Silva *et al. Algorithms for Molecular Biology* (2022) 17:1
https://doi.org/10.1186/s13015-022-00205-z

Algorithms for Molecular Biology

RESEARCH Open Access

A new 1.375-approximation algorithm for sorting by transpositions

Luiz Augusto G. Silva[1*], Luis Antonio B. Kowada[3], Noraí Romeu Rocco[2] and Maria Emília M. T. Walter[1]

Abstract

Background: SORTING BY TRANSPOSITIONS (SBT) is a classical problem in genome rearrangements. In 2012, SBT was proven to be $\mathcal{NP}$-hard and the best approximation algorithm with a 1.375 ratio was proposed in 2006 by Elias and Hartman (EH algorithm). Their algorithm employs *simplification*, a technique used to transform an input permutation π into a *simple permutation* $\hat{\pi}$, presumably easier to handle with. The permutation $\hat{\pi}$ is obtained by inserting new symbols into π in a way that the lower bound of the transposition distance of π is kept on $\hat{\pi}$. The simplification is guaranteed to keep the lower bound, not the transposition distance. A sequence of operations sorting $\hat{\pi}$ can be mimicked to sort π.

Results and conclusions: First, using an algebraic approach, we propose a new upper bound for the transposition distance, which holds for all S_n. Next, motivated by a problem identified in the EH algorithm, which causes it, in scenarios involving how the input permutation is simplified, to require one extra transposition above the 1.375-approximation ratio, we propose a new approximation algorithm to solve SBT ensuring the 1.375-approximation ratio for all S_n. We implemented our algorithm and EH's. Regarding the implementation of the EH algorithm, two other issues were identified and needed to be fixed. We tested both algorithms against all permutations of size $n, 2 \leq n \leq 12$. The results show that the EH algorithm exceeds the approximation ratio of 1.375 for permutations with a size greater than 7. The percentage of computed distances that are equal to transposition distance, computed by the implemented algorithms are also compared with others available in the literature. Finally, we investigate the performance of both implementations on longer permutations of maximum length 500. From the experiments, we conclude that maximum and the average distances computed by our algorithm are a little better than the ones computed by the EH algorithm and the running times of both algorithms are similar, despite the time complexity of our algorithm being higher.

Keywords: Transposition Distance Problem, Sorting by Transpositions, Genome rearrangements, Approximation algorithms

Background

It is known from previous research that the genomes of different species may present essentially the same set of genes in their DNA strands, although not in the same order [1, 2], suggesting the occurrence of mutational events that affect large portions of DNA. These are presumably rare events and, therefore, may provide important clues for the reconstruction of the evolutionary history among species [3, 4]. One such event is the *transposition*, which swaps the position of two adjacent blocks of genes in one chromosome. Considering that there are no duplicated genes, each gene can be represented by an integer and the chromosome by a permutation, then the TRANSPOSITION DISTANCE PROBLEM (TDP) aims to find the minimum number of transpositions required to transform one chromosome into another. TDP can be reduced to the SORTING BY

*Correspondence: laugustogarcia@gmail.com
[1] Departmento de Ciência da Computação, Universidade de Brasília, Brasília, Brazil
Full list of author information is available at the end of the article

TRANSPOSITIONS problem (SBT) using the identity as the target permutation.

The first approximation algorithm to solve SBT was devised in 1998 by Bafna and Pevzer [5], with a 1.5 ratio, based on the properties of a structure called the cycle graph. In 2006, Elias and Hartman [6] presented a 1.375-approximation algorithm (EH algorithm) with time complexity $O(n^2)$, the best known approximation solution so far for SBT. In 2012, Bulteau, Fertin and Rusu [7] demonstrated that SBT is $\mathcal{NP}$-hard.

In a later study, the time complexity of the EH algorithm was improved to $O(n \log n)$ by Cunha et al. [8]. Improvements to the EH algorithm, including heuristics, were proposed by Dias and Dias [9, 10].

Other studies, using different approaches, other than the cycle graph, were also published. For instance, Hausen et al. [11] studied SBT using a structure named toric graph, which was previously devised by Erikson et al. [12], used by the later ones to derive the upper bound of $\left\lfloor \frac{2n-2}{3} \right\rfloor$ for the transposition diameter, the best known so far for SBT. Galvão and Dias [13] studied solutions for SBT using three different structures: permutation codes, a concept previously introduced by Benoît-Gagné and Hamel [14]; breakpoint diagram,[1] introduced by Walter et al. [15]; and longest increasing subsequence, introduced by Guyer et al. [16]. Rusu [17], on the other hand, used a structure called log-list, formerly devised with the name link-cut trees by Sleator and Tarjan [18], to derive another $O(n \log n)$ 1.375-approximation algorithm for SBT. In addition to these, recently, other studies have been proposed involving variations of the transposition event. As examples, Lintzmayer et al. [19] studied the problem of SORTING BY PREFIX AND SUFIX TRANSPOSITIONS, as well as other problems combining variations of the transposition event with variations of the reversal event. Oliveira et al. [20] studied the transposition distance between two genomes considering intergenic regions, a problem they called SORTING PERMUTATIONS BY INTERGENIC TRANSPOSITIONS.

Meidanis and Dias [21] and Mira and Meidanis [22] were the first authors to propose the use of an algebraic approach to solve SBT, as an alternative to the methods based on the cycle graph. The goal was to provide a more formal approach for solving rearrangement problems using known results from the permutation groups theory. Mira et al. [23] have shown the feasibility of using an algebraic approach to solve SBT by formalising the Bafna and Pevzner's 1.5-approximation algorithm [5] using an algebraic tooling.

Regarding the studies using the cycle graph, after the work of Bafna and Pevzner [5], the use of *simplification* [24–26] became predominant. Simplification was introduced to make it easier to deal with long cycles in the cycle graph. In Appendix 1, we show a side effect of simplification, which ultimately makes the EH algorithm, in certain scenarios, to require one extra transposition above the 1.375-approximation ratio and, therefore, we avoided its use in our work. To the best of our knowledge, there is no tooling in the cycle graph literature to deal with long cycles, probably due to the predominance of simplification. For this reason, in this paper, we prefer to use an algebraic approach based on the work of Mira et al. [23]. In it, we are able to handle with long cycles without having to insert new symbols into the original permutation.

This paper is organized as follows. First, we present a brief background on permutation groups, necessary to understand the algebraic approach used in our work, followed by an algebraic formalisation of SBT. Next, we propose a new upper bound for SBT, valid for all S_n, improving the upper bound devised by Bafna and Pevzner [5, 27]. Then, we propose a new approximation algorithm to solve SBT that ensures the 1.375-approximation for all S_n. Finally, we present experimental results on all permutations of length n, $2 \leq n \leq 12$, of implementations of the EH algorithm and ours. The percentage of computed distances that are equal to transposition distance computed by the EH algorithm and ours are compared with others available in the literature. We also investigate the performance of the implementations of both algorithms with longer permutations of sizes ranging from 20 to 500, and compare the results with similar experiments conducted in other studies. Two other issues were found in the EH algorithm, one affecting both published versions [6, 28] and another one affecting only the journal version [28]. The issues are reported in the Appendix 3.

Permutation groups

The results presented next are classical in the literature and their proofs can be found in abstract algebra textbooks [29, 30].

The Symmetric Group S_n on a finite set E of n symbols is the group formed by all *permutations* on n distinct elements of E, defined as bijections from E to itself, under the operation of composition. The product of two permutations is defined as their composition as functions. Thus,

[1] Do not confuse with breakpoint graph.

if α and β are permutations in S_n, then $\alpha \cdot \beta$, or simply $\alpha\beta$, is the function that maps any element x of E to $\alpha(\beta(x))$.

An element $x \in E$ is said to be a *fixed* element of $\alpha \in S_n$, if $\alpha(x) = x$. If there exists a subset $\{c_1, c_2, \ldots, c_{\kappa-1}, c_\kappa\}$ of distinct elements of E, such that

$$\alpha(c_1) = c_2, \alpha(c_2) = c_3, \ldots, \alpha(c_{\kappa-1}) = c_\kappa, \alpha(c_\kappa) = c_1,$$

and α fixes all other elements, then we call α a *cycle*. In *cycle notation*, this cycle is written as $\alpha = (c_1\ c_2 \ldots c_{\kappa-1}\ c_\kappa)$, but any of $(c_2 \ldots c_{\kappa-1}\ c_\kappa\ c_1)$, ..., $(c_\kappa\ c_1\ c_2 \ldots c_{\kappa-1})$ denotes the same cycle α. The number κ is the *length* of α, denoted by $|\alpha|$. In this case, α is also called a κ-*cycle*.

The *support of a permutation* α, denoted $Supp(\alpha)$, is the subset of moved (not fixed) elements of E. Two permutations α and β are said *disjoint*, if $Supp(\alpha) \cap Supp(\beta) = \varnothing$, i.e, if every symbol moved by one is fixed by the other. It is known that, if α and β are disjoint, then they commute as elements of S_n, under the composition operation.

Lemma 1 *Every permutation in S_n can be written as a product of disjoint cycles. This representation, called disjoint cycle decomposition , is unique, regardless of the order in which the cycles are written in the representation.*

For the sake of simplicity, a cycle β *in* or *of* a permutation α is a cycle in the disjoint cycle decomposition of α.

The identity permutation ι is the permutation fixing all elements of E. Fixed elements sometimes are omitted in the cycle notation. However, when necessary they are written as 1-cycles.

Theorem 2 *Every permutation in S_n can be written as a (not unique) product of 2-cycles.*

A permutation α is said to be *even*(*odd*) if it can be written as a product of an even(odd) number of 2-cycles.[2] Next, we present some important results related to the parity of permutations.

Theorem 3 *If a permutation α is written as a product of an even(odd) number of 2-cycles, it cannot be written as a product of an odd(even) number 2-cycles.*

Example 4 The permutation $\rho = [4\ 8\ 3\ 7\ 2\ 6\ 1\ 5]$, in cycle notation, is represented by $(1\ 4\ 7)(2\ 8\ 5)(3)(6)$. In this case, 3 and 6 are fixed elements and could be omitted in this notation. We can say that ρ can be written, in unique form, as a product of two disjoint 3-cycles. This permutation could be written as product of other cycles, but these cycles would not be disjoint. Furthermore, ρ could be written as (1 7) (1 4) (2 5) (2 8), using four 2-cycles, and also as (1 7) (4 7) (1 7) (4 7) (2 5) (2 8), using six 2-cycles.

Theorem 5 *If $\alpha, \beta \in S_n$ are permutations with the same parity, then the product $\alpha\beta$ is even.*

Proposition 6 *Let γ be a κ-cycle. If κ is odd, then γ is an even permutation, otherwise γ is odd.*

In order to avoid misunderstanding with the parity of cycles in the cycle graph formalism, which is opposite to the one classically used in permutation groups, we will always refer to the lengths of the cycles rather than their parity.

Algebraic formalisation of SBT

A permutation $\pi = [\pi_1\ \pi_2 \ldots \pi_n]$ can be represented in many different ways. In the genome rearrangement context, where π models a chromosome, one of the most used representations of π is the *cycle graph* [5], denoted $G(\pi)$ (a formal definition of $G(\pi)$ can be found in the first section of Appendix 1). An alternative representation to the cycle graph is using the algebraic approach proposed by Mira et al. [23], which is the one employed in this paper. In this approach, the permutation π is represented as the $(n+1)$-cycle $\bar{\pi} = (0\ \pi_1\ \pi_2 \ldots \pi_n)$ and the identity as $\bar{\iota} = (0\ 1\ 2\ \ldots\ n)$[3]. A correspondence between the cycles of the product $\bar{\iota}\bar{\pi}^{-1}$ (presented in the sequel) and the set of cycles of $G(\pi)$ is shown in the Appendix 2.

