Sorting by Weighted Reversals, Transpositions, and Inverted Transpositions

Martin Bader Enno Ohlebusch

Faculty of Engineering and Computer Sciences,
University of Ulm, 89069 Ulm, Germany.
Email: martin.bader@uni-ulm.de
enno.ohlebusch@uni-ulm.de

Abstract

During evolution, genomes are subject to genome rearrangements that alter the ordering and orientation of genes on the chromosomes. If a genome consists of a single chromosome (like mitochondrial, chloroplast, or bacterial genomes), the biologically relevant genome rearrangements are (1) *inversions*—also called *reversals*—where a section of the genome is excised, reversed in orientation, and reinserted and (2) *transpositions*, where a section of the genome is excised and reinserted at a new position in the genome; if this also involves an inversion, one speaks of an *inverted transposition*. To reconstruct ancient events in the evolutionary history of organisms, one is interested in finding an optimal sequence of genome rearrangements that transforms a given genome into another genome. It is well known that this problem is equivalent to the problem of "sorting" a signed permutation into the identity permutation. In this paper, we provide a 1.5-approximation algorithm for sorting by weighted reversals, transpositions and inverted transpositions for biologically realistic weights.

1 Introduction

During evolution, genomes are subject to genome rearrangements that alter the ordering and orientation (strandedness) of genes on the chromosomes. Because these events are rare compared to point mutations, they can give us valuable information about ancient events in the evolutionary history of organisms. For this reason, one is interested in the most "plausible" genome rearrangement scenario between two (or multiple) species. More precisely, given two genomes, one wants to find an optimal (shortest) sequence of rearrangement operations that transforms one into the other. Here we will focus on genomes that consist of a single (circular) molecule of DNA such as mitochondrial, chloroplast or bacterial genomes. As usual, the genomes are represented by a signed permutation, i.e., an ordering of signed genes where the sign indicates the orientation (the strand). In this paper we do not consider unsigned permutations. In the single chromosome case, the relevant genome rearrangements are *inversions* (where a section of the genome is excised, reversed in orientation, and reinserted) and *transpositions* (where a section of the genome is excised and reinserted at a new position in the genome; if this also involves an inversion, one speaks of an *inverted transposition*). As is usually done in bioinformatics, we will use the terms "reversal" and "transreversal" as synonyms for "inversion" and "inverted transposition." It is well known that the problem of finding an optimal sequence of rearrangement operations that transforms a permutation into another permutation is equivalent to the problem of "sorting" a permutation by the same set of operations into the identity permutation.
Let us briefly recall what is known for various sets of operations. In a seminal paper, Hannenhalli and Pevzner showed that the problem of sorting signed permutations by reversals can be solved in polynomial time [18]. The Hannenhalli-Pevzner theory was simplified [5] and the running time of their algorithm was improved several times. To date, a subquadratic time algorithm [26] is available, and the reversal distance problem (which asks solely for the minimum number of required reversals, but not for the sequence of reversals) is solvable in linear time [1, 6]. It is also worth mentioning that the problem of sorting an *unsigned* permutation by reversals is NP-hard [10] and the currently best approximation algorithm has the performance ratio 1.375 [8].

If one restricts the set of operations to transpositions (T), to transpositions and reversals (T + R), or to transpositions, reversals, and transreversals (T + R + TR), the complexity of the problem is still unknown. There exist polynomial-time approximation algorithms, and the best of them are listed in the table below.

operations	T	T + R	T + R + TR
performance ratio	1.375	2	1.5
references	[14]	[21, 27]	[20]

The biologically most relevant scenario is the T + R + TR case because in reality genomes are reorganized by all three kinds of operations. Hartman and Sharan provided a very efficient 1.5-approximation algorithm for this case [20], extending a 1.5-approximation algorithm for sorting by transpositions only [19]. However, the drawback of their algorithm is that it applies only to the case in which reversals and transpositions are weighted equally (called the unweighted case in this paper). Because a transposition can create two cycles in the reality-desire diagram while a reversal can create at most one cycle (see below), the algorithm generally favors transpositions. Consequently, the sequence of rearrangement operations returned by that algorithm will often significantly deviate from the "true" evolutionary history

because in most organisms transpositions are observed much less frequently than reversals. Thus, it is desirable to have the possibility of weighting reversals and transpositions differently. Given such weights, the weighted genome rearrangement problem asks for a sorting sequence of rearrangement operations such that the sum of the weights of the operations in the sequence is minimal. That is, a shortest sequence is not necessarily optimal. However, this problem is poorly studied. To our knowledge, there are only two algorithms that tackle it. The first is a $(1+\varepsilon)$-approximation algorithm devised by Eriksen [15]. It uses a weight proportion 2:1 (transposition:reversal) and has the tendency to use as many reversals as possible. The second algorithm is implemented in the software tool DERANGE II [9]. It is a greedy algorithm that works on the breakpoint distance and can only guarantee an approximation ratio of 3.
In this paper, we will present a 1.5-approximation algorithm for any weight proportion between 1:1 and 2:1. Hence, our result closes the gap between the result of Hartman and Sharan [20] for the 1:1 proportion and that of Eriksen [15] for the 2:1 proportion. As the previous state of the art approximation algorithms for this problem, our algorithm proceeds by case analysis. In contrast to them, however, it is based on a (nontrivial) lower bound on the weighted rearrangement distance that is based on the number of odd *and* the number of even cycles in the reality-desire diagram. The running time of our algorithm is $O(n^2)$.

2 Preliminaries

A *signed circular permutation* $\pi = (\pi_1 \dots \pi_n)$ is a permutation of $(1 \dots n)$, in which the indices are cyclic (i.e., n is followed by 1) and each element is labeled by plus or minus. We will use the term "permutation" as short hand for signed circular permutation. The *reflection* of a permutation π is the permutation $(-\pi_n \cdots -\pi_1)$. It is considered to be equivalent to π. Two consecutive elements π_i, π_{i+1} form an *adjacency* if $\pi_i = +x$ and $\pi_{i+1} = +(x+1)$, or if $\pi_i = -x$ and $\pi_{i+1} = -(x-1)$. Otherwise, they form a *breakpoint*. A *segment* $\pi_i \dots \pi_j$ (with $j \geq i$) of a permutation π is a consecutive sequence of elements in π, with π_i as first element and π_j as last element.
There are three possible rearrangement operations on a permutation π. A *transposition* $t(i,j,k)$ (with $i < j$ and $k < i$ or $k > j$) is an operation that cuts the segment $\pi_i \dots \pi_{j-1}$ out of π, and reinserts it before the element π_k. A *reversal* $r(i,j)$ (with $i < j$) is an operation that inverts the order of the elements of the segment $\pi_i \dots \pi_{j-1}$. Additionally, the sign of every element in the segment is flipped. A *transreversal* $tr(i,j,k)$ (with $i < j$ and $k < i$ or $k > j$) is the composition $t(i,j,k) \circ r(i,j)$ of a reversal and a transposition. In other words, the segment $\pi_i \dots \pi_{j-1}$ will be cut out of π, inverted, and reinserted before π_k. A *sequence* of operations $op_1, op_2, \dots, op_k$ applied to a permutation π yields the permutation $op_k \circ op_{k-1} \circ \cdots \circ op_1(\pi)$. In the following, reversals have weight w_r and transpositions as well as transreversals have weight w_t. As reversals usually occur much more frequently than transpositions and transreversals, we assume that $w_r \leq w_t$. The weight of a sequence is the sum of the weights of the operations in it. The problem of *sorting by weighted reversals, transpositions, and inverted transpositions* is defined as follows: Given a permutation π, find a sequence (of these operations) of minimum weight that transforms π into the identity permutation. This minimum weight will be denoted by $w(\pi)$.

In practice, it is also of interest to sort linear permutations. It has been proven by Hartman and Sharan [20] that sorting circular permutations is linearly equivalent to sorting linear

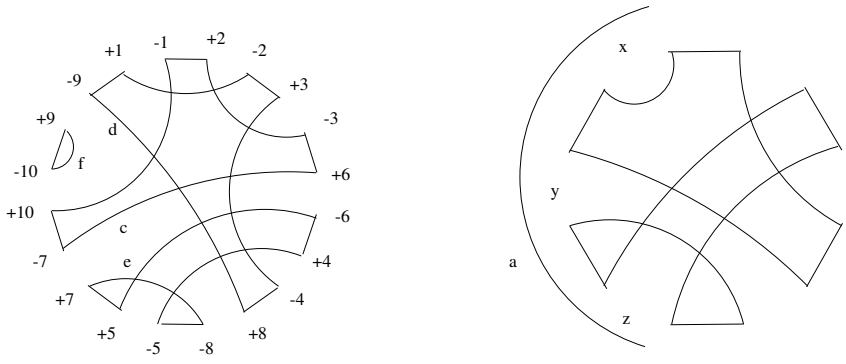

Figure 1: Left: The reality-desire diagram of $\pi = (+1+9+10+7-5+8+4+6+3+2)$ contains the cycles c, d, e, and f. Cycles d and e are intersecting, cycles c and d are interleaving, and all other pairs of cycles do not intersect. Right: The configuration that consists of the cycles d and e. Labels x, y, and z mark three positions in the configuration, and the arc a consists of these positions.

permutations if yet another operation *revrev* is used that inverts two consecutive segments of the permutation. As long as the weights for transreversals and revrevs are the same, the proof also holds for sorting with weighted operations. Hence, our algorithm for circular permutations can be adapted to an algorithm for linear permutations that also uses revrevs.

2.1 The reality-desire diagram

The reality-desire diagram [25] is a graph that helps us analyzing the permutation; see Fig. 1. It is a variation of the breakpoint graph first described in [2]. The reality-desire diagram of a permutation $\pi = (\pi_1 \dots \pi_n)$ can be constructed as follows. First, the elements of π are placed counterclockwise on a circle. Second, each element x of π labeled by plus is replaced with the two nodes $-x$ and $+x$, while each element x labeled by minus is replaced with $+x$ and $-x$. We call the first of these nodes the *left node* of x and the other the *right node* of x. Third, *reality-edges* are drawn from the right node of π_i to the left node of π_{i+1} for each index i (indices are cyclic). Fourth, *desire-edges* or *chords* are drawn from node $+x$ to node $-(x+1)$ for each element x of π. We can interpret reality-edges as the actual neighborhood relations in the permutation, and desire-edges as the desired neighborhood relations.
As each node is assigned exactly one reality-edge and one desire-edge, the reality-desire diagram decomposes uniquely into cycles. The *length* of a cycle is the number of chords in it. A *k-cycle* is a cycle of length k. If k is odd (even), we speak of an *odd* (*even*) cycle. The number of odd (even) cycles in π is denoted by $c_{odd}(\pi)$ ($c_{even}(\pi)$). It is easy to see that a 1-cycle corresponds to an adjacency and vice versa. A reversal cuts the permutation at two positions, while a transposition (transreversal) cuts it at three positions. Hence each of the operations cuts two or three reality-edges and moves the nodes. We say that the operation *acts* on these edges. Desire-edges are never changed by an operation.

