A Faster 1.375-Approximation Algorithm for Sorting by Transpositions

Luís Felipe I. Cunha¹, Luis Antonio B. Kowada², Rodrigo de A. Hausen³, and Celina M.H. de Figueiredo¹

¹ Universidade Federal do Rio de Janeiro, Brasil  
{lfignacio,celina}@cos.ufrj.br  
² Universidade Federal Fluminense, Brasil  
luis@vm.uff.br  
³ Universidade Federal do ABC, Brasil  
haus@compscinet.org

Abstract. Sorting by Transpositions is an NP-hard problem for which several polynomial time approximation algorithms have been developed. Hartman and Shamir (2006) developed a 1.5-approximation algorithm, whose running time was improved to $O(n \log n)$ by Feng and Zhu (2007) with a data structure they defined, the permutation tree. Elias and Hartman (2006) developed a 1.375-approximation algorithm that runs in $O(n^2)$ time. In this paper, we propose the first correct adaptation of this algorithm to run in $O(n \log n)$ time.

Keywords: comparative genomics, genome rearrangement, sorting by transpositions, approximation algorithms.

1 Introduction

By comparing the orders of common genes between two organisms, one may estimate the series of mutations that occurred in the underlying evolutionary process. In a simplified genome rearrangement model, each mutation is a transposition, and the sole chromosome of each organism is modeled by a permutation, which means that there are no duplicated or deleted genes. A transposition is a rearrangement of the gene order within a chromosome, in which two contiguous blocks are swapped. The transposition distance is the minimum number of transpositions required to transform one chromosome into another. Bulteau et al. [3] proved that the problem of determining the transposition distance between two permutations – or Sorting by Transpositions (SBT) – is NP-hard.

Several approaches to handle the SBT problem have been considered. Our focus is to explore approximation algorithms for estimating the transposition distance between permutations, providing better practical results or lowering time complexities.

Bafna and Pevzner [2] designed a 1.5-approximation $O(n^2)$ algorithm, based on the cycle structure of the breakpoint graph. Hartman and Shamir [10], by
considering simple permutations, proposed an easier 1.5-approximation algorithm and, by exploiting a balanced tree data structure, decreased the running time to \( O(n^{3/2} \sqrt{\log n}) \). Feng and Zhu \cite{Feng2008} developed the balanced permutation tree data structure, further decreasing the complexity of Hartman and Shamir’s 1.5-approximation algorithm to \( O(n \log n) \).

Elias and Hartman \cite{Elias2003} obtained, by a thorough computational case analysis of cycles of the breakpoint graph, a 1.375-approximation \( O(n^2) \) algorithm. Firoz et al. \cite{Firoz2012} tried to lower the running time of this algorithm to \( O(n \log n) \) via a simple application of permutation trees, but we later found counter-examples \cite{Zelevinsky2013} that disprove the correctness of Firoz et al.’s strategy.

In this paper, we propose a new algorithm that uses the strategy of Elias and Hartman towards bad full configurations, implemented using permutation trees and achieving both a 1.375 approximation ratio and \( O(n \log n) \) time complexity. Section 2 contains basic definitions, Section 3 presents a strategy to find in linear time a sequence of two transpositions in which both are 2-moves, if it exists, and Section 4 describes our 1.375-approximation algorithm for SBT.

2 Background

For our purposes, a gene is represented by a unique integer and a chromosome with \( n \) genes is a permutation \( \pi = [\pi_0 \pi_1 \pi_2 \cdots \pi_n \pi_{n+1}] \), where \( \pi_0 = 0, \pi_{n+1} = n+1 \) and each \( \pi_i \) is a unique integer in the range \( 1, \ldots, n \). The transposition \( t(i, j, k) \), where \( 1 \leq i < j < k \leq n+1 \) over \( \pi \), is the permutation \( \pi \cdot t(i, j, k) \) where the product interchanges the two contiguous blocks \( \pi_i \pi_{i+1} \cdots \pi_{j-1} \) and \( \pi_j \pi_{j+1} \cdots \pi_{k-1} \). A sequence of \( q \) transpositions sorts a permutation \( \pi \) if \( \pi t_1 t_2 \cdots t_q = i \), where every \( t_i \) is a transposition and \( i \) is the identity permutation \([0 \ 1 \ 2 \ \cdots \ n \ n + 1] \). The transposition distance of \( \pi \), denoted \( d(\pi) \), is the length of a minimum sequence of transpositions that sorts \( \pi \).

Given a permutation \( \pi \), the breakpoint graph of \( \pi \) is \( G(\pi) = (V, R \cup D) \); the set of vertices is \( V = \{0, -1, +1, -2, +2, \ldots, -n, +n, -(n+1)\} \), and the edges are partitioned into two sets, the directed reality edges \( R = \{i \rightarrow j = (\pi_i, -\pi_{i+1}) \mid i = 0, \ldots, n\} \) and the undirected desire edges \( D = \{i \rightarrow -(i+1) \mid i = 0, \ldots, n\} \). Fig. 1 shows \( G([0 \ 1 \ 0 \ 9 \ 8 \ 7 \ 1 \ 6 \ 1 \ 1 \ 5 \ 4 \ 3 \ 2 \ 12]) \), the horizontal lines represent the edges in \( R \) and the arcs represent the edges in \( D \).

![Fig. 1.](image)

\( G([0 \ 1 \ 0 \ 9 \ 8 \ 7 \ 1 \ 6 \ 1 \ 1 \ 5 \ 4 \ 3 \ 2 \ 12]) \). The cycles \( C_2 = (1 \ 3 \ 6) \) and \( C_3 = (5 \ 8 \ 10) \) intersect, but \( C_2 \) and \( C_3 \) are not interleaving; the cycles \( C_1 = (0 \ 2 \ 4) \) and \( C_2 = (1 \ 3 \ 6) \) are interleaving, and so are \( C_3 = (5 \ 8 \ 10) \) and \( C_4 = (7 \ 9 \ 11) \).

Every vertex in \( G(\pi) \) has degree 2, so \( G(\pi) \) can be partitioned into disjoint cycles. We shall use the terms a cycle in \( \pi \) and a cycle in \( G(\pi) \) interchangeably.