Theoretical Computer Science 1024 (2025) 114909

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Sorting genomes by prefix double-cut-and-joins ☆

Guillaume Fertin [a], Géraldine Jean [a,*], Anthony Labarre [b]

[a] Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France
[b] Laboratoire d'Informatique Gaspard Monge, Université Gustave Eiffel, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-77454, Marne-la-Vallée, France

ARTICLE INFO

Keywords:
Genome rearrangements
Prefix reversals
Prefix DCJs
Lower bounds
Algorithmics
Approximation algorithms

ABSTRACT

In this paper, we study the problem of sorting unichromosomal linear genomes by prefix double-cut-and-joins (or DCJs) in both the signed and the unsigned settings. Prefix DCJs cut the leftmost segment of a genome and any other segment, and recombine the severed endpoints in one of two possible ways: one of these options corresponds to a prefix reversal, which reverses the order of elements between the two cuts (as well as their signs in the signed case). Our main results are: (1) new structural lower bounds based on the breakpoint graph for sorting by unsigned prefix reversals, unsigned prefix DCJs, and signed prefix DCJs; (2) two polynomial-time algorithms for sorting by prefix DCJs, both in the signed case (which answers an open question of Labarre [1]) and in the unsigned case; (3) a 1-absolute approximation algorithm for sorting by unsigned prefix reversals for a specific class of permutations.

1. Introduction

Genome rearrangements is a classical paradigm for studying evolution between species. The rationale is to consider species by observing their genomes, which are usually represented as ordered sets of elements (the genes) that can be signed (according to gene orientation, when known). A genome can then evolve by changing the order of its genes, through operations called *rearrangements*, which can be generally described as cutting the genome at different locations, thus forming segments, and rearranging these segments in a different fashion. Given two genomes, a *sorting scenario* is a sequence of rearrangements transforming the first genome into the other. The length of a shortest such sequence of rearrangements is called the *rearrangement distance*. Several specific rearrangements such as reversals, translocations, fissions, fusions, transpositions, and block-interchanges have been defined, and the rearrangement distance together with its corresponding sorting problem have been widely studied either by considering one unique type of rearrangement or by allowing the combination of some of them [2]. The *double-cut-and-join* (or DCJ) operation introduced by Yancopoulos et al. [3] encompasses all the above-mentioned rearrangements: it consists in cutting the genome in two different places and joining the four resulting extremities in any possible way. A DCJ is a *prefix DCJ* whenever one cut is applied to the leftmost position of the genome. The prefix restriction can be applied to other rearrangements such as *prefix reversals*, which prefix DCJs generalise. Whereas the computational complexity of the sorting problems by unrestricted rearrangements has been thoroughly studied and pretty well characterised, there is still a lot of work to do to understand the corresponding prefix sorting problems (see Table 1 in [1] for a summary of existing results). Our interest in prefix rearrangements is therefore mostly theoretical: techniques that apply in the un-

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

* Corresponding author.
E-mail addresses: guillaume.fertin@univ-nantes.fr (G. Fertin), geraldine.jean@univ-nantes.fr (G. Jean), Anthony.Labarre@univ-eiffel.fr (A. Labarre).

https://doi.org/10.1016/j.tcs.2024.114909
Received 25 October 2023; Received in revised form 30 September 2024; Accepted 9 October 2024

Available online 15 October 2024

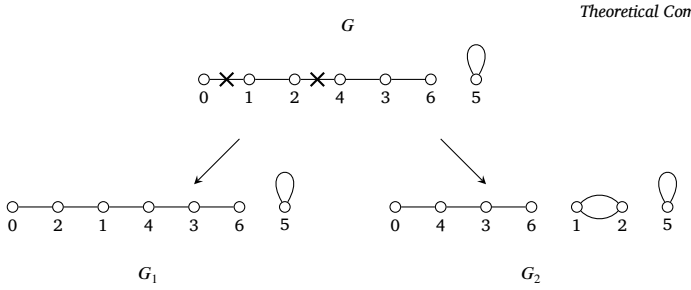

Fig. 1. Example of a prefix DCJ acting on a genome G, showing the two possible ways to join after the cut takes place. Cutting edges $\{0,1\}$ and $\{2,4\}$ from the nonlinear genome G produces genome G_1 with a reversed segment, if we add edges $\{0,2\}$ and $\{1,4\}$, or genome G_2 with an extracted cycle if we add $\{0,4\}$ and $\{1,2\}$ instead.

restricted setting do not directly apply under the prefix restriction, and new approaches are therefore needed to make progress on algorithmic issues and complexity aspects. Since DCJs generalise several other operations, we hope that the insight we gain through their study will shed light on other prefix rearrangement problems.

In this paper, we study the problem of Sorting by Prefix DCJs and, for the sake of simplicity, we consider the case where the source and the target genomes are unichromosomal and linear. This implies that genomes can be seen as (signed) permutations (depending on whether the gene orientation is known or not), although some of the algorithms we design are able to handle more general structures. Moreover, prefix DCJs applied to such genomes allow to exactly mimick three kinds of rearrangement: (i) a *prefix reversal* when the segment between the two cuts is reversed; (ii) a *cycle extraction* when the extremities of the segment between the two cuts are joined; (iii) a *cycle reincorporation* when the cut occurs in a cycle and the resulting linear segment is reincorporated at the beginning of the genome where the leftmost cut occurs.

This paper, which is an extended version of [1], is organised as follows. Based on the study of the *breakpoint graph*, we first show new structural lower bounds for the problems Sorting by Signed Prefix DCJs and Sorting by Unsigned Prefix DCJs. In the latter case, we fix a flaw present in the original paper. Since prefix reversals are particular cases of prefix DCJs, we extend this result to Sorting by Unsigned Prefix Reversals (note that Labarre and Cibulka [4] proved it for Sorting by Signed Prefix Reversals). These preliminary results allow us to answer an open question of Labarre [1], by proving that Sorting by Signed Prefix DCJs is in P, similarly as in the unrestricted case [3]. We also show that Sorting by Unsigned Prefix DCJs can be solved in polynomial time, and provide an additional result concerning the problem Sorting by Unsigned Prefix Reversals: although this problem has been shown to be NP-hard [5], we design a 1-absolute approximation algorithm (i.e. an algorithm providing solutions that are always at most 1 away from the optimum) to solve this problem for a specific class of permutations, which we call *pseudo-simple* permutations.

1.1. Permutations, genomes, and rearrangements

We begin with the simplest models for representing organisms.

Definition 1. A *(unsigned) permutation* of $[n] = \{1, 2, \ldots, n\}$ is a bijective function of $[n]$ onto itself. A *signed permutation* π of $\{\pm 1, \pm 2, \ldots, \pm n\}$ is a bijective function of $\{\pm 1, \pm 2, \ldots, \pm n\}$ onto itself that satisfies $\pi_{-i} = -\pi_i$ (+ signs are usually omitted). The *identity permutation* is the permutation $\iota = (1\ 2\ \cdots\ n)$.

As per standard practice (see e.g. Knuth [6] and Bóna [7]), we will mostly view permutations as sequences of elements rather than as functions; namely, we consider the sequence $(\pi_1\ \pi_2\ \cdots\ \pi_n)$ obtained by concatenating the elements of the permutation π ordered by positions. We use S_n and $S_n^{\pm}$ to denote the set of all unsigned and signed permutations, respectively. The following model is a straightforward generalisation of unsigned permutations.

Definition 2. A *genome* G is a set of vertex-disjoint paths and cycles over $\{0, 1, 2, \ldots, n+1\}$. It is *linear* if it consists of a single path with endpoints 0 and $n+1$. The *identity genome* is the path induced by the sequence $(0, 1, 2, \ldots, n+1)$.

Let us note that a genome may contain loops or parallel edges (see Fig. 1).

Definition 3. Let $e = \{u, v\}$ be an edge of a genome G. Then e is a *breakpoint* if $0 \notin e$, and either $|u - v| \neq 1$ or e has multiplicity two. Otherwise, e is an *adjacency*. The number of breakpoints of G is denoted by $b(G)$.

For instance, the genome with edge multiset $\{\{0,4\}, \{4,3\}, \underline{\{3,6\}}, \{1,2\}, \underline{\{2,1\}}, \underline{\{5,5\}}\}$, which is genome G_2 from Fig. 1, has three breakpoints (underlined). Note that permutations can be viewed as linear genomes using the following simple transformation: given a permutation π, extend it by adding two new elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$, and build the linear genome G_π with edge set $\{\{\pi_i, \pi_{i+1}\} \mid 0 \leq i \leq n\}$. This allows us to use the notion of breakpoints on permutations as well, with the understanding that they apply to the extended permutation, and therefore $b(\pi) = b(G_\pi)$.

Signed permutations generalise to signed genomes using the following notion.

Definition 4. The *unsigned translation* of π is the unsigned permutation π' obtained as follows: (a) transform each element π_i, $1 \leq i \leq n$, into the sequence $\pi'_{2i-1}\ \pi'_{2i}$, where $\pi'_{2i-1}\ \pi'_{2i} = (2\pi_i - 1)\ 2\pi_i$ if $\pi_i > 0$ (resp. $2|\pi_i|\ (2|\pi_i| - 1)$ if $\pi_i < 0$) ; (b) add two new elements $\pi'_0 = 0$ and $\pi'_{2n+1} = 2n + 1$.

For instance, the unsigned translation of the signed permutation $\pi = (-1\ 4\ -3\ -2\ 5)$ is (0 2 1 7 8 6 5 4 3 9 10 11).

