Perfect sorting by reversals is not always difficult[1]

S. Bérard and A. Bergeron and C. Chauve and C. Paul

Juin 2005

Rapport de Recherche LIRMM RR-05042

Abstract

This paper investigates the problem of conservation of combinatorial structures in genome rearrangement scenarios. We characterize a class of signed permutations for which one can compute in polynomial time a reversal scenario that conserves all common intervals, and that is parsimonious among such scenarios. Figeac and Varré (WABI 2004) announced that the general problem is NP-hard. More precisely, we show that there exists a class of permutations for which this computation can be done in linear time with a very simple algorithm, and, for a larger class of signed permutations, the computation can be achieved in subquadratic time. 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. For permutations of this size ($n = 16$), the computation of such scenarios can easily be done by hand.

Résumé

Cet article étudie le problème de la conservation de la structure combinatoire lors de réarrangements génomiques. Nous caractérisons une famille de permutations signées pour lesquelles il est possible de calculer en temps polynomial un scénario de tri par renversement qui conserve tous les intervalles communs. De plus ce scénario reste parcimonieux parmi les scénario parfaits. Figeac et Varré (WABI 2004) avaient annoncé que le problème général est NP-difficile. Plus précisément, nous montrons qu'il existe une classe de permutations pour laquelle ce calcul est effectué par un algorithme linéaire très simple. Plus une plus large classe de permutations signées, ce calcul a un coût sous-quadratique. Nous appliquons ces méthodes au calcul de scénario entre des permutations issues de 16 blocs de synténie du chromosome X de l'humain, la souris et le rat. Pour des permutations de cette taille, le calcul de tels scénario peut être effectué à la main.

Sèverine Bérard
Dépt. de Mathématique et Informatique Appliquée
INRA Toulouse, France
Severine.Berard@toulouse.inra.fr

Anne Bergeron
LaCIM et Dépt. d'Informatique
Université du Québec à Montréal, Canada.
anne@lacim.uqam.ca

Cedric Chauve
LaCIM et Dépt. d'Informatique
Université du Québec à Montréal, Canada.
chauve@lacim.uqam.ca

C. Paul
CNRS - LIRMM
Montpellier, France
paul@lirmm.fr

[1]Research supported by Coopération Franco-Québecoise de Recherche Scientifique et Technologique : "Structures conservées et dupplications pour les réarrangement génomiques"

1 Introduction

The reconstruction of evolution scenarios based on genome rearrangements has proven to be a powerful tool in understanding the evolution of close species. For example, in the last two years, several evolution scenarios have been proposed between mammalian genomes [8, 14], using the MGR and GRIMM softwares [7, 24], or between microbial genomes [12].

The computation of such evolution scenarios relies on the problem of *sorting signed permutations by reversals*: given two chromosomes, represented as sequences of genomic segments – called synteny blocks in [8] –, find a parsimonious sequence of reversals that transforms a chromosome into the other one [15]. However, the number of parsimonious sequences of reversals can be exponential [4]. It is then natural to ask for some additional criteria that can help to select putative scenarios. We are interested in scenarios that do not break combinatorial structures that are present in both chromosomes. Indeed, if two genomes share a common feature, it is likely that their common ancestor did too, which makes evolution scenarios that conserve this feature interesting.

In this work, the combinatorial structures that we consider are *common intervals* [26, 16]. A rearrangement scenario is said to be *perfect* if it does not break any common interval. It was claimed that computing a parsimonious perfect scenario is difficult [13], but that in some non-trivial cases, such scenarios can be computed efficiently [2, 19]. In this paper we describe a class of instances that allow efficient computation of a parsimonious perfect scenario.

In Section 2, we define the links between sorting by reversals and structure conservation, and we state precisely the problem we address in this paper. In Section 3, we relate common intervals and perfect scenarios to a basis of common intervals, the *strong intervals*, that can be represented with a classical data structure in graph theory, the PQ-trees. These observations are consequences of deep relationships between common intervals and the modular decomposition of permutation graphs [10, 18]. This point is central in our approach since we rely strongly, in Section 4, on the combinatorial structure of strong intervals trees to design algorithms that are both efficient – subquadratic time, even linear time in some cases – and simple. The comparison of the mouse and rat X chromosomes, based on the data of [14], yields a tree from which a parsimonious perfect scenario can be deduced immediately (Fig. 2). We also apply these results to the comparison of the human and rat (Fig. 3), and human and mouse (Fig. 4) X chromosomes.

