Available online at www.sciencedirect.com
SCIENCE DIRECT®

Journal of Computer and System Sciences 70 (2005) 300–320

JOURNAL OF COMPUTER AND SYSTEM SCIENCES
www.elsevier.com/locate/jcss

# A 1.5-approximation algorithm for sorting by transpositions and transreversals[☆]

Tzvika Hartman[a], Roded Sharan[b,*]

[a]*Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel*
[b]*School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel*

Received 15 July 2004; received in revised form 14 November 2004

**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-year-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.
© 2005 Elsevier Inc. All rights reserved.

*Keywords:* Genome rearrangements; Sorting by transpositions; Sorting by reversals

## 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 transformed 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 [14,10]. This comparison, which relies on computing global

[☆] A preliminary version of this paper appeared in [9].
[*] Corresponding author.
*E-mail addresses:* tzvi.hartman@weizmann.ac.il (T. Hartman), roded@cs.tau.ac.il (R. Sharan).

0022-0000/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcss.2004.12.006

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 by Caprara [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 in a segment and flips their signs), the problem becomes polynomial [7]. 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 sorting problems with respect to other operations, such as transpositions and transreversals.

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 [2,4,8], and, very recently, a 1.375-approximation algorithm was given for this problem [5].

A transreversal is another 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. Gu et al. [6] gave a 2-approximation algorithm for sorting signed permutations by transpositions and transreversals. Lin and Xue [12] improved this ratio to 1.75 by considering a third rearrangement operation, called revrev, which reverses two contiguous segments. Walter et al. [16] considered the problem of sorting by transpositions and reversals (without transreversals). For the signed (resp., unsigned) case they provided a 2-approximation (resp., 3-approximation) algorithm. All the algorithms mentioned above 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 5 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. This reduction allows us to restrict attention to two operations only—transpositions and transreversals. 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. Third, we characterize cycle configurations in the breakpoint graph and show that it suffices to restrict attention to one type of configuration. Fourth, we develop and characterize a novel cycle representation, which allows us to use previous results on sorting by transpositions only in further eliminating cycle configurations. These characterizations and simplifications are key to our main result: a 1.5-approximation algorithm for sorting both linear and circular permutations by transpositions and transreversals. Furthermore, we exploit a data structure introduced in [11] to implement the algorithm in time $O(n^{3/2}\sqrt{\log n})$, thus improving on the quadratic running time of previous algorithms [6,12].

Our results borrow ideas from our earlier work on sorting by transpositions only [8]. Briefly, the latter paper presents a 1.5-approximation algorithm for sorting by transpositions, which is based on reducing the problem to that of handling circular permutations with simple structure. Here we extend this reduction to signed permutations under both transpositions and transreversals, and show that an even simpler structure can be obtained. We also develop novel characterizations of cycle configurations, allowing us to use some of the sorting techniques presented in [8]. These characterizations are complemented by a detailed case-analysis of the cycle configurations that may arise during the sorting process, which are more involved than in the unsigned case.

The paper is organized as follows: Background on rearrangement operations, permutations and their representation are given in Section 2, where we also describe the reduction to sorting simple circular permutations. The approximation algorithm is given in Section 3. Finally, the $O(n^{3/2}\sqrt{\log n})$ implementation of the algorithm is described in Section 4.

## 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 labeled by a sign of plus or minus. A *segment* of $\pi$ is a consecutive sequence of elements $\pi_i, \dots, \pi_k$ $(k \geqslant i)$. We focus on four rearrangement *operations*. A *reversal* $\rho$ 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* $\tau$ 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 $\pi$, denoted by $d(\pi)$, is the length of the shortest sorting sequence.

### 2.1. 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 $\mu$ and $\mu'$ are *equivalent* if they have the same effect, i.e., $\mu \cdot \pi = \mu' \cdot \pi$ for all $\pi$. 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 [8].

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

[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.

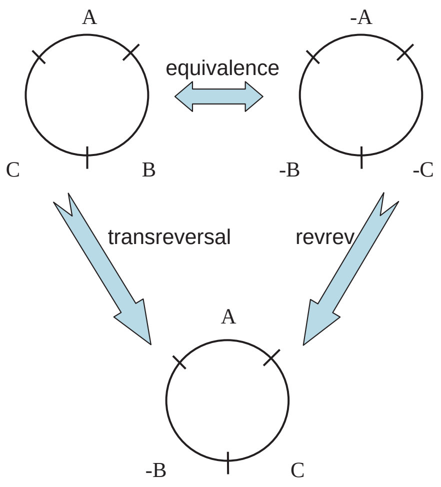

Fig. 1. The equivalence of operations on circular permutations.

**Proof.** For reversals, this result was proven by Meidanis et al. [13]; for transpositions this was shown by Hartman [8]. 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 [13] (see the upper part of Fig. 1). Consider a permutation with three segments: $A$, $B$ and $C$, where $x \in A$. 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 $C$ (resp., $B$ and $C$), since the result is a reflection of the permutation (as illustrated in Fig. 1). Similarly, a revrev that operates on $A$ and $B$ (or $C$) is equivalent to a transreversal that operates on $B$ and $C$. □

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

Conversely, starting with a circular permutation, we can linearize it by removing an arbitrary element, which plays the role of $n + 1$ above. Using similar arguments as in the first direction of the proof, we conclude that an optimal solution for the linear permutation translates to an optimal solution for the circular one. □

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 better

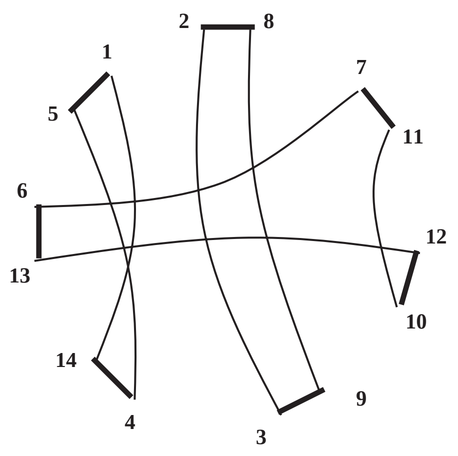

Fig. 2. The circular breakpoint graph of the permutation $\pi = (1\ -4\ \ 6\ -5\ \ 2\ -7\ -3)$, for which $f(\pi) = (1\ 2\ 8\ 7\ 11\ 12\ 10\ 9\ 3\ 4\ 14\ 13\ 6\ 5)$. Black edges are represented as thick lines on the circumference, and gray edges are chords.

motivated biologically compared to revrevs. 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. We note that the problem of sorting circular permutations is important on its own, since many genomes including mitochondrial and bacterial ones are circular.

