

Sorting Genomes by Prefix Double-Cut-and-Joins

Guillaume Fertin¹[0000–0002–8251–2012], Géraldine Jean¹[0000–0002–1534–2682],
and Anthony Labarre²[0000–0002–9945–6774]

¹ Nantes Université, CNRS, LS2N, UMR 6004, F-44000 Nantes, France

² LIGM, CNRS, Université Gustave Eiffel, F-77454 Marne-la-Vallée, France
{guillaume.fertin,geraldine.jean}@univ-nantes.fr,
anthony.labarre@univ-eiffel.fr

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). Depending on whether we consider both options or reversals only, our main results are: (1) new structural lower bounds based on the breakpoint graph for sorting by unsigned prefix reversals, unsigned prefix DCJs, or signed prefix DCJs; (2) a polynomial-time algorithm for sorting by signed prefix DCJs, thus answering an open question in [8]; (3) a $3/2$ -approximation for sorting by unsigned prefix DCJs, which is, to the best of our knowledge, the first sorting by *prefix* rearrangements problem that admits an approximation ratio strictly smaller than 2 (with the obvious exception of the polynomial-time solvable problems); and finally, (4) an FPT algorithm for sorting by unsigned prefix DCJs parameterised by the number of breakpoints in the genome.

Keywords: Genome Rearrangements · Prefix Reversals · Prefix DCJs · Lower Bounds · Algorithmics · FPT · Approximation algorithms.

1 Introduction

Genome rearrangements is a classical paradigm to study 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 strand 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 [5]. The *double-cut-and-join* (or DCJ) operation introduced by Yancopoulos *et al.* [11] encompasses all the rearrangements mentioned above: it consists in cutting the genome in two different places and joining the four 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 [8] for a summary of existing results). Our interest in prefix rearrangements is therefore mostly theoretical: techniques that apply in the unrestricted 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). Moreover, prefix DCJs applied to such genomes allow to exactly mimic 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.

Based on the study of the *breakpoint graph*, we first show new structural lower bounds for the problems SORTING BY UNSIGNED PREFIX DCJs and SORTING BY SIGNED PREFIX DCJs. Since prefix reversals are particular cases of prefix DCJs, we can extend this result to SORTING BY UNSIGNED PREFIX REVERSALS (it has been already shown for SORTING BY SIGNED PREFIX REVERSALS in [9]). Thanks to these preliminary results, we are able to answer an open question from [8] by proving that SORTING BY SIGNED PREFIX DCJs is in P just like the unrestricted case [11]. However, while sorting by unsigned DCJs is NP-hard [4], the computational complexity of the prefix-constrained version of this problem is still unknown. We provide two additional results: a $3/2$ -approximation algorithm, which is, to the best of our knowledge, the first sorting by *prefix* rearrangements problem that admits an approximation ratio strictly smaller than 2 (with the obvious exception of the polynomial-time solvable problems); and an FPT algorithm parameterised by the number of breakpoints in the genome. Due to space constraints, some of the proofs are deferred to the Appendix.

1.1 Permutations, genomes, and rearrangements

We begin with the simplest models for representing organisms.

Definition 1. A (unsigned) permutation of $[n] = \{1, 2, \dots, n\}$ is a bijective application of $[n]$ onto itself. A signed permutation of $\{\pm 1, \pm 2, \dots, \pm n\}$ is a bijective application of $\{\pm 1, \pm 2, \dots, \pm n\}$ onto itself that satisfies $\pi_{-i} = -\pi_i$. The identity permutation is the permutation $\iota = (1 \ 2 \ \dots \ n)$.

We study transformations based on the following well-known operation.

Definition 2. 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 :

$$\rho(i, j) = \begin{pmatrix} 1 & \dots & i-1 & \underline{i} & \underline{i+1} & \dots & j-1 & \underline{j} & \underline{j+1} & \dots & n \\ 1 & \dots & i-1 & j & j-1 & \dots & i+1 & i & j+1 & \dots & n \end{pmatrix}.$$

A signed reversal $\bar{\rho}(i, j)$ 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 :

$$\bar{\rho}(i, j) = \begin{pmatrix} 1 & \dots & i-1 & \underline{i} & \underline{i+1} & \dots & j-1 & \underline{j} & \underline{j+1} & \dots & n \\ 1 & \dots & i-1 & -j & -(j-1) & \dots & -(i+1) & -i & j+1 & \dots & n \end{pmatrix}.$$

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

A reversal ρ applied to a permutation π transforms it into another permutation $\sigma = \pi\rho$. When the distinction matters, we mention whether objects or transformations are signed or unsigned; otherwise, we omit those qualifiers to lighten the presentation. The following model is a straightforward generalisation of unsigned permutations.

Definition 3. A genome G is a collection of vertex-disjoint paths and cycles over $\{0, 1, 2, \dots, n+1\}$. 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 $(0, 1, 2, \dots, n+1)$.

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

Definition 4. 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 set $\{\{0, 4\}, \{4, 3\}, \underline{\{3, 6\}}, \{1, 2\}, \underline{\{2, 1\}}, \underline{\{5, 5\}}\}$ 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 $b(\pi) = b(G_\pi)$.