A 3-cycle $\tau = (\pi_i\ \pi_j\ \pi_k)$ is said to be *applicable* on $\bar{\pi}$ if the symbols π_i, π_j and π_k appear in $\bar{\pi}$ in the same cyclic order they are in τ, i.e., $\bar{\pi} = (\pi_i \ldots \pi_j \ldots \pi_k \ldots)$ [23]. The *application* of τ on $\bar{\pi}$ means multiply τ by $\bar{\pi}$. Thus, and only in this case, the product $\tau\bar{\pi}$ is a $(n+1)$-cycle, such that the symbols between π_i and π_{j-1}, including π_i but not π_j, in $\bar{\pi}$ are "cut" and then "pasted" between π_{k-1} and π_k, thus simulating a transposition on $\bar{\pi}$, as $\tau\bar{\pi} = (\pi_i\ \pi_j\ \pi_k)(\pi_0\pi_1 \ldots \pi_{i-1}\pi_i\pi_{i+1} \ldots \pi_{j-1}\pi_j\pi_{j+1} \ldots \pi_{k-1}\pi_k \ldots \pi_n) = (\pi_0\pi_1 \ldots \pi_{i-1}\pi_j\pi_{j+1} \ldots \pi_{k-1}\pi_i\pi_{i+1} \ldots \pi_{j-1}\pi_k \ldots \pi_n)$.

Example 7 Let $\bar{\pi} = (0\ 4\ 3\ 2\ 1\ 8\ 7\ 6\ 5)$. The 3-cycle $\tau = (0\ 2\ 7)$ is applicable to $\bar{\pi}$ and thus

[2] A 2-cycle is commonly referred to as transposition in the algebra literature. In order to avoid misunderstanding with the terminology, in this paper, "transposition" always refers to swapping two adjacent blocks of symbols in a permutation.

[3] Note that $\bar{\iota} = (0\ 1\ 2\ \ldots\ n)$ is not $\iota = (0)(1)\cdots(n)$.

simulates a transposition. The application $\tau\bar{\pi}$ yields (0 4 3 7 6 5 2 1 8). Now consider the 3-cycle $\tau' = (0\ 1\ 2)$. Note that τ' is not applicable to $\bar{\pi}$, and the result of the product $\tau'\bar{\pi}$ is (0 4 3 7 6 5)(1 8)(2), which is not a $(n+1)$-cycle and therefore does not represent a chromosome in our approach.

Given a $(n+1)$-cycle $\bar{\pi}$, the SORTING BY TRANSPOSITIONS problem (SBT) consists of finding the minimum number t, denoted $d(\bar{\pi})$, of transpositions represented as applicable 3-cycles needed to transform $\bar{\pi}$ into $\bar{\iota} = (0\ 1\ 2\ \ldots\ n)$, i.e.,

$$\tau_t \ldots \tau_1 \bar{\pi} = \bar{\iota}. \quad (1)$$

From the equality above, multiplying both sides by $\bar{\pi}^{-1}$, we have that

$$\tau_t \ldots \tau_1 = \bar{\iota}\bar{\pi}^{-1}. \quad (2)$$

Observe that by Proposition 6 and Theorem 5, the product of two cycles with the same length is an even permutation.

Proposition 8 *The permutation $\bar{\iota}\bar{\pi}^{-1}$ is an even permutation.*

The 3-*norm* [22] of an even permutation $\alpha \in S_n$, denoted by $||\alpha||_3$, corresponds to the smallest ℓ such that $\beta_\ell \ldots \beta_1 = \alpha$, where each β_i, $1 \leq i \leq \ell$, is a 3-cycle. Denote by $c^\circ{}_{odd}(\alpha)$, the number of odd-length cycles, also including 1-cycles, in α, respectively. Mira and Meidanis [22] demonstrated the following result.

Lemma 9 (Mira and Meidanis [22])

$$||\alpha||_3 = \frac{n - c^\circ{}_{odd}(\alpha)}{2}.$$

As $\bar{\iota}\bar{\pi}^{-1}$ is an even permutation (Proposition 8), then, as a corollary, a lower bound for SBT is derived.

Lemma 10 (Mira and Meidanis [22]) *If $\bar{\pi}$ is a $(n+1)$-cycle, then*

$$d(\bar{\pi}) \geq ||\bar{\iota}\bar{\pi}^{-1}||_3 \geq \frac{n + 1 - c^\circ{}_{odd}(\bar{\iota}\bar{\pi}^{-1})}{2}.$$

New upper bound for SBT

In this section, we present our main results. We begin with some basic definitions and results concerning the $\bar{\iota}\bar{\pi}^{-1}$ permutation. Next, we present a new upper bound for SBT and a new 1.375-approximation algorithm.

Cycles of $\bar{\iota}\bar{\pi}^{-1}$

Let γ be a cycle in $\bar{\iota}\bar{\pi}^{-1}$. If $\gamma = (a \ldots b \ldots c \ldots)$ and $\bar{\pi}^{-1} = (a \ldots c \ldots b \ldots)$, i.e., if the symbols a, b and c appear in γ in a cyclic order that is distinct from the one in $\bar{\pi}^{-1}$, then we say (a, b, c) is an *oriented triplet* and γ is an *oriented cycle*. Otherwise, if there is no oriented triplets in γ, then γ is an *unoriented cycle*. A cycle $\eta = (\eta_1\ \eta_2 \ldots \eta_{|\eta|})$ is a *segment* of γ if $\gamma = (\eta_1\ \eta_2 \ldots \eta_{|\eta|} \ldots)$. Observe that by definition, a cycle in $\bar{\iota}\bar{\pi}^{-1}$ is a segment of itself. Analogously, we define a segment of a cycle γ of $\bar{\iota}\bar{\pi}^{-1}$ as *oriented* or *unoriented*.

Let $\delta = (a\ b \ldots)$ and $\epsilon = (d\ e \ldots)$ be two cycles of $\bar{\iota}\bar{\pi}^{-1}$. If $\bar{\pi}^{-1} = (a \ldots e \ldots b \ldots d \ldots)$, i.e., if the symbols of the pairs (a, b) and (d, e) occur in alternate order in $\bar{\pi}^{-1}$, we say these pairs *intersect*, and that δ and ϵ are *intersecting* cycles. A special case is when $\delta = (a\ b\ c \ldots)$ and $\epsilon = (d\ e\ f \ldots)$ are such that $\bar{\pi}^{-1} = (a \ldots e \ldots b \ldots f \ldots c \ldots d \ldots)$, i.e., the symbols of the triplets (a, b, c) and (d, e, f) occur in alternate order in $\bar{\pi}^{-1}$. In this case, δ and ϵ are said to be *interleaving* cycles. Analogously, we define two segments of two $\bar{\iota}\bar{\pi}^{-1}$ cycles as *intersecting* or *interleaving*.

Example 11 Let $\bar{\pi} = (0\ 8\ 7\ 6\ 5\ 1\ 4\ 9\ 3\ 2)$ and $\bar{\iota}\bar{\pi}^{-1} = (0\ 3)\ (1\ 6\ 8)\ (2\ 4)\ (5\ 7\ 9)$. The cycles (0 3) and (2 4) are examples of intersecting cycles whereas (1 6 8) and (5 7 9) are interleaving cycles.

A κ-cycle in $\bar{\iota}\bar{\pi}^{-1}$ is called *short* if $\kappa \leq 3$; otherwise, it is called *long*. Similarly, a segment of a cycle of $\bar{\iota}\bar{\pi}^{-1}$ can be *short* or *long*.

Observe that, from Eq. 2, $\bar{\iota}\bar{\pi}^{-1}{\tau_1}^{-1} \ldots {\tau_t}^{-1} = \iota$, i.e., the application of the transpositions $\tau_1, \ldots, \tau_t$ sorting $\bar{\pi}$ (i.e., transforming $\bar{\pi}$ into $\bar{\iota}$) can be seen as the incremental multiplication of $\bar{\iota}\bar{\pi}^{-1}$ by ${\tau_1}^{-1}, \ldots, {\tau_t}^{-1}$.

Denote by $\Delta c^\circ{}_{odd}(\bar{\iota}\bar{\pi}^{-1}, \tau)$, the difference $c^\circ{}_{odd}(\bar{\iota}\bar{\pi}^{-1}\tau^{-1}) - c^\circ{}_{odd}(\bar{\iota}\bar{\pi}^{-1})$.

Proposition 12 (*Meidanis, Dias and Mira* [21, 22]) *If τ is an applicable 3-cycle then $\Delta c^\circ{}_{odd}(\bar{\iota}\bar{\pi}^{-1}, \tau) \in \{-2, 0, 2\}$.*

The maximum number of cycles in $\bar{\iota}\bar{\pi}^{-1}$ is obtained if and only if $\bar{\iota}\bar{\pi}^{-1}$ is the identity permutation ι. In this case, ι has $n+1$ cycles, being all odd-length (in particular, they are all of length 1).

We denote by μ-*move* an applicable 3-cycle τ such that $\Delta c^\circ{}_{odd}(\bar{\iota}\bar{\pi}^{-1}, \tau) = \mu$. According to the Proposition 12, the possible moves are (-2)-move, 0-move and 2-move.

Configurations and components

A *configuration* Γ is a disjoint product of segments of cycles of $\bar{\iota}\bar{\pi}^{-1}$, such that there is no two segments in Γ

of the same cycle of $\bar{\iota}\bar{\pi}^{-1}$. If $||\Gamma||_3 \leq 8$ then Γ is said to be *small*; otherwise, *big*.

Example 13 Let $\bar{\pi} = (0\ 6\ 5\ 3\ 2\ 1\ 8\ 7\ 4\ 9\ 14\ 13\ 12\ 11\ 10$, so $\bar{\iota}\bar{\pi}^{-1} = (\ 0\ 11\ 13\ )(\ 1\ 3\ 6\ )(\ 2\ 4\ 8\ )(\ 5\ 7\ 9\ )(\ 10\ 12\ 14\ )$. The product $(\ 1\ 3\ 6\ )(\ 2\ 4\ 8\ )$ is a small configuration of $\bar{\iota}\bar{\pi}^{-1}$.

A configuration Γ is *connected* if for any two segments γ_1 and γ_m of Γ, there are segments $\gamma_2, \ldots, \gamma_{m-1}$ in Γ such that for each $i \in [1, m-1]$, γ_i intersects or interleaves with γ_{i+1}. Γ is said to be a *component* if it consists of only one oriented cycle that does not intersect or interleave any other cycle of $\bar{\iota}\bar{\pi}^{-1}$; or it consists of a maximal connected configuration of $\bar{\iota}\bar{\pi}^{-1}$.

Example 14 Let $\bar{\pi} = (0\ 6\ 5\ 3\ 2\ 1\ 8\ 7\ 4\ 9\ 14\ 13\ 12\ 11\ 10)$. As $\bar{\iota}\bar{\pi}^{-1} = (\ 0\ 11\ 13\ )(\ 1\ 3\ 6\ )(\ 2\ 4\ 8\ )(\ 5\ 7\ 9\ )(10\ 12\ 14)$, so $(\ 0\ 11\ 13\ )(\ 10\ 12\ 14\ )$ and $(\ 1\ 3\ 6\ )(\ 2\ 4\ 8\ )(\ 5\ 7\ 9\ )$ are both components of $\bar{\iota}\bar{\pi}^{-1}$.

