Sorting by reversals in subquadratic time

Eric Tannier, Marie-France Sagot

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Sorting by reversals in subquadratic time

Eric Tannier — Marie-France Sagot

N° 5097

Janvier 2004

THÈME 3

ISSN 0249-6399 ISRN INRIA/RR--5097--FR+ENG

INRIA
RHÔNE-ALPES

Sorting by reversals in subquadratic time

Eric Tannier * , Marie-France Sagot †*

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Helix

Rapport de recherche n° 5097 — Janvier 2004 — 15 pages

Abstract: The problem of sorting by signed reversals is inspired by a genome rearrangement problem in computational molecular biology. Given two genomes represented as signed permutations of the same elements (*e.g. orthologous genes*), the problem consists in finding a most parsimonious scenario of reversals that transforms one genome into the other. We propose a method for sorting a signed permutation by reversals in time $O(n\sqrt{n \log n})$. The best known algorithms run in time $O(n^2)$, the main obstacle to an improvement being a costly operation of detection of so-called "safe" reversals. We bypass this detection and, using the same data structure as a previous random approximation algorithm, we achieve the same subquadratic complexity for finding an *exact* optimal solution. This answers an open question by Ozery-Flato and Shamir whether a subquadratic complexity could ever be achieved for solving the problem.

Key-words: genome rearrangement, sorting permutations, reversals, inversions, polynomial time algorithm, computational biology

Work supported by the French program bioinformatique Inter-EPST 2002 "Algorithms for Modelling and Inference Problems in Molecular Biology".

* INRIA Rhône-Alpes, Laboratoire de Biométrie et Biologie Évolutive, Université Claude Bernard, 69622 Villeurbanne cedex, France
† King's College, London, UK

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Tri par inversions de complexité sous-quadratique

Résumé : Le problème du tri des permutations signées par inversions est motivé par la théorie du réarrangement génomique en biologie moléculaire. Étant donnés deux génomes représentés par des permutations signées des mêmes éléments (par exemple des gènes orthologues), le but est de trouver une séquence d'inversions de taille minimum, qui transforme un génome en l'autre. Nous décrivons une méthode de tri des permutations signées par inversion de complexité en temps $O(n\sqrt{n \log n})$. Les meilleurs algorithmes connus sont de complexité $O(n^2)$, l'obsacle principal à une amélioration étant la détection à chaque pas des inversions dites "sûres". Nous éliminons cette étape dans notre méthode, et en utilisant une structure de représentation des permutations introduite précédemment pour construire une heuristique pour le tri par inversions, nous construisons une méthode exacte en temps sous-quadratique. Ceci répond à une question posée par Ozery-Flato et Shamir sur l'existence d'un tel algorithme.

Mots-clés : réarrangement génomique, tri de permutations, inversions, algorithme polynomial, bioinformatique

1 Introduction

Genome rearrangements such as reversals may change the order of the genes (or other markers) in a genome, and also the direction of transcription. We identify the genes with the numbers $1, \ldots, n$, with a plus or minus sign to indicate such direction. Their order will be represented by a *signed permutation* of $\{\pm 1, \ldots, \pm n\}$, that is a permutation π of $\{\pm 1, \ldots, \pm n\}$ such that $\pi[-i] = -\pi[i]$, where $\pi[i]$ denotes the i^{th} element in π. This will be needed when we later introduce the inverse of a signed permutation. In the following, we indicate the sign of an element in a permutation only when it is minus.

The *reversal* of the interval $[i, j] \subseteq [1, n]$ $(i < j)$ is the signed permutation

$$\rho = 1, \ldots, i, -j, \ldots, -(i+1), j+1, \ldots, n.$$

Note that $\pi\rho$ is the permutation obtained from π by reversing the order and flipping the sings of the elements in the interval. If $\rho_1, \ldots, \rho_k$ is a sequence of reversals, we say that it sorts a permutation π if $\pi\rho_1 \cdots \rho_k = Id$ (Id is the all positive identity permutation $1 \ldots n$). We denote by $d(\pi)$ the number of reversals in a minimum size sequence sorting π.

The problem of sorting a signed permutation by reversals is inspired by a problem of genome rearrangement in computational biology. It has been the subject of an extensive literature. The first polynomial algorithm was given by Hannenhalli and Pevzner [3], and ran in $O(n^4)$. After many subsequent improvements, the currently fastest algorithms are those of Kaplan, Shamir and Tarjan [4] running in $O(n^2)$, a linear algorithm for computing $d(\pi)$ only (it does not give the sequence of reversals) by Bader, Moret and Yan [1], and an $O(n\sqrt{n \log n})$ random algorithm by Kaplan and Verbin [5] which gives most of the time an optimal sequence of reversals, but fails on some permutations with very high probability. A reversal ρ is said to be *safe* for a permutation π if $d(\pi\rho) = d(\pi) - 1$. The bottleneck for all existing exact algorithms is the detection of safe reversals. Many techniques were invented to address this problem, but none has a better time complexity than linear, immediately implying a quadratic complexity for the whole method. In a recent paper [6], Ozery-Flato and Shamir compiled and compared the best algorithms, and wrote that: "A central question in the study of genome rearrangements is whether one can obtain a subquadratic algorithm for sorting by reversals".

In this paper, we give a positive answer to Ozery-Flato and Shamir's question. A good knowledge of the Hannenhalli-Pevzner theory and of the data structure used by Kaplan-Verbin is assumed.

