A 1.5-Approximation Algorithm for Sorting by Transpositions and Transreversals

Tzvika Hartman[1] and Roded Sharan[2]

[1] Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science
Rehovot 76100, Israel
tzvi.hartman@weizmann.ac.il
[2] International Computer Science Institute, 1947 Center St., Berkeley, CA 94704
roded@icsi.berkeley.edu

Abstract. One of the most promising ways to determine evolutionary distance between two organisms is to compare the order of appearance of orthologous genes in their genomes. The resulting genome rearrangement problem calls for finding a shortest sequence of rearrangement operations that sorts one genome into the other. In this paper we provide a 1.5-approximation algorithm for the problem of sorting by transpositions and transreversals, improving on a five years old 1.75 ratio for this problem. Our algorithm is also faster than current approaches and requires $O(n^{3/2}\sqrt{\log n})$ time for n genes.

1 Introduction

When trying to determine evolutionary distance between two organisms using genomic data, one wishes to reconstruct the sequence of evolutionary events that have occurred, transforming one genome into the other. One of the most promising ways to trace the evolutionary events is to compare the order of appearance of orthologous genes in two different genomes [1, 2]. This comparison, which relies on computing global rearrangement events, may provide more accurate and robust clues to the evolutionary process than the analysis of local mutations.

In a genome rearrangement problem, the two compared genomes are represented by permutations, where each element stands for a gene, and the goal is to find a shortest sequence of rearrangement operations that transforms (sorts) one permutation into the other. Previous work focused on the problem of sorting a permutation by reversal operations. This problem was shown to be NP-hard [3]. One of the most celebrated results in this area by Hannenhalli and Pevzner shows that for signed permutations (every element of the permutation has a sign, which represents the direction of the corresponding gene; a reversal reverses the order of the elements it operates on and flips their signs), the problem becomes polynomial [4]. The algorithm is based on representing a permutation using a breakpoint graph (we defer a formal definition to Section 2) which decomposes uniquely into disjoint cycles, and studying the effect of a reversal on its cycle decomposition. There has been less progress on the problem of sorting by transpositions. A transposition is a rearrangement operation in which a segment is cut out of the permutation and pasted in a different location. The complexity of sorting by transpositions is still open, although several 1.5-approximation algorithms are known for it [5–7].

I. Jonassen and J. Kim (Eds.): WABI 2004, LNBI 3240, pp. 50–61, 2004.

A transreversal is a biologically motivated operation that combines a transposition and a reversal: A segment is cut out of the permutation, reversed and pasted in another location. In particular, a reversal is also a transreversal. Transpositions and transreversals capture a large fraction of the genomic rearrangements in evolution. Gu et al. [8] gave a 2-approximation algorithm for sorting signed permutations by transpositions and transreversals. Lin and Xue [9] improved this ratio to 1.75 by considering a third rearrangement operation, called revrev, which reverses two contiguous segments. Both algorithms run in quadratic time.

In this paper we study the problem of sorting permutations by transpositions, transreversals and revrevs. The question of whether the 1.75 known ratio for this problem can be improved, has been open for five years. One of the main difficulties in tackling the complexity of this problem is the vast number of possible configurations that need to be considered when analyzing general linear permutations. We make four contributions toward greatly simplifying the problem. First, we show that the sorting problem is equivalent for linear and circular permutations (Section 2). Second, we reduce the general problem of sorting a circular permutation to that of sorting a permutation with a very simple structure: In its breakpoint graph representation all non-trivial cycles are of length 3 (Section 2). Third, we characterize cycle configurations in the breakpoint graph and show that it suffices to restrict attention to one type of configurations. Fourth, we develop and characterize a novel cycle representation, which allows us to use results on sorting by transpositions only, a well-studied problem, in further eliminating cycle configurations. These characterizations and simplifications are key to our main result: a 1.5-approximation algorithm for sorting by transpositions, transreversals and revrevs (Section 3). Furthermore, our algorithm can be implemented in time $O(n^{3/2}\sqrt{\log n})$, which improves on the quadratic running time of previous algorithms [8, 9]. For lack of space, some proofs are shortened or omitted.

