Perfect DCJ rearrangement

Sèverine Bérard[1,2], Annie Chateau[2], Cedric Chauve[3], Christophe Paul[2], and Eric Tannier[4]

[1] Université Montpellier 2, UMR AMAP, Montpellier, F-34000 France.
[2] CNRS, LIRMM, CNRS UMR55076, Université Montpellier 2, Montpellier, France.
[3] Department of Mathematics, Simon Fraser University, Burnaby (BC), Canada
[4] INRIA, LBBE, CNRS UMR5558, Université de Lyon 1; Villeurbanne, France.

Abstract. We study the problem of transforming a multichromosomal genome into another using Double-Cut-and-Join (DCJ) operations. We introduce the notion of DCJ scenario that does not break families of common intervals (groups of genes co-localized in both genomes). Such scenarios are called perfect, and generalize the notion of perfect reversal scenarios. While perfect sorting by reversals is NP-hard if the family of common intervals is nested, we show that finding a shortest perfect DCJ scenario can be answered in polynomial time in this case. Moreover, while perfect sorting by reversals is easy when the family of common intervals is weakly separable, we show that the corresponding problem is NP-hard in the DCJ case. These contrast with previous comparisons between the reversal and DCJ models, that showed that most problems have similar complexity in both models.
Note. This extended abstract appeared in the proceedings of the 6th RECOMB Comparative Genomics Satellite Workshop (RCG 2008), volume 5267 of *Lecture Notes in Computer Science*, pages 158-169, Springer, 2008.

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 operations. 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 (see [10, 23, 11] for example). Probably the most used algorithmic results related to genome rearrangements concern the problem of sorting signed permutations by reversals. This problem aims at computing a shortest sequence of reversals that transforms one signed permutation into another, and can be solved in polynomial time [16, 7, 25]. It was later generalized to handle multichromosomal genomes with linear chromosomes, using rearrangements such as

translocations and chromosomes fusions and fissions [17]. Here, we study a more general rearrangement model on multichromosomal genomes, the Double-Cut-and-Join model (DCJ), that was considered in several recent works [28, 8, 2, 20, 21]. In this model, temporary circular chromosomes can be created, which allows to simulate rearrangements such as transpositions and block-interchanges using two consecutive DCJs [28].

Another way (than pure parsimony) of handling gene order data is to consider groups of genes that are co-localized with the homologous genes (genes having a single common ancestor) in the genomes of different species.These groups are likely together 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* [14]. Computing a reversal scenario of minimum length that preserves a given subset of the common intervals of two signed permutations is NP-hard [14] and several papers have explored this problem, describing families of instances that can be solved in polynomial time [3, 4, 24, 13] and fixed-parameter tractable algorithms [4, 5].

When comparing algorithmic properties of the reversal and DCJ models, the classical 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 [25, 8]; the median problems are both NP-hard[1]. 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 (the $\mathcal{F}$-perfect rearrangement problem) is still NP-hard, our results point to interesting differences between the reversal and DCJ models. If the family of common intervals is *nested*, we show that finding a perfect DCJ scenario of minimum length is solvable in polynomial time, while it is NP-hard for reversals [14]; if the family is *weakly separable*, we

[1] The median problem consists in, given three genomes as an input, find a fourth one (the median) that minimizes the sum of the distance from the median to the three input genomes. This problem has been proved to be NP-hard for permutations in [12], and for multichromosomal genomes in [26].

show that the DCJ problem is NP-hard, while this case was solved in polynomial time for reversals [4].

The paper is organized as follows: in Section 2, we introduce genomes, DCJ operations and common intervals. Then in Section 3, we define perfect DCJ scenarios for multichromosomal genomes and describe some basic properties of such scenarios. In Section 4, we define the different properties of families of common intervals, that result in different complexity status for the perfect rearrangement problems. In Section 5, we describe a polynomial algorithm for the $\mathcal{F}$-perfect rearrangement problem if $\mathcal{F}$ is nested, and in Section 6, we prove NP-hardness of the general problem.