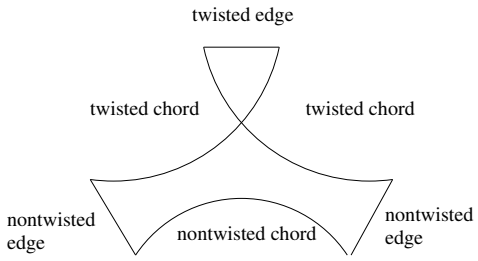

Figure 2: An example for twisted reality-edges and twisted chords.

2.2 Some observations about cycles

The following notions are illustrated in Fig. 1. A *configuration* is a subset of the cycles of the reality-desire diagram of a permutation. Configurations help us to focus on a few cycles in the reality-desire diagram instead of examining the whole diagram. A *position* in a configuration is the position between two consecutive reality-edges in the configuration. An *arc* a is a series of consecutive positions of a configuration, bounded by two reality-edges r_1 and r_2. Two chords d_1 and d_2 are *intersecting* if they intersect in the reality-desire diagram. More precisely, the endpoints of the chords must alternate along the circle in the configuration. Two cycles are intersecting if a pair of their chords is intersecting. Two cycles are *interleaving* if their reality-edges alternate along the circle. A rearrangement operation is called x_y*-move* if it increases the number of cycles by x and the operation is of type y (where r stands for a reversal, t for a transposition, and tr for a transreversal). For example, a transposition that splits one cycle into three is a 2_t-move. A reversal that merges two cycles is a -1_r-move. An $m_1m_2 \ldots m_n$-sequence is a sequence of n operations in which the first is an m_1-move, the second an m_2-move and so on. A cycle c is called *r-oriented* if there is a 1_r-move that acts on two of the reality-edges of c. Otherwise, the cycle is called r-unoriented. A cycle c is called *t-oriented* if there is a 2_t-move or a 2_{tr}-move that acts on three of the reality-edges of c. Otherwise, the cycle is called t-unoriented. A reality-edge is called *twisted* if its adjacent chords are intersecting; see Fig. 2. A chord is called twisted if it is adjacent to a twisted reality-edge; otherwise, it is called nontwisted. A cycle is called *k-twisted* if k of its reality-edges are twisted. If $k = 0$, we also say that the cycle is nontwisted.

Lemma 2.1 *A 2-cycle is r-oriented if and only if it is 2-twisted.*

Proof There are only two possible configurations for a 2-cycle. If the cycle is 2-twisted, a reversal that acts on its reality-edges splits the cycle into two 1-cycles (adjacencies). Otherwise, no such move is possible. □

Lemma 2.2 *(proven in [19]) A 3-cycle is t-oriented if and only if it is 2- or 3-twisted.*

Lemma 2.3 *Let c be a t-unoriented 3-cycle and let a_1, a_2, and a_3 be the three arcs induced by the reality-edges of c. If a transposition acts on three reality-edges, so that one of them is in $a1$, one in $a2$, and one in $a3$, then c becomes t-oriented. More precisely, if c was nontwisted, it becomes 3-twisted, and if c was 1-twisted, it becomes 2-twisted.*

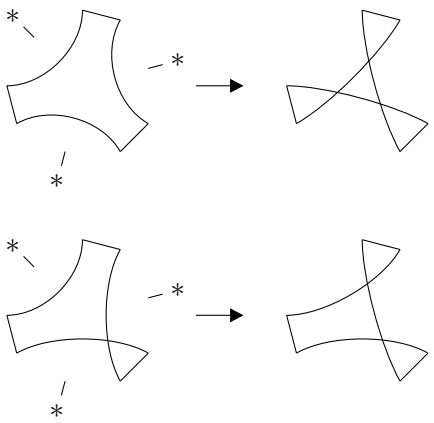

Figure 3: A transposition that acts on three reality-edges in different arcs induced by a t-unoriented cycle c orientates c.

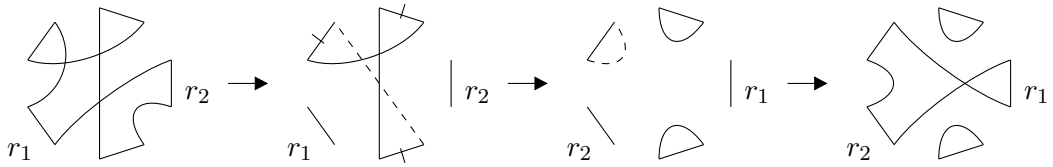

Figure 4: This 5-cycle allows a 2_t-move that splits the cycle into a 3-cycle and two adjacencies. First, we mark two reality-edges connected by a desire-edge with r_1 and r_2, remove them, and close the cycle by a new desire-edge (dashed line). Then, we perform the transposition (third picture). The last step is to reinsert the reality-edges r_1 and r_2.

Proof The transposition just moves a segment containing one of the reality-edges in between the other reality-edges. The results of this move can be seen in Fig. 3. □

Lemma 2.4 *Let c be a 5-cycle and let r_1 and r_2 be two reality-edges in c connected by a desire-edge. If we remove r_1, r_2, and the adjacent desire-edges, and close the remaining cycle with a new desire-edge, then we obtain a new cycle called c' (see Fig. 4 for an example).*

- *There exists a 2_t-move that splits c into a 3-cycle and two adjacencies if and only if there exists a pair of reality-edges r_1, r_2, so that c' is a 3-twisted 3-cycle.*
- *There exists a 2_{tr}-move that splits c into a 3-cycle and two adjacencies if and only if there exists a pair of reality-edges r_1, r_2, so that c' is a 2-twisted 3-cycle.*

Proof An operation can only generate an adjacency if it acts on two reality-edges that are connected by a desire-edge. Therefore, to get two adjacencies, the operation must act on three reality-edges that are connected by desire-edges. We can now simulate every operation

as follows: First, we remove the other two reality-edges (called r_1 and r_2) and their adjacent desire-edges, and close the cycle with a new desire-edge. This corresponds to c'. Now, we perform our operation on c'. As final step, we reinsert r_1 and r_2 into the cycle with the newly created desire-edge. An example can be seen in Fig. 4.
If there exists a pair of reality-edges r_1 and r_2, so that c' is 3-twisted, a transposition will split c' into three adjacencies. If we reinsert r_1 and r_2 into the resulting cycles, we get a 3-cycle and two adjacencies. Conversely, if there is no such pair of reality-edges, there is also no transposition that creates three adjacencies. Therefore, any transposition that acts on three reality-edges that are connected by desire-edges can split c into at most two cycles.
For the 2_{tr}-move, the argumentation is the same, except that a 2_{tr}-move on a 3-cycle requires a 2-twisted 3-cycle. □

Lemma 2.5 *Let c be a t-unoriented 5-cycle and let d be a 3-cycle, such that between each pair of reality-edges of d there is a reality-edge of c. Then a transposition that acts on the reality-edges of d makes c t-oriented, and there is a second move that splits c into a 3-cycle and two adjacencies.*

Proof Let a_1, a_2, and a_3 be the three arcs induced by the reality-edges of d. In one of these arcs (without loss of generality, in a_1) there is only one reality-edge of c. We call this edge c_1. Let c_0 and c_2 be the reality-edges that are directly connected to c_1 by a desire-edge. If c_0 is in arc a_2 and c_2 is in arc a_3 or vice versa, then c_0, c_1, and c_2 form a triple of reality-edges that are connected by desire-edges and all of them are in a different arc induced by the reality-edges of d. If c_0 and c_2 are in the same arc (without loss of generality, in a_2), then one of them must be connected to a reality-edge in a_3 (otherwise, there would be no reality-edge in a_3, a contradiction to our precondition). Again, there is a triple of reality-edges that are connected by desire-edges and all are in different arcs induced by the reality-edges of d. If we remove the remaining two reality-edges and the adjacent desire-edges, and close the cycle with a new desire-edge, we obtain a 3-cycle that must be t-unoriented according to Lemma 2.4. A transposition that acts on the reality-edges of d makes this 3-cycle t-oriented (see Lemma 2.3). Therefore, after the transposition, d fulfills the preconditions of Lemma 2.4 and can be split into a 3-cycle and two adjacencies in the next move. □

Alternative proofs for Lemmata 2.4 and 2.5 can be obtained by using the canonical labeling introduced in [11] and [20].

Lemma 2.6 *(proven in [16]) If a cycle c of length ≥ 2 has a nontwisted chord, then there is another cycle d that intersects with this nontwisted chord of c.*

Two arcs a_1 and a_2 are called *adjacent* if the endpoints of a_1 are connected to the endpoints of a_2 by two chords; see Fig. 5.

Lemma 2.7 *(proven in [20]) Let a_1 and a_2 be two disjoint adjacent arcs in a configuration, so that there is at least one position in the configuration that is not covered by a_1 or a_2. Then there is a cycle with one reality-edge in a_1 or a_2, and another reality-edge that is neither in a_1 nor in a_2.*

Two interleaving 1-twisted 3-cycles form a *1-twisted pair* if their twisted reality-edges are consecutive on the circle. An example of a 1-twisted pair is the left diagram in Fig. 5. Later on (in Lemma 3.17), we will use the fact that a 1-twisted pair has the two adjacent arcs marked in the diagram.

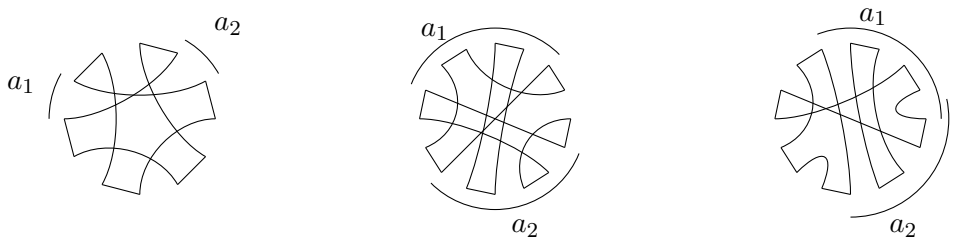

Figure 5: Examples of adjacent arcs. In each diagram, the arcs a_1 and a_2 are adjacent. Note that adjacent arcs can overlap (right diagram).