2 Preliminaries

A *signed permutation* $\pi = [\pi_1 \ \dots \ \pi_n]$ on $n(\pi) \equiv n$ elements is a permutation in which each element is labelled by a sign of plus or minus. A *segment* of π is a consecutive sequence of elements $\pi_i, \dots, \pi_k$ ($k \geq i$). We focus on four rearrangement operations. A *reversal* ρ is an operation that reverses the order of the elements in a segment and flips their signs. If the segment is $\pi_i, \dots, \pi_{j-1}$ then $\rho \cdot \pi = [\pi_1, \ \dots \ , \pi_{i-1}, -\pi_{j-1}, \ \dots \ , -\pi_i, \pi_j, \ \dots \ , \pi_n]$. Two segments $\pi_i, \dots, \pi_k$ and $\pi_j, \dots, \pi_l$ are *contiguous* if $j = k + 1$ or $i = l + 1$. A *transposition* τ exchanges two contiguous (disjoint) segments. If the segments are $A = \pi_i, \dots, \pi_{j-1}$ and $B = \pi_j, \dots, \pi_{k-1}$ then $\tau \cdot \pi = [\pi_1, \ \dots \ , \pi_{i-1} \ , \pi_j, \ \dots \ , \pi_{k-1} \ , \pi_i, \ \dots \ , \pi_{j-1} \ , \pi_k, \ \dots \ , \pi_n]$ (note that the end segments can be empty if $i = 1$ or $k = n + 1$). A *transreversal* $\tau\rho_{A,B}$ is a transposition that exchanges segments A and B and also reverses $A.$, i.e., $\tau\rho_{A,B} \cdot \pi = [\pi_1, \ \dots \ , \pi_{i-1} \ , \pi_j, \ \dots \ , \pi_{k-1} \ , -\pi_{j-1}, \ \dots \ , -\pi_i \ , \pi_k, \ \dots \ , \pi_n]$, and $\tau\rho_{B,A} \cdot \pi = [\pi_1, \ \dots \ , \pi_{i-1} \ , \dots \ , -\pi_{k-1} \ , \dots \ , -\pi_j, \pi_i, \ \dots \ , \pi_{j-1} \ , \pi_k, \ \dots \ , \pi_n]$. A *revrev* operation reverses each of the two segments (without transposing them). Thus, $\rho\rho \cdot \pi = [\pi_1, \ \dots \ , \pi_{i-1}, \ -\pi_{j-1}, \ \dots \ , -\pi_i, \ -\pi_{k-1}, \ \dots \ , -\pi_j, \ \pi_k, \ \dots \ , \pi_n]$.

The problem of finding a shortest sequence of transposition, transreversal and revrev operations that transforms a permutation into the identity permutation is called *sorting*

by transpositions and transreversals[1]. The *distance* of a permutation π, denoted by $d(\pi)$, is the length of the shortest sorting sequence.

Linear vs. Circular Permutations. Key to our approximation algorithm is a reduction from the problem of sorting linear permutations to that of sorting circular permutations (indices are cyclic), on which the analysis is simpler. An operation is said to *operate* on the segments that are affected by it and on the elements in those segments. We say that two operations μ and μ' are *equivalent* if they have the same effect, i.e., $\mu \cdot \pi = \mu' \cdot \pi$ for all π. The following lemma is the basis for the reduction, and is used to prove the subsequent theorem on the equivalence of the sorting problem for linear and circular permutations, similarly to [7].

Lemma 1 *Let x be an element of a circular permutation π, and let μ be an operation that operates on x. Then there exists an equivalent operation μ' that does not operate on x.*

Proof. For reversals, this result was proven by Meidanis et al. [10] and for transpositions by Hartman [7]. For transreversals and revrevs, the claim relies on the observation that a chromosome is equivalent to its reflection, i.e., the reversed sequence of elements with their signs flipped [10] (see the upper part of Figure 1). Consider a permutation with three segments: A, B and C. W.l.o.g. $x \in A$. Then a transreversal that operates on segments A and B and reverses B (resp. A) is equivalent to a revrev that operates on A and C (B and C), since the result is a reflection of the permutation (as illustrated in Figure 1). Similarly, a revrev that operates on A and B (or C) is equivalent to a transreversal that operates on B and C. ■

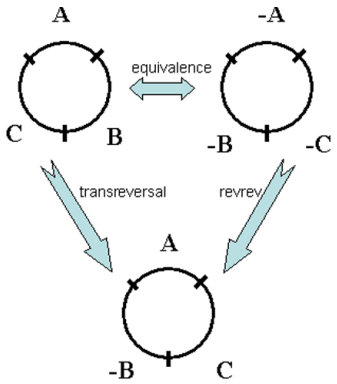

Fig. 1. The equivalence of operations on circular permutations.

