JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 16, Number 10, 2009
© Mary Ann Liebert, Inc.
Pp. 1287–1309
DOI: 10.1089/cmb.2009.0088

Research Article

Computation of Perfect DCJ Rearrangement Scenarios with Linear and Circular Chromosomes

SÈVERINE BÉRARD,[1,2] ANNIE CHATEAU,[2] CEDRIC CHAUVE,[3]
CHRISTOPHE PAUL,[2] and ERIC TANNIER[4]

ABSTRACT

We study the problem of transforming a multichromosomal genome into another using Double Cut-and-Join (DCJ) operations, which simulates several types of rearrangements, as reversals, translocations, and block-interchanges. We introduce the notion of a DCJ scenario that does not break families of common intervals (groups of genes co-localized in both genomes). Such scenarios are called perfect, and their properties are well known when the only considered rearrangements are reversals. We show that computing the minimum perfect DCJ rearrangement scenario is NP-hard, and describe an exact algorithm which exponential running time is bounded in terms of a specific pattern used in the NP-completeness proof. The study of perfect DCJ rearrangement leads to some surprising properties. The DCJ model has often yielded algorithmic problems which complexities are comparable to the reversal-only model. In the perfect rearrangement framework, however, while perfect sorting by reversals is NP-hard if the family of common intervals to be preserved is nested, we show that finding a shortest perfect DCJ scenario can be answered in polynomial time in this case. Conversely, while perfect sorting by reversals is tractable when the family of common intervals is weakly separable, we show that the corresponding problem is still NP-hard in the DCJ case. This shows that despite the similarity of the two operations, easy patterns for revervals are hard ones for DCJ, and *vice versa*.

1. INTRODUCTION

A GENERIC FORMULATION OF GENOME REARRANGEMENT PROBLEMS is, given two genomes and some allowed edit operations, to transform one genome into the other using a minimum number of edit operations (Fertin et al., 2009). The solutions are used to estimate an evolutionary distance between species, and to propose possible scenarios that could explain the differences in terms of gene order between the considered genomes (Braga et al., 2008; Darling et al., 2008; Lemaitre et al., 2009). One of the most famous algorithmic results related to genome rearrangements concerns the problem of sorting signed permutations by reversals. This problem aims at computing a shortest sequence of reversals that transforms one signed

[1]Université Montpellier 2, UMR AMAP, Montpellier, France.
[2]CNRS, LIRMM, CNRS UMR55076, Université Montpellier 2, Montpellier, France.
[3]Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada.
[4]INRIA Rhône-Alpes, Université de Lyon, Lyon; Université Lyon 1, CNRS, UMR5558, Laboratoire de Biométrie et Biologie Evolutive, Villeurbanne, France.

permutation into another, and can be solved in polynomial time (Hannenhalli and Pevzner, 1999; Bergeron et al., 2005; Tannier et al., 2007). It was later generalized to handle, still in polynomial time, multi-chromosomal genomes with linear chromosomes, using rearrangements such as translocations, chromosome fusions and fissions (Hannenhalli and Pevzner, 1995; Jean and Nikolski, 2007). Then, a general operation called *Double Cut-and-Join* (DCJ), was introduced in (Yancopoulos et al., 2005). A DCJ can be, among others, a reversal, a translocation, a fusion or a fission, but two consecutive DCJ operations can also simulate a block-interchange or a transposition. Using this generic operation, genomes with circular and linear chromosomes can be handled, as a DCJ can create circular chromosomes from linear ones.

Another way to conceive the evolution of gene orders is to consider groups of genes that are co-localized in the genomes of different species, as these groups are likely to be present in the common ancestral genome and not disrupted during evolution. For two permutations, such groups of co-localized genes can be modeled by *common intervals*. Following the assumption that such common intervals are preserved during evolution leads naturally to the study of rearrangement scenarios that preserve common intervals. Such scenarios, which may not be shortest among all scenarios, are called *perfect* (Figeac and Varré, 2004). Computing a reversal scenario of minimum length that preserves a given subset of the common intervals of two signed permutations is NP-hard (Figeac and Varré, 2004) and several papers have explored this problem, describing families of instances that can be solved in polynomial time (Bérard et al., 2004, 2007; Sagot and Tannier, 2005; Diekmann et al., 2007) and fixed parameter tractable algorithms (Bérard et al., 2007, 2008b).

When comparing algorithmic properties of the reversal and DCJ models, most problems seem to have similar behaviors: the distance and scenario computations can be solved in polynomial time, yet the best complexity varies for the latter by an $O(\sqrt{n})$ factor (Tannier et al., 2007; Bergeron et al., 2006); the median problems are both NP-hard (Caprara, 2003; Tannier et al., 2009); genome halving problems can be solved in polynomial time in both models (El-Mabrouk and Sankoff, 2003; Warren and Sankoff, 2008; Mixtacki, 2008). In this paper we extend the notion of perfect scenario to the DCJ model. We define a notion of scenario preserving common intervals that also allows to use the property of the DCJ model to create temporary circular chromosomes. While the general problem of computing a shortest DCJ scenario that preserves a family $\mathcal{F}$ of common intervals is still NP-hard, our results point to interesting differences between the reversal and DCJ models. If the family of common intervals is *nested* (the intervals do not overlap), we show that finding a perfect DCJ scenario of minimum length is solvable in polynomial time, while it is NP-hard for reversals (Figeac and Varré, 2004); if the family is *weakly separable* (every interval overlaps with some other interval or is the union of overlapping intervals) we show that the DCJ problem is NP-hard, while this case was solved in polynomial time for reversals (Bérard et al., 2007). Table 1 sums up these results.

The NP completeness proof relies on a specific pattern which we prove to be the only pattern that prevents the existence of a polynomial time algorithm, by designing an algorithm whose exponential part only depends on the presence of this pattern.

This article is organized as follows: in Section 2, we introduce genomes, DCJ operations and common intervals and we state formally the perfect DCJ rearrangement problem. In Section 3, we define the different properties of families of common intervals, that lead to different complexity status for the perfect rearrangement problems. In Section 4, we state fundamental structural properties of DCJ scenarios that are required to prove our main results, described in Sections 5 and 6. The properties described in Section 4 can be seen as the conceptual backbone of perfect sorting by DCJ, while the precise results described in Sections 5 and 6 require much more technical proofs. In Section 5, we prove NP-hardness of the general prefect DCJ rearrangement problem. Then we describe in Section 6 an algorithm to solve the perfect DCJ rearrangement problem, whose complexity is exponential only in a term that depends on the presence of a specific pattern of common intervals, proving then that the problem is fixed parameter tractable.

TABLE 1. COMPLEXITY RESULTS FOR THE SHORTEST $\mathcal{F}$-PERFECT SCENARIO COMPUTATION PROBLEM

	$\mathcal{F}$ *nested*	$\mathcal{F}$ *weakly separable*
Reversal	NP-complete (Figeac and Varré, 2004)	Linear (Bérard et al., 2007)
DCJ	Polynomial	NP-complete

This article is an extended version of Bérard et al. (2008a), which contains algorithms only in particular cases of polynomial time complexity, and does not handle all cases of circular genomes. So an algorithm solving the problem in the most general case, with wider classes of genomes where the problem is actually polynomial, is presented here for the first time.

2. GENOMES, INTERVALS, REARRANGEMENTS, AND PERFECT SCENARIOS

2.1. Genomes and intervals

We follow the modeling of a genome introduced in Bergeron et al. (2006). A *gene a* is an oriented sequence of DNA, identified by its *tail* a_t and its *head* a_h. Tails and heads are the *extremities* of the genes. An *adjacency* is an unordered pair of extremities of genes. A *genome* is a set of adjacencies on a set of genes such that one extremity is contained in at most one adjacency. Each adjacency in a genome means that two gene extremities are consecutive along the DNA molecule. In a genome, each gene extremity is adjacent to zero or one other extremity. An extremity x that is not adjacent to any other extremity is called a *telomere*, and can be written as a *telomeric adjacency* xT with a symbol T (we use the same notation for all telomeres).

For a genome Π on a set of genes, we define the graph G_Π: its vertex set is the set of all gene extremities, and its edge set is composed of a_ta_h for every gene a, plus the adjacencies of Π, except telomeric adjacencies. An example of such a graph is drawn on Figure 1.

The graph G_Π is composed of disjoint paths and cycles. Each connected component of G_Π is called a *chromosome* of Π. A chromosome is said to be *linear* if it is a path, and *circular* if it is a cycle. A genome is said to be *linear* if all its chromosomes are linear, *circular* if all its chromosomes are circular, and *mixed* if both are allowed.

An *interval* of Π is a set of genes I, such that the subgraph of G_Π induced by the extremities of genes in I is connected. For example, $\{12, 4, 14, 1, 7, 8\}$ and $\{14, 1, 7, 8\}$ are intervals of genome Π^{ex}, which is represented in Figure 1. An interval I is said to be a *common interval* of two genomes Π and Γ if it is an interval of both.

2.2. Double cut-and-joins

Given a genome Π, a *Double Cut-and-Join* is an operation ρ acting on two adjacencies pq and rs of Π (p, q, r, s are gene extremities, some being possibly T symbols; in particular, we consider valid the adjacency TT). The DCJ operation *cuts* both pq and rs and *joins* either p with r and q with s, or p with s and q with r, creating two new adjacencies. Examples of DCJ operations are shown in Figure 2.

A DCJ operation can reverse an interval in a genome, join two chromosomes into one (fusion), break one chromosome into two (fission), or exchange two intervals from two different chromosomes, each of these intervals containing a telomere (reciprocal translocation). Two consecutive DCJs may result in a block interchange (two intervals exchange their positions), or a transposition (if these two intervals are consecutive): the first DCJ extracts a set of genes and creates a circular chromosome, while the second DCJ reinserts these genes elsewhere in a chromosome. The DCJ operation is thus a very general framework, where temporary circular chromosomes allow to simulate a wide range of genome rearrangements. It was introduced by Yancopoulos et al. (2005), and since then it has been adopted by many others (Bergeron et al., 2006), sometimes under the name "2-break rearrangements" (Alekseyev and Pevzner, 2008).

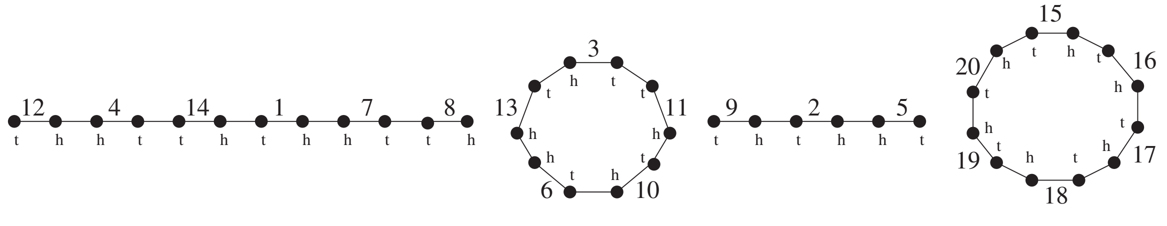

FIG. 1. The graph $G_{\Pi^{ex}}$, where Π^{ex} is given by the union of the linear chromosome $C_1 = \{T12_t, 12_h4_h, 4_t14_t, 14_h1_t, 1_h7_h, 7_t8_t, 8_hT\}$, the circular chromosome $C_2 = \{3_t11_t, 11_h10_t, 10_h6_t, 6_h13_h, 13_t3_h\}$, the linear chromosome $C_3 = \{T9_t, 9_h2_t, 2_h5_h, 5_tT\}$ and the circular chromosome $C_4 = \{15_h16_t, 16_h17_t, 17_h18_t, 18_h19_t, 19_h20_t, 20_h15_t\}$.