Let $(\ a\ b\ c\ )(\ d\ e\ f\ )$ be a configuration of $\bar{\iota}\bar{\pi}^{-1}$ consisting of two intersecting segments. If $\bar{\pi}^{-1} = (a \ldots e \ldots b \ldots f \ldots c \ldots d \ldots)$, i.e., if $(\ a\ b\ c\ )$ and $(\ d\ e\ f\ )$ interleave, then we call it the *unoriented interleaving pair*. On the other hand, if $\bar{\pi}^{-1} = (a \ldots f \ldots b \ldots c \ldots d \ldots e \ldots)$, i.e., $(\ a\ b\ c\ )$ and $(\ d\ e\ f\ )$ only intersect but do not interleave, then we call it the *unoriented intersecting pair*.

Let $\epsilon = (a\ b \ldots)$ be a segment of a configuration Γ. We call the pair (a, b) an *open gate* in Γ, if there is no cycle $(c\ d \ldots)$ in Γ such that (a, b) and (c, d) intersect; and there is no $e \in Supp(\epsilon)$ such that (a, b, e) is an oriented triplet. If Γ is a configuration not containing open gates, then it is a *full configuration*. Observe that the unoriented interleaving pair does not have open gates and therefore it is a full configuration. The unoriented intersecting pair, in its turn, has two open gates.

Sequences of applicable 3-cycles

We also denote by (x, y)-*sequence*, for $x \geq y$, a sequence of x applicable 3-cycles $\tau_1, \cdots, \tau_x$ such that, at least y of them are 2-moves. A (x, y)-sequence is said to be a $\frac{a}{b}$-*sequence* if $x \leq a$ and $\frac{x}{y} \leq \frac{a}{b}$.

Example 15 Let $\bar{\pi} = (0\ 4\ 8\ 3\ 7\ 2\ 6\ 1\ 5\ 9\ 14\ 13\ 12\ 1110)$, so $\bar{\iota}\bar{\pi}^{-1} = (0\ 11\ 13)\ (1\ 7\ 4)\ 4)\ (2\ 8\ 5)\ (3\ 9\ 6)\ (10\ 12\ 14)$. The sequence $\tau_1 = (1\ 4\ 7)$, $\tau_2 = (2\ 8\ 5)$, $\tau_3 = (1\ 4\ 7)$, $\tau_4 = (\ 3\ 9\ 6\ )$ is a (4, 3)-sequence, which is also a $\frac{11}{8}$-sequence.

We say a configuration Γ *allows* the application of a $\frac{a}{b}$-sequence if it is possible to write this sequence using the symbols of $Supp(\Gamma)$.

Auxiliary results

The proofs of some results in this section and the next rely on the analysis of a huge number of cases. Since it is impracticable to enumerate and verify by hand all the cases, we implemented, as Elias and Hartman [6], some computer programs [31] to systematically generate the proofs. In order to facilitate the visualisation and general understanding, the proofs are available to the reader in the form of a friendly web interface [32].

Next we show some auxiliary results.

Corollary 16 *If there is an oriented 3-cycle* $\gamma = (\ a\ b\ c\ )$ *in* $\bar{\iota}\bar{\pi}^{-1}$, *then* $(\ a\ b\ c\ )$ *is a 2-move.*

Proposition 17 *If there is an even-length cycle in* $\bar{\iota}\bar{\pi}^{-1}$, *then a 2-move exists.*

Proof Since $\bar{\iota}\bar{\pi}^{-1}$ is an even permutation (Proposition 8), then there is an even number of even-length cycles in $\bar{\iota}\bar{\pi}^{-1}$. Let $\gamma = (a\ b \ldots)$ and $\delta = (c\ d \ldots)$ be two even-length cycles of $\bar{\iota}\bar{\pi}^{-1}$. We have two cases:

(1) γ and δ intersect. In this case, we have that $\bar{\pi}^{-1} = (a \ldots d \ldots b \ldots c \ldots)$. Then $(a\ b\ c)$ is a 2-move.
(2) γ and δ do not intersect. W.l.o.g, suppose $\bar{\pi}^{-1} = (a \ldots b \ldots c \ldots d \ldots)$. In this case, $(a\ c\ b)$ is a 2-move. □

Lemma 18 *If there is a 5-cycle* $\gamma = (\ a\ d\ b\ e\ c\ )$ *in* $\bar{\iota}\bar{\pi}^{-1}$ *such that* (a, b, c) *is an oriented triplet, then there is a 2-move or a (3, 2)-sequence.*

Proof The possible distinct forms of $\bar{\pi}$ relatively to the positions of the symbols of $Supp(\gamma)$ are listed below. For each one, there is either a 2-move or a (3, 2)-sequence.

(1) $\bar{\pi} = (a \ldots b \ldots c \ldots d \ldots e \ldots)$. $\tau_1 = (\ a\ b\ c\ )$, $\tau_2 = (\ b\ c\ d\ ), \tau_3 = (\ c\ d\ e\ )$.
(2) $\bar{\pi} = (a \ldots b \ldots c \ldots e \ldots d \ldots)$. $\tau_1 = (\ b\ e\ d\ )$.
(3) $\bar{\pi} = (a \ldots b \ldots e \ldots c \ldots d \ldots)$. $\tau_1 = (\ a\ e\ c\ )$.
(4) $\bar{\pi} = (a \ldots e \ldots b \ldots d \ldots c \ldots)$. $\tau_1 = (\ a\ d\ c\ )$.

(5) $\bar{\pi} = (a \dots b \dots e \dots d \dots c \dots)$. $\tau_1 = (\,a\ d\ c\,)$.
(6) $\bar{\pi} = (a \dots d \dots b \dots e \dots c \dots)$. $\tau_1 = (\,a\ d\ b\,)$. □

Note that, by Lemma 18, if $\gamma = (\,a\ d\ b\ e\ c\,)$ is an oriented 5-cycle in $\bar{\iota}\bar{\pi}^{-1}$ such that (a, b, c) an oriented triplet, then $\bar{\pi} = (a \dots b \dots c \dots d \dots e \dots)$ is the only form of $\bar{\pi}$, relatively to the positions of the symbols of $Supp(\gamma)$, for which there is no 2-move. In this case, we call γ the *bad oriented 5-cycle*.

Lemma 19 *If there is an odd-length κ-cycle $\gamma = (a \dots b \dots c \dots)$ in $\bar{\iota}\bar{\pi}^{-1}$ such that $\kappa \geq 7$ and (a, b, c) is an oriented triplet, then there is either a 2-move or (4, 3)-sequence.*

Proof If $(a\ b\ c)$ is a 2-move, then the lemma holds. There is only one case where $(\,a\ b\ c\,)$ would not be a 2-move. W.l.o.g, suppose that this case is

$$\gamma = (\underbrace{d\ e \dots b}_{odd} \mid \underbrace{f \dots c}_{even} \mid \underbrace{g \dots a}_{even} \mid).$$

Vertical bars are used to indicate the locations where γ would be broken if $(\,a\ b\ c\,)$ were applied on $\bar{\pi}$, and subscripts to indicate the parity of the length of the resulting cycles. Note that the cycle γ can be rewritten as the product

$$\gamma = (\underbrace{a \dots}_{odd})(\underbrace{b \dots}_{odd})(\underbrace{c \dots}_{odd})(\,a\ d\ e\ b\ f\ c\ g\,).$$

There is only one form of $\bar{\pi}$ relatively to the symbols of the support of $(\,a\ d\ e\ b\ f\ c\ g\,)$ not allowing the application of a 2-move, which is $\bar{\pi} = (a \dots e \dots f \dots g \dots d \dots b \dots c \dots)$. For this $\bar{\pi}$, $\tau_1 = (a\ e\ f)$, $\tau_2 = (d\ e\ f)$, $\tau_3 = (b\ f\ d)$, $\tau_4 = (a\ c\ g)$ is (4, 3)-sequence of transpositions. □

Lemma 20 *If $\bar{\iota}\bar{\pi}^{-1} \neq \iota$, then a 2-move or (3, 2)-sequence exists.*

Proof If there is an even-length cycle in $\bar{\iota}\bar{\pi}^{-1}$, then by Proposition 17, a 2-move (i.e., a (1, 1)-sequence) exists. Thus, we assume $\bar{\iota}\bar{\pi}^{-1}$ containing only odd-length cycles.

(1) There is an oriented κ-cycle γ in $\bar{\iota}\bar{\pi}^{-1}$. If $\kappa = 3$, then Corollary 16 gives a 2-move and the lemma holds. If $\kappa = 5$, then Lemma 18 gives a 2-move or γ is the bad oriented 5-cycle. In this case, there is a (3, 2)-sequence. On the other hand, if $\kappa \geq 7$, then a 2-move or a (4, 3)-sequence, which contains a (3, 2)-sequence, is given by Lemma 19.
(2) All the cycles of $\bar{\iota}\bar{\pi}^{-1}$ are unoriented. Let $\gamma = (\,a\ b\ c\,)$ be a segment of a cycle of $\bar{\iota}\bar{\pi}^{-1}$. We have two cases:
   (a) γ interleaves with another segment $\delta = (\,d\ e\ f\,)$. In this case, we have that $\bar{\pi} = (a \dots f \dots c \dots e \dots b \dots d \dots)$. Then, $\tau_1 = (\,a\ c\ b\,)$, $\tau_2 = (\,d\ e\ f\,)$ and $\tau_3 = (\,a\ c\ b\,)$ is a (3, 2)-sequence.
   (b) γ intersects with two segments $\delta = (\,d\ e\ f\,)$ and $\epsilon = (\,g\ h\ i\,)$. For each of the 15 distinct forms of $\bar{\pi}$ (enumerated on [32]), relatively to the possible positions of the symbols of γ, δ and ϵ, there is a (3, 2)-sequence. □

Configuration analysis

At this point, we consider $\bar{\iota}\bar{\pi}^{-1}$ consisting only of odd-length unoriented cycles of any size or bad oriented 5-cycles. For the other cases, Corollary 16, Proposition 17 and Lemma 19 give a 2-move or a (4, 3)-sequence.

Our goal is to prove that, if $||\bar{\iota}\bar{\pi}^{-1}||_3 \geq 8$, then a $\frac{11}{8}$-sequence of transpositions exists. The analysis is divided in two parts. In the first part, we analyse configurations obtained from basic ones (defined below) by extension. In the second part, we analyse $\bar{\iota}\bar{\pi}^{-1}$ composed only of small components, not allowing application of $\frac{11}{8}$-sequences.

Extension of basic configurations

The analysis starts with the bad oriented 5-cycle, and the only two connected configurations of 3-norm equal to 2: the unoriented intersecting pair; and the unoriented interleaving pair. From these three *basic configurations*, it is possible to build any other connected configuration of $\bar{\iota}\bar{\pi}^{-1}$ by successive extensions. From a configuration Γ, we can obtain a larger configuration Γ', such that $||\Gamma'||_3 = ||\Gamma||_3 + 1$, extending Γ by three different *sufficient extensions*, as follows:

(1) If Γ has open gates, we can add a new unoriented 3-cycle segment to Γ, closing at least one open gate.
(2) If Γ has no open gates, we can add a new unoriented 3-cycle segment to Γ, so that this segment intersects or interleaves another one in Γ.

(3) Let γ be a segment in Γ. We can increase the length of γ by 2, originating a bad oriented 5-cycle; or a longer unoriented segment, so that at least one open gate is closed, if Γ has open gates; or creating up to two open gates, otherwise.