[1] We do not include revrevs in the problem name, as we provide in the next section a reduction of the problem that allows us to mimic revrevs using transreversals.

Theorem 2 *The problem of sorting linear permutations by transpositions and transreversals is linearly equivalent to the problem of sorting circular permutations by transpositions and transreversals.*

Proof. We show only one direction. Given a linear n-permutation, circularize it by adding an additional element $\pi_{n+1} = x$ and closing the circle. Denote the new circular permutation by π^c. By Lemma 1, any operation on π^c can be mimicked by an operation that does not involve the segment that includes x. Hence, there is an optimal sequence of operations that sorts π^c such that none of them operates on segments that include x. The same sequence can be viewed as a sequence of operations on the linear permutation π, by ignoring x. This implies that $d(\pi) \leq d(\pi^c)$. On the other hand, any sequence of operations on π is also a sequence of operations on π^c, so $d(\pi^c) \leq d(\pi)$. Hence, $d(\pi) = d(\pi^c)$. Moreover, an optimal sequence for π^c implies an optimal sequence for π. ■

We observe that for circular permutations revrevs and transreversals are equivalent operations. Thus, for circular permutations we can restrict attention to transpositions and transreversals, which are more biologically motivated operations. Moreover, combined with Theorem 2, this observation implies that one can reduce the problem of sorting a linear permutation by transpositions, transreversals and revrevs to that of sorting a circular permutation by transpositions and transreversals only.

The Breakpoint Graph. We follow the construction of Bafna and Pevzner for representing signed permutations [11]. First, a permutation π on n elements is transformed into a permutation $f(\pi) = \pi' = (\pi'_1 \ \ldots \ \pi'_{2n})$ on $2n$ elements, by replacing each positive element i by two elements $2i-1, 2i$ (in this order), and each negative element by $2i, 2i-1$. On the extended permutation $f(\pi)$, only operations that cut before odd positions are allowed. This ensures that every operation on $f(\pi)$ can be mimicked by an operation on π. The *breakpoint graph* $G(\pi)$ is an edge-colored graph on $2n$ vertices $\{1, 2, \ldots, 2n\}$. For every $1 \leq i \leq n$, π'_{2i} is joined to π'_{2i+1} by a black edge (denoted by b_i), and $2i$ is joined to $2i+1$ by a gray edge. Here and in the rest of the paper we identify, in both indices and elements, $2n+1$ and 1.

It is convenient to draw the breakpoint graph on a circle, such that black edges are on the circumference and gray edges are chords (see Figure 2). Since the degree of each vertex is exactly 2, the graph uniquely decomposes into cycles. A *k-cycle* is a cycle with k black edges, and it is *odd* if k is odd. k is called the *length* of the cycle. The number of odd cycles in $G(\pi)$ is denoted by $c_{odd}(\pi)$. Gu et al. [8] have shown that for all linear permutations π and operations μ, it holds that $c_{odd}(\mu \cdot \pi) \leq c_{odd}(\pi) + 2$. Their result holds also for circular permutations and can be used to prove the following lower bound on $d(\pi)$:

Theorem 3 ([8]) *For all permutations π, $d(\pi) \geq (n(\pi) - c_{odd}(\pi))/2$.*

Transformation into 3-Permutations. Our goal in this section is to transform the input permutation into a permutation with simple structure, to which we can apply our algorithm and mimic its steps on the original permutation. A permutation is called *simple* if its breakpoint graph contains only k-cycles, where $k \leq 3$. It is called a *3-permutation*

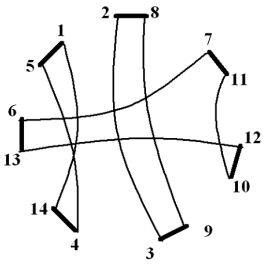

Fig. 2. The circular breakpoint graph of the permutation $\pi = (1\ -4\ 6\ -5\ 2\ -7\ -3)$, for which $f(\pi) = (1\ 2\ 8\ 7\ 11\ 12\ 10\ 9\ 3\ 4\ 14\ 13\ 6\ 5)$. Black edges are represented as thick lines on the circumference, and gray edges are chords.

