Perfect sorting by reversals is not always difficult (extended abstract)

Sèverine Bérard

Département de Mathématiques et Informatique Appliquée, INRA
Toulouse

Anne Bergeron, Cedric Chauve

LaCIM and Département d'Informatique
Université du Québec à Montréal

Christophe Paul

CNRS, LIRMM, Université Montpellier II

Algorithms in Bioinformatics, 5th International Workshop, WABI 2005, volume 3692 of *Lecture Notes in Bioinformatics*, pages 228–238, 2005.

Perfect sorting by reversals is not always difficult
(Extended abstract*)

Sèverine Bérard[1], Anne Bergeron[2], Cedric Chauve[2], Christophe Paul[3]

[1] INRA Toulouse, Dépt. de Mathématique et Informatique Appliquée, France.
E-mail: Severine.Berard@toulouse.inra.fr
[2] LaCIM et Dépt. d'Informatique, Université du Québec à Montréal, Canada.
E-mail: [anne,chauve]@lacim.uqam.ca
[3] CNRS, LIRMM, Montpellier, France. E-mail: paul@lirmm.fr

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. 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 permutations obtained from the X chromosomes of the human, mouse and rat.

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 [9, 14]. 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, find a parsimonious sequence of reversals that transforms a chromosome into the other one. However, the number of parsimonious sequences of reversals can be exponential [5]. 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. In this work, the combinatorial structures that we consider are *common intervals* [22, 15]. A rearrangement scenario is said to be *perfect* if it does not break any common interval. It was claimed in [13] that computing a parsimonious perfect scenario is difficult, but recent works [2, 18] showed that in some non-trivial cases, such scenarios can be computed efficiently. In this paper we describe a class of instances that allow efficient computation of a parsimonious perfect scenario.

* A complete version, including proofs, is available in [3].

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 [11, 17]. This point is central in our approach since we rely, 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. We apply these results to the comparison of the human, mouse and rat X chromosomes, based on data of [14].

2 Sorting by reversals and common intervals

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.

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 is called a *scenario* for P.

Given a signed permutation P on n elements, the problem of *sorting by reversals*, defined in [19], asks for a scenario for P of minimal length among all possible scenarios, also called a *parsimonious* scenario. Currently, the best known algorithm for this problem runs in $O(n^{3/2} \log(n))$ worst-case time [21].

Definition 2. Two distinct intervals I and J *commute* if 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 called *trivial* common intervals.

The notion of *common interval* was introduced in [22]. It was studied, among others, in [15], to model the fact that a group of genes can be rearranged in a genome but still remain connected.

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.*

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 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 trees

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 [17], 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[4], this section describes 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 $P = (1\ 4\ 2\ 5\ 3\ 7\ 8\ 6\ 9)$ are $\{2,3,4,5\}$, $\{7,8\}$, $\{6,7,8\}$, $\{1,2,3,4,5,6,7,8,9\}$ and $\{1\}$,..., $\{9\}$. 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-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 (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 on n elements has $O(n)$ strong intervals.*

Let I be a common interval of a permutation P on n elements 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 $I_i < I_j$.

[4] All the results presented in this section can be seen as direct consequences or corollaries of well known graph theoretical results (see [11] for example).

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 $P_{|\mathcal{I}} = (1\ 2\ 4\ 3\ 5)$.

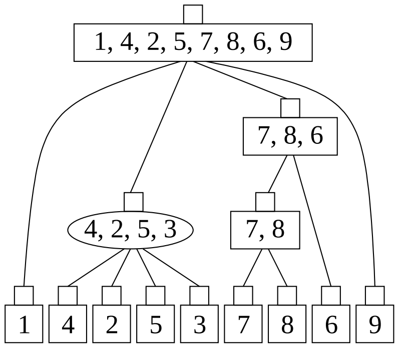

Fig. 1. The strong intervals tree $T_s(P)$ of the permutation $P = (1\ 4\ 2\ 5\ 3\ 7\ 8\ 6\ 9)$. Prime and linear nodes are distinguished by their shape. There are two non-trivial linear nodes, the rectangular nodes: $(7, 8)$ is increasing and $(7, 8, 6)$ is decreasing. There is only one prime node,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

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.*

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 $(7, 8, 6)$.

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 $T_S(P)$.*

This representation for strong intervals was first given implicitly in [15], and explicitly in [16], 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 [15, 16]. A formal link between PQ-trees and conserved structures in signed permutations was first proposed in [4], in the context of conserved intervals, a subset of common intervals.