Definition 5. A *signed genome* G is a perfect matching over the vertex set $\{0, 1, 2, \ldots, 2n + 1\}$. G is *linear* if there exists a signed permutation π such that $E(G) = \{\{\pi'_{2i}, \pi'_{2i+1}\} \mid 0 \leq i \leq n\}$, where π' is the unsigned translation of π. The *signed identity genome* is the perfect matching $\{\{2i, 2i + 1\} \mid 0 \leq i \leq n\}$.

An example of a signed genome for $n = 5$ is $G = \{\{0,4\}, \{1,6\}, \{3,7\}, \{10,8\}, \{9,5\}, \{2,11\}\}$. This genome is linear since there exists the signed permutation $\pi = (-2\ 4\ -5\ 3\ 1)$ whose unsigned translation $\pi' = (0\ 4\ 3\ 7\ 8\ 10\ 9\ 5\ 6\ 1\ 2\ 11)$ enables to give the set of edges of G. The signed identity genome for $n = 5$ is $\{\{0,1\}, \{2,3\}, \ldots, \{10,11\}\}$. It is sometimes convenient to view an unsigned translation σ as a signed genome, by using the elements of σ as vertex set and mapping each pair $(2\sigma_i, 2\sigma_i + 1)$ for $0 \leq i \leq n$ onto an edge $\{2\sigma_i, 2\sigma_i + 1\}$. We study transformations based on the following well-known operation on (signed) permutations, which will be seen to generalise in a natural way to (signed) genomes.

Definition 6. A *reversal* $\rho(i, j)$ with $1 \leq i < j \leq n$ is a permutation that reverses the order of elements between positions i and j:

$$\rho(i, j) = (1\ \cdots\ i-1\ j\ j-1\ \cdots\ i+1\ \ i\ j+1\ \cdots\ n).$$

A *signed reversal* $\overline{\rho}(i, j)$ with $1 \leq i \leq j \leq n$ is a signed permutation that reverses both the order and the signs of elements between positions i and j:

$$\overline{\rho}(i, j) = (1\ \cdots\ i-1\ \ -j\ \ -(j-1)\ \cdots\ -(i+1)\ \ -i\ j+1\ \cdots\ n).$$

If $i = 1$, then $\rho(i, j)$ (resp. $\overline{\rho}(i, j)$) is called a *prefix (signed) reversal*.

A reversal ρ applied to a permutation π transforms it into another permutation $\sigma = \pi\rho$. For instance, if $\pi = (-1\ 4\ \ -3\ \ -2\ 5)$, then the reversal $\rho(2,4)$ transforms π into $\pi\rho = (-1\ \ -2\ \ -3\ 4\ 5)$, while the *signed* reversal $\overline{\rho}(2,4)$ transforms π into $\pi\overline{\rho} = (-1\ 2\ 3\ \ -4\ 5)$. When the distinction matters, we mention whether objects or transformations are signed or unsigned; otherwise, we omit those qualifiers to lighten the presentation.

A reversal can be thought of as an operation that "cuts" (i.e., removes) two edges from a genome, then "joins" the severed endpoints (by adding two new edges) in such a way that the segment between the cuts is now reversed (see e.g. how G transforms into G_1 in Fig. 1). The following operation builds on that view to generalise reversals.

Definition 7. [3] Let $e = \{u, v\} \neq f = \{w, x\}$ be two edges of a genome G. The *double-cut-and-join (or DCJ for short)* δ applied to G transforms G into a genome G' by replacing edges e and f with either $\{\{u, w\}, \{v, x\}\}$ or $\{\{u, x\}, \{v, w\}\}$. Operation δ is a *prefix DCJ* if $0 \in e \cup f$.

Fig. 1 shows an example of a prefix DCJ applied to an unsigned genome, considering both options for replacing edges. DCJs apply to signed genomes as well: they may cut any pair of edges of the perfect matching, and recombine their endpoints in one of two ways.

1.2. Problems

We study several specialised versions of the following problem. A *configuration* is a (possibly signed and / or linear) genome, and the *identity configuration* is the identity genome with structural constraints consistent with the input configuration (see Definitions 1, 2 and 5).

SORTING BY Ω
Input: a configuration G, a number $K \in \mathbb{N}$, and a set Ω of allowed operations.
Question: is there a sequence of at most K operations from Ω that transforms G into the identity configuration?

Specific choices for Ω and the model chosen for G yield the following variants:

- SORTING BY UNSIGNED PREFIX DCJS, where G is a genome and Ω is the set of all prefix DCJs;
- SORTING BY SIGNED PREFIX DCJS, where G is a signed genome and Ω is the set of all prefix DCJs;
- SORTING BY UNSIGNED PREFIX REVERSALS, where G is a linear genome and Ω is the set of all prefix reversals;
- SORTING BY SIGNED PREFIX REVERSALS, where G is a signed linear genome and Ω is the set of all prefix signed reversals.

We refer to the smallest number of operations needed to transform G into the identity configuration as the Ω-*distance* of G. A specific distance is associated to each of the above problems; we use the following notation:

- $pdcj(G)$ for the prefix DCJ distance of an unsigned genome G, and $psdcj(G)$ for its signed version;
- $prd(G)$ for the prefix reversal distance of an unsigned genome G, and $psrd(G)$ for its signed version.

In order to avoid triviality, the input configuration for each problem must satisfy specific structural constraints so that it can be transformed into the identity configuration at all. Namely, two genomes can be transformed into one another by DCJs if and only if their labelled degree sequences coincide (see Bienstock and Günlük [8, Lemma 2.6]); and a genome can be sorted by (possibly prefix and / or signed) reversals if and only if it is linear (or equivalently, if it is a permutation). For Sorting by Unsigned Prefix DCJs, the input genome must contain a path from 0 to $n+1$; elements that do not belong to that path must belong to one or more cycles.

2. A generic lower bounding technique

We present in this section a lower bounding technique which applies to both the signed and the unsigned models, and on which we will build, in subsequent sections, to obtain exact or approximation algorithms.

2.1. Preliminaries

The effect of *algebraic transpositions*, or *exchanges*, on the classical cycles of a permutation, is a key ingredient to our lower bounding technique.

Definition 8. An *exchange* $\varepsilon(i,j)$ with $1 \leq i < j \leq n$ is a permutation that swaps elements in positions i and j:

$$\varepsilon(i,j) = \begin{pmatrix} 1 & \cdots & i-1 & \boxed{i} & i+1 & \cdots & j-1 & \boxed{j} & j+1 & \cdots & n \\ 1 & \cdots & i-1 & \boxed{j} & i+1 & \cdots & j-1 & \boxed{i} & j+1 & \cdots & n \end{pmatrix}.$$

If $i = 1$, then $\varepsilon(i,j)$ is called a *prefix exchange*.

For instance, the exchange $\varepsilon(3,6)$ applied to permutation $\pi = (3\ 1\ 4\ 2\ 6\ 5\ 7)$ transforms π into $\pi\varepsilon = (3\ 1\ 5\ 2\ 6\ 4\ 7)$. We let $\Gamma(\pi)$ denote the (directed) graph of a permutation π, with vertex set $[n]$ and which contains an arc (i,j) whenever $\pi_i = j$. The *length* of a cycle in the graph $\Gamma(\pi)$ is the number of edges it contains. A k-cycle is a cycle of length k: it is *trivial* if $k = 1$, and *nontrivial* otherwise. We let $c(\Gamma(\pi))$ (resp. $c_1(\Gamma(\pi))$) denote the number of cycles (resp. 1-cycles) in $\Gamma(\pi)$. Since exchanges involve elements belonging to one or two cycles of $\Gamma(\pi)$, we have the following well-known fact:

Fact 1. *For any unsigned permutation π and any exchange $\varepsilon(i,j)$, $c(\Gamma(\pi)) - c(\Gamma(\pi\varepsilon(i,j))) \in \{-1,0,1\}$.*

Let $ped(\pi)$ denote the prefix exchange distance of a permutation π. The following result allows the computation of that distance in linear time, and will be useful to our purposes.

Theorem 1. *[9] For any unsigned permutation π, we have*

$$ped(\pi) = n + c(\Gamma(\pi)) - 2c_1(\Gamma(\pi)) - \begin{cases} 0 & \text{if } \pi_1 = 1, \\ 2 & \text{otherwise.} \end{cases}$$

In the following two subsections, we obtain lower bounds on $pdcj(\cdot)$ and $spdcj(\cdot)$ by observing that the effect of (prefix) exchanges on the cycles of a permutation is the same as that of (prefix) DCJs on the cycles of a different graph structure based on permutations — namely, the *(unsigned) breakpoint graph*.

2.2. The signed case

We generalise the following structure, originally introduced by Bafna and Pevzner [10] for signed permutations, to signed genomes.

Definition 9. The *breakpoint graph* of a signed genome G, denoted by $BG(G)$, is the union of G, whose edges are referred to as *black edges*, and the signed identity genome, whose edges are referred to as *grey edges*. The breakpoint graph of a signed permutation is the breakpoint graph of its unsigned translation (viewed as a signed genome, as explained after Definition 5).

See Fig. 2 for an example. Note that, although vertices are merged by the union, edges with the same endpoints but with different colours are not. Breakpoint graphs are 2-regular and as such are the union of disjoint cycles whose edges alternate between both colours, thereby referred to as *alternating cycles*. The *length* of a cycle in a breakpoint graph is the number of black edges it contains. In what follows, we define a k-cycle as a cycle of length k: it is *trivial* if $k = 1$, and *nontrivial* otherwise. We let $c(BG(G))$ (resp. $c_1(BG(G))$) denote the number of cycles (resp. 1-cycles) in $BG(G)$.

