Perfect sorting by reversals is not always difficult

Sèverine Bérard, Anne Bergeron, Cedric Chauve, and Christophe Paul

***Abstract*— We propose new algorithms for computing pairwise rearrangement scenarios that conserve the combinatorial structure of genomes. More precisely, we investigate the problem of sorting signed permutations by reversals without breaking common intervals. We describe a combinatorial framework for this problem that allows to characterize classes of signed permutations for which one can compute in polynomial time a shortest reversal scenario that conserves all common intervals. In particular we define a class of permutations for which this computation can be done in linear time with a very simple algorithm that does not rely on the classical Hannenhalli-Pevzner theory for sorting by reversals. We apply these methods to the computation of rearrangement scenarios between permutations obtained from 16 synteny blocks of the X chromosomes of the human, mouse and rat.**

***Index Terms*— Evolution scenarios, reversals, common intervals.**

I. Introduction

THE reconstruction of evolution scenarios based on genome rearrangements, and in particular reversals and translocations, has proven to be a powerful tool to understand the evolution of groups of species. For eukaryotic genomes, several evolution scenarios have been recently proposed between vertebrates genomes [10], [11], [32], using the MGR and GRIMM softwares [9], [39]. These scenarios lead to interesting insight on the architecture of ancestral genomes, the evolution pattern across different lineages or the presence of genome regions prone to be involved in rearrangements (the so-called "breakpoint reuse" hypothesis) [31], [33], [36]. Putative evolution scenarios based on rearrangements were also computed on large datasets of prokaryotic genomes [3], [16]. In this paper, we describe new combinatorial and algorithmical results for computing such scenarios, based on the combinatorial problem of *sorting by reversals*.

Current approaches for sorting by reversals: At the heart of the computation of such rearrangement scenarios is the encoding of genomes by *signed permutations*, where each element of a permutation represents a genomic segment – from large synteny blocks in [10] to genes in prokaryotic genomes analysis [16] –, and the problem of *sorting signed permutations by reversals*, introduced by Sankoff [35]: given two signed permutations, find a "good" sequence of reversals that transforms one into the other one. [35]. In the original approach, a "good" sequence of reversals is a parsimonious sequence of reversals. This approach was pioneered, among others, by Hannenhalli and Pevzner, who described a combinatorial and algorithmical framework, known as the Hannenhalli-Pevzner theory, leading to polynomial time algorithms for computing parsimonious sequences of reversals sorting signed permutations [21]. Later, their approach was refined and simplified by several authors and the current best algorithm to compute a parsimonious reversal scenario runs in subquadratic time [38]. Note that the best algorithm to compute the length of a parsimonious reversal scenario, known as the *reversal distance*, runs in linear time [2], [8].

However, the approach based on parsimonious pairwise scenarios suffers from at least two limitations. First, it was shown in [6] that the number of such scenarios can be exponential, and it then becomes problematic to pick one in particular. This problem was also addressed, from a statistical point of view, in a recent study of reversals scenarios for metazoan mitochondrial genomes [26]. Another problem is that, when considering more than two species, that is computing evolution scenarios based on a given phylogenetic tree, it is often impossible to compute a multi-species scenario that induces pairwise parsimonious scenarios. Hence the parsimony constraint on computed scenarios has to be relaxed in some way. A successful way to deal with this problem has been to use the *median* approach [9], that still relies on the detection of "good reversals" in the sense of the Hannenhalli-Pevzner theory for computing parsimonious scenarios, but allowing to consider non-optimal reversals. These two problems suggest the need for combinatorial models and algorithms that allow both to compute parsimonious and non-parsimonious scenarios.

Perfect sorting by reversals: In the present work, we are interested in pairwise scenarios between two unichromosomal genomes, represented by two signed permutations, that do not break combinatorial structures – defined in terms of genomic segments – that are present in both permutations. The combinatorial structures we consider here are *common intervals of signed permutations* [7], [22], [41]. Roughly speaking, a common interval of two signed permutations is a set of elements that forms an interval in both permutations, or in other words, that is conserved in the two permutations up to local rearrangements. The rationale for this approach is that the conservation of such groups of genomic segments in two genomes is a character that is likely to have been present in the genome of their common ancestor, and is worth to be considered in the computation of evolution scenarios. One can consider, for example, common intervals defined by operons or über-operons in prokaryotic genomes [27]. Note that the current approaches used to compute rearrangement scenarios

S. Bérard is with the Département de Mathématiques et d'Informatique Appliquées, INRA Toulouse, Chemin de Borde Rouge, BP 52627, Castanet-Tolosan 31326 Cedex, France. Email: Severine.Berard@toulouse.inra.fr

A. Bergeron is with the Départment d'Informatique and Comparative Genomics Laboratory, Université du Québec à Montréal (UQÀM), CP 8888, succ. centre-ville, Montréal (QC) H3C3P8, Canada. Email: anne.bergeron@uqam.ca

C. Chauve is with (1) the Department of Mathematics, Simon Fraser University and (2) the Départment d'Informatique, Comparative Genomics Laboratory and LaCIM, UQÀM, CP 8888, succ. centre-ville, Montréal (QC) H3C3P8, Canada. Email: chauve@lacim.uqam.ca

C. Paul is with the LIRMM, Université Montpellier II, 161 rue Ada Montpellier Cedex 5, France. Email: paul@lirmm.fr

do sometimes produce scenarios that break common intervals, see [4].

The precise problem we address, namely *perfect sorting by reversals*, is the following: given two signed permutations and the set of intervals common to these two permutations, find a shortest sequence of reversals that transforms one permutation into the other without breaking any of the considered common intervals. The scenarios that do not break any common intervals are called *perfect scenarios*. This approach can be seen as a variant of the classical sorting by reversal problem where the parsimony criterion has been relaxed in order to include the conservation of common intervals. This problem can be generalized by considering an arbitrary set of common intervals. These problems were first introduced by Figeac and Varré [17], who described an exponential time algorithm solving the latter one. It was later shown that perfect scenarios that are also parsimonious scenarios can be computed in polynomial time [4], [34]. Our main result is the precise description of a combinatorial framework that leads to polynomial time algorithms to compute perfect scenarios for large classes of signed permutations.

Plan of the paper: In Section II, we define precisely the notions of reversal, scenario, common interval, and the problem of perfect sorting by reversals. In Section III, we introduce the notion of *strong intervals* of a signed permutation. These strong intervals form a linear size basis of the set of common intervals of a permutation. The strong intervals of a signed permutation can be arranged in a tree structure, called the *strong interval tree*, that is a central combinatorial tool to design algorithms computing perfect scenarios. Note that the strong interval tree of a permutation has a deep relationship with the theory of modular decomposition of permutation graphs [7], [30], that is described in Appendix. In Section IV, we show that perfect scenarios can be characterized precisely in terms of the vertices of the strong interval tree, which makes this structure a “guide” for computing perfect scenarios. Building on this fact, we propose (1) a subquadratic time algorithm for computing perfect scenarios that are parsimonious among the set of all perfect scenarios, for large classes of signed permutations, and (2) an exponential time algorithm for the general case, where the exponential time behavior is bounded by a parameter that can be easily read on the strong interval tree. We also show that our algorithms can be used to consider only a subset of the common intervals of a signed permutation. We illustrate our algorithms by computing perfect scenarios between the X chromosomes of the human, mouse and rat genomes, already considered in [18]. In Section V, we extend the results of [4] on a remarkable class of perfect scenarios, called *commuting scenarios*, and we show that such scenarios can be computed in linear time and that the signed permutations that can be sorted by such scenarios can be characterized solely in terms of their strong interval tree. We conclude by some open problems related to perfect scenarios.

II. Sorting by reversals and common intervals

In this section, we introduce the main concepts covered in this paper: signed permutation, reversal, scenario, commuting reversals, common interval and perfect scenario. A *signed permutation* on n elements is a permutation on the set of integers $\{1, 2, \ldots, n\}$ in which each element has a sign, positive or negative. Negative integers are represented by placing a bar over them. An *interval* of a signed permutation is a segment of consecutive elements of the permutation. An interval can be defined by the set of its unsigned elements, called its *content*. However, not every set of integers corresponds to an interval of a given permutation P.

The *reversal* of an interval of a signed permutation reverses the order of the elements of the interval, while changing their signs. Note that every reversal is an interval of the permutation on which it is performed, which leads us to often treat reversals as intervals, and to represent a reversal by the corresponding interval. If P is a permutation, we denote by $\overline{P}$ the permutation obtained by reversing the complete permutation P.

Example 1: Let $P = (1\ \overline{3}\ \overline{2}\ 5\ 4\ 6)$ be a signed permutation on 6 elements, then $\overline{P} = (\overline{6}\ \overline{4}\ \overline{5}\ 2\ 3\ \overline{1})$. Reversing, in P, the interval $(\overline{3}\ \overline{2}\ 5\ 4)$, or equivalently the set $\{2, 3, 4, 5\}$, yields the signed permutation $(1\ \overline{4}\ \overline{5}\ 2\ 3\ 6)$.

Definition 1: Let P and Q be two signed permutations on n elements. A *scenario* between P and Q is a sequence of distinct reversals that transforms P into Q, or P into $\overline{Q}$. The *length* of such a scenario is the number of reversals it contains. When Q is the identity permutation, a scenario between P and Q will be simply called a *scenario* for P.

The fact that the set of scenarios between P and Q contains sequences of reversals that transform P into $\overline{Q}$ models the fact that, in comparative genomics, permutations are used to represent chromosomes. Reversing a complete chromosome does not modify its structure.

Example 2: Reversing successively the intervals $\{2, 3, 5\}$, $\{3, 5\}$, $\{3\}$, $\{4\}$ and $\{4, 5\}$ is a scenario of length 5 for permutation $P = (1\ \overline{3}\ \overline{5}\ \overline{2}\ 4\ 6)$.