3 The Algorithm

We begin by introducing a new scoring function that allows us to show a very good lower bound for sorting by weighted reversals, transpositions, and inverted transpositions. Then, we will use the fact that a permutation can be transformed into an equivalent *simple permutation* (defined below) without violating this lower bound. Because the sorting of the original permutation can be mimicked by the sorting of the simple permutation, we merely have to take care of simple permutations.

3.1 A lower bound

It has been proven by Gu et al. [16] that every operation changes the number of odd cycles by at most two. This fact leads to the following lower bound on $d(\pi)$.

Theorem 3.1 *(goes back to [3, 16, 20]) For any permutation $\pi = (\pi_1 \dots \pi_n)$, the inequality $d(\pi) \geq \frac{n - c_{odd}(\pi)}{2}$ holds, where $d(\pi)$ denotes the minimum number of reversals, transpositions, and inverted transpositions required to sort π into the identity permutation.*

For sorting by *weighted* reversals, transpositions, and inverted transpositions, this bound is not good enough because it does not distinguish between the weights of the operations. More precisely, adapting the bound to the weighted case would lead to the bound $w(\pi) \geq \frac{(n - c_{odd}(\pi))w_r}{2}$ because $w_r \leq w_t$. However, the only way how a reversal can increase c_{odd} by two is to split an even cycle into two odd cycles. We will now define a scoring function that treats such a reversal and a transposition splitting one odd cycle into three odd cycles equally.

Definition 3.2 *The score $\sigma(\pi)$ of a permutation π is defined by*

$$\sigma(\pi) = c_{odd}(\pi) + \left(2 - \frac{2w_r}{w_t}\right) c_{even}(\pi)$$

Let op_i be a rearrangement operation. The weight w_i of op_i is defined to be w_r if op_i is a reversal and w_t otherwise. Furthermore, we define $\Delta\sigma_i = \sigma(op_i(\pi)) - \sigma(\pi)$ to be the gain in score after the application of op_i to the permutation π (a negative gain is possible). By analyzing the gain of the score for each possible operation (w.r.t. the cycles they act on and the cycles they produce), one can verify that for each operation op_i, the inequality $\frac{\Delta\sigma_i}{w_i} \leq \frac{2}{w_t}$ holds provided that $w_r \leq w_t \leq 2w_r$. Moreover, for the two operations discussed immediately before Definition 3.2, the inequality becomes an equality.

Lemma 3.3 *For any permutation $\pi = (\pi_1 \dots \pi_n)$ and weights w_r, w_t with $w_r \leq w_t \leq 2w_r$:*

- *$\sigma(\pi) = n$ if π is the identity permutation.*
- *$\sigma(\pi) \leq n - 1$ if π is not the identity permutation.*

Proof If π is the identity permutation, the reality-desire diagram consists of n 1-cycles (adjacencies), so $\sigma(\pi) = c_{odd}(\pi) = n$. Otherwise, the diagram has at least one cycle of length ≥ 2. Therefore, it has at most $n - 1$ cycles. An odd cycle adds 1 to the score, while an even cycle adds $2 - \frac{2w_r}{w_t}$. With $w_t \leq 2w_r$ it follows that $2 - \frac{2w_r}{w_t} \leq 1$. Thus, $\sigma(\pi) \leq n - 1$. □

Theorem 3.4 *For any permutation π and weights w_r, w_t with $w_r \leq w_t \leq 2w_r$, we have*

$$w(\pi) \geq lb(\pi) \text{ where } lb(\pi) = c_{even}(\pi)w_r + \left(\frac{n - c_{odd}(\pi)}{2} - c_{even}(\pi)\right) w_t$$

Proof Let $op_1, op_2, \dots, op_k$ be an optimal sorting sequence of π, i.e., $w(\pi) = \sum_{i=1}^{k} w_i$. We have $\sigma(\pi) + \sum_{i=1}^{k} \Delta\sigma_i = n$ because π is transformed into the identity permutation, which has score n. It follows from $\Delta\sigma_i \leq w_i \frac{2}{w_t}$ that $n \leq \sigma(\pi) + \sum_{i=1}^{k} w_i \frac{2}{w_t} = \sigma(\pi) + w(\pi)\frac{2}{w_t}$. Hence $w(\pi) \geq (n - \sigma(\pi))\frac{w_t}{2} = lb(\pi)$. □

3.2 Transformation into simple permutations

The analysis of cycles of arbitrary length is rather complicated. For this reason, a permutation will be transformed into a so-called simple permutation. A cycle is called *long* if its length is greater than 3. A permutation is called *simple* if it contains no long cycles. According to [18–21], there is a padding algorithm that transforms any permutation π into a simple permutation $\tilde{\pi}$. Each transformation step increases n and c_{odd} by 1, and leaves c_{even} unchanged. Hence $lb(\tilde{\pi}) = lb(\pi)$. As the padding algorithm just adds elements to π, π can be sorted by using a sorting sequence of $\tilde{\pi}$ in which the added elements are ignored. Consequently, the resulting sorting sequence of π has the same or a smaller weight than the sorting sequence of $\tilde{\pi}$. In the next subsection, we will present an algorithm that takes a simple permutation $\tilde{\pi}$ as input and outputs a sorting sequence $op_1, op_2, \dots, op_k$ of $\tilde{\pi}$ such that $\sum_{i=1}^{k} w_i \leq 1.5\ lb(\tilde{\pi})$. Altogether, this yields a 1.5-approximation for sorting by weighted reversals, transpositions, and inverted transpositions because $\sum_{i=1}^{k} w_i \leq 1.5\ lb(\tilde{\pi}) = 1.5\ lb(\pi) \leq 1.5\ w(\pi)$.

Note that it is not possible to transform 2-cycles into 3-cycles as done in [20] because these transformations would change the score and the lower bound.

3.3 The algorithm for simple permutations

Given a simple permutation π, the overall goal is to find a sorting sequence $op_1, op_2, \dots, op_k$ of π such that $\sum_{i=1}^{k} \Delta\sigma_i \geq \sum_{i=1}^{k} w_i \frac{4}{3w_t}$. By a reasoning similar to the proof of Theorem 3.4, it then follows $\sum_{i=1}^{k} w_i \leq 1.5\ lb(\pi)$. To achieve this goal, we search for a "starting sequence" $op_1, \dots, op_j$ of at most four operations (i.e., $1 \leq j \leq 4$) such that $\sum_{i=1}^{j} \Delta\sigma_i \geq \sum_{i=1}^{j} w_i \frac{4}{3w_t}$. This procedure is iterated (i.e., we next search for a starting sequence of $op_j \circ \dots \circ op_1(\pi)$ etc.) until the identity permutation is reached.

The algorithm starts by searching for an arbitrary cycle c of length ≥ 2 in the reality-desire diagram of π. If the cycle is an r-oriented 2-cycle or a t-oriented 3-cycle, the starting

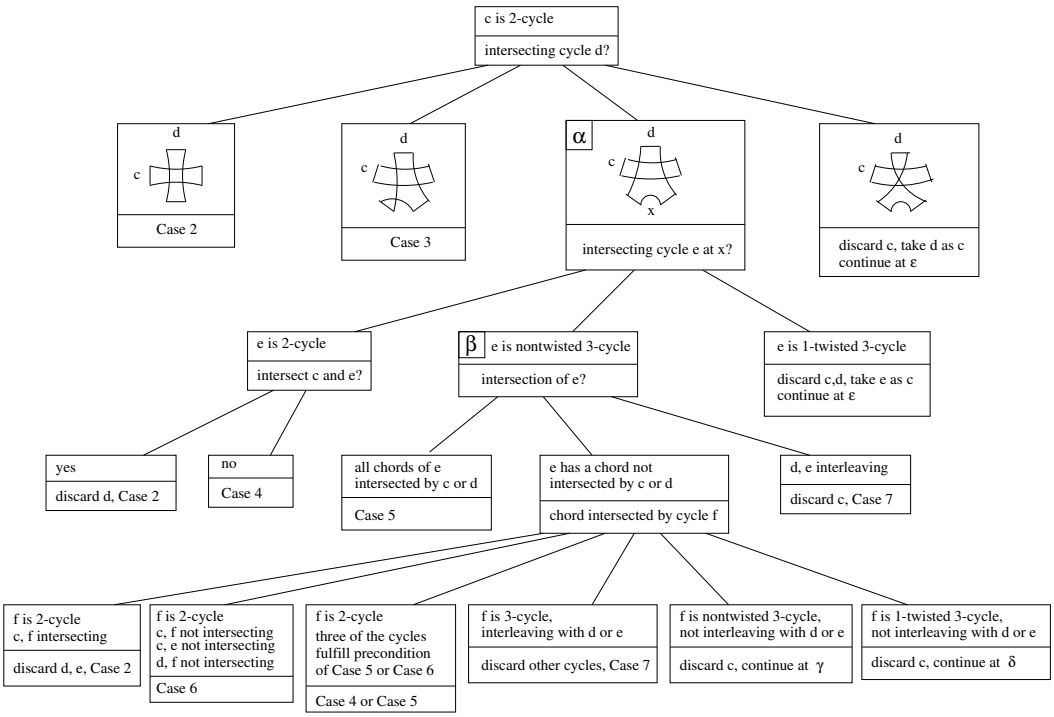

Figure 6: The algorithm's decision tree if it begins with an r-unoriented 2-cycle c. All cycles are considered to be r-unoriented 2-cycles or t-unoriented 3-cycles because r-oriented 2-cycles or t-oriented 3-cycles can directly be eliminated. Cross-references α and β can be found in this figure, γ and δ in Fig. 7, while ε, ζ, and η are in Fig. 8.

sequence can consist solely of the operation op_1 that eliminates this cycle (i.e., op_1 is a 1_r, 2_t or 2_{tr} move that cuts the cycle into 1-cycles). This is because $\frac{\Delta\sigma_1}{w_1} = \frac{2}{w_t} \geq \frac{4}{3w_t}$. Otherwise, according to Lemma 2.6, c must have a nontwisted chord that is intersected by another cycle d. The algorithm now searches for this cycle and examines the configuration of the cycles c and d. Depending on the configuration found, the algorithm either directly outputs a starting sequence that meets the requirements or, again by Lemma 2.6, there must be a chord in the configuration that is intersected by a cycle e that is not yet in the configuration. Consequently, the algorithm searches for this cycle and adds it to the configuration. This goes on until a configuration is found for which a starting sequence can be provided. The algorithm is based on a decision tree that can be found in Figures 6, 7, and 8. Note that every configuration consists of at most four cycles.

We will now list all configurations that have to be considered, i.e., all configurations in the decision tree for which a convenient starting sequence exists. The configurations consisting solely of 3-cycles have already been described in [19] and [20]. A careful inspection of the starting sequences provided there reveals that these sequences also work in our case. However, some of the sequences are only optimal for the unweighted case, and others are rather difficult to implement, so we improved these sequences. For all cases where we use the starting sequences provided in [19] or [20], we will refer to the source of the starting sequences. For all other cases, we will provide our own starting sequences.