if it contains only 1-cycles and 3-cycles. A transformation from π to $\hat{\pi}$ is called *safe* if $n(\pi) - c_{odd}(\pi) = n(\hat{\pi}) - c_{odd}(\hat{\pi})$, i.e., if it maintains the lower bound of Theorem 3. Next, we show how to transform an arbitrary permutation into a 3-permutation using safe transformations (that is, maintaining the lower bound, but not the exact distance). Our starting point is the standard safe transformation into simple permutations (cf. [7]). Hence, it suffices to show how to convert 2-cycles into 3-cycles using safe transformations.

Let C be a 2-cycle and let $b = (\pi'_{2i}, \pi'_{2i+1})$ be one of its black edges. A (C, b)-*padding* extends the original permutation π by adding a new element $\pi_i + 1$, and renaming all elements $j > \pi_i + 1$ by $j + 1$ (the renaming is done on the absolute values of the elements and then their signs are reintroduced, e.g., -3 is renamed to -4). The new element $\pi_i + 1$ has the same sign as π_i, and is located after (resp. before) π_i if it is positive (negative). Finally, the sign of π_i is flipped. The effect on the breakpoint graph is that C is transformed into a 3-cycle (see Figure 3 for an example). Overall, the permutation after the padding has an additional element and one more odd cycle.

Lemma 4 *Every simple permutation π can be transformed into a 3-permutation $\hat{\pi}$ by safe paddings. Moreover, every sorting of $\hat{\pi}$ mimics a sorting of π with the same number of operations.*

Proof. Let π be a simple permutation that contains a 2-cycle C and let $b \in C$. Let $\overline{\pi}$ be the permutation obtained by applying a (C, b)-padding on π. Clearly, $n(\overline{\pi}) = n(\pi) + 1$, and $c_{odd}(\overline{\pi}) = c_{odd}(\pi) + 1$, so the padding is safe. This process can be repeated until a 3-permutation $\hat{\pi}$ is eventually obtained. Since $\hat{\pi}$ is obtained from π by padding new elements, every operation of $\hat{\pi}$ can be mimicked on π by ignoring the padded elements. ■

In the rest of the paper, we shall restrict attention to circular 3-permutations and often refer to the 3-cycles in our breakpoint graph simply as cycles. In Section 3 we show how to sort a 3-permutation using at most $1.5l$ operations, where l is the lower bound of Theorem 3. By Theorem 2 and Lemma 4 this implies a 1.5-approximation algorithm for sorting arbitrary circular and linear permutations.

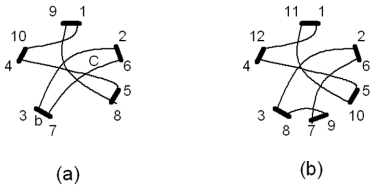

Fig. 3. (a) The breakpoint graph of the permutation $\pi = (1, -3, -4, 2, -5)$. (b) The graph of $(1, -3, -5, 4, 2, -6)$, which is obtained by a (C, b)-padding.

Cycle Types. An operation that cuts some black edges is said to *act on* these edges. It is called a k*-operation* if it increases the number of odd cycles by k. An odd cycle is called *oriented* if there is a 2-operation that acts on three of its black edges; otherwise, it is *unoriented*. A *configuration* of cycles is a subgraph of the breakpoint graph that contains one or more cycles. There are four possible configurations of single 3-cycles, which are shown in Figure 4(a-d). It is easy to verify that cycles a and b are unoriented, whereas c and d are oriented. A black edge is called *twisted* if its two adjacent gray edges cross each other as chords in the circular breakpoint graph. A cycle is k*-twisted* if k of its black edges are twisted. For example, in Figure 4 cycle a is 0-twisted and c is 2-twisted.