4 Computing perfect scenarios

We now turn to the description of efficient algorithms for computing parsimonious perfect scenarios for large classes of signed permutations. The central point is a characterization of perfect scenarios in terms of $T_S(P)$. We assume that $T_S(P)$ is given (see [6] for a simple algoritm to build it).

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 $T_S(P)$.*

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. 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. 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 [11].

A *signed* tree is a tree in which each node has a sign, $+$ or $-$. We associate to an unambiguous tree $T_S(P)$ 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]. In Fig. 2 the labels of the nodes are given with respect to the order of the blocks of the mouse chromosome.

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

Theorem 2. *If $T_S(P)$ is definite, the set of nodes having 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, when $T_S(P)$ is definite, computing a parsimonious perfect scenario for P is almost immediate. 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.

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

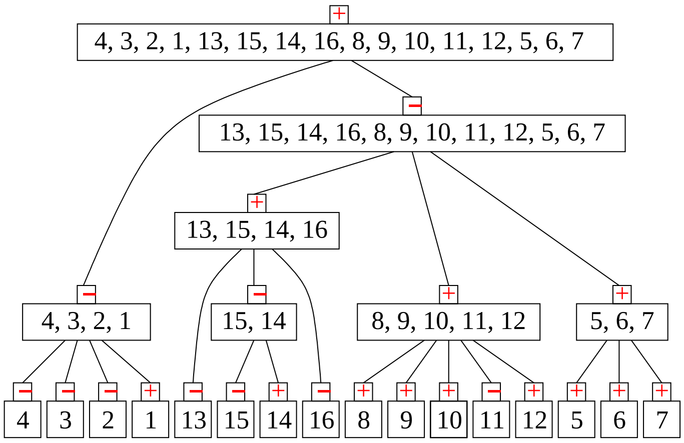

Fig. 2. Comparing the rat and mouse X chromosomes. The set of nodes having a sign different from the sign of their parent form a parsimonious perfect scenario that transforms the rat X chromosome into the mouse X chromosome: $(4, 3, 2, 1)$, (1), $(13, 15, 14, 16, 8, 9, 10, 11, 12, 5, 6, 7)$, $(13, 15,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

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 tree.