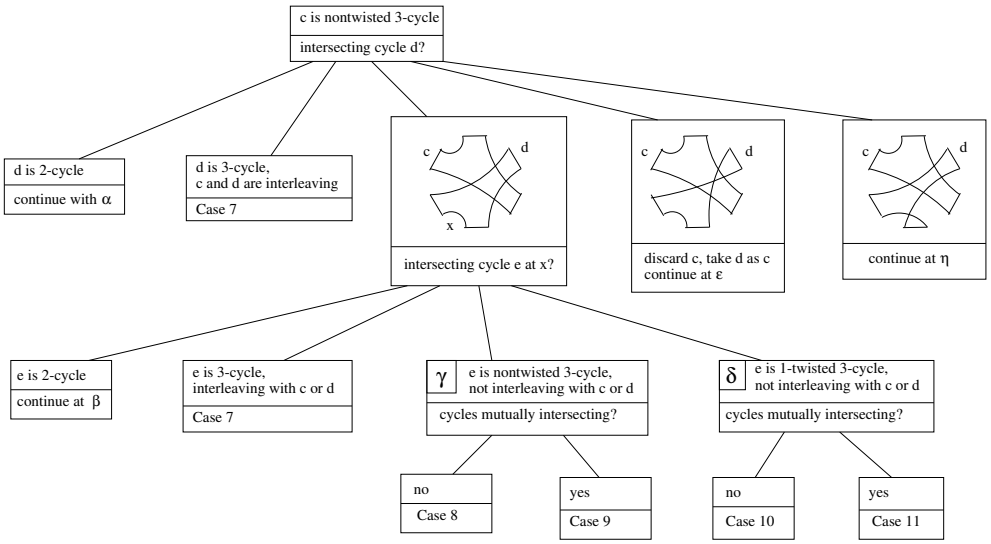

Figure 7: The algorithm's decision tree if it begins with a nontwisted 3-cycle c. All cycles are considered to be r-unoriented 2-cycles or t-unoriented 3-cycles because r-oriented 2-cycles or t-oriented 3-cycles can directly be eliminated. Cross-references α and β can be found in Fig. 6, γ and δ in this figure, while ε, ζ, and η are in Fig. 8.

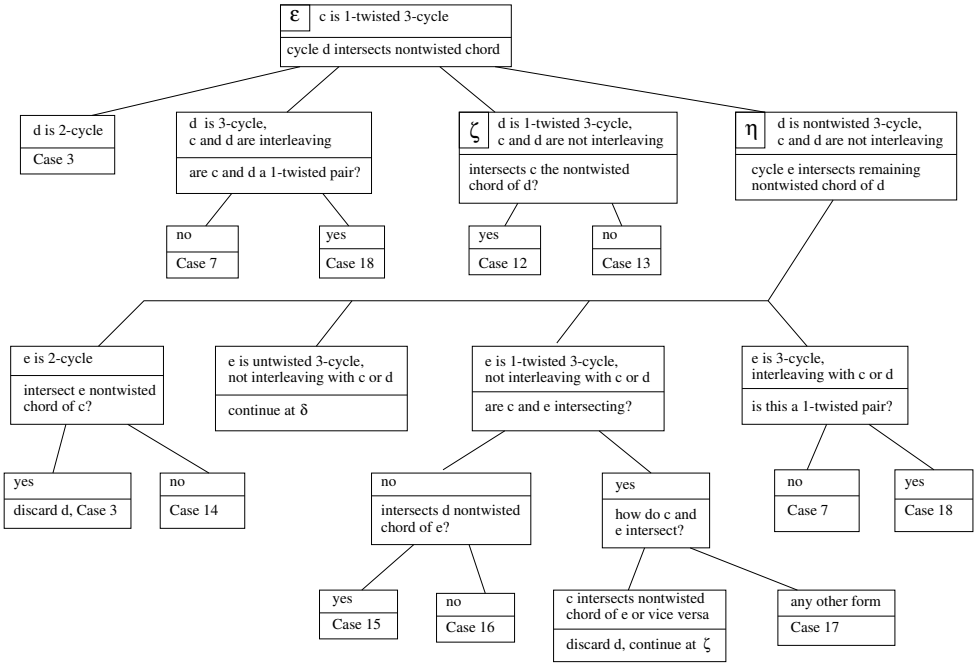

Figure 8: The algorithm's decision tree if it begins with a 1-twisted 3-cycle c. Again, all cycles are r-unoriented 2-cycles or t-unoriented 3-cycles. Cross-references α and β can be found in Fig. 6, γ and δ in Fig. 7, while ε, ζ and η are in this figure.

Case 1 *c is an r-oriented 2-cycle or a t-oriented 3-cycle.*

For the following cases, we assume that all cycles are neither r-oriented 2-cycles nor t-oriented 3-cycles (otherwise we can reduce the case to Case 1).

Case 2 *c and d are two intersecting 2-cycles (example: left configuration of Fig. 9).*

Case 3 *A 2-cycle c intersects the nontwisted chord of a 1-twisted 3-cycle d (example: left configuration of Fig. 10) .*

Case 4 *c and e are 2-cycles, whereas d is a nontwisted 3-cycle. c and e are not intersecting, and each nontwisted chord of d is intersected by c or e (example: left configuration of Fig. 11).*

Case 5 *c is a 2-cycle, whereas d and e are intersecting nontwisted 3-cycles. c intersects the nontwisted chords of d and e that are not intersected by the other 3-cycle (example: left configuration of Fig. 12).*

Case 6 *d and e are two intersecting nontwisted 3-cycles, whereas c and f are 2-cycles. c intersects with the nontwisted chord of d that is not intersected by e, and f intersects with the nontwisted chord of e that is not intersected by d. c and d do not intersect with f, and e does not intersect with c (examples: left configurations of Fig. 13).*

Case 7 *c and d are interleaving 3-cycles, and they do not form a 1-twisted pair (examples: left configurations of Fig. 14).*

Case 8 *c, d, and e are all nontwisted 3-cycles. d is intersecting with both c and e, whereas c and e are not intersecting. The sequences are taken from [19].*

Case 9 *c, d, and e are mutually intersecting nontwisted 3-cycles. The sequences are taken from [19].*

Case 10 *c and d are nontwisted 3-cycles, e is a 1-twisted 3-cycle. c and e are not intersecting (example: left configuration of Fig. 15).*

Case 11 *Two nontwisted 3-cycles c and d and a 1-twisted 3-cycle e are mutually intersecting (examples: left configurations of Fig. 18).*

Case 12 *Two 1-twisted 3-cycles c and d are intersecting, such that the nontwisted chord of each cycle is intersected by the other cycle (examples: left configurations of Fig. 19).*

Case 13 *c and d are two intersecting 1-twisted 3-cycles. d intersects the nontwisted chord of c, but c does not intersect the nontwisted chord of d (examples: Fig. 20).*

Case 14 *c is a 1-twisted 3-cycle, and d is a nontwisted 3-cycle that intersects the nontwisted chord of c. The remaining chord of d (the one not intersected by c) is intersected by a 2-cycle e that does not intersect the nontwisted chord of c (examples: left configurations of Fig. 21).*

Case 15 *c is a nontwisted 3-cycle, d and e are 1-twisted 3-cycles. c intersects with the nontwisted chords of d and e; d and e do not intersect each other. The sequences are taken from [20].*

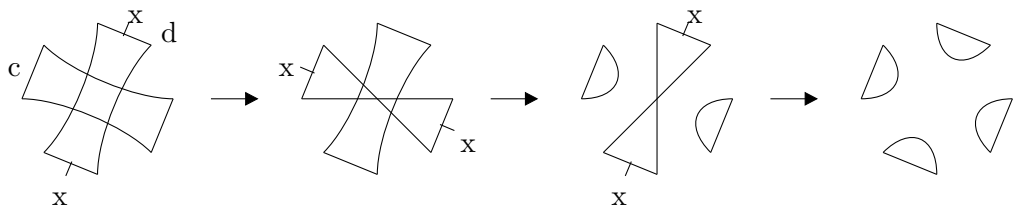

Figure 9: Sequence for Case 2. In this and the following figures, the operations act on the edges marked with x or *. Two x represent a reversal, three * represent a transposition, and two x and a * represent a transreversal that inverts the segment between the two x.

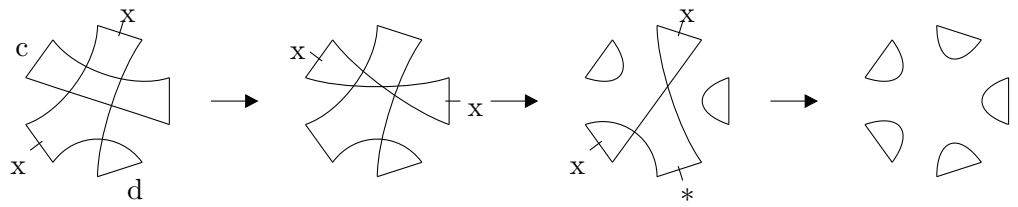

Figure 10: Sequence for Case 3.

Case 16 *c is a nontwisted 3-cycle, d and e are 1-twisted 3-cycles. c intersects with the nontwisted chord of d and the twisted chord of e. d and e do not intersect each other. The sequences are taken from [20].*

Case 17 *c is a nontwisted 3-cycle, d and e are 1-twisted 3-cycles. c intersects with the nontwisted chord of d and an arbitrary chord of e. d and e intersect with their twisted chords, but are not interleaving. The sequences are taken from [20].*

Case 18 *Two 1-twisted 3-cycles c and d form a 1-twisted pair (examples: left configurations of Fig. 22).*

The following lemmata provide the starting sequences that are not taken from [19] or [20].

Lemma 3.5 *For Case 1, there is a 1_r-sequence (if c is an r-oriented 2-cycle), a 2_t-sequence, or a 2_{tr}-sequence (if c is a t-oriented 3-cycle) that can be applied to the permutation.*

Proof This is the trivial case, the sequences consist of just one operation. The existence of these operations follows immediately from the definition of an r-oriented or t-oriented cycle. Note that for these operations, $\frac{\Delta\sigma}{w} = \frac{2}{w_t}$. □

Lemma 3.6 *For Case 2 there exists a $0_r1_r1_r$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = -2$.*

Proof The sequence is described in Fig. 9. Note that $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{3w_t}$. □

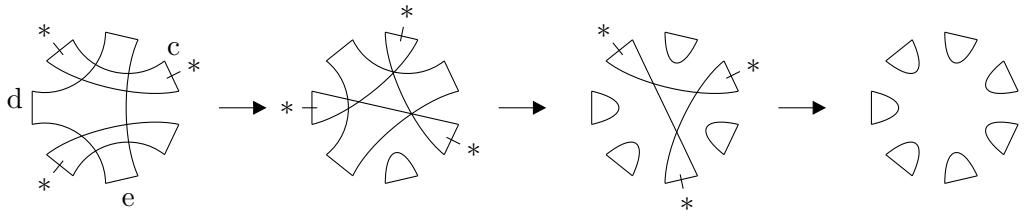

Figure 11: Sequence for Case 4.

