A 1.375-Approximation Algorithm for Sorting by Transpositions

Isaac Elias[1] and Tzvika Hartman[2]

[1] Dept. of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden. isaac@nada.kth.se.

[2] Dept. of Molecular Genetics, Weizmann Institute of Science, Rehovot 76100, Israel. tzvi.hartman@weizmann.ac.il. Work done while at the Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science.

Abstract. Sorting permutations by transpositions is an important problem in genome rearrangements. 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 this problem is still open and it has been a ten-year-old open problem to improve the best known 1.5-approximation algorithm. In this paper we provide a 1.375-approximation algorithm for sorting by transpositions. The algorithm is based on a new upper bound on the diameter of 3-permutations. In addition, we present some new results regarding the transposition diameter: We improve the lower bound for the transposition diameter of the symmetric group, and determine the exact transposition diameter of 2-permutations and simple permutations.

1 Introduction

When estimating the evolutionary distance between two organisms using genomic data one wishes to reconstruct the sequence of evolutionary events that transformed one genome into the other. In the 1980's, evidence was found that some species have essentially the same set of genes, but that their gene order differs [17, 13]. This suggests that global rearrangement events, such as reversals and transpositions of genome segments, can be used to trace the evolutionary path between genomes. As opposed to local point mutations (i.e., insertions, deletions, and substitutions of nucleotides) global rearrangements are rare and may therefore provide more accurate and robust clues to the evolution.

In the last decade, a large body of work was devoted to genome rearrangement problems. Genomes are represented by permutations, with the genes appearing as elements. 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 the reader is referred to [18–20].

The problem of sorting permutations by reversals has been studied extensively. It was shown to be NP-hard [8], and several approximation algorithms have been suggested [4, 7, 9]. 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 [14, 6, 3, 21].

There has been significantly less progress on the problem of sorting by transpositions. A transposition is a rearrangement operation, in which a segment is cut out of the permutation, and pasted in a different location. The complexity of sorting by transpositions is still open. It was first studied by Bafna and Pevzner [5], who devised a 1.5-approximation algorithm, which runs in quadratic time. The algorithm was simplified by Christie [9] and further by Hartman [12], which also proved that the analogous problem for circular permutations is equivalent. Eriksson et al. [10] provided an algorithm that sorts any given permutation on n elements by at most $2n/3$ transpositions, but has no approximation guarantee.

The transposition diameter of the symmetric group S_n is unknown. Bafna and Pevzner [5] proved an upper bound of $\frac{3}{4}n$, which was improved to $\frac{2}{3}n$ by the algorithm of Eriksson et al. [10]. A lower bound of $\lfloor\frac{n-1}{2}\rfloor+1$ (for circular permutations) is given in [9, 10, 16], where it was conjectured to be the transposition diameter, except for $n = 13$ and $n = 15$.

In this paper we study the problem of sorting permutations by transpositions. We begin with some results regarding the transposition diameter. We prove a lower bound of $\lfloor\frac{n}{2}\rfloor + 1$ on the transposition diameter of the symmetric group of circular permutations, which shows that the conjecture of [9, 10, 16] is not accurate. Next, we deal with three subsets of the symmetric group (that have been considered in the genome rearrangement literature): simple permutations, 2-permutations, and 3-permutations. We show that the diameter for 2-permutations is $\frac{n}{2}$ (for circular permutations of size n), and for simple permutations is $\lfloor\frac{n}{2}\rfloor$. We prove an upper bound of $11\left\lfloor\frac{n}{24}\right\rfloor + \left\lfloor 3\frac{(n/3 \mod 8)}{2}\right\rfloor + 1$ on the diameter of 3-permutations. Then we derive our main result: A 1.375-approximation algorithm for sorting by transpositions, improving on the ten-year-old 1.5 ratio.

Our main result, like many other results in genome rearrangements, is based on a rigorous case analysis. However, since the number of cases is huge, we developed a computer program that systematically generates the proof. Each case in the proof is discrete and consists of a few elementary steps that can be verified by hand and thus it is a proof in the conventional mathematical sense. Since it is not practical to manually verify the proof as a whole, we have written a verification program, which takes the proof as an input and verifies that each elementary step in the proof is correct. The proof, along with the program, is presented in a user-friendly web interface [1]. A well-known example of a computer assisted proof is that of the *Four Color Theorem* [2] (see [22] for a list of other proofs).