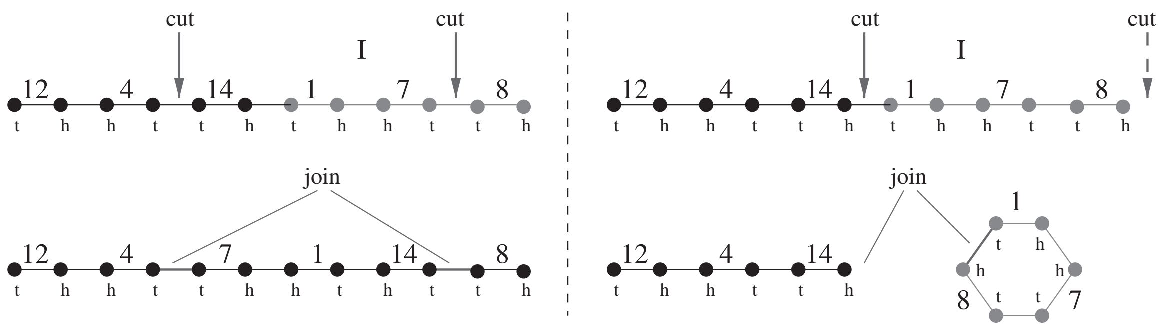

FIG. 2. Two examples of DCJ operations. Left: the DCJ cuts 4_t14_t and 7_t8_t and joins 4_t7_t and 14_t8_t (it is a reversal). Right: the DCJ cuts 14_h1_t and 8_hT and joins 14_hT and 8_h1_t. This operation produces a circular chromosome. The first operation breaks the interval $I = \{1, 7, 8\}$ whereas the second preserves it.

A sequence S of k DCJ operations transforming one genome Π into another genome Γ is called a *DCJ scenario of length k* for the two genomes. The minimum number of DCJ operations needed to transform Π into Γ is the *DCJ-distance* and denoted by $d(\Pi, \Gamma)$.

2.3. Perfect DCJ scenarios

The adjacencies of a genome Π can be partitioned into three classes with respect to a subset I of its genes. This distinction will be central in establishing our results.

Definition 1. Let I be a subset of genes of a genome. An adjacency pq (p and q possibly being T symbols) is said to be *inside I* if the two genes of which p and q are extremities belong to I; it is *outside I* if the two genes of which p and q are extremities do not belong to I; it is a *border* adjacency of I (also called a border of I) if one of the genes of which p and q are extremities belongs to I but not the other.

Notice that a T symbol does not correspond to a gene. So an adjacency involving a T symbol is either outside I or a border of any set I of genes. An interval of a genome Π has then zero or two border adjacencies.

Definition 2. Let I be a set of genes of a genome Π, which has at most two border adjacencies. A DCJ acting on Π *preserves I* if, in the resulting genome, I still has at most two border adjacencies. A DCJ that does not preserve I is said to *break I.*

For example, in Figure 2, the DCJ operation on the left does not preserve the interval $\{1, 7, 8\}$ but the operation on the right does preserve this interval. In other words, a set of genes composed of a linear chromosome and a (possibly empty) set of circular chromosomes is broken by a DCJ if this DCJ creates more than one linear chromosome.

Note that in the genome resulting from a DCJ operation preserving an interval I, I may not be an interval anymore; indeed, a proper subset of the genes that belong to I can form a circular chromosome, which implies that the graph induced by I is not connected anymore. For example, if $I = \{1, 4, 7, 14\}$ in the genome Π^{ex} and a DCJ cuts adjacencies 4_t14_t and 1_h7_h and joins 1_h with 14_t and 4_t with 7_h, then this DCJ preserves I although the subgraph induced by I in the resulting genome is not connected anymore as it contains a path and a cycle. This definition of preserving an interval allows to use the property of the DCJ model to create temporary circular chromosomes to simulate rearrangements such as block interchanges and transpositions, without considering that the creation of temporary circular chromosomes breaks common intervals.

Definition 3. Given a family $\mathcal{F}$ of common intervals of two genomes Π and Γ, a DCJ scenario transforming Π into Γ is said to be *$\mathcal{F}$-perfect* if every DCJ preserves all intervals in $\mathcal{F}$.

The **$\mathcal{F}$-Perfect DCJ problem** consists in, given Π, Γ, and $\mathcal{F}$, computing a $\mathcal{F}$-perfect DCJ scenario of minimum length transforming Π into Γ. When genomes are restricted to signed permutations (they have only one chromosome) and circular chromosomes are not allowed, this definition coincides with the one of perfect scenarios of reversals (Figeac and Varré, 2004; Bérard et al., 2004, 2007, 2008b; Sagot and Tannier, 2005; Diekmann et al., 2007).

3. FAMILIES OF COMMON INTERVALS

We now consider some properties of families of sets that can be satisfied by sets of common intervals of two genomes. We distinguish two cases: the first case concerns the common intervals of linear chromosomes or circular chromosomes with different gene contents, the second case concerns common intervals of circular chromosomes with equal gene contents. This distinction is necessary to present convenient definitions of the properties, but does not change much the general algorithm.

Weakly partitive and weakly bipartitive families of intervals. Weakly partitive and weakly bipartitive families of intervals were introduced in Cunningham and Edmonds (1980). They are useful combinatorial tools often involved in problems concerning modular decomposition of graphs. Here we need to introduce them to characterize the families of common intervals we want to preserve in a perfect DCJ scenario.

In general, given a subset I of a set E, the complementary subset of I is denoted $\bar{I}^E = E \setminus I$. Two subsets I, J of E are said to *overlap* if none of $I \cap J$, $I \setminus J$, $J \setminus I$ is empty. Two subsets I, J of E are said to *bi-overlap* if the two bipartitions $(I, \bar{I}^E)$ and $(J, \bar{J}^E)$ overlap, which occurs when none of $I \cap J, I \cap \bar{J}^E, J \cap \bar{I}^E$ and $\bar{I}^E \cap \bar{J}^E$ is empty.

A family $\mathcal{F}$ of subsets of E is *weakly partitive* if it contains all singletons of E and E itself, and if for every two overlapping subsets I and J of $\mathcal{F}$, $I \cup J$, $I \cap J$, $I \setminus J$ and $J \setminus I$ belong to $\mathcal{F}$. Given a family $\mathcal{F}$, we denote by $\mathcal{F}^p$ the smallest weakly partitive family that contains $\mathcal{F}$.

A family $\mathcal{F}$ of subsets of E is *weakly bipartitive* if it contains all singletons of E and E itself, and $I \in \mathcal{F}$ if and only if $\bar{I}^E \in \mathcal{F}$, and for every two bi-overlapping subsets I and J of $\mathcal{F}$, $I \cup J$, $I \cap J$, $I \setminus J$ and $J \setminus I$ belong to $\mathcal{F}$. Given a family $\mathcal{F}$, we denote by $\mathcal{F}^b$ the smallest weakly bipartitive family that contains $\mathcal{F}$.

Property 1. (de Montgolfier, 2003; McConnell and de Montgolfier, 2005). The family of all common intervals of two linear chromosomes is weakly partitive, and the family of all common intervals of two circular chromosomes having the same gene content is weakly bipartitive.

Nested and weakly separable families of intervals. In a weakly partitive family of common intervals, a common interval I of two genomes Π and Γ is called *strong* if I does not overlap any other common interval. It is *maximal* if it is strong and not contained in another common interval.

Definition 4. A family $\mathcal{F}$ is called *nested*[1] if every element of $\mathcal{F}$ is strong (note this implies that $\mathcal{F} = \mathcal{F}^p$). A family $\mathcal{F}$ of common intervals is called *weakly separable*[2] if every *strong* interval of $\mathcal{F}$ with at least three elements is the union of two overlapping intervals of $\mathcal{F}$.

Note that if $\mathcal{F}$ is weakly separable, it does not imply that $\mathcal{F}^p$ is weakly separable. Note also that, as soon as there are intervals of $\mathcal{F}$ with at least three elements, the nested property and the weakly separable property are mutually exclusive, i.e., $\mathcal{F}$ cannot be both weakly separable and nested.

The decomposition tree of weakly partitive families. Let $\mathcal{F}$ be a family of common intervals of two genomes Π and Γ, such that $\mathcal{F}$ does not cover circular chromosomes with equal gene content. By definition, the sub-family of strong intervals of $\mathcal{F}^p$ is nested. It follows that we can represent the strong common intervals of Π and Γ by a forest, in which each node is a strong common interval of $\mathcal{F}^p$, and its

[1]Note that this definition of nested common intervals differs from the one given in Hoberman and Durand (2005), where a common interval of size k is nested if it contains common intervals of length $1, 2, \dots, k-1$.

[2]The terminology *weakly separable* is inspired by the notion of separable permutations. In separable permutations, the common intervals with the identity have the property (Bouvel & Rossin, 2006). The converse is not true however, justifying the term "weakly."

children are the maximal strong common intervals of $\mathcal{F}^p$ it properly contains (Bérard et al., 2007; Hsu and McConnell, 2003). Each component of this forest is a rooted tree, in which the root is a maximal common interval of Π and Γ. An example of such a tree is given in Figure 3. Given a maximum common interval, the tree can be computed in linear time and space (Bergeron et al., 2008). A node of the forest of strong intervals is called *prime* if it has at least three children and it properly contains no common interval including more than one of its children. The node is *linear* if it has two elements or it is the union of two overlapping common intervals, both containing a subset of its children. An important property of weakly partitive families is that any strong interval of $\mathcal{F}^p$ corresponds to a node which is either prime or linear (Bérard et al., 2007).

Property 2. If a family $\mathcal{F}$ is nested, then all nodes of the decomposition forest of $\mathcal{F}^p$ with at least three children are prime. Furthermore, if a family $\mathcal{F}$ is weakly separable, then the decomposition forest of $\mathcal{F}^p$ has no prime node whose parent is a prime node.

Proof. If $\mathcal{F}$ is nested, then trivially $\mathcal{F}^p = \mathcal{F}$ and as no two intervals overlap, no node of the decomposition tree with at least three children may fit the linear definition. Now if $\mathcal{F}$ is weakly separable, suppose $\mathcal{F}^p$ contains a prime node which is the parent of a prime node N. Then N is not in $\mathcal{F}$, otherwise it is the union of overlapping intervals, so cannot be prime. Then N is the intersection, union or difference of intervals of $\mathcal{F}$. If it is the intersection or the difference of two intervals of $\mathcal{F}$, then there are two different overlapping intervals included in the parent of N, which thus cannot be prime. If it is the union of two overlapping intervals, then it is not prime itself. ■

We do not need to discuss the decomposition tree of weakly bipartitive families, that is known as PC-tree (Hsu and McConnell, 2003), as one of the fundamental properties we describe in Section 4 is that handling these families can be reduced to handling weakly partitive families.

Previous results on perfect rearrangement problems. It is known (Figeac and Varré, 2004) that, given a nested family of common intervals $\mathcal{F}$ of two permutations, it is NP-hard to compute a perfect scenario of reversals of minimum length. Conversely, if $\mathcal{F}$ is weakly separable, by Property 2, the algorithm described in Bérard et al. (2007, 2008b) computes an $\mathcal{F}$-perfect reversal scenario in polynomial time. We prove here the exact opposite results for multichromosomal genomes with DCJ operations. While the hurdle in rearranging genomes by reversals are prime nodes, they are linear nodes of certain types for DCJ rearrangement problems.

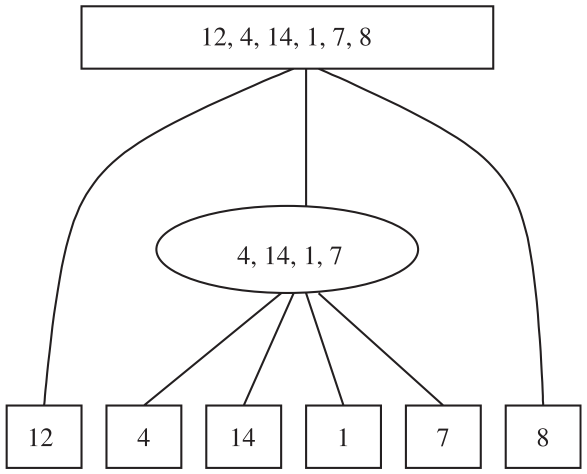

FIG. 3. The tree that represents the strong common intervals of the maximal common interval $I = \{12, 4, 14, 1, 7, 8\}$ of Π^{ex} and Γ^{ex}, given by the union of $C_1' = \{T12_t, 12_h14_h, 14_t7_h, 7_t4_t, 4_h1_h, 1_t8_t, 8_h2_t, 2_h6_t, 6_hT\}$, $C_2' = \{T9_t, 9_h3_t, 3_h10_t, 10_h5_t, 5_h11_h, 11_t13_h, 13_tT\}$ and $C_3' = \{15_h19_t, 19_h18_t, 18_h17_t, 17_h20_h, 20_t16_t, 16_h15_t\}$. Prime nodes are surrounded by an ellipse, while linear nodes are framed by a rectangle.