## *2.2. The breakpoint graph*

We follow the construction of Bafna and Pevzner for representing signed permutations [1]. First, a permutation $\pi$ on $n$ elements is transformed into a permutation $f(\pi) = \pi' = (\pi'_1 \dots \pi'_{2n})$ on $2n$ elements. $f(\pi)$ is obtained by replacing each positive element $i$ with two elements $2i-1, 2i$ (in this order), and each negative element with $2i, 2i-1$. For 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 $\pi$. In the rest of the paper we identify, in both indices and elements, $2n+1$ and 1.

**Definition 1.** The *breakpoint graph* $G(\pi)$ is an edge-colored graph on $2n$ vertices $\{1, 2, \dots, 2n\}$. For every $1 \leqslant i \leqslant n$, $\pi'_{2i}$ is joined to $\pi'_{2i+1}$ by a black edge, and $2i$ is joined to $2i+1$ by a gray edge.

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 Fig. 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 $c_{\text{odd}}(\pi)$. Gu et al. [6] have shown that for all linear permutations $\pi$ and operations $\mu$ (reversals, transpositions, transreversals or revrevs), it holds that $c_{\text{odd}}(\mu \cdot \pi) \leqslant c_{\text{odd}}(\pi) + 2$. Their result holds also for circular permutations and can be used to prove the following lower bound on $d(\pi)$:

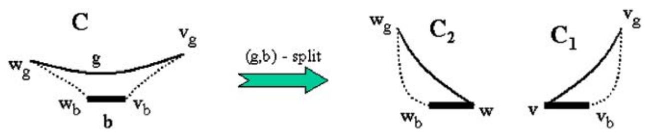

Fig. 3. A $(g, b)$-split (taken from [8]). A dashed line indicates a path.

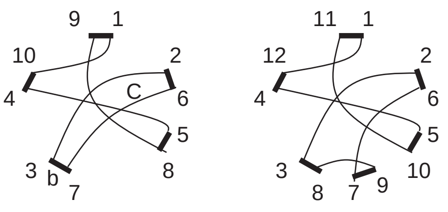

Fig. 4. (a) The breakpoint graph of the permutation $\pi = (1, -3, -4, 2, -5)$. (b) The graph of $(1, -3, -5, 4, 2, -6)$, which is obtained by a $(C, b)$-padding, where $b = (3, 7)$ and $C$ is the 2-cycle containing $b$.

**Theorem 3** (*Gu et al. [6]*). *For all permutations* $\pi$, $d(\pi) \geqslant (n(\pi) - c_{\text{odd}}(\pi))/2$.

### 2.3. *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 $k \leqslant 3$. It is called a 3-*permutation* if it contains only 1-cycles and 3-cycles. A transformation from $\pi$ to $\hat{\pi}$ is called *safe* if $n(\pi) - c_{\text{odd}}(\pi) = n(\hat{\pi}) - c_{\text{odd}}(\hat{\pi})$, 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. We note that the transformation maintains only the lower bound, not the exact distance. Our starting point is the standard safe transformation into simple permutations (cf. [8]). For completeness, we describe it briefly in the sequel.

The transformation into a simple permutation is done by a series of safe cycle splits. Let $b = (v_b, w_b)$ be a black edge and $g = (v_g, w_g)$ be a gray edge belonging to the same cycle $C = (\ldots, v_b, w_b, \ldots, w_g, v_g, \ldots)$ in $G(\pi)$. A $(g, b)$-*split* of $G(\pi)$ creates a new graph $G(\hat{\pi})$ with one more cycle by: (1) removing edges $b$ and $g$; (2) adding two new vertices $v$ and $w$; (3) adding new black edges $(v_b, v)$ and $(w, w_b)$; and (4) adding new gray edges $(w_g, w)$ and $(v, v_g)$. This transformation is demonstrated in Fig. 3. The reader is referred to [8] for a proof that a split results in a new permutation (with one more element) and that every permutation can be transformed into a simple one using safe splits.

It remains 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 $\pi$ 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 $\pi_i$, and is placed after (resp., before) $\pi_i$ if it is positive (resp., negative). Finally, the sign of $\pi_i$ is flipped. The effect on the breakpoint graph is that $C$ is transformed into a 3-cycle (see Fig. 4 for an example). Overall, the permutation after the padding has an additional element and one more odd cycle.

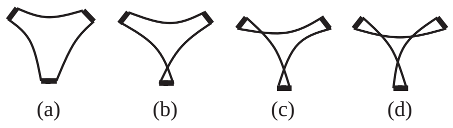

Fig. 5. Configurations of 3-cycles. (a,b) Unoriented 3-cycles. (c,d) Oriented 3-cycles.

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

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

## 2.4. Cycle configurations

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 $k$. 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.

**Definition 2.** 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 Fig. 5a–d. It is easy to verify that cycles $a$ and $b$ are unoriented, whereas $c$ and $d$ are oriented (see Observation 5 below).

**Definition 3.** A black edge is called *twisted* if its two adjacent gray edges cross each other in the circular breakpoint graph. A cycle is *k-twisted* if $k$ of its black edges are twisted. For example, in Fig. 5 cycle $a$ is 0-twisted and $c$ is 2-twisted.

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

**Proof.** For a 3-twisted cycle, a transposition is a 2-operation; for a 2-twisted cycle, a transreversal that reverses the segment between the two twists is a 2-operation. □

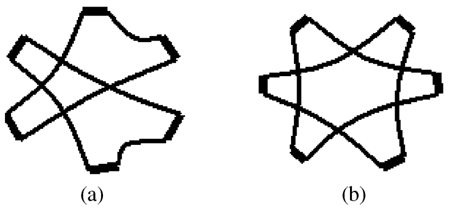
Fig. 6. (a) A pair of intersecting 3-cycles. (b) A pair of interleaving 3-cycles.

We now give terminology to describe certain configurations of cycles. A pair of black edges is said to be *connected* if they are connected by a gray edge. A pair of connected black edges is *coupled* if they are read in the same direction when reading the edges along the cycle (for example, the top edges in Fig. 5b are coupled, and so are all pairs of edges in Fig. 5a).