Given a signed permutation P on n elements, the problem of *sorting by reversals*, introduced by Sankoff in [35], asks for a *parsimonious scenario*, which is a scenario for P of minimal length among all possible scenarios. The first polynomial time algorithm solving this problem was given by Hannenhalli and Pevzner in [21]. Subsequent improvements were proposed, in particular in [23], [24], and the best known algorithm in $O(n\sqrt{n \log(n)})$ time [38].

Definition 2: Two distinct intervals I and J *commute* if their contents trivially intersect, that is either $I \subset J$, or $J \subset I$, or $I \cap J = \emptyset$. If intervals I and J do not commute, they *overlap*.

Definition 3: Let P be a signed permutation on n elements. A *common interval* of P is a set of one or more integers that is an interval in both P and the identity permutation Id_n. Note that any such set is also an interval of $\overline{P}$ and of $\overline{Id_n}$. The singletons and the set $\{1, 2, \ldots, n\}$ are always common intervals, and are called *trivial* common intervals.

Example 3: The common intervals of $P = (1\ \overline{3}\ \overline{2}\ 5\ 4\ 6)$ are $\{2, 3\}$, $\{1, 2, 3\}$, $\{4, 5\}$, $\{4, 5, 6\}$, $\{2, 3, 4, 5\}$, $\{2, 3, 4, 5, 6\}$,

$\{1,2,3,4,5\}$, $\{1,2,3,4,5,6\}$, and the singletons $\{1\}$, ..., $\{6\}$.

The notion of *common interval* was introduced in [41]. It was studied, among others in [22], to model the fact that a group of genes can be rearranged in a genome but still remains connected. It was also studied, in connection with reversal scenarios, in [4], [34]. In [41], Uno and Yagiura proposed the first algorithm to compute the set of common intervals of a permutation P in time $O(n+N)$, where N is the number of such common intervals. However, N can be of size $O(n^2)$. Heber and Stoye [22] defined the subset of *irreducible* common intervals that contains $O(n)$ common intervals and forms a representation of the common intervals, in the sense that every common interval is a chain of overlapping *irreducible* common intervals. They proposed an $O(kn)$ time algorithm to compute the set of irreducible common intervals of k permutations of n elements. A simpler algorithm is given in [7] (see also [12] for a related work).

Definition 4: Let P be a signed permutation. A scenario S for P is called a *perfect scenario* if every reversal of S commutes with every common interval of P. A perfect scenario of minimal length is called a *parsimonious perfect scenario*.

Note that, if I is a common interval of P and J is an interval of P that does not commute with I, then reversing J in P leads to a permutation P' such that I is not a common interval of P'. Hence, if J belongs to a scenario for P, then the set of common intervals of P is not conserved during this scenario, which explains the above definition.

Remark 1: From a biological and evolutionary point of view, it can be natural to be interested in scenarios that do not break a precise subset of the common intervals. All our results apply to this more general problem. However, for the clarity of the exposition, we will consider perfect scenarios as defined in Definition 4, and we will refer to Section IV-C for the general problem.

There always exists a perfect scenario for a given signed permutation P [17]. However, the authors of [17] claim that computing a parsimonious perfect scenarios for an arbitrary set of common intervals is intractable: NP-hard in general. Hence the difficulty of the problem relies in the parsimonious aspect. The main goal of this paper is to propose efficient algorithms to compute parsimonious perfect scenarios for large classes of signed permutations. Our results rely on the *strong interval tree* of a signed permutation described in the next section.

III. STRONG INTERVAL TREE

As the number of common intervals of a permutation P on n elements can be quadratic in n, an efficient algorithm (i.e. subquadratic time) for computing perfect scenarios should rely on a space efficient encoding of the set of common intervals. This section states structural properties of the set of common intervals of a permutation P that are central in the design of the algorithms for computing perfect scenarios. As Section IV-C considers the same problem, but with respect to a subset of common intervals, all the following results are special cases of those presented in Section IV-C. Thus, the proofs will only be presented in Section IV-C.

It should be noticed that, in [30], the author pointed out a correspondence between common intervals of permutations and the concept, well studied in graph theory, of *modules* of graphs. Thereby, all the results presented in this section and in Section IV-C can be seen as direct consequences or corollaries of well known graph theoretical results about modules in graphs and modular decomposition of graphs (or more generally of the framework of partitive set families [29]). A short description of the link between common intervals and modules of graphs is given in Appendix.

First, we can remark that being a common interval for an interval I has nothing to do with the sign of the elements of I. Therefore all the structural results presented in this section are valid for both signed and unsigned permutations, and for the sake of simplicity, we omit the signs.

Let I be a common interval of a permutation P on n and $x \in \{1,2,\ldots,n\}$ such that $x \notin I$. It follows from the definition of common interval that either x is larger than all elements of I or x is smaller than all elements of I. The order relation between x and I will be denoted $x < I$ or $I < x$. Similarly, for two *disjoint* common intervals I and J, $I < J$ means that any element of I is smaller than any element of J.

Definition 5: A common interval I of a permutation P is a *strong* interval of P if it commutes with every common interval of P.

Example 4: The strong intervals of permutation $P = (1\ 4\ 2\ 5\ 3\ 7\ 8\ 6\ 9)$ are $\{2,3,4,5\}$, $\{7,8\}$, $\{6,7,8\}$, $\{1,2,3,4,5,6,7,8,9\}$ and the singletons $\{1\}$, ..., $\{9\}$. The singletons and $\{1,2,\ldots,9\}$ are the *trivial* strong intervals of P.

It follows from Definition 5 that the inclusion order of the set of strong intervals defines an n-leaf tree, denoted by $T_s(P)$, whose leaves are the singletons, and whose root is the interval containing all elements of the permutation. We call the tree $T_s(P)$ the *strong interval tree* of P (see Fig. 1), and we identify a vertex of $T_s(P)$ with the strong interval it represents. Since each strong interval with more than one element, or equivalently each internal vertex of $T_s(P)$, has at least two children in $T_s(P)$, a permutation has $O(n)$ strong intervals. But the most interesting property of the set of strong intervals is that it forms a "basis" of the set of common intervals.

Definition 6: Let P be a permutation. A partition $\mathcal{I} = \{I_1,\ldots,I_k\}$ of the elements of P into common intervals is a congruence partition. The *quotient permutation* associated to $\mathcal{I}$, denoted $P_{|\mathcal{I}}$, is defined as follows:

(i precedes j in $P_{|\mathcal{I}}$) if and only if (I_i precedes I_j)

Example 5: For the permutation $P = (1\ 4\ 2\ 5\ 3\ 7\ 8\ 6\ 9)$ of Fig. 1, the partition $\mathcal{I} = \{\{1\},\{2,3,4,5\},\{7,8\}, \{6\},\{9\}\}$ is a congruence partition of P into intervals, and $P_{|\mathcal{I}} = (1\ 2\ 4\ 3\ 5)$.

As shown below, a congruence partitions for a permutation P "inherits" common intervals from P. More formally:

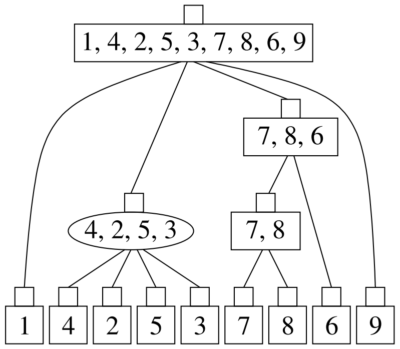

Fig. 1. The strong interval tree $T_s(P)$ of the permutation $P = (1\ 4\ 2\ 5\ 3\ 7\ 8\ 6\ 9)$. Prime and linear vertices (described later in this section) are distinguished by their shape. There are two non-trivial linear vertices, the rectangular vertices: $(7, 8)$ is increasing and $(7, 8, 6)$ is decreasing. There is only one prime vertex, the round vertex $(4, 2, 3, 5)$.

Lemma 1: Let $\mathcal{I} = \{I_1, \ldots, I_k\}$ be a congruence partition of a permutation P. Then $J = \{j, \ldots, h\}$ is a common interval of the quotient partition $P_{|\mathcal{I}}$ if and only if $K = \bigcup_{j \le i \le h} I_i$ is a common interval of P.

Proof. $\Rightarrow$ Assume J is a common interval of $P_{|\mathcal{I}}$. As $\mathcal{I}$ is a partition, each element of P belongs to a unique interval of $\mathcal{I}$. Let $x \notin K$ and I_ℓ be the common interval of $\mathcal{I}$ containing x (notice that $\ell \notin J$). Assume by contradiction the existence of $y, y' \in K$ such that $y < x < y'$. Then I_h and $I_{h'}$, the common intervals of $\mathcal{I}$ containing respectively y and y' are distinct (notice also that $h, h' \in J$). It follows that, in the quotient permutation $P_{|\mathcal{I}}$, $h < \ell < h'$, which contradicts the assumption that J is a common interval of $P_{|\mathcal{I}}$.

$\Leftarrow$ Assume J is not a common interval of P. Then there exists $\ell \notin J$ and $h, h' \in J$ such that $h < \ell < h'$. Thereby for any $y \in I_h$ and $y' \in I_{h'}$ (which both belong to K), there exists $x \in I_\ell$ (i.e. $x \notin K$) such that $y < x < y'$. It follows that K is not a common interval of P. □

The following decomposition theorem shows the importance of the congruence partition whose common intervals are the maximal strong intervals. For permutation $P = (1\ 4\ 2\ 5\ 3\ 7\ 8\ 6\ 9)$, this congruence partition is $\mathcal{J} = \{\{1\}, \{2, 3, 4, 5\}, \{6, 7, 8\}, \{9\}\}$.