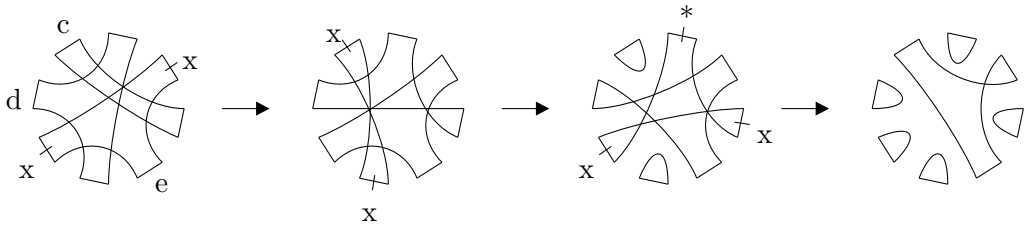

Figure 12: Sequence for Case 5.

Lemma 3.7 *For Case 3, there exists a $1_r1_r1_r$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = -1$.*

Proof The sequence is described in Fig. 10. Note that $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{2w_r+2w_t}{3w_rw_t}$. This value varies from $\frac{4}{3w_t}$ (for $w_t : w_r = 1 : 1$) to $\frac{2}{w_t}$ (for $w_t : w_r = 2 : 1$). □

Lemma 3.8 *For Case 4, there exists a $0_t2_t2_t$-sequence with $\Delta c_{odd} = 6$ and $\Delta c_{even} = -2$.*

Proof The sequence is described in Fig. 11. Note that $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{2}{3w_t} + \frac{4w_r}{3w_t^2}$. This value varies from $\frac{2}{w_t}$ (for $w_t : w_r = 1 : 1$) to $\frac{4}{3w_t}$ (for $w_t : w_r = 2 : 1$). □

Lemma 3.9 *For Case 5, there exists a $0_r1_r2_{tr}$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = -1$.*

Proof The sequence is described in Fig. 12. Note that $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{2w_r+2w_t}{w_t(2w_r+w_t)}$. This value varies from $\frac{4}{3w_t}$ (for $w_t : w_r = 1 : 1$) to $\frac{3}{2w_t}$ (for $w_t : w_r = 2 : 1$). □

Lemma 3.10 *For Case 6, there exists a $0_t2_t2_t2_t$-sequence with $\Delta c_{odd} = 8$ and $\Delta c_{even} = -2$, or a $0_t2_t2_t$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = 0$.*

Proof There are three possible configurations. For each of them, a sequence is described in Fig. 13. For the $0_t2_t2_t2_t$-sequence, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{1}{w_t} + \frac{w_r}{w_t^2}$. This value varies from $\frac{2}{w_t}$ (for $w_t : w_r = 1 : 1$) to $\frac{3}{2w_t}$ (for $w_t : w_r = 2 : 1$). For the $0_t2_t2_t$-sequence, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{3w_t}$. □

Lemma 3.11 *For Case 7, there exists a sequence with weight $w \leq 3w_t$, $\Delta c_{odd} = 4$ and $\Delta c_{even} = 0$.*

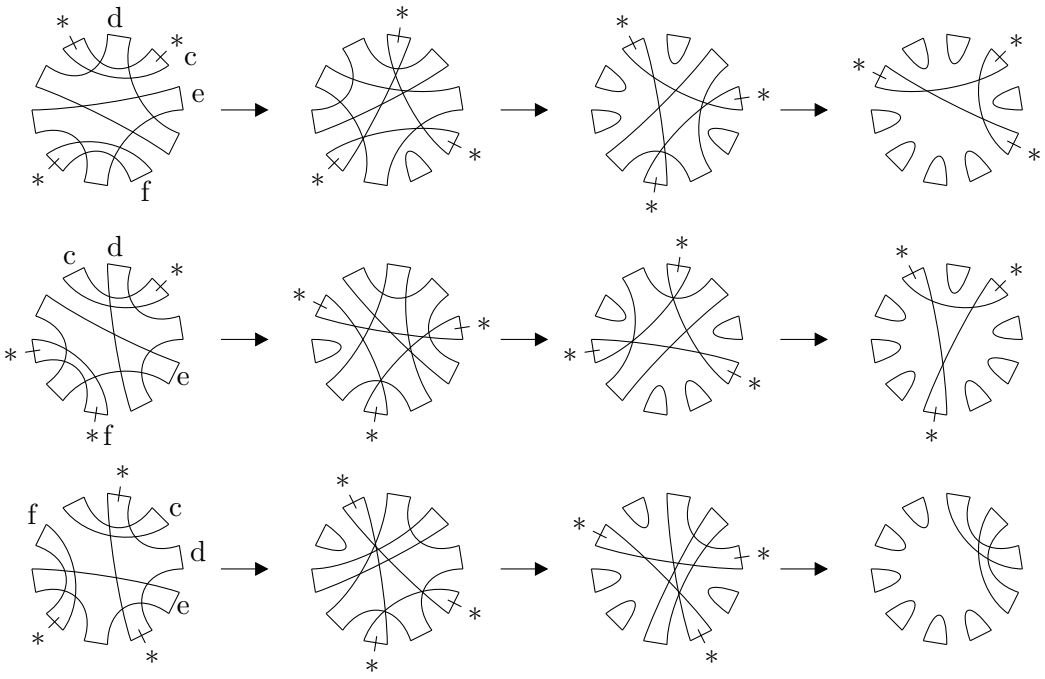

Figure 13: Sequences for Case 6. For the first two sequences, the resulting configurations consist of 10 adjacencies (not drawn in the figure).

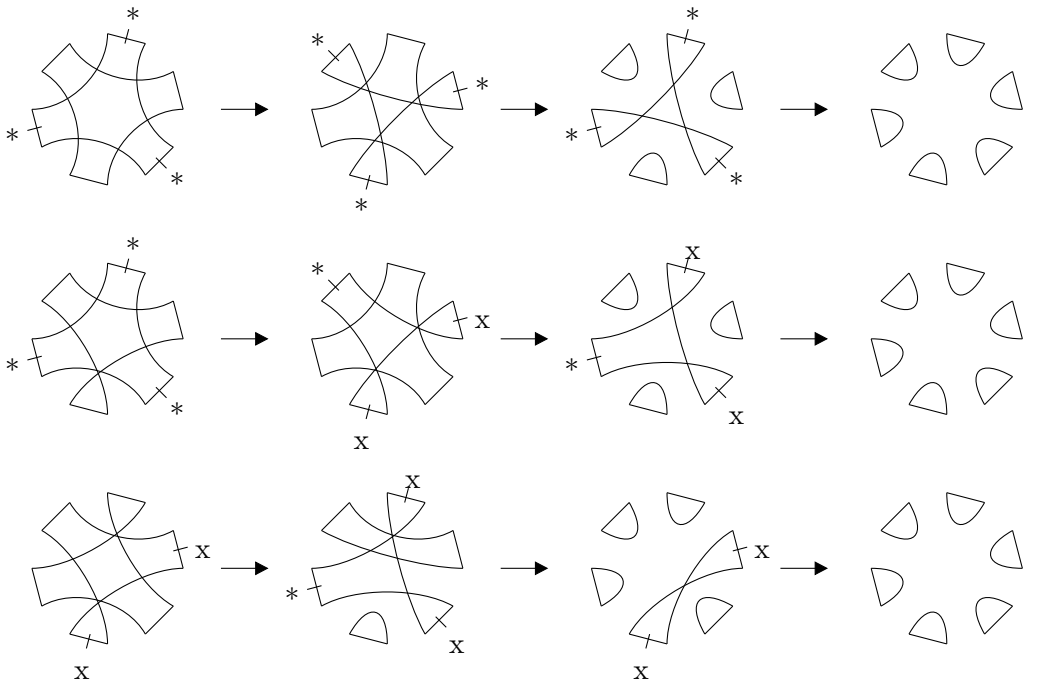

Figure 14: Sequences for Case 7.

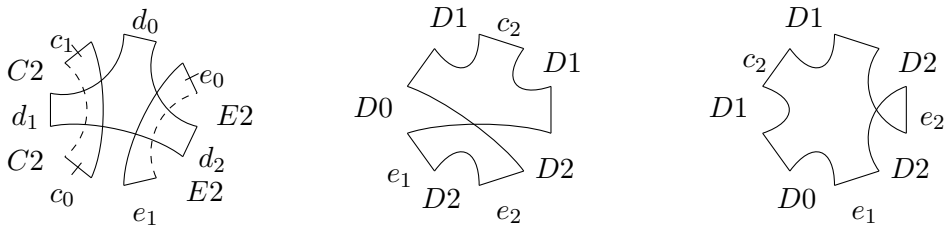

Figure 15: Left: The general case of Case 10. Dashed lines represent two desire-edges separated by a reality-edge. One of the edges e_0 and e_1 also can be twisted. c_2 must be at one of the positions marked with $C2$, and e_2 must be at one of the positions marked with $E2$. Middle: This is the resulting 5-cycle after the first transposition if e_0 was twisted. Right: This is the resulting 5-cycle after the first transposition if e_0 was not twisted. In both pictures, the possible positions for the reality-edges of d are marked with $D0$, $D1$, and $D2$.

Proof There are three cases that can occur. For all of them, the sequence is described in Fig. 14. For the first two sequences, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{3w_t}$. For the last one, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{2w_r+w_t} \geq \frac{4}{3w_t}$. □

The first two of these sequences are taken from [19] and [20]. For the last case, we could improve the sequence by increasing the number of reversals while decreasing the number of transpositions. Note that the sequence described in [20] is optimal for the unweighted case.