2 Sorting by reversals and common intervals

In this section, we introduce the main concepts covered in this paper: signed permutations, reversals, scenarios, commuting reversals, common intervals and perfect scenarios. We illustrate these definitions in a single example at the end of the section.

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. One can define an interval by giving the set of its unsigned elements, called the *content* of the interval. 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 lead 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.

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.

Given a signed permutation P on n elements, the problem of *sorting by reversals*, stated by Sankoff in [20], asks for a scenario for P of minimal length among all possible scenarios, also called a parsimonious scenario. The first polynomial time algorithm solving this problem was given by Hannenhalli and Pevzner in [15]. Subsequent improvements were proposed, and currently, the best known algorithm for this problem runs in $O(n^{3/2}\log(n))$ worst-case time [23].

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 called *trivial* common intervals.

The notion of *common interval* was introduced in [26]. It was studied, among others, in [16], to model the fact that a group of genes can be rearranged in a genome but still remain connected. It was also studied, in connection with reversal scenarios, in [2, 19].

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.

Example 1 *Let* $P = (\overline{3}\ \overline{2}\ \overline{5}\ \overline{4}\ 1)$ *be a signed permutation.*

1. *Reversing the interval* $(\overline{2}\ \overline{5}\ \overline{4}\ 1)$*, or equivalently the set* $\{1,2,4,5\}$*, yields the signed permutation* $(\overline{3}\ \overline{1}\ 4\ 5\ 2)$.
2. *The common intervals of P are* $\{2,3\},\{4,5\},\{2,3,4,5\},\{1,2,3,4,5\}$ *and the singletons* $\{1\}$*,* $\{2\}$*,* $\{3\}$*,* $\{4\}$ *and* $\{5\}$.
3. $(\{1,2,3,4,5\},\{2,3,4,5\},\{2,3\},\{4,5\},\{1\})$ *is a perfect scenario that transforms* P *into* Id_5.
4. $(\{2,3,4,5\},\{2,3\},\{4,5\},\{1\})$ *is a parsimonious perfect scenario for* P*, transforming* P *into the reverse* $\overline{Id_5}$ *of the identity.*
5. $(\{1,4,5\},\{1,2,3\})$ *is a parsimonious scenario, but it is not perfect since, for example, the reversal* $\{1,4,5\}$ *overlaps the common interval* $\{2,3,4,5\}$.

As shown in [13], given a signed permutation P, there exists at least one perfect scenario for P. However, the authors of [13] claim that the construction of parsimonious perfect scenarios is computationally difficult: they state that computing a parsimonious perfect scenario between two signed permutations is NP-hard in general. Hence the difficulty of the problem relies in the parsimonious aspect.

The main goal of this paper is to propose algorithms for computing parsimonious perfect scenarios that are efficient for large classes of signed permutations (Section 4). Our results rely on the *strong intervals tree* of a signed permutation described in the next section.

3 Strong intervals 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. In [18], the author pointed out a correspondence between common intervals of permutations and the concept, well studied in graph theory, of *modules* of graphs. Inspired from the *modular decomposition* theory[2], this section formally states structural properties of the set of common intervals of a permutation P that are central in the design of the algorithms in Section 4.

Let us first remark that being a common interval for a set 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. For the sake of simplicity, we omit the signs which will be reintroduced in the next section.

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.

For example, 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 $\{1\}, \ldots, \{9\}$. The singletons and $\{1,2,\ldots,9\}$ are the *trivial* strong intervals of P.

[2]All the results presented in this section can be seen as direct consequences or corollaries of well known graph theoretical results (see [10] for example).

