n)$	$O(n \log^2 n)$	$\Theta(n \log n)$	$O(\log n)$	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 for $\alpha < 3$	1
				1 for $\alpha \geq 3$	

Results and Discussion

In this paper, we considered sorting 0/1 sequences and permutations using restricted-length-weighted reversals. First, we proved tight and nearly tight (upper and lower) bounds for all $\alpha \geq 0$. Our results on these bounds are for linear permutations as well as circular permutations. When $k = O(n/\log n)$ (k is the max-

imum length of each reversal and n is the length of the permutation), our upper bounds are tight for both classes of permutations and 0/1 sequences. Second, we gave approximation algorithms yielding nontrivial approximation ratios for all $1 \leq \alpha < 2$.

Table 2 summarizes our results. Note that the lower and upper bounds in the upper table are true for both linear and circular permutations. Also, when $\alpha \geq 2$, short reversals are always preferred to long ones; thus the results for our problem are identical with those for the problem considered in Bender *et al.*

Table 2 The Results Obtained in Our Work

α value	Lower bounds	Upper bounds	
		Permutation	0/1 sequence
$0 \leq \alpha < 1$	$\Omega(n + n^2k^{\alpha-2})$	$O(n \log n + n^2k^{\alpha-2})$	$\Theta(n + n^2k^{\alpha-2})$
$\alpha = 1$	$\Omega(n \log n + n^2/k)$	$O(n \log n \log k + n^2/k)$	$\Theta(n \log k + n^2/k)$
$1 < \alpha < 2$	$\Omega(n^2k^{\alpha-2})$	$\Theta(n^2k^{\alpha-2})$	$\Theta(n^2/k^{\alpha-2})$
$\alpha \geq 2$	$\Omega(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$

k value	Approximation ratio	
	Permutation	0/1 sequence
$k = \Omega(n)$	$O(\log n)$	$O(1)$
other k's	$2(\log_2 n)^2 + \log_2 n$	$2\log_2 n + 1$

Lower bounds

We establish the lower bounds for sorting both linear and circular permutations of n elements based on the lower bounds of 0/1 sequences, and analyze the case when $\alpha \leq 2$ for the other case is trivial. Our lower bounds are asymptotically tight because there exists algorithms to sort any 0/1 sequence with the cost asymptotically equal to these bounds.

Linear permutations

The lower bound in this case is established based on the notion of *inversion*. Given a sequence $S = s_1 s_2 \ldots s_l$, an inversion is a pair of positions $i < j$ such that $s_i > s_j$. The number of inversions in a 0/1 sequence of x 0's and y 1's is at most xy, which is smaller than $(x + y)^2/4$. Therefore, a reversal of length l can remove at most $l^2/4$ inversions from a 0/1 sequence.

The sorted sequence has no inversion. Noticing that the sequence $S = 11\ldots100\ldots0$, where there are equal numbers of 0's and 1's, contains $n^2/4$ inversions, we establish our lower bounds for linear permutations based on S.

Theorem 1: When $\alpha < 2$, the cost required to sort S is at least $n^2k^{\alpha-2}$.

Proof: Let $l_1, l_2, \ldots, l_p$ be the length of reversals in the optimal reversal series to sort S, such that $l_i \leq k$ for all $1 \leq i \leq p$. Let n_i be the number of inversions removed by performing reversal l_i, we have $n_i \leq l_i^2/4$. The optimal cost is $\sum_{i=1}^{p} l_i^\alpha = l_1^\alpha + l_2^\alpha + \cdots + l_p^\alpha$. We have:

$$\begin{aligned} & l_1^\alpha + l_2^\alpha + \cdots + l_p^\alpha \\ = \;& n_1 \tfrac{l_1^\alpha}{n_1} + n_2 \tfrac{l_2^\alpha}{n_2} + \cdots + n_p \tfrac{l_p^\alpha}{n_p} \\ \geq \;& 4n_1 \tfrac{l_1^\alpha}{l_1^2} + 4n_2 \tfrac{l_2^\alpha}{l_2^2} + \cdots + 4n_p \tfrac{l_p^\alpha}{l_p^2} \\ \geq \;& 4n_1k^{\alpha-2} + 4n_2k^{\alpha-2} + \cdots + 4n_pk^{\alpha-2} \\ = \;& 4k^{\alpha-2}(n_1 + n_2 + \cdots + n_p) \\ = \;& n^2k^{\alpha-2} \end{aligned}$$

Hence, the theorem holds.