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 G_1 in Figure 1). The following operation builds on that view to generalise reversals.

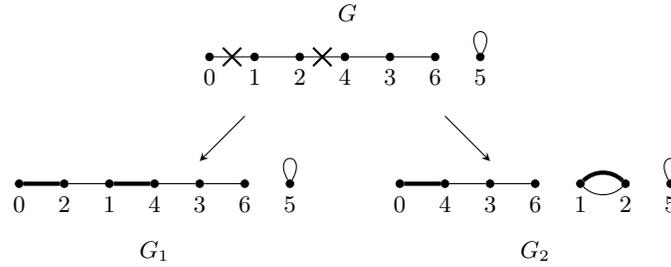


Fig. 1. Cutting edges $\{0, 1\}$ and $\{2, 4\}$ from the nonlinear genome G produces genome G_1 with a reversed segment, if we add edges $\{0, 2\}$ and $\{1, 4\}$, or genome G_2 with an extracted cycle if we add $\{0, 4\}$ and $\{1, 2\}$ instead.

Definition 5. [11] Let $e = \{u, v\} \neq f = \{w, x\}$ be two edges of a genome G . The double-cut-and-join (or DCJ for short) $\delta(e, f)$ applied to G transforms G into a genome G' by replacing edges e and f with either $\{\{u, w\}, \{v, x\}\}$ or $\{\{u, x\}, \{v, w\}\}$. δ is a prefix DCJ if either $0 \in e$ or $0 \in f$.

DCJs that do not correspond to reversals extract paths from genomes and turn them into cycles (see G_2 in Figure 1). Signed permutations can be generalised to signed genomes as well. The definition of a signed linear genome is more complicated than in the unsigned case, and is based on the following notion.

Definition 6. Let π be a signed permutation. The unsigned translation of π is the unsigned permutation π' obtained by mapping π_i onto the sequence $(2\pi_i - 1, 2\pi_i)$ if $\pi_i > 0$, or $(2|\pi_i|, 2|\pi_i| - 1)$ if $\pi_i < 0$, for $1 \leq i \leq n$; and adding two new elements $\pi'_0 = 0$ and $\pi'_{2n+1} = 2n + 1$.

Definition 7. A signed genome G is a perfect matching over the set $\{0, 1, 2, \dots, 2n + 1\}$. G is linear if there exists a signed permutation π such that $E(G) = \{\{\pi'_{2i}, \pi'_{2i+1}\} \mid 0 \leq i \leq n\}$. The signed identity genome is the perfect matching $\{\{2i, 2i + 1\} \mid 0 \leq i \leq n\}$.

DCJs immediately generalise to signed genomes: they may cut any pair of edges of the perfect matching, and recombine their endpoints in one of two ways.

Finally, we will be using different kinds of graphs in this work with a common notation. The *length* of a cycle in a graph G is the number of elements³ it contains, and a k -cycle is a cycle of length k : it is *trivial* if $k = 1$, and *nontrivial* otherwise. We let $c(G)$ (resp. $c_1(G)$) denote the number of cycles (resp. 1-cycles) in G .

³ The definition of an element will depend on the graph structure and will be explicitly stressed.

1.2 Problems

We study several specialised versions of the following problem. A *configuration* is a permutation or a genome, and the *identity configuration* is the identity permutation or genome, depending on the type of the initial configuration.

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 linear genome and Ω is the set of all prefix DCJs;
- SORTING BY SIGNED PREFIX DCJS, where G is a signed linear genome and Ω is the set of all prefix DCJs;
- SORTING BY UNSIGNED PREFIX REVERSALS, where G is an unsigned permutation and Ω is the set of all prefix reversals;
- SORTING BY SIGNED PREFIX REVERSALS, where G is a signed permutation 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;
- $prd(\pi)$ for the prefix reversal distance of an unsigned permutation π , and $psrd(\pi)$ for its signed version.

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 The Signed Case

We generalise a lower bounding technique introduced in the context of SORTING BY SIGNED PREFIX REVERSALS [9]. It is based on the following structure.

Definition 8. [2] *Given a signed permutation π , let π' be its unsigned translation. The breakpoint graph of π is the undirected edge-bicoloured graph $BG(\pi)$ with ordered vertex set $(\pi'_0 = 0, \pi'_1, \pi'_2, \dots, \pi'_{2n}, \pi'_{2n+1} = 2n + 1)$ and whose edge set consists of:*

- black edges $\{\pi'_{2i}, \pi'_{2i+1}\}$ for $0 \leq i \leq n$;
- grey edges $\{\pi'_{2i}, \pi'_{2i} + 1\}$ for $0 \leq i \leq n$.

See Figure 2 for an example. Following Definition 7, the *breakpoint graph* of a signed linear genome is simply the union of that genome (which plays the role of black edges) and of the signed identity genome (which plays the role of grey edges). 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*. Black edges play the role of elements in that graph, so the *length* of a cycle in a breakpoint graph is the number of black edges it contains.