In the next section, we briefly describe the usual tools (in particular the omnipresent breakpoint graph) for dealing with permutations and reversals. We mention some operations for sorting by reversals without giving any details (concerning, for instance, hurdle detection and clearing for which linear methods exist). The aim of this paper is to deal only with the most costly part of the method, that is sorting a permutation without hurdles. We introduce a new and elegant way of transforming the breakpoint graph of a permutation π by applying

reversals either on π *or on its inverse permutation* π^{-1}. In section 3, we describe the method to optimally sort by reversals. With any classical data structure, the time complexity of this algorithm is $O(n^2)$, but even in this case it presents a special interest because it bypasses the detection of safe reversals which is considered as the most costly operation. Then in the last section, we indicate the data structure used to achieve subquadratic time complexity.

2 Mathematical tools

To simplify exposition, we require that the first and last elements of a permutation remain unchanged. We therefore adopt the usual transformation which consists in adding the elements 0 and $n+1$ to $\{1,\ldots,n\}$, with $\pi[0] = 0$, and $\pi[n+1] = n+1$. The obtained permutation is called an *augmented permutation*. In this paper, all permutations are augmented, and we omit to mention it from now on. The inverse permutation π^{-1} of π is the permutation such that $\pi\pi^{-1} = Id$.

2.1 Breakpoint graphs for a permutation and its inverse

The *breakpoint graph* $BG(\pi)$ of a permutation π is a graph with vertex set V defined as follows: for each integer i in $\{1,\ldots,n\}$, let i^d (the *departure*) and i^a (the *arrival*) be two vertices in V; add to V the two vertices 0^d and $(n+1)^a$. Observe that all vertex labels are non negative numbers. For simplicity and to avoid having to use absolute values, we may later refer to vertex $(-i)^x$ (for $x = a$ or d). This will be the same as vertex i^x. The edge set E of $BG(\pi)$ is the union of two perfect matchings denoted by R, the *reality edges* and D, the *dream edges* (in the literature, reality and dream edges are sometimes called reality and desire edges, or, in a more prosaic way, black and gray edges):

- R contains the edges $(\pi[i])^d(\pi[i+1])^a$ for all $i \in \{0,\ldots,n\}$;
- D contains an edge for all $i \in \{0,\ldots,n\}$, from i^d if $\pi^{-1}[i]$ is positive, from i^a if $\pi^{-1}[i]$ is negative, to $(i+1)^a$ if $\pi^{-1}[i+1]$ is positive, and to $(i+1)^d$ if $\pi^{-1}[i+1]$ is negative.

The reality edges define the permutation π (what you have), and the dream edges define Id (what you want to have). Reality edges always go from a departure to an arrival, but in dreams, everything can happen. An example of a breakpoint graph for a permutation is given in Figure 1.

To avoid case checking, in the notation of an edge, the mention of departures and arrivals may be omitted. For instance, $i(i+1)$ is a dream edge, indicating nothing as concerns the signs of $\pi^{-1}[i]$ and $\pi^{-1}[i+1]$.

A reality edge $(\pi[i])^d(\pi[i+1])^a$ is *oriented* if $\pi[i]$ and $\pi[i+1]$ have opposite signs, and *unoriented* otherwise. A dream edge $i(i+1)$ is *oriented* if $\pi^{-1}[i]$ and $\pi^{-1}[i+1]$ have opposite signs (that is, if the edge joins two departures or two arrivals), and *unoriented* otherwise.

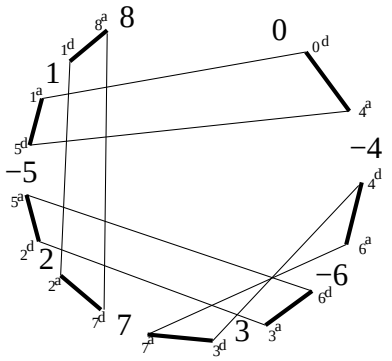

Figure 1: The breakpoint graph of the permutation $0\ -4\ -6\ 3\ 7\ 2\ -5\ 1\ 8$. The dash edges are reality edges, and the thin ones are dream edges. The permutation should be read clockwise from 0 to $n+1$. This circular representation makes the cycles in the graph more visible. The edges that cross in the drawing correspond to crossing edges.

In the example of Figure 1, $(0,4)$, $(6,3)$, $(2,5)$ and $(5,1)$ are oriented reality edges, while $(3,4)$, $(6,7)$ are oriented dream edges.

To every dream edge $i(i+1)$, we associate the interval $[|\pi^{-1}[i]|, |\pi^{-1}[i+1]|]$ (or $[|\pi^{-1}[i+1]|, |\pi^{-1}[i]|]$ if $|\pi^{-1}[i]| > |\pi^{-1}[i+1]|$). Two dream edges are said to *cross* if their associated intervals intersect but one is not contained in the other. Only dream edges may cross in a breakpoint graph.

Dream and reality edges are trivially and uniquely decomposed into cycles (the sets of both types of edges are perfect matchings of the vertices). By the cycles of a permutation π, we mean the cycles of $R \cup D$ in $BG(\pi)$. We call the *size* of a cycle the number of dream edges it contains (it is half the usual length of a cycle). Two cycles are said to *cross* if two of their edges cross.

A *component* $\mathcal{C}$ of $BG(\pi)$ is an inclusionwise minimal subset of its cycles, such that no cycle of $\mathcal{C}$ crosses a cycle outside $\mathcal{C}$. An inclusionwise minimal component without any oriented edge is called a *hurdle*. In the example of Figure 1, there is a single component which is not a hurdle.