2 Preliminaries

Let $\pi = (\pi_1 \ \dots \ \pi_n)$ be a permutation on n elements. Define a *segment* A in π as a sequence of consecutive 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 an exchange of two disjoint contiguous segments. If the segments are $A = \pi_i, \dots, \pi_{j-1}$ and $B = \pi_j, \dots, \pi_{k-1}$, then the result of applying τ on π, denoted $\tau \cdot \pi$, is $(\pi_1 \ \dots \ \pi_{i-1} \ \pi_j \ \dots \ \pi_{k-1} \ \pi_i \ \dots \ \pi_{j-1} \ \pi_k \ \dots \ \pi_n)$ (note that the end segments can be empty if $i = 1$ or $k - 1 = n$).

The problem of finding a shortest sequence of transpositions, which transforms a 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.

The problem of sorting linear permutations of size n is equivalent to sorting circular permutations of size $n + 1$ [12]. Many of the following definitions, as well as the presentation of the algorithm, are more clear for circular permutations. Therefore we present our results for circular permutations and, due to the equivalence, they are true also for linear ones. In a circular permutation there is an element 0, and the equivalent linear permutation can be obtained by simply removing this element.

Breakpoint Graph. The breakpoint graph [5] is a graph representation of a permutation, which is classical in the genome rearrangements literature. In this graph every element of the permutation is represented by a left and a right vertex. As defined below, every vertex is connected to one black and one gray edge. The intuitive idea is that the black edges describe the order in the permutation and the gray edges describe the order in the identity permutation. Throughout the paper all permutations are circular and therefore, for both indices and elements, we identify n and 0.

Definition 1 *Let $\pi = (\pi_0 \ \dots \ \pi_{n-1})$ be a permutation. The* breakpoint graph *$G(\pi)$ is a edge-colored graph on $2n$ vertices $\{l_0, r_0, l_1, r_1, \dots, l_{n-1}, r_{n-1}\}$. For every $0 \leq i \leq n - 1$, connect r_i and l_{i+1} by a gray edge, and for every π_i, connect l_{π_i} and $r_{\pi_{i-1}}$ by a black edge, denoted by b_i.*

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 1(a)).

Cycles. 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 $c_{odd}(\pi)$, and let $\Delta c_{odd}(\pi, \tau) = c_{odd}(\tau \cdot \pi) - c_{odd}(\pi)$. Bafna and Pevzner proved the following useful lemma:

Lemma 1 (Bafna and Pevzner [5]) *For all permutations π and transpositions τ, $\Delta c_{odd}(\pi, \tau) \in \{-2, 0, 2\}$.*

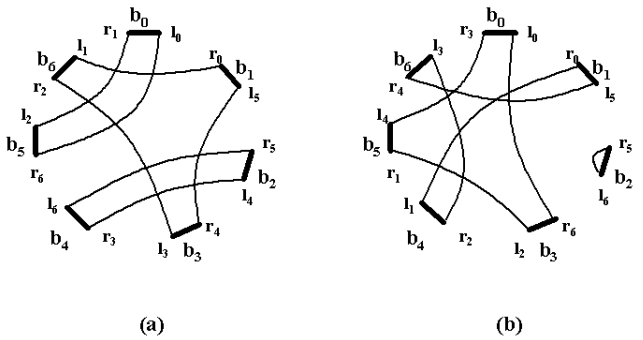

Fig. 1. (a) The circular breakpoint graph of the permutation $\pi = (0\ 5\ 4\ 3\ 6\ 2\ 1)$. Black edges are represented as thick lines on the circumference, and gray edges are chords. The three cycles are $(b_1\ b_3\ b_6)$, $(b_2\ b_4)$, and $(b_5\ b_0)$. (b) The circular breakpoint graph of π after applying a transposition on black edges b_0, b_2 and b_5.

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 1 implies the following lower bound on $d(\pi)$:

Theorem 2 (Bafna and Pevzner [5]) *For all permutations* π, $d(\pi) \geq (n(\pi) - c_{odd}(\pi))/2$.