2 Genomes, intervals and rearrangements

We follow the modeling of a genome used in [8]. 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 gene extremities. A *genome* is a set of adjacencies on a set of genes. Each adjacency in a genome means that two gene extremities are consecutive on 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 $a_t a_h$ for every gene a, plus the adjacencies of Π, except telomeric adjacencies. An example of such a graph is drawn on Figure 1.

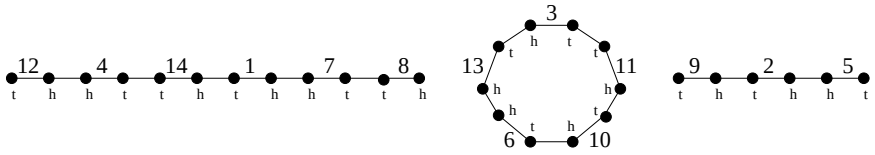

Fig. 1. The graph $G_{\Pi^{ex}}$, where Π^{ex} is given by the union of $C_1 = \{T12_t, 12_h4_h, 4_t14_t, 14_h1_t, 1_h7_h, 7_t8_t, 8_hT\}$, $C_2 = \{3_t11_t, 11_h10_t, 10_h6_t, 6_h13_h, 13_t3_h\}$ and $C_3 = \{T9_t, 9_h2_t, 2_h5_h, 5_tT\}$.

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.

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 on 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.

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 pr and qs, or ps and qr, creating two new adjacencies. Examples of DCJ operations are shown in Figure 2.

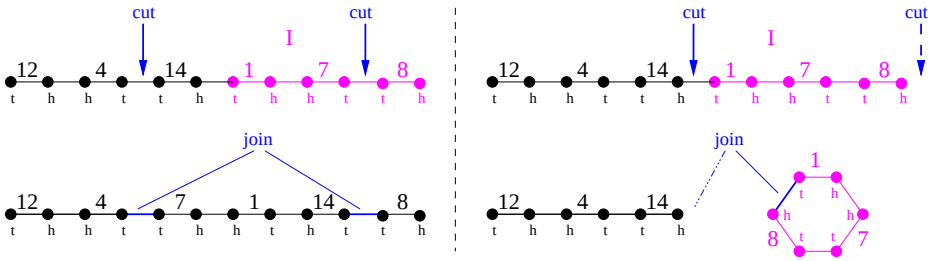

Fig. 2. Two examples of DCJ operations. Left: the DCJ cuts 4_t14_t and 7_t8_t and joins 4_t7_t and 14_t8_t (it is a reversal). Right: the DCJ cuts 14_h1_t and 8_hT and joins 14_hT and 8_h1_t. This operation produces a circular chromosome. The first operation breaks the interval $I = \{1, 7, 8\}$ whereas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

A DCJ operation can reverse an interval in a genome, fuse two chromosomes into one, fisse one chromosome into two, or exchange two intervals from two different chromosomes, both 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, introduced by Yancopoulos *et al.* [28] and since adopted by many others [8, 20, 1], sometimes under the name "2-break rearrangements" [2].

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 $d(\Pi, \Gamma)$.

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: 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 called *outside* if the two genes of which p and q are extremities do not belong to I; it is a *border* adjacency if one of the genes of which p and q are extremities belongs to I but not the other. In these three definitions, a T symbol is considered to be outside I.

Note that an interval of Π has zero or two border adjacencies. Let I be any 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. For example, on Fig. 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. A DCJ that does not preserve I is said to *break* I.

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 temporary circular chromosomes are not allowed, this definition coincides with the one of perfect scenarios of reversals [14, 3–5, 24, 13].

With this definition of $\mathcal{F}$-perfect DCJ scenarios the elements of an interval I of $\mathcal{F}$ can be not consecutive at some point of such a scenario, provided that the elements of I are split into at most one linear segment and possibly several circular segments. This allows to use the property of the DCJ model to create temporary circular chromosomes.

4 Families of common intervals

Given two genomes Π and Γ, two common intervals are said to *overlap* if their intersection is not empty and none is contained in the other. A common interval I of Π and Γ is *strong* if I does not overlap any other common interval. It is *maximal* if it is strong and not contained in another common interval.