Example 21 We can extend the configuration Γ of Example 13 using extension 1, yielding $\Gamma' = (1\ 8\ 10)(5\ 7\ 12)(9\ 11\ 13)$. Then, with extension 2, we obtain $\Gamma'' = (1\ 8\ 10)(2\ 4\ 6)(5\ 7\ 12)(9\ 11\ 13)$. Finally, with extension 3, we obtain $\Gamma''' = (0\ 3\ 5\ 7\ 12)(1\ 8\ 10)(2\ 4\ 6)(9\ 11\ 13)$.

A *sufficient configuration* is a configuration obtained by successively extending one of the basic configurations referred above. The computerised analysis proves the following result.

Lemma 22 *If it is possible to build a sufficient configuration Γ of $\bar{\iota}\bar{\pi}^{-1}$ such that Γ is big, then Γ allows a $\frac{11}{8}$-sequence.*

Observe that our definition of configuration extension is similar to the one devised by Elias and Hartman [6]. However, Elias and Hartman [6] only handled with configurations consisting of (unoriented) 3-cycles, while our definition includes the generation of configurations containing longer segments.

Lemma 22 could be proven generating all the possible big configurations of 3-norm equal to 9 by extending the three basic configurations and then, for each, search for a $\frac{11}{8}$-sequence. However, this would be too time consuming. Instead, our computer program [31] employs a depth first search approach, in which, starting from the basic configurations, if we succeed in finding a $\frac{11}{8}$-sequence for a sufficient configuration, then we do not extend it further. The output of the program [31], which proves Lemma 22, is composed of 382,064 HTML files, one for each analysed case.

Analysis of small full configurations which do not allow $\frac{11}{8}$-sequences

To conclude the analysis, now we handle the small full configurations for which the program [31] did not find $\frac{11}{8}$-sequences, and that can occur as small components in $\bar{\iota}\bar{\pi}^{-1}$. Small components not allowing $\frac{11}{8}$-sequences are called *bad small components*.

Lemma 23 *The bad small components are the following*:

(1) *The bad oriented 5-cycle*;
(2) *The unoriented interleaving pair*;
(3) *The unoriented necklaces of size 4, 5 and 6;*[4] *and*
(4) *The twisted necklace of size 4.*

An *unoriented necklace* of size s is a component of s unoriented 3-cycles such that each cycle intersects with exactly two other cycles. The *twisted necklace* of size 4 is similar to the necklace of size 4, but two of its cycles intersect with the three others.

With the exception of the bad oriented 5-cycle, the bad small components listed above are the same ones found by Elias and Hartman [6], despite of the generation of configurations containing longer segments in our analysis.

With the help of computer program [31], we prove the following result.

Lemma 24 *If there is a configuration Λ of $\bar{\iota}\bar{\pi}^{-1}$ consisting only of bad small components such that $||\Lambda||_3 \geq 8$, then Λ allows a $\frac{11}{8}$-sequence.*

In order to prove Lemma 24, starting from each of the bad small components listed above, we successively extend them by adding another bad small component to the configuration, until finding a $\frac{11}{8}$-sequence. It turns out that no combination of bad small components with 3-norm greater than 7 was extended. The proof for Lemma 24 is composed of 842 HTML files.

New upper bound

The results presented in the previous section allow us to prove the corollary below. It follows from Proposition 17, part 1 from Lemma 20, which implies that, if we have an odd-length oriented cycle in $\bar{\iota}\bar{\pi}^{-1}$, than we have a 2-move, a (4, 3)-sequence, or this cycle is the bad oriented 5-cycle; and Lemmas 22 and 24 .

Corollary 25 *If $||\bar{\iota}\bar{\pi}^{-1}||_3 \geq 8$, then a $\frac{11}{8}$-sequence exists.*

On the other hand, if $||\bar{\iota}\bar{\pi}^{-1}||_3 < 8$, we only guarantee the existence of $\frac{3}{2}$-sequences. In the next section, we

[4] These components can be visualised on our site [32].

Table 1 For all $0 \leq r \leq 7$ such that $m = 8l + r$ and $l \geq 0$, the approximation ratio given by Algorithm 1 is at most $\frac{11}{8} = 1.375$

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

show that even in this scenario, the approximation ratio obtained by our algorithm is at most 1.375.

Finally, the last results prove the following upper bound for SBT.

Theorem 26

$$\begin{aligned} d(\bar{\pi}) \leq\ & 11\left\lfloor \frac{||\bar{\iota}\bar{\pi}^{-1}||_3}{8} \right\rfloor + \left\lfloor \frac{3(||\bar{\iota}\bar{\pi}^{-1}||_3 \bmod 8)}{2} \right\rfloor \\ \leq\ & 11\left\lfloor \frac{n+1-c^{\circ}{}_{odd}(\bar{\iota}\bar{\pi}^{-1})}{16} \right\rfloor + \\ & \left\lfloor \frac{3((n+1-c^{\circ}{}_{odd}(\bar{\iota}\bar{\pi}^{-1})) \bmod 16)}{4} \right\rfloor. \end{aligned}$$

Let $c_{odd}(\pi)$ be the number of odd cycles in $G(\pi)$ (see first section of Appendix 1). Since $c^{\circ}{}_{odd}(\bar{\iota}\bar{\pi}^{-1}) = c_{odd}(\pi)$, the result above can be restated replacing $\bar{\pi}$ and $c^{\circ}{}_{odd}(\bar{\iota}\bar{\pi}^{-1})$, by π and $c_{odd}(\pi)$ respectively. Thus, we derive the following upper bound for SBT, depending only on n and $c_{odd}(\pi)$.

Theorem 27

$$\begin{aligned} d(\pi) \leq\ & 11\left\lfloor \frac{n+1-c_{odd}(\pi)}{16} \right\rfloor \\ & + \left\lfloor \frac{3((n+1-c_{odd}(\pi)) \bmod 16)}{4} \right\rfloor. \end{aligned}$$

The new upper bound above improves the upper bound on the transposition distance devised by Bafna and Pevzner [5], valid for all S_n, based on their 1.5-approximation algorithm [27]. This upper bound allows us to obtain the following upper bound on the transposition diameter (TD).

Corollary 28 $TD(n) \leq 11\left\lfloor \frac{n}{16} \right\rfloor + \left\lfloor \frac{3(n \bmod 16)}{4} \right\rfloor$

The upper bound on the transposition diameter above, although tighter, for $n \geq 16$, than the one devised by Bafna and Pevzner [5] of $\left\lfloor \frac{3}{4}n \right\rfloor$ is not tighter than the one devised by Erikson et al. [12] of $\left\lfloor \frac{2n-2}{3} \right\rfloor$, for $n \geq 9$.

A new 1.375-approximation algorithm

In this section, we present a new 1.375-approximation algorithm for SBT (Algorithm 1). For a permutation $\pi \in S_n$, the algorithm returns an approximated distance between $\bar{\pi}$ and $\bar{\iota}$ or, equivalently, between π and ι. Intuitively, while $||\bar{\iota}\bar{\pi}^{-1}||_3 \geq 8$, it repeatedly applies $\frac{11}{8}$-sequences of transpositions on $\bar{\pi}$. When $||\bar{\iota}\bar{\pi}^{-1}||_3 < 8$, the algorithm only guarantees the application of $\frac{3}{2}$-sequences.

To reach the intended approximation ratio of 1.375 even when $||\bar{\iota}\bar{\pi}^{-1}||_3 < 8$, the algorithm has to search for a (2, 2)-sequence in its first step. In order to identify such a sequence, a look-ahead approach is used, meaning that the algorithm verifies if there is a second 2-move, after applying a first 2-move, generated either from an oriented cycle or from two even-length cycles of $\bar{\iota}\bar{\pi}^{-1}$.

Theorem 29 *The time complexity of Algorithm 1 is* $O(n^6)$.

Proof The time complexity of $O(n^6)$ is determined by the search for a (2, 2)-sequence. In order not to miss a 2-move, all triplets of an oriented cycle have to be checked to detect an oriented triplet leading to a 2-move, which is $O(n^3)$. Finding a 2-move by combining three symbols of two even-length cycles of $\bar{\iota}\bar{\pi}^{-1}$ requires $O(n^2)$. Thus, searching for a (2, 2)-sequence with the look-ahead technique to check if there is an extra 2-move needs time $O(n^6)$.

The largest loop of the algorithm (line 12) needs time $O(n^4)$, while the last loop is $O(n)$. □

Theorem 30 *Algorithm 1 is a 1.375-approximation algorithm for SBT.*

Proof We note that this proof follows a very similar approach to the one used by Elias and Hartman [6]. Let $f(x) = 11\left\lfloor \frac{x}{8} \right\rfloor + \left\lfloor \frac{3(x \bmod 8)}{2} \right\rfloor$. Depending on line 3, there are two cases.

Table 2 Comparison of the maximum approximation ratios given by the EH algorithm with ours (Alg1)

n	Transposition diameter	Max. approx.ratio		Averageapprox. ratio		Average distance		Number of times EH exceeded the 1.375-approx.	Time to sort all permutations[a]	
		EH	Alg1	EH	Alg1	EH	Alg1		EH	Alg1
2	1	1.00	1.00	1.0	1.0	1.00	1.00	0	$< 1s$	$< 1s$
3	2	1.00	1.00	1.0	1.0	1.20	1.20	0	$< 1s$	$< 1s$
4	3	1.00	1.00	1.0	1.0	1.6086	1.6086	0	$< 1s$	$< 1s$
5	3	1.00	1.00	1.0	1.0	2.0924	2.0924	0	$< 1s$	$< 1s$
6	4	$1.333\overline{3}$	1.00	1.0004	1.0	2.6063	2.6050	0	$< 1s$	$< 1s$
7	5	$1.333\overline{3}$	1.25	1.0129	1.0113	3.1762	3.1704	0	$< 1s$	$< 1s$
8	6	1.5	1.25	1.0210	1.0183	3.7178	3.7076	2	$< 2s$	$< 2s$
9	6	1.5	1.25	1.0301	1.0256	4.2796	4.2603	20	$\approx 10s$	$\approx 13s$
10	7	1.5	1.25	1.0341	1.0282	4.8051	4.7772	110	$\approx 3m$	$\approx 2m$
11	8	1.5	$1.333\overline{3}$	1.0392	1.0321	5.3526	5.3157	440	$\approx 35m$	$\approx 30m$
12	9	1.5	$1.333\overline{3}$	1.0415	1.0336	5.8694	5.8248	1448	$\approx 8.5h$	$\approx 8.1h$

The table includes other metrics such as the average approximation ratio and average distance given by each algorithm and the number of times the EH algorithm exceeds the 1.375-approximation ratio as well as the time consumed by each algorithm to sort all permutations of each size. Decimal values are truncated to 4 places

[a] The permutations of each size were sorted in parallel using a pool of 8 threads

Table 3 Comparison of the percentage of computed distances that are equal to transposition distance, given by different algorithms (WDM [15], M [23], BPwh [34] and DD [9]), in comparison to the EH algorithm and ours

n	WDM	M	BPwh	DD	EH	Alg1
2	–	100.00	100.00	–	100.00	100.00
3	–	100.00	100.00	–	100.00	100.00
4	–	100.00	100.00	100.00	100.00	100.00
5	–	100.00	100.00	100.00	100.00	100.00
6	99.17	100.00	100.00	100.00	99.86	100.00
7	98.58	100.00	100.00	100.00	94.90	95.47
8	97.11	99.69	99.91	100.00	91.64	92.65
9	96.05	99.17	99.72	99.99	86.62	88.54
10	94.12	98.09	–	99.97	83.80	86.53
11	92.81	96.90	–	–	79.40	82.98
12	–	–	–	–	76.67	80.91

Decimal values are truncated to 2 places

(1) There is a (2, 2)-sequence. As stated by Lemma 10, it is not possible to sort $\bar{\pi}$ using a sequence with less than $||\bar{\iota}\bar{\pi}^{-1}||_3$ 2-moves. Let $m = ||\bar{\iota}\bar{\pi}^{-1}||_3 - 2$ be the 3-norm of $\bar{\iota}\bar{\pi}^{-1}$ after the application of a (2, 2)-sequence. Algorithm 1 sorts $\bar{\pi}$ using a maximum of $f(m) + 2$ transpositions, giving an approximation ratio of at most $\frac{f(m)+2}{m+2}$. In Table 1, we can see that, $\frac{f(m)+2}{m+2} \leq \frac{11}{8}$, for all $0 \leq r \leq 7$ such that $m = 8l + r$ and $l \geq 0$.

