A Simpler 1.5-Approximation Algorithm for Sorting by Transpositions

Tzvika Hartman

Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel tzvi@cs.weizmann.ac.il
http://www.cs.weizmann.ac.il/~tzvi

Abstract. An important problem in genome rearrangements is sorting permutations by transpositions. Its complexity is still open, and two rather complicated 1.5-approximation algorithms for sorting linear permutations are known (Bafna and Pevzner, 96 and Christie, 98). In this paper, we observe that the problem of sorting circular permutations by transpositions is equivalent to the problem of sorting linear permutations by transpositions. Hence, all algorithms for sorting linear permutations by transpositions can be used to sort circular permutations. Our main result is a new 1.5-approximation algorithm, which is considerably simpler than the previous ones, and achieves running time which is equal to the best known. Moreover, the analysis of the algorithm is significantly less involved, and provides a good starting point for studying related open problems.

1 Introduction

When trying to determine evolutionary distance between two organisms using genomic data, one wishes to reconstruct the sequence of evolutionary events that have occurred, transforming one genome into the other. One of the most promising ways to trace the evolutionary events is to compare the order of appearance of identical (or orthologous) genes in two different genomes. In the 1980's, evidence was found that different species have essentially the same set of genes, but their order may differ between species [16, 12]. This suggests that global rearrangement events (such as reversals and transpositions of genome segments) can be used to trace the evolutionary path between genomes. Such rare events may provide more accurate and robust clues to the evolution than local mutations (i.e. insertions, deletions, and substitutions of nucleotides).

In the last decade, a large body of work was devoted to genome rearrangement problems. Genomes are represented by permutations, where each element stands for a gene. Circular genomes (such as bacterial and mitochondrial genomes) are represented by circular permutations. The basic task is, given two permutations, to find a shortest sequence of rearrangement operations that transforms one permutation into the other. Assuming that one of the permutations is the identity permutation, the problem is to find the shortest way of sorting a permutation using a given rearrangement operation (or set of operations). For more background on genome rearrangements refer to [19, 17, 18, 20].

The problem of sorting permutations by reversals has been studied extensively. It was shown to be NP-hard [6], and several approximation algorithms have been suggested [2, 7, 5]. On the other hand, for signed permutations (every element of the permutation has a sign, + or -, which represents the direction of the gene), a polynomial algorithm for sorting by reversals was first given by Hannenhalli and Pevzner [11]. Subsequent work improved the running time of the algorithm, and simplified the underlying theory [13, 4, 1]. The problem of sorting signed circular permutations by reversals was shown to be equivalent to the linear case [15].

There has been less progress in studying the problem of sorting by transpositions. A transposition is a rearrangement operation, in which a segment is cut out of the permutation, and pasted in a different location. The complexity of sorting by transpositions is still open. It was first studied by Bafna and Pevzner [3], who devised a rather complicated 1.5-approximation algorithm, which runs in quadratic time. Christie [8] gave a somewhat simpler $O(n^4)$ algorithm with the same approximation ratio. Eriksson et al. [9] provided an algorithm that sorts any given permutation on n elements by at most $2n/3$ transpositions, but has no approximation guarantee. The problem of sorting by both reversals and transpositions was addressed in [21, 10, 14].

In this paper we study the problem of sorting permutations by transpositions. First, we prove that the problem of sorting circular permutations by transpositions is equivalent to the problem of sorting linear permutations by transpositions. Hence, all algorithms for sorting linear permutations by transpositions can be used to sort circular permutations. Then, we derive our main result: A new quadratic 1.5-approximation algorithm, which is considerably simpler than the previous ones [3, 8]. Thus, the algorithm achieves running time which is equal to the best known [3], with the advantage of being much simpler. Moreover, the analysis of the algorithm is significantly less involved, and provides a good starting point for studying related open problems.

The paper is organized as follows. In Section 2 we first prove the equivalence between the problem of sorting linear and circular permutatiosn by transpositions. Then, we review some classical genome rearrangement results, and show that every permutation can be transformed into a so-called simple permutation. Our main result, a new and simple 1.5-approximation algorithm for sorting permutations by transpositions, is introduced in Section 3. We conclude with a short discussion and some open problems (Section 4).