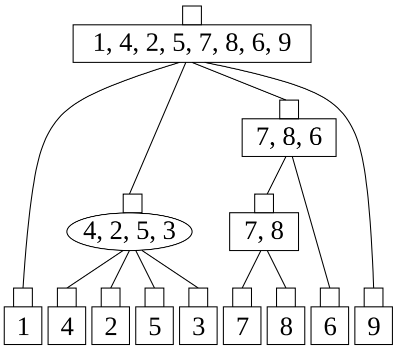

Figure 1: The strong intervals tree $T_s(P)$ of the permutation $P = (1\ 4\ 2\ 5\ 3\ 7\ 8\ 6\ 9)$. Prime and linear nodes (described later in this section) are distinguished by their shape. There are two non-trivial linear nodes, the rectangular nodes: $(7, 8)$ is increasing and $(7, 8, 6)$ is decreasing.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

It follows from Definition 5 that the inclusion order of the set of strong intervals defines an n-leaves 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 intervals tree* of P (see Fig. 1), and we identify a node of $T_s(P)$ with the strong interval it represents.

Since each strong interval with more than one element, or equivalently each internal node of $T_s(P)$, has at least two children in $T_s(P)$, we have immediately:

Proposition 1 *A permutation P on n elements has $O(n)$ strong intervals.*

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. Hence, for two *disjoint* common intervals I and J, we can define the relation $I < J$ by extending the order relation on integers that belong to I and J.

Definition 6 Let P be a permutation and $\mathcal{I} = \{I_1, \ldots, I_k\}$ be a partition of the elements of P into strong intervals. The *quotient permutation* of P with respect to $\mathcal{I}$, denoted $P_{|\mathcal{I}}$, is the permutation on k elements such that i precedes j if and only if $I_i < I_j$.

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

Theorem 1 *Let P be a permutation on n elements and $\mathcal{I} = \{I_1, \ldots, I_k\}$ be the partition of P into strong intervals given by the children of the root of $T_S(P)$. Then exactly one of the following is true:*

1. *$P_{|\mathcal{I}}$ is Id_k (the identity permutation on k elements).*

2. *$P_{|\mathcal{I}}$ is $\overline{Id_k}$ (the reverse of the identity permutation on k elements).*

3. *The only common intervals of $P_{|\mathcal{I}}$ are trivial.*

Proof. Suppose that $P_{|\mathcal{I}}$ is neither Id_k nor $\overline{Id_k}$, and that there exists a non-trivial common interval of $P_{|\mathcal{I}}$. It follows that the strong intervals tree of $P_{|\mathcal{I}}$ has at least one internal node, and then there exists a strong common interval $J = \{i, \ldots, j\}$ of $P_{|\mathcal{I}}$. We will prove that the set $K = \cup_{i \leq h \leq j} I_h$ is a common interval of P, which contradicts the fact that I_h, for $h \in J$, are children of the root.

The set $K = \cup_{i \leq h \leq j} I_h$ is an interval of P, since the intervals I_h occur consecutively in P. For any h, $i \leq h \leq j$, an integer $x \notin K$ is either larger or smaller than all elements of I_h, and such integers exists since J is a non-trivial common interval of $P_{|\mathcal{I}}$. Suppose that $x \notin K$, and that there exists two distinct integers h and h' in J such that x is larger than all elements of I_h, and smaller than all elements of $I_{h'}$. Since $\mathcal{I}$ is a partition of P, x belongs to a unique interval I_ℓ of $\mathcal{I}$ with $\ell \notin J$, since $x \notin K$. It follows that $I_h < I_\ell < I_{h'}$, which implies that, in the quotient permutation $P_{|\mathcal{I}}$, $h < \ell < h'$, and then contradicts the fact that J is a common interval of $P_{|\mathcal{I}}$. Therefore such a pair h and h' does not exist, which implies that for any $x \notin K$, x is either larger or smaller than all elements of K. In other words K is a common interval of P, and as J was a strong interval of $P_{|\mathcal{I}}$, K is a strong interval of P. □

Theorem 1 induces a classification of the nodes of the strong intervals tree $T_s(P)$ that is central in the design of our algorithms: let P_I be the quotient permutation defined by the children of an internal node I of $T_s(P)$. The node 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 nodes are the linear nodes, and the round node $(4, 2, 5, 3)$ is the unique prime node. The only decreasing linear node in this tree is $(7, 8, 6)$.