(2) There is no (2, 2)-sequence. If $||\bar{\iota}\bar{\pi}^{-1}||_3 = 1$, then there is only one oriented 3-cycle in $\bar{\iota}\bar{\pi}^{-1}$. In this case, there is a 2-move and the theorem holds. Otherwise, we can raise the lower bound of Lemma 10 by 1, since at least one 0-move is required to sort $\bar{\pi}$. Let $m = ||\bar{\iota}\bar{\pi}^{-1}||_3$. The approximation ratio given by Algorithm 1 is at most $\frac{f(m)}{m+1}$. Table 1 also shows that, $\frac{f(m)}{m+1} \leq \frac{11}{8}$, for all $0 \leq r \leq 7$ such that $m = 8l + r, l \geq 0$. □

Results and discussion

We implemented Algorithm 1 and the EH algorithm, having tested both using the Rearrangement Distance Database provided by GRAAu [33]. We computed all

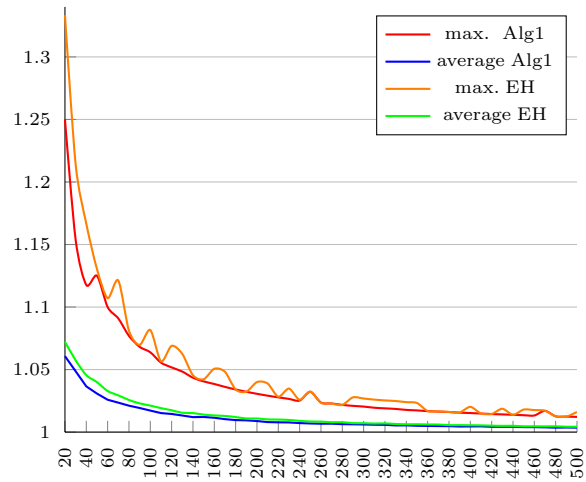

Fig. 1 Average and maximum approximation ratios obtained for each size in our dataset of longer permutations

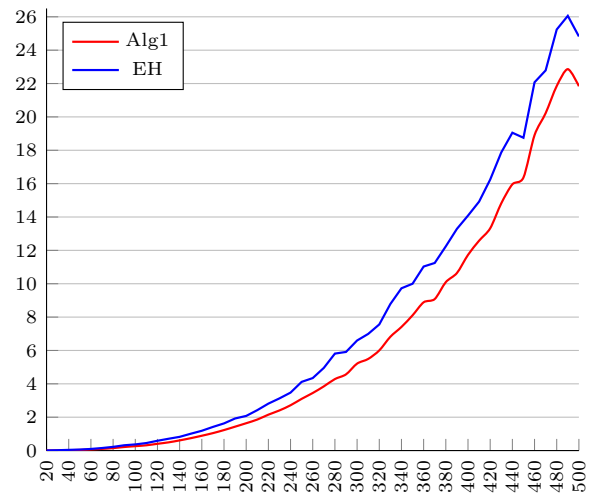

Fig. 2 Time in minutes each algorithm took to sort the all the 1,000 instances of each each size of our dataset of longer permutations

transposition distances using both algorithms for all permutations of size n, $2 \leq n \leq 12$.

As presented by Table 2, the approximation ratio obtained by the EH algorithm exceeds 1.375. On the other hand, our proposed algorithm does not exceed the ratio of $1.3333\overline{3}$. However, we presume that approximations of 1.375 could appear for permutations in S_n, $n \geq 16$, since in order to exist an (11, 8)-sequence, $Supp(\bar{\iota}\bar{\pi}^{-1})$ has to have at least 17 symbols.

We also compared (Table 3) the percentage of computed distances that are equal to transposition distance outputted by our algorithm and EH's with others available in the literature. In particular, we added to the comparison an algorithm with an approximation ratio higher than 1.5, but with good results [34]; one using a similar algebraic approach [23], 1.5-approximation; and another one that also uses an EH-like strategy with an approximation ratio of 1.375 [9].

As shown by Table 3, regarding the percentage of computed distances that are equal to the transposition distance metric, the best algorithm seems to be the algorithm of Dias and Dias [9], although they do not present results for $n > 10$. Importantly, this algorithm employs several heuristics, some introduced by a previous work [35], to improve the performance of the EH algorithm. One of these heuristics is exactly a search for a second 2-move using a look-ahead technique. However, it is not clear whether their heuristic never misses a (2, 2)-sequence, if it exists. Also, Dias and Dias [9] does not state the complexity of their algorithm, but we believe that, by analysing the algorithm [35] which they were based on, the time complexity is higher than $O(n^3)$.

The performance of our algorithm and EH's were also investigated for longer permutations. For this, we created a dataset of longer permutations with sizes ranging from 20 to 500 (incremented by 10). For each of the 49 sets, 1000 instances were randomly generated and sorted using both algorithms. Figure 1 shows the maximum and the average approximation ratios obtained from both ones. It should be noted that the approximation ratios were calculated in relation to the lower bound given by Theorem 33, since it is impracticable to calculate the exact distance for such long permutations. A similar experiment was conducted by Dias and Dias [35], but in their experiment, they worked with smaller sets, also ranging from 20 to 500 (incremented by 10), but containing only 100 instances. By comparing the results, we may conclude that our algorithm and theirs achieve similar results. Dias and Dias [9] also conducted experiments with longer permutations, but with sizes ranging only from 10 to 100 (incremented by 10), where each set contained 100 instances, and collected the running times. By comparing the results presented in their paper, we may conclude that our algorithm performs better than theirs.

Algorithm 1 A new 1.375-approximation algorithm for SBT

function SBT1375($\bar{\pi}$)
  $d \leftarrow 0$
  if there is a $(2,2)$-sequence **then**
    apply a $(2,2)$-sequence
    $d \leftarrow d + 2$
  end if
  while there is an odd cycle in $\bar{\iota}\bar{\pi}^{-1}$ **do**
    apply a 2-move ▷ Proposition 17
    $d \leftarrow d + 1$
  end while
  let Θ be the product of the unmarked cycles of $\bar{\iota}\bar{\pi}^{-1}$
  while $\Theta \neq \iota$ **do**
    if there is a 2-move from an oriented cycle of Θ **then**
      apply a 2-move
      $d \leftarrow d + 1$
    else if there is an even oriented κ-cycle in Θ such that $\kappa \geq 7$ **then**
      apply a $(4,3)$-sequence ▷ Lemma 19
      $d \leftarrow d + 4$
    else
      take a 3-cycle segment γ from a cycle of Θ
      $\Gamma \leftarrow \gamma$
      try to extend Γ eight times
      if Γ is big **then**
        apply a $\frac{11}{8}$-sequence of x 3-cycles ▷ Lemma 22
        $d \leftarrow d + x$
      else if Γ allows a $\frac{11}{8}$-sequence of x 3-cycles **then**
        apply a $\frac{11}{8}$-sequence of x 3-cycles
        $d \leftarrow d + x$
      else
        mark the cycles of Γ ▷ Γ is a bad small component of $\bar{\iota}\bar{\pi}^{-1}$
      end if
    end if
    let Λ be the product of the marked cycles of $\bar{\iota}\bar{\pi}^{-1}$
    if $||\Lambda||_3 \geq 8$ **then**
      unmark the cycles of Λ
      apply a $\frac{11}{8}$-sequence of x 3-cycles ▷ Lemma 24
      $d \leftarrow d + x$
    end if
  end while
  while $\bar{\iota}\bar{\pi}^{-1} \neq \iota$ **do**
    apply a $\frac{3}{2}$-sequence with x 3-cycles ▷ Lemma 20 (can be a 2-move, i.e., a $(1,1)$-sequence, or a $(3,2)$-sequence)
    $d \leftarrow d + x$
  end while
  return d
end function

Figure 2 shows how much time each algorithm (ours and EH's) took to sort all the 1000 instances of each of the 49 sets. The results presented by this figure show that, despite having a high time complexity, our algorithm has good performance in practice, even outperforming EH's.

The dataset of longer permutations used in our experiments, the statistics computed, as well as the source code of the implementation of the EH algorithm and ours are available at [31]. All experiments were executed on a computer equiped with a Core i7 vPro 8^{th} Gen processor, with 4 cores and 8 threads, and 48GB of RAM.

Conclusions

In this paper, we first proposed a new upper bound for the transposition distance, using an algebraic approach, which holds for all S_n. Next, we proposed a new approximation algorithm to solve SBT ensuring the 1.375-approximation ratio for all S_n. To the best of our knowledge, this is the first algorithm guaranteeing an approximation ratio below 1.5 not using simplification. We show in Appendix 1 that the EH algorithm may require one extra transposition above the 1.375-approximation ratio. This occurs when there is a

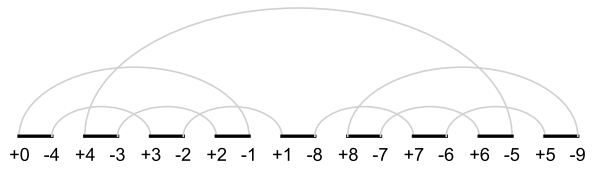

Fig. 3 Cycle graph of [4 3 2 1 8 7 6 5]. The black edges are the horizontal ones

first (2, 2)-sequence in the original permutation that is "missed" during simplification, and bad small components remain in the cycle graph after the application of any number of $\frac{11}{8}$-sequences.

Implementations of the EH algorithm and ours were tested against permutations of maximum length of 12. The results showed that our algorithm does not exceed the 1.375-approximation ratio and produces a higher percentage of computed distances that are equal to transposition distance, when compared to those computed by the EH algorithm. These percentages were also compared to others available in the literature. Considering this metric, the algorithm with the best results seems to be the one of Dias and Dias [9], although they do not present results for $n > 10$. Two other issues were identified when implementing the EH algorithm and they are reported in the Appendix 3. The first one has to do with the application of $\frac{11}{8}$-sequences when the cycle graph contains only bad small components [6]. The second one is related to the application of (3, 2)-sequences when there is no $\frac{11}{8}$-sequence to apply, and affects both versions of the algorithm outlined in [6] and [28].

We conducted an experiment involving longer permutations of maximum length 500. The results showed that our algorithm outperforms the EH algorithm, both in relation to the approximation ratios obtained and running times. Still, on the longer permutations, our algorithm seems to be comparable to the one of Dias and Dias [35], when we consider the approximation ratios obtained by both. Regarding the running times, Dias and Dias [35] also performed some simulations for permutations with a maximum size of 100. Considering only the results for permutations with this maximum size, our algorithm seems faster.

The time complexity of our algorithm is high. A possible future work could be the investigation of a more efficient way to find a (2, 2)-sequence at the beginning of our algorithm. Following a different direction, another future work could be the investigation of "good" simplifications, i.e., simplifications that do not have the effect of missing a (2, 2)-sequence when it exists. We have no clue whether such a "good" simplification always exists or not. In any case, we have the intuition that to find it, if it exists, the computational cost would be the same as searching for a (2, 2)-sequence.