By definition, every transposition must cut three black edges. The transposition that cuts black edges b_i, b_j and b_k is said to *apply on* these edges (see Figure 1). If these black edges are in cycle C, then the transposition is said to apply on C. A transposition τ is a *k-move* if $\Delta c_{odd}(\pi, \tau) = k$. A cycle is called *oriented* if there is a 2-move that is applied on three of its black edges; otherwise, it is *unoriented*.

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.

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 *simple* if it contains only short cycles. A permutation π is *simple* if $G(\pi)$ is simple, and is a *2-permutation* (resp. *3-permutation*) if $G(\pi)$ contains only 2-cycles (3-cycles).

A common technique in genome rearrangement literature is to transform permutations with long cycles into simple permutations. This transformation consists of inserting new elements into the permutations and thereby splitting the long cycles. The reader is referred to [12] for a thorough description. If $\hat{\pi}$ is the permutation attained by inserting elements into π then $d(\pi) \leq d(\hat{\pi})$, since inserting new elements only can result in a permutation that requires more moves to be sorted. Such a transformation is called *safe* if it maintains the lower bound of Theorem 2, i.e., if $n(\pi) - c_{odd}(\pi) = n(\hat{\pi}) - c_{odd}(\hat{\pi})$.

Lemma 3 (Lin and Xue [15]) *Every permutation can be transformed safely into a simple one.*

Note that the transformation only maintains the lower bound, not the exact distance[3]. 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 4 (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.*

The 1.375-approximation given in this paper first transforms the given permutation π into an equivalent simple permutation $\hat{\pi}$, then it finds a sorting sequence for $\hat{\pi}$, and, finally, the sorting of $\hat{\pi}$ is mimicked on π. Therefore, throughout most of the paper we will be concerned with simple permutations and short cycles.

Configurations and Components. 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 triplet defines a partition of the cyclic sequence 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 defined by 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 2c). *Two triplets* of black 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 2e).

A *configuration* of cycles is a subgraph of the breakpoint graph that is induced by one or more cycles. There are only two possible configurations of a 3-cycle in a breakpoint graph, which are shown in Figure 2 (a and b). It is easy to verify that the 3-cycle in (a) is oriented, and (b) is unoriented. A configuration A is a *sub-configuration* of a configuration B if the cycles in A form a subset of the cycles in B. A configuration A is *connected* if for any two cycles c_1 and c_k of A there are cycles $c_2, \ldots, c_{k-1}$ such that, for each $i \in [1, k-1]$, c_i intersects with c_{i+1}. A *component* is a maximal connected configuration in a breakpoint graph. The *size* of configurations and components is the number of cycles they contain, and are said to be *unoriented* if all their cycles are unoriented. They are called *small* if their size is at most 8; otherwise they are *big*.

In a configuration, an *open gate* is a pair of black edges of a 2-cycle or an unoriented 3-cycle that does not intersect with another cycle. The following is an important lemma by Bafna and Pevzner.

Lemma 5 (Bafna and Pevzner [5]) *Every open gate intersects with some other cycle in the breakpoint graph.*

[3] Unlike in the problem of sorting by reversals [11], in which the analogous transformation maintains the exact distance.

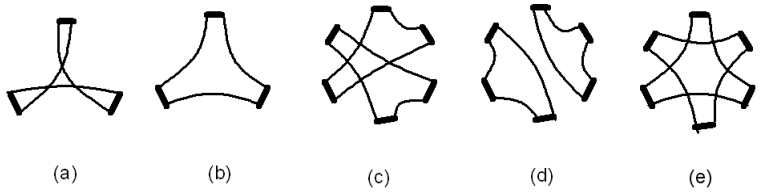

Fig. 2. Configurations of 3-cycles. (a) An oriented 3-cycle. (b) An unoriented 3-cycle. (c) Intersecting 3-cycles. (d) Non-intersecting 3-cycles. (e) Interleaving 3-cycles.

A configuration not containing open gates is referred to as a *full configuration*. For example, the configuration in 2(e) is full, whereas 2(c) has two open gates.

Sequence of Transpositions. An (x, y)-*sequence* of transpositions on a simple permutation (for $x \geq y$) is a sequence of x transpositions, such that at least y of them are 2-moves and that leaves a simple permutation at the end. For example, a 0-move followed by two consecutive 2-moves (which is called a $(0, 2, 2)$-sequence in previous papers [9, 12]) is a $(3, 2)$-sequence. A configuration (or component or permutation) *has* a (x, y)-sequence, if it is possible to apply such a sequence on its cycles.