The cost of the optimal reversal series to sort a sequence when restricted to reversals whose lengths do not exceed k must be greater than or equal to the minimum cost to sort the same sequence when using any reversals. Thus, we combine Theorem 1 and the lower bounds established in Bender *et al* to get the following lower bounds on the cost of the optimal reversal series in which no reversal has the length greater than k for different α:

$$\begin{array}{ll} \Omega(n + n^2k^{\alpha-2}) & \text{for } 0 \leq \alpha < 1, \\ \Omega(n \log n + n^2/k) & \text{for } \alpha = 1, \\ \Omega(n^2k^{\alpha-2}) & \text{for } 1 < \alpha < 2. \end{array}$$

Circular permutations

Consider the circular sequence $S = 0101\ldots01$. We choose an arbitrary 0 from S, and count the other 0's and 1's in that sequence. Let d_i denote the distance between the i^{th} 0 and the i^{th} 1 and $d_S =$

$\sum_{i=1}^{n/2} d_i = n/2$. After performing an optimal reversal series $l_1, l_2, \ldots, l_p$, we obtain the sorted sequence S' with $d_{S'} = \Theta(n^2)$.

The optimal cost to sort S is $\sum_{i=1}^{p} l_i^\alpha = l_1^\alpha + l_2^\alpha + \cdots + l_p^\alpha$. Consider a reversal of length l, it can increase the distance between the i^{th} 0 and the i^{th} 1 of S by at most l, and it can increase at most l such distances. Thus a reversal of length l can increase d_S by at most l^2. Let x_i be the amount that d_S increases by performing the reversal l_i in the series, we have $x_i \leq l^2$. Then

$$\begin{aligned} & l_1^\alpha + l_2^\alpha + \cdots + l_p^\alpha \\ = \; & x_1 \tfrac{l_1^\alpha}{x_1} + x_2 \tfrac{l_2^\alpha}{x_2} + \cdots + x_p \tfrac{l_p^\alpha}{x_p} \\ \geq \; & x_1 \tfrac{l_1^\alpha}{l_1^2} + x_2 \tfrac{l_2^\alpha}{l_2^2} + \cdots + x_p \tfrac{l_p^\alpha}{l_p^2} \\ \geq \; & x_1 k^{\alpha-2} + x_2 k^{\alpha-2} + \cdots + x_p k^{\alpha-2} \\ = \; & k^{\alpha-2}(x_1 + x_2 + \cdots + x_p) \\ = \; & \Omega(n^2 k^{\alpha-2}) \end{aligned}$$

Thus, we have the following theorem:

Theorem 2: When $\alpha < 2$, the minimum cost required to sort S must be at least $\Omega(n^2 k^{\alpha-2})$.

Again, combining Theorem 2 and the lower bounds established in Swidan *et al* (*18*) gives us the lower bounds of sorting circular permutations, which are identical with those of sorting linear permutations.

Upper bounds

Here we present the algorithms for sorting any n-element permutation and analyze their worst-case cost. The worst-case costs used by these algorithms are the upper bounds on the diameter of sorting by restricted-length-weighted reversals. We again note that bubble sort gives a tight upper bound for the case $\alpha \geq 2$. Furthermore, by "cutting" a circular permutation at any point and treat it as a linear permutation, we obtain an algorithm with the same asymptotic cost to sort a circular permutation. Therefore, we only consider sorting a linear permutation with $\alpha < 2$.

To sort a 0/1 sequence $S = s_1 s_2 \ldots s_n$ with only reversals whose lengths do not exceed k, we divide S into small segments such that for each segment (i, j):

1. There are at most $\lfloor k/2 \rfloor$ 0's and at most $\lfloor k/2 \rfloor$ 1's in $\{s_i, s_{i+1}, \ldots, s_j\}$.

2. There are either $\lfloor k/2 \rfloor$ 0's or $\lfloor k/2 \rfloor$ 1's in $\{s_i, s_{i+1}, \ldots, s_{j+1}\}$.

We first sort each segment separately based on the algorithms described in Bender *et al* for 0/1 sequences and on the corresponding cost model. Then we perform bubble sort on blocks of 0's and 1's, in which each swap is mimicked by a reversal of length not greater than k.