An *arc* is a segment of the circumference of a circular breakpoint graph. An arc *contains* all black edges whose endpoints belong to the arc's segment. 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 $b$ and $b'$ *induce* two disjoint arcs on the circle, one between $i_2$ and $j_1$ and the other between $j_2$ and $i_1$. Two arcs are called *adjacent* if both endpoints of each arc are connected by gray edges to the endpoints of the other arc. (For example, the arcs induced by the pairs $(1, 2)$ and $(4, 5)$ in Fig. 7.)

Consider a cycle $C$ and two of its black edges $b, b'$. Let $A$ be the arc induced by $b$ and $b'$ that does not contain any other black edge of $C$ (the other induced arc will contain all edges of $C$ except $b$ and $b'$). We shall refer to any black edge of another cycle as lying *between b* and $b'$, if this edge is contained in $A$.

Two pairs of black edges are called *intersecting* if they alternate in the order of their 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 Fig. 6a). Two intersecting cycles are called *interleaving* if their black edges alternate in their order of occurrence along the circle (see Fig. 6b). Thus, the relation between two cycles is one of: (1) non-intersecting; (2) intersecting but non-interleaving (which we will simply call intersecting); or (3) interleaving.

Given a breakpoint graph $G(\pi)$, we define its *complement* as the graph formed from $G(\pi)$ by replacing each black edge that connects $\pi'_{2i}$ to $\pi'_{2i+1}$ with a black edge that connects $\pi'_{2i-1}$ to $\pi'_{2i}$ (this notion is related to Caprara's Hamiltonian Matching [3]). By construction, every vertex $i$ is connected in the complement graph to $i - 1$ and $i + 1$, hence:

**Observation 6.** *The complement breakpoint graph of a permutation is a cycle of length* $2n$.

The following lemma, proved originally by [6], follows from the latter observation:

**Lemma 7** (*Gu et al. [6]*). *Let* $(b_1, b_2)$ *be a pair of coupled black edges. Then there exists another pair of black edges that intersects with* $(b_1, b_2)$.

**Proof.** Suppose to the contrary that no pair intersects $(b_1, b_2)$. Then the complement graph contains at least two disjoint cycles, one in each of the arcs induced by the endpoints of $b_1$ and $b_2$, in contradiction to Observation 6. □

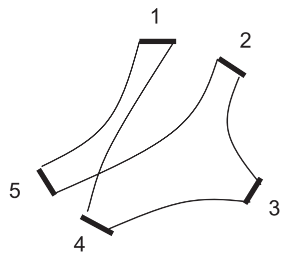

Fig. 7. A 5-cycle with signed canonical labeling $(1, -4, -3, -2, 5)$.

**Lemma 8.** *Let $i_1$ and $i_2$ be a pair of adjacent arcs. Then there exist two connected black edges $b_1$ and $b_2$ such that*: (1) *$b_1$ is either in $i_1$ or in $i_2$; and* (2) *$b_2$ is neither in $i_1$ nor in $i_2$.*

**Proof.** Suppose to the contrary that no such gray edge exists. Then in the complement graph there is a cycle that lies in arcs $i_1$ and $i_2$, but does not include all the vertices, a contradiction. $\square$

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*. As we show next, any graph with a 1-twisted cycle has a closed configuration.

**Observation 9.** *Let $G(\pi)$ be a breakpoint graph that contains a 1-twisted cycle C. Then $G(\pi)$ contains a closed configuration.*

**Proof.** By Lemma 7 there exists another cycle $D$ that intersects with the coupled edges of $C$. The configuration which consists of cycles $C$ and $D$ is closed. $\square$

## *2.5. Canonical labeling of cycles*

In this section we develop a characterization of oriented cycles that allows us to borrow some of the theory developed in [8] for unsigned permutations. A useful tool that we will require is the signed canonical labeling[2] of a cycle in a breakpoint graph, which we present next.

For a given cycle $C$ (of any length), consider the labeling of its black edges obtained by labeling an arbitrary edge by 1, and labeling the rest of the cycle's black edges according to their occurrence in clockwise order along the circle. The *signed canonical labeling* of $C$ is the signed permutation obtained by starting with the edge labeled 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 labeled 1 is positive, and otherwise it is negative (see Fig. 7). This definition captures the notion of twists in 3-cycles; indeed, a 0-twisted 3-cycle has labeling $(1, 2, 3)$, a 1-twisted cycle has labeling $(1, -3, -2)$, etc. Note that a cycle typically has more than one possible canonical labeling, since it depends on the choice of the first edge.

[2] A generalization of the notion of canonical labeling [4].

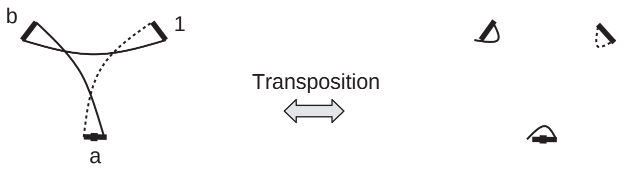

Fig. 8. A general 5-cycle that has a canonical labeling that starts with $1, b, a$. The dashed line stands for a path of two black edges and two gray edges (the black edges can be located anywhere along the circle).

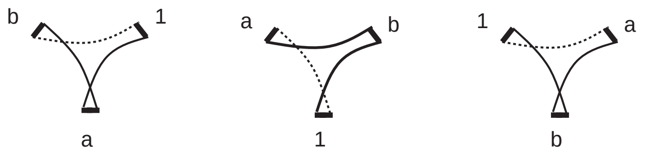

Fig. 9. General 5-cycles that have a canonical labeling that starts with $1, -a, -b$ and $1, -b, a$ and $1, b, -a$.

A canonical labeling 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:

**Lemma 10.** *A 5-cycle is oriented iff it has an oriented canonical labeling.*

**Proof.** A general 5-cycle that has a canonical labeling that starts with $1, b, a$ (where $1 < a < b$) is depicted in Fig. 8. A transposition that acts on $1, a$ and $b$ transforms the cycle into two 1-cycles and one 3-cycle, showing that the 5-cycle is oriented. A general 5-cycle that has a canonical labeling that starts with $1, -a, -b$ (resp., $1, -b, a$ or $1, b, -a$) is depicted in Fig. 9. In these cases we consider a transreversal that acts on these three edges, while reversing the segment between edges $a$ and $b$ (resp., 1 and $a$, or $b$ and 1). This operation breaks the 5-cycle into two 1-cycles and one 3-cycle, implying that the 5-cycle is oriented.