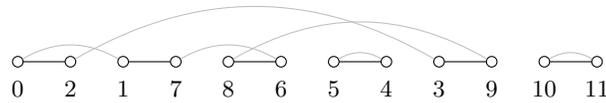


Fig. 2. The breakpoint graph $BG(\pi)$ of $\pi = -1\ 4\ -3\ -2\ 5$.

To bound the prefix DCJ distance, we use a connection between the effect of a DCJ on the breakpoint graph and the effect of *algebraic transpositions*, or *exchanges*, on the classical cycles of a permutation.

Definition 9. 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.

We let $\Gamma(\pi)$ denote the (directed) graph of a permutation π , with vertex set $[n]$ and which contains an arc (i, j) whenever $\pi_i = j$. Exchanges act on two elements that belong either to the same cycle in $\Gamma(\pi)$ or to two different cycles, and therefore $|c(\Gamma(\pi)) - c(\Gamma(\pi\varepsilon(i, j)))| \leq 1$. The following result allows the computation of the prefix exchange distance $ped(\pi)$ in polynomial time, and will be useful to our purposes.

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

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

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

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

Proof. As observed in [11], a DCJ acts on at most two cycles of $BG(G)$ and can therefore change the number of cycles by at most one. This analogy with the effect of exchanges on the cycles of a permutation is preserved under the prefix constraint, and the lower bound then follows from Theorem 1. \square

Since (prefix) signed reversals are a subset of (prefix) signed DCJs, the result below from [9] is a simple corollary of Theorem 2.

Theorem 3. [9] *For any signed permutation π , we have*

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

2.2 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 slightly different, but the definition of the length of a cycle remains unchanged.

Definition 10. [2] *The unsigned breakpoint graph of an unsigned permutation π is the undirected edge-bicoloured graph $UBG(\pi)$ with ordered vertex set $(\pi_0 = 0, \pi_1, \pi_2, \dots, \pi_n, \pi_{n+1} = n + 1)$ and whose edge set consists of:*

- black edges $\{\pi_i, \pi_{i+1}\}$ for $0 \leq i \leq n$;
- grey edges $\{\pi_i, \pi_i + 1\}$ for $0 \leq i \leq n$.

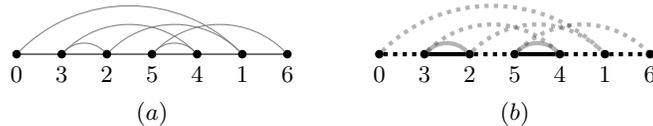


Fig. 3. (a) The unsigned breakpoint graph $UBG(\pi)$ of $\pi = 3\ 2\ 5\ 4\ 1$; (b) an optimal decomposition of $UBG(\pi)$ into two trivial cycles (thick) and one 4-cycle (dotted).

Figure 3(a) shows an example of an unsigned breakpoint graph. Following Definition 3, the *breakpoint graph of an unsigned linear genome* is simply the union of that genome (which plays the role of black edges) and of the identity genome (which plays the role of grey edges). Vertices 0 and $n + 1$ in the unsigned breakpoint graph have degree 2, and all other vertices have degree 4. The unsigned breakpoint graph also decomposes into alternating cycles, but the decomposition is no longer unique. For any genome G and an arbitrary decomposition \mathcal{D} of $UBG(G)$, let $c^{\mathcal{D}}$ (resp. $c_1^{\mathcal{D}}$) denote the number of cycles (resp. trivial cycles) of $UBG(G)$ in \mathcal{D} . We call \mathcal{D} *optimal* if it minimises $c^{\mathcal{D}} - 2c_1^{\mathcal{D}}$ (see Figure 3(b)). The following result characterises optimal decompositions (see Appendix for the proof).

Lemma 1. *Let G be a genome and \mathcal{D} be a decomposition of $UBG(G)$. Then \mathcal{D} is optimal iff it maximises the number of trivial cycles and minimises the number of nontrivial cycles.*

As a result, we obtain the following lower bound on the prefix DCJ distance, 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 $UBG(G)$.

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

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

Proof. Follows from the fact that DCJs affect the number of cycles in a decomposition by at most one, Theorem 1, and Lemma 1. \square

As an immediate corollary, the above lower bound is also a lower bound on $prd(\pi)$, since (prefix) reversals are a subset of (prefix) DCJs.

Corollary 1. *For any unsigned permutation π , we have*

$$prd(\pi) \geq n + 1 + c^*(UBG(\pi)) - 2c_1^*(UBG(\pi)) - \begin{cases} 0 & \text{if } \pi_1 = 1, \\ 2 & \text{otherwise.} \end{cases} \quad (4)$$

We now show that an optimal decomposition 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 [3] (note that in that context, an optimal decomposition *maximises* the 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. [7,10] *A bicoloured connected graph contains an alternating Eulerian cycle iff the number of incident edges of each colour is the same at every vertex.*

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

Proof. Straightforward: extract all trivial cycles from $UBG(G)$. Each connected component in the resulting graph then corresponds to a cycle (Corollary 2). \square

Finally, we note that the lower bound of Theorem 4 is always at least as large as the number of breakpoints (see Appendix for the proof).