A family $\mathcal{F}$ of common intervals is *weakly partitive* if for every two overlapping intervals I and J of $\mathcal{F}$, $I \cup J$, $I \cap J$, $I - J$ and $J - I$ belong to $\mathcal{F}$. We denote by $\mathcal{F}^*$ the unique smallest weakly partitive family that contains $\mathcal{F}$; $\mathcal{F}^*$ can be computed in polynomial time. It follows immediately from [4] that a DCJ scenario is $\mathcal{F}$-perfect if and only if it is $\mathcal{F}^*$-perfect. A family $\mathcal{F}$ is called *nested* if every element of $\mathcal{F}$ is strong (note this implies that $\mathcal{F} = \mathcal{F}^*$). $\mathcal{F}$ is called *weakly separable* if every *strong* interval of $\mathcal{F}$ with at least three elements is the union of two overlapping intervals of $\mathcal{F}$. Of course, 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.

By definition, the sub-family of strong intervals of $\mathcal{F}^*$ for a family $\mathcal{F}$ 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 $\mathcal{F}^*$, and its children are the maximal strong common intervals of $\mathcal{F}^*$ it properly contains (see [4, 18]). 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 tree is given in Fig. 3. Given a maximum common interval, the tree can be computed in linear time and space [6, 18]. A node of the forest of strong intervals is called *prime* if it has at least four children and it properly contains no common interval including more than one of its children. It is *linear* if it has two elements or it is the union of two overlapping common intervals, both containing a subset of its children. Any strong interval of $\mathcal{F}^*$ is either prime or linear (see for instance [4]).

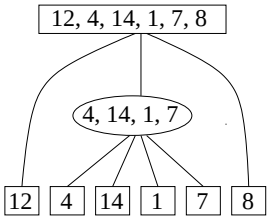

Fig. 3. The tree that represents the strong common intervals of the maximal common interval $I = \{12, 4, 14, 1, 7, 8\}$ of Π^{ex} and Γ^{ex}, given by the union of $C_1 = \{T12_t, 12_h14_h, 14_t7_h, 7_t4_t, 4_h1_h, 1_t8_t, 8_h2_t, 2_h6_t, 6_hT\}$ and $C_2 = \{T9_t, 9_h3_t, 3_h10_t, 10_h5_t, 5_h11_h,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

It is known [14] 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[2], the algorithm described in [4] computes an $\mathcal{F}$-perfect reversal scenario in polynomial time. We prove here the exact opposite results for multichromosomal genomes with DCJ operations.

5 $\mathcal{F}$ nested: A polynomial-time solvable case

We give here an algorithm to solve the $\mathcal{F}$-perfect DCJ problem if $\mathcal{F}$ is a nested family.

Sorting an interval. We say that a common interval I is *sorted* in Π with respect to 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.

We can distinguish different kinds of DCJ with respect to a common interval I. 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. Note that a DCJ does not break I if and only if it cuts inside or outside I.

Lemma 1. *If a DCJ scenario S_0 between two genomes Π and 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.*

This lemma, that is an equivalent to a lemma stated for reversals in [14], implies that the DCJs that sort I may always be applied before the ones that rearrange the remaining of the genome.

Outline of the algorithm. The algorithm can be decomposed into three main steps:

1. Compute the maximal common intervals of Π and Γ. This can be done by computing the maximal common connected components of G_Π and G_Γ, with techniques presented for example in [15].

[2] The terminology *weakly separable* is inspired by the notion of separable permutations, that are the permutations whose common intervals with the identity define a strong interval tree with no prime node. For a weakly separable family of common intervals, the strong intervals forest can have prime nodes, but no edge can be incident to two prime nodes, and these prime nodes belong to $\mathcal{F}^*$ but not $\mathcal{F}$ and are then only implicitly defined by $\mathcal{F}$.

2. For each maximal common interval I, compute the tree of (strong) intervals that lie in I. By a preorder traversal of this tree, sort each node assuming its children have been sorted, by a technique we describe further.
3. Finally, after all maximal common intervals have been sorted, compute a parsimonious series of DCJ that creates all the remaining adjacencies of Γ, that are not inside any maximal common interval, for example with the technique described in [8].