2 Preliminaries

2.1 Linear and Circular Permutations

Let $\pi = [\pi_1 \ \dots \ \pi_n]$ be a permutation on n elements. Define a *segment* A in π as a consecutive sequence of elements $\pi_i, \dots, \pi_k$ $(k \geq i)$. Two segments $A = \pi_i, \dots, \pi_k$ and $B = \pi_j, \dots, \pi_l$ are *contiguous* if $j = k+1$ or $i = l+1$. A *transposition* τ on π is the exchange of two disjoint contiguous segments (Figure 1a). If the segments are

$A = \pi_i, \ldots, \pi_{j-1}$ and $B = \pi_j, \ldots, \pi_{k-1}$, then by performing τ on π, the resulting permutation, denoted $\tau \cdot \pi$, is $[\pi_1 \ \ldots \ \pi_{i-1} \ \pi_j \ \ldots \ \pi_{k-1} \ \pi_i \ \ldots \ \pi_{j-1} \ \pi_k \ \ldots \ \pi_n]$ (note that the end segments can be empty if $i = 1$ or $k - 1 = n$). We shall say that τ *cuts* π *before* positions i, j and k. We say that τ *involves* index l if $i \leq l < k$, i.e., if l belongs to one of the two exchanged segments.

In circular permutations, one can define analogously a transposition as the exchange of two contiguous segments. Note that here the indices are cyclic, so the disjointness of the exchanged segments is a meaningful requirement. The transposition partitions a circular permutation into three segments, as opposed to at most four in a linear permutation (see Figure 1). Since there are only two cyclic orders on three segments, and each two of the three segments are contiguous, the transposition can be represented by exchanging any two of them. Note that the number of possible transpositions on a linear n-permutation is $\binom{n+1}{3}$, since there are $n+1$ possible cut points of segments. In contrast, in a circular n-permutation there are only $\binom{n}{3}$ possibilities.

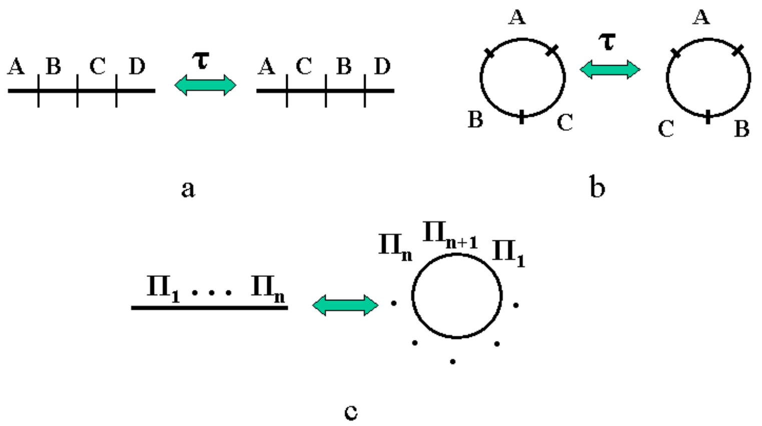

Fig. 1. (a) A transposition τ, which is applied on a linear permutation, and exchanges segments B and C. (b) A transposition τ, which is applied on a circular permutation. τ can be viewed as exchanging A and B, or B and C, or A and C. (c) A one-to-one transformation between linear and circular permutations. In the circular permutation, a new element, π_{n+1}, is introduced.

The problem of finding a shortest sequence of transpositions, which transforms a (linear or circular) permutation into the identity permutation, is called *sorting by transpositions*. The *transposition distance* of a permutation π, denoted by $d(\pi)$, is the length of the shortest sorting sequence.

Theorem 1 *The problem of sorting linear permutations by transpositions is linearly equivalent to the problem of sorting circular permutations by transpositions.*

Proof. Given a linear n-permutation, circularize it by adding an $n+1'$st element $\pi_{n+1} = x$, and closing the circle (see Figure 1c). Call the new circular permutation π^c. By the discussion above, any transposition on π^c can be represented by the two segments that do not include x. Hence, there is an optimal sequence of transpositions that sorts π^c, and none of them involves x. The same sequence can be viewed as a sequence of transpositions on the linear permutation π, by ignoring x. This implies that $d(\pi) \leq d(\pi^c)$. On the other hand, any sequence of transpositions on π is also a sequence of transpositions on π^c, so $d(\pi^c) \leq d(\pi)$. Hence, $d(\pi) = d(\pi^c)$. Moreover, an optimal sequence for π^c provides an optimal sequence for π.

For the other direction, starting with a circular permutation, we can linearize it by removing an arbitrary element, which plays a role of x above (see Figure 1c). The same arguments imply that an optimal solution for the linear permutation translates to an optimal solution for the circular one.