Proposition 2. *For any unsigned genome G , the lower bound from Equation 3 is greater than or equal to $b(G)$, and the gap that separates both bounds can be arbitrarily large.*

3 Prefix DCJs

3.1 Signed Prefix DCJs

We give a polynomial-time algorithm for SORTING BY SIGNED PREFIX DCJs.

Theorem 5. *The SORTING BY SIGNED PREFIX DCJs problem is in P.*

Proof. We show that the lower bound of Theorem 2 is tight. For convenience, let $g(G)$ denote the right-hand side of Equation 1, and let π' denote the unsigned translation of the underlying signed permutation π from which G is obtained (recall Definition 7 and the fact that G is linear):

- if $\pi'_1 \neq 1$: then the grey edge $\{\pi'_1, x\}$ connects by definition π'_1 to an element $x \in \{\pi'_1 - 1, \pi'_1 + 1\}$. Let $\{x, y\}$ be the black edge incident with x ; then the prefix DCJ that replaces $\{0, \pi'_1\}$ and $\{x, y\}$ with $\{0, y\}$ and $\{\pi'_1, x\}$ creates one or two new 1-cycles, depending on the value of y . Let G' denote the resulting genome:

1. if $y \neq 1$, then

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

2. if $y = 1$, then

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

Therefore, the value of the lower bound decreases by one in both cases.

- otherwise, let i be the smallest index such that $|\pi'_{2i-1} - \pi'_{2i}| \neq 1$. Then the prefix DCJ that replaces black edges $\{0, \pi'_1\}$ and $\{\pi'_{2i-1}, \pi'_{2i}\}$ with $\{0, \pi'_{2i-1}\}$ and $\{\pi'_1, \pi'_{2i}\}$ decreases the number of 1-cycles by 1. Let us again use G' to denote the resulting genome; then

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

□

3.2 Unsigned Prefix DCJs

The complexity of the SORTING BY UNSIGNED PREFIX DCJs problem remains open, and we conjecture it to be NP-complete. Here, we prove two results, both

based on the number of breakpoints. The first one is a $3/2$ -approximation algorithm for solving SORTING BY UNSIGNED PREFIX DCJs (Theorem 6), the second one is a FPT algorithm with respect to $b(G)$ (Theorem 7).

We start with our approximation algorithm. First, observe that prefix DCJs on linear genomes may produce nonlinear genomes, but the structure of these genomes is nonetheless not arbitrary. We characterise some of their properties in the following result, which will be useful later on.

Lemma 2. *Let G be a linear genome and S be an arbitrary sequence of prefix DCJs that transform G into a new genome G' . Then:*

1. G' contains exactly one path, whose endpoints are 0 and $n + 1$;
2. if G' contains any other component, then that component is a cycle.

Proof. By induction on $k = |S|$. If $k = 0$, then the claim clearly holds. Otherwise, let δ be a prefix DCJ that cuts edges $e = \{0, v\}$ and $f = \{w, x\}$ from a genome G'' obtained from G by $k - 1$ prefix DCJs; by hypothesis, 0 and $n + 1$ are the endpoints of the only path P of G'' . If both e and f belong to P , then δ either extracts a subpath Q from P that will become a cycle, or reverses a subpath R of P ; in both cases, neither Q nor R contains 0 nor $n + 1$, which become extremities of $P \setminus Q$ (or of the path obtained from P by reversing R). Otherwise, since $e = \{0, v\}$, by hypothesis f belongs to a cycle, and both ways of recombining the extremities of e and f yield a path starting with 0 and ending with $n + 1$, preserving any other cycle of G'' . \square

We will need the following lower bound.

Lemma 3. *For any genome G , we have $pdcj(G) \geq b(G)$. Moreover, if G is unsorted and contains $\{0, 1\}$ and $\{1, 2\}$, then $pdcj(G) > b(G)$.*

Proof. The first claim follows directly from Theorem 4 and Proposition 2. For the second claim, if G is unsorted and contains $\{0, 1\}$ and $\{1, 2\}$, then any new edge $\{1, y\}$ that would replace $\{0, 1\}$ would yield a breakpoint — either because 1 and y cannot be consecutive in values or, in the event that $y = 2$, because edge $\{1, 2\}$ would get multiplicity 2 and thereby would also count as a breakpoint. \square

We are now ready to prove our upper bound on $pdcj(G)$.

Lemma 4. *For any linear genome G , we have $pdcj(G) \leq \frac{3b(G)}{2}$.*

Proof. Assume G is not the identity genome, in which case the claim trivially holds. We have two cases to consider:

1. if $\{0, v\} \in G$ with $v \neq 1$, then G contains an element $x \in \{v - 1, v + 1\}$ that is not adjacent to v . By Lemma 2, every vertex in G has degree 1 or 2, so x has a neighbour y such that $\{x, y\}$ is a breakpoint (either because $|x - y| \neq 1$ or because $\{x, y\}$ has multiplicity two). The prefix DCJ that replaces $\{0, v\}$ and the breakpoint $\{x, y\}$ with the adjacency $\{v, x\}$ and $\{0, y\}$ yields a genome G' with $b(G') = b(G) - 1$.