Observation 5 *A 3-cycle is oriented iff it is 2- or 3-twisted.*

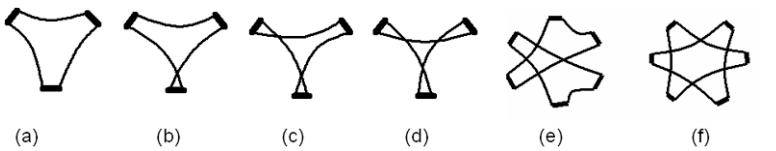

Fig. 4. Configurations of 3-cycles. (a-b) Unoriented 3-cycles. (c-d) Oriented 3-cycles. (e) A pair of intersecting 3-cycles. (f) A pair of interleaving 3-cycles.

Let $b = (i_1, i_2)$ and $b' = (j_1, j_2)$ be two black edges in the breakpoint graph such that i_1,i_2,j_1 and j_2 occur in this order along the circle. Then b_1 and b_2 *induce* two disjoint *arcs* on the circle, one between i_2 and j_1 and the other between j_2 and i_1. Two pairs of black edges are called *intersecting* if they alternate in their order of occurrence along the circle. A pair of black edges intersects with cycle C, if it intersects with a pair of black edges that belong to C. Cycles C and D intersect if there is a pair of black edges in C that intersect with D (see Figure 4e). Two cycles are *interleaving* if their black edges alternate in their order of occurrence along the circle (see Figure 4f). A *1-twisted pair* is a pair of 1-twisted cycles, whose twists are consecutive on the circle in a configuration that consists of these two cycles only. A pair of black edges is said to be *coupled* if they are connected by a gray edge and when reading the edges along the circle they are read in the same direction. (For example, the top edges in Figure 4b are coupled, and so are all pairs of edges in Figure 4a).

The following lemma will be useful in the sequel:

Lemma 6 ([8]) *Let (b_1, b_2) be a pair of coupled black edges. Then there exists a cycle C that intersects with (b_1, b_2).*

3 The Algorithm

A $(0, 2, 2)$-*sequence* is a sequence of three operations, of which the first is a 0-operation and the next two are 2-operations. Since a 2-operation is the best possible in one step, a series of $(0, 2, 2)$-sequences guarantees a 1.5 approximation ratio. A 1-twisted cycle is called *closed* (w.r.t. a configuration) if its two coupled edges intersect with some other cycle in the configuration. A configuration is *closed* if at least one of its 1-twisted cycles is closed; otherwise it is called *open*. In the following we shall consider only closed configurations, since an open configuration implies the existence of a closed one (by Lemma 6). For each possible closed configuration we shall prove the existence of a $(0, 2, 2)$-sequence of operations. First, we deal with interleaving cycle pairs.

Lemma 7 *Let π be a permutation that contains two unoriented, interleaving cycles C and D that do not form a 1-twisted pair. Then π admits a $(0, 2, 2)$-sequence.*

Proof. If both cycles are 0-twisted then a $(0, 2, 2)$-sequence of transpositions is given in [7]. Suppose that C is 0-twisted and D 1-twisted (resp. both are 1-twisted and their twists are not consecutive on the circle). Let a, b and c be the three arcs that are induced by the black edges of C, and let a be the arc that contains the twist of D. First apply a 0-transposition that acts on the black edges of C. This makes D 2-twisted, so it is possible to eliminate it using a 2-transreversal. The latter operation makes C 2-twisted (resp. 3-twisted). A 2-transreversal (2-transposition) on C completes the $(0, 2, 2)$-sequence. ■

In order to deal with intersecting cycles we borrow some of the theory developed in [7] for unsigned permutations. As we show below, some of this theory carries also to our case with some modifications. A useful tool that we will require is the signed canonical labelling[2] of a cycle which we present next.

For a given cycle (of any length), consider the labelling obtained by picking an arbitrary black edge of a cycle, labelling it 1, and labelling the rest of the cycle's edges according to their order of occurrence along the circle. The *signed canonical labelling* of a cycle is the signed permutation obtained by starting with the edge labelled 1 and reading the labels in the order they appear along the cycle, where the signs stand for the direction in which the edge is read: An edge that is visited in the same direction as the edge labelled 1 is positive, and otherwise it is negative (see, e.g., Figure 5). This definition captures the notion of twists in 3-cycles; indeed, a 0-twisted 3-cycle has labelling $(1, 2, 3)$, 3-twisted has labelling $(1, 3, 2)$, etc. Note that a cycle may have more than one possible canonical labelling. A canonical labelling of a 5-cycle is called *oriented* if it starts with $1, b, a$ or $1, -a, -b$ or $1, -b, a$ or $1, b, -a$, where $1 < a < b$. The motivation for this definition comes from the following observation:

Observation 8 *A 5-cycle is oriented iff it has an oriented canonical labelling.*

[2] A generalization of the notion of canonical labelling in [6].