Theorem 1: Let P be a permutation on n elements and $\mathcal{I} = \{I_1, \ldots, I_k\}$ be the partition of P into maximal strong intervals of P other than P itself. Then:

1) either any set of consecutive elements in $P_{|\mathcal{I}}$ is a common interval of $P_{|\mathcal{I}}$;
2) or the only common intervals of $P_{|\mathcal{I}}$ are trivial.

Moreover in case 1), either $P_{|\mathcal{I}} = Id_k$ or $P_{|\mathcal{I}} = \overline{Id_k}$. Thereby we say that $P_{|\mathcal{I}}$ is *linear* if it satisfies case 1), and *prime* otherwise.

Proof. See the proof of Theorem 5. □

Using Lemma 1, the above Theorem applied on the strong intervals enables us to show that the strong interval tree is a compact representation – it only requires $O(n)$ space – of the set of all common intervals, which is possibly a set of quadratic size.

Proposition 1: An interval I of a signed permutation P is a common interval if and only if it is either a vertex of $T_s(P)$, or the union of consecutive children of a linear vertex of $T_s(P)$.

Proof. Let I be a common interval of P which is not strong. By definition of strong intervals, there exists a smallest strong interval J that contains I, and I commutes with all children of J, which proves that I is the union of a subset of the children of J. These children have to be consecutive because I is an interval of P. Finally, it follows from Theorem 1 that J has to be linear. Indeed, if J is prime, point 2 of Theorem 1 implies that any non-singleton subset of children of I is not a common interval of P.

The converse is a direct consequence of Theorem 1 and Lemma 1. □

Hence, Theorem 1 induces a classification of the vertices of the strong interval tree $T_s(P)$ that is central in our algorithms: let P_I be the quotient permutation defined by the children of an internal vertex I of $T_s(P)$. The vertex I, or equivalently the strong interval I of P, is either:

1) *Increasing linear*, if P_I is the identity permutation, or
2) *Decreasing linear*, if P_I is the reverse of the identity permutation, or
3) *Prime*, otherwise.

For example, in Fig. 1, the rectangular vertices are the linear vertices, and the round vertex $(4, 2, 5, 3)$ is the unique prime vertex. The only decreasing linear vertex in this tree is $(7, 8, 6)$.

This representation for strong intervals was first given implicitly in [22], and explicitly in [25], where it was shown that $T_s(P)$ can be related to a data structure widely used in graph theory, called PQ-tree. It can be computed in $O(n)$ time using algorithms described in [7], [22], [25]. A formal link between PQ-trees and conserved structures in signed permutations with application to comparative genomics was first proposed in [5], in the context of conserved intervals, a subset of common intervals.

IV. Computing perfect scenarios.

We now describe efficient algorithms to compute parsimonious perfect scenarios for large classes of signed permutations. The crux is the use of the strong interval tree as a guide (we assume it is given, and refer to [7], [12] for simple algorithms building this tree.) Indeed, we obtain a characterization of perfect scenarios of a signed permutation P in terms of $T_s(P)$:

Proposition 2: A scenario S for a permutation P is perfect if and only if each of the reversals of S is either a vertex of $T_s(P)$, or the union of children of a prime vertex of $T_s(P)$.

Proof. Suppose that S is a scenario for permutation P, and that every reversal of S is either a vertex of $T_s(P)$, or the union of children of a prime vertex of $T_s(P)$. Let I be a reversal of S. If I is a vertex of $T_s(P)$, I is a strong interval and then commutes with every common interval of P. Now, assume that I is the union of children of a prime vertex J. I obviously commutes with any common interval not contained in J, and with any common interval contained in a child of J. Hence it remains to show that I commutes with any common interval that is union of children of J, but there are none by definition of a prime vertex. It follows that I commutes with every common interval of P, and then S is a perfect scenario.

Conversely, suppose that S is a perfect scenario, let I be a reversal of S, and consider the partition $I_1, I_2, \ldots, I_k$ of I in which the part containing an element x of I is the largest strong interval included in I and that contains x. If $k \geq 2$, then $I_1, I_2, \ldots I_k$ must all be children of a same parent J in $T_s(P)$, otherwise I would not commute with the vertices of $T_s(P)$ that are parents of I_j's. If J is a linear vertex, then I must be equal to J, otherwise I would overlap an interval formed by a leftover child of J and one of the intervals of I, and such an interval is a common interval of P by the points 1 and 2 of Theorem 1. Therefore, either $k = 1$ and I is a vertex of $T_s(P)$, or $k > 1$ and the vertex J must be prime. □

Computing a perfect scenario S thus amounts to identify leaves, linear vertices and union of children of prime vertices of $T_s(P)$ that are the reversals of S. In the remaining of this section, we show that, even if the general problem of computing parsimonious perfect scenarios was claimed to be difficult [17], it can be done efficiently for a large class of signed permutations, defined in terms of the structure of their strong interval tree, as defined below.

Definition 7: A strong interval tree $T_s(P)$ is *unambiguous* if every prime vertex has a linear parent, and *ambiguous* otherwise. If $T_s(P)$ has no prime vertices, it is *definite*. Note that a definite tree is unambiguous. *ambiguous*.

To identify reversals belonging to parsimonious perfect scenarios, we give a *sign* to vertices of $T_s(P)$.

Definition 8: A *signed* tree is a strong interval tree $T_s(P)$ in which we associate a sign, $+$ or $-$, to the vertices according to the following rules:

1) the sign of a leaf x is the sign of the corresponding element in P;
2) the sign of a linear vertex is $+$, if the vertex is increasing, and $-$ if the vertex is decreasing;
3) the sign of a prime vertex inherits the sign of its parent if this latter vertex is linear.

Note that these rules can leave some vertices with no sign. Fig. 2, 3 and 4 show signed strong interval trees associated with the permutations obtained by comparing 16 synteny blocks of the human, mouse and rat X chromosomes [18][1]. In Fig. 2 the labels of the vertices are given with respect to the order of the blocks of the mouse chromosome.

$$
\begin{aligned}
\text{Human} &= \;1\;2\;3\;4\;5\;6\;7\;8\;9\;10\;11\;12\;13\;14\;15\;16\\
\text{Mouse} &= \;\bar{6}\;\bar{5}\;4\;13\;14\;\overline{15}\;16\;1\;\bar{3}\;9\;\overline{10}\;11\;12\;\bar{7}\;8\;\bar{2}\\
\text{Rat} &= \;\overline{13}\;\bar{4}\;5\;\bar{6}\;\overline{12}\;\bar{8}\;\bar{7}\;2\;1\;\bar{3}\;9\;10\;11\;14\;\overline{15}\;16
\end{aligned}
$$

A. Computing perfect scenarios with unambiguous trees.

If a tree $T_s(P)$ is unambiguous, due to the definition of unambiguous trees and the constraints imposed on signs of vertices, there is a unique way to affect signs to all the vertices of $T_s(P)$. Next Lemma identifies reversals that must belong to any perfect scenario, thus to any parsimonious perfect scenario. It applies to all trees, definite, unambiguous and ambiguous.

Lemma 2 (The Parity Lemma): Let I be a vertex of the tree $T_s(P)$ of a signed permutation P. If I has a linear parent and a sign different from the sign of its parent, then I belongs to any perfect scenario for P.

Proof. Let S be a perfect scenario, and I be a vertex with negative sign, whose linear parent J has a positive sign, and such that $I \notin S$. Notice that I can not be prime. Since J is linear and $I \notin S$, by Proposition 2, any reversal of S that contains I also contains J.

Let m be the number of reversals of S containing J. If m is even, as J has a positive sign, S sorts P to Id, by definition of increasing vertices. If I is a leaf, it will still be negative after an even number of reversals, contradicting the fact that S sorts P to Id. If I is a linear vertex with a negative sign (it is decreasing by Definition 8), then its first child is greater than its last, and will still be after an even number of reversals, again contradicting that S sorts P to Id. A symmetric argument holds if m is odd, or if I has a positive sign, and J a negative sign. □

For definite trees, the parity Lemma yields the following theorem. We will study the case of permutations whose strong interval tree is definite in more details in Section V.

Theorem 2: Let P be a signed permutation. If $T_s(P)$ is definite, then the set of vertices that have a sign different from the sign of their parent is a parsimonious perfect scenario for P. Moreover, no other reversal than these vertices belongs to a parsimonious perfect scenario for P.

Proof. If $T_s(P)$ is definite, Proposition 2 implies that a parsimonious perfect scenario S consists of a set of vertices of $T_s(P)$. The Parity Lemma leads to the fact that every parsimonious perfect scenario is composed of the vertices of $T_s(P)$ that have a sign different from their parent. □

Given the tree $T_s(P)$, Theorem 2 implies that computing a parsimonious perfect scenario for P is almost immediate, when $T_s(P)$ is definite. The comparison of the rat and mouse

[1]The positions of blocks 5 and 6 in our data differs from [18], following a correction of the mouse genome assembly.

X chromosomes yields a definite tree, Fig. 2, and the corresponding scenario can be obtained by comparing the signs of the $O(n)$ vertices. When such a scenario exists, it is unique up to the order of the reversals, since each of them commutes with all the others.

Corollary 1: Let P be a signed permutation on n elements. If $T_s(P)$ is definite, then computing a parsimonious perfect scenario for P can be done in $O(n)$ time.

Proof. Computing $T_s(P)$ can be done in $O(n)$ time [7], and it follows from Theorem 2 that the reversal of P can be computed by a single traversal of $T_s(P)$ that gives signs to its vertices. □

We next turn to the more general case of unambiguous trees. Recall that a prime vertex inherits its sign from its parent, and that any reversal that is a union of children of a prime vertex commutes with all common intervals, thus may belong to a perfect scenario. Algorithm 1 describes how to obtain a parsimonious perfect scenario in the case of unambiguous trees. The basic idea is to compute, for each prime vertex I of the tree, any parsimonious scenario that sorts the children of vertex I in increasing or decreasing order, depending on the sign of I. Then, it suffices to deal with linear vertices whose parent is linear in the same way than for a definite tree.