The following result is the basis of the previous 1.5-approximation algorithms and will be used throughout the paper.

Lemma 6 (Christie [9] and Hartman [12]) *For every permutation (except for the identity permutation) there exists either a 2-move or a* $(3, 2)$*-sequence.*

Corollary 7 *For every permutation that has an oriented cycle and contains at least three 3-cycles there exists a* $(4, 3)$*-sequence.*

Transposition Diameter. The *transposition diameter*, $\mathrm{TD}(n)$, of the symmetric group is the maximum value of $d(\pi)$ taken over all permutations of n elements, i.e., $\mathrm{TD}(n) \triangleq \max_\pi d(\pi)$. Similarly, the transposition diameter of simple permutations TDS, 2-permutations TD2, and 3-permutations TD3, is the longest distance for any such permutation to the identity. [4]

3 Transposition Diameter Results

In this section, we first provide a lower bound on the transposition diameter. Then, we determine the exact transposition diameter of 2-permutations and simple permutations, and find an upper bound for the diameter of 3-permutations. Recall that throughout the paper by permutations we mean circular permutations. However, sorting linear permutations of size n is equivalent to sorting circular permutations of size $n + 1$ [12]. Therefore all bounds can be applied directly to linear permutations by replacing n with $n + 1$.

[4] The term *diameter* is somehow misleading for subsets of the symmetric group which are not a sub-group. However, we will stick to this term for the sake of consistency.

Previous works [10, 9, 16] on the transposition diameter have conjectured[5] that the most distant permutation is the reversed permutation $(0\ n-1\ \ldots\ 1)$. That is, it was believed that the transposition diameter is $\lfloor \frac{n-1}{2} \rfloor + 1$. However, the theorem below, which is proved in the full version of the paper, disproves this conjecture. Although the improvement of the bound is minor, we believe that this result is important since lower bounds on transposition problems are quite rare and hard to obtain.

The proofs of the following theorems are given in the full version of the paper.

Theorem 8 $TD(n) \geq \lfloor \frac{n}{2} \rfloor + 1$.

For linear permutations of size n the lower bound is given by $\lfloor \frac{n+1}{2} \rfloor + 1$.

Theorem 9 $TD2(n) = \frac{n}{2}$.

Theorem 10 $TDS(n) = \lfloor \frac{n}{2} \rfloor$.

3.1 Diameter for 3-permutations

The main result given in this section is an upper bound for the diameter of 3-permutations, which is the basis of the 1.375-approximation algorithm for sorting by transpositions (Section 4). This result, like many other results in genome rearrangements, is based on a rigorous case analysis. However, since the number of cases is huge, we developed a computer program that systematically analyzes all the cases. Below we describe the case analysis.

Our goal is to show that every 3-permutation with at least 8 cycles has an (x, y)-sequence such that $x \leq 11$ and $\frac{x}{y} \leq \frac{11}{8}$. Such a sequence is referred to as an $\frac{11}{8}$-*sequence*. By Corollary 7, we need only consider unoriented configurations, since a $(4, 3)$-sequence is an $\frac{11}{8}$-sequences. Thus, in the sequel, when we say configurations we refer to unoriented configurations. The case analysis is done in two steps. In the first step, below, all big components are shown to have an $\frac{11}{8}$-sequence. In the second step, which is described ion the full version of the paper, we consider permutations with at least 8 cycles such that all components are small and prove that also these permutations have an $\frac{11}{8}$-sequence.

Analysis of Unoriented Configurations The enumeration over all components starts from the basic building blocks: connected configurations consisting of two unoriented cycles. There are only two such configurations, the *unoriented interleaving pair* (Figure 2e) and the *unoriented intersecting pair* (Figure 2c). From these two configurations it is possible to build any other unoriented connected configuration by successively *adding* new unoriented cycles to the configuration. Adding a cycle to a configuration is done by inserting its black edges somewhere in the configuration. If it is possible to create a configuration B by adding a cycle to a configuration A, then B is said to be an *extension* of A. For

[5] [10] conjectured that this was the case with exceptions for $n = 14$ and $n = 16$.

example, both the unoriented interleaving and intersecting pair are extensions of the configuration of only one unoriented 3-cycle.