A breakpoint graph represents both the current genome (as black edges) and the target genome (as grey edges). The breakpoint graph of the identity genome is the only breakpoint graph with $n+1$ cycles. Therefore, this representation allows to reduce the

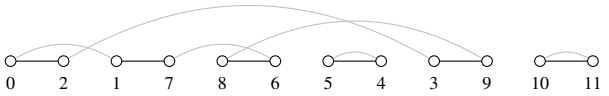

Fig. 2. The breakpoint graph of the unsigned translation obtained from $\pi = (-1\ 4\ -3\ -2\ 5)$.

SORTING BY Ω problem to increasing the number of cycles in a breakpoint graph up to $n+1$ by using only operations from the set Ω. As a consequence, by studying the effect of a single operation on the number of cycles, lower bounds can be obtained by showing that the number of cycles cannot increase by more than some increment at every step. The following result is an example of that strategy.

Theorem 2. *For any signed linear genome G, we have*

$$psdcj(G) \geq n + 1 + c(BG(G)) - 2c_1(BG(G)) - \begin{cases} 0 & \text{if } \{0,1\} \in G, \\ 2 & \text{otherwise.} \end{cases} \tag{1}$$

Proof. As observed by Yancopoulos et al. [3], a DCJ acts on at most two cycles of $BG(G)$ and can therefore change the number of cycles by at most one. Their effect on $BG(G)$ is therefore exactly the same as that of exchanges (see Fact 1), and this analogy is preserved under the prefix constraint. Since prefix DCJs cannot increase the number of cycles in $BG(G)$ faster than prefix exchanges in $\Gamma(\pi)$, the lower bound then follows from Theorem 1. Note that Equation (1) uses $n+1$ rather than n, since the number of elements is now the number of black edges; the condition that depends on $\{0,1\} \in G$ is the breakpoint graph analogue of $\pi_1 = 1$, since in both cases, the first component of the underlying graph structure is a cycle of length 1. □

Since (prefix) signed reversals are a subset of (prefix) signed DCJs, the result below by Labarre and Cibulka [4] is a simple corollary of Theorem 2.

Theorem 3. *[4] For any signed permutation π, we have*

$$psrd(\pi) \geq n + 1 + c(BG(\pi)) - 2c_1(BG(\pi)) - \begin{cases} 0 & \text{if } \pi_1 = 1, \\ 2 & \text{otherwise.} \end{cases} \tag{2}$$

2.3. The unsigned case

We now show that our lower bounds apply to the unsigned setting as well. The definition of the breakpoint graph in the unsigned case is identical to the signed case, except that the union acts on different structures; the definition of the length of a cycle remains unchanged.

Definition 10. The *unsigned breakpoint graph* of a genome G, denoted by $UBG(G)$, is the union of G, whose edges are referred to as *black edges*, and the identity genome, whose edges are referred to as *grey edges*. The breakpoint graph of a permutation $\pi = (\pi_1\ \pi_2\ \cdots\ \pi_n)$ is the breakpoint graph of the linear genome $(0, \pi_1, \pi_2, \ldots, \pi_n, n+1)$.

Fig. 3(*a*) shows an example of an unsigned breakpoint graph. Vertices 0 and $n+1$ in the unsigned breakpoint graph have degree 2, and all other vertices have degree 4. Just like its signed counterpart, the unsigned breakpoint graph also decomposes into alternating cycles, but this time the decomposition is no longer unique, and the choices made in the course of computing such a decomposition impact the value of the lower bound we provide next.

As explained in the paragraph before Theorem 2, a genome with a breakpoint graph whose cycle decomposition is "closest" to that of the identity will require fewer operations to sort. This motivates the need for decompositions that will optimise some parameter related to that decomposition, as formalised in the next definition.

Definition 11. For any genome G and an arbitrary decomposition $\mathscr{D}$ of $UBG(G)$ into alternating cycles, let $c^{\mathscr{D}}$ (resp. $c_1^{\mathscr{D}}$) denote the number of cycles (resp. trivial cycles) of $\mathscr{D}$. We call $\mathscr{D}$ *optimal* if it minimises $c^{\mathscr{D}} - 2c_1^{\mathscr{D}}$.

Fig. 3(*b*) shows an example of an optimal decomposition. The reason for this choice stems from the following lower bound, where $c^*(UBG(G))$ and $c_1^*(UBG(G))$ denote, respectively, the number of cycles and the number of 1-cycles in an optimal decomposition of $UBG(G)$: since we can only use prefix DCJs, each of which can increase the number of cycles in the breakpoint graph by at most 1, choosing the right decomposition with respect to the identity constitutes our only leeway. We discuss the problem of computing such a decomposition in Proposition 1.

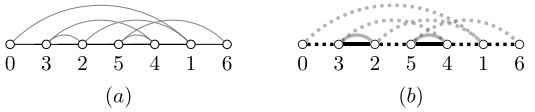

Fig. 3. (*a*) The unsigned breakpoint graph $UBG(\pi)$ of $\pi = (3\ 2\ 5\ 4\ 1)$; (*b*) an optimal decomposition of $UBG(\pi)$ into two trivial cycles (thick) and one 4-cycle (dotted).

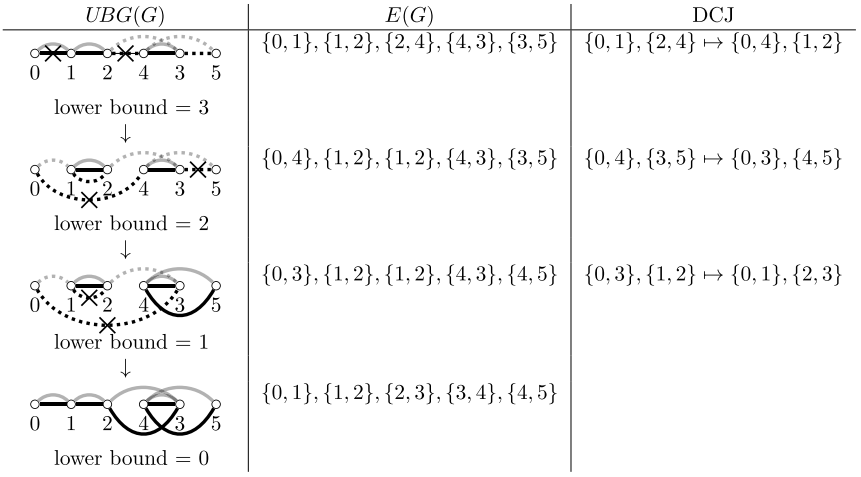

Fig. 4. An optimal sorting sequence of prefix DCJs for an instance with $n = 4$, and in which the presence of both $\{0,1\}$ and $\{1,2\}$ prevents the creation of a new 1-cycle. At each step, we show an optimal decomposition of $UBG(G)$: dotted edges belong to the nontrivial cycles of the decomposition. For instance, at the first step, we have $c^*(UBG(G)) = 4$ and $c_1^*(UBG(G)) = 3$. The value of the lower bound of Theorem 4 is also shown at each step.

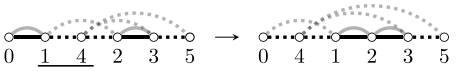

Fig. 5. An optimal prefix reversal with respect to Equation (4) for permutation (1 4 2 3): it moves 1 out of the way and preserves the number of trivial and nontrivial cycles (resp. 2 and 1). The decompositions of both unsigned breakpoint graphs are optimal, and dotted edges belong to the nontrivial cycles of the decomposition.

Theorem 4. *For any genome G, we have*

$$pdcj(G) \geq n + 1 + c^*(UBG(G)) - 2c_1^*(UBG(G)) - \begin{cases} 0 & \text{if } \{0,1\} \in G \text{ and } \{1,2\} \in G, \\ 1 & \text{if } \{0,1\} \in G \text{ and } \{1,2\} \notin G, \\ 2 & \text{otherwise.} \end{cases} \tag{3}$$

Proof. As in the proof of Theorem 2, prefix DCJs affect the cycles in a decomposition of $UBG(G)$ in the same way that prefix exchanges affect the cycles of a permutation. The particular case where $\{1,2\} \notin G$ allows us to make progress in the case where $\{0,1\} \in G$: since $\{1,2\} \notin G$, G contains at least one breakpoint involving 2, say $\{2,y\}$; applying the prefix DCJ that replaces $\{\{0,1\},\{2,y\}\}$ with $\{\{0,y\},\{1,2\}\}$ replaces the 1-cycle $(0,1)$ in $UBG(G)$ with a new 1-cycle $(1,2)$ in $UBG(G')$, and the number of nontrivial cycles is unaffected. Since prefix DCJs cannot increase the number of cycles in $BG(G)$ faster than prefix exchanges in $\Gamma(\pi)$, except in the case we just analysed, the lower bound then follows from Theorem 1 and the optimality of a decomposition. As in the proof of Theorem 2, Equation (3) uses $n+1$ rather than n, since the number of elements is now the number of black edges. □

Fig. 4 shows an optimal sorting sequence for an instance of length $n = 4$ that contains both $\{0,1\}$ and $\{1,2\}$. Since (prefix) reversals are (prefix) DCJs that preserve the path structure of a linear genome, the lower bound of Theorem 4 is also a lower bound on $prd(\pi)$.

Corollary 1. *For any permutation π on $n \geq 2$ elements, let $p_2 = \pi_{\pi_2^{-1}-1}$ (i.e., p_2 is the element that appears right before 2 in π); we have*

$$prd(\pi) \geq n + 1 + c^*(UBG(\pi)) - 2c_1^*(UBG(\pi)) - \begin{cases} 0 & \text{if } \pi_1 = 1 \text{ and } \{p_2,2\} \text{ is an adjacency,} \\ 1 & \text{if } \pi_1 = 1 \text{ and } \{p_2,2\} \text{ is a breakpoint,} \\ 2 & \text{otherwise.} \end{cases} \tag{4}$$

Fig. 5 shows an example for the case where $\pi_1 = 1$ and $\{p_2,2\}$ is a breakpoint.