In the rest of the paper, we will discuss only circular permutations. As implied by Theorem 1, all the results on circular permutations hold also for linear ones. We prefer to work with circular permutations since it simplifies the analysis.

2.2 The Breakpoint Graph

We transform a permutation π on n elements into a permutation $f(\pi)$ on $2n$ elements, by replacing each element i by two elements $2i-1, 2i$. On the doubled permutation $f(\pi)$, we allow only transpositions that cut before odd positions. This ensures that no transposition cuts between $2i-1$ and $2i$, and therefore every transposition on π can be mimicked by a transposition on $f(\pi)$. We call such transposition *legal*. We now define the circular breakpoint graph, which is the circular version of the breakpoint graph [2]. Throughout, in both indices and elements, we identify $2n+1$ and 1.

Definition 1 *Let $\pi = (\pi_1 \ \dots \ \pi_n)$ be a circular permutation, and $f(\pi) = \pi' = (\pi'_1 \ \dots \ \pi'_{2n})$. The breakpoint graph $G(\pi)$ is an edge-colored graph on $2n$ vertices $\{1, 2, \dots, 2n\}$. For every $1 \leq i \leq n$, π'_{2i} is joined to π'_{2i+1} by a black edge (denoted by b_i), and $2i$ is joined to $2i+1$ by a gray edge.*

Note that unlike previous studies of transpositions [3, 8], we chose to double the number of vertices and work with an undirected graph, as done in the signed case [2]. It is convenient to draw the breakpoint graph on a circle, such that black edges are on the circumference and gray edges are chords (see Figure 2). We shall use this representation throughout the paper.

Since the degree of each vertex is exactly 2, the graph uniquely decomposes into cycles. Denote the number of cycles in $G(\pi)$ by $c(\pi)$. The length of a cycle is the number of black edges it contains. A *k-cycle* is a cycle of length k, and it is *odd* if k is odd. The number of odd cycles is denoted by $c_{odd}(\pi)$. Define $\Delta c(\pi, \tau) = c(\tau \cdot \pi) - c(\pi)$, and $\Delta c_{odd}(\pi, \tau) = c_{odd}(\tau \cdot \pi) - c_{odd}(\pi)$.

Bafna and Pevzner proved the following useful lemma (This - and other results we quote - was proved for linear permutations, but holds also for circular ones):

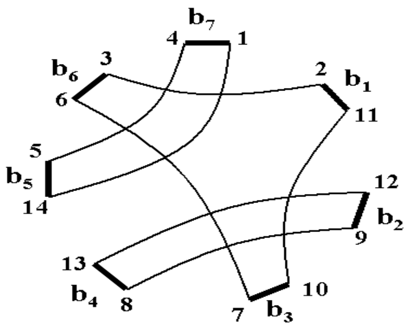

Fig. 2. The circular breakpoint graph of the permutation $\pi = (1\ 6\ 5\ 4\ 7\ 3\ 2)$, for which $f(\pi) = (1\ 2\ 11\ 12\ 9\ 10\ 7\ 8\ 13\ 14\ 5\ 6\ 3\ 4)$. Black edges are represented as thick lines on the circumference, and gray edges are chords.

Lemma 2 (Bafna and Pevzner [3]) *For all permutations π and transpositions τ, it holds that $\Delta c(\pi, \tau) \in \{-2, 0, 2\}$, and $\Delta c_{odd}(\pi, \tau) \in \{-2, 0, 2\}$.*

Let $n(\pi)$ denote the number of black edges in $G(\pi)$. The maximum number of cycles is obtained iff π is the identity permutation. In that case, there are $n(\pi)$ cycles, and all of them are odd (in particular, they are all of length 1) . Starting with π with c_{odd} odd cycles, Lemma 2 implies the following lower bound on $d(\pi)$:

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

By definition, every legal transposition must cut three black edges. The transposition that cuts black edges b_i, b_j and b_k is said to *act on* these edges. A transposition τ is a *k-transposition* if $\Delta c_{odd}(\pi, \tau) = k$. A cycle is called *oriented* if there is a 2-transposition that acts on three of its black edges; otherwise, it is *unoriented.*

Observation 4 *There are only two possible configurations of 3-cycles that can be obtained by legal transpositions.*

The two possibilities are shown in Figure 3. It is easy to verify that the left 3-cycle is unoriented, and the right one is oriented.