There are $O(n/k)$ segments, and each segment has length of $O(k)$. Let us denote the cost to sort each subsequence whose length l does not exceed k by $B'(l)$. The cost to sort a permutation is:

$$B(n) \leq O(n/k)B'(k) + O(n^2/k^2)f(k).$$

To sort a permutation π, it needs to divide the sequence around the median and recursively sort both halves. In order to divide around the median, we map each element less than the median to 0 and map each element greater than or equal to the median to 1. Thus, the cost to sort a permutation is:

$$P(n) \leq 2P(n/2) + B(n).$$

Theorem 3: When the cost of reversals is $f(l) = l^\alpha, 0 \leq \alpha < 1$, for linear permutations, each 0/1 sequence can be sorted with cost $O(n + n^2 k^{\alpha-2})$, and each permutation can be sorted with cost $O(n \log n + n^2 k^{\alpha-2})$.

Proof: The cost to sort a 0/1 sequence of length not exceed k is $B'(l) = O(l)$. We have $B(n) = O(n/k)O(k) + O(n^2/k^2)O(k^\alpha)$. Hence, $B(n) = O(n + n^2 k^{\alpha-2})$. For the permutation, the cost is $P(n) = 2P(n/2) + O(n + n^2 k^{\alpha-2})$. Hence, $P(n) = O(n \log n + n^2 k^{\alpha-2})$.

Theorem 4: When the cost of reversals is $f(l) = l$, for linear permutations, each 0/1 sequence can be sorted with cost $O(n \log k + n^2/k)$, and each permutation can be sorted with cost $O(n \log n \log k + n^2/k)$.

Proof: The cost to sort a 0/1 sequence of length not exceed k is $B'(l) = O(l \log l)$. We have $B(n) = O(n/k)O(k \log k) + O(n^2/k^2)O(k)$. Hence, $B(n) = O(n \log k + n^2/k)$. For the permutation, the cost is $P(n) = 2P(n/2) + O(n \log k + n^2/k)$. Hence, $P(n) = O(n \log n \log k + n^2/k)$.

The above upper bounds for 0/1 sequences are tight. When k is small, *i.e.* $k^{2-\alpha} = O(n/\log n)$, we have $n \log n = O(n^2 k^{\alpha-2})$. For this case, the upper bounds for permutations are also tight. When $k = n$, our results are the same as that in Bender *et al.*

Next, we consider the case when $1 < \alpha < 2$.

Theorem 5: When the cost of reversals is $f(l) = l^\alpha, 1 \leq \alpha < 2$, for linear permutations, each 0/1 sequence can be sorted with cost $O(n^2 k^{\alpha-2})$, and each permutation can be sorted with cost $O(n \log n + n^2 k^{\alpha-2})$.

Proof: The cost to sort a 0/1 sequence of length not exceed k is $B'(l) = O(l^\alpha)$. We have $B(n) = O(n/k)O(k^\alpha) + O(n^2/k^2)O(k^\alpha)$. Hence, $B(n) = O(n^2k^{\alpha-2})$. For the permutation, the cost is $P(n) = 2P(n/2) + O(n^2k^{\alpha-2})$. Hence, $P(n) = O(n^2k^{\alpha-2} + n\log n)$.

In this case, the upper bounds are tight for both 0/1 sequences and permutations. When $k = n$, our results are the same as that in Bender *et al.*

Approximation algorithms for $1 \leq \alpha < 2$

Constant approximation algorithm for $S = 11\ldots100\ldots0$

We present 2-approximation algorithms for the sequence $S = 11\ldots100\ldots0$. Although they only solve the problem on one special sequence, these algorithms are important since they serve as the basis to sort general 0/1 sequences.

Let a and b be the length of the blocks of 1's and 0's in S, respectively. Without loss of generality, assume that $a \leq b$. When $b < k/2$, the sequence can be sorted optimally by a single reversal. The following theorem states that there are 2-approximation algorithms to sort S in the remaining cases.

Theorem 6: There is a 2-approximation algorithm for SBRLR (S, k, α).

Proof: We consider four cases based on the parity of k and the relationship between a and $k/2$.