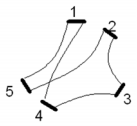

Fig. 5. A 5-cycle with signed canonical labelling $(1, -4 - 3 - 2, 5)$.

Lemma 9 *Let C be a 5-cycle that admits a 2-transposition. Then any 5-cycle with the same canonical labelling up to a reversal of one element is also oriented. If in addition the canonical labelling of C is $(1, 4, 2, 3, 5)$ then a 5-cycle with the same canonical labelling up to a reversal of two consecutive elements is also oriented.*

The above lemma is the basis for handling the case of two intersecting 0-twisted cycles, which we present next:

Lemma 10 *Let π be a permutation that contains a closed configuration in which there are two intersecting 0-twisted cycles C and D. Then π admits a $(0, 2, 2)$-sequence.*

Proof. Since C and D are intersecting, C has a pair of coupled edges that do not intersect with D. By Lemma 6 there exists a cycle E that intersects with this pair of edges. The case in which E is 0-twisted was treated in [7]. If E is 1-twisted there are two cases to consider:

1. D and E are non-intersecting. Our starting point is the $(0, 2, 2)$-sequences for configurations of three 0-twisted cycles given in Figure 6, where two of the cycles are non-intersecting, and the third one intersects both. In our case, one of the non-intersecting cycles corresponds to E and is 1-twisted. Depending on the location of the twist in E, it is always possible to apply the first two transpositions shown in Figure 6 to the closed configuration. (The first transposition is applied to the edges shown in the figure, if all are non-twisted, or to a symmetric set of edges). By Lemma 9, the resulting 5-cycle is oriented, which completes the $(0, 2, 2)$-sequence.

2. D and E are intersecting. Consider the $(0, 2, 2)$-sequences for three mutually intersecting 0-twisted cycles given in Figure 7. In our case E is 1-twisted. If all three edges d, e_1 and e_2 that are cut by the first transposition are non-twisted, then we apply the first two transpositions as in Figure 7. By Lemma 9, the resulting 5-cycle, F, is oriented. The same holds for any set of symmetric edges that are non-twisted. The only closed configurations in which no such symmetric set is possible is when some arc induced by a pair of black edges of C contains a single twist. There are three such configurations, for which a $(0, 2, 2)$-sequence is described in Figure 8. ■

Next, we deal with closed configurations that include two intersecting, 1-twisted cycles. We need the following observation:

Observation 11 *Let π be a permutation that contains a 2-twisted cycle C and a 1-twisted cycle D, such that C and D are intersecting and there is a single non-twist of D in the arc induced by the two twists of C. Then π admits two consecutive 2-operations.*

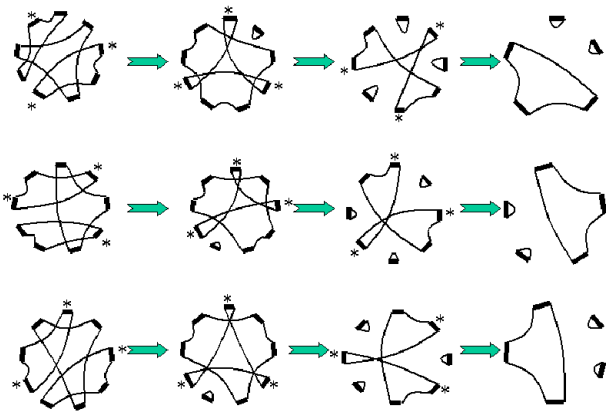

Fig. 6. $(0, 2, 2)$-sequences for three 0-twisted cycles, where two of the cycles are non-intersecting, and a third one intersects both (taken from [7]). At each step the transposition acts on the three black edges marked by a star. For simplicity, every 1-cycle is shown only when it is formed and not in subsequent graphs.

Fig. 7. Three mutually intersecting 0-twisted cycles (taken from [7]). A dashed line represents a path.