Given a cyclic sequence of elements $i_1, \ldots, i_k$, an *arc* is an interval in the cyclic order, i.e., a set of contiguous elements in the sequence. The pair (i_j, i_l) $(j \neq l)$ defines two disjoint arcs: $i_j, \ldots, i_{l-1}$ and $i_l, \ldots, i_{j-1}$. Similarly, a triple defines a partition of the cycle into three disjoint arcs. We say that two pairs of black edges (a, b) and (c, d) are *intersecting* if a and b belong to different arcs of the pair (c, d). A pair of black edges intersects with cycle C, if it intersects with a pair of black edges that belong to C. Cycles C and D intersect if there is a pair of black edges in C that intersect with D (see Figure 4a). Triples of black

Fig. 3. The only two possible configurations of 3-cycles. The left one is unoriented, and the right one is oriented.

a b

Fig. 4. (a) A pair of intersecting 3-cycles. (b) A pair of interleaving 3-cycles.

edges are *interleaving* if each of the edges of one triple belongs to a different arc of the second triple. Two 3-cycles are interleaving if their edges interleave (see Figure 4b).

Throughout the paper, we use the term permutation also when referring to the breakpoint graph of the permutation (as will be clear from the context). For example, when we say that π contains an oriented cycle, we mean that $G(\pi)$ contains an oriented cycle.

2.3 Transformation into Equivalent Simple Permutations

A k-cycle in the breakpoint graph is called *short* if $k \leq 3$; otherwise, it is called *long*. A breakpoint graph is called *simple* if it contains only short cycles. A permutation π is called *simple* if $G(\pi)$ is simple. Following [11, 14], we show how to transform an arbitrary permutation into a simple one, while maintaining the lower bound of Theorem 3.

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 $G(\pi)$ is a sequence of operations on $G(\pi)$, resulting in a new graph $\hat{G}(\pi)$ with one more cycle, as follows:

- Removing edges b and g.
- Adding two new vertices v and w.
- Adding two new black edges (v_b, v) and (w, w_b).
- Adding two new gray edges (w_g, w) and (v, v_g).

Figure 5 shows a (g, b)-split transforming a cycle C in $G(\pi)$ into two cycles C_1 and C_2 in $\hat{G}(\pi)$. Note that the order of the nodes of each edge along the cycle

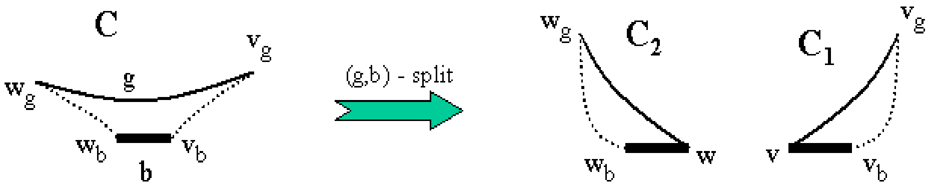

Fig. 5. A (g, b)-split. A dashed line indicates a path.

is important, as other orders may not split the cycle. Hannenhalli and Pevzner [11] show that for every (g, b)-split on a permutation π of n elements, there is a permutation $\hat{\pi}$ of $n+1$ elements, which is obtained by inserting an element into π, such that $\hat{G}(\pi) = G(\hat{\pi})$. Thus, a (g, b)-split can be viewed as a transformation from π to $\hat{\pi}$. A (g, b)-split is called *safe* if $n(\pi) - c_{odd}(\pi) = n(\hat{\pi}) - c_{odd}(\hat{\pi})$, i.e., if it maintains the lower bound of Theorem 3.

Lemma 5 (Lin and Xue [14]) *Every permutation can be transformed into a simple one by safe splits.*

Proof. Let π be a permutation that contains a long cycle C. Let b_1 be a black edge in C. Denote by b_2 and b_3 the black edges that are connected to b_1 via a gray edge. Let g be the gray edge that is connected to b_2 but not to b_1. Then a (g, b_3)-split breaks C into a 3-cycle and a $(k-2)$-cycle in $\hat{\pi}$. Clearly, $n(\hat{\pi}) = n(\pi) + 1$, and $c_{odd}(\hat{\pi}) = c_{odd}(\pi) + 1$, so the split is safe. This process can be repeated until a simple permutation is eventually obtained.

We say that permutation π is *equivalent* to permutation $\hat{\pi}$ if $n(\pi) - c_{odd}(\pi) = n(\hat{\pi}) - c_{odd}(\hat{\pi})$.