The following operation establishes the correspondence between dream and reality in $BG(\pi)$ and $BG(\pi^{-1})$. Let $(BG(\pi))^{-1}$ be the graph resulting from applying the following transformations to $BG(\pi)$:

- change each vertex label i^a into i^d and i^d into i^a whenever $\pi^{-1}[i]$ is negative;
- change each vertex label $(\pi[i])^a$ into i^a, and $(\pi[i])^d$ into i^d;
- change dream into reality and reality into dream.

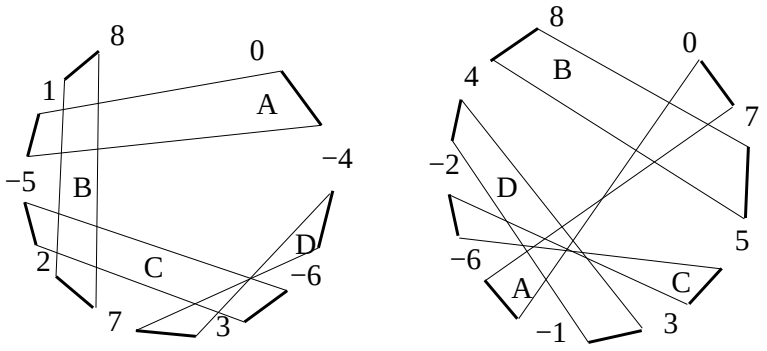

Figure 2: A breakpoint graph and its inverse. The correspondence of the cycles in shown.

The result of such a transformation applied to the example of Figure 1 is given in Figure 2.

Lemma 1 $(BG(\pi))^{-1} = BG(\pi^{-1})$.

Proof. By definition, $(BG(\pi))^{-1}$ and $BG(\pi^{-1})$ have the same vertex set. There is a reality edge $(\pi[i])^d(\pi[i+1])^a$ in $BG(\pi)$ for all $i \in \{0, \ldots, n\}$. In $(BG(\pi))^{-1}$, it becomes a dream edge from i^d if $\pi[i]$ is positive or from i^a if $\pi[i]$ is negative, to $(i+1)^d$ if $\pi[i+1]$ is positive or to $(i+1)^a$ if $\pi[i+1]$ is negative. This corresponds exactly to the definition of a dream edge of $BG(\pi^{-1})$. Furthermore, there is a dream edge $i(i+1)$ in $BG(\pi)$ for all $i \in \{0, \ldots, n\}$. In $(BG(\pi))^{-1}$, it becomes a reality edge from $(\pi^{-1}[i])^d$ to $(\pi^{-1}[i+1])^a$.

Therefore, $(BG(\pi))^{-1}$ and $BG(\pi^{-1})$ have the same sets of dream and reality edges. □

Observe that as a consequence, a cycle of π is also a cycle of π^{-1}.

2.2 Sorting simple permutations

A *simple permutation* is a permutation whose breakpoint graph contains only cycles of size 1 and 2 (called 1-cycles and 2-cycles). When dealing with simple permutations, we use "cycle" to mean 2-cycle, and "adjacency" to mean 1-cycle. Simple permutations are worth studying because of the following result.

Lemma 2 *[3] For any permutation π, there is a simple permutation π' such that $d(\pi) = d(\pi')$, and it is possible to deduce an optimal solution S for sorting π from any optimal solution S' for sorting π'. Transformations from π to π', and from S' to S are achievable in linear time.*

Let π be a simple permutation. The number of (2-)cycles is denoted by $c(\pi)$. By Lemma 1, $c(\pi) = c(\pi^{-1})$. The following is an easy but useful remark.

Lemma 3 *For any reversal ρ, $\rho = \rho^{-1}$, and if $\pi\rho_1 \cdots \rho_k = Id$, where $\rho_1, \ldots, \rho_k$ are reversals, then $\pi^{-1} = \rho_1 \cdots \rho_k$, $\pi^{-1}\rho_k \cdots \rho_1 = Id$, and $\rho_1 \cdots \rho_k \pi = Id$.*

Hannenhalli and Pevzner [3] proved the following fundamental result, which is the basis of the theory for sorting by reversals. We restrict ourselves to permutations without hurdles, because the best algorithms all begin with a procedure to clear hurdles in linear time, and we have nothing to add to this procedure. Therefore, we suppose hurdles have already been cleared.

Lemma 4 *[3] If π is a simple permutation without hurdles, $d(\pi) = c(\pi)$.*

Any optimal solution has to break one cycle at each step, and create two adjacencies. In consequence, the set of reversals of any optimal sequence is in bijection with the set of cycles of π. In the sequel, we use the same notation for reversals and cycles of π. For instance, we consider the permutation $\pi\rho$, for ρ a cycle of π. Of course, if application of the reversal ρ breaks the cycle ρ, the latter is no longer a cycle in $\pi\rho$.

2.3 Contraction and removal of cycles

In what follows, permutations are always considered to be both without hurdles and simple (the appropriate linear time tranformations are described in [2, 3, 6, 4]). Two edges are said to have the same orientation if they are both oriented or both unoriented.