Case 1. k is even and $a \geq k/2$. Let $k = 2k'$, $a = k'i+p$, and $b = k'j+q$ for $k', i, j \geq 1$ and $p, q < k'$. By the inversion argument similar to that used in deriving lower bounds, we can prove that $L = 4\,a\,b\,k^{\alpha-2}$ is a lower bound of the cost of sorting S. We prove that the following algorithm sorts S with a cost at most $2L$.

1. Divide the block of 1's into i blocks of length k' and 1 block of length p such that the block of length p is at the right end of the original block.

2. Divide the block of 0's into j blocks of length k' and 1 block of length q such that the block of length q is at the left end of the original block.

3. Move the block of p 1's to the right end of S.

4. Move the block of q 0's to the left end of S.

5. Sort the remaining blocks by bubble sort, and each swap is mimicked by a reversal of length k.

The cost of this algorithm is analyzed in the following. Step 3 has the cost $j(p+k')^\alpha + (p+q)^\alpha$, Step 4 has $i(q+k')^\alpha$, and Step 5 has $ij(2k')^\alpha$. Hence, the total cost of this algorithm is:

$$A = j(p+k')^\alpha + i(q+k')^\alpha + (p+q)^\alpha + ij(2k')^\alpha.$$

Let $F(p,q) = 8abk^{\alpha-2} - A$
$= 8(k'i+p)(k'j+q)k^{\alpha-2} - A,$

we have:

$$\begin{aligned}\frac{\partial F}{\partial q} &= 8(k'i+p)k^{\alpha-2} - i\alpha(q+k')^{\alpha-1} - \alpha(p+q)^{\alpha-1}\\ &\geq 8k'ik^{\alpha-2} - i\alpha k^{\alpha-1} - \alpha k^{\alpha-1}\\ &\geq 4ik^{\alpha-1} - 2ik^{\alpha-1} - 2ik^{\alpha-1}\\ &\geq 0\end{aligned}$$

Similarly, $\frac{\partial F}{\partial q} \geq 0$ and thus F is increasing on both p and q. Hence $F(p,q) \geq F(0,0) > 0$. So $A < 2L$.

Case 2. k is even and $a < k/2$. We augment the inversion argument with simple calculus. First, it can be shown that the function $f(x) = (x+a)^\alpha/ax$ is decreasing on $(0, \frac{a}{\alpha-1})$ and increasing on $(\frac{a}{\alpha-1}, +\infty)$. Thus, $f(x) \geq f(t)$ where $t = \min\{k-a, \frac{a}{\alpha-1}\}$ for all $x \in (0, k-a]$.

Now let $l_1, l_2, \ldots, l_p$ be the length of reversals in the optimal reversal series to sort S, such that $l_i \leq k$ for all $1 \leq i \leq p$, and n_i be the number of inversions removed by performing reversal l_i. If $l_i \leq 2a$, we have $n_i \leq l_i^2/4$ and hence

$$\frac{l_i^\alpha}{n_i} \geq l_i^{\alpha-2} \geq (2a)^{\alpha-2} = f(a) \geq f(t).$$

Otherwise we have $n_i \leq a(l_i - a)$ and hence $l_i^\alpha/n_i \geq f(l_i - a) \geq f(t)$. Hence $\frac{l_i^\alpha}{n_i} \geq f(t)$ for all $1 \leq i \leq p$.

The optimal cost to sort S is then:

$$\begin{aligned}l_1^\alpha + l_2^\alpha + \cdots + l_p^\alpha &= n_1\tfrac{l_1^\alpha}{n_1} + n_2\tfrac{l_2^\alpha}{n_2} + \cdots + n_p\tfrac{l_p^\alpha}{n_p}\\ &\geq (n_1 + n_2 + \cdots n_p)f(t)\\ &= abf(t)\\ &= \tfrac{b(a+t)^\alpha}{t}\end{aligned}$$

Let $L = \frac{b(a+t)^\alpha}{t}$, the following algorithm sorts S by a cost at most $2L$:

1. Since $(a+b) > 2k' \geq (t+a)$, $b = t\,i + r$ for some $i \geq 1$, we divide the block of 0's into i block of length t and one block of length r.