Conversely, consider a 5-cycle $C$ that admits a 2-operation. By definition, this operation creates two 1-cycles and one 3-cycle. An example for the case of a 2-transposition is given in Fig. 8. Thus, starting clockwise with the edge that will be part of the 3-cycle as edge 1, we obtain the requested canonical labeling (see, e.g., Fig. 8). □

**Lemma 11.** *Let $C$ be a 5-cycle that admits a 2-transposition, and let $D$ be a cycle that has the same canonical labeling as $C$, up to flipping the sign of one element. Then $D$ is also oriented.*

**Proof.** By the proof of Lemma 10, $C$ has a canonical labeling that starts with $1, b, a$, where $1 < a < b$. Let $x$ be the element whose sign is flipped. If $x$ is one of the last two elements in the canonical labeling then $D$ still has a labeling that starts with $1, b, a$ and, thus, admits a 2-transposition. If $x$ is the second (resp., third) element of $D$ then $D$ has a canonical labeling that starts with $1, -b, a$ (resp., $1, b, -a$). Hence, by Lemma 10 the 5-cycle is oriented. If $x$ is the first element, we observe that the reflection of $D$ has a canonical labeling that starts with $1, -a, -b$. □

## 3. The algorithm

In this section we provide a 1.5 approximation algorithm for sorting by transpositions and transreversals. We first develop an algorithm for sorting 3-permutations, and then use the results of Section 2.3 to generalize it to arbitrary permutations. By definition, an oriented cycle can be eliminated by a 2-operation that acts on its black edges. Thus, from now on, we will only consider unoriented cycles. Since configurations involving only 0-twisted cycles were handled in [8], by Observation 9 we may restrict attention to closed configurations. For each possible closed configuration we shall prove the existence of a $(0, 2, 2)$-sequence of operations.

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. The following lemma deals with interleaving cycle pairs:

**Lemma 12.** *Let $\pi$ be a permutation that contains two unoriented, interleaving cycles $C$ and $D$ that do not form a* 1*-twisted pair. Then $\pi$ admits a* $(0, 2, 2)$*-sequence.*

**Proof.** If both cycles are 0-twisted then a $(0, 2, 2)$-sequence of transpositions is given in [8]. Suppose that $C$ is 0-twisted and $D$ 1-twisted (resp., both are 1-twisted and their twists are not consecutive on the circle). 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 (resp., 2-transposition) on $C$ completes the $(0, 2, 2)$-sequence. The $(0, 2, 2)$-sequences are depicted in Fig. 10. □

In order to deal with intersecting cycles we use the notion of canonical labeling of cycles, defined in Section 2.5. The following lemma handles the case of two intersecting 0-twisted cycles.

**Lemma 13.** *Let $\pi$ be a permutation that contains a closed configuration with two intersecting,* 0*-twisted cycles $C$ and $D$. Then $\pi$ 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 7 there exists a cycle $E$ that intersects with this pair of edges. The case in which $E$ is 0-twisted

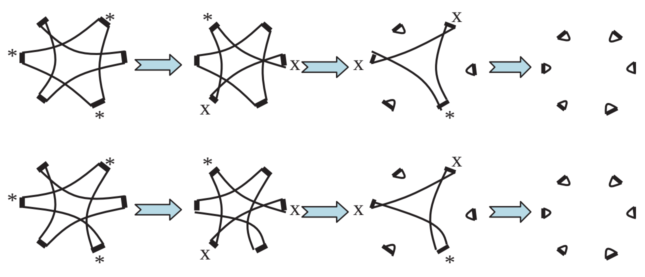

Fig. 10. $(0, 2, 2)$-Sequences for the two cases of two interleaving cycles considered in Lemma 12. Here and throughout the paper, three asterisks represent a transposition that acts on the three marked black edges. Two x's and a asterisk stand for a transreversal that acts on the three marked edges and reverses the segment between the two x's.

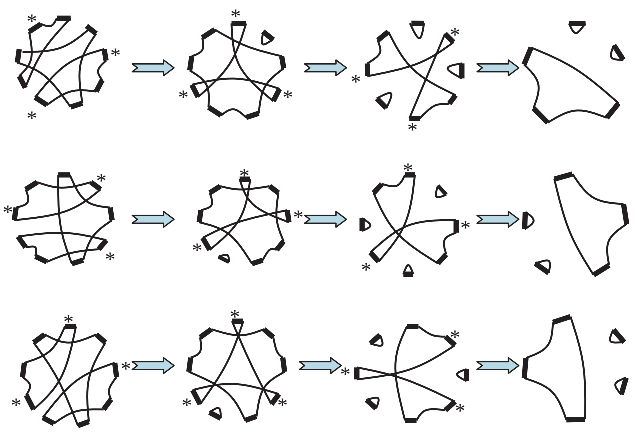
Fig. 11. $(0, 2, 2)$-sequences for three 0-twisted cycles, where two of the cycles are non-intersecting, and a third one intersects both (taken from [8]). At each step the transposition acts on the three black edges marked by an asterisk. For simplicity, every 1-cycle is shown only when it is formed and not in subsequent graphs.

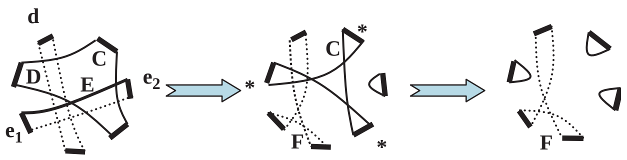

Fig. 12. Three mutually intersecting 0-twisted cycles (taken from [8]). A dashed line represents a path.

was treated in [8]. 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 Fig. 11, 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 Fig. 11 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. Since the three configurations given here are symmetric with respect to the two non-intersecting cycles, we can ensure that the black edge(s) from cycle $E$ that are involved in the transposition do not include a twist. Indeed, if this is not the case, we simply exchange the choice of edges between $D$ and $E$, choosing in each case symmetric edges from the other cycle. By Lemma 11, 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 Fig. 12. In our case either $D$ or $E$ are 1-twisted. If all three edges $d$, $e_1$ and $e_2$ that are cut by the first transposition are non-twisted, we apply the first two transpositions as in Fig. 12. By Lemma 11, 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 $(0, 2, 2)$-sequences are described in Fig. 13. □

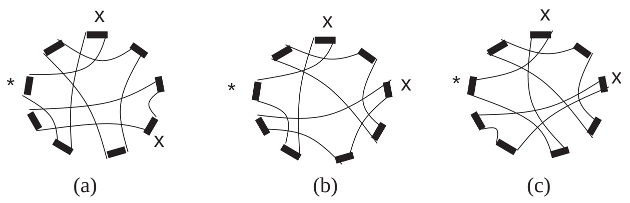

Fig. 13. (0, 2, 2)-sequences for some cycle configurations that contain two intersecting 0-twisted cycles. First we apply a 0-transreversal on the three marked edges, such that the segment between the two x's is reversed, resulting in an oriented 3-cycle and a 5-cycle. Next, we eliminate the oriented 3-cycle and are left with a 5-cycle, which can be verified to be oriented by Lemma 10. Hence, a (0, 2, 2)-sequence is possible.