Consider a full configuration C and let A be a sub-configuration of C. Then C can be constructed by a series of extensions of A. In particular, this means that A can be extended into configuration B, that also is a sub-configuration of C. If A has an open gate then there is such extension B that is closing the open gate (since by definition, C has no open gates), i.e., the pair of black edges constituting the open gate in A intersects with a pair of black edges in B. Otherwise, there is an extension B with at most one open gate that is also a sub-configuration of C.

From the discussion above it follows that there are two types of extensions that are sufficient for building any component. These *sufficient extensions* are (1) extensions closing open gates and (2) extensions of full configurations, such that the extended configuration has at most one open gate. We refer to configurations that are realizable through a series of sufficient extensions from either the unoriented interleaving pair or the unoriented intersecting pair as *sufficient configurations*. Note that in particular, this means that every sufficient configuration has at most two open gates.

The following lemma is proved by our computerized case analysis:

Lemma 11 *Every unoriented sufficient configuration of 9 cycles has an* $\frac{11}{8}$*-sequence.*

By definition every big component has a sufficient configuration of size 9. Therefore the above lemma states that if a permutation contains a big component then there is an $\frac{11}{8}$-sequence.

One way of proving Lemma 11 would be to give a sorting for each of the sufficient configurations of 9 cycles. Such a case analysis would be too time consuming even for a computer. Instead, we utilize the notion of sufficiency and the fact that a sorting sequence for a configuration is also a sub-sorting for every extension of it. In Figure 3 we describe the case analysis which intuitively can be thought of as a breadth first search. When performing the analysis it turns out that no configuration of 10 cycles is added to the queue. This means that all sufficient configurations of 9 cycles have an $\frac{11}{8}$-sequence.

1. Initiate a queue of configurations to contain the unoriented interleaving pair and the unoriented intersecting pair.
2. While the queue is non-empty do:
   (a) Remove the first configuration, A, from the queue.
   (b) For each sufficient extension B of A do:
      i. If B does not have an $\frac{11}{8}$-sequence add it to the queue.
      ii. Otherwise give the sorting sequence for B.

Fig. 3. A brief description of the case analysis.

It should be stressed that the program itself is not a proof of the lemma. The proof is the case analysis which is the output of the program. Although each separate case can be verified by hand it is not an appealing thought to verify

80.000 such cases. To remedy this, the case analysis is presented in a user-friendly web interface [1] facilitating a general understanding of its correctness. Moreover, to affirm the correctness we have written a small verification program. This program verifies the proof by verifying (1) that every given sorting is a correct sorting and (2) that all sufficient extensions are considered. Thus the proof as a whole can be checked by verifying the correctness of this small program.

To complete the analysis we now consider small components. Small components that do not have an $\frac{11}{8}$-sequence are called *bad small components*. Our computerized enumeration found that there are only five such components. The second step of the case analysis, which is described in the full version of the paper, is to show that permutations with at least 8 cycles that contain only bad small components have an $\frac{11}{8}$-sequence.

Lemma 12 *Let π be a permutation with at least 8 cycles that contains only bad small components. Then, π has an $\frac{11}{8}$-sequence.*

The conclusion of the case analysis in this section is the corollary below. It follows from Lemmas 11 and 12 and is the basis of the $\frac{11}{8} = 1.375$ approximation algorithm.

Corollary 13 *Every 3-permutation with at least 8 cycles has an $\frac{11}{8}$-sequence.*

The Diameter for 3-permutations Here we present an upper bound on the diameter for 3-permutations (the proof is given in the full version). In 3-permutations of size n the number of cycles is $c = n/3$. Let $g(c) \triangleq 11\lfloor c/8 \rfloor + \lfloor 3(c \mod 8)/2 \rfloor$ and define f as follows:

$$f(c) \triangleq \begin{cases} g(c) + 1 \text{ if } c \mod 8 = 1 \\ g(c) \qquad \text{otherwise} \end{cases} \tag{1}$$

Note that $f(8l + r) = 11l + f(r)$. This function gives the upper bound on the diameter for 3-permutations.