2. Move the block of 1's to the right end of S using i reversal of length $t+a$ and a reversal of length $a+r$.

The cost of this algorithm is:

$$\begin{aligned}i(t+a)^\alpha + (a+r)^\alpha &\leq (i+1)(t+a)^\alpha\\ &\leq 2i(t+a)^\alpha\\ &\leq 2\tfrac{b}{t}(t+a)^\alpha\\ &= 2L\end{aligned}$$

Case 3. k is odd and $a \geq k/2$. Similar to Case 1.

Case 4. k is odd and $a < k/2$. Similar to Case 2.

$(2\log_2|S|+1)$-*approximation algorithm for* 0/1 *sequences*

We utilize the algorithms in previous section to sort general 0/1 sequences. First we sort a sequence S where $|S| = 2^t$ by the following algorithm:

1. Divide S into two halves L and R.

2. Sort L and R recursively. After this step we have the sequence $00\ldots011\ldots100\ldots011\ldots1$.

3. Use the algorithms in the previous section to swap the first block of 1's with the second block of 0's.

The performance of this algorithm is given by Theorem 7. In proving this theorem, we intensively use the concept of *reduction of a reversal* on a subsequence S' of a sequence S. If r is a reversal on S, we define the reduction $r_{|S'}$ of r on S' to be the transformation that reverses the order of elements of S', which are affected by r. For example, assume that r affects the segment $s_1s_2s_3s_4s_5$, and s_1, s_3, s_4 are the elements of S', then $r_{|S'}$ reverses the order of s_1, s_3, s_4 in S', that is, it puts s_1 to the position of s_4 and vice versa. It is readily verified that $r_{|S'}$ is a reversal on S', that is, it reverses the order of consecutive elements of S'. Furthermore, if $R = r^1, r^2, \ldots, r^k$ sorts S, then $R_{|S'} = r^1_{|S'}, r^2_{|S'}, \ldots, r^k_{|S'}$ sorts S'.

Proposition 1: Let OPT_S, OPT_L, OPT_R be the optimal cost to sort S, L, R, respectively, we have $OPT_S \geq OPT_L + OPT_R$.

Proof: Let R be a reversal series that sorts S. For a reversals r, we have $|r_{|L}| + |r_{|R}| = |r|$, where $|x|$ denotes the length of reversal x. Since $1 \leq \alpha \leq 2$, we have $|r_{|L}|^\alpha + |r_{|R}|^\alpha \leq |r|^\alpha$. Hence, the total cost to sort L and R is less than that to sort R. This completes the proof.

Proposition 2: Let S be a 0/1 sequence and k be a position. If there are i 1's on the left of k and j 0's on the right of k, then the cost of sorting S is bigger than that of sorting $T = 1^i0^j$.

Proof: We map the t^{th} 1 on the left of k in S to the t^{th} 1 of T, and the t^{th} 0 on the right of k in S to the t^{th} 0 of T. Let T' be the subsequence of S whose elements are mapped to those of T. The reduction of any reversal sequence sorting S on T' sorts T' and hence sorts T. Hence the cost of sorting T must not exceed the cost of sorting S.

Theorem 7: The algorithm above sorts a 0/1 sequence S with the cost of $2OPT_S\log_2|S|$ when $|S| = 2^t$ for some integer t.

Proof: Let $C(S), C(L), C(R)$ be the cost of sorting S, L, and R by the algorithm, and D be the cost of Step 3. Proposition 2 shows that $D \leq 2OPT_S$. Hence, we have the following recurrence:

$$C(S) \leq C(L) + C(R) + 2OPT_S.$$

By an induction on $|S|$ using Proposition 1, we can verify that $C(S) = 2OPT_S\log_2|S|$.

To sort a general sequence S, let $2^k \leq |S| < 2^{k+1}$, we insert $(2^{k+1} - |S|)$ 0's to the left of S to get a sequence S' of length 2^{k+1}. It is obvious that the cost to sort S is at most the cost to sort S', and any algorithm that sorts S also sorts S' with the same cost. Hence $OPT_S = OPT_{S'}$, and

$$C(S) \leq C(S') \leq 2\log_2|S'|OPT_{S'} \leq (2\log_2|S|+1)OPT_S.$$

Hence, there is a $(2\log_2|S|+1)$-approximation algorithm to sort general 0/1 sequences.