The experiment with small permutations of maximum length 12 showed that the percentages of computed distances by our algorithm that are equal to transposition distance are low compared to others in the literature. A possible way to improve the results would be investigating the adoption of heuristics.

Finally, we intend to use the algebraic approach presented in this paper to study and solve other rearrangement events affecting one chromosome, e.g., reversals and block-interchange.

Appendix 1: Cycle graph and the extra transposition

Cycle graph

Let $\pi = [\pi_1\, \pi_2 \ldots \pi_n]$ be a permutation. A *transposition* $\tau(i,j,k)$, with $1 \leq i < j < k \leq n+1$, "cuts" the symbols from the interval $[\pi_i, \pi_{j-1}]$ and then "pastes" them right after π_{k-1}. Thus, the application of $\tau(i,j,k)$ on π, denoted $\tau(i,j,k) \cdot \pi$, yields $[\pi_1 \ldots \pi_{i-1}\, \pi_j \ldots \pi_{k-1}\, \pi_i \ldots \pi_{j-1}\, \pi_k \ldots \pi_n]$.

Given two permutations π and σ, the TRANSPOSITION DISTANCE PROBLEM (TDP) corresponds to finding the minimum t (the transposition distance between π and σ) such that the sequence of transpositions $\tau_1, \cdots, \tau_t$ transforms π into σ i.e., $\tau_t \ldots \tau_1 \cdot \pi = \sigma$. Note that the transposition distance between π and σ equals the transposition distance between $\sigma^{-1} \circ \pi$ and the identity permutation $\iota = [1\, 2 \ldots n]$. The problem of SORTING BY TRANSPOSITIONS (SBT) is the problem of finding the transposition distance between a permutation π and ι, denoted by $d(\pi)$.

In the genome rearrangements literature, a widely used graphical representation for a permutation is the cycle graph[5] [5]. In order to construct the cycle graph of $\pi = [\pi_1\, \pi_2\, \ldots\, \pi_n]$, we first extend π by adding two extra elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$. So, the *cycle graph* of π, denoted by $G(\pi)$, is a directed graph consisting of a set of vertices $\{+0, -1, +1, -2, +2, \ldots, -n, +n, -(n+1)\}$ and a set of colored (black or gray) edges. For all $1 \leq i \leq n+1$, the black edges connect $-\pi_i$ to $+\pi_{i-1}$. For $0 \leq i \leq n$, the gray edges connect vertex $+i$ to vertex $-(i+1)$. Intuitively, the black edges indicate the current state of the genes, related to their arrangement in the first chromosome represented by π, while the gray edges indicate the desired order of the genes in the second permutation, represented by $\iota = [1\, 2\, \ldots\, n]$. In

[5] In their work, Elias and Hartman [6] use an equivalent circular representation, which they call breakpoint graph.

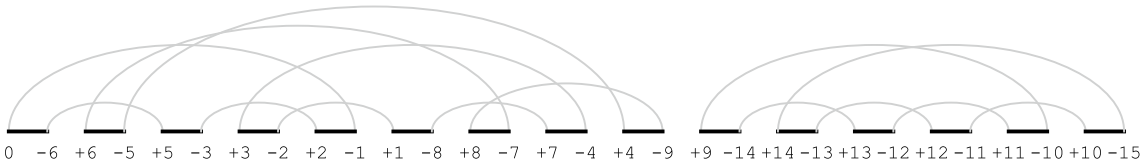

Fig. 4 Cycle graph of [6 5 3 2 1 8 7 4 9 14 13 12 11 10]

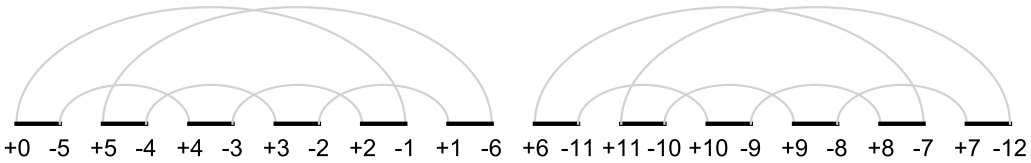

Fig. 5 Cycle graph of [5 4 3 2 1 6 11 10 9 8 7], mapped from [4.1 4 3 2 1 4.2 8.1 8 7 6 5] using consecutive integers, obtained by the simplification of [4 3 2 1 8 7 6 5] (Figure 3)

the figures below, the directions of the edges are omitted since they can be easily inferred by observing the signs of the vertices.

Example 31 Figure 3 shows the cycle graph of $\pi = [4\ 3\ 2\ 1\ 8\ 7\ 6\ 5]$ with 9 black edges, $(-9,+5)$, $(-5,+6)$, $\cdots$, $(-3,+4)$, $(-4,+0)$, and 9 gray edges, $(+0,-1),(+1,-2),(+2,-3),\cdots,(+7,-8),(+8,-9)$.

Both in-degree and out-degree of each vertex in $G(\pi)$ are 1, corresponding to one black edge entering a vertex v and another gray edge leaving v. This induces in $G(\pi)$ a unique decomposition into cycles. A κ-cycle is a cycle C in $G(\pi)$ with κ black edges. In addition, C is said to be a *long cycle*, if $k > 3$, otherwise, C is said to be a *short cycle*. If κ is even (odd), then we also say that C is an *even* (*odd*) *cycle*.

The maximum number of $n+1$ cycles in $G(\pi)$ is obtained if and only if π is the identity permutation ι. In this case, each cycle is composed of exactly one black edge and one gray edge. Let us denote by $c_{odd}(\pi)$ the number of odd cycles in $G(\pi)$, and $\Delta c_{odd}(\pi,\tau) = c_{odd}(\tau\cdot\pi) - c_{odd}(\pi)$ the variation on the number of odd cycles in $G(\pi)$ and $G(\tau\cdot\pi)$, after the *application* of a transposition τ. Bafna and Pevzner [5] demonstrated the following result.

Lemma 32 (Bafna and Pevzner [5]) $\Delta c_{odd}(\pi,\tau) \in \{-2,0,2\}$.

A μ-*move* is a transposition τ such that $\Delta c_{odd}(\pi,\tau) = \mu$. Note that according to lemma above, the possible moves are 2-move, 0-move and (-2)-move. From Lemma 32, Bafna and Pevzner [5] derived the following lower bound.

Theorem 33 (Bafna and Pevzner [5]) $d(\pi) \geq \frac{n+1-c_{odd}(\pi)}{2}$

The black edges of $G(\pi)$ can be numbered from 1 to $n+1$ by assigning a label i to each black edge $(-\pi_i,+\pi_{i-1})$. A κ-cycle C visiting the black edges $i_1,\ldots,i_\kappa$, in the order imposed by the cycle, can be written in κ different ways, depending on the first black edge visited. If not otherwise specified, we will assume that the initial edge i_1 of C is chosen as the greatest value, i.e., i_1 is such that $i_1 > i_s$, for all $s \in \{2,\ldots,\kappa\}$. With this condition, if $i_1,\cdots,i_\kappa$ is a decreasing sequence, C is called an *unoriented* cycle; otherwise C is *oriented*. Two pairs of black edges are said *intersecting* if there are cycles $C = (\ldots,a,b,\ldots)$ and $D = (\ldots,e,f,\ldots)$ in $G(\pi)$ such that either $a > e > b > f$ or $e > a > f > b$. In this case, C and D are also said to be *intersecting cycles*. Similarly, the triplets of black edges (a, b, c) and (d, e, f) are *interleaving* if there are cycles $C = (\ldots,a,b,c,\ldots)$ and $D = (\ldots,d,e,f,\ldots)$ such that either $a > d > b > e > c > f$ or $d > a > e > b > f > c$. In such case, C and D are also said to be *interleaving cycles*.

Example 34 The cycles (5, 3, 1), (8, 6, 4), (15, 13, 11) and (14, 12, 10) of $G([6\ 5\ 3\ 2\ 1\ 8\ 7\ 4\ 9\ 14\ 13\ 12\ 11\ 10])$ (Fig. 4) are unoriented, while (9, 2, 7) is oriented. Furthermore, (5, 3, 1) and (8, 6, 4) are intersecting and the cycles (15, 13, 11) and (14, 12, 10) are interleaving.

$$\tau(4,6,9)\cdot[4\ 3\ 2\ \underline{1\ 8}\ \underline{\underline{7\ 6\ 5}}] = [4\ 3\ 2\ 7\ 6\ 5\ 1\ 8]$$
$$\tau(3,5,8)\cdot[4\ 3\ \underline{2\ 7}\ \underline{\underline{6\ 5\ 1}}\ 8] = [4\ 3\ 6\ 5\ 1\ 2\ 7\ 8]$$
$$\tau(2,4,7)\cdot[4\ \underline{3\ 6}\ \underline{\underline{5\ 1\ 2}}\ 7\ 8] = [4\ 5\ 1\ 2\ 3\ 6\ 7\ 8]$$
$$\tau(1,3,6)\cdot[\underline{4\ 5}\ \underline{\underline{1\ 2\ 3}}\ 6\ 7\ 8] = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$$

Fig. 6 Sorting $\pi = [4\ 3\ 2\ 1\ 8\ 7\ 6\ 5]$ with 4 transpositions

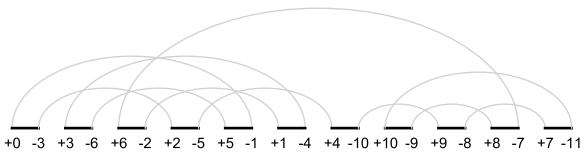

Fig. 7 Cycle graph of $[3\ 6\ 2\ 5\ 1\ 4\ 10\ 9\ 8\ 7]$

$$\tau(6,8,11)\cdot[3\ 6\ 2\ 5\ 1\ \underline{4\ 10}\ \underline{9\ 8\ 7}] = [3\ 6\ 2\ 5\ 1\ 9\ 8\ 7\ 4\ 10]$$
$$\tau(5,7,10)\cdot[3\ 6\ 2\ 5\ \underline{1\ 9}\ \underline{\underline{8\ 7\ 4}}\ 10] = [3\ 6\ 2\ 5\ 8\ 7\ 4\ 1\ 9\ 10]$$
$$\tau(3,6,9)\cdot[3\ 6\ \underline{2\ 5\ 8}\ \underline{\underline{7\ 4\ 1}}\ 9\ 10] = [3\ 6\ 7\ 4\ 1\ 2\ 5\ 8\ 9\ 10]$$
$$\tau(2,4,8)\cdot[3\ \underline{6\ 7}\ \underline{\underline{4\ 1\ 2\ 5}}\ 8\ 9\ 10] = [3\ 4\ 1\ 2\ 5\ 6\ 7\ 8\ 9\ 10]$$
$$\tau(1,3,5)\cdot[\underline{3\ 4}\ \underline{\underline{1\ 2\ 5}}\ 6\ 7\ 8\ 9\ 10] = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$$

Fig. 8 Sorting $\pi' = [3\ 6\ 2\ 5\ 1\ 4\ 10\ 9\ 8\ 7]$ with 5 transpositions

Simplification

Simplification is a technique introduced aiming to facilitate handling with long cycles of $G(\pi)$ [24]. It consists of inserting new elements, usually fractional numbers, into π transforming it into a new *simple permutation* $\hat{\pi}$, so that $G(\hat{\pi})$ contains only short cycles. After the transformation, the elements of $\hat{\pi}$ can be mapped to consecutive integers. The positions of the new symbols can vary, but the insertion must be through *safe transformations*.