4. PROPERTIES OF SHORTEST DCJ SCENARIOS

We present in this section several fundamental results on the DCJ distance that will be required to prove the hardness of the perfect DCJ rearrangement problem and to describe our algorithm. The first subsection describes known results on the DCJ distance. The next two subsections describe basic properties of perfect sorting intervals with DCJ: we show that the problem can be reduced to consider only strong intervals of weakly partitive families, and that these strong intervals can be considered independently of each other. Finally, we introduce the central notion of the "sorting direction" of an interval, which is the main difference with the problem of perfect sorting by reversals.

4.1. Computing a shortest DCJ scenario

To compute the DCJ distance, we use the *breakpoint graph* $BP(\Pi, \Gamma)$ of two genomes Π and Γ defined on the same set of genes. This graph, traditionally used in the problem of sorting by reversals (Hannenhalli and Pevzner, 1999), is the graph whose vertex set is the set of extremities of the genes, and in which there is an edge between two vertices x and y if xy is an adjacency in either Π (these are Π-edges) or Γ (Γ-edges). Note that T symbols do not participate. Vertices in this graph have degree zero, one or two; so the graph is a set of paths and cycles, where some paths may have no edge (Fig. 4).

The *DCJ-distance* is immediately readable from the breakpoint graph, as stated by Theorem 1, that restates the main result of Bergeron et al. (2006) in terms of the breakpoint graph instead of the *adjacency graph.*[3]

Theorem 1. (Bergeron et al., 2006). *For two genomes Π and Γ with n genes, let $c(\Pi, \Gamma)$ be the number of cycles of the breakpoint graph $BP(\Pi, \Gamma)$, and $p(\Pi, \Gamma)$ be the number of paths with an even number of edges (including trivial paths with no edge). The DCJ distance is*

$$d(\Pi, \Gamma) = n - \left(c(\Pi, \Gamma) + \frac{p(\Pi, \Gamma)}{2} \right).$$

4.2. The independence of sorting a single interval

We now prove a lemma on the possibility of doing the DCJs inside an interval independently of the DCJs outside, which is an equivalent, for the DCJ model, of a lemma stated for reversals in Figeac and Varré (2004).

Definition 5. We say that a common interval I is *sorted* in a genome Π with respect to a genome Γ if the set of adjacencies inside I in Π contains the set of adjacencies inside I in Γ. If a DCJ scenario results in a genome where I is sorted, we say that this scenario *sorts I* with respect to Γ.

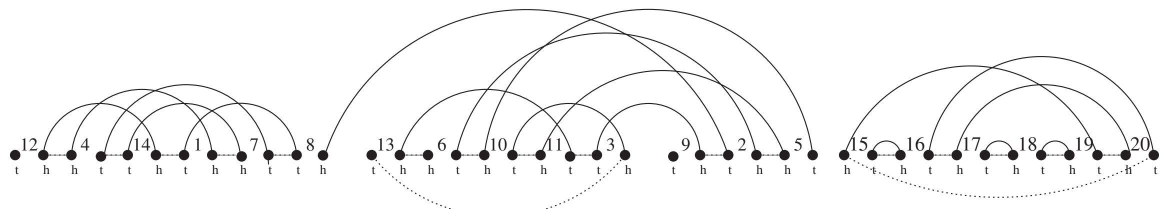

FIG. 4. The breakpoint graph of the genomes Π^{ex} given by the union of $C_1 = \{T12_t, 12_h4_h, 4_t14_t, 14_h1_t, 1_h7_h, 7_t8_t, 8_hT\}$, the circular chromosome $C_2 = \{3_t11_t, 11_h10_t, 10_h6_t, 6_h13_h, 13_t3_h\}$, the linear chromosome $C_3 = \{T9_t, 9_h2_t, 2_h5_h, 5_tT\}$ and the circular chromosome $C_4 = \{15_h16_t, 16_h17_t, 17_h18_t, 18_h19_t, 19_h20_t, 20_h15_t\}$. and Γ^{ex}, given by the union of $C'_1 = \{T12_t, 12_h14_h, 14_t7_h, 7_t4_t, 4_h1_h, 1_t8_t, 8_h2_t, 2_h6_t, 6_hT\}$, $C'_2 = \{T9_t, 9_h3_t, 3_h10_t, 10_h5_t, 5_h11_h, 11_t13_h, 13_tT\}$ and $C'_3 = \{15_h19_t, 19_h18_t, 18_h17_t, 17_h20_h, 20_t16_t, 16_h15_t\}$. Π^{ex}-edges are dotted lines, and Γ^{ex}-edges are plain lines.

[3]The breakpoint graph $BP(\Pi, \Gamma)$, introduced for permutations in Hannenhalli and Pevzner (1999), is the line-graph of the *adjacency graph* introduced in Bergeron et al. (2006).

Note that in the definition above, we do not say that I is sorted with respect to Γ if the set of adjacencies inside I in Π *equals* the set of adjacencies inside I in Γ. Indeed we also consider the case where I forms a circular chromosome in Π and not in Γ, in which case there is an additional adjacency inside I in Π.

We now distinguish different ways a DCJ acts on an interval I, following the classification of adjacencies described in Definition 1.

Definition 6. A DCJ ρ cuts *inside I* if it cuts either two inside adjacencies or one inside and one border adjacency. On the contrary, a DCJ ρ cuts *outside I* if it cuts either two outside adjacencies, one outside and one border adjacency, two border adjacencies, or one inside and one outside adjacency in the case I does not have any border adjacency.

Property 3. A DCJ operation preserves I if and only if it cuts inside or outside I.

Proof.

1. *If a DCJ operation preserves I then it cuts inside or outside I.* Suppose that a DCJ D does not cut inside nor outside I, then by Definition 6, I has two border adjacencies, say i_1o_1 and i_2o_2 and D cuts one inside adjacency, say pq and one outside adjacency say rs, with $i_1, i_2, o_1, o_2, p, q, r, s$ being gene extremities. i_1, i_2, p, q are extremities of genes belonging to I and o_1, o_2, r, s are extremities of genes outside I. The DCJ D removes the adjacencies pq and rs, and creates either pr and qs, or ps and qr. In the two cases, these two new adjacencies are border adjacencies of I, leading I to have four border adjacencies. Thus by Definition 2, D breaks I.
2. *If a DCJ operation cuts inside or outside I then it preserves I.* Cutting inside or outside an interval does not change its number of border adjacencies except cutting outside an interval I if I has no border adjacency, in this latter case, I has 2 border adjacencies in the resulting genome. So it follows immediately from Definition 2 that if a DCJ operation cuts inside or outside I then it preserves I. ■

Lemma 1. *If a DCJ scenario S_0 transforming a genome Π into a genome Γ does not break a common interval I, then there exists a DCJ scenario $S = S_1S_2$ of same length as S_0 for which all operations in S_1 cut inside I and all operations in S_2 cut outside I.*

Proof. Let S_0 be a DCJ scenario of length k between two genomes Π and Γ. No operation breaks interval I, so by Property 3, every operation cuts inside or outside I. In a scenario of length k, each operation has a numbered position from 1 to k. Let S be the scenario with the same length as S_0, such that among all such scenarios, the sum of the positions of all operations cutting inside I is minimized. Suppose that in S there is an operation cutting inside I that occurs after an operation not cutting inside I. In this case, let A and B be two consecutive operations such that B cuts inside I and not A. If A and B cut four adjacencies with no gene extremity in common, then simply exchanging the positions of A and B in S gives another scenario transforming Π into Γ and the sum of the positions of all operations cutting inside I is strictly less than in S, contradicting the definition of S.

So A and B have some common points, one adjacency broken by B and one adjacency broken by A share a common gene extremity. We show now that we can replace A and B by two DCJs C and D such that, C cuts inside I, D cuts outside I and applying C and D instead of A and B produces the same result (i.e. the same set of adjacencies). The proof contains two cases depending on the shape of I (note that the second case is not relevant if I is a circular chromosome in both genomes).

1. First case: I contains two border adjacencies and possibly $l \geq 0$ circular chromosomes. As A cuts outside I and shares a common point with B then A cuts an outside adjacency, say sr, and one of the border adjacencies of I, say pq with p inside I and q outside. A creates one border adjacency, say b_1 and one outside adjacency, say o with $(o, b_1) \in \{(rq, sp), (sq, rp)\}$. As B cuts inside I and shares a common gene extremity with A, then B cuts an inside adjacency, say tu and the border adjacency created by $A : b_1$. B creates an inside adjacency, say i, and a border adjacency, say b_2 with $(i, b_2) \in \{(pt, su), (pu, st), (pu, rt), (pt, ru)\}$. We define the DCJ C that cuts pq and tu to create i and b_3, with $(i, b_3) \in \{(pt, qu), (pu, qt)\}$ and the DCJ D that cuts sr and b_3 to create o and b_2. C cuts inside I, D cuts outside I and applying C and D instead A and B produces the same result.
2. Second case: I does not contain any border adjacencies and $l \geq 1$ circular chromosomes. As A cuts outside I and shares a common point with B then A cuts an outside adjacency, say sr, and an inside adjacency, say vw. A creates two border adjacencies b_1 and b_2 with $(b_1, b_2) \in \{(sv, rw), (sw, rv)\}$. As B cuts inside I and shares a common

gene extremity with A, then B cuts an inside adjacency, say tu and one of the border adjacencies created by A. B creates an inside adjacency, say i, and a border adjacency, say b_3 with $(i, b_3) \in \{(tv, su), (uv, st), (uw, rt), (tw, ru), (uw, st), (wt, su), (tv, ru), (uv, tr)\}$. We define the DCJ C that cuts tu and vw to create i and i', with $(i, i') \in \{(tv, wu), (tw, uv)\}$ and the DCJ D that cuts sr and i' to create b_1 and b_2. C cuts inside I, D cuts outside I and applying C and D instead A and B produces the same result. ■

4.3. Families of intervals

We now show that, for any family $\mathcal{F}$ of common intervals of two genomes, computing a perfect DCJ scenario for $\mathcal{F}$ can be reduced to computing a perfect DCJ scenario for a weakly partitive family of common intervals. We first consider both the cases of weakly partitive and bi-partitive families, as bi-partitive families are the natural combinatorial framework for circular chromosomes with equal gene content.

Property 4.
1. For any family $\mathcal{F}$ of common intervals of two genomes Π and Γ such that no interval of $\mathcal{F}$ is equal to the gene content of both a circular chromosome of Π and a circular chromosome of Γ, a DCJ scenario transforming Π into Γ is $\mathcal{F}$-perfect if and only if it is $\mathcal{F}^p$-perfect.
2. For any family $\mathcal{F}$ of common intervals of two unichromosomal circular genomes Π and Γ on the same set of genes $\mathcal{G}$, if $\cup_{I \in \mathcal{F}} I = \mathcal{G}$, a DCJ scenario transforming Π into Γ is $\mathcal{F}$-perfect if and only if it is $\mathcal{F}^b$-perfect.

Proof.
1. Obviously if a scenario is $\mathcal{F}^p$-perfect, then it is $\mathcal{F}$-perfect, as $\mathcal{F} \subseteq \mathcal{F}^p$.
   Let S be a $\mathcal{F}$-perfect scenario transforming Π into Γ. Let I and J be two overlapping intervals of $\mathcal{F}$ belonging to S. At some stage of the scenario, I and/or J may contain circular chromosomes. Let i_1 and i_2 be the two border adjacencies of I if I does not only contain circular chromosomes, and j_1 and j_2 be the two border adjacencies of J, if J does not only contain circular chromosomes. Without loss of generality, if $I \cap J$ does not only contain circular chromosomes, let $j_1 \in I$ and $i_2 \in J$. Thus, for every $K \in \{I \cup J, I \cap J, I \setminus J, J \setminus I\}$, either K is composed of one or more circular chromosomes and so has no border adjacency or K has two border adjacencies:

   —i_1 and j_2 if $K = I \cup J$;
   —j_1 and i_2 if $K = I \cap J$;
   —i_1 and j_1 if $K = I \backslash J$;
   —i_2 and j_2 if $K = J \backslash I$.