An interesting property of linear nodes is the fact that increasing and decreasing nodes alternate when they are direct descendant of each other. More formally:

Property 1 *In a strong intervals tree, a child of an increasing (resp. a decreasing) linear node is either a prime node or a decreasing (resp. an increasing) linear node.*

Finally, we show that the strong intervals 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 2 *Let P be a signed permutation. An interval I of P is a common interval of P if and only if it is either a node of $T_S(P)$, or the union of consecutive children of a linear node of $T_S(P)$.*

Proof. Let I be a common interval of P. If I is a strong interval, the statement holds. Otherwise, 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 I has to be linear. Indeed, if I is prime, the point 3 of Theorem 1 implies that any subset of the children of I is not a common interval of P.

Conversely, it is immediate to see that every node or set of consecutive children of a linear node is a common interval of P. □

This representation for strong intervals was first given implicitly in [16], and explicitly in [17], 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 $O(n)$ worst-case time using algorithms similar to the ones given in [16, 17]. A formal link between PQ-trees and conserved structures in signed permutations was first proposed in [3], in the context of conserved intervals, a subset of common intervals.

4 Computing perfect scenarios

We now turn to the description of our main results, the design of efficient algorithms for computing parsimonious perfect scenarios for large classes of signed permutations. The central point of this section is the use strong intervals trees as guides for such computations. We the assume that $T_S(P)$ is given, and we refer to [5] for simple algorithms building this tree. Indeed, given a signed permutation P, $T_S(P)$ encodes all common intervals of P, which allows, in particular, a characterization of perfect scenarios in terms of $T_S(P)$:

Proposition 3 *A scenario S for a permutation P is perfect if and only if each of the reversals of S is either a node of $T_S(P)$, or the union of children of a prime node of $T_S(P)$.*

Proof. Suppose that S is a scenario for permutation P, and that every reversal of S is either a node of $T_S(P)$, or the union of children of a prime node of $T_S(P)$. Let I be a reversal of S. If I is a node 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 node 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 node. 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 $I_1, I_2, \dots I_k$ 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 $I_1, I_2, \dots I_k$ must all be children of a same parent J in $T_S(P)$, otherwise I would not commute with the nodes of $T_S(P)$ that are parents of I_j's. If J is a linear node, then I must be equal to J, otherwise I would overlap a 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 node of $T_S(P)$, or $k > 1$ and the node J must be prime. □

Computing a perfect scenario S thus amounts to identify leaves, linear nodes and union of children of prime nodes of $T_S(P)$ that are the reversals of S. We now show that, even if the general problem of computing parsimonious perfect scenarios was claimed to be difficult [13], it can be done efficiently for a large class of signed permutations, defined in terms of the structure of their strong intervals tree.

A strong intervals tree $T_S(P)$ is *unambiguous* if every prime node has a linear parent, and *definite* if it has no prime nodes. Note that a definite tree is unambiguous. A tree that is not unambiguous is said to be *ambiguous*. We will show that, for definite trees, there is essentially a unique perfect scenario for P (Theorem 2), and for unambiguous trees, we can compute a parsimonious perfect scenario in subquadratic time (Theorem 3).

Remark 1 *Note that definite strong intervals trees are also known as* co-trees *in the theory of modular decomposition of graphs [10].*

A *signed* tree is a tree in which each node has a sign, $+$ or $-$. If a tree $T_S(P)$ is unambiguous, we associate to $T_S(P)$ the following signed tree $T'_S(P)$:

1. The sign of a leaf x is the sign of the corresponding element in P.
2. The sign of a linear node is $+$, if the node is increasing, and $-$ if the node is decreasing.
3. The sign of a prime node is the sign of its parent.

Fig. 2, Fig. 3 and Fig. 4 show signed strong intervals trees associated to the permutations obtained by comparing 16 synteny blocks of the human, mouse and rat X chromosomes [14][3]. Note that, in Fig. 2 the labels of the nodes are given with respect to the order of the blocks of the mouse chromosome.

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

Theorem 2 *If $T_S(P)$ is definite, then the set of nodes that have a sign different from the sign of their parent is a parsimonious perfect scenario for P.*

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

The proof of Theorem 2 relies on the following lemma that identifies reversals that must belong to any perfect scenario, thus to any parsimonious perfect scenario.

Lemma 1 (The Parity Lemma) *For any unambiguous tree $T_S(P)$, if a node I has a linear parent and a sign different from the sign of its parent, then I belongs to any perfect scenario.*

Proof. Let S be a perfect scenario, and I a node with negative sign, whose linear parent J has a positive sign, and such that $I \notin S$. Since J is linear and $I \notin S$, by Proposition 3, any reversal of S that contains I also contains J.