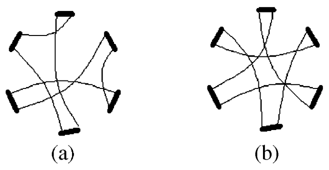

Fig. 14. Examples of configurations that admit a (2, 2)-sequence by (a) Observation 14 and (b) Observation 16.

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

**Observation 14.** *Let $\pi$ be a permutation that contains a 2-twisted cycle $C$ and a 1-twisted cycle $D$, such that $C$ and $D$ are intersecting and none of the arcs induced by the two twists of $C$ contains both non-twists of $D$ (see, e.g., Fig. 14(a)). Then $\pi$ admits two consecutive 2-operations.*

**Proof.** Applying a 2-transreversal on $C$ eliminates it, while making $D$ 2-twisted. Thus, two consecutive 2-operations are possible. □

**Observation 15.** *Let $C$ be a 2-twisted cycle such that in a given configuration there are no black edges from other cycles between its two twists. Then it is possible to apply a 2-operation on $C$ that does not affect other cycles in the configuration.*

**Proof.** A 2-transreversal on $C$ switches the segment between its two twists with a segment between a twist and a non-twist of $C$. Since the former segment does not involve black edges from other cycles in the configuration, the claim follows. □

**Observation 16.** *Let $C$ and $D$ be two 2-twisted, interleaving cycles. Then these cycles admit two consecutive 2-operations iff at least three of their twists are consecutive on the circle.*

**Proof.** Suppose that $C$ and $D$ have at least three consecutive twists (see, e.g., Fig. 14(b)). Then a 2-operation on any of them leaves the other cycle 2-twisted, and the claim follows. Conversely, suppose to the contrary that the non-twist of each cycle lies between the two twists of the other cycle (which is

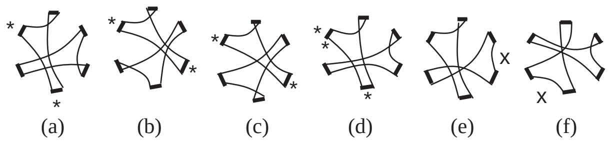

Fig. 15. Closed configurations of two intersecting 1-twisted cycles.

the only configuration in which there are no three consecutive twists). Then a 2-operation on one cycle makes the other one 0-twisted, a contradiction. □

**Lemma 17.** *Let $\pi$ be a permutation that contains a closed configuration with two intersecting,* 1-*twisted cycles C and D. Then $\pi$ admits a* $(0, 2, 2)$-*sequence.*

**Proof.** There are six possible cases, shown in Fig. 15. For cases (a–d), we first apply a 0-reversal that acts on the black edges that are marked by an asterisk. This makes the other cycle 2-twisted, and two additional 2-operations follow from Observation 14. For cases (e–f), we observe that by Lemma 7 there is another cycle that has a black edge in the arc denoted $x$, and a black edge in one of the other five arcs. We apply a reversal that acts on these two edges of the third cycle. If the edges are coupled then the reversal does not affect the cycle. Otherwise, it breaks this 3-cycle into a 1-cycle and a 2-cycle (which we safely transform into a 3-cycle). Thus, in both cases the reversal is a 0-operation. To show that two 2-operations follow, we consider three possible cases with respect to the resulting configuration of $C$ and $D$:

- The resulting configuration consists of a 1-twisted cycle and a 2-twisted cycle. In all cases the configuration fulfills the condition of Observation 14 and, thus, two 2-operations follow.
- $C$ and $D$ intersect and are both 2-twisted. In all cases one of the cycles fulfills the condition of Observation 15 and can be eliminated, while leaving the other cycle 2-twisted. Thus, two 2-operations are possible.
- $C$ and $D$ interleave and are both 2-twisted. In all cases $C$ and $D$ have at least three consecutive twists on the circle, and two 2-operations follow from Observation 16. □

The following two lemmas deal with closed configurations that involve two intersecting cycles, one of which is 0-twisted and the other 1-twisted.

**Lemma 18.** *Let $\pi$ be a permutation that contains a* 0-*twisted cycle C that intersects with the coupled edge pairs of two non-intersecting,* 1-*twisted cycles D and E. Then $\pi$ admits a* $(0, 2, 2)$-*sequence.*

**Proof.** Since cycles $D$ and $E$ both intersect with $C$, there is an arc $a$ induced by two black edges of $C$ that includes edges from both $D$ and $E$. We apply a 0-transreversal on the edges of $C$ that reverses $a$. In the resulting permutation, $D$ and $E$ are both 2-twisted and remain non-intersecting, implying two 2-transreversals. An example of such a $(0, 2, 2)$-sequence is given in Fig. 16. □

**Lemma 19.** *Let $\pi$ be a permutation that contains a* 0-*twisted cycle, which intersects with the coupled edges of a* 1-*twisted cycle. Then $\pi$ admits a* $(0, 2, 2)$-*sequence.*

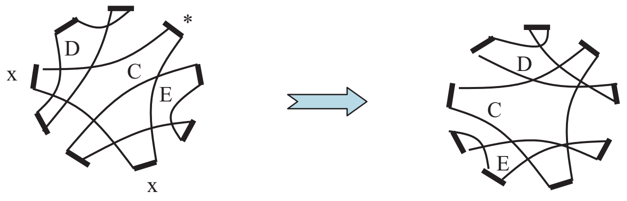

Fig. 16. One possible scenario of Lemma 18. $C$ is a 0-twisted cycle that intersects with the coupled edge pairs of two non-intersecting, 1-twisted cycles $D$ and $E$. First we apply a 0-transreversal on the three marked edges. Now both $D$ and $E$ become oriented and remain non-intersecting, allowing two 2-transreversals.