   For every $K \in \{I \cup J, I \cap J, I \setminus J, J \setminus I\}$, a DCJ breaks K if and only if K has two border adjacencies and the DCJ cuts an inside adjacency of K and an outside adjacency of K (see Def. 6 and Prop. 3). This DCJ creates two new border adjacencies of K, these new border adjacencies are also new border adjacencies of I and/or J (depending on which interval K is and if $K = I \cup J$ depending on where is the cut inside adjacency). Thus if a DCJ breaks $K \in \{I \cup J, I \cap J, I \setminus J, J \setminus I\}$ then it breaks I and/or J. Therefore a DCJ that does not break I nor J does not break $K \in \{I \cup J, I \cap J, I \setminus J, J \setminus I\}$. So S is $\mathcal{F}^p$-perfect.
2. Obviously if a scenario is $\mathcal{F}^b$-perfect, then it is $\mathcal{F}$-perfect, as $\mathcal{F} \subseteq \mathcal{F}^b$.
   Let S be a $\mathcal{F}$-perfect scenario transforming Π into Γ. Let I and J be two bi-overlapping subsets of $\mathcal{F}$ belonging to S. By definition, if I and J bi-overlap, they overlap. Thus, as demonstrated for the previous case, if S does not break I nor J then S does not break any K for $K \in \{I \cup J,\ I \cap J, I \setminus J, J \setminus I\}$. It remains to be shown that if S does not break I, S does not break $\bar{I}^E$. Note that I has no border adjacency if and only if $\bar{I}^E$ has no border adjacency and that I has two border adjacencies if and only if $\bar{I}^E$ has two border adjacencies, in this latter case I and $\bar{I}^E$ have the same border adjacencies. A DCJ D breaks $\bar{I}^E$ if and only if $\bar{I}^E$ has two border adjacencies and D cuts an inside adjacency of $\bar{I}^E$, say p, and an outside adjacency of $\bar{I}^E$, say q. As I and $\bar{I}^E$ are complementary subsets in E, p is outside I and q is inside I, so D breaks I. Therefore a DCJ that does not break I does not break $\bar{I}^E$. So S is $\mathcal{F}^b$-perfect. ■

So we may assume that families of common intervals we consider are either weakly partitive or weakly bipartitive (in the case where both genomes contain a circular chromosome with the same gene content which is covered by non-trivial common intervals to be preserved).

Assume now that the two genomes Π and Γ have a circular chromosome with the same gene content, say X, and that the family $\mathcal{F}$ of common intervals to be preserved covers X. Without any loss of generality, we consider that Π and Γ are the circular chromosomes, and $\mathcal{F}$ is the weakly bipartitive closure of the set of common intervals between Π and Γ that must be preserved. We show now how we can handle these circular chromosomes using a weakly partitive family.

Conceptually, the general idea, a classical trick in the theory of partitive families, consists in rooting the (unrooted) decomposition tree of $\mathcal{F}$ at an arbitrary leaf (corresponding to a gene x of X), which results in the decomposition tree of a weakly partitive family. So let x be an arbitrary gene of X. We define $\mathcal{F}_{\bar{x}}$ as the subfamily of $\mathcal{F}$ containing all common intervals I that do not contains the gene x. Let $\Pi_{lin}^{\bar{x}}$ (resp. $\Gamma_{lin}^{\bar{x}}$) be the linear chromosome obtained from Π (resp. Γ) by cutting the gene x (Fig. 5).

The following lemma is the main result of this section, as it implies that, given any pair of genomes Π and Γ and family $\mathcal{F}$ of common intervals, computing an $\mathcal{F}$-perfect scenario requires only to consider a weakly partitive family.

Lemma 2. *A DCJ scenario transforming Π into Γ is $\mathcal{F}$-perfect if and only if it is a $\mathcal{F}_{\bar{x}}$-perfect scenario transforming $\Pi_{lin}^{\bar{x}}$ into $\Gamma_{lin}^{\bar{x}}$.*

Proof. One direction is trivial: as $\mathcal{F}_{\bar{x}}$ is a subset of $\mathcal{F}$, any $\mathcal{F}$-perfect scenario is also $\mathcal{F}_{\bar{x}}$-perfect. Conversely, suppose that a scenario transforming $\Pi_{lin}^{\bar{x}}$ into $\Gamma_{lin}^{\bar{x}}$ is not a $\mathcal{F}$-perfect scenario transforming Π into Γ. This means that it breaks an interval I of $\mathcal{F}$. If this interval does not contain x, then it belongs to $\mathcal{F}_{\bar{x}}$ and the scenario is not $\mathcal{F}_{\bar{x}}$-perfect. Now suppose I contains x. As the scenario is not perfect, there is a DCJ that cuts inside and outside I. The adjacency inside I is outside $\bar{I}$ and the adjacency outside I is inside $\bar{I}$. So the DCJ breaks $\bar{I}$ as well which, by definition of $\mathcal{F}$, belongs to $\mathcal{F}$. As $\bar{I}$ does not contain x it also belongs to $\mathcal{F}_{\bar{x}}$. Thus the scenario is not $\mathcal{F}_{\bar{x}}$-perfect. ■

4.4. The sorting direction of an interval

Lemma 1 states that any scenario preserving an interval I can start by sorting I. Here, we state that there are typically three different ways of sorting an interval I. First, once sorted, I can have either 0 border adjacencies (it forms a circular chromosome) or 2. In the latter case, the two gene extremities of I that belong to a border adjacency in Γ can form adjacencies with the two gene extremities outside of I that frame I in Π in two ways. These three situations define the three possible *orientations*, or *sorting directions* of I. The fundamental problem in perfect rearrangement problems is then, for a given interval, to decide to which direction it has to be sorted, or equivalently, to determine the exact set of adjacencies involving genes in I that have to be constructed from Π using DCJs.

In the reversal model, where only two sorting directions are allowed as circular chromosomes are not considered, this decision was at the heart of the hardness of perfect sorting (Figeac and Varré, 2004; Bérard et al., 2007, 2008b), as for some intervals it was impossible to decide which orientation was the most parsimonious. Here we extend such results to the DCJ model and show that the ability to sort an interval into a circular chromosome has for consequence that one of the three available choices is always more parsimonious. This point is central in the fact that some hard problems in the reversal model become tractable when using DCJs instead of reversals.

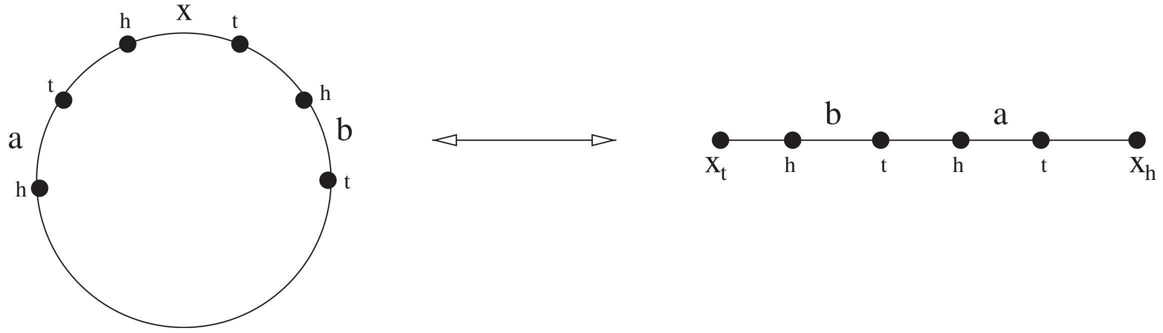

FIG. 5. Transforming a circular chromosome into a linear chromosome by cutting a gene.

Let I be a common interval of two genomes Π and Γ. If I has border adjacencies in Π let x_Π and y_Π be the extremities of genes that *are not in I* and belong to the two border adjacencies of I in Π (they may be T symbols). If I has no border adjacencies in Π, let $x_\Pi = y_\Pi = T$. If I has border adjacencies in Γ, let m_Γ and M_Γ be the extremities of genes that *are in I* and belong to the two border adjacencies of I in Γ. Figure 6 illustrates these notations.

We wish to sort the interval I with respect to Γ, that is we want to obtain a genome Π' from Π which contains every adjacency inside I in Γ. The following property is immediate.

Property 5. If Π has been sorted by DCJs that cut only inside an interval I, resulting in a genome Π', then only one of the three following situations can occur:

1. $x_\Pi m_\Gamma$ and $M_\Gamma y_\Pi$ are adjacencies in Π',
2. $x_\Pi M_\Gamma$ and $m_\Gamma y_\Pi$ are adjacencies in Π',
3. $m_\Gamma M_\Gamma$ and $x_\Pi y_\Pi$ are adjacencies in Π'.

Definition 7. Given I, x_Π, m_Γ , M_Γ and y_Π, we say that I is sorted *positively* if Property 5.1 holds, *negatively* if Property 5.2 holds or *neutrally* if Property 5.3 holds. The way I is sorted is called its *sorting direction or orientation* in Π'.

We denote by $\Pi|I^+$ (resp. $\Pi|I^-$ and $\Pi|I^N$) the genome obtained from Π, in which I is sorted positively (resp. negatively and neutrally) with respect to Γ. Note that $\Pi|I^N$ contains a circular chromosome. It is clear that $d(\Pi|I^-, \Pi|I^+) = d(\Pi|I^-, \Pi|I^N) = d(\Pi|I^+, \Pi|I^N) = 1$.

Remark 1. The sorting direction of a common interval I, when it is positive or negative, depends on the choice of which gene extremity that is not in I but belongs to a border adjacency of I in Π is called x_Π (resp. y_Π and which gene extremity that is not in I but belongs to a border adjacency of I in Γ is called m_Γ (resp. M_Γ). However, we stress that these two choices have no influence, as the sorting direction is unambiguously defined by the border adjacencies of I in the genome resulting from sorting I. For example, let $I = \{2, 3\}, \Pi = \{T1_t, 1_h2_t, 2_h3_t, 3_h4_t, 4_h5_t, 5_hT\}$ and $\Gamma = \{T5_t, 5_h3_t, 3_h2_t, 2_h1_t, 1_h4_t, 4_hT\}$. Sorting I with respect to Π involves creating the adjacency 3_h2_t. Assume that I is sorted and the two border adjacencies in the resulting genome are 1_h2_h and 3_t4_t (this resulting genome is then $\{T1_t, 1_h2_h, 2_t3_h, 3_t4_t, 4_h5_t, 5_hT\}$). If $x_\Pi = 1_h$, $y_\Pi = 4_t$, $m_\Gamma = 2_h$ and $M_\Gamma = 3_t$, then I has been sorted positively. However, if the choice for x_Π and y_Π had been different ($x_\Pi = 4_t$, $y_\Pi = 1_h$), then the sorting direction of I is negative. This illustrates the fact that the sorting direction depends on a decision that is specific to I (the choice of which gene extremities are labeled x_Π, y_Π, m_Γ and M_Γ), and then, without loss of generality, we take that decision arbitrarily for every strong interval.

Lemma 3. *Let Π and Γ be two genomes and let I be a set of genes that has two border adjacencies in Γ and at most two in Π. Then one and only one of the three following possibilities holds:*

1. $d(\Pi, \Pi|I^+) = d(\Pi, \Pi|I^-) - 1 = d(\Pi, \Pi|I^N) - 1$ *and* $d(\Pi, \Gamma) = d(\Pi, \Pi|I^+) + d(\Pi|I^+, \Gamma)$;
2. $d(\Pi, \Pi|I^-) = d(\Pi, \Pi|I^+) - 1 = d(\Pi, \Pi|I^N) - 1$ *and* $d(\Pi, \Gamma) = d(\Pi, \Pi|I^-) + d(\Pi|I^-, \Gamma)$;
3. $d(\Pi, \Pi|I^N) = d(\Pi, \Pi|I^+) - 1 = d(\Pi, \Pi|I^-) - 1$ *and* $d(\Pi, \Gamma) = d(\Pi, \Pi|I^N) + d(\Pi|I^N, \Gamma)$.