Theorem 14 $TD3(n) \leq f(\frac{n}{3}) \leq 11\left\lfloor \frac{n}{24} \right\rfloor + \left\lfloor 3\frac{(n/3 \mod 8)}{2} \right\rfloor + 1.$

4 The Approximation Algorithm

Now we are ready to present our main result: Algorithm *Sort*, which is a 1.375-approximation algorithm for sorting by transpositions (Figure 4). Intuitively, the algorithm sorts the permutation by repeatedly applying $(11, 8)$-sequences and since $\frac{11}{8} = 1.375$ we get the desired approximation ratio (based on the lower bound of Theorem 2). The following lemma, whose proof is deferred to the full version of the paper, analyzes the time complexity of the algorithm.

Lemma 15 *The time complexity of Algorithm Sort is $O(n^2)$.*

Algorithm *Sort* (π)

1. Transform permutation π into a simple permutation $\hat{\pi}$ (Lemma 3).
2. Check if there is a $(2,2)$-sequence. If so, apply it.
3. While $G(\hat{\pi})$ contains a 2-cycle, apply a 2-move (Christie [9]).
4. While $G(\hat{\pi})$ contains at least 8 cycles apply an $\frac{11}{8}$-sequence (Corollary 13).
5. While $G(\hat{\pi})$ contains a 3-cycle, apply a $(3,2)$-sequence (Lemma 6).
6. Mimic the sorting of π using the sorting of $\hat{\pi}$ (Lemma 4).

Fig. 4. A high-level description of the approximation algorithm *Sort*.

Theorem 16 *Algorithm Sort is a* 1.375*-approximation algorithm for sorting permutations by transpositions, and it runs in quadratic time.*

Proof. The running time is shown in Lemma 15. We now prove the approximation ratio. Depending on Step 2, there are two cases: either there is a $(2,2)$-sequence or not. Let c_3 (resp. c_2) represent the number of 3-cycles (2-cycles) in $G(\hat{\pi})$ after Step 2.

Case 1 In Step 2 if a $(2,2)$-sequence exists. According to the lower bound in Theorem 2 the best possible sorting is that using only 2-moves. Specifically this means that $\hat{\pi}$ can not be sorted better than first applying two 2-moves and then another $c_3 + c_2$ 2-moves to sort the remaining cycles. Therefore a lower bound for any sorting of $\hat{\pi}$ is $c_3 + c_2 + 2$. The algorithm gives a sorting using $2 + \frac{c_2}{2} + f(c_3 + \frac{c_2}{2})$ moves; 2 moves in Step 2, $c_2/2$ moves in Step 3 creating $c_2/2$ 3-cycles, and by the proof of Theorem 14 at most $f(c_3 + \frac{c_2}{2})$ moves in Steps 4 and 5. Thus the approximation ratio of the algorithm is

$$\frac{2 + \frac{c_2}{2} + f(c_3 + \frac{c_2}{2})}{c_3 + c_2 + 2} = \frac{2 + y + f(x)}{x + y + 2} \leq \frac{f(x) + 2}{x + 2}, \tag{2}$$

where $x = c_3 + \frac{c_2}{2}$ and $y = \frac{c_2}{2}$. In Table 1 the last expression of Equation 2 is shown to be bounded from above by $\frac{11}{8}$.

Case 2 In Step 2 if a $(2,2)$-sequence does not exist then there are $c_2 + c_3$ cycles in $G(\hat{\pi})$ and at least $c_3 + c_2 + 1$ moves are required to sort $\hat{\pi}$; at least one 0-move and by Theorem 2 at least $c_3 + c_2$ 2-moves are required. The algorithm gives a sorting using $\frac{c_2}{2} + f(c_3 + \frac{c_2}{2})$ moves. Thus the approximation ratio of the algorithm is

$$\frac{\frac{c_2}{2} + f(c_3 + \frac{c_2}{2})}{c_3 + c_2 + 1} = \frac{f(x) + y}{x + y + 1} \leq \frac{f(x)}{x + 1}, \tag{3}$$

where again $x = c_3 + \frac{c_2}{2}$ and $y = \frac{c_2}{2}$. In Table 1 the last expression of Equation 3 is shown to be bounded from above by $\frac{11}{8}$. □

5 Discussion and Open Problems

The main result of this paper is a 1.375-approximation algorithm for sorting by transpositions. In addition, there are some new advances regarding the transposition diameter. The main open problems are to determine the complexity