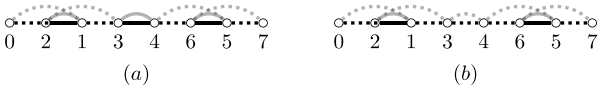

Fig. 6. Two optimal decompositions of $UBG(\pi)$ with $\pi = (2\,1\,3\,4\,6\,5)$. (a) A decomposition obtained using the algorithm described in the proof of Proposition 1, i.e., which first maximises the number of trivial cycles (namely, 3) and then minimises the number of nontrivial cycles (namely, 2). (b) Another optimal decomposition with only 2 trivial cycles and 1 nontrivial cycle.

We now show that an optimal decomposition of an unsigned breakpoint graph into alternating cycles can be found in polynomial time. This contrasts with the problem of finding an optimal decomposition in the case of sorting by unrestricted reversals, which was shown to be NP-complete [11] (note that in that context, an optimal decomposition *maximises* the total number of cycles). Recall that an *alternating Eulerian cycle* in a bicoloured graph G is a cycle that traverses every edge of G exactly once and such that the colours of every pair of consecutive edges are distinct.

Corollary 2. *[12,13] A bicoloured connected graph contains an alternating Eulerian cycle iff the number of incident edges of each colour is the same at every vertex; i.e., iff every vertex v incident with k_v edges of the first colour is incident with k_v edges of the second colour.*

Proposition 1. *There exists a polynomial-time algorithm for computing an optimal decomposition for $UBG(G)$.*

Proof. Let us consider the following algorithm: extract all trivial cycles from $UBG(G)$, then let each connected component in the resulting graph (ignoring isolated vertices) be a nontrivial cycle (Corollary 2 guarantees that the remaining components are alternating cycles). This yields a decomposition $\mathscr{D}$, which we now show is optimal. For this, note that by construction, no decomposition of $UBG(G)$ contains strictly more than $c_1^{\mathscr{D}}$ trivial cycles — in other words, $\mathscr{D}$ contains the highest possible number of trivial cycles. Moreover, also by construction, no decomposition containing $c_1^{\mathscr{D}}$ trivial cycles can have strictly less than $c^{\mathscr{D}} - c_1^{\mathscr{D}}$ nontrivial cycles.

Aiming for a contradiction, let us now assume that $\mathscr{D}$ is not optimal. Then there exists another decomposition $\mathscr{E}$ such that

$$c^{\mathscr{D}} - 2c_1^{\mathscr{D}} > c^{\mathscr{E}} - 2c_1^{\mathscr{E}}. \tag{5}$$

As previously argued, we have $c_1^{\mathscr{E}} \leq c_1^{\mathscr{D}}$. Now, if $c_1^{\mathscr{E}} = c_1^{\mathscr{D}}$, as discussed above, any decomposition, including $\mathscr{E}$, has at least $c^{\mathscr{D}} - c_1^{\mathscr{D}}$ nontrivial cycles. In other words, we have $c^{\mathscr{E}} - c_1^{\mathscr{E}} \geq c^{\mathscr{D}} - c_1^{\mathscr{D}}$, which we can rewrite as $c^{\mathscr{E}} - 2c_1^{\mathscr{E}} \geq c^{\mathscr{D}} - 2c_1^{\mathscr{D}}$ (since we suppose $c_1^{\mathscr{E}} = c_1^{\mathscr{D}}$). However, this contradicts Equation (5). Therefore, the only remaining case is the following: $c_1^{\mathscr{E}} < c_1^{\mathscr{D}}$.

Assume that $c_1^{\mathscr{D}} = c_1^{\mathscr{E}} + x$, for some integer $x > 0$, and, w.l.o.g., let us consider $\mathscr{E}$ to be a decomposition for which x is minimised. In other words, x of the $c_1^{\mathscr{D}}$ trivial cycles in $\mathscr{D}$ are contained in the nontrivial cycles of $\mathscr{E}$ (see e.g. Fig. 6(b), in which the trivial cycle $(3,4)$ of the decomposition in Fig. 6(a) is now part of a nontrivial cycle).

A merge involving a trivial cycle c occurs with either one or two cycles, that may be trivial or nontrivial, which leads to five possible cases. Let first $c_{nt}^{\mathscr{D}} = c^{\mathscr{D}} - c_1^{\mathscr{D}}$ (resp. $c_{nt}^{\mathscr{E}} = c^{\mathscr{E}} - c_1^{\mathscr{E}}$) denote the number of nontrivial cycles in $\mathscr{D}$ (resp. in $\mathscr{E}$). The five cases to consider are the following:

1. c is merged with one nontrivial cycle. In that case, $x = 1$ and the number of nontrivial cycles remains unchanged, i.e. $c_{nt}^{\mathscr{E}} - c_{nt}^{\mathscr{D}} = 0$;
2. c is merged with one trivial cycle. In that case, $x = 2$ and the number of nontrivial cycles is increased by one, i.e. $c_{nt}^{\mathscr{E}} - c_{nt}^{\mathscr{D}} = 1$;
3. c is merged with two nontrivial cycles. In that case, $x = 1$ and the number of nontrivial cycles is decreased by one, i.e. $c_{nt}^{\mathscr{E}} - c_{nt}^{\mathscr{D}} = -1$;
4. c is merged with one nontrivial and one trivial cycle. In that case, $x = 2$ and the number of nontrivial cycles remains unchanged, i.e. $c_{nt}^{\mathscr{E}} - c_{nt}^{\mathscr{D}} = 0$;
5. finally, c is merged with two trivial cycles. In that case, $x = 3$ and the number of nontrivial cycles increases by 1, i.e. $c_{nt}^{\mathscr{E}} - c_{nt}^{\mathscr{D}} = 1$.

In each of the above five cases, we can see that $c_{nt}^{\mathscr{E}} - c_{nt}^{\mathscr{D}} \geq -x$. However, the above inequality contradicts Equation (5). Indeed, we can rewrite it $c_{nt}^{\mathscr{D}} - c_1^{\mathscr{D}} > c_{nt}^{\mathscr{E}} - c_1^{\mathscr{E}}$, which yields $c_{nt}^{\mathscr{E}} - c_{nt}^{\mathscr{D}} < -x$, since we assumed $c_1^{\mathscr{D}} = c_1^{\mathscr{E}} + x$. This shows that $\mathscr{D}$ is optimal. □

Note that maximising the number of trivial cycles is only one way of reaching optimality. For instance, Fig. 6(b) shows a decomposition that does not maximise the number of trivial cycles, yet is optimal.

3. Algorithms for sorting by prefix DCJs

3.1. Signed prefix DCJs

We give a polynomial-time algorithm for Sorting by Signed Prefix DCJs. It is based on the notion of breakpoint, which we defined in the unsigned setting (Definition 3), and which generalises to the signed setting in a straightforward way: a *breakpoint* in a signed genome is an edge that does not contain 0 and that does not appear in the identity genome. Our result relies on the following simple observation.

Observation 1. *Let G be a signed linear genome, and let $g(G)$ denote the right-hand side of Equation (1). Then $g(G) = 0$ iff G is the identity genome.*

Proof. Clearly, if G is the identity genome, then $\{0, 1\} \in G$ and $BG(G)$ contains $n + 1$ cycles, which all are trivial. Thus $c(BG(G)) = c_1(BG(G)) = n + 1$ and we obtain $g(G) = 0$.

If $g(G) = 0$, let us denote $c^+(BG(G)) = c(BG(G)) - c_1(BG(G))$ the number of nontrivial cycles in $BG(G)$. Then:

1. either $\{0, 1\} \in G$, in which case $g(G) = 0$ can be rewritten as $c_1(BG(G)) - c^+(BG(G)) = n + 1$. However, since $c_1(BG(G))$ can never exceed $n + 1$, the only way to satisfy the above equality is when $c_1(BG(G)) = n + 1$ and $c^+(BG(G)) = 0$, that is when G is the identity genome.
2. or $\{0, 1\} \notin G$, in which case $g(G) = 0$ can be rewritten as $c_1(BG(G)) - c^+(BG(G)) = n - 1$. However, since $\{0, 1\} \notin G$, 0 and 1 belong to a nontrivial cycle C; therefore, $c_1(BG(G)) \leq n - 1$ because C contains at least 2 edges, and thus prevents the existence of at least 2 trivial cycles (among the $n + 1$ potential ones). Consequently, $c_1(BG(G)) - c^+(BG(G)) = n - 1$ is only possible in this case when $c_1(BG(G)) = n - 1$ and $c^+(BG(G)) = 0$, a contradiction to the existence of the nontrivial cycle C. □

Theorem 5. *The* Sorting by Signed Prefix DCJs *problem is in* P.

Proof. Let G be any signed genome (other than the identity, in order to avoid triviality). For convenience, as in the proof of Observation 1, let $g(G)$ denote the right-hand side of Equation (1). We will show that the lower bound of Theorem 2 for Sorting by Signed Prefix DCJs is also an upper bound, by proving that every unsorted genome G admits a prefix DCJ that transforms G into another genome G' in such a way that $g(G') - g(G) = -1$. Doing this, we are able, in $g(G)$ steps, to obtain a genome G^* such that $g(G^*) = 0$; by Observation 1, we conclude that G^* is the identity genome. Thus, inductively, starting from G^*, this shows that every intermediate genome G^α we obtained going from G to G^* satisfies $psdcj(G^\alpha) \leq g(G^\alpha)$. In particular, $psdcj(G) \leq g(G)$, which, combined with Theorem 2, shows $psdcj(G) = g(G)$.