2. otherwise, $\{0, 1\} \in G$. If $\{1, 2\} \notin G$, then 2 has a neighbour y in G such that $\{2, y\}$ is a breakpoint, in which case the prefix DCJ that replaces $\{0, 1\}$ and $\{2, y\}$ with $\{0, y\}$ and $\{1, 2\}$ yields a genome G' with $b(G') = b(G) - 1$. If $\{1, 2\} \in G$, 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 the prefix DCJ δ_1 that replaces $\{0, 1\}$ and $\{k, \ell\}$ with $\{0, \ell\}$ and $\{1, k\}$ yields a genome G' which contains the cycle $(1, 2, \dots, k)$ and with $b(G') = b(G)$. Although δ_1 does not reduce the number of breakpoints, G' allows us to apply two subsequent operations that do:
- (a) since $\{0, \ell\} \in G'$ with $\ell \neq 1$, the analysis of case 1 applies and guarantees the existence of a prefix DCJ δ_2 that produces a genome G'' with $b(G'') = b(G') - 1$.
 - (b) δ_2 replaces $\{0, \ell\}$ and breakpoint $\{a, b\}$ with $\{0, a\}$ and adjacency $\{b, \ell\}$. Since $\{k, \ell\}$ was a breakpoint in G , we have $k < \ell - 1$. Moreover, δ_1 extracted from G a cycle consisting of all elements in $\{1, 2, \dots, k\}$. Therefore, the breakpoint $\{a, b\}$ cut by δ_2 belongs to a component of G'' different from that cycle, which means that $a > k > 1$ and in turn implies that case 1 applies again: there exists a third prefix DCJ δ_3 transforming G'' into a genome G''' such that $b(G''') = b(G'') - 1 = b(G) - 2$.

This implies that, in the worst case, i.e. when $\{0, 1\} \in G$ and $\{1, 2\} \in G$, there exists a sequence of three prefix DCJs that yields a genome G''' with $b(G''') = b(G) - 2$. Therefore, starting with $b(G)$ breakpoints, we can decrease this number by two using at most three prefix DCJs. Since the identity genome has no breakpoint, we conclude that $pdcj(G) \leq \frac{3b(G)}{2}$. \square

Lemma 3 and Lemma 4 immediately imply the existence of a 3/2-approximation for sorting by prefix DCJs, as stated by the following theorem.

Theorem 6. *The SORTING BY UNSIGNED PREFIX DCJS problem is 3/2-approximable.*

Note that Lemma 4 also allows us to show that our approximation algorithm is tight for an unbounded number of genomes. Incidentally, this also shows that the lower bound of Equation 3 is optimal for an unbounded number of genomes (see Appendix for the proof).

Observation 1. *There exists an unbounded number of genomes for which the algorithm described in proof of Lemma 4 is optimal.*

We now turn to proving that SORTING BY UNSIGNED PREFIX DCJS is FPT, as stated by the following theorem.

Theorem 7. *The SORTING BY UNSIGNED PREFIX DCJS problem is FPT parameterised by $b(G)$.*

Proof. The main idea is to use the search tree technique in a tree whose arity and depth are both bounded by a function of $b(G)$. For this, we will use the

notion of *strip* in a genome G , which is defined as a maximal set of consecutive edges (in a path or a cycle of G) that contains no breakpoint. The *length* of a strip is the number of elements it contains, strips of length k are called k -strips; 1-strips are also called *singletons*, and strips of length > 2 are called *long strips*. We need the following result (see Appendix for the proof).

Observation 2. *For any instance of SORTING BY UNSIGNED PREFIX DCJs, there always exists a shortest sorting sequence of prefix DCJs that never cut a long strip.*

Now let us describe our search tree technique: at every iteration starting from G , guess in which location, among the available 2-strips and breakpoints, to operate the rightmost cut. Once this is done, guess among the two possibilities allowed by a DCJ to reconnect the genome. By definition, every strip is framed by breakpoints. Therefore, any genome G has at most $b(G)$ 2-strips (recall that $\{0, x\}$ is never a breakpoint). Altogether, this shows that, at each iteration, the rightmost cut has to be chosen among at most $2b(G)$ possibilities. Because there are two ways to reconnect the cuts in a DCJ, the associated search tree has arity at most $4b(G)$. Moreover, its depth is at most $\frac{3b(G)}{2}$ since $pdcj(G) \leq \frac{3b(G)}{2}$ (Theorem 6). Thus the above described algorithm uses a search tree whose size is a function of $b(G)$ only, which proves the result. More precisely, the overall complexity of the induced algorithm is in $O^*((4b(G))^{1.5b(G)})$. \square

4 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 provided several algorithmic results for both signed and unsigned cases, including computational complexity, approximation and FPT algorithms. Nevertheless, several questions remain open: while we have shown that SORTING BY SIGNED PREFIX DCJs is a polynomial-time solvable problem, what about the computational complexity of SORTING BY UNSIGNED PREFIX DCJs? We were able to design a 3/2-approximation algorithm for the latter problem, which makes it to the best of our knowledge the first occurrence of a prefix rearrangement problem of unknown complexity where a ratio better than 2 has been obtained. Is it possible to improve it further, by making good use of the new lower bound introduced in section 2? Whether or not this lower bound can help improve the 2-approximation ratios known for both SORTING BY UNSIGNED PREFIX REVERSALS and SORTING BY SIGNED PREFIX REVERSALS remains open. Finally, we have 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.