$\big(2(\log_2 n)^2+\log_2 n\big)$-*approximation algorithm for linear permutations*

Again, we first give the algorithm to sort the sequence S where $|S| = 2^k$ for some integer k:

1. Find the median of the permutation.

2. Divide the permutation into halves with the right half containing all the elements bigger than the median, and the left half containing all the elements smaller than the median. We can do this by considering the elements bigger than the median 1's and the elements smaller than the element 0's and invoke the algorithm in the previous section.

3. Sort the two halves recursively.

The performance of this algorithm is given in the following theorem:

Theorem 8: Let OPT_S be the optimal cost of sorting a permutation S. The above algorithm sorts S in $2(\log_2|S|)^2OPT_S$.

Proof: Since the cost of Step 2 is at most $2(\log_2|S|)$, the cost of this algorithm satisfies the recurrence:

$$C(|S|) \leq 2C(|S|/2) + 2\log_2|S|OPT_S.$$

The theorem then follows a simple induction.

To sort a general permutation S where $2^t \leq |S| < 2^{t+1}$, we concatenate S with the sequences $|S|(|S|+1)\cdots2^{t+1}$ to get a permutation S' of length 2^{t+1}, and apply the above algorithm. By similar arguments as in the previous section, we obtain the $\big(2(\log_2 n)^2 + \log_2 n\big)$ approximation ratio.

A more exact analysis of the approximation ratio when $k = \Omega(|S|)$

Here, we prove that the above approximation algorithms for $1 \leq \alpha < 2$ and $k = \Omega(|S|)$ has the approximation ratios of $O(1)$ for 0/1 sequences and $O(\log n)$ for permutations. For a 0/1 sequence S of length n, let $w(i, S)$ denote the number of wrong-sided elements according to position i, that is, 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. The potential function $W(S)$ is defined as follows:

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

Using a result in Bender *et al*, we have $W(S) - W(S_L) - W(S_R) \geq cw(x, S)^{\alpha}$, where $c = 1/8\,[(3/4)^{\alpha-1} - (1/4)^{\alpha-1}]$ and x is the number of 0's in S.

Let $k \geq c_1 n$, we need to prove that $C(S) \leq C(S_L) + C(S_R) + Tw(x, S)^{\alpha}$ for some constant T. In other words, this is equivalent to proving that the cost to sort the sequence $S' = 11\ldots100\ldots0$ whose length is $w(x, S)$ should not exceed $Tw(x, S)^{\alpha}$.

Let $k' = k/2$, suppose there are $p\,k'$ 1's and qk' 0's in S'. The cost to sort S' using the above approximation algorithm is $2p\,qk^{\alpha}$, whereas $w(x, S) = (p+q)k/2$. Besides, $p\,k' + qk' = n$, thus either $p\,k' \leq n/2$ or $qk' \leq n/2$. Assume $p\,k' \leq n/2$, hence $p \leq n/k \leq 1/c_1$. We have:

$$\begin{aligned} w(x, S)^{\alpha} &= [(p+q)k/2]^{\alpha} \\ &= c(p+q)^{\alpha}k^{\alpha}\tfrac{4}{(3/2)^{\alpha-1}-(1/2)^{\alpha-1}} \\ &\geq c(p+q)^{\alpha}k^{\alpha}\tfrac{4}{(3/2)^{2-1}-(1/2)^{2-1}} \\ &\quad (\text{since } 1 \leq \alpha < 2) \\ &\geq 4c(p+q)k^{\alpha} \\ &\geq 4cc_1 p\,qk^{\alpha} \end{aligned}$$

Hence, we can choose $T = 1/2\,cc_1$. By induction, we have $C(S) = O\big(W(S)\big)$, that is, the above algorithm is an $O(1)$-approximation algorithm for 0/1 sequences.

When S is a permutation, using the approximation algorithm for permutations, we have $C(S) = 2C(S/2) + O(OPT)$. An easy induction shows that $C(S) = O(OPT \log n)$, that is, the above algorithm is an $O(\log n)$-approximation algorithm for permutations.

Conclusion

We have presented tight or nearly tight lower and upper bounds for the problem of sorting by restricted-length-weighted reversals, and also the approximation algorithms for a wide range of α. These results can be extended in various directions.