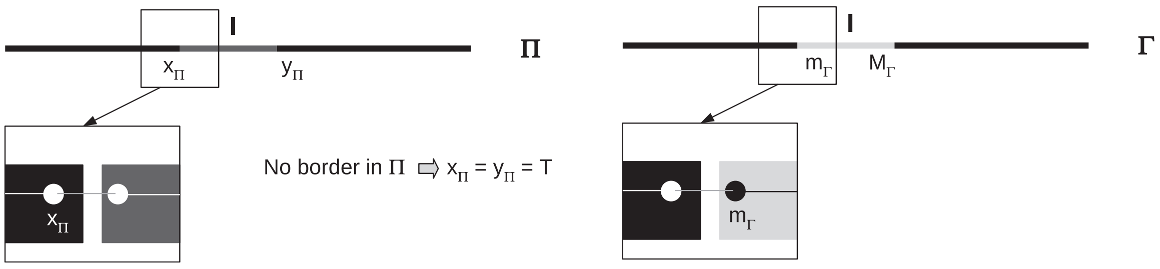

FIG. 6. Notations for the definition of sorting directions.

Proof. Let G^+, G^- and G^N be the respective breakpoint graphs of genomes Π and, respectively, $\Pi|I^+$, $\Pi|I^-$, and $\Pi|I^N$. We must consider three cases: ■

1. *Neither x_Π nor y_Π are T symbols* (in other words x_Π and y_Π are vertices in the breakpoint graphs). The three breakpoint graphs are identical for every edges, except that G^+ contains the edges $x_\Pi m_\Gamma$ and $M_\Gamma y_\Pi$, whereas G^- contains the edges $x_\Pi M_\Gamma$ and $m_\Gamma y_\Pi$, and G^N contains the edges m_Γ M_Γ and $x_\Pi y_\Pi$. Each of these four vertices, in either of these three graphs, has degree 2, so deleting from one graph the two edges joining the vertices x_Π, M_Γ, m_Γ, y_Π yields two paths coupling these four vertices two by two, and this matching is independent of which graph is considered. According to which couples are joined by a path, one of the three graphs G^+, G^- and G^N has one more cycle than the two others. Suppose for example that x_Π and m_Γ are joined by a path, and M_Γ and y_Π are joined by a path, then G^+ has two cycles involving those vertices, whereas G^- and G^N have only one. As the three graphs are identical everywhere else, the formula of Theorem 1 yields $d(\Pi,\Pi|I^+) = d(\Pi,\Pi|I^-) - 1 = d(\Pi,\Pi|I^N) - 1$. The two other cases, where x_Π - M_Γ and m_Γ - y_Π are joined by paths, or x_Π - y_Π and m_Γ - M_Γ are joined by paths, are symmetric.
2. *x_Π or y_Π is a T symbol (but not both)* (in other words, I is a linear chromosome extremity and then I has at least one border adjacency that is a telomeric adjacency). Suppose that y_Π is a T symbol (the case where x_Π is a T symbol is symmetric). Let M_Π be the gene extremity of I involved in its telomeric adjacency (there is a special case if $M_\Pi = M_\Gamma$ or if $M_\Pi = m_\Gamma$ explained at the end of this item). In the three breakpoint graphs G^+, G^- and G^N:

   — Some vertices have degree 0, the ones representing the gene extremities of Π involved in telomeric adjacencies, except M_Π;
   — Two vertices have degree 1: M_Π in the three graphs and

   - M_Γ in G^+,
   - m_Γ in G^-,
   - x_Π in G^N;

   — Every other vertices have degree 2.

   The three graphs differ only by one edge, say e, between the vertices M_Γ , m_Γ and x_Π:

   — $e = x_\Pi - m_\Gamma$ in G^+,
   — $e = x_\Pi - M_\Gamma$ in G^-,
   — $e = m_\Gamma - M_\Gamma$ in G^N;

   If we delete the edge e from the graphs, we obtain four vertices of degree 1 which define two paths C_1 and C_2. In two out of the three graphs, the edge e joins the paths C_1 and C_2 to form a longer path C, and in the third graph e closes one of these two paths, then producing a cycle, and does not modify the other path, say C_i with $i \in \{1, 2\}$. In this last graph, there is one more cycle than in the other two but we must verify that the paths C and C_i have the same parity to use the formula of Theorem 1. Remember that in these breakpoint graphs there are two classes of edges: the Π-edges and the $\Pi|I^x$-edges, with $x \in \{+, -, N\}$, and that in every path or cycle, consecutive edges never belong to the same class. So every cycle has even length and a path beginning and finishing by edges of the same class (resp. from different classes) has odd (resp. even) length. Suppose that I contains k genes, in the three graphs the vertex set can be split into two sets A and B. A contains the $2k$ gene extremities of I plus x_Π and B contains all the vertices representing gene extremities not belonging to I except x_Π. All the vertices of B have either a degree 0 or are involved in a cycle of length 2. We are interested in the edges incident to the vertices of A and more precisely to the ones belonging to C_1 and C_2. All $2k + 1$ vertices of A have degree 2 except two of them, so the sum of the degrees of these vertices is $2^* (2k - 1) + 2 = 4k$ and the number of edges incident to these vertices is then $2k$. Among these edges, some of them can be involved in cycles, but as a cycle has even length, the number of edges belonging to C_1 and C_2 plus e is even. Without loss of generality, say that C_1 is the even-length path and C_2 the odd-length path. Looking more precisely at the four vertices at the extremities of C_1 and C_2:

   — M_Π is incident to a $\Pi|I^x$-edge;
   — The other degree one vertex before the removal of e is incident to a Π-edge;
   — e is a $\Pi|I^x$-edge, so the two vertices incident to e are also incident to a Π-edge.

   Therefore, M_Π is incident to a $\Pi|I^x$-edge, and M_Γ , m_Γ and x_Π are incident to a Π-edge in either of the three graphs after the removal of e. So, the path with extremity M_Π has even length (it is C_1), whereas the other, C_2, has odd length. The edge e either joins C_1 and C_2 or closes the path C_2 (it cannot close C_1 as C_1 begins with a $\Pi|I^x$-edge). We thus check that C and C_i have the same parity. Suppose that G^+ is the graph with the cycle more than the other two, the formula of Theorem 1 yields $d(\Pi, \Pi|I^+) = d(\Pi, \Pi|I^-) - 1 = d(\Pi, \Pi|I^N) - 1$. The two other cases are symmetric.

For the special case where $M_{\Pi} = M_{\Gamma}$ (resp. $M_{\Pi} = m_{\Gamma}$), G^{+} (resp. G^{-}) has a degree 0 vertex instead of two degree 1 vertices (the two other graphs keep the same structure). This vertex is a path of length 0, so even, and this is in this graph that e forms a cycle with C_2.

3. *Both x_{Π} and y_{Π} are T symbols* (in other words I has no border adjacency in Π). In this case, the three graphs are identical except that G^{N} has an additional edge joining $M_{\ \Gamma}$ to m_{Γ}. This edge creates a cycle where in the other two graphs there is a odd-length path. Therefore, the formula of Theorem 1 yields $d(\Pi, \Pi|I^{N}) = d(\Pi, \Pi|I^{+}) - 1 = d(\Pi, \Pi|I^{-}) - 1$.

Let S be a scenario that sorts I positively in $d(\Pi,\Pi|I^{+})$ operations. We prove now that all operations in S create an adjacency that is in Γ but not in Π. By Theorem 1, this implies that all operations in S are optimal, and then $d(\Pi, \Gamma) = d(\Pi, \Pi|I^{+}) + d(\Pi|I^{+}, \Gamma)$. Suppose there is an operation ρ in S that does not create such an adjacency of Γ, and that ρ cuts adjacencies ab and cd and joins ac and bd. Let ρ' be the DCJ following ρ in S and assume that ρ' creates an adjacency of Γ. We can replace ρ and ρ' by two DCJs that results in the same genome and such that the second one does not create an adjacency in Γ: if ρ' cuts neither ac nor bd, then we can simply swap ρ and ρ', while, if ρ' cuts ac and xy and joins ax and cy, then we can replace ρ and ρ' by two DCJs, one that first cuts ab and xy and joins ax and by, then a second one that cuts by and cd and joins cy and bd. So we can assume without loss of generality that there is a point in S where all adjacencies of Γ inside I have been created, and all other operations come after. But here I is sorted, so no other operation is needed. This proves that each DCJ operation is optimal. ■

To summarize this section, to compute a prefect DCJ scenario:

— only the strong intervals of a weakly partitive family need to be preserved (Lemma 2),
— sorting an interval I of such a family only requires to create the adjacencies that are inside this interval in Γ, which can be done independently of sorting the strong intervals that are not included in I (Lemma 1),
— for each strong interval I, the fundamental question is to decide its sorting direction and there is only one parsimonious choice for this direction (Lemma 3),
— and finally, once this choice has been done, the classical greedy approach for sorting by DCJ can be applied to sort I (proof of Lemma 3).

These properties are at the heart of the results we describe in the next two sections.

5. $\mathcal{F}$ WEAKLY SEPARABLE: A HARDNESS RESULT

In general, the problem of $\mathcal{F}$-perfect DCJ rearrangement is hard, and even with weakly separable families of common intervals. This is the DCJ version of the NP-hardness for perfect sorting by reversals (Figeac and Varré, 2004), but it contrasts with the fact that perfect sorting by reversals with a weakly separable family of common intervals can be done in linear time (Bérard et al., 2007) (the decomposition tree of a weakly separable family of common intervals has no pairs of adjacent prime nodes).

Theorem 2. *The $\mathcal{F}$-perfect DCJ problem is NP-hard, even if $\mathcal{F}$ is weakly separable.*

Proof. We use a reduction from the NP-complete Balanced Graph Decomposition (BGD) problem, which was used in Caprara (2003) to prove the hardness of some reversal problems.

A graph G is *balanced bicolored* if is has only vertices of degree 2 and 4, and its edges are colored in red and blue such that, for each vertex v, there are as many blue edges as red edges incident to v. A cycle of G is said to be an *alternating cycle* if two consecutive edges do not have the same color. Given a balanced bicolored graph and an integer k, the BGD problem consists in deciding whether its edge set can be partitioned into k alternating cycles.

We first describe how to transform, in polynomial time, an instance of the BGD problem, i.e., a balanced bicolored graph G, into the breakpoint graph $BP(\Pi,\Gamma)$ of two genomes. We then identify a set $\mathcal{F}$ of common intervals of Π and Γ for which the size of an $\mathcal{F}$-perfect DCJ scenario depends on the number of vertices of G and the number of alternating cycles.

To build $BP(\Pi, \Gamma)$ from a balanced bicolored graph G, each degree 2 (resp. 4) vertex of G is transformed into two (resp. 8) vertices of $BP(\Pi, \Gamma)$ which can be seen as the extremities of genes of Π and Γ (Fig. 7).

— For each degree 2 vertex u of G, we define two gene extremities u_h and u_t such that u_hu_t is an adjacency of Π. One of the two edges of G incident to u becomes incident to u_h and the other to u_t. These two edges will correspond to Γ adjacencies.

— For each degree 4 vertex v of G, we define 8 genes extremities, say v_h^1, v_t^1, v_h^2, v_t^2, v_h^3, v_t^3, v_h^4, v_t^4. The blue edges of G incident to v are transformed into Γ adjacencies incident to v_h^3 and v_h^4, while the red edges incident to v become Γ adjacencies incident to v_t^1 and v_t^2. Moreover we create two new Γ adjacencies, namely $v_h^2v_t^3$ and $v_h^1v_t^2$, and four new Π adjacencies $v_h^1v_h^4$, $v_t^3v_t^4$, $v_t^1v_h^2$ and $v_h^3v_t^2$.