The first and last step use known techniques that are described in the literature. The core of our method is the second step, which we now describe into details. Lemma 1 implies that a perfect scenario between two genomes can be computed during a preorder traversal of each tree of common intervals in such a way that, when processing a node I, all its children are already sorted with respect to Γ.

The sorting direction of an interval. We now consider a strong common interval I of 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.

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. We will use only DCJs that cut inside I, so in Π', there is a limited number of possibilities regarding the border adjacencies of I in Π'. If I has no border adjacencies in Γ, the set of adjacencies in Π' is unambiguous: it is the set of adjacencies inside I in Γ, plus the adjacency $x_\Pi y_\Pi$. But if I has border adjacencies in Γ, there are three possibilities for the border adjacencies:

1. $x_\Pi m_\Gamma$ and $M_\Gamma y_\Pi$ are adjacencies in Π', and in this case we say that I is sorted *positively*;
2. $x_\Pi M_\Gamma$ and $m_\Gamma y_\Pi$ are adjacencies in Π', and in this case Π is sorted *negatively*;
3. $m_\Gamma M_\Gamma$ and $x_\Pi y_\Pi$ are adjacencies in Π', and in this case Π is sorted *neutrally*

The way I is sorted is called its *sorting direction* [3] in Π'.

[3] Note that the notions of positive or negative sorting direction of a common interval are strongly related to the choice of gene extremities x_Π, y_Π, m_Γ and M_Γ (two

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

As it was shown in [5] for the reversal model, the main difficulty for sorting a genome while preserving common intervals is to choose among the sorting directions of these intervals. The following lemma greatly simplifies this choice in the DCJ model.

Lemma 2. *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:*

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

The algorithm. This yields a method for sorting a maximal common interval, where the DCJ operations to apply can be computed using the algorithm presented in [8]:

Algorithm 1: $\mathcal{F}$-Perfect sorting of a maximal common interval I of genomes Π and Γ, given the strong interval tree T of a nested family $\mathcal{F}$ of common intervals in I.

LET $\Pi' = \Pi$
FOR *each interval* $I' \subseteq I$ *of* Π *and* Γ *in a post-traversal order of* T
  {Note: all children of I' are sorted}
  IF I' *has no border adjacencies in* Γ {Note: possible only if $I' = I$}
    Sort I' *with a minimum number of DCJs inside* I' *and outside its children*
  Else
    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*
  LET Π' *denote the resulting genome.*

Lemma 3. *Given two genomes Π and Γ, a nested family $\mathcal{F}$ of common intervals and a maximal element I of $\mathcal{F}$, Algorithm 1 computes a DCJ scenario that sorts I with respect to Γ and preserves all the intervals of $\mathcal{F}$ contained in I. The scenario is minimum, and no scenario achieves the same number of operations and sorts I with another direction.*

different choices are possible, and it will swap the positive and negative sorts). We choose arbitrarily and independently for all strong intervals.

Lemma 2, together with Lemma 3, provides the general result:

Theorem 1. *Given two genomes Π and Γ on n genes, and a nested family $\mathcal{F}$ of common intervals, a minimum $\mathcal{F}$-perfect scenario of length $d(\Pi, \Gamma)$ can be computed in time $O(n^2)$.*

Note that this result 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 [3, 24, 13].

6 $\mathcal{F}$ general (even 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 NP-hardness for reversals [14], but contrasts with the linear time solution when $\mathcal{F}$ is supposed to be weakly separable [4].

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

The NP-hardness proof relies on a very simple pattern: it uses the fact that it is possible to sort an interval of shape (3 2 1) in Π and (1 2 3) in Γ either neutrally or negatively in three operations, and it is impossible to choose between the two directions. No DCJ scenario sorts this pattern positively in less than 4 operations, while preserving the intervals $\{1, 2\}$ and $\{2, 3\}$. From this, we can deduce two interesting remarks that will be developed in an extended version of this paper:

- The behavior of this perfect DCJ problem is different from the perfect rearrangement problem where temporary circular chromosomes have to be reinserted immediately to simulate block-interchanges. Indeed, for the latter, a block interchange would have sorted (3 2 1) into (1 2 3) while preserving all intervals. It is not the case when the block-interchange has to be simulated by two consecutive DCJs. This points an interesting difference between DCJ problems and block-interchange problems, and calls for further thoughts on the relationship between the DCJ model and the reversals and block-interchange model.
- The pattern that causes NP-hardness is limited to linear strong intervals with three elements. It is then possible to devise FPT algorithms based on the number of such patterns in the genomes. Like the FPT algorithms for reversals [4, 5], this should lead to efficient algorithms to solve the perfect DCJ problem.