Algorithm 1: Computing a parsimonious perfect scenario for unambiguous $T_s(P)$

```
S is an empty scenario.
For each prime vertex $I$ of $T_s(P)$
    $P_I$ is the quotient permutation of $I$ over its children
    If the sign of $I$ is positive Then
        compute a parsimonious scenario $T$ from $P_I$ to $Id$
    Else compute a parsimonious scenario $T$ from $P_I$ to $\overline{Id}$
    Deduce the corresponding scenario $T'$ on the children of $P_I$
    Add the reversals of $T'$ to scenario $S$
End for each
Add to $S$ the linear vertices and leaves having a linear parent
and a sign different from the sign of their parent.
```

Fig. 3 shows the signed tree associated to the permutations of the human and rat X chromosomes. This tree is unambiguous: it has one prime vertex $(4, 5, 6, 12, 8, 7, 2, 1, 3, 9, 10, 11)$ whose parent is a decreasing linear vertex. The quotient permutation of this vertex over its five children is $P_I = (2\ \overline{5}\ \overline{3}\ 1\ 4)$, and a parsimonious scenario that sorts P_I to $\overline{Id}$ is given by: $\{1, 3, 4\}$, $\{1, 3\}$, $\{1\}$, $\{2, 3, 4, 5\}$, $\{3, 4, 5\}$. Note that if the corresponding five reversals are applied to the rat chromosome, the resulting permutation has a definite tree.

The time complexity of Algorithm 1 depends on the time complexity of the sorting by reversals algorithm used to compute a reversal scenario that sorts the children of a prime vertex. Using the $O(n\sqrt{n\log(n)})$ algorithm described in [38], we have:

Theorem 3: Let P be a signed permutation on n elements. If $T_s(P)$ is unambiguous, Algorithm 1 computes a parsimonious perfect scenario for P in subquadratic $O(n\sqrt{n\log(n)})$ time.

Proof. The time complexity bound is obtained by observing that, first computing $T_s(P)$ requires $O(n)$ time [7], and second that the sorting by reversals procedure will be applied on permutations of size $n_1, \ldots, n_k$, the number of children of the k prime vertices of $T_s(P)$, with $n_1 + \ldots + n_k \leq n$, and the best current sorting by reversals algorithm running in $O(n\sqrt{n\log(n)})$ time [38].

We now prove that the sequence of reversals that is computed is a scenario for P. First, it is immediate that the quotient permutation of a prime vertex has to be sorted into Id or $\overline{Id}$, according to its sign. This is done during the first phase of the algorithm, that deals with prime vertices. The argument for the second phase – leaves and linear vertices whose parent is linear – is similar to the one used in the proof of Theorem 2.

Finally we prove that the computed scenario is parsimonious among perfect scenarios. Given a parsimonious perfect scenario, the subsequence of reversals that are contained in a prime vertex I sorts I in increasing or decreasing order. Suppose that S' is a parsimonious scenario shorter than the scenario S produced by Algorithm 1. Then there is at least one prime vertex I such that the number of reversals of S' that are contained in I is less than the number of reversals of S that are contained in I. Let R and R' be the subsequences of reversals of S and S' that are contained in I. One of R and R' sorts I in increasing order, and the other in decreasing order. Otherwise, one of them would not be parsimonious. Suppose that the sign of the parent of I is positive, then R sorts I in increasing order, and R' sorts I in decreasing order. By the same argument that was used in the Parity Lemma, I must belong to S', and thus to R', and removing I from S' produces a shorter scenario, contradicting the hypothesis that S' was parsimonious. A similar argument holds if the sign of the parent of I is negative. □

Remark 2:
1) Any improvement on the complexity of sorting by reversals will immediately leads to a similar improvement in the complexity of Algorithm 1.
2) Note that Algorithm 1 can be easily modified in order to compute the length of a parsimonious perfect scenario – the *perfect distance* – in $O(n)$ time, as computing the reversal distance requires $O(n)$ time [8].

Remark 3: Algorithm 1 and its variant computing the perfect distance, as well as the linear time algorithm computing the strong interval tree of a signed permutation, are available on-line on the web site of the CGL, at the address `http://cgl.bioinfo.uqam.ca/`.

B. Computing perfect scenarios with ambiguous trees.

When $T_s(P)$ is ambiguous, the sign of some prime vertices is undefined, but one can then apply the following brute-force algorithm to sort P. Such an algorithm is a generalization of the algorithms we described for unambiguous trees and was described using another formalism in [17]. It has a worst-case time complexity that is exponential in the number of prime vertices whose parent is prime, and thus is efficient if the number of such edges is small.

As an example, consider Fig. 4 that shows the signed tree associated to the permutations of the human and mouse X

$$
\begin{array}{rcccccccccccccccc}
\text{Mouse} = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\text{Rat} = & \bar{4} & \bar{3} & \bar{2} & 1 & \bar{1}3 & \bar{1}5 & 14 & \bar{1}6 & 8 & 9 & 10 & \bar{1}1 & 12 & 5 & 6 & 7
\end{array}
$$

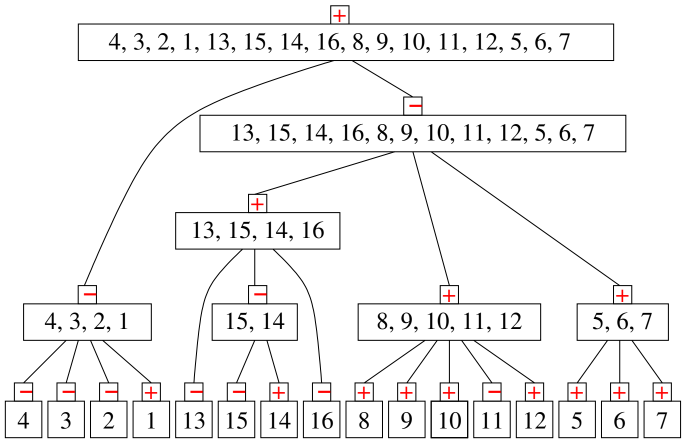

Fig. 2. Comparing the rat and mouse X chromosomes: the set of vertices that have a sign different from the sign of their parent form a parsimonious perfect scenario that transforms the rat X chromosome into the mouse X chromosome in 11 reversals: $(4, 3, 2, 1)$, (1), $(13, 15, 14, 16, 8, 9, 10, 11, 12, 5, 6, 7)$, $(13, 15, 14, 16)$, (13), $(15, 14)$, (14), (16), $(8, 9, 10, 11, 12)$, (11), $(5, 6, 7)$.

$$
\begin{array}{rcccccccccccccccc}
\text{Human} = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\text{Rat} = & \bar{1}3 & \bar{4} & 5 & \bar{6} & \bar{1}2 & \bar{8} & \bar{7} & 2 & 1 & \bar{3} & 9 & 10 & 11 & 14 & \bar{1}5 & 16
\end{array}
$$

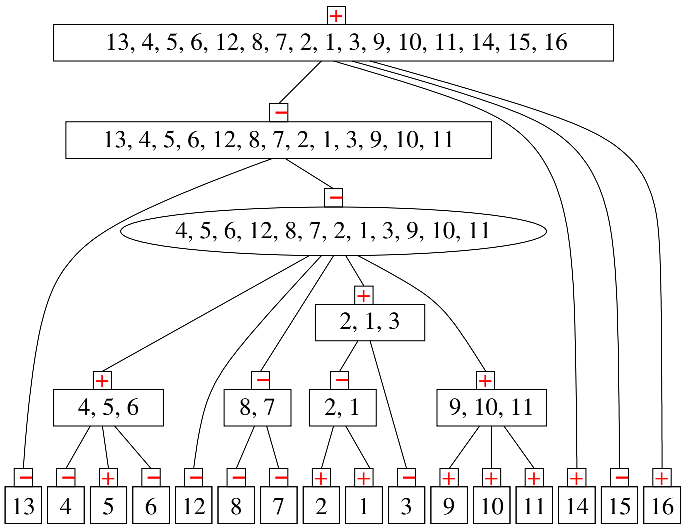

Fig. 3. Comparing the human and rat X chromosomes: a parsimonious perfect scenario is obtained by sorting the five children $(4, 5, 6)$, (12), $(8, 7)$, $(2, 1, 3)$ and $(9, 10, 11)$ in decreasing order using any parsimonious scenario that sorts the quotient permutation $P_I = (2\ \bar{5}\ \bar{3}\ 1\ 4)$, and then reversing the linear vertices and leaves whose linear parent have a different sign: $(13, 4, 5, 6, 12, 8, 7, 2, 1, 3, 9, 10, 11)$, (4), (6), $(2, 1)$, (2), (1), (3), (15). The length of the scenario is 13.

Algorithm 2: Computing a parsimonious perfect scenario for ambiguous $T_s(P)$

```
Let $I_1, \dots I_k$ be the vertices of $T_S(P)$ whose sign is undefined.
For every binary word $W$ of length $k$ do
    Give to every unsigned vertex $I_j$ the sign $+$ if $W[j] = 1$
    or the sign $-$ if $W[j] = 0$.
    Apply Algorithm 1 on the resulting signed tree.
End for every
Return a parsimonious scenario among the resulting set of $2^k$
perfect scenarios.
```

chromosomes. This tree is ambiguous since its root is a prime vertex, and we must try to sort this vertex both to Id and to $\overline{Id}$. In this case, both parsimonious scenarios have the same length.

Theorem 4: Let P be a signed permutation on n elements. If $T_s(P)$ is ambiguous with k unsigned vertices, Algorithm 2 computes a parsimonious perfect scenario for P in $O(2^k \times n\sqrt{n \log(n)})$ time.