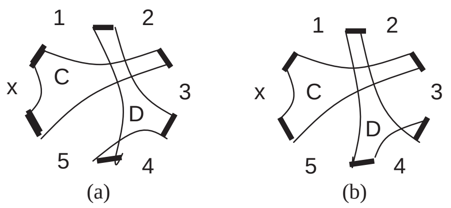

Fig. 17. Closed configurations consisting of two intersecting cycles, such that one is 0-twisted and the other is 1-twisted.

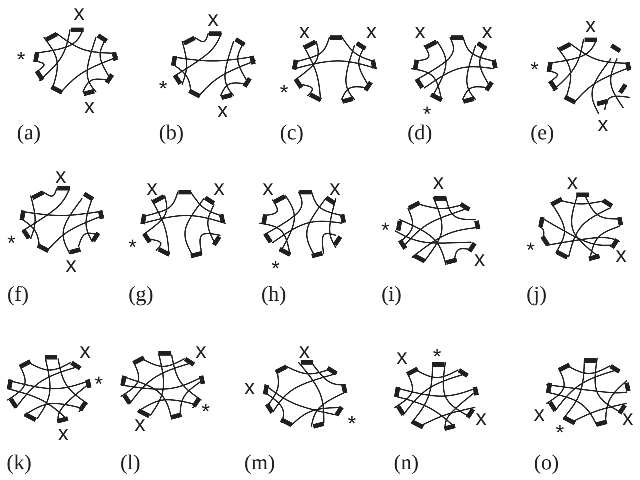

Fig. 18. (0, 2, 2)-sequences for configurations described in Table 1.

**Proof.** There are two possible configurations of a 0-twisted cycle $C$ that intersects the coupled edges of a 1-twisted cycle $D$, depicted in Fig. 17. By Lemma 7, there is another cycle $E$ that has a black edge in the arc marked by $x$, and in some other arc(s) (see Fig. 17). If $E$ is 0-twisted then a $(0, 2, 2)$-sequence follows from Lemma 13. Otherwise, $E$ is 1-twisted and $(0, 2, 2)$-sequences for all possible configurations are summarized in Table 1 (see also Fig. 18). □

Table 1
(0, 2, 2)-sequences for the two configurations shown in Fig. 17

| Configuration of cycles $C$ & $D$ | 2nd black edge of $E$ in arc | 3rd black edge of $E$ in arc | Twisted edge in arc | (0, 2, 2)-sequence by |
|---|---|---|---|---|
| a,b | 1 | 2 | all cases | Lemma 17 |
| a,b | 1 or 2 | 3 or 4 or 5 | all cases | Lemma 12 |
| a | 3 | 4 | 4 | Figure 18(k) |
| a | 3 | 4 | $x$ | Figure 18(l) |
| a | 3 | 4 | 3 | Lemma 12 |
| b | 3 | 4 | 4 | Figure 18(n) |
| b | 3 | 4 | $x$ | Lemma 12 |
| b | 3 | 4 | 3 | Figure 18(o) |
| a,b | 3 | 5 | all cases | Lemma 17 |
| a | 4 | 5 | 4 | Figure 18(m) |
| a | 4 | 5 | $x$ or 5 | Lemma 17 |
| b | 4 | 5 | all cases | Lemma 17 |
| a,b | 1, 5 | same arc | $E$ closed by $C$ | Lemma 18 |
| a | 1, 5 | same arc | $E$ not closed by $C$ | Figure 18(a–d) |
| b | 1, 5 | same arc | $E$ not closed by $C$ | Figure 18(e–h) |
| a,b | 2, 3 | same arc | all cases | Lemma 17 |
| a | 4 | same arc | $E$ not closed by $C$ | Figure 18(i–j) |
| a | 4 | same arc | $E$ closed by $C$ | Lemma 17 |
| b | 4 | same arc | all cases | Lemma 17 |

Each row of the table describes possible configurations of cycles $C$, $D$ and $E$. The first column indicates to which of the two configurations from Fig. 17 this row refers. As explained in the proof of Lemma 19, cycle $E$ has black edges in arc $x$ and in some other arc(s), with the possible locations listed in the second and third columns. The fourth column specifies the location of the twist of $E$, completing the description of the configuration of cycles $C$, $D$ and $E$. The last column points to a reference for a (0, 2, 2)-sequence in each case.

The final configuration that needs to be taken care of involves 1-twisted pairs of interleaving cycles. This case is handled by the following lemma:

**Lemma 20.** *Let $\pi$ be a permutation that contains $k \geqslant 2$ mutually interleaving* 1*-twisted cycles, such that all their twists are consecutive on the circle and k is maximal with this property. Then $\pi$ admits a* (0, 2, 2)*-sequence.*

**Proof.** Such an arrangement is possible iff the edges of all cycles alternate along the circle (see Fig. 19). Consider the arc induced by the twist of the $k$th cycle and the first non-twisted edge of the first cycle; further consider the arc induced by the twist of the first cycle and the second non-twisted edge of the $k$th cycle. These two arcs (marked by an asterisk in Fig. 19) are adjacent, so by Lemma 8 there is a cycle $C$ that has a black edge in one of these arcs, and another black edge in some other arc.

Now, if $C$ interleaves with one of the $k$ interleaving cycles then a (0, 2, 2)-sequence follows from Lemma 12 (the case in which $C$ is 1-twisted and interleaves with all $k$ cycles is impossible, since it contradicts the maximality of $k$). If $C$ has an edge that lies between the two non-twists of one of the $k$ cycles, then these two intersecting cycles form a closed configuration; a (0, 2, 2)-sequence then follows from Lemma 19, if $C$ is 0-twisted (resp., Lemma 17, if $C$ is 1-twisted). If $C$ is 1-twisted, has an edge between two twists of

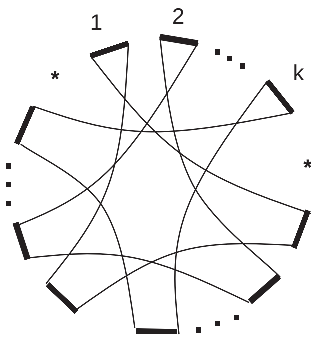

Fig. 19. Mutually interleaving 1-twisted cycles.