Lemma 3.12 *For Case 10, there exists a $0_t2_t2_t$-sequence or a $0_t2_t2_{tr}$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = 0$.*

Proof The general case is illustrated in the left picture of Fig. 15. Dashed lines represent two desire-edges separated by a reality-edge. e_0 or e_1 also can be twisted. Note that c_2 (the third reality-edge of cycle c) must be at one of the positions marked with $C2$, and e_2 (the third reality-edge of cycle e) must be at one of the positions marked with $E2$. The first move is a transposition that acts on c_0, e_0, and c_1, cutting a part of the cycle c and inserting it into e. This move makes d 3-twisted (see Lemma 2.3) and transforms c and e into a 5-cycle and an adjacency. If e_0 was twisted, the configuration of the 5-cycle is the one described in the middle picture of Fig. 15, otherwise it is the one described in the right picture (where also e_2 can be twisted instead of e_1). $D0$, $D1$, and $D2$ mark the possible positions for the reality-edges of cycle d. In any case, the 5-cycle is t-unoriented, and each pair of reality-edges of d is separated by a reality-edge of the 5-cycle. Therefore, the preconditions of Lemma 2.5 are fulfilled, and a transposition that acts on the edges of d (this is a 2_t-move) makes the 5-cycle t-oriented, allowing a second 2_t-move or a 2_{tr}-move. Note that for the resulting sequence, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{3w_t}$. □

This case has also been described by Hartman and Sharan [20]. They start their sequences with the first move of the corresponding sequence of Case 8, such that the first transposition does not act on a twisted edge. However, there is a case where the first move must act on the twisted edge, and this case cannot be solved by using a symmetric configuration, as conjectured in [20].

Lemma 3.13 *For Case 11, there exists a $0_t2_t2_t$-sequence or a $0_t2_t2_{tr}$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = 0$.*

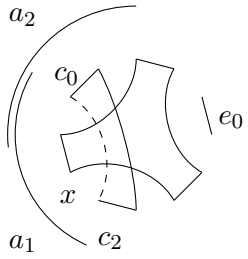

Figure 16: The general situation in Case 11. The dotted line represents two chords separated by the reality-edge c_1. c_1 must be in the arc a_1. At least one of the reality-edges e_1 and e_2 must be at the position marked with x. Cycle e has no reality-edge in arc a_2. For clarity, the chords of cycle e are not drawn in the diagram.

Figure 17: If both chords e_1 and e_2 were at the position marked with x in Fig. 16, the resulting 5-cycle has one of these configurations after the second move. Left: This is the configuration if e_0 was twisted. The dotted line represents two chords separated by the reality-edge c_1, and c_1 must be at one of the positions marked with x. Right: This is the configuration if e_1 was twisted. If e_2 was twisted, the resulting configuration is symmetric.

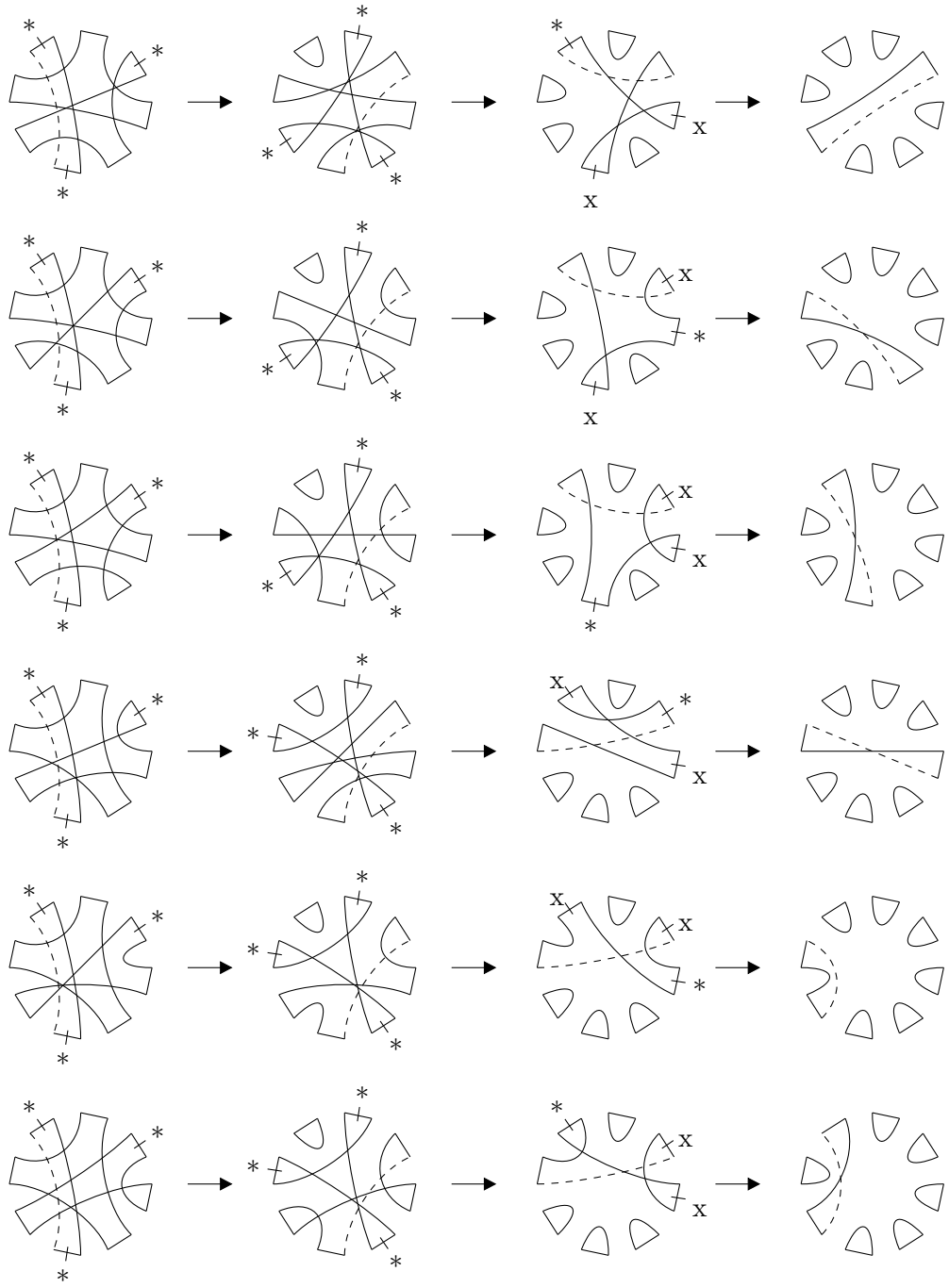

Figure 18: The sequences for the six possible configurations of Case 11 if cycle e has only one reality-edge in the arc between c_0 and c_2. The dotted line represents two chords separated by the reality-edge c_1.

Proof The general situation is illustrated in Fig. 16. The dotted line represents two chords separated by the reality-edge c_1. Because there exist many possibilities for the twist of cycle e, the chords of e are not in the diagram. As c is nontwisted, c_1 must be in arc a_1. At least one of the reality-edges of e is at the position marked with x. Without loss of generality, we can assume that no reality-edge of cycle e is in the arc a_2. The first move is always a transposition that acts on c_0, c_2, and e_0. This makes d 3-twisted, and the second move is the transposition that eliminates d. For the third move, we must distinguish between two cases for the positions of e_1 and e_2 in the starting configuration, which result in different configurations for the 5-cycle after the first transposition:

- Both e_1 and e_2 are at the position marked with x in Fig. 16. If e_0 was twisted, the resulting configuration for the 5-cycle can be seen in the left picture of Fig. 17. Otherwise, the configuration is that of the right picture (shown is the configuration where e_1 was twisted; if e_2 was twisted, the result is symmetric). Before the first move, the reality-edge c_1 must be at one of the positions marked with x in Fig. 16. Note that it cannot be between e_1 and e_2 because c and e are non-interleaving. For both positions, the preconditions of Lemma 2.4 are fulfilled, allowing a 2_{tr}-move.
- If only one of e_1 and e_2 is at the position marked with x in Fig. 16, then there are six possible configurations. For each of them, a sequence is described in Fig. 18.

Note that for the resulting sequence, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{3w_t}$. □

This case has also been handled in [20]. Our lemma provides a more general view of the configuration than the one provided there. This reduces the number of subcases that have to be considered and results in a simpler implementation.

Lemma 3.14 *For Case 12, there exists a* $1_r2_{tr}1_r$*-sequence with* $\Delta c_{odd} = 4$ *and* $\Delta c_{even} = 0$.

Proof There are three possible configurations, and the sequences for these configurations are shown in Fig. 19. Note that for these sequences, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{w_t+2w_r} \geq \frac{4}{3w_t}$. □

This case has also been described in [20]. However, the sequences provided there are only optimal for the unweighted case.

Lemma 3.15 *For Case 13, we can either apply Case 3 or we can use a sequence described in [20].*

Proof There are two possible configurations, as illustrated in Fig. 20. According to Lemma 2.6, the chord marked with an x must intersect with another cycle. If this is a 2-cycle, we can discard the cycle c and apply Case 3. If this is a 3-cycle, we can use a sequence described in [20]. This is either a $0_r2_{tr}2_{tr}$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = 0$ or a $1_r2_{tr}2_{tr}$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = 1$. Note that for these sequences, either $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{2w_t+w_r}$ or $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{6w_t-2w_r}{w_t(2w_t+w_r)}$. The first value varies from $\frac{4}{3w_t}$ (for $w_t : w_r = 1 : 1$) to $\frac{8}{5w_t}$ (for $w_t : w_r = 2 : 1$). The second value varies from $\frac{4}{3w_t}$ (for $w_t : w_r = 1 : 1$) to $\frac{2}{w_t}$ (for $w_t : w_r = 2 : 1$). □

Lemma 3.16 *For Case 14, there is a* $0_t2_t2_t1_r$*-sequence or a* $0_t2_t2_{tr}1_r$*-sequence with* $\Delta c_{odd} = 6$ *and* $\Delta c_{even} = -1$.

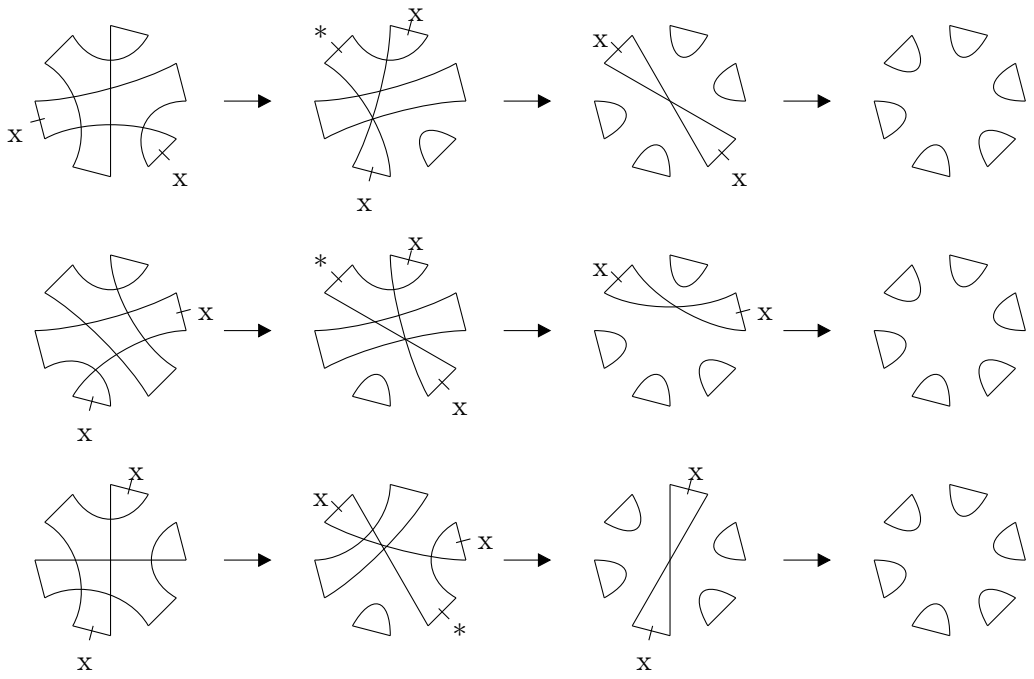

Figure 19: Sequences for Case 12.