Proof. The parity Lemma and a proof similar than for Theorem 3 prove the statement. $\square$

Human =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Mouse =	$\overline{6}$	$\overline{5}$	4	13	14	$\overline{15}$	16	1	$\overline{3}$	9	$\overline{10}$	11	12	$\overline{7}$	8	$\overline{2}$

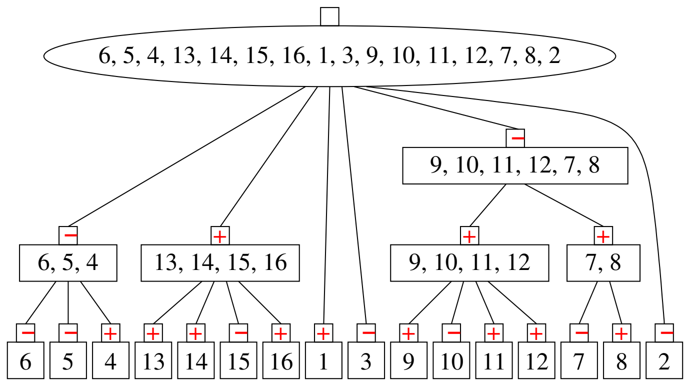

Fig. 4. Comparing the human and mouse X chromosomes: the root has no sign but its children can be sorted to Id or $\overline{Id}$ in 6 reversals using a parsimonious scenario that sorts the quotient permutation $P_I = (\overline{4}\ 6\ 1\ \overline{3}\ \overline{5}\ 2)$. A parsimonious perfect scenario would also contain the reversals: (4), (15), $(9, 10, 11, 12)$, (10), $(7, 8)$, (7). The total length of the scenario is 12.

Constructing permutations that are hard to perfectly sort, that is whose strong interval tree is ambiguous, requires to break almost any structure in a given permutation. The smallest example of a hard to sort permutation is given in Fig. 5.

C. Computing perfect scenarios for a subset of common intervals.

From a practical point of view, it is worth to recall that the interest in computing scenarios that do not break common intervals relies on the assumption that genes, or other genomic markers, cluster in such groups for functional reasons, like co-transcription for example. However, it is possible that clusters of genomic markers appear by "chance" in the data, or are not supported by any functional evidence, and it would then not be relevant to impose that such intervals should not be broken during an evolution scenario, which leads to the following generalization of the problem we addressed until now: Given a permutation P of length n, its set $\mathcal{C}$ of common intervals and a subset $\mathcal{F} \subseteq \mathcal{C}$, find a scenario S for P that does not break any interval from $\mathcal{F}$ and is parsimonious among such scenarios. We call such a scenario a *parsimonious perfect scenario for P with regard to $\mathcal{F}$*. We say that S *respects* the intervals of $\mathcal{F}$.

In this section, we show that the algorithms presented in Sections IV-A and IV-B can be applied to solve this problem without any modification. To that aim, we show that, given a set of common intervals, which are believed to be pertinent from the biological point of view, an interval tree can be constructed. Moreover this tree has the same properties than the strong interval tree. For the paper to be self-contained we propose a proof of this result that generalizes Theorem 1. But as already mentioned in Section III, it is a special case of known graph theoretical results (see Appendix).

Definition 9: Let $\mathcal{F}$ be a set of common intervals of a signed permutation P. The *closure $\mathcal{F}^*$ of $\mathcal{F}$* is the smallest set of common intervals of P that contains $\mathcal{F}$, all trivial common intervals of P and such that for any $I_1 \in \mathcal{F}^*$ and $I_2 \in \mathcal{F}^*$ if I_1 and I_2 overlap, then $I_1 \cup I_2$, $I_1 \cap I_2$, $I_1 \backslash I_2$ and $I_2 \backslash I_1$ belong to $\mathcal{F}^*$.

A simple brute-force algorithm computes this closure in polynomial time. It is worth to note that the family $\mathcal{C}$ of all common intervals of a signed permutation is closed ($\mathcal{C} = \mathcal{C}^*$).

Lemma 3: Let P be a signed permutation and $\mathcal{F}$ a set of common intervals of P. A scenario for P is a perfect scenario with regard to $\mathcal{F}$ if and only if is a perfect scenario with regard to $\mathcal{F}^*$.

Proof. If a scenario S for P is a parsimonious perfect scenario with regard to $\mathcal{F}$, then for each pair of overlapping intervals S respects, say I and J, S respects also $I \cup J$, $I \cap J$, $I \backslash J$ and $J \backslash I$. Therefore, S is also a parsimonious perfect scenario with regard to $\mathcal{F}^*$. As $\mathcal{F} \subseteq \mathcal{F}^*$, the converse is true. □

From now on, we consider the set $\mathcal{F}^*$ and we are interested in computing a parsimonious perfect scenario for P with regard to $\mathcal{F}^*$. First, we can notice that the notion of strong intervals can be used with $\mathcal{F}^*$: an interval of $\mathcal{F}^*$ is strong with regard to $\mathcal{F}^*$ if it does not overlap with any other interval of $\mathcal{F}^*$. It follows immediately that we can define (as we did for $T_s(P)$) an inclusion tree of the strong intervals of $\mathcal{F}^*$, denoted by $T_s^{\mathcal{F}^*}(P)$, that, when $\mathcal{F}^* = \mathcal{C}$, turns out to be the strong interval tree of P. An example of such a tree is depicted in Figure 6.

We now show that the tree $T_s^{\mathcal{F}^*}(P)$, for a strict subset $\mathcal{F}^*$ of the common intervals of a signed permutation P, has a structure similar to the strong interval tree of P in terms of prime and linear vertices. The following result is a generalization of Theorem 1.

Theorem 5: Let P be a permutation on n elements and $\mathcal{I} = \{I_1, \ldots I_k\}$ be the congruence partition of P into maximal strong intervals of $\mathcal{F}^*$other than P itself. Then exactly one of the following is true:

1) either every union of consecutive elements $I = \{i, \ldots j\}$ of $P_{|\mathcal{I}}$ is a common interval of $P_{|\mathcal{I}}$ and $K = \bigcup_{i \le h \le j} I_j$

$$\begin{array}{rccccccc} \text{Identity} = & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \text{P} = & 2 & \overline{5} & 7 & 4 & \overline{6} & 1 & 3 \end{array}$$

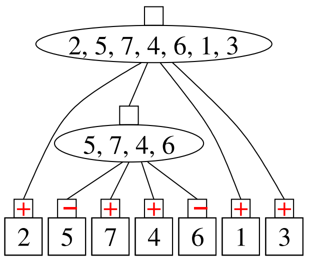

Fig. 5. A hard to sort permutation: if both vertices are sorted in increasing order, or both are sorted in decreasing order, then the resulting perfect scenarios are not parsimonious.

$$\begin{array}{rcccccccccccccccc} \text{Mouse} = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ \text{Rat} = & \overline{4} & \overline{3} & \overline{2} & 1 & \overline{13} & \overline{15} & 14 & \overline{16} & 8 & 9 & 10 & \overline{11} & 12 & 5 & 6 & 7 \end{array}$$

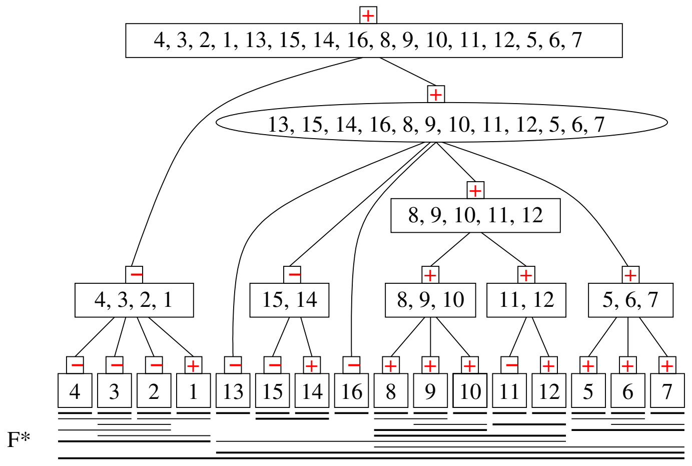

Fig. 6. The strong interval tree obtained when comparing the rat and mouse X chromosomes with a subset of their common intervals (intervals $(13, 14, 15)$, $(14, 15, 16)$, $(13, 14, 15, 16)$, $(10, 11)$, $(9, 10, 11)$, $(10, 11, 12)$, $(8, 9, 10, 11)$ and $(9, 10, 11, 12)$ were removed from the set of common intervals).

belongs to $\mathcal{F}^*$ (moreover either $P_{|\mathcal{I}} = Id_k$ or $P_{|\mathcal{I}} = \overline{Id_k}$);

2) or no union of intervals of $\mathcal{I}$ belongs to $\mathcal{F}^*$.

Proof. By induction on the size k of $P_{|\mathcal{I}}$. The cases $k = 1, 2$ and 3 are easy to check. Assume it is true for any value less than k.

If $P_{|\mathcal{I}}$ has a non-trivial common interval $I = \{i, \dots h\}$, then by Lemma 1, $K = \bigcup_{i \le h \le j} I_j$ is a common interval of P. By definition of $\mathcal{I}$, K cannot be a strong interval of $\mathcal{F}^*$. So either it does not belong to $\mathcal{F}^*$, or it is not strong for $\mathcal{F}^*$. Assume K belongs to $\mathcal{F}^*$ (is not strong) and that I has been chosen maximal for such K.