Lemma 5 *In any cycle of a permutation, the two dream edges have the same orientation, and the two reality edges have the same orientation.*

Proof. Suppose two dream edges e, f of a same cycle had a different orientation, say e joins two departures, and f joins a departure and an arrival. The cycle would therefore have three departures and one arrival, which is impossible since reality edges always join one departure and one arrival. All the similar and dual cases are treated in the same way. □

A cycle is said to be *contractible* if its dream edges are oriented. In the example of Figure 1, cycle D is contractible as both $(3,4)$ and $(6,7)$ are oriented dream edges.

A cycle is said to be *removable* if its reality edges are oriented. In the example of Figure 1, cycles A and C are removable because, respectively, $(0,4)$ and $(5,1)$, $(6,3)$ and $(2,5)$, are oriented reality edges. Cycle B is neither removable nor contractible as none of its edges is oriented.

It is straightforward from the definitions and from Lemma 1 that a cycle ρ is removable in π if and only if it is contractible in π^{-1}. Indeed, if a reality edge is oriented in $BG(\pi)$, it becomes an oriented dream edge in $BG(\pi)^{-1} = BG(\pi^{-1})$, and vice-versa. The following lemma is another important result in the theory of sorting by reversals.

Lemma 6 *[3] A cycle ρ of π is contractible if and only if $c(\pi\rho) = c(\pi) - 1$.*

Observe that this does not mean that $d(\pi\rho) = d(\pi) - 1$, because $\pi\rho$ may have hurdles, and in this lies all the difficulty of sorting by reversals. From this and from Lemmas 1 and 3, we deduce immediately:

Lemma 7 *A cycle ρ of π is removable if and only if $c(\rho\pi) = c(\pi) - 1$.*

Let ρ be a cycle of π, with reality edges $(\pi[i])^d(\pi[i+1])^a$ and $(\pi[j])^d(\pi[j+1])^a$ (suppose $i < j$). Let $BG(\pi)/\rho$ be the graph obtained from $BG(\pi)$ by:

1. deleting the two reality edges of ρ, and replacing them by two edges parallel to the two dream edges of ρ;
2. changing the vertex label $(\pi[r])^a$ into $(\pi[r])^d$ and tthe vertex label $(\pi[r])^d$ into $(\pi[r])^a$ for every $r \in [i+1, j]$.

We call this operation the *contraction* of ρ in $BG(\pi)$.

Lemma 8 *[3] If a cycle ρ of a permutation π is contractible, then $BG(\pi)/\rho = BG(\pi\rho)$.*

It is possible to write a similar "dual" lemma for removability. Let $BG(\pi) \setminus \rho = (BG(\pi)^{-1}/\rho)^{-1}$. We call this operation the *removal* of ρ in $BG(\pi)$. It is clear that it consists in deleting the two dream edges of ρ, replacing them by two edges parallel to the two reality edges of ρ (thus obtaining two adjacencies), and changing the labels r of the vertices for every $r \in [|\pi[i+1]|, |\pi[j]|]$.

Lemma 9 *If a cycle ρ of a permutation π is removable, then $BG(\pi) \setminus \rho = BG(\rho\pi)$.*

The proof is straightforward from the previous Lemma together with Lemma 1. Contracting a contractible cycle in a breakpoint graph corresponds therefore exactly to applying the corresponding reversal to the permutation. In a similar way, removing a removable cycle corresponds to applying the corresponding reversal to the inverse of the permutation. In other words, contracting a cycle is changing the reality to make it closer to your dreams, and removing a cycle is changing your dreams to make them closer to the reality.

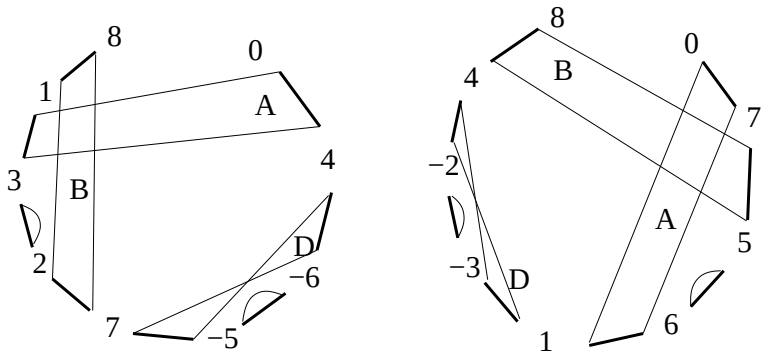

Figure 3: Removal of a cycle in the permutation of Figure 2, and contraction of the same cycle in the inverse permutation.

The following facts about contraction and removal can easily be verified.

Lemma 10 *Let ρ, ρ' be two distinct cycles in π. If ρ is contractible, then ρ' is removable in π if and only if it is removable in $\pi\rho$. Conversely, if ρ is removable, ρ' is contractible in π if and only if it is contractible in $\rho\pi$. Moreover, if ρ is removable in π, then two edges cross in π if and only if they cross in $\rho\pi$.*

Proof. Let $(\pi[r])^d(\pi[r+1])^a$ be a reality edge of ρ' in π. If r is in the interval of the reversal ρ, then $(\pi[r+1])^d(\pi[r])^a$ is a reality edge in $\pi\rho$, and both $\pi[r]$ and $\pi[r+1]$ changed sign. Otherwise, $(\pi[r])^d(\pi[r+1])^a$ is still a reality edge in $\pi\rho$, and $\pi[r]$ and $\pi[r+1]$ have same sign. In both cases, the edge preserves the same orientation. Consequently, ρ' is removable in π if and only if it is removable in $\pi\rho$. The dual statement is deducible from Lemmas 1 and 9.