Lemma 6 (Hannenhalli and Pevzner [11]) *Let $\hat{\pi}$ be a simple permutation that is equivalent to π, then every sorting of $\hat{\pi}$ mimics a sorting of π with the same number of operations.*

In the following, we show how to sort a simple permutation by transpositions. We prove that the number of transpositions is within a factor of 1.5 from the lower bound of Theorem 3. Thus, we obtain a 1.5-approximation algorithm for sorting simple permutations. The above discussion implies that this algorithm translates into a 1.5-approximation algorithm for an arbitrary permutation: Transform the permutation into an equivalent simple permutation (Lemma 5), sort it, and then mimic the sorting on the original permutation (Lemma 6).

3 The Algorithm

In this section we provide a 1.5-approximation algorithm for sorting permutations by transpositions. We first develop an algorithm for simple permutations, and then use the results of Section 2.3 to prove the general case. Recall that the breakpoint graph of a simple permutation contains only 1-, 2- and 3-cycles. Our goal is to obtain a graph with only 1-cycles, which is the breakpoint graph of the identity permutation. Thus, the sorting can be viewed as a process of transforming the 2- and 3-cycles into 1-cycles.

First we deal with the case that the permutation contains a 2-cycle:

Lemma 7 (Christie [8]) *If π is a permutation that contains a 2-cycle, then there exists a 2-transposition on π.*

By definition, an oriented 3-cycle can be eliminated by a 2-transposition that acts on its black edges. Suppose from now on that all 2-cycles were eliminated by applying Lemma 7, and all oriented 3-cycles were eliminated. The only remaining problem is how to handle unoriented 3-cycles. This is the case we analyze henceforth.

A *(0,2,2)-sequence* is a sequence of three transpositions, of which the first is a 0-transposition, and the next two are 2-transpositions. Note that a $(0,2,2)$-sequence increases the number of odd cycles by 4 out of 6 that are the maximum possible in 3 steps, and thus a series of $(0,2,2)$-sequences preserves a 1.5 approximation ratio. We shall show below that such a sequence is always possible.

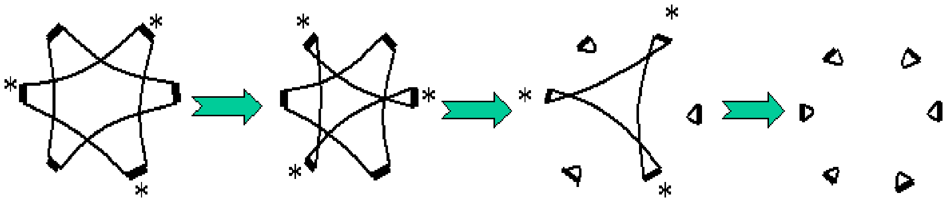

Fig. 6. A $(0,2,2)$-sequence of transpositions for two interleaving unoriented 3-cycles. At each step the transposition acts on the three black edges that are marked by a star.

Lemma 8 *Let π be a permutation that contains two interleaving unoriented 3-cycles. Then, there exists a $(0,2,2)$-sequence of transpositions on π.*

Proof. The $(0,2,2)$-sequence is described in Figure 6.

Lemma 9 *Let C and D be two intersecting unoriented 3-cycles that are not interleaving. Then, there exists a transposition which transforms C and D into a 1-cycle and an oriented 5-cycle.*

Proof. Let c_1, c_2 and c_3 be the three black edges of C. Assume, without loss of generality, that (c_1, c_2) intersects with D. We shall in fact prove a stronger statement, namely, for any choice of a black edge $d \in D$ such that (d, c_3) intersects