Now it remains to show that every unsorted genome G admits a prefix DCJ that transforms G into another genome G', in such a way that $g(G') - g(G) = -1$. In what follows, we refer to the only neighbour of a vertex in G (resp. in the identity genome) as its *black* (resp. *grey*) neighbour, which is guaranteed to be unique since signed genomes are perfect matchings. For convenience, let $g(G)$ denote the right-hand side of Equation (1). Let v be the black neighbour of 0; we have two cases to consider:

1. if $v = 1$, apply a prefix DCJ δ acting on $\{0, v\}$ and any breakpoint $\{x, y\}$ and which reconnects the endpoints arbitrarily. Since δ merges the trivial cycle $(0, 1)$ with a nontrivial cycle in $BG(G)$, the number of trivial cycles decreases by 1, as does the total number of cycles. The resulting genome G' satisfies

$$\begin{aligned} g(G') - g(G) &= n + 1 + c(BG(G)) - 1 - 2(c_1(BG(G)) - 1) - 2 \\ &\quad - (n + 1 + c(BG(G)) + 2(c_1(BG(G))) \\ &= -1. \end{aligned}$$

2. otherwise, $v \neq 1$: let x be the grey neighbour of v and y be the black neighbour of x, and apply the prefix DCJ δ' that replaces $\{0, v\}$ and $\{x, y\}$ with $\{0, y\}$ and $\{v, x\}$. Since δ' connects v to its grey neighbour, the number of trivial cycles increases by 1 (if $y \neq 1$) or 2 (if $y = 1$). Moreover, before applying δ', there is an alternating path going through elements 0, v, x and y in $BG(G)$, which means that those elements belong to the same cycle in $BG(G)$. Applying δ' splits that cycle into two distinct cycles, and therefore the total number of cycles increases by 1 in both cases. We distinguish between two subcases based on y's value:
   (a) if $y \neq 1$, then

$$\begin{aligned} g(G') - g(G) &= n + 1 + c(BG(G)) + 1 - 2(c_1(BG(G)) + 1) - 2 \\ &\quad - (n + 1 + c(BG(G)) + 2(c_1(BG(G))) + 2 \\ &= -1. \end{aligned}$$

   (b) if $y = 1$, then

$$\begin{aligned} g(G') - g(G) &= n + 1 + c(BG(G)) + 1 - 2(c_1(BG(G)) + 2) \\ &\quad - (n + 1 + c(BG(G)) + 2(c_1(BG(G))) + 2 \\ &= -1. \end{aligned}$$

As a consequence, the value of our lower bound always decreases by 1 in all cases, which completes the proof. □

Algorithm 1 implements the approach outlined in the proof of Theorem 5, without the need to build the breakpoint graph. Indeed, the other endpoint x of the grey edge incident to 0's neighbour v can be deduced from v's parity: since the identity's edge set is $\{\{2i, 2i + 1\} \mid 0 \leq i \leq n\}$ (see Definition 5), if v is odd, then its neighbour in the identity is $v - 1$; likewise, if v is even, then

Algorithm 1: SORTINGBYPSDCJ(G).

Input: A signed genome G.
Output: An optimal sorting sequence of prefix DCJs for G.

```
S ← empty sequence;
B ← G's breakpoints;
while B ≠ ∅ do
    v ← 0's neighbour in G;
    if v = 1 then
        extract any breakpoint {x, y} from B;
        G ← G \ {{0, v}, {x, y}} ∪ {{0, x}, {v, y}};
        S.append(({{0, v}, {x, y}}, {{0, x}, {v, y}}));
        B ← B ∪ {{v, y}};
    else
        x ← v + 1 − 2(v mod 2);                          // v's neighbour in the identity
        y ← x's neighbour in G;
        G ← G \ {{0, v}, {x, y}} ∪ {{0, y}, {v, x}};
        S.append(({{0, v}, {x, y}}, {{0, y}, {v, x}}));
        B ← B \ {{x, y}};
return S;
```

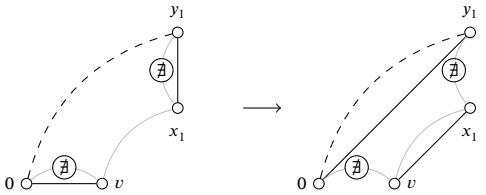

Fig. 7. Proof of Theorem 6, in the case where $\{0, v\} \in G$ with $v \neq 1$. Dashed edges correspond to alternating paths. Grey edges that are known *not* to exist in UBG are struck out. In order to be as general as possible, G is intentionally not drawn linearly.

its neighbour in the identity is $v + 1$. Encoding G's edges using an array storing each vertex's only neighbour allows us to query neighbours and perform DCJs in $O(1)$ time, and a simple set data structure for breakpoints allows us to extract or add them in $O(1)$ time as well. Finally, the distance's value is bounded by $O(n)$, and checking whether G is sorted reduces to checking whether the set of breakpoints is empty, which yields an $O(n)$ running time for Algorithm 1.

3.2. Unsigned prefix DCJs

We now turn to the problem of SORTING BY UNSIGNED PREFIX DCJS, for which we give a polynomial-time algorithm.

Theorem 6. *For any genome G, we have*

$$pdcj(G) = n + 1 + c^*(UBG(G)) - 2c_1^*(UBG(G)) - \begin{cases} 0 & \text{if } \{0,1\} \in G \text{ and } \{1,2\} \in G, \\ 1 & \text{if } \{0,1\} \in G \text{ and } \{1,2\} \notin G, \\ 2 & \text{otherwise.} \end{cases} \tag{6}$$

Proof. Theorem 4 shows that the right-hand side of Equation (6), that we will denote $g(G)$ for convenience, is a lower bound for SORTING BY UNSIGNED PREFIX DCJS. Note that, using arguments similar to those discussed in Observation 1, $g(G)$ is equal to 0 iff G is the identity genome. Then, assume G is not the identity genome, in which case the claim trivially holds. To show that $g(G)$ is also an upper bound, we use arguments similar to those provided in the proof of Theorem 5. More precisely, we first prove that it is always possible to find a prefix DCJ that transforms G into another genome G' in such a way that $g(G)$ decreases by 1. Combining the above property with the fact that $g(G) = 0$ iff G is the identity genome allows us to conclude that $pdcj(G) = g(G)$. Let $c = c^*(UBG(G))$ and $c_1 = c_1^*(UBG(G))$, as well as $c' = c^*(UBG(G'))$ and $c_1' = c_1^*(UBG(G'))$. We have two main cases to consider:

1. If $\{0, v\} \in G$ with $v \neq 1$ (see Fig. 7), then G contains an element $x_1 \in \{v - 1, v + 1\}$ that is not adjacent to v. In turn, since $\{v, x_1\} \notin G$, x_1 must be adjacent to an element y_1 such that $\{x_1, y_1\}$ is a breakpoint. We distinguish between the following subcases depending on the value of y_1.
   (a) If $y_1 \neq 1$: we apply the prefix DCJ δ_1 that replaces $\{\{0, v\}, \{x_1, y_1\}\}$ with $\{\{0, y_1\}, \{v, x_1\}\}$. Since $UBG(G)$ contains the alternating path $(0, v, x_1, y_1)$, those four elements belong to the same cycle in $UBG(G)$; δ_1 replaces that path with a shortcut of length 1 (namely, $\{0, y_1\}$), and the cycle remains nontrivial since $y_1 \neq 1$. Therefore, we have $c' = c + 1$ and $c_1' = c_1 + 1$, and:

$$g(G') - g(G) = n + 1 + c + 1 - 2c_1 - 2 - 2 - (n + 1 + c - 2c_1 - 2) = -1.$$

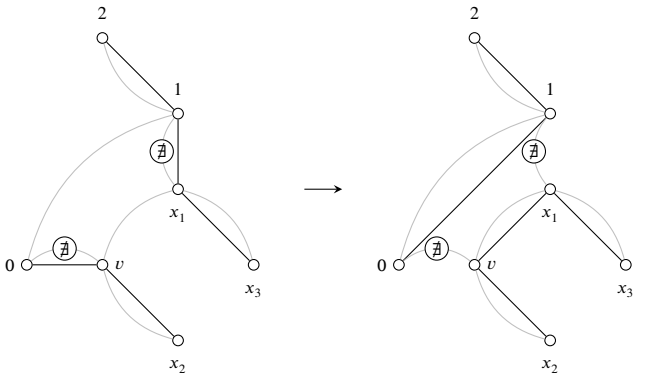

Fig. 8. Proof of Theorem 6, case 1(c)ii. In order to be as general as possible, G is intentionally not drawn linearly.