References

1. Sheldon B. Akers, Balakrishnan Krishnamurthy, and Dov Harel. The star graph: An attractive alternative to the n -cube. In *Proceedings of the Fourth International*

- Conference on Parallel Processing*, pages 393–400. Pennsylvania State University Press, August 1987.
2. Vineet Bafna and Pavel A. Pevzner. Genome Rearrangements and Sorting by Reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.
 3. Alberto Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12(1):91–110 (electronic), January 1999.
 4. Xin Chen. On sorting unsigned permutations by double-cut-and-joins. *Journal of Combinatorial Optimization*, 25(3):339–351, apr 2013.
 5. Guillaume Fertin, Anthony Labarre, Irena Rusu, Eric Tannier, and Stéphane Vialette. *Combinatorics of Genome Rearrangements*. Computational Molecular Biology. MIT Press, 2009.
 6. John Kececioğlu and David Sankoff. Efficient bounds for oriented chromosome inversion distance. In Maxime Crochemore and Dan Gusfield, editors, *Combinatorial Pattern Matching*, pages 307–325, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
 7. Anton Kotzig. Moves without forbidden transitions in a graph. *Matematický časopis*, 18(1):76–80, 1968.
 8. Anthony Labarre. Sorting by Prefix Block-Interchanges. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation (ISAAC)*, volume 181 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
 9. Anthony Labarre and Josef Cibulka. Polynomial-time sortable stacks of burnt pancakes. *Theor. Comput. Sci.*, 412(8-10):695–702, 2011.
 10. Pavel A. Pevzner. DNA physical mapping and alternating eulerian cycles in colored graphs. *Algorithmica*, 13(1/2):77–105, 1995.
 11. Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

Appendix: Omitted Proofs

Proof (Lemma 1). We prove both directions separately.

\Rightarrow : if $c_1^{\mathcal{D}}$ is not maximal, then \mathcal{D} contains a nontrivial cycle from which a trivial cycle can be extracted. This yields a new decomposition \mathcal{E} with

$$c^{\mathcal{E}} - 2c_1^{\mathcal{E}} = c^{\mathcal{D}} + 1 - 2(c_1^{\mathcal{D}} + 1) < c^{\mathcal{D}} - 2c_1^{\mathcal{D}}.$$

Likewise, if $c^{\mathcal{D}} - c_1^{\mathcal{D}}$ is not minimal, then \mathcal{D} must contain two nontrivial cycles which can be merged into one, which decreases the quantity $c^{\mathcal{D}} - 2c_1^{\mathcal{D}}$.

\Leftarrow : if \mathcal{D} is not optimal, then there exists another decomposition \mathcal{E} with

$$c^{\mathcal{D}} - 2c_1^{\mathcal{D}} > c^{\mathcal{E}} - 2c_1^{\mathcal{E}}. \quad (5)$$

We distinguish between the following three cases:

1. if $c_1^{\mathcal{D}} = c_1^{\mathcal{E}}$, then $c^{\mathcal{D}} > c^{\mathcal{E}}$, so $c^{\mathcal{D}} - c_1^{\mathcal{D}} > c^{\mathcal{E}} - c_1^{\mathcal{E}} = c^{\mathcal{E}} - c_1^{\mathcal{E}}$ and therefore \mathcal{D} does not minimise the number of nontrivial cycles;
2. if $c^{\mathcal{D}} = c^{\mathcal{E}}$, then $c_1^{\mathcal{D}} < c_1^{\mathcal{E}}$, and therefore \mathcal{D} does not maximise the number of trivial cycles;
3. if neither of the above holds, then
 - (a) either $c^{\mathcal{D}} < c^{\mathcal{E}}$, in which case Equation 5 implies $c_1^{\mathcal{D}} < c_1^{\mathcal{E}}$ and therefore \mathcal{D} does not maximise the number of trivial cycles;
 - (b) or $c^{\mathcal{D}} > c^{\mathcal{E}}$, in which case either $c_1^{\mathcal{D}} < c_1^{\mathcal{E}}$, and therefore \mathcal{D} does not maximise the number of trivial cycles; or $c_1^{\mathcal{D}} > c_1^{\mathcal{E}}$, in which case Equation 5 yields $c^{\mathcal{D}} - c_1^{\mathcal{D}} > c^{\mathcal{E}} - c_1^{\mathcal{E}} + (c_1^{\mathcal{D}} - c_1^{\mathcal{E}})$, which implies that \mathcal{E} has fewer nontrivial cycles than \mathcal{D} since $c_1^{\mathcal{D}} - c_1^{\mathcal{E}} > 0$.