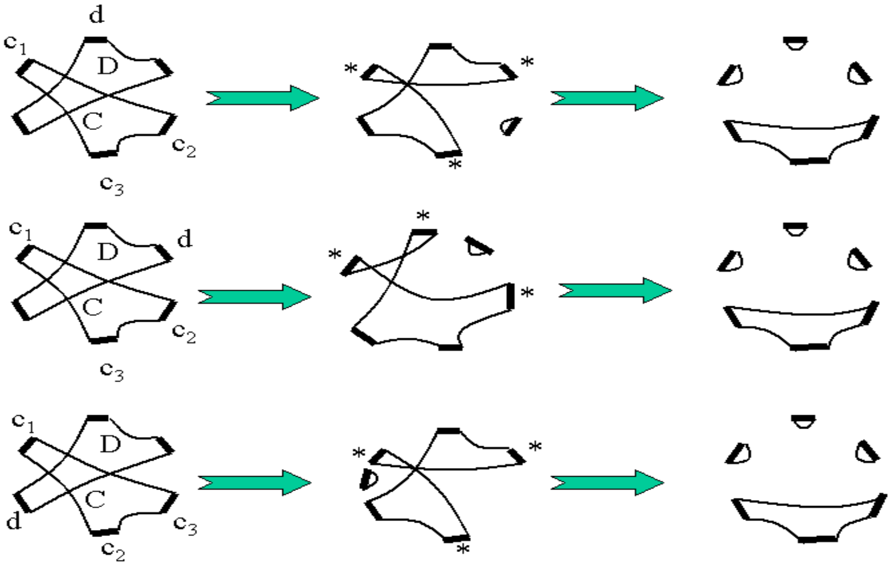

Fig. 7. The three possible cases of two intersecting unoriented 3-cycles that are not interleaving. In each case, the transposition that acts on edges c_1, c_2, and d, transforms C and D into a 1-cycle and an oriented 5-cycle.

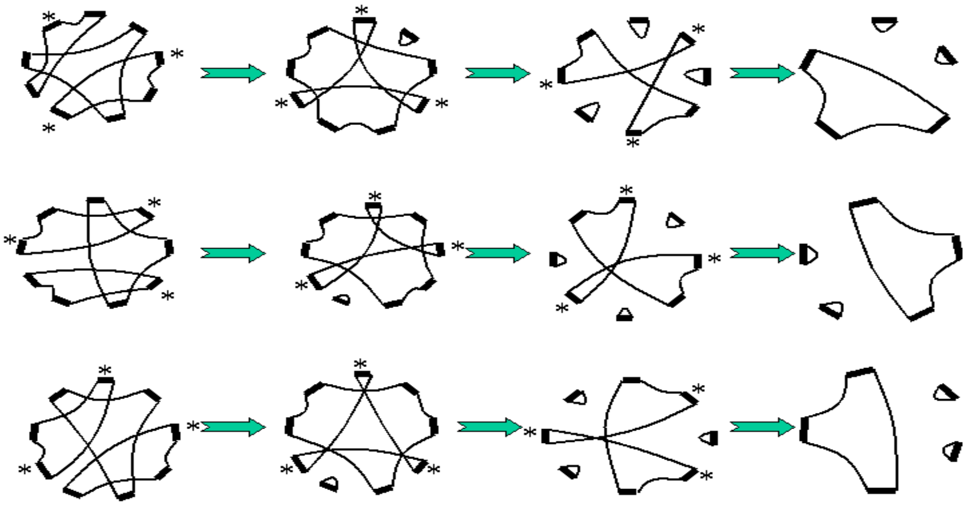

Fig. 8. The three possible cases of three intersecting unoriented 3-cycles, such that no pair is interleaving and two out of the three are non-intersecting. In each case, a $(0, 2, 2)$-sequence of transpositions is shown. For simplicity, every 1-cycle is shown only when it is formed and not in subsequent graphs (since it is not affected by transpositions in later steps).

with (c_1, c_2), the transposition on c_1, c_2 and d satisfies the lemma. Depending on the number of black edges of D in the arc of (c_1, c_2) that avoids c_3, there are three possible cases to consider for d, which are shown in Figure 7. In each case, the first transposition, which acts on c_1, c_2 and d, transforms 3-cycles C and D into a 1-cycle and a 5-cycle. Then, in order to show that the 5-cycle is oriented, a 2-transposition which acts on three edges of its edges is shown.

The following lemma claims that every pair of black edges in an unoriented cycle must intersect with some other cycle.

Lemma 10 (Bafna and Pevzner [3]) *Let C be an unoriented cycle in $G(\pi)$, and let b_i and b_j be black edges of C. Then, (b_i, b_j) intersects with some cycle in $G(\pi)$.*

We say that cycle E is *shattered* by cycles C and D if every pair of edges in E intersects with a pair of edges in C or with a pair of edges in D.

Corollary 11 *Let π be a simple permutation that contains no 2-cycles, no oriented 3-cycles, and no pair of interleaving 3-cycles. Then, there exists a triple of 3-cycles C, D and E in $G(\pi)$, such that E is shattered by C and D.*

Proof. Let E be an unoriented 3-cycle in $G(\pi)$, which consists of the black edges b_1, b_2 and b_3. By Lemma 10, there is a cycle C that intersects with the pair (b_1, b_2). This means that cycle C intersects also, without loss of generality, with the pair (b_2, b_3). Since E and C are not interleaving, there is another cycle D, that intersects with the pair (b_1, b_3) (again by Lemma 10). Hence, E is shattered by C and D.