(b) If $y_1 = 1$ and $\{1,2\} \notin G$: we apply the prefix DCJ δ_1 from the previous case. Both c_1 and c increase by 2 since δ_1 also yields the trivial cycle $(0,1)$ in G'. Since $\{0,1\} \in G'$ and $\{1,2\} \notin G'$, we obtain:

$$g(G') - g(G) = n + 1 + c + 2 - 2c_1 - 4 - 1 - (n + 1 + c - 2c_1 - 2) = -1.$$

(c) If $y_1 = 1$ and $\{1,2\} \in G$: consider x_1's other black neighbour $z_1 \neq y_1$: if $\{x_1, z_1\}$ is a breakpoint, then we apply the analysis used in case 1(a), replacing y_1 with z_1 (so we apply a different prefix DCJ δ_2, which replaces $\{\{0,v\},\{x_1,z_1\}\}$ with $\{\{0,z_1\},\{v,x_1\}\}$). Otherwise, consider v's other grey neighbour $x_2 \in \{v-1, v+1\}$. x_2 cannot be adjacent to 1 since 1's neighbours are x_1 and 2.

i. If x_2 is not adjacent to v, then it forms a breakpoint with some element w. In that case, we can again apply the analysis used in case 1(a), replacing x_1 with x_2 and y_1 with w (so we apply a prefix DCJ δ_3, which replaces $\{\{0,v\},\{x_2,w\}\}$ with $\{\{0,w\},\{v,x_2\}\}$).

ii. Otherwise, v has neighbours 0 and x_2. Moreover, element $x_1 \neq x_2$ with $x_1 \in \{v-1, v+1\}$ is adjacent to 1 and has a neighbour x_3 such that $\{x_1, x_3\}$ is an adjacency (since otherwise we would apply the above analysis involving z_1). Then there exists an optimal decomposition of $UBG(G)$ with a 2-cycle induced by vertices $(0, v, x_1, 1)$ that is itself surrounded by the 1-cycles (v, x_2), (x_3, x_1) and $(1,2)$ (see Fig. 8). Note that even if $x_2 = x_3$, the 2-cycle of interest in an optimal decomposition remains the same since the 1-cycles (v, x_2), (x_3, x_1) and $(1,2)$ still surround the 2-cycle $(0, v, x_1, 1)$. Therefore, we can apply the prefix DCJ δ_1 from cases 1(a) and 1(b), which replaces $\{\{0,v\},\{x_1,1\}\}$ with $\{\{0,1\},\{v,x_1\}\}$. δ_1 replaces a nontrivial cycle with two new trivial cycles, and we have:

$$g(G') - g(G) = n + 1 + c + 1 - 2c_1 - 4 - 0 - (n + 1 + c - 2c_1 - 2) = -1.$$

2. Otherwise, $\{0,1\} \in G$.

(a) If $\{1,2\} \notin G$ (see Fig. 9(*a*)), then G contains the breakpoint $\{2, y\}$, in which case a prefix DCJ δ_4 which replaces $\{\{0,1\},\{2,y\}\}$ with $\{\{0,y\},\{1,2\}\}$ yields a genome G' with $c'_1 = c_1$, since cycle $(0,1)$ was replaced with cycle $(1,2)$. Note that 1 and y belong to the same nontrivial cycle in an optimal decomposition of $UBG(G)$, otherwise those separate nontrivial cycles could have been merged into a single one using the alternating path $(1, 2, y)$. They also belong to the same cycle in $UBG(G')$, so $c' = c$ and we obtain:

$$g(G') - g(G) = n + 1 + c - 2c_1 - 2 - (n + 1 + c - 2c_1 - 1) = -1.$$

(b) If $\{1,2\} \in G$ (see Fig. 9(*b*)), then let k be the closest element to 0 in the only path of G such that the next vertex ℓ forms a breakpoint with k. Then a prefix DCJ δ_5 which replaces $\{\{0,1\},\{k,\ell\}\}$ with $\{\{0,\ell\},\{1,k\}\}$ yields genome G'. The removal of black edge $\{0,1\}$ leads to a decrease of 1 in the number of trivial cycles in $UBG(G)$. Moreover, k and ℓ belong to the same nontrivial cycle in $UBG(G)$, and remain in that cycle in $UBG(G')$ due to the replacement of black edge $\{k,\ell\}$ with the alternating path $(k, 1, 0, \ell)$. Therefore $c' = c - 1$, and we obtain:

$$g(G') - g(G) = n + 1 + c - 1 - 2(c_1 - 1) - 2 - (n + 1 + c - 2c_1) = -1. \quad \square$$

Algorithm 2 implements the approach outlined in the proof of Theorem 6, without the need to build the breakpoint graph. Encoding G's edges using an array storing each vertex's neighbours (at most 2) allows us to query neighbours and perform DCJs in $O(1)$ time, and a simple set data structure for breakpoints allows us to extract or add them in $O(1)$ time as well. The first breakpoint in the only path of G (line 30) corresponds to the breakpoint with the smallest element of G. We can easily record this element the first time we extract breakpoints and the update can be done in $O(1)$ time. Finally, the distance's value in the unsigned case is bounded by $O(n)$, and checking whether G is sorted reduces to checking whether the set of breakpoints is empty, which yields a $O(n)$ running time for Algorithm 2.

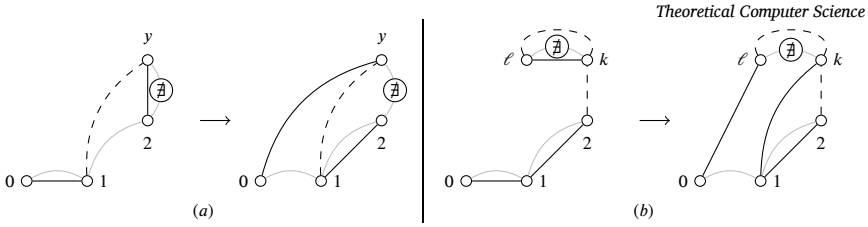

Fig. 9. Proof of Theorem 6: the cases where $\{0,1\} \in G$. Dashed edges correspond to alternating paths. Grey edges that are known *not* to exist in UBG are struck out.

Algorithm 2: SORTINGBYPDCJ(G).

Input: A genome G.
Output: An optimal sorting sequence S of prefix DCJs for G.

$S \leftarrow$ empty sequence;
while G *is not sorted* **do**
  $v \leftarrow 0$'s neighbour in G;
  if $v = 1$ **then**
    $x_1 \leftarrow$ an element in $\{v-1, v+1\}$ such that $\{v, x_1\} \notin G$;
    $y_1 \leftarrow$ a neighbour of x_1 such that $\{x_1, y_1\}$ is a breakpoint;
    if $y_1 \neq 1$ ***or*** $\{1,2\} \notin G$ **then** // cases 1(a) and 1(b)
      $G \leftarrow G \setminus \{\{0,v\},\{x_1,y_1\}\} \cup \{\{0,y_1\},\{v,x_1\}\}$;
      S.append($\{\{0,v\},\{x_1,y_1\}\},\{\{0,y_1\},\{v,x_1\}\}$);
    else // case 1(c)
      $z_1 \leftarrow$ the other neighbour of x_1;
      if $\{x_1, z_1\}$ *is a breakpoint* **then**
        $G \leftarrow G \setminus \{\{0,v\},\{x_1,z_1\}\} \cup \{\{0,z_1\},\{v,x_1\}\}$;
        S.append($\{\{0,v\},\{x_1,z_1\}\},\{\{0,z_1\},\{v,x_1\}\}$);
      else
        $x_2 \leftarrow$ the other grey neighbour of v;
        if $\{v, x_2\} \notin G$ **then** // case 1(c)i
          $w \leftarrow$ a neighbour of x_2 such that $\{w, x_2\}$ is a breakpoint;
          $G \leftarrow G \setminus \{\{0,v\},\{x_2,w\}\} \cup \{\{0,w\},\{v,x_2\}\}$;
          S.append($\{\{0,v\},\{x_2,w\}\},\{\{0,w\},\{v,x_2\}\}$);
        else // case 1(c)ii
          $G \leftarrow G \setminus \{\{0,v\},\{x_1,y_1\}\} \cup \{\{0,y_1\},\{v,x_1\}\}$;
          S.append($\{\{0,v\},\{x_1,y_1\}\},\{\{0,y_1\},\{v,x_1\}\}$);
  else
    if $\{1,2\} \notin G$ **then** // case 2(a)
      $y \leftarrow$ a neighbour of 2 such that $\{2, y\}$ is a breakpoint;
      $G \leftarrow G \setminus \{\{0,1\},\{2,y\}\} \cup \{\{0,y\},\{1,2\}\}$;
      S.append($(\{\{0,1\},\{2,y\}\},\{\{0,y\},\{1,2\}\})$);
    else // case 2(b)
      $\{k, \ell\} \leftarrow$ the first breakpoint in the only path of G;
      $G \leftarrow G \setminus \{\{0,1\},\{k,\ell\}\} \cup \{\{0,\ell\},\{1,k\}\}$;
      S.append($(\{\{0,1\},\{k,\ell\}\},\{\{0,\ell\},\{1,k\}\})$);
return S;

4. Prefix reversals

We start by showing that the lower bound of Corollary 1 (more precisely, Equation (4)) is always at least as large as the number of breakpoints.

Proposition 2. *For any unsigned permutation π, the lower bound of Equation (4) is greater than or equal to $b(\pi)$. Moreover, the gap that separates both bounds can be arbitrarily large.*

Proof. Consider any permutation π of length n, and let us optimally decompose $UBG(\pi)$ as described in the proof of Proposition 1. In order to prove the first part of the statement, we distinguish between three cases, which are the ones that appear in Equation (4). For this, recall that $p_2 = \pi_{\pi_2^{-1}-1}$ is the element that appears right before 2 in π.

1. If $\pi_1 = 1$ and $\{p_2, 2\}$ is an adjacency, then the lower bound of Equation (4) has value $n + 1 + c^*(UBG(\pi)) - 2c_1^*(UBG(\pi))$. Note that, by definition, each of the $c_1^*(UBG(\pi))$ trivial cycles in our decomposition of $UBG(\pi)$ corresponds to an edge $\{i, i+1\}$, which is therefore an adjacency. Moreover, since $\pi_1 = 1$, then 0 and 1 form a trivial cycle, and consequently

$$n + 1 = b(\pi) + c_1^*(UBG(\pi)). \tag{7}$$

We can rewrite Equation (4) as follows:

$$prd(\pi) \geq \underbrace{(n+1-c_1^*(UBG(\pi))}_{A} + \underbrace{(c^*(UBG(\pi)) - c_1^*(UBG(\pi)))}_{B}.$$

By Equation (7), we know that $A = b(\pi)$, and since B is the number of nontrivial cycles in our decomposition, we have $B \geq 0$.