□

Proof (Proposition 2). In order to prove the inequality, we distinguish between two cases:

1. if $\{0, 1\} \notin G$, then the lower bound from Equation 3 has value

$$n + c^*(UBG(G)) - 2c_1^*(UBG(G)) - 1. \quad (6)$$

Note that, by definition, each trivial cycle in $UBG(G)$ corresponds to an edge $\{i, i + 1\}$, which is therefore an adjacency. Moreover, since $\{0, 1\} \notin G$, 0 and 1 do not form a trivial cycle, and consequently

$$n = b(G) + c_1^*(UBG(G)). \quad (7)$$

Finally, since $\{0, 1\} \notin G$, G is not the identity genome and we have

$$c^*(UBG(G)) \geq c_1^*(UBG(G)) + 1. \quad (8)$$

Combining equations (6), (7) and (8) yields the claim.

2. if $\{0, 1\} \in G$, then the lower bound from Equation 3 has value

$$n + c^*(UBG(G)) - 2c_1^*(UBG(G)) + 1. \quad (9)$$

In that case, since 0 and 1 form a trivial cycle, we have

$$n + 1 = b(G) + c_1^*(UBG(G)); \quad (10)$$

moreover, since $c^*(UBG(G)) \geq c_1^*(UBG(G))$ always holds, combining equations (9) and (10) yields the claim.

In order to show that the gap between the lower bound of Theorem 4 and $b(G)$ can be arbitrarily large, consider a linear genome G with $n = 6p$ for an arbitrary integer $p \geq 2$. Genome G corresponds to permutation π which is the concatenation of subpermutations $\sigma_1, \sigma_2, \dots, \sigma_p$, where for every $1 \leq i \leq p$, $\sigma_i = (6i - 5)(6i - 3)(6i - 1)(6i - 4)(6i - 2)6i$. For instance, when $p = 3$, we have $\pi = \underbrace{1\ 3\ 5\ 2\ 4\ 6}_{\sigma_1} \underbrace{7\ 9\ 11\ 8\ 10\ 12}_{\sigma_2} \underbrace{13\ 15\ 17\ 14\ 16\ 18}_{\sigma_3}$.

Formally, G contains the following edges, for every $1 \leq i \leq p$: $\{6i - 5, 6i - 3\}$, $\{6i - 3, 6i - 1\}$, $\{6i - 1, 6i - 4\}$, $\{6i - 4, 6i - 2\}$, $\{6i - 2, 6i\}$ and $\{6i, 6i + 1\}$; G also contains edge $\{0, 1\}$.

It can be seen that $c_1^*(UBG(G)) = p + 1$, which correspond to edges $\{6i, 6i + 1\}$, $0 \leq i \leq p$. Consequently, $b(G) = n + 1 - c_1^*(UBG(G)) = 5p$. Finally, G 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 $UBG(G)$. As a consequence, when trivial cycles are removed from $UBG(G)$, p connected components remain, each induced by elements of $[6i - 5; 6i]$, $1 \leq i \leq p$. Hence, as argued in proof of Proposition 1, G contains p nontrivial cycles. This allows us to conclude that $c^*(UBG(G)) = 2p + 1$. Altogether, since $\{0, 1\} \in G$, we have that our lower bound $n + 1 + c^*(UBG(G)) - 2c_1^*(UBG(G))$ evaluates to $6p$, while as mentioned above, $b(G) = 5p$, which is the sought result. \square

Proof (Observation 1.). Let $n = 4p$ where $p \geq 2$ is any integer, and let G be the linear genome corresponding to the following permutation

$$\pi = 1\ 2\ n(n-1)\ 5\ 6(n-4)(n-5)\ 9\ 10(n-8)(n-9)\ 13\ 14 \dots 8\ 7(n-3)(n-2)\ 4\ 3$$

See Figure 4 for an example in the case $p = 4$. More formally, genome G contains the following edges:

1. $\{4i + 1, 4i + 2\}$ for every $0 \leq i \leq p - 1$,
2. $\{4i + 2, n - 4i\}$ for every $0 \leq i \leq p - 1$,
3. $\{4i + 3, n - 4i + 1\}$ for every $0 \leq i \leq p - 1$,
4. $\{4i + 4, 4i + 3\}$ for every $0 \leq i \leq p - 1$, and
5. $\{0, 1\}$.

Edge sets (1), (4) and (5) above correspond to $p + p + 1$ trivial cycles, while edge sets (2) and (3) correspond to the $2p$ breakpoints that exist. Moreover,

when the $2p + 1$ trivial cycles are removed from G , in $UBG(G)$ we are left with a collection of p 2-cycles: for each $0 \leq i \leq p - 1$, we have a 2-cycle whose two black edges are taken from sets (2) and (3) (namely $\{4i + 2, n - 4i\}$ and $\{4i + 3, n - 4i + 1\}$), and the two grey edges are $\{4i + 2, 4i + 3\}$ and $\{n - 4, n - 4i + 1\}$.