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 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 [8]. 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 [22, 27].

In an extended version of this paper, we will describe a fixed parameter polynomial algorithm for the problem of perfect DCJ rearrangement, using the number of patterns used in the NP-hardness proof as a parameter. A natural problem that could benefit from such an algorithm is the perfect reversal median [9], or perfect DCJ-median [1, 19]. We also plan to investigate the relationships between the general DCJ model and the reversal/translocation/block-interchange model, as the problem of computing a perfect scenario seems to be the first one where these two models differ. This seems to be surprising, as those two models have always been considered to be equivalent, since two DCJs simulate block-interchanges. We will also address the case of genomes with circular chromosomes using the notion of PC-trees [18]. Eventually, 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

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 ANR JC05_49162 "REGLIS" and NT05-3_45205 "GENOMICRO". A. Chateau is supported by the ANR BLAN07-1_185484 "CoCoGen".

References

1. Z. Adam and D. Sankoff. The ABC of MGR with DCJ. *Evol. Bioinformatics*, 4:69–74, 2008.

2. M. Alekseyev and P. Pevzner. Multi-break rearrangements and chromosomal evolution. *Theor. Comput. Sci.*, 2008. In press.
3. S. Bérard, A. Bergeron, and C. Chauve. Conservation of combinatorial structures in evolution scenarios. In *RCG 2004*, volume 3388 of *LNCS/LNBI*, pages 1–14, 2004.
4. S. Bérard, A. Bergeron, C. Chauve, and C. Paul. Perfect sorting by reversals is not always difficult. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 4:4–16, 2007.
5. S. Bérard, C. Chauve, and C. Paul. A more efficient algorithm for perfect sorting by reversals. *Inform. Proc. Letters*, 106:90–95, 2008.
6. A. Bergeron, C. Chauve, F. de Montgolfier, and M. Raffinot. Computing common intervals of k permutations, with applications to modular decomposition of graphs. In *ESA 2005*, volume 3669 of *LNCS*, pages 779–790, 2005.
7. A. Bergeron, J. Mixtacki, and J. Stoye. *Mathematics of Evolution and Phylogeny*, chapter The inversion distance problem. Oxford University Press, 2005.
8. A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. In *WABI 2006*, volume 4175 of *LNCS/LNBI*, pages 163–173, 2006.
9. M. Bernt, D. Merkle, and M. Middendorf. A fast and exact algorithm for the perfect reversal median. In *ISBRA 2007*, volume 4463 of *LNCS/LNBI*, pages 305–316, 2007.
10. G. Bourque and P. Pevzner. Genome-scale evolution: reconstructing gene orders in the ancestral species. *Genome Res.*, 12:26–36, 2002.
11. M. Braga, M.-F. Sagot, C. Scornavacca, and E. Tannier. Exploring the solution space of sorting by reversals with experiments and an application to evolution. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 2008.
12. A. Caprara. The reversal median problem. *INFORMS J. Comp.*, 15:93–113, 2003.
13. Y. Diekmann, M.-F. Sagot, and E. Tannier. Evolution under reversals: Parsimony and conservation of common intervals. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 4:301–109, 2007.
14. M. Figeac and J.-S. Varré. Sorting by reversals with common intervals. In *WABI 2004*, volume 3240 of *LNCS/LNBI*, pages 26–37, 2004.
15. M. Habib, C. Paul, and M. Raffinot. Common connected components of interval graphs. In *CPM 2004*, volume 3109 of *LNCS*, pages 347–358, 2004.
16. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, 46:1–27, 1999.
17. S. Hannenhalli and P. A. Pevzner. Transforming men into mice: polynomial algorithm for genomic distance problem. In *FOCS 1995*, pages 581–592, 1995.
18. W.-L. Hsu and R. M. McConnell. PC trees and circular-ones arrangements. *Theor. Comput. Sci.*, 296:99–116, 2003.
19. R. Lenne, C. Solnon, T. Stutzle, E. Tannier, and M. Birattari. Reactive stochastic local search algorithms for the genomic median problem. In *EVOCOP 2008*, volume 4972 of *LNCS*, pages 266–276, 2008.
20. Y. Lin and al. An efficient algorithm for sorting by block-interchange and its application to the evolution of vibrio species. *J. Comput. Biol.*, 12:102–112, 2005.
21. L. Lu, Y. Huang, T. Wang, and H.-T. Chiu. Analysis of circular genome rearrangement by fusions, fissions and block-interchanges. *BMC Bioinformatics*, 7:295, 2006.
22. J. Mixtacki. Genome halving under DCJ revisited. In *COCOON 2008*, 2008.
23. W. Murphy and *et al.* Dynamics of mammalian chromosome evolution inferred from multispecies comparative maps. *Science*, 309:613–617, 2005.
24. M.-F. Sagot and E. Tannier. Perfect sorting by reversals. In *COCOON 2005*, volume 3595 of *LNCS*, pages 42–51, 2005.