Let m be the number of reversals of S that contain J. If m is even, and since J has a positive sign, then S sorts P to Id, by definition of increasing nodes. 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 node with a negative sign, then its first child is greater than its last, and will still be after an even number of reversals, again contradicting the fact 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. □

[3]The positions of blocks 5 and 6 in our data differs from [14], following a correction of the mouse genome assembly.

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

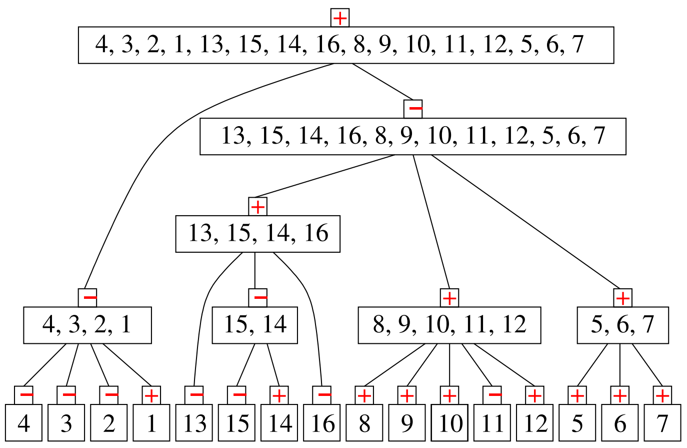

Figure 2: Comparing the rat and mouse X chromosomes: the set of nodes that have a sign different from the sign of their parent form a parsimonious perfect scenario that transforms the rat X chromosome into the mouse X chromosome in 11 reversals: $(4, 3, 2, 1)$, (1), $(13, 15, 14, 16, 8, 9, 10, 11, 12,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

Proof.[Theorem 2] If $T_S(P)$ is definite, Proposition 3 implies that a parsimonious perfect scenario S consists of a set of nodes of the tree $T_S(P)$. From Property 1, all internal nodes have a different sign than their parent, so they must all belong to S, by the Parity Lemma. The root never belongs to a parsimonious perfect scenario because it would be a useless reversal, since a perfect scenario is defined with respect to both Id and $\overline{Id}$. Any leaf that has a different sign from its parent must belong to S, by the Parity Lemma, and any leaf that has the sign of its parent must not, by an argument similar to the proof of the Parity Lemma. Thus S is uniquely determined. □

We next turn to the more general case of unambiguous trees. Recall that a prime node inherits its sign from its parent, and that any reversal that is a union of children of a prime node 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 node I of the tree, any parsimonious scenario that sorts the children of node I in increasing or decreasing order, depending on the sign of I. Then, it suffices to deal with linear nodes whose parent is linear in the same way than for a definite treeFig. 3 shows the signed tree associated to the permutations of the human and rat X chromosomes. This tree is unambiguous: it has one prime node $(4, 5, 6, 12, 8, 7, 2, 1, 3, 9, 10, 11)$ whose parent is a decreasing linear node. The quotient permutation of this node 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 node. Using the $O(n^{3/2} \log n)$ algorithm described in [23], we have:

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

```
S is an empty scenario.
For each prime node 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 any parsimonious scenario T from $P_I$ to $Id$
        Else compute any 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
Add to S the linear nodes and leaves having a linear parent and a sign different from the sign of their parent.
```