Finally, we prove the last statement: let $\pi[i]\pi[i+1]$ and $\pi[j]\pi[j+1]$ (suppose $i < j$) be two reality edges of π. They are the dream edges $i(i+1)$ and $j(j+1)$ in π^{-1} according to Lemma 1. They cross in π^{-1} if $|\pi(i)| < |\pi(j)| < |\pi(i+1)| < |\pi(j+1)|$. Apply a reversal on a cycle of π where these edges are not involved. The edges stay with the same numbers (in absolute value) at their extremities (even if the labels may have changed), and the relation of their order is conserved. They are therefore crossing in $(\pi\rho)^{-1} = \rho\pi^{-1}$ if and only if they are crossing in π^{-1}. □

3 Constructing the sequence of reversals

We can sort each oriented component of π separately. We therefore suppose without loss of generality that there is only one component in $BG(\pi)$. We call a *valid sequence* of a permutation π an ordering of a subset of it 2-cycles $\rho_1, \ldots, \rho_k$, such that for all $i \in \{1, \ldots, k\}$, ρ_i is a removable cycle of $\rho_{i-1} \ldots \rho_1\pi$. In other words, $\rho_1, \ldots, \rho_k$ is valid if $c(\rho_k \cdots \rho_1\pi) = c(\pi) - k$.

A valid sequence is said to be *maximal* if no cycle of $\rho_k \cdots \rho_1\pi$ is removable. It is *total* if $k = c(\pi)$, that is if $\rho_k \cdots \rho_1\pi = Id$.

The algorithm for sorting a simple permutation is the following:

1. compute a maximal valid sequence of π;
2. increase the size of the sequence by adding some cycles inside it while it is not total.

Observe that the above algorithm constructs a sequence of removable instead of contractible cycles. The latter would seem to be a more direct way of doing the same thing. It is probably possible to do so by directly contracting instead of removing cycles (that is, by sorting π instead of π^{-1}). However, we believe the proofs are much easier in this case because the structure of the breakpoint graph is much more stable after a removal than after a contraction.

The first step of the algorithm consists in simply detecting removable cycles and removing them while there exist removable cycles in the result. We now concentrate on the second step, which consists in, given a maximal valid sequence $\rho_1, \ldots, \rho_k$, adding some cycles to it.

We prove the following result:

Theorem 1 *If S is a maximal but not total valid sequence of reversals for a permutation π, there is a nonempty sequence S' of reversals such that S may be split into two parts $S = S_1, S_2$, and S_1, S', S_2 is a maximal valid sequence of reversals for π.*

Proof. Let $S = \rho_1, \ldots, \rho_k$ be a maximal valid sequence of π. Let $\mathcal{C}$ be the set of cycles of the permutation $\rho_k \cdots \rho_1 \pi$ (it is composed of all the cycles of π minus the cycles in S). The set $\mathcal{C}$ is a union of unoriented components (all reality edges are unoriented else there would remain a removable cycle, therefore all have same orientation making all dream edges unoriented). Since there is only one component in $BG(\pi)$, one cycle of S has to cross one cycle of $\mathcal{C}$. Choose such a cycle ρ_l in S, such that l is maximum for this property. Let $S_1 = \rho_1, \ldots, \rho_{l-1}$ and $S_2 = \rho_l, \ldots, \rho_k$. These will be the way of splitting S in two, as described in the theorem. Let $\pi_1 = \rho_{l-1} \cdots \rho_1 \pi$ (this is the permutation obtained by removing the cycles of S_1 in π).

We first prove some lemmas.

Lemma 11 *At least one cycle of each component of $\mathcal{C}$ crossed by ρ_l is removable in π_1.*

Proof. Since a removal does not change the orientation of the dream edges (Lemma 10), and $\mathcal{C}$ is unoriented in $\rho_k \cdots \rho_1 \pi$, no cycle of $\mathcal{C}$ is contractible in π or in π_1. Let ρ be a cycle of $\mathcal{C}$ intersecting ρ_l in π_1. Let $\pi_1[i]\pi_1[i+1]$ be a reality edge of ρ_l. Since ρ_l is removable, the reality edge is oriented and $\pi_1[i]$ and $\pi_1[i+1]$ have opposite signs. Let $\pi_1[j]\pi_1[j']$ $(j < j')$ be a dream edge of ρ crossing ρ_l. We have that $j < i$ and $j' > i+1$. Since $\pi_1[j]\pi_1[j']$ is unoriented, $\pi_1[j]$ and $\pi_1[j']$ have the same sign. Among the reality edges $(\pi_1[r])^d(\pi_1[r+1])^a$ for $r \in [j, j']$, there is therefore a positive even number of oriented ones (there has to be an even number of changes of sign, and at least one between i and $i+1$). Thus at least one removable cycle distinct from ρ_l has to cross ρ. By the maximality of l, this cycle is in $\mathcal{C}$. □

Let $S' = \rho'_1, \ldots, \rho'_m$ be a valid sequence of cycles of $\mathcal{C}$ in π_1, such that $\mathcal{C} \setminus \{\rho'_1 \ldots \rho'_m\}$ contains no removable cycle in $\rho'_m \ldots \rho'_1 \pi_1$.