One direction is to strengthen the approximation ratio algorithms or determine the hardness of the problem based on the value of k or/and the value of α. Furthermore, the approximation algorithm for the case where $\alpha < 1$ is still open.

We can also work with other length-weighted functions that are consistent with some meaningful distribution. Another extension is to consider the more general problem when f is a step function:

$$f(l) = \begin{cases} f_1(l) & \text{if } l \leq k_1 \\ f_2(l) & \text{if } k_1 < l \leq k_2 \\ \ldots \end{cases}$$

Finally, it is interesting and important to find a cost function that can be verified in practice.

Acknowledgements

The authors thank Drs. Louxin Zhang and Wing-Kin Sung for important suggestions on the content of the paper, and thank anonymous reviewers for their helpful comments.

References

1. Sessions, S.K. 1990. Chromosomes: molecular cytogenetics. In *Molecular Systematics* (eds. Hillis, D.M. and Moritz, C.), pp. 156-204. Sinauer Associates, Inc., Sunderland, USA.
2. O'Brien, S.J. (ed.) 1993. *Genetic Maps: Locus Maps of Complex Genomes.* Cold Spring Harbor Laboratory Press, New York, USA.
3. Palmer, J.D., *et al.* 1988. Evolutionary significance of inversions in legume chloroplast DNAs. *Curr. Genet.* 14: 65-74.
4. Dobzhansky, T. 1970. *Genetics of the Evolutionary Process.* Columbia University Press, New York, USA.
5. Sankoff, D., *et al.* 2004. The distribution of inversion lengths in bacteria. In *Proceedings of the RECOMB 2004 International Workshop on Comparative Genomics*, pp. 97-108, Bertinoro, Italy.

6. Kececioglu, J.D. and Sankoff, D. 1993. Exact and approximation algorithms for the inversion distance between two permutations. In *Proceedings of the* 4th *Annual Symposium on Combinatorial Pattern Matching*, pp. 87-105, Padova, Italy.
7. Hannenhalli, S. and Pevzner, P.A. 1995. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. In *Proceedings of the* 27th *Annual ACM Symposium on the Theory of Computing*, pp. 178-189, Las Vegas, USA.
8. Caprara, A. 1997. Sorting by reversals is difficult. In *Proceedings of the Fisrt Annual International Conference on Computational Biology*, pp. 75-83, Santa Fe, USA.
9. Bafna, V. and Pevzner, P.A. 1993. Genome rearrangements and sorting by reversals. In *Proceedings of the* 34th *Annual IEEE Symposium on Foudations of Computer Science*, pp. 148-157, Palo Alto, USA.
10. Berman, P. and Hannenhalli, S. 1996. Fast sorting by reversals. In *Proceedings of the* 7th *Annual Symposium on Combinatorial Pattern Matching*, pp. 168-185, Laguna Beach, USA.
11. Kaplan, H., *et al.* 1997. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the* 8th *Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 344-351, New Orleans, USA.
12. Bader, D.A., *et al.* 2001. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. Comput. Biol.* 8: 483-491.
13. Kaplan, H. and Verbin, E. 2003. Efficient data structures and a new randomized approach for sorting signed permutations by reversals. In *Proceedings of the* 14th *Annual Symposium on Combinatorial Pattern Matching*, pp. 170-185, Morelia, Mexico.
14. Tannier, E. and Sagot, M.F. 2004. Sorting by reversals in subquadratic time. In *Proceedings of the* 15th *Annual Symposium on Combinatorial Pattern Matching*, pp. 1-13, Istanbul, Turkey.
15. Chen, T. and Skiena, S.S. 1996. Sorting with fixed length reversals. *Discrete Appl. Math.* 71: 269-295.
16. Pinter, R.Y. and Skiena, S. 2002. Genomic sorting with length-weighted reversals. *Genome Inform. Ser. Workshop Genome Inform.* 13: 103-111.
17. Bender, M.A., *et al.* 2004. Improved bounds on sorting with length-weighted reversals. In *Proceedings of the* 15th *Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 919-928, New Orleans, USA.
18. Swidan, F., *et al.* 2004. Sorting by length-weighted reversals: dealing with signs and circularity. In *Proceedings of the* 15th *Annual Symposium on Combinatorial Pattern Matching*, pp. 32-46, Istanbul, Turkey.