Lemma 12 *Let π be a permutation that contains three unoriented 3-cycles C, D and E, such that E is shattered by C and D. Then, there exists a $(0, 2, 2)$-sequence of transpositions on π.*

Proof. If two out of the three cycles are interleaving, the (0,2,2)-sequence follows from Lemma 8. Otherwise, there are two general cases:

1. Two out of the three cycles are non-intersecting. In this case, there are three possible configurations of the cycles, which are shown in Figure 8. For every sub-case, a $(0, 2, 2)$-sequence is shown.
2. The three cycles are mutually intersecting. The general case is illustrated in Figure 9. Since cycles C and D are unoriented, the condition of the proof of Lemma 9 is fulfilled. Thus, we can apply a 0-transposition that acts on edges c_1, c_2, and d, and obtain a new oriented cycle F. Now we apply a 2-transposition on E (which has also become oriented). Cycle F remains oriented, since the latter transposition does not change its structure. Thus, another 2-transposition is possible on the edges of F, which completes the $(0, 2, 2)$-sequence.

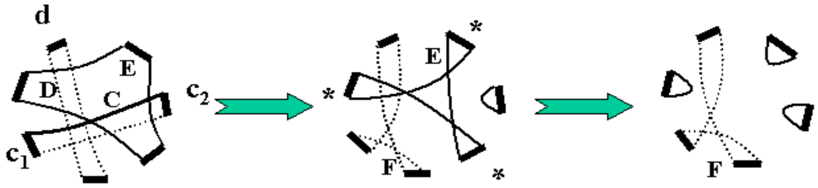

Fig. 9. Three mutually intersecting unoriented cycles. A dashed line represents a path of length 1 or 3. Note that edges c_1 and c_2 are connected by a single gray edge.

Our algorithm is described in Figure 10. Note that none of the steps 2 - 5 create a long cycle, and thus the permutation remains simple throughout the algorithm. Note also that in steps 4 and 5 we do not create 2-cycles or oriented 3-cycles, and hence, there is no need to iterate over steps 2 and 3. Corollary 11 guarantees us that for every simple permutation, at least one of the steps 2 - 5 may be applied, unless the permutation is sorted (and contains only 1-cycles).

Algorithm ***Sort*** **(π)**

1. Transform permutation π into an equivalent simple permutation $\hat{\pi}$ (Lemma 5).
2. While $G = G(\hat{\pi})$ contains a 2-cycle, apply a 2-transposition (Lemma 7).
3. While G contains an oriented 3-cycle, apply a 2-transposition on it.
4. If two interleaving 3-cycles are found in G, apply a $(0, 2, 2)$-sequence (Lemma 8).
5. If a shattered unoriented 3-cycle is found in G, apply a $(0, 2, 2)$-sequence (Lemma 12).
6. If G contains a cycle of length greater than 1, go to step 4.
7. Mimic the sorting of π using the sorting of $\hat{\pi}$ (Lemma 6).

Fig. 10. 1.5-approximation algorithm for sorting by transpositions.

Performing only steps 2 - 6 is an algorithm in its own, and is denoted by *Algorithm SortSimple*. The following theorem claims that algorithm SortSimple is a 1.5-approximation algorithm for sorting simple permutations:

Theorem 13 *Algorithm* SortSimple *is a 1.5-approximation algorithm for simple permutations, and it runs in time $O(n^2)$.*

Proof. The sequence of transpositions that is generated by the algorithm contains only 2-transpositions and $(0, 2, 2)$-sequences of transpositions. Therefore, every sequence of three transpositions 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 transpositions performed is $O(n)$, and so is the time to perform a transposition. Thus, it suffices to show that at each step, we can decide in linear time which transposition to apply. Steps 2 and 3 are trivial. Checking whether two pairs of black edges intersect can be done in constant time. Let A be an unoriented 3-cycle. For each pair of black edges (b_i, b_j) in A, scan all 3-cycles (in linear time) and check if they intersect with (b_i, b_j). If there is a 3-cycle B that intersects with all three pairs of edges of A, then A interleaves with B, and we can apply step 4. Otherwise, there are two cycles B and C such that A is shattered by B and C, and step 5 can be applied.