Therefore there is $K' \in \mathcal{F}^*$ overlapping K. Moreover K' can be chosen as the union of a set I' of intervals of $\mathcal{I}$. Let us consider K' (and thus I') maximal. As K and K' are maximal and as $\mathcal{F}^*$ is closed, $K \cup K' = \{1, \dots n\}$ and thus $I \cup I' = \{1, \dots k\}$. Moreover $I \setminus I'$ (resp. $I' \setminus I$) is a singleton. Otherwise $K \setminus K'$ (resp. $K' \setminus K$) would be a common interval of $\mathcal{F}^*$, union of at least two intervals of $\mathcal{I}$. By definition of $\mathcal{I}$ it cannot be a strong interval of $\mathcal{F}^*$. As $K \cup K' = \{1, \dots n\}$, we could therefore find a common interval of $P_{|\mathcal{I}}$ overlapping both $K \setminus K'$ and K' (resp. $K' \setminus K$ and K). But as $\mathcal{F}^*$ is closed, it would contradict the maximality of K' (resp. K).

Now as $I \setminus I'$ and $I' \setminus I$ are singletons, as $I \cap I'$ is a common interval of $P_{|\mathcal{I}}$, we have $I \cap I' = \{2, \dots k-1\}$ and $K \cap K' = \bigcup_{2 \le j \le k-1} I_j$. So the induction can be applied on the sub-permutation P' of P induced by the elements of $\{1 \dots n\} \setminus I_1$ for which the congruence partition into maximal non-trivial strong interval of $\mathcal{F}^*$ is $\mathcal{I}' = \{I_2 \dots I_k\}$. As $K \cap K'$ belongs to $\mathcal{F}^*$, $P'_{|\mathcal{I}'}$ satisfies case 1). To end the proof we need to show that $I_1 \cup I_2$ belongs to $\mathcal{F}^*$. The opposite would mean that the union of I_2 and the intervals that follow I_2 in P' is not an interval belonging to $\mathcal{F}^*$: contradiction.

Notice that pushing further the induction proof would show that in case 1) either $P_{|\mathcal{I}} = Id_k$ or $P_{|\mathcal{I}} = \overline{Id_k}$. This is a consequence of the fact that $\{1, 2\}$, $\{2 \dots k-1\}$ are intervals of $P_{|\mathcal{I}}$. $\square$

From this result, we can deduce immediately that the main concepts related to the strong interval tree we used in Sections IV-A and IV-B can be used with $T_s^{\mathcal{F}^*}(P)$, in particular (1) the notions of linear (case 1 of Theorem 5) vertices, increasing and decreasing, and prime (case 2 of Theorem 5) vertices, (2) the notions of definite, unambiguous and ambiguous strong interval tree and (3) the definition of the sign of a vertex of the strong interval tree (Definition 8). The only difference is that the alternation of increasing and decreasing linear vertices does not hold anymore. Moreover proofs of the Parity Lemma and Theorems 2 and 3 do not require that the set of considered common intervals is the set of all common intervals of the considered signed permutation P. It follows immediately, that, if we denote by Algorithm 3 the algorithm obtained by replacing $T_S(P)$ by $T_s^{\mathcal{F}^*}(P)$ in Algorithm 2, the following result, generalizes Theorems 2, 3 and 4.

Theorem 6: Let P be a signed permutation of n and $\mathcal{F}$ a set of common intervals of P. Given $T_s^{\mathcal{F}^*}(P)$, Algorithm 3 computes a parsimonious perfect scenario with regard to $\mathcal{F}$ in

- $O(n)$ time if $T_s^{\mathcal{F}^*}(P)$ is definite;
- $O(n\sqrt{n\log(n)})$ time if $T_s^{\mathcal{F}^*}(P)$ is unambiguous;
- $O(2^k \times n\sqrt{n\log(n)})$ if $T_s^{\mathcal{F}^*}(P)$ is ambiguous with k unsigned vertices.

V. Commuting permutations and scenarios.

We now consider a class of scenarios, namely *commuting scenarios*, that were introduced in the framework of parsimonious scenarios in [4]. We show that the class of commuting scenarios is a remarkable class of perfect scenarios, as they correspond exactly to the perfect scenarios for signed permutations whose strong interval tree is definite (Theorem 7).

Definition 10: A scenario S for a signed permutation P is said to be *commuting* if every pair of reversals of S commutes.

A remarkable feature of commuting scenarios is that the order of reversals in such scenarios does not matter. Indeed, given a commuting scenario S for a signed permutation P, it follows immediately from Definition 10 that any sequence of reversals that is a reordering of the reversals of S is a scenario for P.

Proposition 3: A commuting scenario S for a signed permutation P is a perfect scenario.

Proof. In order to prove the statement, we will prove that every common interval I of P commutes with all reversals of S.

Since S is a commuting scenario, one can, without loss of generality, consider that S begins with all the reversals that commute with I. Applying these reversals leads to a permutation P' such that I is a common interval of P'. Let S' be the set of remaining reversals, those that do not commute with I. As S is a scenario for P, S' is a scenario for P'.

If all reversals in S' are disjoint, there are at most two of them: one overlaps I on its right extremity and one overlaps I on its left extremity. Then, as I is a common interval of P', applying them to P' leads to a signed permutation that is not the identity permutation, neither the reversed identity. This contradicts the fact that S' is a scenario for P'.

Suppose that S' contains at least two non-disjoint intervals that overlap I on the same extremity, say its right extremity – the argument is completely symmetrical if we consider the left extremity. Let J be the largest of the intervals that intersects I at its right extremity, J' the second largest, and I' the non-empty set of elements of I that do not belong to J. Since S is a commuting scenario, J' is strictly contained in J, and all other reversals of S' are either disjoint of J and J', or included in J'. Therefore, J can be partitioned into three disjoint intervals, J_1, J' and J_2, such that J_1 is contained in I, J_2 is disjoint from I, and at least one of J_1 and J_2 is non-empty. Applying both J and J' to P' results in a signed permutation where the elements of J_1 are to the right of the elements of J' that are themselves to the right of the elements of J_2, that are at the right of the elements of I'. Let P'' be the resulting permutation.
$P' = \dots\ I'\ J_1\ J'\ J_2\ \dots\ \longrightarrow P'' = \dots\ I'\ \overline{J_2}\ J'\ \overline{J_1}\ \dots$

By the choice of J and J' as the maximal intervals overlapping the right extremity of I, this structure will remain unchanged when applying all other reversals of S' – recall that none of them contains I by definition of S' –. If J_1 is not empty, the elements of J_2 that are not in I will end up between I' and J_1, while if J_2 is not empty, the elements of J_1 will end up between I' and $J' \cap I$. This results in a signed permutation Q such that I is not a common interval of Q, and then Q can not be the identity of the reversed identity. This contradicts the fact that S is a scenario for P and completes the proof. □

Lemma 4: Every reversal of a commuting scenario S for a signed permutation P is a common interval of P.

Proof. It follows from the non existence of overlapping intervals in commuting scenarios. □

Definition 11: A signed permutation P that can be sorted by a commuting scenario is said to be a *commuting permutation*.

We now state the main result of this section: a characterization, in terms of the structure of their strong interval tree, of the class of commuting permutations.

Theorem 7: A signed permutation P is commuting if and only if its strong interval tree $T_s(P)$ is definite.

Proof. First, if $T_s(P)$ is definite, then it follows immediately from Theorem 2 that every parsimonious perfect scenario for P is commuting. Indeed, the reversals of such scenarios are the vertices of $T_s(P)$, which implies that every pair of reversals commute.

Now, suppose that P is commuting, and let S be a commuting scenario for P. As S is a commuting scenario, it is a perfect scenario (Proposition 3), and then every reversal of S is either a linear vertex of $T_s(P)$ or a set of children of a prime vertex of $T_s(P)$ (Proposition 2). Moreover, if there is a reversal I of S that is a set of children of a prime vertex, and I is different of this vertex, then I is not a common interval of P, by definition of prime vertices, which contradicts Lemma 4. Hence, if $T_s(P)$ contains a prime vertex, no reversal of S can be a strict subset of the children of this vertex, which contradicts immediately the fact that S is a scenario for P. It follows that $T_s(P)$ does not have prime vertices. □

Corollary 2: Let P be a commuting permutation. Then all perfect scenarios for P are commuting, are parsimonious perfect scenarios, and have the same set of reversals, namely the vertices of $T_s(P)$ that have a sign different from their parent.

Proof. Direct consequence of Proposition 2 and of Theorems 2 and 7. □

We now conclude this section with a few remarks about commuting scenarios and permutations. First, note that these notions were introduced in a more restrictive framework, that is commuting scenarios that are parsimonious, and with no reference to the strong interval tree, in [4]. In particular, the proof of Proposition 3 is an immediate extension of a similar result in [4]. Next, from a combinatorial and algorithmical point of view, the class of commuting permutations is remarkable for several reasons, the main ones being stated in Corollary 2. Moreover this is, as far as we know, the largest non trivial class of signed permutations that can be sorted (1) in linear time, as it only requires to construct the strong interval tree and to perform a single pass on this tree, and (2) without having to rely on the classical Hannenhalli-Pevzner theory for sorting by reversals [8], [21]. Finally, the fact that the order of reversals of commuting scenarios does not matter is a striking feature, especially as commuting permutations have appeared in the analysis of real datasets, as for example the comparison of the mouse and rat X chromosomes described in Section IV, but also in a comparison of human chromosome 16 and mouse chromosome 11 [4], [32].

VI. Concluding remarks and future work

Summary of results: We described in this paper a combinatorial and algorithmical framework for computing perfect scenarios, that leads to efficient algorithms for large classes of signed permutations. From the algorithmic point of view, the central aspect of our work is the link between the computation of perfect scenarios and the strong interval tree of a signed permutation, which can be seen as similar, for the combinatorial criterion of perfection, of what the overlap graph of Hannenhalli-Pevzner theory is for the criterion of parsimony. We also introduced the classes of commuting permutations and scenarios that has remarkable combinatorial and algorithmical properties and deserves to be investigated in more details.

Finally, we think that the new insight we bring in this paper, and especially the introduction of the strong interval tree, opens the way to many interesting questions, that range from very combinatorial problems to applications of our algorithmical tools in the analysis of real datasets. We describe below some of these questions.