A transformation of π into $\hat{\pi}$ is said to be *safe* if, after the insertion of the new elements, the lower bound of Theorem 33 is maintained, i.e., $n(\pi) - c_{odd}(\pi) = n(\hat{\pi}) - c_{odd}(\hat{\pi})$, where $n(\pi)$ and $n(\hat{\pi})$ denote the number of black edges in π and $\hat{\pi}$, respectively. If $\hat{\pi}$ is a permutation obtained from π through safe transformations, then we say π and $\hat{\pi}$ are *equivalent*. Lin and Xue [25] have shown that every permutation can be transformed into an equivalent simple one through safe transformations. A sorting of $\hat{\pi}$ can be mimicked to sort π using the same number of transpositions [24].

It is important to note that a permutation can be simplified in many different ways. Figure 5 shows the cycle graph of a possible simple permutation obtained by the simplification of [4 3 2 1 8 7 6 5] (Fig. 3). For a complete description of simplification and related results, the reader is referred to [24–26].

Configurations and components

The concepts presented in this section were originally introduced by Elias and Hartman [6] in the context of the simple permutations, with a special focus on the 3-permutations, from which they derived their main results. As our work does not involve simplification, we modified some of them so that they could be extended to any permutation in S_n and also to facilitate the correlation between the algebraic approach used in this paper with the method of Elias and Hartman [6].

A *configuration* of cycles is a subgraph of $G(\pi)$ induced by one or more cycles. A configuration A is *connected*, if for any two cycles C_1 and C_m of A, there are cycles $C_2, \ldots, C_{m-1}$ in A such that for each $i \in [1, m-1]$, C_i intersects or interleaves with C_{i+1}. A *component* is a configuration consisting of only one oriented cycle that does not intersect or interleave any other cycle of $G(\pi)$; or consisting of a maximal connected configuration in $G(\pi)$.

Let A be a configuration induced only by odd cycles. The *3-norm* of A, denoted by $||A||$, is the value $\frac{b-c(A)}{2}$, where b is the number of black edges of A and $c(A)$ is the number of cycles in A. If $||A|| \leq 8$, then A is referred as being *small*; otherwise, *big*. The 3-norm concept was not defined in Elias and Hartman [6]. The intuition behind it is that it reflects the number of 3-cycles a configuration containing cycles of arbitrary (odd) lengths would have if it were "simplified".

Example 35 The 3-norm of the configuration $\{(9, 6, 8, 2, 4, 1, 3, 5, 7)\}$ from $G([4\ 3\ 2\ 1\ 8\ 7\ 6\ 5])$ (Fig. 3) is 4 and, consequently, it is a small configuration.

Example 36 The 3-norms of the configurations $\{(7, 4, 1), (8, 5, 2), (9, 6, 3)\}$ and $\{(14, 12, 10), (15, 13, 11)\}$ from $G([4\ 8\ 3\ 7\ 2\ 6\ 1\ 5\ 9\ 14\ 13\ 12\ 11\ 10])$ (Fig. 9) are 3 and 2, respectively.

An *open gate* is a pair of black edges (a, b) of a cycle C in A, such that one of its cyclic forms is $C = (a, b, \ldots)$, that does not intersect with any other cycle in A and there is no black edge c in C, such that, if $a > b$, then a, b, c is not a decreasing sequence; or, if $b > a$, then b, c, a is not a decreasing sequence either. A configuration not containing open gates is called *full configuration*.

Example 37 The configurations $\{(7, 4, 1), (8, 5, 2), (9, 6, 3)\}$ and $\{(14, 12, 10), (15, 13, 11)\}$ are small full components of $G([4\ 8\ 3\ 7\ 2\ 6\ 1\ 5\ 9\ 14\ 13\ 12\ 11\ 10])$ (Fig. 9).

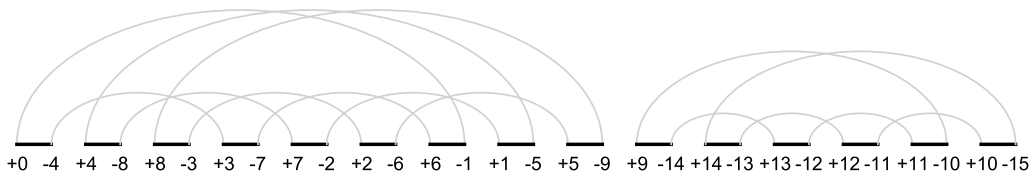

Fig. 9 Cycle graph of [4 8 3 7 2 6 1 5 9 14 13 12 11 10], mapped from $\hat{\pi}' =$ [3.1 6.1 3 6 2 5 1 4 6.2 10.1 10 9 8 7] using consecutive integers

Sequences of transpositions

A sequence of transpositions $\tau_1, \cdots, \tau_x$ is said to be a (x, y)-*sequence*, for $x \geq y$, is a sequence of x transpositions such that, at least y of them are 2-moves. A (x, y)-sequence is an $\frac{a}{b}$-*sequence* if $\frac{x}{y} \leq \frac{a}{b}$ and $x \leq a$.

Example 38 The sequence $\tau_1 = \tau(1,4,7)$, $\tau_2 = \tau(2,5,8)$, $\tau_3 = \tau(1,4,7)$, $\tau_4 = \tau(3,6,9)$ is a (4, 3)-sequence, which is also a $\frac{11}{8}$-sequence, for [4 8 3 7 2 6 1 5 9 14 13 12 11 10] (Fig. 9).

The extra transposition

The first step of the EH algorithm is the simplification of the input permutation. In this section, we show that there are simplifications that, although producing equivalent simple permutations, causes the EH algorithm to require one extra transposition above the approximation of 1.375. Two examples are explored next.

Consider the permutation $\pi =$ [4 3 2 1 8 7 6 5] shown in Fig. 3. The lower bound given by Theorem 33 is 4, also its exact distance, corresponding to the application of four 2-moves, shown in Fig. 6. One simplification of π generates the permutation [4.1 4 3 2 1 4.2 8.1 8 7 6 5], which mapped to consecutive integers is $\hat{\pi} =$ [5 4 3 2 1 6 11 10 9 8 7] (Fig. 5). Note that the lower bound of $\hat{\pi}$ is 4 as well. However, there is no $\frac{11}{8}$-sequence to apply on $\hat{\pi}$. In fact, to optimally sort $\hat{\pi}$, two (3, 2)-sequences are required. Therefore the EH algorithm using $\pi =$ [4 3 2 1 8 7 6 5] as input, even applying an optimal sorting on $\hat{\pi} =$ [5 4 3 2 1 6 11 10 9 8 7], yields 6 transpositions. However, the algorithm should require at most 5 transpositions to not exceed the 1.375-approximation ratio.

The following example shows that, even if there are $\frac{11}{8}$-sequences of transpositions to apply on $\hat{\pi}$, the EH algorithm may require one transposition above the approximation ratio of 1.375. Take the permutation $\pi' =$ [3 6 2 5 1 4 10 9 8 7] (Fig. 7), with both the lower bound and distance equal to 5, corresponding to the application of five 2-moves, shown in Fig. 8. A simplified version of π' is [3.1 6.1 3 6 2 5 1 4 6.2 10.1 10 9 8 7], which mapped to consecutive integers is $\hat{\pi}' =$ [4 8 3 7 2 6 1 5 9 14 13 12 11 10] (Fig. 9). The EH algorithm sorts $\hat{\pi}'$ optimally by applying a (4, 3)-sequence, followed by a (3, 2)-sequence, in a total of 7 transpositions. However, the algorithm should not require more than 6 transpositions to not exceed the 1.375-approximation ratio.

In both examples above, an initial (2, 2)-sequence is "missed" during the simplification process. This sequence is essential to guarantee the 1.375 approximation ratio when *bad small components* remain in $G(\hat{\pi})$ after the application of a number of $\frac{11}{8}$-sequences (Theorem 22 [6]). These are small full configurations which do not allow the application of $\frac{11}{8}$-sequences. It is important to stress that the extra transposition will be necessary regardless of the number of bad small components remaining in the cycle graph after applying a sequence of $\frac{11}{8}$-sequences (any number of), as long as the total number of remnant 3-cycles is less than 8 and the initial (2, 2)-sequence that possibly existed initially, was "missed" during the simplification.

It was already known by the literature that simplification maintained the lower bound, but not the transposition distance. However, it was not known that the simplification could have the effect of missing an initial (2, 2)-sequence. In principle, the EH algorithm could be modified to guarantee the 1.375-approximation ratio, and no extra transposition, by looking for the (2, 2)-sequence in its first step, applying it case it exists, and only then simplifying the resulting permutation. However, using the already known techniques, this new "modified" EH algorithm would not keep the original time complexity of $O(n^2)$.

Appendix 2: Correspondence between the cycle graph and $\bar{\iota}\bar{\pi}^{-1}$

The product $\bar{\iota}\bar{\pi}^{-1}$, in the algebraic approach, produces cycles corresponding exactly to the same cycles of $G(\pi)$. If we follow the edges of the cycles in $G(\pi)$ taking note

of the labels of the vertices where the gray edges enter, disregarding the sign, and changing the label $-(n+1)$ to 0, we obtain exactly the same cycles of $\bar{\iota}\bar{\pi}^{-1}$. It is easy to see, therefore, that $\bar{\iota}\bar{\pi}^{-1}$ and $G(\pi)$ have the same number of cycles and the corresponding cycles have all the same length. Also, the cycles of $\bar{\iota}\bar{\pi}^{-1}$ have the same relevant properties of the cycles of $G(\pi)$, such as orientation. The relationships between the cycles, i.e., intersection and interleaving, are identical as well. Furthermore, configurations and components are also corresponding concepts between the two structures (see section "Configurations and components" in Appendix 1).

Example 39 Let $\pi = [6\ 5\ 3\ 2\ 1\ 8\ 7\ 4\ 9\ 14\ 13\ 12\ 11\ 10]$ ($G(\pi)$ depicted in Fig. 4). As seen in Example 34, the cycles of $G(\pi)$ are (5, 3, 1), (8, 6, 4), (9, 2, 7), (14, 12, 10) and (15, 13, 11). Now let $\bar{\pi} = (\,0\ 6\ 5\ 3\ 2\ 1\ 8\ 7\ 4\ 9\ 14\ 13\ 12\ 11\ 10\,)$, so that $\bar{\iota}\bar{\pi}^{-1} = (\,0\ 11\ 13\,)(\,1\ 3\ 6\,)(\,2\ 4\ 8\,)(\,5\ 7\ 9\,)(\,10\ 12\ 14\,)$. Take the cycle (5, 3, 1) of $G(\pi)$. If we follow the procedure above, we obtain the $\bar{\iota}\bar{\pi}^{-1}$ cycle $(\,1\ 3\ 6\,)$. The same procedure takes (8, 6, 4) to $(\,2\ 4\ 8\,)$, (9, 2, 7) to $(\,5\ 7\ 9\,)$ (note that these cycles are equally oriented), (14, 12, 10) to $(\,10\ 12\ 14\,)$; and finally (15, 13, 11) to $(\,0\ 11\ 13\,)$. Also, observe that (5, 3, 1) and (8, 6, 4) are intersecting, the same way as $(\,1\ 3\ 6\,)$ and $(\,2\ 4\ 8\,)$. Furthermore, the pairs (14, 12, 10) and (15, 13, 11); and $(\,10\ 12\ 14\,)$ and $(\,0\ 11\ 13\,)$ are interleaving. Finally, the same pairs of cycles form small components in both structures.