25. E. Tannier, A. Bergeron, and M.-F. Sagot. Advances on sorting by reversals. *Discrete Appl. Math.*, 155:881–888, 2007.
26. E. Tannier, C. Zheng, and D. Sankoff. Multichromosomal genome median and halving problems. In *Proceedings of WABI'08*, 2008.
27. R. Warren and D. Sankoff. Genome halving with double cut and join. In *APBC 2008*, pages 231–240, 2008.
28. S. Yancopoulos, O. Attie, and R. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21:3340–3346, 2005.

A The DCJ distance and the breakpoint graph

The formula for the DCJ distance based on the breakpoint graph is much used in the proofs of our results. The *breakpoint graph* of two genomes Π and Γ on the same set of genes, denoted by $BP(\Pi, \Gamma)$, is the bipartite graph which 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 (see Fig. 4).

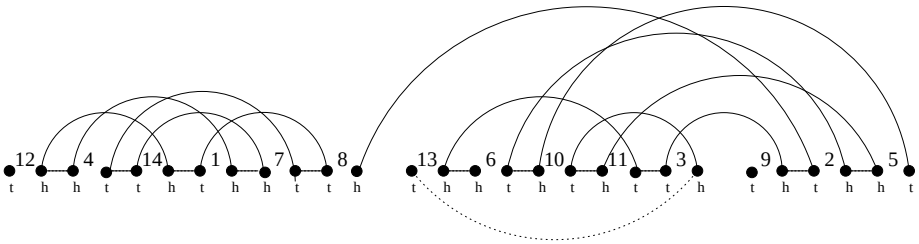

Fig. 4. The breakpoint graph of the genomes Π^{ex} (see Figure 1) and Γ^{ex}, given by the union of $C_1 = \{T12_t, 12_h14_h, 14_t7_h, 7_t4_t, 4_h1_h, 1_t8_t, 8_h2_t, 2_h6_t, 6_hT\}$ and $C_2 = \{T9_t, 9_h3_t, 3_h10_t, 10_h5_t, 5_h11_h, 11_t13_h, 13_tT\}$. Π^{ex}-edges are dotted lines, and Γ^{ex}-edges are plain lines.

The *DCJ-distance* is immediately readable from the breakpoint graph, as stated by Theorem 3, that restates the main result of [8] in terms of the breakpoint graph in place of the *adjacency graph*[4].

Theorem 3. [8] *For two genomes Π and Γ, 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. The DCJ distance is*

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

The basis for this result is Lemma 4, that is implicit in [8], and states that any – greedy – DCJ that creates an adjacency that is present in Γ but not in Π is optimal.

[4] The breakpoint graph $BP(\Pi, \Gamma)$, introduced for permutations in [16], is the line-graph of the *adjacency graph* introduced in [8].

Lemma 4. *For two genomes Π and Γ, if a DCJ operation on Π results in a genome Π' containing an adjacency that is present in Γ but not in Π, then $d(\Pi, \Gamma) = d(\Pi', \Gamma) + 1$.*