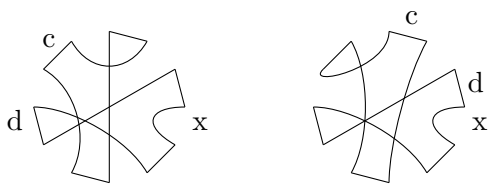

Figure 20: The two possible starting configurations for Case 13. The chord marked with x must intersect with another cycle.

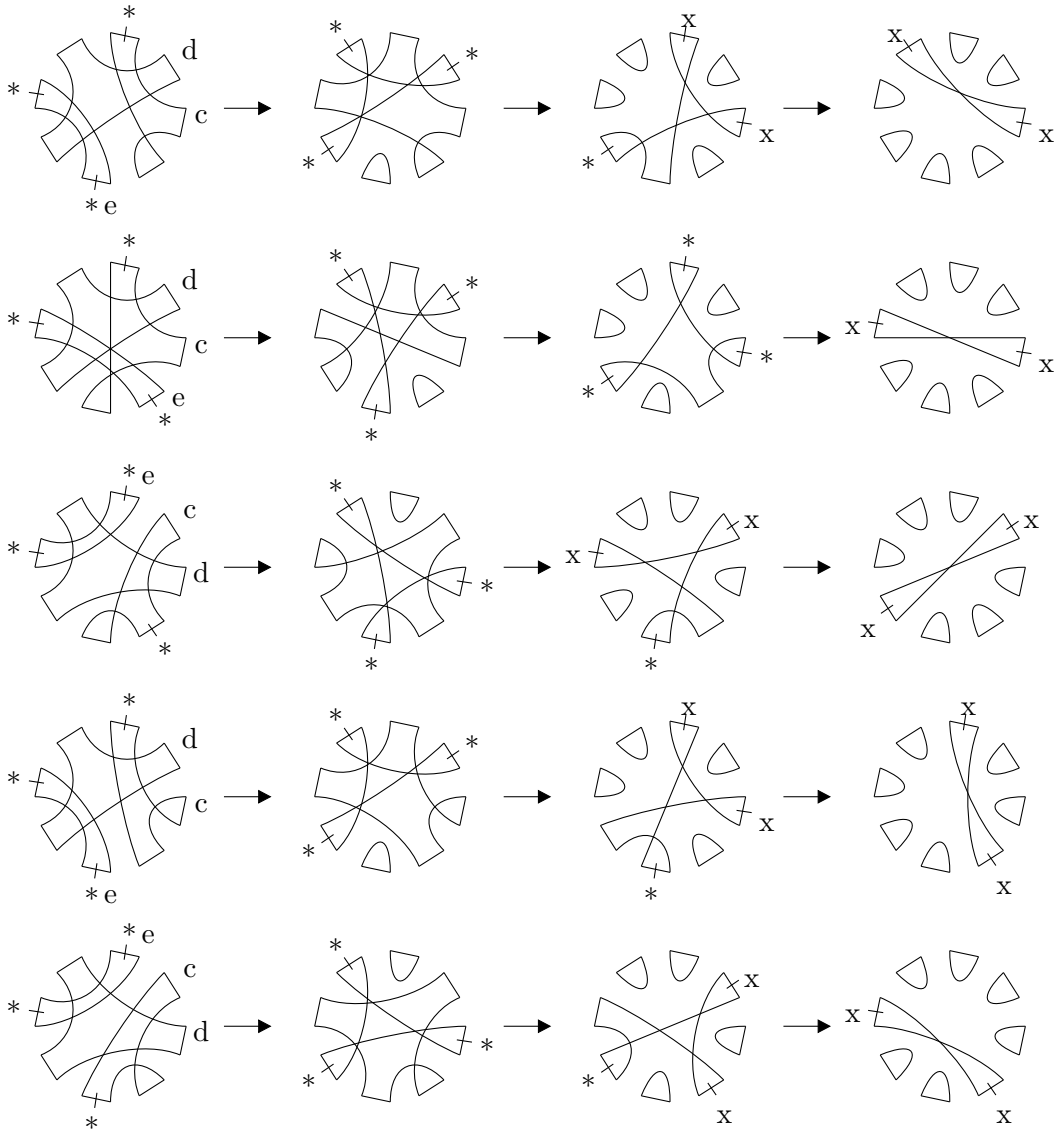

Figure 21: Sequences for Case 14. The resulting configurations consist of 8 adjacencies (not drawn in the figure).

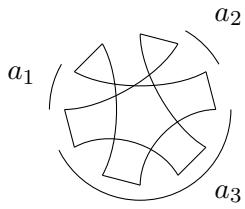

Figure 22: Two 1-twisted 3-cycles form a 1-twisted pair. The arcs a_1 and a_2 are adjacent, so there is a third cycle that has at least one reality-edge in a_1 or a_2, and at least one reality-edge that is neither in a_1 nor in a_2 (see Lemma 2.7). Depending on the position of this edge (in a_3 or between the two twisted reality-edges), we can choose a sequence for this case.

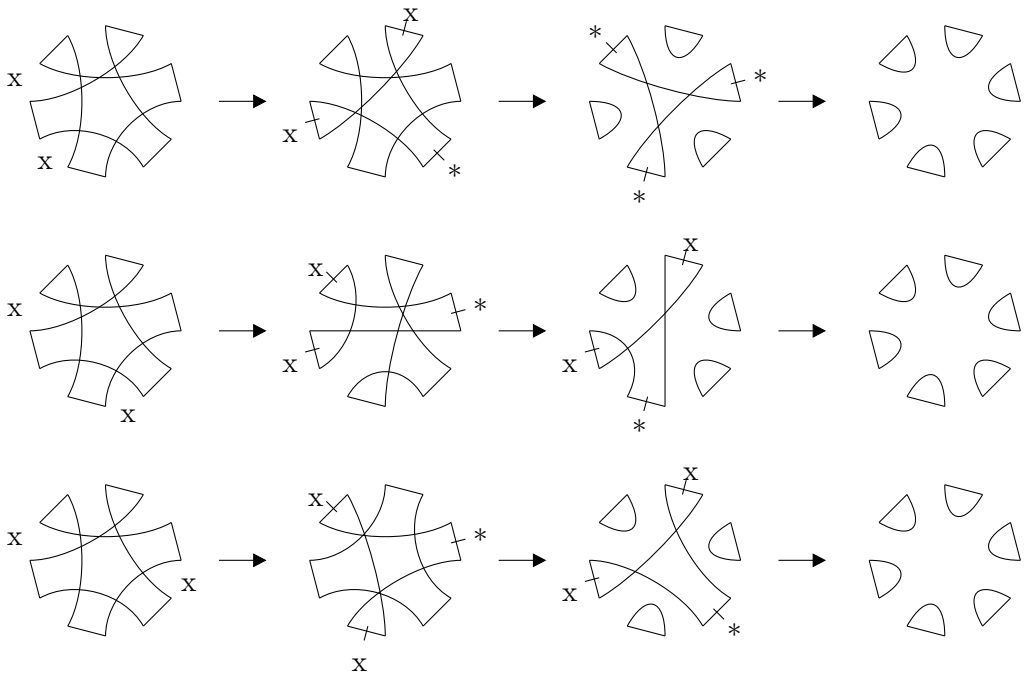

Figure 23: Sequences for eliminating a 1-twisted pair if there is a third cycle e that intersects at least one of the nontwisted chords of the 1-twisted pair. The first move is a reversal that acts on two reality-edges of e, the other two moves act on the reality-edges of the 1-twisted pair.

Proof There are five possible configurations. For all of them, a sequence is described in Fig. 21. Note that for these sequences, we have $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4w_t+2w_r}{w_t(3w_t+w_r)}$. This value varies from $\frac{10}{7w_t}$ (for $w_t : w_r = 2:1$) to $\frac{3}{2w_t}$ (for $w_t : w_r = 1:1$). □

Lemma 3.17 *For Case 18, we can either apply Lemma 3.15, or we can apply a sequence with $w = 2w_t + w_r$ and $\Delta\sigma \geq 4$, or we can apply a $0_{tr}2_t2_t$-sequence or a $0_{tr}2_t2_{tr}$-sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = 0$, or we can apply a $0_r2_{tr}1_r$ sequence with $\Delta c_{odd} = 4$ and $\Delta c_{even} = -1$.*

Proof The starting configuration is illustrated in Fig. 22. The arcs a_1 and a_2 in the figure are adjacent. According to Lemma 2.7, there must be a cycle e that has at least one reality-edge (let this be e_1) in one of the arcs, and at least one reality-edge (let this be e_2) that is in none of these arcs. Without loss of generality, we can assume that e_1 is in the arc a_1. Now, we must distinguish between the possible positions of e_2:

- If e_2 is in the arc a_3, we begin with a reversal that acts on e_1 and e_2. For all of the three possible positions of e_2, a sequence is described in Fig. 23. Each sequence has the weight $w = 2w_t + w_r$. If the first reversal does not split the cycle e, the sequence increases c_{odd} by 4 and leaves c_{even} unchanged, therefore the gain in the score is 4. If the first reversal splits the cycle e, we either get one additional even cycle, or an even cycle will be split into two odd cycles. Both cases increase the score.

- If e_2 is not in the arc a_3, we can assume without loss of generality that e has no reality-edge in a_3. If e is 1-twisted and c or d intersects the nontwisted chord of e, then we can apply Lemma 3.15. Otherwise, there are six configurations we have to consider. For each of them, a sequence is described in Fig. 24.