Table 1. Function table showing that the approximation ratio of the algorithm is $\frac{11}{8} = 1.375$. In the table $x = 8l + r$ and thus $f(x) = 11l + f(r)$.

r	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
f(r)	0	2	3	4	6	7	9	10
$\frac{\mathbf{f(x)+2}}{\mathbf{x+2}}$	$\frac{11l+2}{8l+2}$	$\frac{11l+4}{8l+3}$	$\frac{11l+5}{8l+4}$	$\frac{11l+6}{8l+5}$	$\frac{11l+8}{8l+6}$	$\frac{11l+9}{8l+7}$	$\frac{11l+11}{8l+8}$	$\frac{11l+12}{8l+9}$
$\frac{\mathbf{f(x)}}{\mathbf{x+1}}$	$\frac{11l}{8l+1}$	$\frac{11l+2}{8l+2}$	$\frac{11l+3}{8l+3}$	$\frac{11l+4}{8l+4}$	$\frac{11l+6}{8l+5}$	$\frac{11l+7}{8l+6}$	$\frac{11l+9}{8l+7}$	$\frac{11l+10}{8l+8}$

of sorting by transpositions, and to find the transposition diameter. We believe that our results give new insights for further investigation of these problems. In particular, our characterization of components which are "hard-to-sort" may be a key to better lower bounds and approximation algorithms.

Empirical evidence indicate that the upper bound given for the diameter of 3-permutations is very close to the true diameter. If this is correct, then there are permutations at distance 1.375 times the lower bound of Theorem 2. That is, finding a better lower bound is essential for improving the approximation ratio.

Acknowledgments We are grateful to Prof. Ron Shamir, Prof. Jens Lagergren, and Elad Verbin for many invaluable discussions. We also wish to thank Prof. Jens Lagergren for reading early drafts. The first author is grateful to Prof. Benny Chor for hosting him at Tel Aviv University.

References

1. `http://www.sbc.su.se/~cherub/SBT1375_proof`.
2. K. Appel and W. Haken. Every planar map is four colorable part I: Discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.
3. 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.
4. V. Bafna and P. A. Pevzner. Genome rearragements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.
5. V. Bafna and P. A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, May 1998. Preliminary version in the *Proceedings of SODA*, 1995.
6. A. Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. In *Combinatorial Pattern Matching (CPM '01)*, pages 106–117, 2001.
7. P. Berman, S. Hannanhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proc. of 10th Eurpean Symposium on Algorithms (ESA'02)*, pages 200–210. Springer, 2002. LNCS 2461.
8. A. Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12(1):91–110, February 1999.
9. D. A. Christie. *Genome Rearrangement Problems*. PhD thesis, University of Glasgow, 1999.
10. H. Eriksson, K. Eriksson, J. Karlander, L. Svensson, and J. Wastlund. Sorting a bridge hand. *Discrete Mathematics*, 241(1-3):289–300, 2001.

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.
12. T. Hartman. A simpler 1.5-approximation algorithm for sorting by transpositions. In *Combinatorial Pattern Matching (CPM '03)*, volume 2676, pages 156–169, 2003.
13. 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.
14. 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.
15. G. H. Lin and G. Xue. Signed genome rearrangements by reversals and transpositions: Models and approximations. *Theoretical Computer Science*, 259:513–531, 2001.
16. J. Meidanis, M. E. Walter, and Z. Dias. Transposition distance between a permutation and its reverse. In *Proceedings of the 4th South American Workshop on String Processing*, pages 70–79, 1997.
17. 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.
18. P. A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, 2000.
19. D. Sankoff and N. El-Mabrouk. Genome rearrangement. In *Current Topics in Computational Molecular Biology*. MIT Press, 2002.
20. R. Shamir. Algorithms in molecular biology: Lecture notes, 2002. Available at http://www.math.tau.ac.il/∼rshamir/algmb/01/algmb01.html.
21. E. Tannier and M.F. Sagot. Sorting by reversals in subquadratic time. In *Combinatorial Pattern Matching (CPM '04)*, volume 3109, pages 1–13, 2004.
22. U. Zwick. Computer assisted proof of optimal approximability results. In *Symposium On Discrete Mathematics (SODA-02)*, pages 496–505, 2002.