Lemma 12 *The cycle ρ_l is removable in $\rho'_m \cdots \rho'_1 \pi_1$.*

Proof. Let $\pi' = \rho'_{m-1} \cdots \rho'_1 \pi_1$. Let $\pi'[i]\pi'[i+1]$ and $\pi'[j]\pi'[j+1]$ be the two reality edges of ρ'_m in π'. Let M be the number chosen among $\pi'[i], \pi'[i+1], \pi'[j], \pi'[j+1]$ with highest absolute value, and m be the number with lowest absolute value. Since ρ'_m is removable in π', $\pi'[i]$ and $\pi'[i+1]$ have opposite signs, and $\pi'[j]$ and $\pi'[j+1]$ also have opposite signs. Recall that no cycle of $\mathcal{C}$ is contractible in π, so none is in π', from Lemma 10. Then the two dream edges $\pi'[i]\pi'[j+1]$ and $\pi'[j]\pi'[i+1]$ of ρ'_m are unoriented. In consequence $\pi'[i]$ and $\pi'[j+1]$ have the same sign, and $\pi'[j]$ and $\pi'[i+1]$ also have the same sign. Note that $\pi'[j+1] = \pi'[i] + 1$ and $\pi'[i+1] = \pi'[j] + 1$, so M and m cannot be adjacent in the graph, and they have opposite signs. Now remove cycle ρ'_m from π'. Removing a cycle is deleting the dream edges and replacing them by two edges parallel to the reality edges, and changing the labels of the vertices of the graph whose absolute values are between $|m|$ and

$|M|$, excluding m and M themselves (see Lemma 9). In the obtained graph, both M and m participate in two adjacencies, and they have opposite signs. So there is an odd number of changes of signs between M and m, therefore an odd number of oriented reality edges. Since oriented reality edges belong to a removable 2-cycle (they only go by pairs), at least one of these 2-cycles crosses ρ'_m in π'. Since there is no removable cycle of $\mathcal{C}$ in $\rho'_m\pi'$ (by hypothesis), and ρ_l is the only cycle outside $\mathcal{C}$ that may cross a cycle of $\mathcal{C}$, this removable cycle is ρ_l. □

Lemma 13 *The sequence S_1, S', S_2 is a valid sequence of π.*

Proof. We already know that $\rho_1, \ldots, \rho_{l-1}, \rho'_1, \ldots, \rho'_m, \rho_l$ is a valid sequence, by Lemma 12. Let now $\pi_1 = \rho_l \cdots \rho_1\pi$, and $\pi_2 = \rho_l\rho'_m \cdots \rho'_1\rho_{l-1} \ldots \rho_1\pi$. Let D_1 be the component of $BG(\pi_1)$ induced by the cycles $\rho_{l+1}, \ldots, \rho_k$ in π_1. Since each component of a breakpoint graph may be sorted separately, we can say that $\rho_{l+1}, \ldots, \rho_k$ is a total valid sequence for D_1. Let now D_2 be the component of $BG(\pi_1)$ induced by the cycles $\rho_{l+1}, \ldots, \rho_k$ in π_2. A cycle ρ_i is contractible in D_1 if and only if it is contractible in D_2, and two cycles cross in D_1 if and only if they cross in D_2 (they differ only by some removals, which do not change contractibility, nor "crossingness"). Then $\rho_{l+1}, \ldots, \rho_k$ is a total valid sequence for D_2. Finally, $\rho_1, \ldots, \rho_{l-1}, \rho'_1, \ldots, \rho'_m, \rho_l, \ldots, \rho_k$ is valid in π. □

This concludes the proof of Theorem 1 because we have a valid sequence S_1, S', S_2, and S' is non empty (see Lemma 11). □

In the example of Figure 1, with the labels of the cycles as indicated in Figure 2, the algorithm may work as follows: cycle C is removed, then cycle D. The sequence C, D is maximal since A and B are not removable in $DC\pi$. It is not total, so we must find the last cycle in the ordered sequence $\{C, D\}$ which crosses a cycle in $\{A, B\}$. This is cycle C which crosses B (C is thus ρ_l, $S_1 = \emptyset$ and $S_2 = \{C, D\}$). In $BG(\pi) \setminus \{C, D\}$, only A is removable in π. We therefore remove A. This makes B removable and we remove it. There are no more removable cycles in $BG(\pi) \setminus \{C, D\}$ (indeed, there are no more cycles), so A, B, C, D is a total valid sequence and $DCBA\pi = Id$.

Observe that in the case of this example, by working directly on π^{-1}, it is easy to find that D, C, B, A is a total valid sequence.

4 Complexity

With a classical data structure, applying a reversal and picking a contractible (or removable) cycle is achievable in linear time. As a consequence, any algorithm that has to apply a reversal at each step, even bypassing the search for a safe reversal, will run in $O(n^2)$. This is the case for our algorithm. In practice, even an $O(n^2)$ implementation should be an improvement on the existing algorithms, and we are not sure that an implementation of the data structure we describe next leads indeed to a practical improvement. However, it is an important mathematical question whether sorting a signed permutation by reversals can be

done in a theoretical subquadratic time complexity. We therefore use a more sophisticated implementation to make it run *exactly* in $O(n\sqrt{n \log n})$.