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

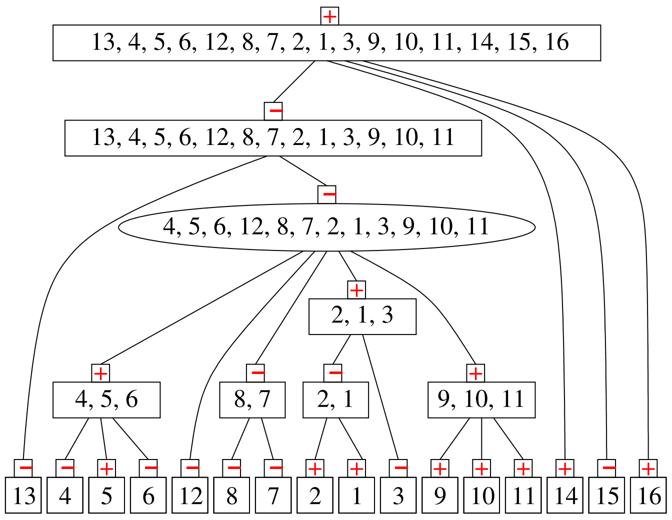

Figure 3: Comparing the human and rat X chromosomes: a parsimonious perfect scenario is obtained by sorting the five children $(4,5,6)$, (12), $(8,7)$, $(2,1,3)$ and $(9,10,11)$ in decreasing order using any parsimonious scenario that sorts the quotient permutation $P_I = (2\ \overline{5}\ \overline{3}\ 1\ 4)$, and then reversing the linear nodes 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.

Theorem 3 *If $T_S(P)$ is unambiguous, Algorithm 1 computes a parsimonious perfect scenario for P in subquadratic time.*

Proof. The time complexity bound is obtained by observing 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 nodes of $T_S(P)$, and that $n_1 + \ldots + n_k \leq n$.

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 node 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 nodes. The argument for the second phase – leaves and linear nodes 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 prefect scenarios. Given a parsimonious perfect scenario, the subsequence of reversals that are contained in a prime node 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 node 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. $\square$

When $T_S(P)$ is ambiguous, the sign of some prime nodes is undefined. A general algorithm to compute parsimonious perfect scenario would repeatedly apply Algorithm 1 to all possible sign assignments to prime nodes that do not have linear parents. As an example, consider Fig. 4 that shows the signed tree associated to the permutations of the human and mouse X chromosomes. This tree is ambiguous since its root is a prime node, and we must try to sort this node both to Id and to $\overline{Id}$. In this case, both parsimonious scenarios have the same length. Such an algorithm, that is a generalization of the algorithms we described in this section for unambiguous trees, and was described using another formalism in [13]. It has a worst-case time complexity that is exponential in the number of prime nodes whose parent is prime, and thus is efficient if the number of such edges is small.

Permutations that arise from the comparison of genomic sequences are not "random", and this could partly explain why perfect sorting is not difficult for the permutations we considered. Constructing permutations that are hard to perfectly sort requires to break almost any structure in a given permutation. The smallest example of a hard to sort permutation is given in Fig. 5.

5 Conclusion

From the algorithmic point of view, the central aspect of our work is the detailed description of the link between computing perfect scenarios and the strong intervals tree of a signed permutation. We gave a new description of the exponential time algorithm of [13] that highlights many of its properties. In particular, in Section 4, we characterized classes of signed permutations for which the computation of a parsimonious perfect scenario can be done efficiently.

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

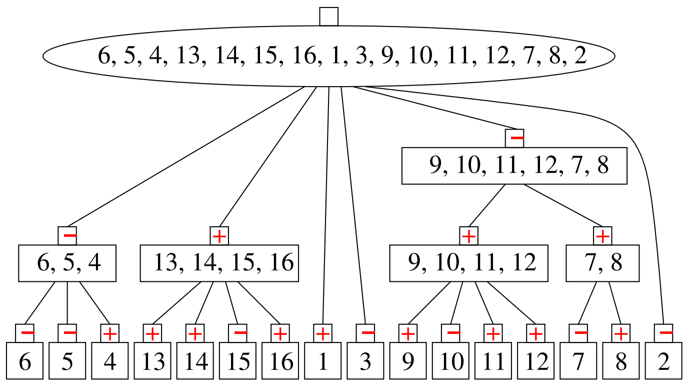

Figure 4: Comparing the human and mouse X chromosomes: the root has no sign but its children can be sorted to Id or $\overline{Id}$ in 6 reversals using a parsimonious scenario that sorts the quotient permutation $P_I = (\bar{4}\ 6\ 1\ \bar{3}\ \bar{5}\ 2)$, A parsimonious perfect scenario would also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

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

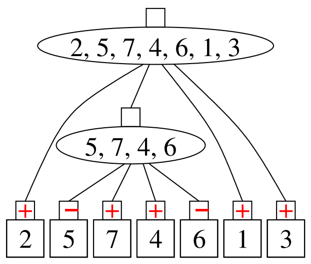

Figure 5: A hard to sort permutation: if both nodes are sorted in increasing order, or both are sorted in decreasing order, then the resulting perfect scenarios are not parsimonious.