Proof. Applying a 2-transreversal on C eliminates it, while making D 2-twisted.■

Lemma 12 *Let π be a permutation that contains a closed configuration with two intersecting, 1-twisted cycles. Then π admits a $(0, 2, 2)$-sequence.*

Proof. There are six possible cases, shown in Figure 9. For cases (a-d), we first apply a 0-reversal that acts on the black edges that are marked by a star. This makes the other cycle 2-twisted, and two additional 2-operations follow from Observation 11. For cases (e-f) we observe that by Lemma 6 there is another cycle that has a black edge in the arc denoted x, and a black edge in one of the other 5 arcs. We apply a 0-reversal that acts on these two edges. If the resulting configuration contains two 2-twisted cycles then the permutation can be shown to admit two 2-operations. Otherwise, two 2-operations follow from Observation 11. ■

The following lemma deals with a closed configuration which involves two intersecting cycles, one of which is 0-twisted and the other 1-twisted. The subsequent lemma deals with 1-twisted pairs of interleaving cycles. The proofs of both lemmas can be found at `http://www.icsi.berkeley.edu/~roded/transrev.pdf`.

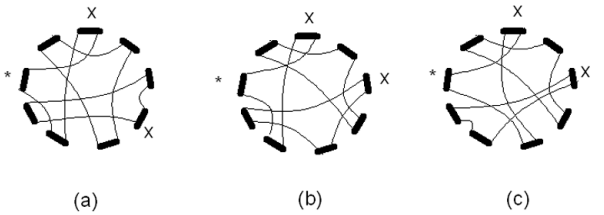

Fig. 8. $(0, 2, 2)$-sequences for some cycle configurations that contain two intersecting 0-twisted cycles. First we apply a 0-transreversal on the three marked edges, such that the segment between the two x's is reversed, resulting in an oriented 3-cycle and a 5-cycle. Next, we eliminate the 3-cycle and are left with an oriented 5-cycle, which allows us to complete the $(0, 2, 2)$-sequence.

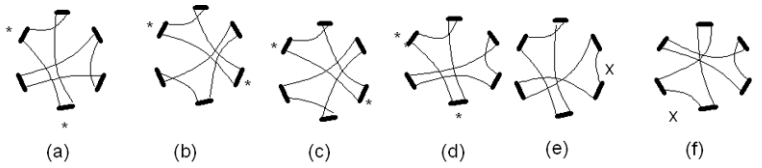

Fig. 9. Closed configurations of two intersecting 1-twisted 3-cycles.

Lemma 13 *Let π be a permutation that contains a 0-twisted cycle, which intersects with the coupled edges of a 1-twisted cycle. Then π admits a $(0, 2, 2)$-sequence.*

Lemma 14 *Let π be a permutation that contains $k \geq 2$ mutually interleaving 1-twisted cycles, such that all their twists are consecutive on the circle and k is maximal with this property. Then π admits a $(0, 2, 2)$-sequence.*

We are now ready to state our main result:

Theorem 15 *There is a 1.5-approximation algorithm for sorting by transpositions and transreversals, which runs in $O(n^{3/2}\sqrt{\log n})$ time.*

Proof. Our algorithm is described in Figure 10. The sequence of operations generated by the algorithm contains only 2-operations and $(0, 2, 2)$-sequences of operations. Therefore, every sequence of three operations increases the number of odd cycles by at least 4 out of 6 possible in 3 steps (as implied from the lower bound of Theorem 3). Hence, the approximation ratio is 1.5.

We sketch the proof of the running time. Steps 1 and 3 can be done in linear time. The number of iterations in Step 2) is linear. Note that identifying pairs of interleaving and intersecting cycles can be done by applying the following query a constant number of times: Given a gray edge find an arbitrary gray edge that intersects it. Thus, the most time-consuming tasks in each iteration are the application of operations to the permutation, and the above query. These two tasks can be performed in $O(\sqrt{n \log n})$ time using the data structure of Kaplan and Verbin [12], as we show next.