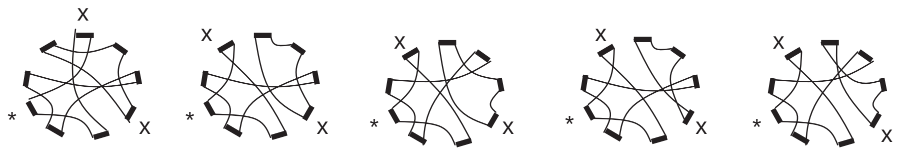

Fig. 20. $(0, 2, 2)$-sequences for some closed configurations that involve a 1-twisted pair and a third cycle that intersects at least one of the other cycles.

the $k$ cycles and forms a closed configuration with one of the cycles it intersects, then a $(0, 2, 2)$-sequence follows from Lemma 17. There are five other configurations that are not covered by these cases. The five configurations and $(0, 2, 2)$-sequences for them are depicted in Fig. 20. □

We are now ready to describe our sorting algorithm, which is given in Fig. 21. For 3-permutations, Step 2 suffices for the sorting and we call this part *Algorithm Sort3Perm*. The following lemma shows that Algorithm Sort3Perm is a quadratic 1.5-approximation algorithm for sorting 3-permutations:

**Lemma 21.** *Algorithm* Sort3Perm *is a* 1.5-*approximation algorithm for sorting* 3-*permutations*, *running in time* $O(n^2)$.

**Proof.** First, we observe that it suffices to consider only closed configurations, since for configurations that contain only 0-twisted cycles a $(0, 2, 2)$-sequence is given in [8]; in all other cases, by Observation 9 there must exist a closed configuration.

The sequence of operations that is 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 now analyze the running time of the algorithm. The number of iterations in step 2 is linear, since in every iteration two cycles are eliminated and there is a linear number of cycles. Deciding whether two given cycles are interleaving or intersecting can be done in constant time. Thus, the bottleneck in each iteration is to apply an operation to the permutation, and to find an arbitrary cycle that intersects a given coupled pair of black edges (Step 2b). The latter task can be performed by a procedure that finds

```
Algorithm Sort ($\pi$)

1. Transform $\pi$ 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 to it.
    (b) Otherwise, find a cycle $D$ that intersects with a coupled pair of $C$.
    (c) If $D$ is oriented then apply a 2-operation to it.
    (d) Else if $C$ and $D$ interleave, apply a $(0,2,2)$-sequence* (Lemmas 12, 20).
    (e) Else if $C$ or $D$ are 1-twisted, apply a $(0,2,2)$-sequence* (Lemmas 17, 19).
    (f) Otherwise, apply a $(0,2,2)$-sequence* (Lemma 13).
    (g) If new 2-cycles were introduced by the last operations, transform $\hat{\pi}$
        into a 3-permutation $\hat{\pi}'$ by safe paddings (Lemma 4), and let $\hat{\pi} = \hat{\pi}'$.
3. Mimic the sorting of $\pi$ using the sorting of $\hat{\pi}$ (Lemma 4).
```

Fig. 21. Algorithm Sort. (*) If any oriented cycle is found to be involved in the resulting configurations, a 2-operation is applied to it and the algorithm continues with the next iteration.

an arbitrary pair of connected black edges that intersects with a given pair of connected black edges. These tasks can be done easily in linear time. The number of new 2-cycles introduced in each iteration is constant and, therefore, Step 2g takes constant time. Overall, the algorithm can be implemented in quadratic time. □

Now we are ready to prove the correctness of Algorithm Sort:

**Theorem 22.** *Algorithm* Sort *is a* 1.5-*approximation algorithm for sorting arbitrary permutations by transpositions and transreversals, and it runs in time* $O(n^2)$.

**Proof.** By Lemma 21, we are guaranteed that $alg(\hat{\pi}) \leqslant 1.5d(\hat{\pi})$, where $alg(\hat{\pi})$ is the number of operations used by Algorithm Sort3Perm to sort $\hat{\pi}$. Thus, by Theorem 3

$$alg(\hat{\pi}) \leqslant 1.5d(\hat{\pi}) \leqslant 1.5\left(\frac{n(\hat{\pi}) - c_{\text{odd}}(\hat{\pi})}{2}\right) = 1.5\left(\frac{n(\pi) - c_{\text{odd}}(\pi)}{2}\right) \leqslant 1.5d(\pi).$$

Using Lemma 4, we can sort $\pi$ by $alg(\hat{\pi})$ operations, which implies an approximation ratio of 1.5.

Since the transformation into 3-permutations can be done in linear time, Lemma 21 implies that the running time of Algorithm Sort is $O(n^2)$. □

## 4. An $O(n^{3/2}\sqrt{\log n})$ implementation of Algorithm Sort

In this section we give a faster implementation of Algorithm Sort, using a special data structure, introduced in [11]. As explained in the proof of Lemma 21, the bottleneck in each iteration is to apply an

$$\left(\begin{array}{ccccc|ccccc|cccc} 1 & 2 & 11 & 12 & 10 & 9 & 7 & 8 & 14 & 13 & 6 & 5 & 3 & 4 \\ \scriptstyle 9 & \scriptstyle 13 & \scriptstyle 5 & \scriptstyle 10 & \scriptstyle 3 & \scriptstyle 8 & \scriptstyle 11 & \scriptstyle 6 & \scriptstyle 1 & \scriptstyle 4 & \scriptstyle 7 & \scriptstyle 14 & \scriptstyle 2 & \scriptstyle 12 \end{array}\right.$$

(a)

$$[10\ 11\ 1\ 12\ 2]\quad[14\ 13\ 8\ 9\ 7]\quad[3\ 6\ 4\ 5]$$

(b)

Fig. 22. Let $\pi = (1\ 6\ \ -5\ 4\ \ -7\ \ -3\ 2)$, $f(\pi) = (1\ 2\ 11\ 12\ 10\ 9\ 7\ 8\ 14\ 13\ 6\ 5\ 3\ 4)$. (a) Partition of the permutation into roughly $\Theta(\sqrt{n/\log n})$ blocks. Below each element, the location of its mate in $\pi$ is indicated. (b) The internal order of the elements in each block (according to the order of their mates in $\pi$.)

operation to the permutation, and to find an arbitrary cycle that intersects a given coupled pair of black edges (Step 2b). In the sequel we describe a data structure that allows to perform these tasks in sub-linear time. This data structure is similar to the one suggested by Kaplan and Verbin in [11], although here the required tasks are slightly different.

