A Simpler 1.5-Approximation Algorithm for Sorting by Transpositions

Tzvika Hartman * Ron Shamir †

September 5, 2003

1 Introduction

A common approach in comparative genomics is to compare the order of appearance of orthologous genes in different genomes. 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 (such as reversals, transpositions and translocations) that transforms one permutation into the other. Assuming that one of the permutations is the identity permutation, the problem is to find the shortest number of operations for sorting a permutation, using a particular rearrangement operation (or a set of operations).

The problem of sorting permutations by reversals has been studied extensively. There has been less progress on the problem of sorting by transpositions. A transposition is a rearrangement operation, in which a segment is cut out of the permutation, and pasted in a different location. The complexity of sorting by transpositions is still open. Two rather complicated 1.5-approximation algorithms were devised [2, 3].

In this work we study the problem of sorting by transpositions. First, we prove that the problem of sorting circular permutations by transpositions is equivalent to the problem of sorting linear ones. 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 [2, 3]. Thus, the algorithm achieves running time which is equal to the best known [2], 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. An extended abstract of this work appeared in [5].

2 Preliminaries

Linear and Circular Permutations Our first result is the equivalence between the linear and circular cases:

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

In our work 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.

The Breakpoint Graph The combinatorial object which underlies most genome rearrangement studies is the *breakpoint graph* [1]. This graph, denoted by $G(\pi)$, is constructed from the given permutation π. It has black and gray edges, and is uniquely decomposed into disjoint cycles. The *length* of a cycle is the number of black edges in it. The maximum number of odd-length cycles is obtained iff π is the identity permutation. It was shown [2] that after applying a transposition, the number of odd-length cycles in the resulting graph may increase by at most 2. This implies a lower bound on the transposition distance, which was used to prove the approximation ratio in [2, 3, 5].

Transformation into Equivalent Simple Permutations A permutation π is called *simple* if its breakpoint graph contains only cycles of length at most 3. Following [4, 6], we show how to transform an arbitrary permutation into a simple one, while maintaining the lower bound discussed in the previous paragraph. Moreover, every sorting of the resulting simple permutation mimics a sorting of the original permutation with the same number of transpositions.

*Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel (tzvi@cs.weizmann.ac.il).
†School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel. (rshamir@post.tau.ac.il.)

In this work, we show a 1.5-approximation algorithm for sorting simple permutations. The above discussion implies that this algorithm translates into a 1.5-approximation algorithm for arbitrary permutations.

3 The Algorithm

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 graph of the identity permutation. Thus, the sorting can be viewed as a process of breaking the 2- and 3-cycles into 1-cycles. A *k-transposition* is a transposition that increases the number of odd cycles in the breakpoint graph by k. The task of resolving 2-cycles was solved by Christie:

Lemma 2 (Christie [3]) *If π is a permutation that contains a 2-cycle, then there exists a 2-transposition on π.*

A 3-cycle which can be split into three 1-cycles by a single transposition is called *oriented*. Thus, a 2-transposition is possible if there exists an oriented cycle (which is easy to identify). Hence, the remaining problem is how to handle unoriented 3-cycles. 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 that such a sequence is always possible (unless there is a 2-transposition).

Two 3-cycles are *interleaving* if their black edges appear alternatively along the circle in the breakpoint graph. Similarly, two pairs of black edges are *intersecting* if they appear alternatively in the breakpoint graph. 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.

The following results will be used in the algorithm:

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

Lemma 4 *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.*

Corollary 5 *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.*

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

Figure 1: 1.5-approximation algorithm for sorting by transpositions.

Our main result is formulated in the following theorem:

Theorem 6 *Algorithm* Sort *is a 1.5-approximation algorithm for general permutations, and it runs in time $O(n^2)$.*

References

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

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

[3] D. A. Christie. *Phd. Thesis*, University of Glasgow, 1999.

[4] S. Hannenhalli and P. Pevzner. *Journal of the ACM* 46(1): 1-27, 1999.

[5] T. Hartman. *Proc. of CPM'03*: 156–169, 2003.

[6] G. H. Lin and G. Xue. *Theoretical Computer Science* 259: 513–531, 2001.