In 2003, Kaplan and Verbin [5] invented a clever data structure which allows to pick a contractible reversal and apply it in sublinear time. We use the same data structure, just adding some flags in order to be able to perform all the additional operations we need. Furthermore, we use the structure to represent the permutation π^{-1} instead of π. Indeed, we are looking for removable cycles in π, that is for contractible ones in π^{-1}. As in [5], we split the permutation into blocks of size $O(\sqrt{n \log n})$. We assign a flag to each block, turned "on" if the block should be read in the reverse order, changing the sign of the elements. We also add to each block a balanced binary tree (such as a splay tree for instance), where the elements of the blocks are stored in the nodes, sorted by increasing position of their successor in π^{-1}. This means that $\pi^{-1}[i]$ is before $\pi^{-1}[j]$ if $\pi(\pi^{-1}[i] + 1) < \pi(\pi^{-1}[j] + 1)$. Each node of the tree corresponds to a dream edge of π^{-1} (each 2-cycle is represented by two nodes). Each node keeps the information on the orientation of the associated dream edge (that is, on the contractibility and therefore also on the removability in π of the cycle in which the edge participates), and a "running total" storing the number of oriented edges in the subtree.

Up to this point, we have described exactly the data structure of Kaplan and Verbin. We now add some flags and numbers to the nodes of the trees in order to perform our own queries. Let $\mathcal{C}$ be a subset of the 2-cycles of π^{-1}. At the beginning, $\mathcal{C}$ is all the 2-cycles in $BG(\pi)$. To each node, a new flag is added, turned "on" if the corresponding edge is in a cycle of $\mathcal{C}$, and a "running total" stores the number of oriented edges that belong to $\mathcal{C}$ in the subtree. Figure 4 gives a representation of the data structure applied to the inverse permutation of Figure 2 (right side), split in two blocks. Observe that we do not need to know the number of all oriented edges, only of those which are in $\mathcal{C}$.

In this way, the following queries are achievable in the same way, and same complexity, as in [5].

Lemma 14 *[5] It is possible to know in constant time whether there is a removable cycle in π, and to pick one such cycle in time $O(\log n)$.*

With the additional flags, and proceeding exactly in the same way, we have that:

Lemma 15 *It is possible to know in constant time whether there is a removable cycle of π in $\mathcal{C}$, and if there is any, to pick one such cycle in time $O(\log n)$.*

Lemma 16 *It is possible to contract a cycle and maintain the data structure in time $O(\sqrt{n \log n})$.*

Proof. All we have to maintain as information besides what was done in [5] is whether an edge is in $\mathcal{C}$. In order to do this, we just need to remove an edge from $\mathcal{C}$ when it is contracted and then to update the running totals. The number of elements of $\mathcal{C}$ is thus decreased by one

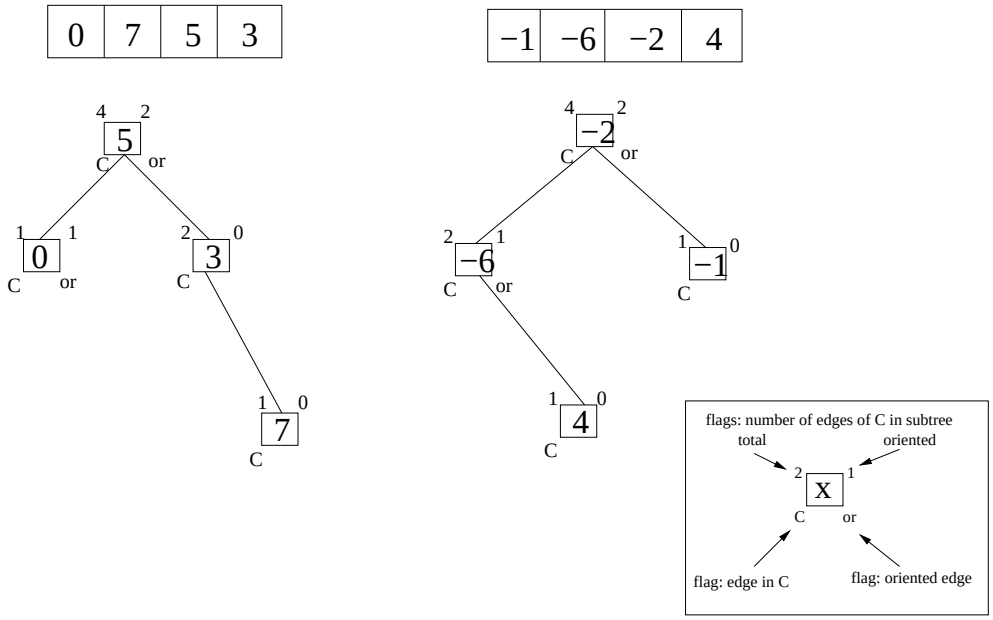

Figure 4: The data structure for the permutation of Figure 2 (right side) at the beginning of the algorithm (all edges are in $\mathcal{C}$).

in all the blocks containing the contracted cycle, and we calculate the difference between the number of elements in $\mathcal{C}$ and the former number of contractible cycles in $\mathcal{C}$ at each flipped block. □

Figure 5 represents the data structure at a later step in the algorithm. As described before, the algorithm picks cycles C and D, removes them, and since there is no other removable cycle, replaces them. The result of this is shown in the figure.

Theorem 2 *There exists an algorithm which finds a total sorting sequence in time* $O(n\sqrt{n \log n})$.

Proof. We first find a maximal valid sequence $\rho_1, \ldots, \rho_k$. This takes $O(k\sqrt{n \log n})$ time. If the sequence is total, we are done. Otherwise, we are left with a set $\mathcal{C}$ of unremoved cycles.