Note that for the sequences with $w = 2w_t + w_r$ and $\sum \Delta\sigma_i \geq 4$, $\frac{\sum \Delta\sigma_i}{\sum w_i} \geq \frac{4}{2w_t+w_r} \geq \frac{4}{3w_t}$. For the $0_{tr}2_t2_t$-sequence and $0_{tr}2_t2_{tr}$-sequence, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{4}{3w_t}$. For the $0_r2_{tr}1_r$-sequence, $\frac{\sum \Delta\sigma_i}{\sum w_i} = \frac{2w_r+2w_t}{w_t(w_t+2w_r)}$. This value varies from $\frac{4}{3w_t}$ (for $w_t : w_r = 1:1$) to $\frac{3}{2w_t}$ (for $w_t : w_r = 2:1$). □

This case has also been handled in [20]. However, Hartman and Sharan worked with $k \geq 2$ mutually interleaving 3-cycles, such that each pair forms a 1-twisted pair. We can solve this case directly for $k = 2$ without extending the configuration. This leads to an easier and more efficient implementation of this case.

4 Conclusions

We have provided a 1.5-approximation for sorting by weighted reversals, transpositions, and inverted transpositions with a time complexity of $O(n^2)$:

- The initial transformation into a simple permutation can be done in $O(n)$ steps [18]. The same is true for the back-transformation.

- Searching for a cycle that intersects a given desire edge in the reality-desire diagram can be done in $O(n)$ steps. As the depth of the decision tree can be bounded by a constant, the time complexity of finding a starting sequence is also $O(n)$.

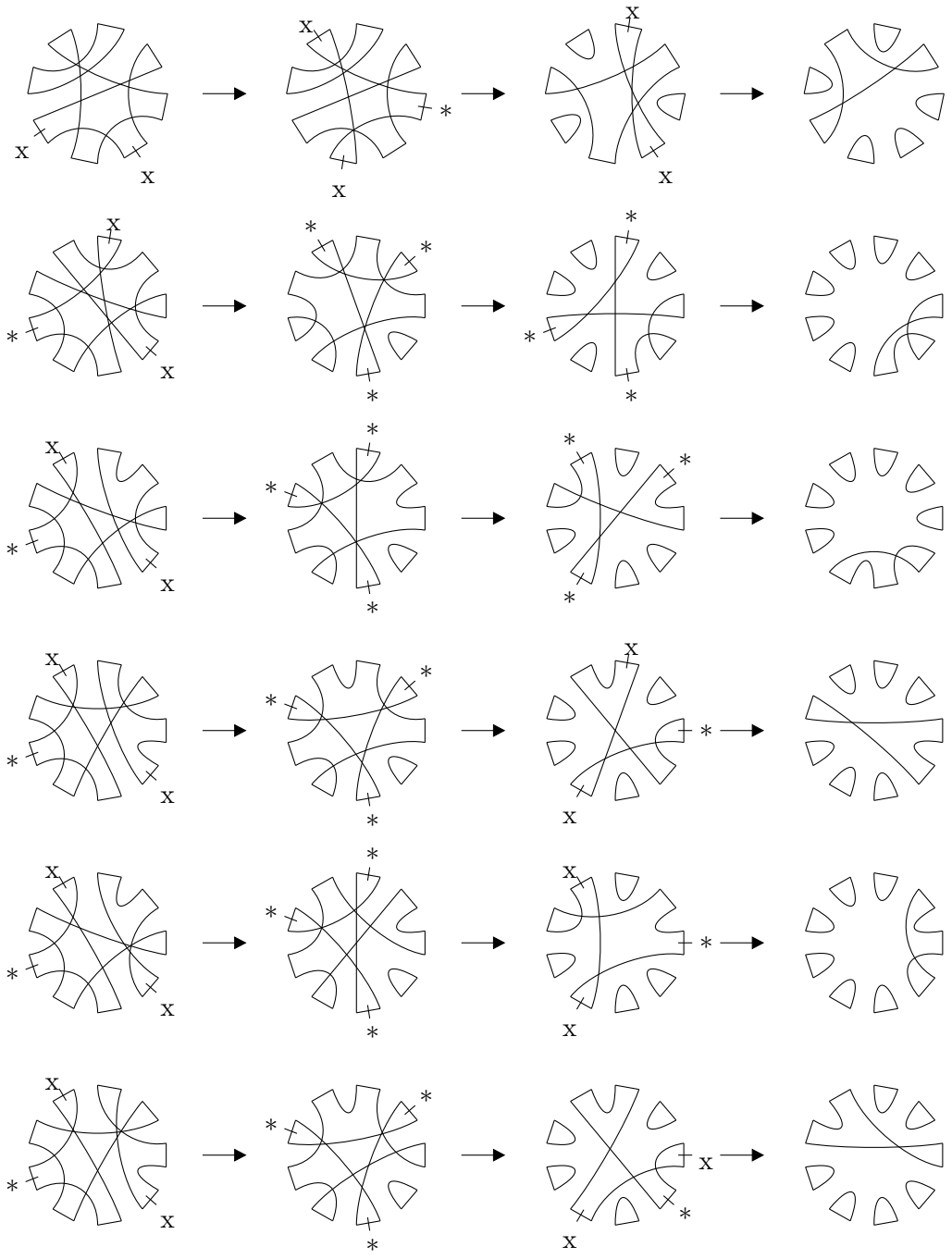

Figure 24: Sequences for two interleaving 3-cycles forming a 1-twisted pair. These are the sequences where we could not find a cycle that intersects one of the nontwisted chords of the 1-twisted pair.

- Each permutation π can be sorted with at most $1.5\frac{lb(\pi)}{w_r} = O(n)$ operations (in the worst case, $w(\pi) = 1.5\ lb(\pi)$ and all operations are reversals). Therefore, finding all necessary starting sequences to sort the permutation has a time complexity of $O(n^2)$.
- Applying an operation to the permutation can be done in $O(n)$ time. This step must be performed at most $O(n)$ times.

In practice, the algorithm can be improved by combining it with a greedy strategy: Instead of beginning with the first cycle in the reality-desire diagram, we start the search at each cycle in the diagram, and use a sequence with the best gain in score. This increases the running time by a factor of n, but the algorithm will find better sorting sequences, and changes in the weight ratio result in different sorting sequences. We have implemented this algorithm in $C++$, and it is available as web application (`http://erde.informatik.uni-ulm.de:3002/metamorphosis`).
Of course, the algorithm works on a very simplified model of biological reality, and the ultimate goal is to also take large-scale insertions, deletions [12] and duplications [13, 24] into account. Moreover, in the future the approach should be extended to multichromosomal genomes, as it has been done with some other biological models [7, 17, 22, 28]. Another area of research is to consider the length of inversions [4, 23].

References

[1] D.A. Bader, B.M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8:483–491, 2001.

[2] V. Bafna and P.A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[3] V. Bafna and P.A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, 1998.

[4] M.A. Bender, D. Ge, S. He, H. Hu, R.Y. Pinter, S. Skiena, and F. Swidan. Improved bounds on sorting with length-weighted reversals. In *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 912–921. ACM, 2004.

[5] A. Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. *Discrete Applied Mathematics*, 146(2):134–145, 2005.

[6] A. Bergeron, J. Mixtacki, and J. Stoye. Reversal distance without hurdles and fortresses. In *Proc. 15th Annual Symposium on Combinatorial Pattern Matching*, volume 3109 of *Lecture Notes in Computer Science*, pages 388–399. Springer-Verlag, 2004.

[7] A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. In *Proc. 6th International Workshop on Algorithms in Bioinformatics*, volume 4175 of *Lecture Notes in Computer Science*, pages 163–173. Springer-Verlag, 2006.

[8] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proc. 10th Annual European Symposium on Algorithms*, volume 2461 of *Lecture Notes in Computer Science*, pages 200–210. Springer-Verlag, 2002.

[9] M. Blanchette, T. Kunisawa, and D. Sankoff. Parametric genome rearrangement. *Gene*, 172:GC11–17, 1996.

[10] A. Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12:91–110, 1999.

[11] D.A. Christie. *Genome Rearrangement Problems.* PhD thesis, University of Glasgow, 1998.

[12] N. El-Mabrouk. Sorting signed permutations by reversals and insertions/deletions of contiguous segments. *Journal of Discrete Algorithms*, 1(1):105–122, 2001.

[13] N. El-Mabrouk. Reconstructing an ancestral genome using minimum segments duplications and reversals. *Journal of Computer and System Sciences*, 65:442–464, 2002.

[14] I. Elias and T. Hartman. A 1.375-approximation algorithm for sorting by transpositions. In *Proc. 5th International Workshop on Algorithms in Bioinformatics*, volume 3692 of *Lecture Notes in Bioinformatics*, pages 204–215. Springer-Verlag, 2005.

[15] N. Eriksen. $(1+\epsilon)$-approximation of sorting by reversals and transpositions. *Theoretical Computer Science*, 289(1):517–529, 2002.

[16] Q.-P. Gu, S. Peng, and H. Sudborough. A 2-approximation algorithms for genome rearrangements by reversals and transpositions. *Theoretical Computer Science*, 210(2):327–339, 1999.

[17] S. Hannenhalli and P.A. Pevzner. Transforming men into mice (polynomial algorithm for genetic distance problem). In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 581–592, 1995.

[18] S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):1–27, 1999.

[19] T. Hartman and R. Shamir. A simpler and faster 1.5-approximation algorithm for sorting by transpositions. *Information and Computation*, 204(2):275–290, 2006.

[20] T. Hartman and R. Sharan. A 1.5-approximation algorithm for sorting by transpositions and transreversals. *Journal of Computer and System Sciences*, 70(3):300–320, 2005.

[21] G.-H. Lin and G. Xue. Signed genome rearrangement by reversals and transpositions: models and approximations. *Theoretical Computer Science*, 259:513–531, 2001.

[22] M. Ozery-Flato and R. Shamir. Two notes on genome rearrangements. *Journal of Bioinformatics and Computational Biology*, 1(1):71–94, 2003.

[23] R.Y. Pinter and S. Skiena. Genomic sorting with length-weighted reversals. In *Genome Informatics 2002*, volume 13, pages 103–111. Universal Academy Press, 2002.

[24] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15:909–917, 1999.

[25] J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology.* PWS Publishing Company, 1997.

[26] E. Tannier and M.-F. Sagot. Sorting by reversals in subquadratic time. In *Proc. 15th Annual Symposium on Combinatorial Pattern Matching*, volume 3109 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2004.

[27] M.E.M.T. Walter, Z. Dias, and J. Meidanis. Reversal and transposition distance of linear chromosomes. In *Proc. Symposium on String Processing and Information Retrieval*, pages 96–102. IEEE Computer Society, 1998.

[28] S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.