Algorithm *Sort* (π)

1. Transform π into a 3-permutation $\hat{\pi}$ (Lemma 4).
2. While $G(\hat{\pi})$ contains a 3-cycle C do:
   (a) If C is oriented, apply a 2-operation.
   (b) Otherwise, find a cycle D that intersects with a coupled pair of C.
   (c) If C and D interleave, apply a $(0, 2, 2)$-sequence (Lemmas 7, 14).
   (d) Else if C or D are 1-twisted, apply a $(0, 2, 2)$-sequence (Lemmas 12, 13).
   (e) Otherwise, apply a $(0, 2, 2)$-sequence (Lemma 10).
3. Mimic the sorting of π using the sorting of $\hat{\pi}$ (Lemma 4).

Fig. 10. Algorithm Sort. After Step 2(a) we assume that all cycles involved in the configuration of C are unoriented. Obviously, if an oriented cycle is involved, then a 2-operation can be applied on it.

For simplicity, we describe the data structure for linear permutations. Consider the breakpoint graph of a linear permutation with n elements. Each gray edge is represented by a pair of vertices, called *mates*. The permutation is partitioned into $\Theta(\sqrt{\frac{n}{\log n}})$ blocks of $\Theta(\sqrt{n \log n})$ vertices each. A splay tree [13] is attached to each block, in which the vertices of the block are maintained according to the order of their mates in the permutation. In [12] it is shown that a reversal can be applied in time $O(\sqrt{n \log n})$. Hence, transpositions (resp. transreversals) can be easily implemented using two (three) reversals in the same time bound. As for queries, we denote the pair of vertices by v_1 and v_2. We may assume that v_1 and v_2 are first elements in their blocks [12], so we need not consider parts of blocks. We scan all blocks that are between v_1 and v_2, and ask if the mate of the leftmost vertex in the block appears in the permutation before v_1, or the mate of the rightmost vertex element appears after v_2. If there is a block that satisfies the condition, then we found an intersecting pair; otherwise, there is no such pair. ■

Acknowledgments

We would like to thank Ron Shamir for many invaluable discussions, and Elad Verbin for discussions on the data structure. TH was supported in part by an Israel Science foundation grant 309/02 (PI: Ron Shamir) and by ISF grant IS265/02. RS was supported in part by NSF ITR grant CCR-0121555.

References

1. Palmer, J.D., Herbon, L.A.: Tricircular mitochondrial genomes of Brassica and Raphanus: reversal of repeat configurations by inversion. Nucleic Acids Research **14** (1986) 9755–9764
2. Hoot, S.B., Palmer, J.D.: Structural rearrangements, including parallel inversions, within the chloroplast genome of Anemone and related genera. J. Molecular Evooution **38** (1994) 274–281
3. Caprara, A.: Sorting permutations by reversals and Eulerian cycle decompositions. SIAM Journal on Discrete Mathematics **12** (1999) 91–110

4. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. Journal of the ACM **46** (1999) 1–27
5. Bafna, V., Pevzner, P.A.: Sorting by transpositions. SIAM Journal on Discrete Mathematics **11** (1998) 224–240
6. Christie, D.A.: Genome Rearrangement Problems. PhD thesis, University of Glasgow (1999)
7. Hartman, T.: A simpler 1.5-approximation algorithm for sorting by transpositions. In: Proc. 14th Annual Symposium on Combinaotrial Pattern Matching (CPM '03), Springer (2003) 156–169
8. Gu, Q.P., Peng, S., Sudborough, H.: A 2-approximation algorithm for genome rearrangements by reversals and transpositions. Theoretical Computer Science **210(2)** (1999) 327–339
9. Lin, G.H., Xue, G.: Signed genome rearrangements by reversals and transpositions: Models and approximations. In: Proc. COCOON '99, Lecture Notes in Computer Science. Volume 1627., Berlin Heidelberg, Springer-Verlag (1999) 71–80
10. Meidanis, J., Walter, M.E., Dias, Z.: Reversal distance of signed circular chromosomes. manuscript (2000)
11. Bafna, V., Pevzner, P.A.: Genome rearragements and sorting by reversals. SIAM Journal on Computing **25** (1996) 272–289
12. Kaplan, H., Verbin, E.: Effficient data structures and a new randomized approach for sorting signed permutations by reversals. In: Proc. 14th Annual Symposium on Combinaotrial Pattern Matching (CPM '03), Springer (2003) 170–185
13. Sleator, D.D., Tarjan, R.E.: Self-adjusting binary search trees. J. Assoc. Comput. Mach. **32** (1985) 652–686