We must then identify ρ_l in the sequence such that there is a removable cycle in $\mathcal{C}$ in $\rho_{l-1} \ldots \rho_1 \pi$, with l maximum for this property. This can be done by putting back (applying the reversals again) one by one the removed cycles $\rho_k, \ldots, \rho_l$, checking at each step (in constant time) if there is a removable cycle in $\mathcal{C}$, and stopping as soon as one is found. This is achievable in time $O((k-l)\sqrt{n \log n})$. The sequence $\rho_k, \ldots, \rho_l$ is a sequence of *safe* contractible reversals for π^{-1}, and is fixed once for ever. It will represent the last reversals of the final sequence. It is not modified nor looked at anymore in the algorithm, and this allows to maintain the complexity.

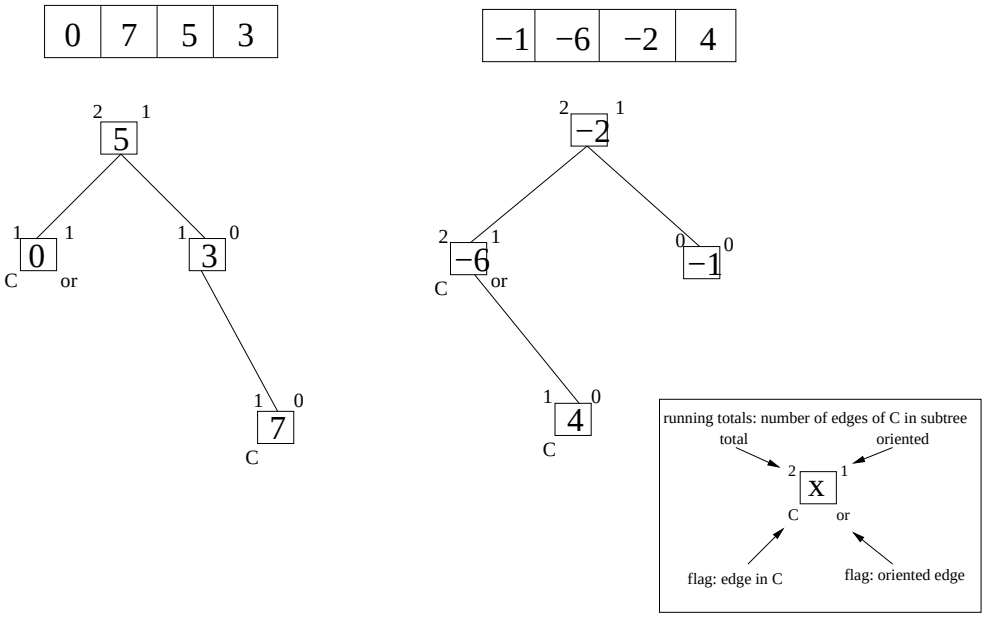

Figure 5: The data structure for the permutation of Figure 2 (right side) at a later step of the algorithm, when cycles C and D have been removed, and then replaced (but remain removed from $\mathcal{C}$).

We then have to remove, while there are any, removable cycles in $\mathcal{C}$. This leads to the sequence $\rho_{k+1}, \ldots, \rho_{k'}$ which we insert after the sequence $\rho_1, \ldots, \rho_{l-1}$, while at the same time deleting it from $\mathcal{C}$ in the same way we have already seen. If $\rho_1, \ldots, \rho_{l-1}, \rho_{k+1}, \ldots, \rho_{k'}, \rho_l, \ldots, \rho_k$ is total, we are done. Otherwise, we start again replacing the removed cycles one by one in the reverse order but starting this from k' (the sequence $\rho_l, \ldots, \rho_k$ is not touched anymore). We do so until there is a removable cycle in the new $\mathcal{C}$, and start again while $\mathcal{C}$ is not empty.

Each cycle in the sequence is removed once, and replaced at most once in the whole procedure. The total complexity is then $O(n\sqrt{n \log n})$. □

The remaining bottleneck of the new algorithm is now the complexity at each step of applying a reversal to a permutation and of keeping the set of contractible or removable cycles. This is what has to be improved in the future if one wishes to obtain a lower theoretical complexity for sorting a signed permutation by reversals.

References

[1] D.A. Bader, B.M.E. Moret and M. Yan, "A linear-time algorithm for compting inversion distance between signed permutations with an experimental study", *Proceedings of the 7th Workshop on Algorithms and Data Structures* (2001), 365–376.

[2] P. Berman and S. Hannenhalli, "Fast sorting by reversals", *Proceedings of the 7th Symposium on Combinatorial Pattern Matching* (1996), Lecture Notes in Computer Science, vol. 1075, 168–185.

[3] S. Hannenhalli and P. Pevzner, "Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals", *Proceedings of the 27th ACM Symposium on Theory of Computing* (1995), 178–189.

[4] H. Kaplan, R. Shamir and R.E. Tarjan, "Faster and simpler algorithm for sorting signed permutations by reversals", *SIAM Journal on Computing* 29 (1999), 880-892.

[5] H. Kaplan and E. Verbin "Efficient data structures and a new randomized approach for sorting signed permutations by reversals", *Proceedings of the 14th Symposium on Combinatorial Pattern Matching* (2003), Lecture Notes in Computer Science, vol. 2676, 170–185.

[6] M. Ozery-Flato and R. Shamir, "Two notes on genome rearrangement", *Journal of Bioinformatics and Computational Biology*, 1 (2003), 71–94.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399