In [19], 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 subquadratic time, extending a previous result of [2]. 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 algorithm for computing the reversal distance proposed in [1, 6]. However, the computation of a parsimonious perfect scenario can ask for an exponential time depending on the strong intervals 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 intervals tree, the class of signed permutations for which a parsimonious perfect scenario is also parsimonious among all possible scenarios.

From a practical point of view, it is worth to recall that the interest in computing scenarios that do not break any common intervals relies on the assumption that genes, or other genomic markers, cluster in such groups for functional reasons, like co-transcription for example. Of course, it is possible that clusters of genes exist by "chance", or are not supported by any functional evidence, and it would not be relevant to impose that such intervals should not be broken. Note however, that the algorithms developed in this work apply without any modification: given a set of common intervals that are believed to be pertinent from the evolutionary point of view, they define a set of strong intervals, and then a PQ-tree, and one can apply our method on this PQ-tree.

Another interesting point, related to the above remark, is the importance, with respect to the computation of perfect scenarios, of edges of the strong intervals tree whose two incident vertices are prime. These edges are the heart of the difficulty of the general computational problem. However, if it appears that a prime node has not to be broken during the evolution, for functional reasons, 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 sequences that coordinates the action of the group of genes present in this genomic segment. It follows that if one considers these framing sequences in the signed permutation, they would prevent, in the strong intervals tree, that the corresponding prime node has for parent a prime node. Recent works that explore the properties of the chromosomal regions between synteny blocks are of interest with regard to this problematic [21, 25].

Among other future directions, it would be interesting to consider the median problem, that is one of the main tools in the computation of reversals scenarios [7]. This problem has been shown to be NP-hard [9] in the general case, but the question has not been settled if one restricts every scenario to be perfect. Finally, the most natural extension would be to consider signed sequences instead of signed permutations. Some work has been done on common intervals of signed sequences [11, 22], but representations of these that could play the role of strong intervals are yet to be discovered.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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*, pages 99–108, 2002.

[5] A. Bergeron, C. Chauve, F. de Montgolfier and M. Raffinot. Computing common intervals of K permutations, with applications to modular decomposition of graphs. To appear in *European Symposium on Algorithms, ESA 2005*, (to be published in *Lecture Notes in Comput. Sci.*, Springer-Verlag), 2005.

[6] A. Bergeron, J. Mixtacki and J. Stoye. Reversal distance without hurdles and fortresses. In *Combinatorial Pattern Matching, 15th Annual Symposium, CPM 2004*, volume 3109 of *Lecture Notes in Comput. Sci.*, pages 388–399, Springer-Verlag, 2004.

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

[8] 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.

[9] A. Caprara. Formulations and hardness of multiple sorting by reversals. In *RECOMB'99: Proceedings of the Third Annual International Conference on Computational Molecular Biology*, pages 84–94, ACM Press, 1999.

[10] M. Chein, M. Habib and M. C. Maurer. Partitive hypergraphs. *Discrete Math.*, 37(1):35–50, 1981.

[11] G. Didier. Common intervals of two sequences. In *Algorithms in Bioinformatics, Third International Workshop, WABI 2003*, volume 2812 of *Lecture Notes in Bioinformatics*, pages 17–24, Springer-Verlag, 2003.

[12] 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.

[13] 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.

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

[15] 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.)

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] D. Sankoff and P. Trinh. Chromosomal breakpoint re-use in the inference of genome sequence rearrangements. In *RECOMB'04: Proceedings of the 8th Annual International Conference on Computational Molecular Biology*, pages 30–35, ACM Press, 2004. (Extended version to appear in *J. Comp. Biol.*)

[22] 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.

[23] E. Tannier and M.-F. Sagot. Sorting by reversals in subquadratic time. In *Combinatorial Pattern Matching, 15th Annual Symposium, CPM 2004*, volume 3109 of *Lecture Notes in Comput. Sci.*, pages 1–13, Springer-Verlag, 2004.

[24] G. Tesler. GRIMM: genome rearrangements web server. *Bioinformatics*, 18(3):492–493, 2002.

[25] 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*).

[26] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.