By construction, the Γ and Π adjacencies form two perfect matchings of G', and thus define two genomes without telomeres. The union of these two sets of adjacencies defines the breakpoint graph $BP(\Pi, \Gamma)$.

Also by construction, for any degree 4 vertex of G, the subsets of genes $\{v^1, v^2\}$, $\{v^2, v^3\}$ and $\{v^1, v^2, v^3\}$ are common intervals of the genomes Π and Γ. Let us denote by $\mathcal{F}$ this set of common intervals. Formally:

$$\mathcal{F} = \{\{v^1, v^2\}, \{v^2, v^3\}, \{v^1, v^2, v^3\} \mid v \text{ is a degree 4 vertex of } G\}$$

Only the intervals of type $\{v^1,v^2,v^3\}$ are strong intervals, and they are the union of overlapping intervals, so this family is weakly separable. Then the genomes Π and Γ together with $\mathcal{F}$ form an instance of the $\mathcal{F}$-perfect DCJ problem, with a weakly separable family of common intervals. It remains to relate the number of cycles in G to the perfect DCJ distance of the reduced instance, which is done in Lemma 4 below and concludes the proof of Theorem 2.

Lemma 4. *Let G be a balanced bicolored graph with n_2 degree 2 vertices and n_4 degree 4 vertices and let Γ, Π be the genomes resulting from the reduction described above. There is an $\mathcal{F}$-perfect DCJ scenario of length k for Π, Γ if and only if there are $n_2 + 5n_4 - k$ alternating cycles in G.*

Proof. ($\Rightarrow$) Suppose there is an $\mathcal{F}$-perfect scenario of length k. By Lemma 1 it can be assumed that this scenario sorts a linear strong interval $\{v^1,v^2,v^3\}$ first. This interval may be sorted positively, negatively or neutrally. The number of operations to sort it positively or neutrally is 3, while the number of operations to sort it negatively is 2, but it is impossible to sort it negatively in less than 4 operations while preserving $\{v^1,v^2\}$ and $\{v^2,v^3\}$. So we may assume that the scenario sorts it positively or neutrally. Consider the step of the scenario where all such intervals are sorted either positively or neutrally. The breakpoint graph is then a set of cycles, and the Γ-edges of these cycles form alternating red/blue edge cycles in G. So $k = 3n_4 + d$, where d is the number of remaining operations to do. And we have $d = n - c$ by Theorem 1 and because the genomes have no telomeres, with $n = n_2 + 2n_4$. So $k = n_2 + 5n_4 - c$, and there are $n_2 + 5n_4 - k$ alternating cycles in G.

($\Leftarrow$) Suppose there are $n_2 + 5n_4 - k$ alternating cycles in G. For each degree 4 vertex, it is possible to find two couples of blue/red edges, according to the edges that are incident in the cycles. Now construct a

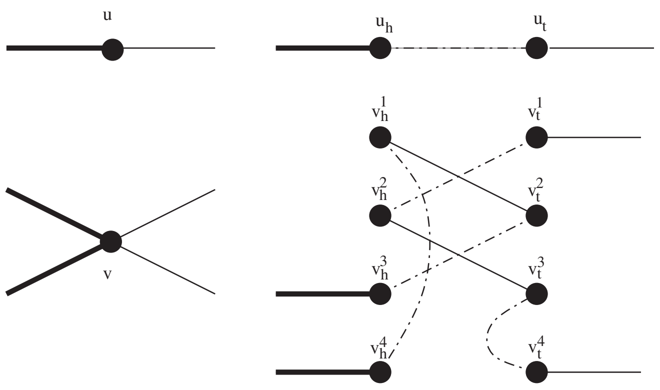

FIG. 7. On the left, u and v are vertices of the balanced bicolored graph G (blue edges are thick, while the reds are thin). On the right hand-side of the figure it s depicted how vertices u and v transform in the breakpoint graph $BP(\Gamma,\Pi)$. The dotted edges are Π adjacencies.

scenario that sorts every linear strong interval $\{v^1, v^2, v^3\}$ either neutrally or positively to match either $v4_h v4_t$ and $v_t^1 v_h^3$, or $v4_t v_h^3$ and $v_t^1 v4_h$, according to the incident edges that need to be paired. This scenario needs $3n_4$ DCJs. And the remaining DCJs are $2n_4 + n_2 - (n_2 + 5n_4 - k) = k - 3n_4$, so the scenario needs exactly k DCJs. ■

6. AN EXACT ALGORITHM

We now present an exact algorithm to compute a shortest DCJ scenario between two genomes Π and Γ that preserves a weakly partitive family $\mathcal{F}$ of common intervals. We first describe the general structure of this algorithm, then the details of each major step. This algorithm can be decomposed into three main steps:

Algorithm 1 Computing an $\mathcal{F}$-Perfect DCJ scenario between genomes Π and Γ

Step 1. *For each maximal common interval I_M, compute the decomposition tree of the strong intervals that are included in I_M.*
Step 2. *For each of these decomposition trees, during a post-order traversal of this tree, sort each node I assuming its children have been sorted using Algorithm 2.*
Step 3. *Finally, after all maximal common intervals have been sorted, compute a parsimonious series of DCJs that creates all the remaining adjacencies of Γ, that are not inside any maximal common interval.*

We first describe steps 1 and 3, that use known techniques and algorithms, before proceeding to step 2, that is the core of our new method.

Step 1: Defining a set of independant subproblems. In this first step, we break down the shortest perfect DCJ-scenario problem into a set of subproblems defined in terms of maximal common intervals. To do so, we first compute the set of all maximal common intervals of Π and Γ, by computing the common connected components of G_Π and G_Γ. Indeed, an interval is defined as a connected subgraph, and common connected components are exactly the maximal subgraphs that are connected in both graphs. Efficient techniques for this step can be found in Habib et al. (2004) for example.

By definition of common intervals, every interval $\mathcal{F}$ belongs to such a maximal common interval and we denote by $\mathcal{F}_{I_M}$ the subset of the intervals of $\mathcal{F}$ that are included in a maximal common interval I_M. If I_M is the gene content of a circular chromosome of Π and one in Γ, then, we choose an arbitrary gene inside I_M and linearize the circular chromosome. Otherwise, following Property 4.1, we compute the rooted decomposition tree of $\mathcal{F}^p_{I_M}$. Decomposition trees can be computed efficiently (see proof of Theorem 3). We denote by $\mathcal{T}(\mathcal{F}, \Pi, \Gamma)$ the forest of decomposition trees.

By definition of maximal common intervals, and by Lemma 1, we can now process each problem associated to a maximal common interval I_M independently of the other ones.

Step 3: Sorting adjacencies outside maximal common intervals. After Step 2 has been completed, all adjacencies of Γ that were inside maximal common intervals of Π and Γ have been resolved by a sequence of DCJ operations that preserved all common intervals of $\mathcal{F}$. Therefore, to complete the sorting of Π into Γ, we only need to create the adjacencies of Γ that are not in any maximal common interval. This may be done by applying any algorithm computing DCJ scenarios, the fastest and simplest being the one of Bergeron et al. (2006). This requires linear time.

Step 2: Sorting inside a maximal common interval. We now consider a maximal common interval I_M and a rooted decomposition tree of a weakly partitive family of common intervals of Π and Γ included in I_M. The general principle of this step is to sort these common intervals in a bottom-up way, during a post-order traversal of the decomposition tree. For a given node I of the decomposition tree, following Lemma 1 we can indeed assume that its children have been sorted (i.e., all adjacencies of Γ inside its children have been created). Moreover, from this assumption, we can assume that we do not need to consider DCJs that cut inside any child J of I. This yields Algorithm 2.

Algorithm 2 $\mathcal{F}$-Perfect sorting of a maximal common interval I_M of genomes Π and Γ, given the strong interval tree T of a family $\mathcal{F}_{I_M}$ of common intervals in I_M

```
LET $\Pi' = \Pi$
FOR each interval $I \subseteq I_M$ of $\Pi$ and $\Gamma$ in a post-order traversal of $T$
  {Note: all children of $I$ are sorted}
  Case 1. $I$ is prime
    Compute $k = \min(d(\Pi', \Pi' \setminus I^+), d(\Pi', \Pi' \setminus I^-), d(\Pi', \Pi' \setminus I^N))$
    Sort I with k DCJs inside I and outside its children using Bergeron et al. (2006)
  Case 2. $I$ is linear
    Give one of the possible two orders to the children of I
    For each child of I, let $x_\Pi$ and $m_\Gamma$ correspond to the adjacencies linking it to the
previous one in the chosen order, and $y_\Pi$, $M_\Gamma$ correspond to the adjacencies
linking it to the following one in the chosen order
      Case 2.1 At least one child of I is sorted positively
        Sort I by flipping to positive all non positive children
      Case 2.2 All children of I are sorted neutrally
        Sort I neutrally by joining the consecutive children
      Case 2.3 One child of I is sorted negatively, and all others neutrally
        Sort I by flipping to negative all neutral children
      Case 2.4 Two children are sorted negatively, and all others neutrally
        Create a circular chromosome from I by cutting its two border adjacencies.
        Join all the neutral children to the circular chromosome
      Case 2.5 Three children are sorted negatively, and all others neutrally
        Try two possibilities:
          1. Sort I positively by flipping all children to positive,
          2. Independently, Sort I neutrally by performing three operations
        Note: this part yields an exponential running time.
      Case 2.6 At least four children are negative, and all others are neutral
        Sort I positively by flipping all children to positive
  LET $\Pi'$ denote the resulting genome.
```

Lemma 5. *Given two genomes Π and Γ, a family $\mathcal{F}$ of common intervals and a maximal element I_M of $\mathcal{F}$, Algorithm 2 computes a DCJ scenario that sorts I_M with respect to Γ and preserves all the intervals of $\mathcal{F}$ contained in I_M. Moreover, no scenario can sort I_M while preserving $\mathcal{F}$ in fewer operations and no scenario achieves the same number of operations and sorts I_M in another direction.*

Proof. Every maximal common interval I_M is sorted using Algorithm 2, which sorts every child I of I_M in the decomposition tree rooted by I_M. For every child $I \subseteq I_M$, I belongs to the family $\mathcal{F}$ and the algorithm uses only DCJ operations inside I and outside its children to sort it, so it preserves every interval $I \in \mathcal{F}$.

Now we may prove that no scenario sorts any strong interval I (so it is true also for I_M) in fewer operations.

We analyze each case separately.

Case 1. The interval I is prime.

The interval I has two border adjacencies in Γ, and we suppose the algorithm sorts I positively (the other cases are symmetric). By Lemma 3, it uses $d(\Pi,\Pi \setminus I^+)$ operations, and no scenario may sort I in fewer operations. Again by Lemma 3, $d(\Pi, \Pi \setminus I^+) = d(\Pi, \Pi \setminus I^-) - 1 = d(\Pi, \Pi \setminus I^N) - 1$ so it is not possible to sort I in another direction in the same number of operations.

Case 2. The node corresponding to I is linear. We subdivide this case following the successive conditions in the algorithm, illustrating each case with a figure.

— **Case 2.1** (Fig. 8) If there is at least one child of I that is sorted positively, then the breakpoint graph of Π and $\Pi \setminus I^+$ has at least two components, and the edges bordering the positive elements do not belong to the same cycle. There is one less cycle in $BP(\Pi \setminus I^-,\Gamma)$ and $BP(\Pi \setminus I^N,\Gamma)$, so $d(\Pi, \Pi \setminus I^+) = d(\Pi, \Pi \setminus I^-) - 1 = d(\Pi, \Pi \setminus I^N) - 1$ and flipping all non positive elements to positive is perfect, and takes $d(\Pi,\Pi \setminus I^+)$ operations.