Exponential complexity of the general problem: Despite the existence of efficient algorithms for computing perfect scenarios for some classes of permutations, the best known algorithm for the general problem still runs in exponential time. However, using the strong interval tree, we are now able to identify the problematic structures that lead to an exponential behavior, namely the prime vertices whose parent is prime. We conjecture that the perfect sorting by reversals problem is Fixed Parametrized Tractable if we choose as parameter the *maximal prime degree of prime vertices* (the maximal number of prime vertices children of a same prime vertex). It would mean the complexity to handle permutations for which this parameter is bounded is actually polynomial.

From a practical point of view, it is relevant to ask wether if in real datasets the above parameter is bounded or remains small. A preliminary study of several datasets of eukaryotic and prokaryotic genomes (results not shown) suggests a positive answer. Actually all the corresponding strong interval trees exhibit very few prime vertices whose parent is prime. However, if it turns that a prime vertex represents genomic markers that are functionally related, it is likely that the segment of the chromosome corresponding to this interval is framed in the genome, either upstream, or downstream, or even both, by regulatory sequences. It follows that if one considers these framing sequences in the signed permutation, they could prevent, in the strong interval tree, that the corresponding prime vertex has for parent a prime vertex. Recent works that explore the properties of the chromosomal regions between synteny blocks are of interest with regard to this problem [34], [40].

Combining parsimony and conservation of common intervals: In [34], it was shown that when, for a given signed permutation, there exists a parsimonious scenario that is also a perfect scenario, computing such a scenario can be done in polynomial time, extending a previous result of [4]. In the present work, one can, once a parsimonious perfect scenario has been computed, check whether this scenario is also parsimonious, using for example one of the linear time algorithms for computing the reversal distance proposed in [2], [8]. However, the computation of a parsimonious perfect scenario can ask for an exponential time depending on the strong interval tree. In order to close the gap between these two approaches in computing perfect scenarios, it would be interesting to characterize, in terms of strong interval trees, the class of signed permutations for which a parsimonious perfect scenario is also parsimonious among all possible scenarios. This problem raises interesting combinatorial questions about the links between the Hannenhalli-Pevzner theory and the strong interval tree.

Other applications of the strong interval tree in computing evolutionary scenarios: In the current study, we considered only reversals. However, several other evolutionary events should be considered in computing evolution scenarios, especially with multi-chromosomal genomes, such as translocations, transpositions or block interchanges [42]. It would also be interesting to see how the strong interval tree can be considered in the recent Bayesian approaches to rearrangements scenarios [26]. One could for example ask if perfect scenarios are more significant than parsimonious scenarios, or if reversals belonging to perfect scenarios are more likely to appear in significant scenarios. An interesting question related to this point would be to study the difference between reversals given by linear vertices and prime vertices. Since they have a stronger structure in terms of common intervals, could reversals corresponding to linear vertices be more significant ? It is also natural to consider evolution scenarios for more than two genomes and the median problem, that is one of the main tools used for this purpose. For example, it could be interesting to consider the reversals given by the strong interval tree as putative evolutionary events along the branches of a phylogenetic tree, in a similar way reversals given by the Hannenhalli-Pevzner theory are used in MGR [9].

Alternative applications of the strong interval tree in comparative genomics: We believe that the strong interval tree of a signed permutation is a very versatile tool for comparative genomics. Indeed, it is immediate to extend this notion to several genomes, which allows to exhibit conserved structures in all or part of a given set of genomes. This approach was used for example to determine putative ancestral genomes without computing evolution scenarios [1], [5], [13]. Another potential application would be the identification of so-called evolutionary hot-spots for rearrangements, that is genome segments that are likely to be involved in several rearrangements, a problem that has been the subject of an intense debate recently [31], [33], [36]. Indeed, we believe that the structure of the strong interval tree could be useful to pinpoint genomic regions at the border of conserved sets of genomic segments. It would be interesting to investigate the nature of these segments in the dataset used in [31] for example.

Acknowledgements. Research supported by the "60^{eme} Commission Permanente Franco-Québecoise de Coopération Scientifique", and by grants from NSERC, CNRS and INRA.

Appendix
Modular decomposition of graphs

In Section III, we mentioned links between common intervals of a pair of permutations and the modular decomposition of graphs. This appendix is a short presentation of this correspondence. For further results and complete presentation of the modular decomposition theory, the interested reader should refer to [29]. Let us first introduce the graph theoretical concepts and then establish the links with common intervals.

Definition 12: A *module* of a graph $G = (V, E)$ is a subset S of vertices such that any vertex $x \notin S$ is either adjacent to any vertex of S or to none of them.

Notice that any singleton vertex set, as well the whole vertex set of a graph are modules, namely the *trivial modules*. A graph whose modules are precisely the trivial modules is called a *prime graph*. Any vertex subset of the complete graph (similarly, of the stable graph[2]) is also a module. For these reasons, the complete graph and the stable graph are called *degenerated graphs* in the scope of modular decomposition theory. The last simple cases of modules of a graph are its connected components or the connected components of its complement.

A module M is *strong* if it overlaps no other module, that is any module $M' \neq M$ satisfies either $M \cap M' = \emptyset$ or $M \subset M'$ or $M' \subset M$. It follows that trivial modules are strong modules. The family of strong modules naturally defines an inclusion tree, called the *modular decomposition tree* (see Figure 7), which will be denoted $MD(G)$. Next Lemma states that the family of strong modules is a basis (or a generating family) of the set of modules of the graph.

[2]The complete graph has an edge between any pair of vertices while the stable graph has no edge at all.

Lemma 5: [29] Any module of a graph G is either a strong module or the union of strong modules all sons of a degenerated vertex of $MD(G)$.

Let $\mathcal{P} = \{M_1, \dots M_k\}$ be a partition of the vertex set of a graph G. If for any $1 \leqslant i \leqslant k$, M_i is a module of G, then $\mathcal{P}$ is called a *congruence partition*. Congruence partitions play an important role in the modular decomposition theory and its algorithmic aspects. The first property to be noticed is that if M and M' are modules of a congruence partition, then they are either adjacent (any vertex of M neighbors any vertex of M') or non-adjacent in G. It follows that given a congruence partition $\mathcal{P}$, we define its *quotient graph*, $G_{|\mathcal{P}}$ as the subgraph induced by a vertex set $V(G_{|\mathcal{P}})$ satisfying $|V(G_{|\mathcal{P}}) \cap M_i| = 1$ for any $1 \leqslant i \leqslant k$ (see Figure 7).

Theorem 8: [14] Let G be a graph and $\mathcal{P} = \{M_1 \dots M_k\}$ the congruence partition containing the maximal non-trivial strong modules. Then exactly one of the following is true:

1) G is not connected ($G_{|\mathcal{P}}$ is the stable)
2) The complement graph of G is not connected ($G_{|\mathcal{P}}$ is the clique)
3) G and its complement are connected ($G_{|\mathcal{P}}$ is prime)

A *representative graph* is associated with any strong module M: the quotient of the subgraph $G[M]$ (induced by vertices of M) by the congruence partition $\mathcal{M}$ of M into maximal strong modules included in M (see Figure 7). Strong modules whose representative graphs are a clique or a stable are labeled degenerate, while the others are labeled prime. To establish the links between modules of a graph and common intervals, let us explain how a permutation π naturally defines a graph.

Definition 13: Let π be a permutation of the set $\{1 \dots n\}$. The *permutation graph* $G_\pi = (V, E_\pi)$ has vertex set $V = \{1 \dots n\}$ and edge set $E_\pi = \{(i, j) \mid i < j \text{ and } \pi(i) > \pi(j)\}$.

For example, the graph of Figure 7 is a permutation graph: it is the graph G_π with $\pi = (6, 7, 5, 1, 4, 10, 11, 9, 8, 2, 3)$. The main observation is established by the following recently observed property [12], [30]:

Lemma 6: An interval I is a strong common interval of π if and only if I is a strong module of G_π.

Theorem 1 and Proposition 1 directly follows as corollaries of Theorem 8 and Lemma 5. We should finally mention that the modular decomposition theory is part of a more general framework of *partitive families* (see [29]). The algorithmic aspect of modular decomposition has known many developments in the last decade. Very efficient algorithms have been proposed for, given a graph, computing its modular decomposition tree (see [15], [19], [20], [28]).

Remark 4: Note that definite strong interval trees, that define commuting permutations (Section V), are also known as *co-trees* in the theory of modular decomposition of graphs.

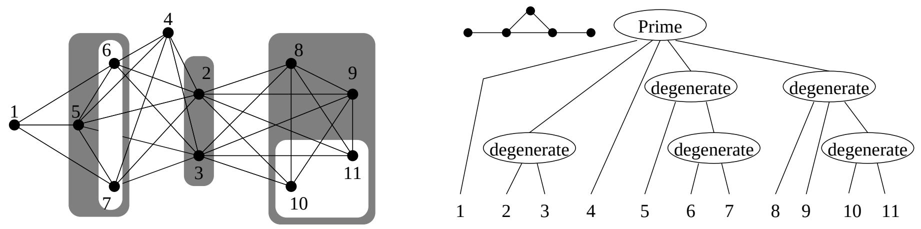

Fig. 7. A graph $G = (V, E)$ and its modular decomposition tree $MD(G)$. The non-trivial strong modules are $\{2, 3\}$, $\{5, 6, 7\}$, $\{10, 11\}$ and $\{8, 9, 10, 11\}$. Any strong module but the module V is degenerated. The *representative graph* of the root is displayed on its left. That representative graph is the *quotient graph* $G_{|\mathcal{P}}$ with $\mathcal{P} = \{\{1\}, \{2, 3, 4\}, \{5\}, \{6, 7\}, \{8, 9, 10, 11\}\}$.