2. If $\pi_1 = 1$ and $\{p_2, 2\}$ is a breakpoint, then the lower bound of Equation (4) has value $n+1+c^*(UBG(\pi)) - 2c_1^*(UBG(\pi)) - 1$. As in the previous case, 0 and 1 form a trivial cycle, so Equation (7) also holds. We can therefore rewrite Equation (4) as follows:

$$prd(\pi) \geq \underbrace{(n+1-c_1^*(UBG(\pi)))}_{A} + \underbrace{(c^*(UBG(\pi)) - c_1^*(UBG(\pi)) - 1)}_{C}.$$

As discussed in the previous case, Equation (7) implies that $A = b(\pi)$. Moreover, we have $C \geq 0$: indeed, since π contains the breakpoint $\{p_2, 2\}$, it cannot be the identity permutation, and therefore any decomposition of $UBG(\pi)$ contains at least one nontrivial cycle. In other words, $c^*(UBG(\pi)) \geq c_1^*(UBG(\pi)) + 1$, and we are done.

3. Finally, if $\pi_1 \neq 1$, then the lower bound of Equation (4) has value $n+1+c^*(UBG(\pi)) - 2c_1^*(UBG(\pi)) - 2$. In that case, since 0 and 1 do not form a trivial cycle but is (by definition) not counted as a breakpoint, we have

$$n = b(\pi) + c_1^*(UBG(\pi)). \tag{8}$$

We can rewrite Equation (4) as follows:

$$prd(\pi) \geq \underbrace{(n - c_1^*(UBG(\pi))}_{D} + \underbrace{(c^*(UBG(\pi)) - c_1^*(UBG(\pi)) - 1)}_{C}.$$

As discussed in the previous case, since π is not the identity genome, we have $C \geq 0$. The fact that $D = b(\pi)$ follows from Equation (8).

In order to show that the gap between the lower bound of Equation (4) and $b(\pi)$ can be arbitrarily large, consider the following permutation built on $n = 6p$ elements, for any arbitrary integer $p \geq 2$: π is the concatenation of subpermutations $\sigma_1, \sigma_2, \ldots, \sigma_p$, where for every $1 \leq i \leq p$, we have

$$\sigma_i = (6i-5)\,(6i-3)\,(6i-1)\,(6i-4)\,(6i-2)\,6i.$$

For instance, when $p = 3$, we have

$$\pi = (\underbrace{1\,3\,5\,2\,4\,6}_{\sigma_1}\ \underbrace{7\,9\,11\,8\,10\,12}_{\sigma_2}\ \underbrace{13\,15\,17\,14\,16\,18}_{\sigma_3}).$$

Now consider the optimal decomposition of $UBG(\pi)$ as described in the proof of Proposition 1. It can be seen that $c_1^*(UBG(\pi)) = p+1$, which corresponds to edges $\{6i, 6i+1\}$, $0 \leq i \leq p$. Thus, since $\pi_1 = 1$, Equation (7) holds and we have $b(\pi) = n+1-c_1^*(UBG(\pi)) = 5p$. Finally, π has been built in such a way that, for every $1 \leq i \leq p$, the elements of the interval $[6i-5; 6i]$ form a cycle in $UBG(\pi)$. As a consequence, when trivial cycles are removed from $UBG(\pi)$, p connected components remain, each induced by the elements of $[6i-5; 6i]$, $1 \leq i \leq p$. Hence, the optimal decomposition described in the proof of Proposition 1 yields p nontrivial cycles. This allows us to conclude that $c^*(UBG(\pi)) = 2p+1$. Altogether, since $\pi_1 = 1$ and $\{p_2, 2\}$ is a breakpoint, we are in case 2 of Equation (4), i.e. $prd(\pi) \geq n+1+c^*(UBG(\pi)) - 2c_1^*(UBG(\pi)) - 1$. The above discussion allows us to conclude that $prd(\pi) \geq 6p-1$, while as mentioned above, $b(G) = 5p$, which is the sought result. □

We now show that the longstanding ratio of 2 for Sorting by Prefix Reversals can be improved for permutations that are *pseudo-simple* (see Definition 12 below). The performance of our algorithm is measured additively: more precisely, for any pseudo-simple permutation π, it always returns a solution of size at most $prd(\pi) + 1$.

Definition 12. A permutation π is *pseudo-simple* if $UBG(\pi)$ admits an optimal decomposition with no cycle of length longer than 2.

For instance, permutation (2 1) is pseudo-simple, and Fig. 10(*a*) shows an optimal decomposition of its unsigned breakpoint graph into cycles of length ≤ 2. By contrast, the unsigned breakpoint graph of permutation (2 3 1) does admit a decomposition into cycles of length ≤ 2 (see Fig. 10), but that decomposition is not optimal. We name these permutations by analogy with *simple* signed permutations, which are signed permutations whose breakpoint graph has no cycle of length longer than 2. The following structural property of pseudo-simple permutations will prove useful.

Observation 2. *For any pseudo-simple permutation π, let $\mathscr{D}^*$ be an optimal decomposition of $UBG(\pi)$. Then for every pair of black edges that share a vertex, at least one of these edges belongs to a trivial cycle.*

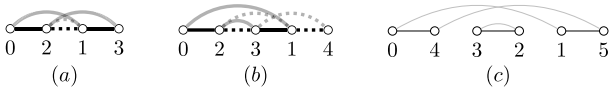

Fig. 10. (*a*) A decomposition of the unsigned breakpoint graph of pseudo-simple permutation $\pi = (2\ 1)$ into two cycles of length ≤ 2. (*b*) A non-optimal decomposition of $UBG((2\ 3\ 1))$ into two 2-cycles. (*c*) The breakpoint graph of $(-2\ -1)$, a spin of pseudo-simple permutation $(2\ 1)$ which satisfies Observation 3.

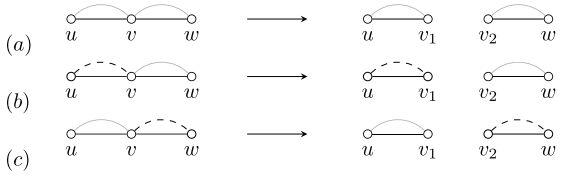

Fig. 11. The transformation outlined in Observation 3 and its impacts: (a) $\{u, v\}$ and $\{v, w\}$ both belong to trivial cycles ; (b) $\{v, w\}$ belongs to a trivial cycle while $\{u, v\}$ belongs to a 2-cycle ; $\{u, v\}$ belongs to a trivial cycle while $\{v, w\}$ belongs to a 2-cycle. Dashed edges correspond to alternating paths.

Proof. For any permutation π, let $\mathscr{D}^*$ be any optimal decomposition of $UBG(\pi)$, and let $\{u, v\}$ and $\{v, w\}$ be two black edges that share the vertex v. If one of these edges belongs to a trivial cycle, we are done. Otherwise, $\{u, v\}$ and $\{v, w\}$ must belong to the same 2-cycle (if they belonged to separate 2-cycles, then they could be merged into a 4-cycle, which would contradict the optimality of $\mathscr{D}^*$), and therefore $\{u, v\}$ and $\{v, w\}$ also appear as grey edges. Now consider the decomposition $\mathscr{D}'$ obtained from $\mathscr{D}^*$ by splitting the 2-cycle involving $\{u, v\}$ and $\{v, w\}$ into two trivial cycles. If c (resp. c_1) denotes the number of cycles (resp. trivial cycles) in $\mathscr{D}^*$, then in $\mathscr{D}'$ we have $c'_1 = c_1 + 2$ trivial cycles and $c' = c + 1$ cycles. But in that case, we have $c' - 2c'_1 = (c + 1) - 2(c_1 + 2)$, i.e., $c' - 2c'_1 = (c - 2c_1) - 3$, which contradicts the minimality of $c - 2c_1$ and thereby the optimality of $\mathscr{D}^*$ as well. □

The following notion provides a useful connection between both classes.

Definition 13. [14] A signed permutation $\vec{\pi}$ in $S_n^{\pm}$ is a *spin* of an unsigned permutation π in S_n if $\pi_i = |\vec{\pi}_i|$ for $1 \leq i \leq n$.

We simplify Hannenhalli and Pevzner's definition [14] of "(un)oriented" edges as follows.

Definition 14. For any permutation π, let C be a 2-cycle in $UBG(\pi)$. C is *oriented* if the positions of the endpoints of its grey edges have the same parity, and *unoriented* otherwise.

For instance, the dotted 2-cycle in Fig. 10(*b*) is oriented (the endpoints of $\{2, 1\}$ are $\{1, 3\}$ and those of $\{3, 4\}$ are $\{2, 4\}$), while the other 2-cycle is unoriented (the endpoints of $\{0, 1\}$ are $\{0, 3\}$ and those of $\{2, 3\}$ are $\{1, 2\}$).

Observation 3. *Every pseudo-simple permutation π has a simple spin σ such that every 2-cycle in $UBG(\pi)$ is mapped onto a 2-cycle with the same orientation in $BG(\sigma)$.*

See Fig. 10(*c*) for an example.