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.
```

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 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 [21], we have:

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

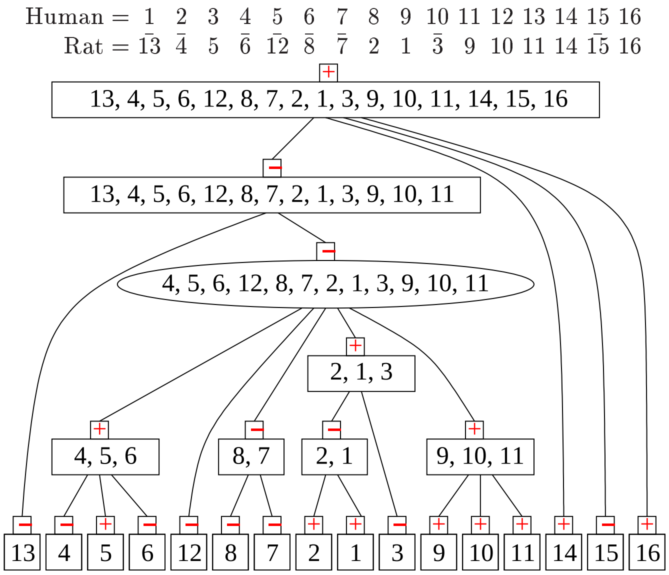

Fig. 3. Comparing the human and rat X chromosomes: a parsimonious perfect scenario is obtained by sorting the five children $(4,5,6)$, (12), $(8,7)$, $(2,1,3)$ and $(9,10,11)$ in decreasing order using any parsimonious scenario that sorts the quotient permutation $P_I = (2\ \overline{5}\ \overline{3}\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

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 is a generalization of the algorithm 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 nodes is small.

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

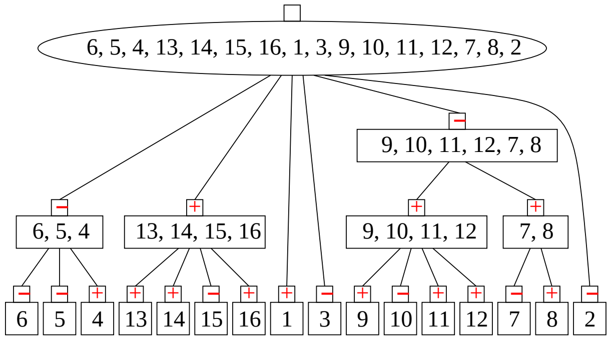

Fig. 4. Comparing the human and mouse X chromosomes: the root has no sign but its children can be sorted to Id or $\overline{Id}$ in 6 reversals using a parsimonious scenario that sorts the quotient permutation $P_I = (\bar{4}\ 6\ 1\ \bar{3}\ \bar{5}\ 2)$, A parsimonious perfect scenario would also contain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

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 sort perfectly requires to break almost any structure in a given permutation. The smallest example of a hard to sort permutation is given by the permutation $P = (2\ \bar{5}\ 7\ 4\ \bar{6}\ 1\ 3)$.

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.

In [18], it was shown that when there exists a parsimonious scenario that is also a perfect scenario for a given signed permutation, 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, 7]. However, the computation of a parsimonious perfect scenario can require 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 can be used 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.

Among other future directions, it would be interesting to consider the median problem, that is one of the main reasons for computing reversals scenarios [8]. This problem has been shown to be NP-hard [10] 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 [12, 20], but structures that could play the role of strong intervals are yet to be discovered in this case.

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 *RCG 2004*, volume 3388 of *Lecture Notes in Bioinformatics*, p. 1–15, 2005.
3. S. Bérard, A. Bergeron, C. Chauve and C .Paul. Perfect sorting by reversals is not always difficult. Technical Report LIRMM RR-05042 (Montpellier, France), 2005.
4. A. Bergeron, M. Blanchette, A. Chateau and C. Chauve Reconstructing ancestral gene orders using conserved intervals. In *WABI 2004*, volume 3240 of *Lecture Notes in Bioinformatics*, p. 14–25, 2004.
5. 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*, p. 99–108, 2002.
6. 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 *ESA 2005*, (to be published in *Lecture Notes in Comput. Sci.*), 2005.
7. A. Bergeron, J. Mixtacki and J. Stoye. Reversal distance without hurdles and fortresses. In *CPM 2004*, volume 3109 of *Lecture Notes in Comput. Sci.*, p. 388–399, 2004.

8. G. Bourque and P. A. Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Res.*, 12(1):26–36, 2002.
9. 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.
10. A. Caprara. Formulations and hardness of multiple sorting by reversals. In *RECOMB'99*, p. 84–94, ACM Press, 1999.
11. M. Chein, M. Habib and M. C. Maurer. Partitive hypergraphs. *Discrete Math.*, 37(1):35–50, 1981.
12. G. Didier. Common intervals of two sequences. In *WABI 2003*, volume 2812 of *Lecture Notes in Bioinformatics*, p. 17–24, 2003.
13. M. Figeac and J.-S. Varré. Sorting by reversals with common intervals. In *WABI 2004*, volume 3240 of *Lecture Notes in Bioinformatics*, p. 26–37, 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. Heber and J. Stoye. Finding all common intervals of k permutations. In *CPM 2001*, volume 2089 of *Lecture Notes in Comput. Sci.*, p. 207–218, 2001.
16. G.M. Landau, L. Parida and O. Weimann. Using PQ trees for comparative genomics. In *CPM 2005*, volume 3537 of *Lecture Notes in Comput. Sci.*, p. 128–143, 2005.
17. F. de Montgolfier *Décomposition modulaire des graphes. Théorie, extensions et algorithmes.* Ph.D. thesis, Université Montpellier II (France), 2003.
18. M.-F. Sagot and E. Tannier. Perfect sorting by reversals. In *COCOON 2005*, volume 3595 of *Lecture Notes in Comput. Sci.*, pages 42–52, 2005.
19. D. Sankoff. Edit distance for genome comparison based on non-local operations. In *CPM 1992*, volume 644 of *Lecture Notes in Comput. Sci.*, p. 121–135, 1992.
20. T. Schmidt and J. Stoye. Quadratic time algorithms for finding common intervals in two and more sequences. In *CPM 2004*, volume 3109 of *Lecture Notes in Comput. Sci.*, p. 347–358, 2004.
21. E. Tannier and M.-F. Sagot. Sorting by reversals in subquadratic time. In *CPM 2004*, volume 3109 of *Lecture Notes in Comput. Sci.*, p. 1–13, 2004.
22. T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.