For clarity, we describe the data structure for linear permutations. Consider the doubled permutation $f(\pi)$ (see Section 2.2), which is denoted here simply by $\pi$. A connected pair of black edges $(b_1, b_2)$ is represented by the pair $(2i, 2i+1)$, which is connected by a gray edge. Thus, $\pi$ is a union of disjoint pairs. The two elements in each pair are called *mates*. We need a data structure that supports the following tasks in sub-linear time:

- *Query*$(\pi, e_1, e_2)$: Find a pair that intersects $(e_1, e_2)$ in $\pi$.
- *Operation*$(\pi, e_1, e_2, e_3)$: Apply an operation on $\pi$ that cuts before elements $e_1$, $e_2$ and $e_3$ (for reversals $e_2 = e_3$).

Next, we describe the data structure. The permutation $\pi$ is divided into $\Theta(\sqrt{n/\log n})$ blocks of size $\Theta(\sqrt{n \log n})$ each. The elements in each block are ordered according to the order of their mates in $\pi$ (see an example in Fig. 22). A splay tree [15] is attached to each block, in which the elements of the block are maintained. This is a balanced binary search tree that is re-balanced via rotations, and supports splits and concatenations in logarithmic time. We also maintain a lookup-table that contains for each element a pointer to its block.

In order to maintain signed permutations we introduce a "*reverse*" *flag* for each node in the splay tree. This flag indicates whether the elements of its subtree should be reversed (in order and sign). The reverse flag of a node can be flipped by exchanging the order of its two children, and flipping their own flags. The ability to clear the reverse flag allows us to implement splits and concatenations, while correctly maintaining the permutation [11].

As shown in [11], we may assume that queries and operations involve only elements that are on block boundaries. More specifically, for queries we assume in the sequel that $e_1$ is the first element in its block and $e_2$ is the last element in its block. For operations we assume in the sequel that $e_1$, $e_2$ and $e_3$ are all first elements in their blocks.

**Lemma 23.** *Tasks Query and Operation can be performed in time $O(\sqrt{n \log n})$.*

**Proof.**

- *Query*$(\pi, e_1, e_2)$: Let $B_1$ and $B_2$ be the blocks that contain $e_1$ and $e_2$ (the blocks are found by using the lookup-table) and assume, w.l.o.g., that $B_1$ is located before $B_2$. For each block which is before $B_1$ or after $B_2$ do the following: Split the attached splay tree after location $e_1$ and before $e_2$, and consider the subtree that is bounded by these two elements. If this subtree is not empty then pick an arbitrary element in it. By construction, this element and its mate are intersecting with $(e_1, e_2)$, i.e., the query is answered. Otherwise, continue to the next block. The split is done in logarithmic time. Since there are $O(\sqrt{n/\log n})$ blocks between blocks $B_1$ and $B_2$, the total time is $O(\sqrt{n \log n})$.

- *Operation*$(\pi, e_1, e_2, e_3)$: A transposition can be done by applying three reversals, and a transreversal or a revrev can be mimicked by two reversals. Since a reversal takes time $O(\sqrt{n \log n})$ [11], all operations can be done within the same time bound. □

**Corollary 24.** *Each iteration in Step* 2 *of Algorithm Sort can be implemented in time* $O(\sqrt{n \log n})$.

We are now ready to state the main result of this paper:

**Theorem 25.** *Algorithm Sort is a* 1.5*-approximation algorithm for sorting by transpositions and transreversals, which runs in time* $O(n^{3/2}\sqrt{\log n})$.

**Proof.** The number of iterations in the algorithm is linear. By Corollary 24, each iteration can be implemented in time $O(\sqrt{n \log n})$. In total, the algorithm runs in time $O(n^{3/2}\sqrt{\log n})$. □

## Acknowledgments

We thank Ron Shamir for many invaluable discussions, and Elad Verbin for discussions on the data structure. T.H. was supported in part by an Israel Science foundation Grant 309/02 (PI: Ron Shamir) and by ISF Grant IS265/02. The work of R.S. was done while doing his post-doc at the University of California at Berkeley, and was supported in part by NSF ITR Grant CCR-0121555.

## References

[1] V. Bafna, P.A. Pevzner, Genome rearrangements and sorting by reversals, SIAM J. Comput. 25 (2) (1996) 272–289.
[2] V. Bafna, P.A. Pevzner, Sorting by transpositions, SIAM J. Discrete Math. 11 (2) (1998) 224–240.
[3] A. Caprara, Sorting permutations by reversals and Eulerian cycle decompositions, SIAM J. Discrete Math. 12 (1) (1999) 91–110.
[4] D.A. Christie, Genome rearrangement problems, Ph.D. Thesis, University of Glasgow, 1999.
[5] I. Elias, T. Hartman, A 1.375-approximation algorithm for sorting by transpositions, Submitted, 2004.
[6] Q.P. Gu, S. Peng, H. Sudborough, A 2-approximation algorithm for genome rearrangements by reversals and transpositions, Theoret. Comput. Sci. 210 (2) (1999) 327–339.
[7] S. Hannenhalli, P. Pevzner, Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals, J. ACM 46 (1999) 1–27.
[8] T. Hartman, A simpler 1.5-approximation algorithm for sorting by transpositions, in: Proceedings of the 14th Annual Symposium on Combinatorial Pattern Matching (CPM '03), Springer, Berlin, 2003, , pp. 156–169.
[9] T. Hartman, R. Sharan. A 1.5-approximation algorithm for sorting by transpositions and transreversals, in: Proceedings of the 4th Workshop on Algorithms in Bioinformatics (WABI'04), 2004, pp. 50–61.
[10] S.B. Hoot, J.D. Palmer, Structural rearrangements, including parallel inversions, within the chloroplast genome of Anemone and related genera, J. Mol. Evol. 38 (1994) 274–281.

[11] H. Kaplan, E. Verbin, 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 (CPM ’03), Springer, Berlin, 2003, , pp. 170–185.
[12] G.H. Lin, G. Xue, Signed genome rearrangements by reversals and transpositions: models and approximations, Theoret. Comput. Sci. 259 (2001) 513–531.
[13] J. Meidanis, M.E. Walter, Z. Dias, Reversal distance of signed circular chromosomes, Unpublished, 2000.
[14] J.D. Palmer, L.A. Herbon, Tricircular mitochondrial genomes of Brassica and Raphanus: reversal of repeat configurations by inversion, Nucleic Acids Research 14 (1986) 9755–9764.
[15] D.D. Sleator, R.E. Tarjan, Self-adjusting binary search trees, J. Assoc. Comput. Mach. 32 (1985) 652–686.
[16] M.E. Walter, Z. Dias, J. Meidanis, Reversal and transposition distance of linear chromosomes, in: String Processing and Information Retrieval: A South American Symposium (SPIRE 98), 1998, pp. 96–102.