Proof. Let π be a pseudo-simple permutation, and $\mathscr{D}^*$ an optimal decomposition of $UBG(\pi)$. We transform $UBG(\pi)$ into a new graph B by splitting each vertex v in $UBG(\pi)$, except 0 and $n + 1$, into an ordered pair (v_1, v_2) of vertices, which we refer to as *siblings* (see Fig. 11). The split maps black edges $\{u, v\}$ and $\{v, w\}$ in $UBG(\pi)$ onto black edges $\{u, v_1\}$ and $\{v_2, w\}$ in B, and at least one of $\{u, v\}$ or $\{v, w\}$ belongs to a trivial cycle (see Observation 2), which means that at least one of these black edges also appears as a grey edge in $UBG(\pi)$.

Trivial cycles in $UBG(\pi)$ are mapped onto trivial cycles in B, which allows us to preserve the other grey edge without ambiguity: i.e., if grey edge $\{u, v\}$ (resp. $\{v, w\}$) belongs to $UBG(\pi)$, then grey edge $\{u, v_1\}$ (resp. $\{v_2, w\}$) belongs to B. The graph B is the breakpoint graph of a spin σ of π, which we obtain from π by assigning numbers in $\{1, 2, \ldots, 2n + 1\}$ as follows: starting from vertex $i = 0$, assign number $i + 1$ to its grey neighbour x in B; if $i < 2n$, x comes from a split vertex in $UBG(\pi)$, and its sibling is the only vertex to its left or to its right that is not connected to x by a black edge. That sibling receives number $i + 2$, and we repeat the process until all vertices have been labelled, thereby completing the inverse of the unsigned translation (recall Definition 4). Siblings in increasing (resp. decreasing) order yield a $+$ (resp. $-$) sign for the element they originate from. Finally, the orientation and the length of all cycles in $\mathscr{D}^*$ are clearly preserved in B, and therefore σ is simple. □

Simple permutations can be sorted by prefix signed reversals in polynomial time. Their distance can be computed using the following structure and Theorem.

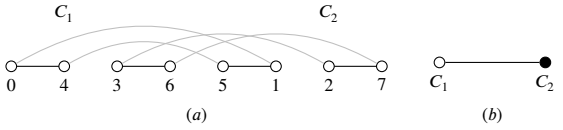

Fig. 12. (*a*) The breakpoint graph of the signed permutation $\pi = (-2\ -3\ 1)$. (*b*) The corresponding graph $H(\pi)$. C_1 is the leftmost vertex and is nonoriented, while C_2 is oriented.

Definition 15. [15] Let $H(\pi)$ be the graph whose vertices are the cycles in $BG(\pi)$ and whose edges connect two vertices if the corresponding cycles intersect. A *component* of $BG(\pi)$ is a connected component of $H(\pi)$; it is *oriented* if a vertex of that component in $H(\pi)$ corresponds to an oriented cycle in $BG(\pi)$, and *unoriented* otherwise.

See Fig. 12 for an example. Following Labarre and Cibulka [4], we refer to the component that contains the leftmost cycle as the *leftmost* component.

Theorem 7. *[4] For every simple permutation π in $S_n^{\pm}$, we have:*

$$psrd(\pi) = n + 1 + c(BG(\pi)) - 2c_1(BG(\pi)) + t(\pi) - \begin{cases} 0 & \text{if } \pi_1 = 1 \\ 2 & \text{otherwise} \end{cases},$$

where $t(\pi) = 1$ if $\pi_1 \neq 1$ and the leftmost component of $BG(\pi)$ is unoriented, and 0 otherwise.

Corollary 3. *There is a polynomial-time approximation algorithm for* Sorting by Prefix Reversals *which, applied on any pseudo-simple permutation π, returns a solution of size at most $prd(\pi) + 1$.*

Proof. Let π be a pseudo-simple permutation and σ be its simple spin obtained using Observation 3. Then $prd(\pi) \leq psrd(\sigma)$, whose value can be computed using Theorem 7; since σ is a spin of π, any sorting sequence for σ can be applied to π by ignoring signs. For the additive performance guarantee, let $g(\pi)$ denote the right-hand side of Corollary 1 and note that, as observed for Equation (1) (in Observation 1) and Equation (3) (in the proof of Theorem 6), $g(\pi)$ is equal to 0 iff π is the identity genome. We also note that $\pi_1 = 1$ if and only if $\sigma_1 = 1$. Moreover, we have $c(BG(\sigma)) = c^*(UBG(\pi))$ and $c_1(BG(\sigma)) = c_1^*(UBG(\pi))$. We show that $psrd(\sigma) - g(\pi) \leq 1$, distinguishing between the following two cases.

1. If $\pi_1 = 1$, then $\sigma_1 = 1$ and $t(\sigma) = 0$. In order to maximise $psrd(\sigma) - g(\pi)$, assume $\{p_2, 2\}$ is a breakpoint; we have

$$\begin{aligned} psrd(\sigma) - g(\pi) &\leq n + 1 + c(BG(\sigma)) - 2c_1(BG(\sigma)) \\ &\quad - (n + 1 + c^*(UBG(\pi)) - 2c_1(UBG(\pi)) - 1) \\ &= 1. \end{aligned}$$

2. If $\pi_1 \neq 1$, then $\sigma_1 \neq 1$; we have

$$\begin{aligned} psrd(\sigma) - g(\pi) &= n + 1 + c(BG(\sigma)) - 2c_1(BG(\sigma)) + t(\sigma) - 2 \\ &\quad - (n + 1 + c^*(UBG(\pi)) - 2c_1(UBG(\pi)) - 2) \\ &= t(\sigma) \leq 1. \quad \square \end{aligned}$$

We omit the pseudocode of the algorithm outlined in the proof of Corollary 3: it is sufficient to use Algorithm 4.1 in Labarre and Cibulka's paper [4], which yields an optimal sorting sequence for any simple permutation σ, and can be implemented to run in time $O(n^{3/2})$.

5. Conclusions and future work

In this paper, we focused on the problem of sorting genomes by prefix DCJs, a problem that had not yet been studied in its prefix-constrained version. We showed that both the signed and the unsigned versions are solvable in polynomial time. Since prefix DCJs generalise prefix reversals, we also provided several algorithmic results for Sorting by Unsigned Prefix Reversals, including a 1-absolute, thus nearly optimal, approximation algorithm. Nevertheless, several questions remain open: for instance, can the new lower bound introduced in section 2 help improve the 2-approximation ratios known for both Sorting by Signed Prefix Reversals and Sorting by Unsigned Prefix Reversals? Although a proof has eluded us so far, we think that our 1-absolute approximation algorithm for pseudo-simple permutations is in fact optimal. Moreover, can we take advantage of the study on pseudo-simple permutations to extend the result in Corollary 3 to obtain a c-absolute approximation algorithm (with c constant) for more general permutations? Finally, we studied the case where both source and target genomes are unichromosomal and linear; it would be interesting to extend this study to a more general context where input genomes can be multichromosomal and not necessarily linear.

CRediT authorship contribution statement

Guillaume Fertin: Writing – review & editing, Writing – original draft, Formal analysis, Conceptualization. **Géraldine Jean:** Writing – review & editing, Writing – original draft, Formal analysis, Conceptualization. **Anthony Labarre:** Writing – review & editing, Writing – original draft, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] A. Labarre, Sorting by prefix block-interchanges, Theor. Comput. Sci. 958 (2023) 113857, https://doi.org/10.1016/j.tcs.2023.113857.
[2] G. Fertin, A. Labarre, I. Rusu, E. Tannier, S. Vialette, Combinatorics of Genome Rearrangements, Computational Molecular Biology, MIT Press, 2009.
[3] S. Yancopoulos, O. Attie, R. Friedberg, Efficient sorting of genomic permutations by translocation, inversion and block interchange, Bioinformatics 21 (2005) 3340–3346.
[4] A. Labarre, J. Cibulka, Polynomial-time sortable stacks of burnt pancakes, Theor. Comput. Sci. 412 (2011) 695–702.
[5] L. Bulteau, G. Fertin, I. Rusu, Pancake flipping is hard, J. Comput. Syst. Sci. 81 (2015) 1556–1574, https://doi.org/10.1016/j.jcss.2015.02.003, https://www.sciencedirect.com/science/article/pii/S0022000015000124.
[6] D.E. Knuth, Sorting and Searching, The Art of Computer Programming, vol. 3, Addison-Wesley, 1995.
[7] M. Bóna, Combinatorics of Permutations, second edition, Discrete Mathematics and Its Applications, CRC Press, 2012.
[8] D. Bienstock, O. Günlük, A degree sequence problem related to network design, Networks 24 (1994) 195–205, https://doi.org/10.1002/net.3230240402.
[9] S.B. Akers, B. Krishnamurthy, D. Harel, The star graph: an attractive alternative to the n-cube, in: Proceedings of the Fourth International Conference on Parallel Processing, Pennsylvania State University Press, 1987, pp. 393–400.
[10] V. Bafna, P.A. Pevzner, Genome rearrangements and sorting by reversals, SIAM J. Comput. 25 (1996) 272–289.
[11] A. Caprara, Sorting permutations by reversals and Eulerian cycle decompositions, SIAM J. Discrete Math. 12 (1999) 91–110 (electronic).
[12] A. Kotzig, Moves without forbidden transitions in a graph, Mat. čas. 18 (1968) 76–80.
[13] P.A. Pevzner, DNA physical mapping and alternating eulerian cycles in colored graphs, Algorithmica 13 (1995) 77–105.
[14] S. Hannenhalli, P.A. Pevzner, To cut... or not to cut (applications of comparative physical maps in molecular evolution), in: É. Tardos (Ed.), Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 28-30 January 1996, Atlanta, Georgia, USA, ACM/SIAM, 1996, pp. 304–313.
[15] S. Hannenhalli, P.A. Pevzner, Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals, J. ACM 46 (1999) 1–27.