Therefore, since $\{0, 1\} \in G$, Equation 3 yields

$$\begin{aligned} pdcj(G) &\geq n + 1 + c^*(UBG(G)) - 2c_1^*(UBG(G)) \\ &= 4p + 1 + (3p + 1) - 2(2p + 1) \\ &= 3p. \end{aligned}$$

Therefore, Equation 3 yields $pdcj(G) \geq 3p$. Moreover, since $b(G) = 2p$, and by Lemma 4, we conclude that $pdcj(G) \leq 3p$. Thus $pdcj(G) = 3p$, which shows that the lower bound from Equation 3 and the algorithm described in Lemma 4 are optimal. \square

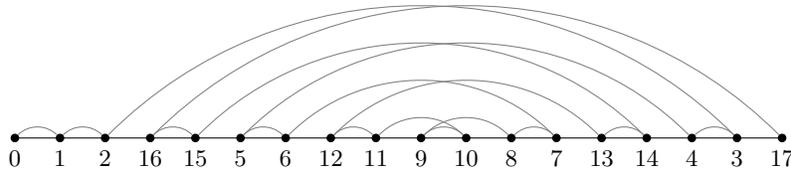


Fig. 4. The breakpoint graph of a genome from the family described in the proof of Observation 1., for $p = 4$.

In the following, we will observe an explicit sorting sequence for a genome, rather than the mere length of such a sequence. We refer to such sequence as a *scenario*, and call it *optimal* if no shorter scenario exists for the given genome.

Proof (Observation 2.). The proof is adapted from proof of Theorem 3 in [6], and relies on similar arguments. However, two main differences exist in our context: first, genomes are unsigned, and thus we need to rely on *long* strips here, whereas in [6] strips of length 2 or more are sufficient. Second, Theorem 3 in [6] is concerned with reversals. The fact that we discuss prefix DCJs here implies more cases to discuss (related to DCJs only, not to the fact that operations are prefix). More precisely, observe a prefix DCJ δ in a genome G . Since by definition δ cuts the edge containing 0, and since by Lemma 2 G is composed of one path P (containing 0) together with cycles, there are three cases to consider: (i) the second cut of δ is in P and δ is a reversal, (ii) the second cut of δ is in P and δ creates a new cycle in G , (iii) the second cut of δ is in a cycle C of G , and δ reincorporates C in P .

Rather than presenting a lengthy case by case analysis, we prefer here to insist on the general arguments that ensure the proof is correct. Indeed, as it

turns out, these arguments are similar whatever case we are in (cases (i), (ii) or (iii)).

The rationale of the proof is as follows: consider a genome, an optimal scenario \mathcal{S} that sorts it, and suppose that in \mathcal{S} , at least one prefix DCJ cuts a long strip. Let δ be the last such prefix DCJ, let G be the genome on which δ is applied, and let S be the long strip that it cuts. We will show the following property, that we call \mathcal{P} for convenience: it is always possible to find an alternate scenario \mathcal{S}' that sorts G , is of same length as \mathcal{S} , and does not cut any long strip. In that case, it is possible to apply \mathcal{P} to every prefix DCJ that cuts a long strip, from the last to the first one, and altogether, the observation is proved.

Now let us describe the alternate scenario. Recall that δ cuts strip S , and suppose S is split into S_A and S_B . Wlog, let us suppose that the longest substrip between S_A and S_B is S_A . Since S is a long strip, we have that S_A is of length at least 2, and in particular we know whether it is ascending or descending (i.e., the successive elements in S_A are in increasing or decreasing order). We then replace δ by the prefix DCJ δ' that does not cut S , replaces S_A by S and deletes S_B from G . The remaining prefix DCJs in scenario \mathcal{S} , among which none of them cuts a long strip, are adapted in \mathcal{S}' as follows: every time S_A is involved in an operation, replace it by S ; besides, delete S_B from all genomes between G and the identity.

The fact that our alternate scenario \mathcal{S}' also sorts G relies on the following argument: in the original scenario \mathcal{S} , strip S will be eventually grouped again, either as S (if it was ascending) or as its reverse (if it was descending), so as to reach the identity genome. We just need to make sure that after the last prefix DCJ in scenario \mathcal{S}' , strip S has the same orientation as in \mathcal{S} . However, this is the case since \mathcal{S} and \mathcal{S}' agree on S_A , which is of length at least 2, and thus carries information on its “orientation” (ascending or descending). Thus strip S in \mathcal{S}' is reversed the same number of times as S_A in \mathcal{S} , which is the sought property.

One final specificity needs to be taken into account. Indeed, it could happen that, in \mathcal{S}' , a prefix DCJ δ_1 which did not cut a strip in \mathcal{S} may now cut a strip. In that case, it suffices to observe that δ_1 occurs strictly after δ . Thus we can reproduce our previous argument to δ_1 , until no prefix DCJ cuts a long strip – which always happens since at distance is 0 or 1 to the identity genome, trivially no strip is cut. \square