References

[1] Z. Adam and D. Sankoff. Phylogenomics based on conserved genomic segments. Poster at the *3rd RECOMB Comparative Genomics Satellite Workshop*, Trinity College, Dublin (Ireland), September 18th-20th 2005.

[2] D. A. Bader, B. M. E. Moret and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. Comp. Biol.*, 8(5):483–491, 2001.

[3] E. Belda, A. Moya and F. J. Silva. Genome rearrangement distances and gene order phylogeny in γ-Proteobacteria. *Mol. Biol. Evol.* 22(6):1456–1467, 2005.

[4] S. Bérard, A. Bergeron and C. Chauve. Conserved structures in evolution scenarios. In *Comparative Genomics, RECOMB 2004 International Workshop, RCG 2004*, volume 3388 of *Lecture Notes in Bioinformatics*, pages 1–15, Springer-Verlag, 2005.

[5] A. Bergeron, M. Blanchette, A. Chateau and C. Chauve. Reconstructing ancestral gene orders using conserved intervals. In *Algorithms in Bioinformatics, 4th International Workshop, WABI 2004*, volume 3240 of *Lecture Notes in Bioinformatics*, pages 14–25, Springer-Verlag, 2004.

[6] A. Bergeron, C. Chauve, T. Hartman and K. St-Onge. On the properties of sequences of reversals that sort a signed permutation. In *JOBIM 2002 Journées Ouvertes Biologie, Informatique, Mathématiques*, Saint-Malo (France), June 10th-12th, pages 99–108, 2002.

[7] A. Bergeron, C. Chauve, F. de Montgolfier and M. Raffinot. Computing common intervals of K permutations, with applications to modular decomposition of graphs. In *Algorithms - ESA 2005, 13th Annual European Symposium*, volume 3669 of *Lecture Notes in Comput. Sci.*, pages 779–790, Springer-Verlag, 2005.

[8] A. Bergeron, J. Mixtacki and J. Stoye. The inversion distance problem. In *Mathematics of evolution and phylogeny*, (O. Gascuel ed.), pages 262–290, Oxford University Press, 2005.

[9] G. Bourque and P. A. Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Res.*, 12(1):26–36, 2002.

[10] G. Bourque, E. M. Zdobnov, P. Bork, P. A. Pevzner and G. Tesler. Comparative architectures of mammalian and chicken genomes reveal highly variables rates of genomic rearrangements across different lineages. *Genome Res.*, 15(1):98–110, 2005.

[11] G. Bourque, P. A. Pevzner and G. Tesler. Reconstructing the genomic architecture of ancestral mammals: Lessons from human, mouse, and rat genomes. *Genome Res.*, 14(4):507–516, 2004.

[12] B.M. Bui Xuan, M. Habib and C. Paul. Revisiting Uno and Yagiura's Algorithm. In *International Symposium on Algorithms and Computation - ISAAC 2005*, volume 3827 of *Lecture Notes in Comput. Sci.*, pages 146–155, Springer-Verlag, 2005.

[13] A. Chateau. Méthodes de reconstruction de génomes ancestraux : quelques reflexions. Poster 59 at *JOBIM 2005 Journées Ouvertes Biologie, Informatique, Mathématiques*, Lyon (France), July 6th-8th 2005.

[14] M. Chein, M. Habib and M.-C. Maurer. Partitive hypergraphs. *Discrete Math.* 37:35–50, 1981.

[15] E. Dahlhaus, J. Gustedt and R. McConnell. Efficient and practical algorithm for sequential modular decomposition algorithm. *J. Algorithms* 41(2):360–387, 2001.

[16] J.V. Earnest-DeYoung, E. Lerat and B.M.E. Moret. Reversing gene erosion: Reconstructing ancestral bacterial genomes from gene-content and order data. In *Algorithms in Bioinformatics, 4th International Workshop, WABI 2004*, volume 3240 of *Lecture Notes in Bioinformatics*, pages 1–13, Springer-Verlag, 2004.

[17] M. Figeac and J.-S. Varré. Sorting by reversals with common intervals. In *Algorithms in Bioinformatics, 4th International Workshop, WABI 2004*, volume 3240 of *Lecture Notes in Bioinformatics*, pages 26–37, Springer-Verlag, 2004.

[18] R. A. Gibbs *et al.*. Genome sequence of the brown Norway rat yields insights into mammalian evolution. *Nature*, 428(6982):493–521, 2004.

[19] M. Habib, F. de Montgolfier and C. Paul. A simple linear-time modular decomposition algorithm. In *9th Scandinavian Workshop on Algorithm Theory - SWAT 2004*, volume 3111 of *Lecture Notes in Comput. Sci.*, pages 187–198, Springer-Verlag, 2004.

[20] M. Habib, C. Paul and L. Viennot. Partition refinement : an interesting algorithmic tool kit. *International J. of Foundation of Comput. Sci.* 10(2):147–170, 1999.

[21] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, 46(1):1–27, 1999. (Preliminary version in *STOC'95: Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 178–189, ACM Press, 1995.)

[22] S. Heber and J. Stoye. Finding all common intervals of k permutations. In *Combinatorial Pattern Matching, 12th Annual Symposium, CPM 2001*, volume 2089 of *Lecture Notes in Comput. Sci.*, pages 207–218, Springer-Verlag, 2001.

[23] H. Kaplan, R. Shamir and R. E. Tarjan. A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Computing*, 29(3):880–892, 1999.

[24] H. Kaplan and E. Verbin. Sorting signed permutations by reversals revisited. *J. Comput. Sys. Sci.*, 70(3):321–341, 2005.

[25] G.M. Landau, L. Parida and O. Weimann. Using PQ trees for comparative genomics. In *Combinatorial Pattern Matching, 16th Annual Symposium, CPM 2005*, volume 3537 of *Lecture Notes in Comput. Sci.*, pages 128–143, Springer-Verlag, 2005.

[26] B. Larget, D. L. Simon, J. B. Kadane and D. Sweet. A bayesian analysis of metazoan mitochondrial genome arrangements. *Mol. Biol. Evol.* 22(3):486–495, 2005.

[27] W. C. Lathe 3rd, B. Snel and P. Bork. Gene context conservation of a highier order than operons. *Trends Biochem. Sci.* 25(10):474–479, 2000.

[28] R. McConnell and J. Spinrad. Ordered vertex partitioning. *Discrete Math. and Theoretical Comput. Sci.* 4:45–60, 2000.

[29] R.H. Möhring and F.J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Math.* 19:257–356, 1984.

[30] F. de Montgolfier. *Décomposition modulaire des graphes. Théorie, extensions et algorithmes*. Ph.D. thesis, Université Montpellier II (France), 2003.

[31] W. J. Murphy *et al.*. Dynamics of mammalian chromosome evolution inferred from multispecies comparative maps. *Science* 309(5734):613–617, 2005.

[32] P. A. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution: Lessons from human and mouse genomes. *Genome Res.*, 13(1):37–45, 2003.

[33] P. A. Pevzner and G. Tesler. Human and mouse genomic sequences reveal extensive breakpoint reuse in mammalian evolution. *Proc. Natl. Acad. Sci. U.S.A.*, 100(13):7672–7677, 2003.

[34] M.-F. Sagot and E. Tannier. Perfect sorting by reversals. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005*, volume 3595 of *Lecture Notes in Comput. Sci.*, pages 42–52, Springer-Verlag, 2005.

[35] D. Sankoff. Edit distance for genome comparison based on non-local operations. In *Combinatorial Pattern Matching, Third Annual Symposium, CPM 92*, volume 644 of *Lecture Notes in Comput. Sci.*, pages 121–135, Springer-Verlag, 1992.

[36] D. Sankoff and P. Trinh. Chromosomal breakpoint reuse in genome sequence rearrangements. *J. Comp. Biol* 12(6):812–821, 2005.

[37] T. Schmidt and J. Stoye. Quadratic time algorithms for finding common intervals in two and more sequences. In *Combinatorial Pattern Matching, 15th Annual Symposium, CPM 2004*, volume 3109 of *Lecture Notes in Comput. Sci.*, pages 347–358, Springer-Verlag, 2004.
[38] E. Tannier, A. Bergeron and M.-F. Sagot. Advances on sorting by reversals. *Discrete Applied Math.*, to appear. 2005.
[39] G. Tesler. GRIMM: genome rearrangements web server. *Bioinformatics*, 18(3):492–493, 2002.
[40] P. Trinh, A. McLysaght and D. Sankoff. Genomic features in the breakpoint regions between syntenic blocks. *Bioinformatics* 20(Suppl. 1):i318–i325, 2004. (*ISMB/ECCB'04: Proceedings 12th International Conference on Intelligent Systems for Molecular Biology/ 3rd European Conference of Computational Biology*).
[41] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.
[42] S. Yancopoulos, O. Attie and S. Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005.

A nne Bergeron received a PhD Degree in computer science from the University of Montreal in 1990. She is a full professor in the Department of Computer Science at University of Quebec at Montreal. Her research interests are the use of combinatorial and mathematical tools to understand genome organization and evolution.

Sèverine Bérard Sèverine Bérard received a PhD Degree in computer science from the University of Montpellier (France) in 2003. She is a full time researcher in the Department of Applied Mathematics and Computer Science at the French National Institute for Agronomy Research (INRA). Her main research interest is the use of algorithmics and combinatorial tools to compare biological sequences (DNA, RNA, and whole genomes).

Cedric Chauve Cedric Chauve obtained a PhD in computer science from the University of Bordeaux I (France) in 2000. Since 2001 he is professor in the department of computer science at University of Québec in Montréal. His current research interest is the development of combinatorial models and efficient algorithms for comparative genomics.

Christophe Paul is a full time CNRS researcher. He obtained his PhD in Computer Science from Montpellier University (France) in 1998 . His research interests are mainly algorithms and graph theory.