— **Case 2.2** (Fig. 9) If all k children of I are sorted neutrally, then $BP(\Pi \setminus I^N,\Gamma)$ has one cycle whose edges are inside I but not inside its children, while $BP(\Pi \setminus I^+,\Gamma)$ and $BP(\Pi \setminus I^-,\Gamma)$ both have one odd path, so by Theorem 1 $d(\Pi, \Pi \setminus I^N) = d(\Pi, \Pi \setminus I^+) - 1 = d(\Pi, \Pi \setminus I^-) - 1$, and $k-1$ perfect DCJ operations sort I neutrally. All of them construct an adjacency of Γ that is not an adjacency in Π, so $d(\Pi,\Pi \setminus I^N) = k-1$.

— **Case 2.3** (Fig. 10) If one child of I is sorted negatively, while all $k-1$ others are sorted neutrally, then the breakpoint graph of Π and $\Pi \setminus I^-$ has at least two components, and the edges bordering the negative element do not belong to the same cycle. There is one less cycle in $BP(\Pi \setminus I^+,\Gamma)$ and $BP(\Pi \setminus I^N,\Gamma)$. So $d(\Pi, \Pi \setminus I^-) = d(\Pi, \Pi \setminus I^+) - 1 = d(\Pi, \Pi \setminus I^N) - 1 = k-1$ and flipping all neutral elements to negative by creating an adjacency at each time is perfect, and takes $d(\Pi,\Pi \setminus I^-)$ operations.

— **Case 2.4** (Fig. 11) If two children X and Y are sorted negatively, while all $k-2$ others are sorted neutrally, then let ax and yb be the two border edges of I if they exist. Then there is a path in $BP(\Pi \setminus I,\Gamma)$ joining x and y and not containing the middle edge, which is adjacency between X and Y. So $BP(\Pi \setminus I^N,\Gamma)$ has one more cycle than $BP(\Pi \setminus I^+,\Gamma)$ and $BP(\Pi \setminus I^-,\Gamma)$, and there is a perfect scenario sorting I in $k-1$ DCJs. In the case I is the gene content of a circular chromosome, that is, I has no border adjacencies, then there is no orientation choice to make, and we still have $d(\Pi, \Pi \setminus I^N) = d(\Pi, \Pi \setminus I^+) - 1 = d(\Pi, \Pi \setminus I^-) - 1 = k-2$, so the only optimal way to sort I is to join every small circular chromosome into the larger one containing two children at the beginning.

— **Case 2.5** (Fig. 12) Three children are sorted negatively, while all the others are sorted neutrally. If I has border adjacencies, then we meet here the exponential part of the algorithm. There are two ways to sort I in three operations plus the number of neutrally sorted children: one positively, and one neutrally. It is the pattern which proves the NP-completeness of the result. As the algorithm tries both possibilities, it is clearly optimal (note it is possible to sort negatively in two operations plus the number of neutrally sorted children, but the first operation is not perfect).

— **Case 2.6** (Fig. 13) At least four children are sorted negatively, while all the others are sorted neutrally.

Claim 1. Every DCJ preserving the common intervals that cuts two adjacencies belonging to the linear component of I cuts the two border adjacencies of one single element or the border adjacencies of a whole linear component of I.

Proof. Suppose it cuts at two adjacencies which are not the two border adjacencies of a single element nor the border adjacencies of the whole component. Then there are at least two children X and Y of I between the two adjacencies and one child Z not between the two adjacencies (suppose without loss of generality that Z is before X and X is before Y). Then in the resulting genome, the minimal interval containing X and Z has at least three border adjacencies: at both sides of Z, and after X. So one common interval is not preserved. □

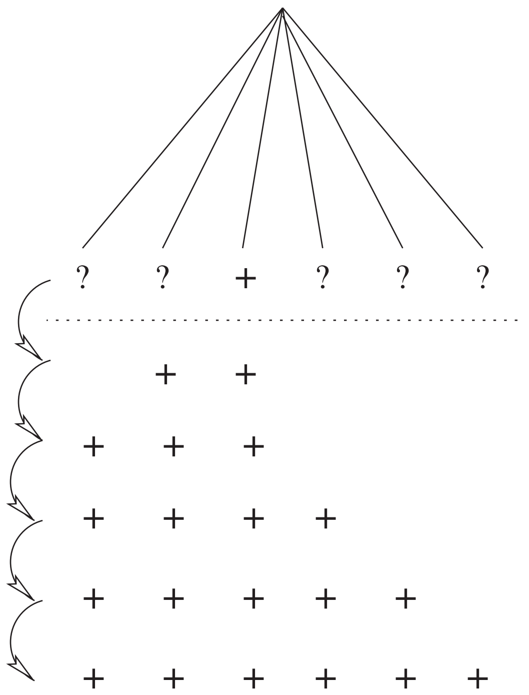

FIG. 8. Case 2.1: there is at least one child of I that is sorted positively.

Claim 2. In the case of a linear node, in any optimal perfect scenario, at least one negative child of I is sorted positively.

Proof. Indeed, let J be a child of I which is sorted negatively and has one predecessor (say $J-1$) and one successor (say $J+1$) in the order given to the children of I. As there are at least four negative children, there are at least two such nodes J. The two border adjacencies of J have to be cut because they are in Π but not in Γ. By Claim 1, this can be achieved by cutting the two border adjacencies of J, $J+1$ or $J-1$. If none of these are sorted positively during a scenario, the only other possibility is to sort them neutrally. Suppose one is sorted neutrally. Then there remain three negative children. With the analysis of Case 2.5 above, we know that three operations plus the number of neutrally sorted children is necessary to sort the interval. This gives four operations plus the number of initially neutrally sorted children plus one, since we have sorted neutrally one additional child. This is more than what is needed to sort all children positively. □

So in the case of a linear node, one negative child of I is sorted positively. By the analysis of Case 2.1, the fastest way to complete the scenario is to sort all children positively one by one. ■

Theorem 3. *Let Π and Γ be two genomes on n genes, and $\mathcal{F}$ be a family of common intervals of Π and Γ. Let ℓ be the number of linear nodes in $\mathcal{T}(\mathcal{F}, \Pi, \Gamma)$. A minimum $\mathcal{F}$-perfect scenario that transforms Π into Γ can be computed in time $O(2^{\ell}n^2)$.*

Proof. The correctness of the algorithm follows directly from Lemma 3 and Lemma 5, and we only need to address the complexity of the algorithm.

Computing the common connected components of two graphs composed of paths and cycles is achieved in linear time (Habib et al., 2004). This gives the maximal common intervals. Computing the forest of decomposition trees can also be done in linear time (Hsu and McConnell, 2003).

In Algorithm 2, there is at most a constant number of DCJ distance computations performed at each node of the strong interval tree of I, and each can be performed in linear time (Bergeron et al., 2006). However, due to the case when three children are sorted negatively and all others are sorted neutrally, and that we need to complete the sorting of the maximal common interval I with two different ways to sort I, there can be in the worst case 2^{ℓ} scenarios sorting I that are computed, the shortest one being kept, each involving $O(n)$ DCJ

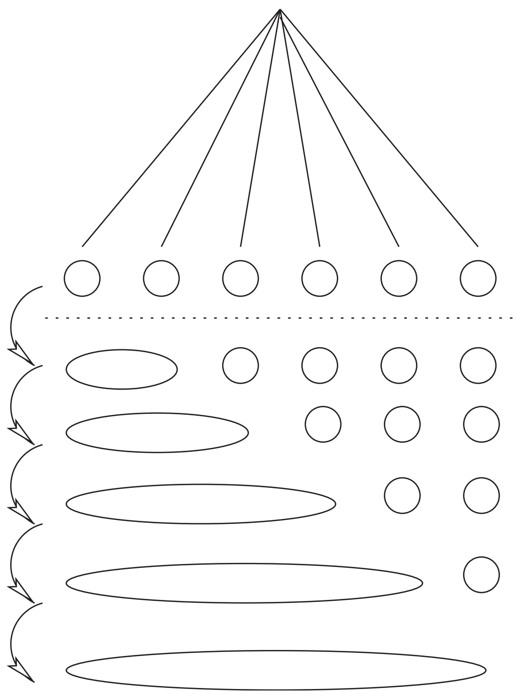

FIG. 9. Case 2.2: all children of I are sorted neutrally.

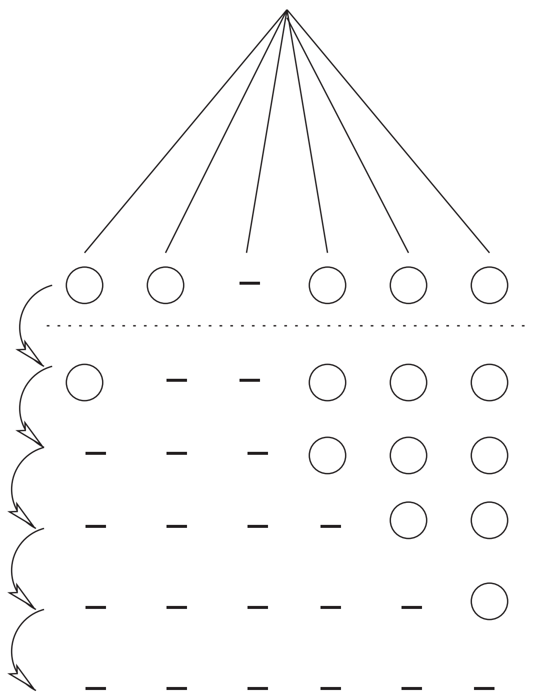

FIG. 10. Case 2.3: one child of *I* that is sorted negatively, the others are sorted neutrally.

distance computations, so $O(2^{\ell}m^2)$ worst-case time on each maximal common interval containing *m* genes. Altogether, for all maximal common intervals, as they are processed independently, this gives a complexity of $O(2^{\ell}n^2)$ for the second step.

It then remains to create the adjacencies of *Γ* that are not in any maximal common interval and are not in *Π′* and for that task, as we do not require to preserve any common interval, we can use the algorithm of Bergeron et al. (2006). By Lemma 3, every maximal common interval is sorted with a minimum number of operations, and one operation is enough to sort it in another direction. So if a minimum perfect scenario *S* between *Π* and *Γ* requires to sort a maximal common interval *I* in a direction that is not chosen by the algorithm, it takes one additional operation to sort *I* in this direction (i.e. to correct the direction it has been sorted into by our algorithm). This implies that *S* is not shorter than the perfect DCJ scenario obtained with our algorithm, which proves the fact that our algorithm computes a minimum perfect DCJ scenario. ■

Note that the exponential complexity is expressed in terms of linear nodes, unlike in the case of reversals where it was linked to prime nodes (Bérard et al., 2007, 2008b). Moreover, the required pattern is very specific and it can be expected that it happens rarely. Finally, as a consequence, if a forest of decomposition trees does not contain any linear node, the algorithm computes in polynomial time a perfect scenario that is also a shortest scenario.

Corollary 1. *Let Π and Γ be two genomes on n genes, and a family $\mathcal{F}$ of common intervals of Π and Γ. If $\mathcal{T}(\mathcal{F}, \Pi, \Gamma)$ does not contain any linear node, in particular if $\mathcal{F}$ is nested, a minimum $\mathcal{F}$-perfect scenario of length d(Π,Γ) can be computed in time $O(n^2)$.*

Proof. The time complexity of the scenario computation follows from Theorem 3. So it remains to prove that the number of DCJ operations in the computed scenario is *d*(*Π*,*Γ*). It follows from Theorem 1 and the fact that any DCJ operation in a scenario computed by our algorithm creates an adjacency of *Γ* as we use the optimal algorithm of Bergeron et al. (2006). ■

This result then defines a class of instances where a perfect scenario is also parsimonious. These instances are defined only in terms of the structure of the considered common intervals and not in terms of their breakpoint graph, which differs from similar results in the reversal model (Bérard et al., 2004; Sagot

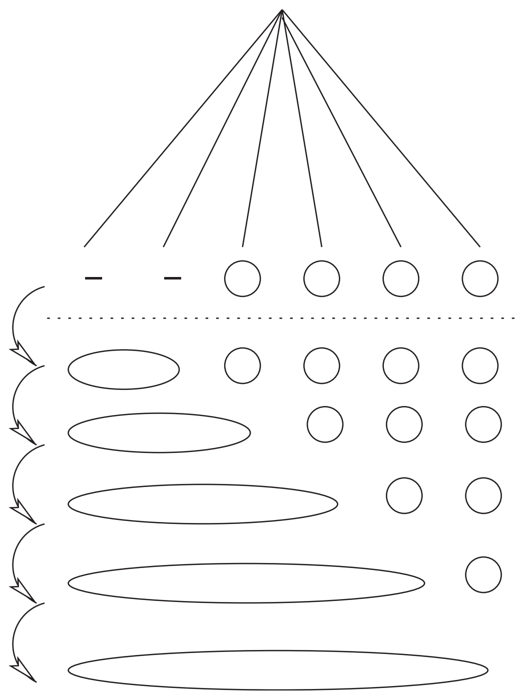

FIG. 11. Case 2.4: two children of *I* are sorted negatively, the others are sorted neutrally.

and Tannier, 2005; Diekmann et al., 2007). For example, if the family is nested, the algorithm runs in polynomial time.