Now we are ready to prove the main result of the paper:

Theorem 14 *Algorithm* Sort *is a 1.5-approximation algorithm for general permutations, and it runs in time $O(n^2)$.*

Proof. By Theorem 13, we are guaranteed that $alg(\hat{\pi}) \leq 1.5 \cdot d(\hat{\pi})$, where $alg(\hat{\pi})$ is the number of transpositions used by Algorithm SortSimple to sort $\hat{\pi}$. Thus, by Theorem 3,

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

Using Lemma 6, we can sort π by $alg(\hat{\pi})$ transpositions, which implies an approximation ratio of 1.5.

Since steps 1 and 7 can be done in linear time, Theorem 13 implies that the running time of Algorithm Sort is $O(n^2)$.

4 Discussion and Open Problems

In this paper we studied the problem of sorting permutations by transpositions, gave a simple quadratic 1.5-approximation algorithm for the problem, and simplified the underlying theory. We believe that this is an important step towards solving some related open problems. The main open problem is to determine the complexity of sorting by transpositions. Devising algorithms with better approximation ratio and/or faster running time is also desirable. Another direction, which is more biologically relevant, is to consider algorithms for sorting permutations by a set of rearrangement operations (such as reversals, transpositions and translocations).

Acknowledgements

I am very grateful to my advisor, Ron Shamir, for many invaluable and helpful discussions and for reading the manuscript carefully. I would like to thank Roded Sharan for fruitful discussions, and Vineet Bafna for help in understanding the complexity of the Bafna-Pevzner Algorithm. This work was supported in part by the Israel Science Foundation (grant 309/02).

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(5):483–491, 2001.
2. V. Bafna and P. A. Pevzner. Genome rearragements 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, May 1998.
4. A. Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. In *Proc. 12th Annual Symposium on Combinaotrial Pattern Matching (CPM '01)*, 2001.
5. P. Berman, S. Hannanhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proc. of 10th Eurpean Symposium on Algorith,s (ESA'02)*, pages 200–210. Springer, 2002. LNCS 2461.
6. A. Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12(1):91–110, February 1999.
7. D. A. Christie. A 3/2-approximation algorithm for sorting by reversals. In *Proc. ninth annual ACM-SIAM Symp. on Discrete Algorithms (SODA 98)*, pages 244–252. ACM Press, 1998.
8. D. A. Christie. *Genome Rearrangement Problems*. PhD thesis, University of Glasgow, 1999.
9. H. Eriksson, K. Eriksson, J. Karlander, L. Svensson, and J. Wastlund. Sorting a bridge hand. *Discrete Mathematics*, 241(1-3):289–300, 2001.
10. Q. P. Gu, S. Peng, and H. Sudborough. A 2-approximation algorithm for genome rearrangements by reversals and transpositions. *Theoretical Computer Science*, 210(2):327–339, 1999.
11. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46:1–27, 1999. (Preliminary version in Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing 1995 (STOC 95), pages 178–189).
12. S. B. Hoot and J. D. Palmer. Structural rearrangements, including parallel inversions, within the chloroplast genome of Anemone and related genera. *J. Molecular Evooution*, 38:274–281, 1994.
13. H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal of Computing*, 29(3):880–892, 2000. (Preliminary version in Proceedings of the eighth annual ACM-SIAM Symposium on Discrete Algorithms 1997 (SODA 97), ACM Press, pages 344–351).
14. G. H. Lin and G. Xue. Signed genome rearrangements by reversals and transpositions: Models and approximations. *Theoretical Computer Science*, 259:513–531, 2001.
15. J. Meidanis, M. E. Walter, and Z. Dias. Reversal distance of signed circular chromosomes. manuscript, 2000.
16. J. D. Palmer and L. A. Herbon. Tricircular mitochondrial genomes of Brassica and Raphanus: reversal of repeat configurations by inversion. *Nucleic Acids Research*, 14:9755–9764, 1986.
17. P. A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, 2000.
18. D. Sankoff and N. El-Mabrouk. Genome rearrangement. In T.Jiang, T. Smith, Y. Xu, and M. Q. Zhang, editors, *Current Topics in Computational Molecular Biology*. MIT Press, 2002.

19. J. Setubal and J. Meidanis. *Introduction to Computational Biology*. PWS Publishing Co., 1997.
20. R. Shamir. Algorithms in molecular biology: Lecture notes, 2002. Available at http://www.math.tau.ac.il/~rshamir/algmb/01/algmb01.html.
21. M. E. Walter, Z. Dias, and J. Meidanis. Reversal and transposition distance of linear chromosomes. In *String Processing and Information Retrieval: A South American Symposium (SPIRE 98)*, 1998.