Appendix 3: Other issues found in the EH algorithm

It should be noted that Elias and Hartman [6] have not provided a publicly available implementation of their algorithm, which we could use as a reference. To the best of our knowledge, the only implementation of the EH algorithm reported in the literature, without the use of heuristics, is the one of Dias and Dias [35], but their implementation is not publicly available either. This led us to implement the EH algorithm from scratch.

It is worthy of note that our implementation of the EH algorithm in closer to the version[6] previously presented on WABI in 2005 [28], since we found an issue in the algorithm outlined in [6] (the algorithms are presented differently in both versions of their work). The issue has to do with the application of $\frac{11}{8}$-sequences when $G(\hat{\pi})$ contains only bad small components. As presented on [6], once all bad small components are identified, the algorithm enters a loop and continuously applies $\frac{11}{8}$-sequences (given by their Lemma 17 [6]), until the number of cycles in $G(\hat{\pi})$ is less than 8. However, we found cases where the application of $\frac{11}{8}$-sequences given by Lemma 17 [6] can create small components in $G(\hat{\pi})$ that are not bad, which can eventually prevent the application of the lemma in the next iterations. One such case is when we have a permutation consisting of multiple unoriented necklaces of size 6 [6] side by side. To give an illustration, take $\hat{\pi} = [17\ 16\ 3\ 2\ 1\ 6\ 5\ 4\ 9\ 8\ 7\ 12\ 11\ 10\ 15\ 14\ 13\ 18\ 35\ 34\ 21\ 20\ 19\ 24\ 23\ 22\ 27\ 26\ 25\ 30\ 29\ 28\ 33\ 32\ 31]$ whose $G(\hat{\pi})$ consists precisely of two unoriented necklaces of size 6 side by side. Since the sum of 3-cycles is 12, Lemma 17 [6] guarantees us the existence of a $\frac{11}{8}$-sequence. The (11, 8)-sequence given by Elias and Hartman [6] for this permutation (by combining two unoriented necklaces of size 6 side by side) is $\tau_1 = \tau(1,3,5)$, $\tau_2 = \tau(7,11,26)$, $\tau_3 = \tau(9,13,35)$, $\tau_4 = \tau(4,10,34)$, $\tau_5 = \tau(2,13,30)$, $\tau_6 = \tau(1,18,20)$, $\tau_7 = \tau(6,17,32)$, $\tau_8 = \tau(5,14,22)$, $\tau_9 = \tau(15,27,35)$, $\tau_{10} = \tau(18,28,36)$, $\tau_{11} = \tau(6,19,35)$. After applying this sequence, we have $\hat{\pi} = [1\ 2\ 3\ 4\ 20\ 21\ 22\ 27\ 26\ 25\ 30\ 31\ 32\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 24\ 23\ 5\ 6\ 7\ 8\ 9\ 10\ 29\ 28\ 33\ 34\ 35]$. Observe that now $G(\hat{\pi})$ contains a small component of 4 unoriented 3-cycles that despite being small, is not bad.

To avoid the issue described above in our implementation of the EH algorithm, we have made a change in which we apply a $\frac{11}{8}$-sequence as soon as the sum of the number of 3-cycles of the the bad small components, as they are identified in the main loop, is greater than 7, inside the loop itself (line 5 of the algorithm outlined in [6]), as opposed to its position within a loop of its own (line 6 [6]). Similar solution is employed by our Algorithm 1 (line 34).

We found another issue in the last loop of both versions of the EH algorithm [6, 28]. It is not always possible to apply a (3, 2)-sequence at that point. Sometimes, only a 2-move exists, as the Lemma 7 [6] itself states. Take, for instance, the permutation $\hat{\pi} = [14\ 13\ 3\ 2\ 1\ 6\ 5\ 4\ 9\ 8\ 7\ 12\ 11\ 10]$ whose $G(\hat{\pi})$ consists of an unoriented necklace of size 5. Observe that there is no $\frac{11}{8}$-sequence to apply on $\hat{\pi}$. In the last loop [6, 28], Elias and Hartman [6] applies two (3, 2)-sequences: $\tau_1 = \tau(1,10,14)$, $\tau_2 = \tau(4,6,15)$, $\tau_3 = \tau(3,5,14)$, then $\tau_1 = \tau(4,8,9)$, $\tau_2 = \tau(2,5,8)$, $\tau_3 = \tau(1,3,6)$. After applying these sequences, we have $\hat{\pi} = [1\ 6\ 7\ 8\ 2\ 3\ 4\ 5\ 9\ 10\ 11\ 12\ 13\ 14]$ whose $G(\hat{\pi})$ contains only one oriented 3-cycle, making it impossible to apply a further (3, 2)-sequence. In this particular case, the 2-move $\tau(2,5,9)$ concludes the sorting of $\hat{\pi}$. Our implementation [31] of the EH algorithm includes a "fix" for this issue applying a (3, 2)-sequence or a 2-move, depending on the case.

[6] As this version uses one single loop to apply $\frac{11}{8}$-sequences.

Acknowledgements
The authors kindly thank Isaac Elias for the invaluable discussion. They also thank Annachiara Korchmaros, and the anonymous reviewers, whose comments helped to improve this manuscript. MEMTW thanks CNPq for the fellowship (Project 310785/2018-9). LAGS thanks CAPES for the doctoral scholarship (Grant 88887.639024/2014-01).

Authors' contributions
First draft: LAGS, MEMTW, NRR. Proofs and algorithm implementation: LAGS. Final manuscript: LAGS, LABK, MEMTW. All authors read and approved the final manuscript.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Departmento de Ciência da Computação, Universidade de Brasília, Brasília, Brazil. [2]Departmento de Matemática, Universidade de Brasília, Brasília, Brazil. [3]Instituto de Computação, Universidade Federal Fluminense, Niterói, Brazil.

Received: 3 February 2021 Accepted: 2 January 2022
Published online: 15 January 2022

References

1. Nadeau JH, Taylor BA. Lengths of chromosomal segments conserved since divergence of man and mouse. Proc Natl Acad Sci USA. 1984;81(3):814–8.
2. Palmer JD, Herbon LA. Plant mitochondrial dna evolves rapidly in structure, but slowly in sequence. J Mol Evol. 1988;28:87–97.
3. Koonin EV. Orthologs, paralogs, and evolutionary genomics. Ann Rev Genetics. 2005;39:309–38.
4. Yue F, Zhang M, Tang J. Phylogenetic reconstruction from transpositions. BMC Genomics. 2008;9(S15):10–1186147121649215.
5. Bafna V, Pevzner PA. Sorting by transpositions. SIAM J Discret Math. 1998;11(2):224–40.
6. Elias I, Hartman T. A 1.375-approximation algorithm for sorting by transpositions. IEEE/ACM Trans Comput Biol Bioinf. 2006;3(4):369–79.
7. Bulteau L, Fertin G, Rusu I. Sorting by transpositions is difficult. SIAM J Discret Math. 2012;26(3):1148–80.
8. Cunha LF, Kowada LA, Hausen RD, de Figueiredo CM. A faster 1.375-approximation algorithm for sorting by transposition. J Comput Biol. 2015;22(11):1044–56.
9. Dias U, Dias Z. An improved 1375-approximation algorithm for the transposition distance problem. In: Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology, 2010; pp. 334–337.
10. Dias U, Dias Z. Heuristics for the transposition distance problem. J Bioinform Comput Biol. 2013;11(5):1–17.
11. Hausen RA, Faria L, Figueiredo CMH, Kowada LAB. Unitary toric classes, the reality and desire diagram, and sorting by transpositions. SIAM J Discrete Math. 2010;24(3):792–807.
12. Eriksson H, Eriksson K, Karlander J, Svensson L, Wästlund J. Sorting a bridge hand. Discret Math. 2001;241(1–3):289–300.
13. Galvão G, Dias Z. On the approximation ratio of algorithms for sorting by transpositions without using cycle graphs. In: BSB, Germany: Springer. 2012; pp. 25–36.
14. Benoît-Gagné M, Hamel S. A new and faster method of sorting by transpositions. In: Annual Symposium on Combinatorial Pattern Matching. Germany: Springer. 2007; pp. 131–141.
15. Walter MEMT, Dias Z, Meidanis J. A new approach for approximating the transposition distance. In: Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00). SPIRE '00, p. 199. IEEE Computer Society, Washington, DC, USA. 2000. http://dl.acm.org/citation.cfm?id=829519.830850
16. Guyer SA, Heath LS, Vergara JPC. Subsequence and run heuristics for sorting by transpositions. Technical report: Virginia Polytechnic Institute & State University; 1997.
17. Rusu I. log-lists and their applications to sorting by transpositions, reversals and block-interchanges. Theoret Comput Sci. 2017;660:1–15.
18. Sleator DD, Tarjan RE. A data structure for dynamic trees. J Comput Syst Sci. 1983;26(3):362–91.
19. Lintzmayer CN, Fertin G, Dias Z. Sorting permutations by prefix and suffix rearrangements. J Bioinform Comput Biol. 2017;15(01):1750002.
20. Oliveira AR, Jean G, Fertin G, Brito KL, Dias U, Dias Z. A 3.5-approximation algorithm for sorting by intergenic transpositions. In: International Conference on Algorithms for Computational Biology. Berlin: Springer. 2020;16–28.
21. Meidanis J, Dias Z. An Alternative Algebraic Formalism for Genome Rearrangements. In: Sankoff D, Nadeau JH, eds. Springer, Dordrecht. 2000; pp. 213–223.
22. Mira CVG, Meidanis J. Algebraic formalism for genome rearrangements. Technical Report, Institute of Computing, University of Campinas. 2005.
23. Mira CVG, Dias Z, Santos HP, Pinto GA, Walter MEMT. Transposition distance based on the algebraic formalism. In: Advances in Bioinformatics and Computational Biology. Proceedings of the Third Brazilian Symposium on Bioinformatics. Berlin Heidelberg, Germany: Springer; 2008. p. 115–26.
24. Hannenhalli S, Pevzner PA. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. J ACM. 1999;46(1):1–27.
25. Lin GH, Xue G. Signed genome rearrangement by reversals and transpositions: models and approximations. Theoret Comput Sci. 2001;259(1):513–31.
26. Hartman T, Shamir R. A simpler and faster 1.5-approximation algorithm for sorting by transpositions. Inf Comput. 2006;204(2):275–90.
27. Fertin G, Labarre A, Rusu I, Vialette S, Tannier E. Combinatorics of Genome Rearrangements. London, En: MIT press; 2009.
28. Elias I, Hartman T. A 1.375-approximation algorithm for sorting by transpositions. In: International Workshop on Algorithms in Bioinformatics. Germany: Springer. 2005; pp. 204–215.
29. Dummit DS, Foote RM. Abstract Algebra. Hoboken, NJ: Wiley; 2004.
30. Gallian J. Contemporary abstract algebra. 7th ed. Boston, MA: Brooks Cole; 2009.
31. https://github.com/luizaugustogarcia/tdp1375/; 2020.
32. http://tdp1375proof.s3-website.us-east-2.amazonaws.com/; 2020.
33. Galvão GR, Dias Z. An audit tool for genome rearrangement algorithms. J Exp Algorithmics (JEA). 2015;19:1–7.
34. Walter MEMT, Sobrinho MC, Oliveira ETG, Soares LS, Oliveira AG, Martins TES, Fonseca TM. Improving the algorithm of bafna and pevzner for the problem of sorting by transpositions: a practical approach. J Discrete Algorithms. 2005;3(2):342–61.
35. Dias U, Dias Z. Extending Bafna-Pevzner algorithm. In: Proceedings of the International Symposium on Biocomputing. ISB '10. ACM, New York, NY. 2010. pp. 23–1238.

Publisher's Note