7. CONCLUSION

We proved in this paper that $\mathcal{F}$-perfect sorting by DCJ is NP-hard in general, and even if $\mathcal{F}$ is a weakly separable family of common intervals. On the other hand, it has a polynomial time solution when $\mathcal{F}$ is nested. This contrasts with perfect sorting by reversals that is hard if $\mathcal{F}$ is nested, and easy if $\mathcal{F}$ is a weakly

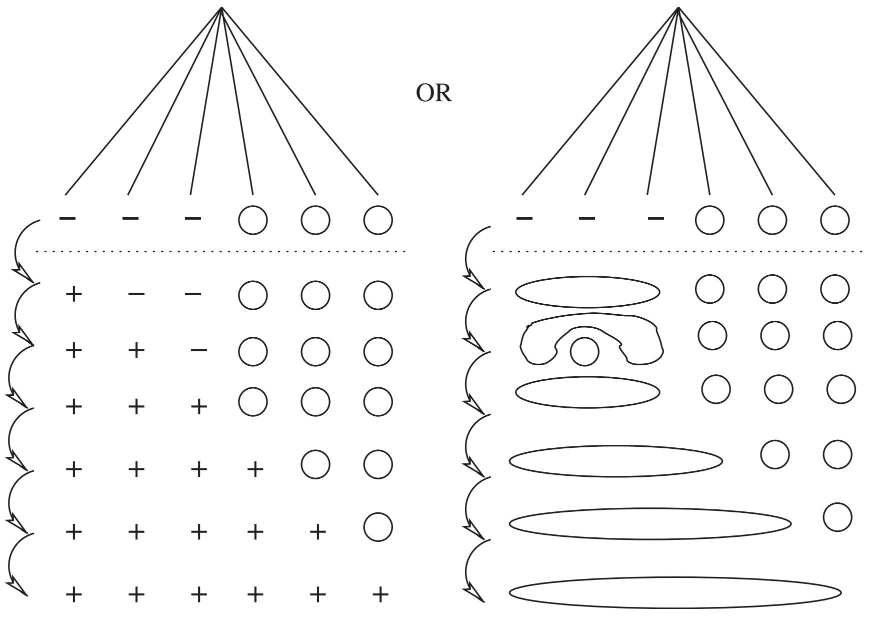

FIG. 12. Case 2.5: three children of *I* are sorted negatively, the others are sorted neutrally.

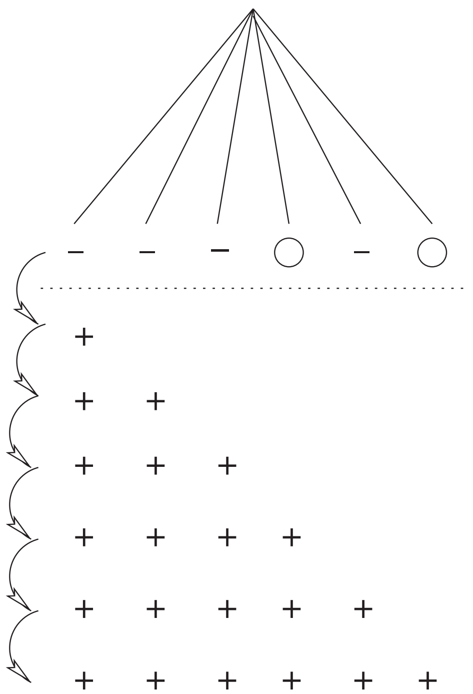

FIG. 13. Case 2.6: at least four children of I are sorted negatively, the others are sorted neutrally.

separable. The key to these results is the ability of DCJ to create temporary circular chromosomes, that was already the important factor in the fact that sorting with DCJ is simpler than with reversals (Bergeron et al., 2006). This illustrates that the DCJ model, both by its combinatorial simplicity and its pertinence for modeling genome rearrangements, offers an interesting way to attack several genome rearrangement problems (Mixtacki, 2008; Warren and Sankoff, 2008).

We also described a fixed parameter polynomial time algorithm for the problem of perfect DCJ rearrangement, using the patterns used in the NP-hardness proof as a parameter. It runs precisely in $O(2^{\ell}n^2)$, where ℓ is the number of times the algorithm runs into the configuration of a linear or circular node with three negative children and all other children neutral. The parameter ℓ is naturally bounded from above by the number of circular or linear nodes in the forest of decomposition trees. The average-time complexity of our algorithm is still open, and could be attacked with techniques similar than the ones used in Bouvel et al. (2009).

A natural problem that could benefit from such an algorithm is the perfect reversal median (Bernt et al., 2007), or perfect DCJ-median.

It is also tempting to investigate the relationships between the general DCJ model and the reversal/translocation/block-interchange model, as those two models have always been considered to be mostly equivalent, since two DCJs simulate block-interchanges. But computing a perfect scenario seems to be a case where these two models differ. Indeed, transforming chromosome $\{T3_t, 3_h2_t, 2_h1_t, 1_hT\}$ into $\{T1_t, 1_h2_t, 2_h3_t, 3_hT\}$ can be achived by one obvious block-interchange, which trivially preserves all common intervals. However, the two double cut-an-joins that simulate this block-interchange always break some common interval, so a perfect DCJ scenario has at least three operations. A polynomial algorithm for perfect rearrangement by reversals and block-interchanges is still possible and the question remains open.

Finally, the algorithm we describe for nested families of common intervals runs in quadratic time, but we think there is a linear time solution, with a smart treatment of prime nodes.

ACKNOWLEDGMENTS

We would like to thank Fabien de Montgolfier for his kind answer to some of our questions on bipartitive families and the reviewers for a very careful reading of the manuscript. C. Chauve is supported by grants from NSERC and SFU. C. Paul is supported by the ANR (grant ANR-O6-BLAN-0148-01 "GRAAL"). E. Tannier is funded by the ANR (grant ANR-08-GENM-036-01 "CoGeBi"). A. Chateau is supported by the ANR (grant BLAN07-1_185484 "CoCoGen").

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Alekseyev, M., and Pevzner, P. 2008. Multi-break rearrangements and chromosomal evolution. *Theor. Comput. Sci.* 395, 193–202.

Bérard, S., Bergeron, A., and Chauve, C. 2004. Conservation of combinatorial structures in evolution scenarios. *LNCS/LNBI* 3388, 1–14.

Bérard, S., Bergeron, A., Chauve, C., et al. 2007. Perfect sorting by reversals is not always difficult. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 4, 4–16.

Bérard, S., Chateau, A., Chauve, C., et al. 2008a. Perfect DCJ rearrangements. *LNCS* 5267, 158–169.

Bérard, S., Chauve, C., and Paul, C. 2008b. A more efficient algorithm for perfect sorting by reversals. *Inform. Proc. Letters* 106, 90–95.

Bergeron, A., Mixtacki, J., and Stoye, J. 2005. *Mathematics of Evolution and Phylogeny*. Oxford University Press, New York.

Bergeron, A., Mixtacki, J., and Stoye, J. 2006. A unifying view of genome rearrangements. *LNCS/LNBI* 4175, 163–173.

Bergeron, A., Chauve, C., de Montgolfier, F., et al. 2008. Computing common intervals of k permutations, with applications to modular decomposition of graphs. *SIAM J. Discrete Math.* 22, 1022–1039.

Bernt, M., Merkle, D., and Middendorf, M. 2007. A fast and exact algorithm for the perfect reversal median. *LNCS/LNBI* 4463, 305–316.

Bouvel, M., and Rossin, D. 2006. The longest common pattern problem for two permutations. *Pure Math. Appl.* 17, 55–69.

Bouvel, M., Chauve, C., Mishna, M., et al. 2009. Average-case analysis of perfect sorting by reversals. *LNCS* 5577, 314–325.

Braga, M., Sagot, M.-F., Scornavacca, C., et al. 2008. Exploring the solution space of sorting by reversals with experiments and an application to evolution. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 5, 348–356.

Caprara, A. 2003. The reversal median problem. *INFORMS J. Comp.* 15, 93–113.

Cunningham, W., and Edmonds, J. 1980. A combinatorial decomposition theory. *Can. J. Math.* 32, 734–765.

Darling, A., Mikls, I., and Ragan, M. 2008. Dynamics of genome rearrangement in bacterial populations. *PLoS Genet.* 4.

de Montgolfier, F. 2003. Décomposition modulaire des graphes. Théorie, extensions et algorithmes [Ph.D. dissertation]. Université Montpellier II, France.

Diekmann, Y., Sagot, M.-F., and Tannier, E. 2007. Evolution under reversals: parsimony and conservation of common intervals. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 4, 301–109.

El-Mabrouk, N., and Sankoff, D. 2003. The reconstruction of doubled genomes. *SIAM. Comput.* 32, 754–792.

Fertin, G., Labarre, A., Rusu, I., et al. 2009. *Combinatorics of Genome Rearrangements*. MIT Press, Cambridge, MA.

Figeac, M., Varré, J.-S. 2004. Sorting by reversals with common intervals. *LNCS/LNBI* 3240, 26–37.

Habib, M., Paul, C., and Raffinot, M. 2004. Common connected components of interval graphs. *LNCS* 3109, 347–358.

Hannenhalli, S., and Pevzner, P. 1999. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM* 46, 1–27.

Hannenhalli, S., and Pevzner, P.A. 1995. Transforming men into mice: polynomial algorithm for genomic distance problem. *FOCS* 581–592.

Hoberman, R., and Durand, D. 2005. The incompatible desiderata of gene cluster properties. *LNCS/LNBI* 3678, 73–87.

Hsu, W.-L., and McConnell, R.M. 2003. PC trees and circular-ones arrangements. *Theor. Comput. Sci.* 296, 99–116.

Jean, G., and Nikolski, M. 2007. Genome rearrangements: a correct algorithm for optimal capping. *Inform. Proc. Lett.* 104, 14–20.

Lemaitre, C., Braga, M., Sagot, M.-F., et al. 2009. Footprints of inversions at present and past pseudoautosomal boundaries in human sex chromosomes. *Genome Biol. Evol.* (in press).

McConnell, R., and de Montgolfier, F. 2005. Algebraic operations on pq-trees and modular decomposition trees. *31st Int. Workshop Graph-Theoretic Concepts Comput. Sci.*

Mixtacki, J. 2008. Genome halving under DCJ revisited. *LNCS* 5092, 276–286.

Sagot, M.-F., and Tannier, E. 2005. Perfect sorting by reversals. *LNCS* 3595, 42–51.

Tannier, E., Bergeron, A., and Sagot, M.-F. 2007. Advances on sorting by reversals. *Discrete Appl. Math.* 155, 881–888.

Tannier, E., Zheng, C., and Sankoff, D. 2009. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinform.* 10, 120.

Warren, R., and Sankoff, D. 2008. Genome halving with double cut and join. *APBC* 231–240.

Yancopoulos, S., Attie, O., and Friedberg, R., 2005. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* 21, 3340–3346.

Address correspondence to:
Dr. Eric Tannier
INRIA Rhônes-Alpes
Laboratoire de Biométrie et Biologie Evolutive
Université de Lyon 1
43 boulevard du 11 novembre 1918
Villeurbanne F-69622, France

E-mail: Eric.Tannier@inria.fr