Chapter 3

Sorting by reversals

Summary

The classical algorithmic problems in pairwise comparative genomics are to compute the rearrangement distance between two genomes [33], that correspond to the minimum number of rearrangement events that are required to transform one genome into the other, and to determine an optimal sequence of events to transform one genome into the other. These problems have several variations, according to the events that may be considered [63].

Our research is mostly focused on rearrangement problems restricted to reversal events and, in this chapter, we talk about sorting one unichromosomal genome into another by reversals when gene duplications and insertions are not allowed. Observe that we also assume that the order of the genes is known in both genomes, which often is not true in practice [66]. One of the first studies that proposed algorithms to compute the reversal distance between two genomes was developed by Kececioglu and Sankoff [38], with an approach that does not take into account the orientation of the genes. Later this approach, called *unsigned sorting by reversals*, was proven to lead to an NP-hard problem [22]. We worked on a different approach, called *signed sorting*

by reversals, or simply *sorting by reversals*, in which the orientation of the genes is taken into account. Kececioglu and Sankoff [38] had already observed that some aspects of signed sorting by reversals were easier to analyze, and, indeed, this approach can be solved in polynomial time [32, 33], as we will describe in this chapter.

Despite the simplifications (not considering duplications or insertions and assuming that the order of the genes is known in both genomes) mentioned above, the sorting by reversals problem is very interesting. From the biological point of view, as we said before, reversals are frequently observed, specially in prokaryotes. And reversals are also interesting from the algorithmic point of view. First we note that it is always possible to sort a genome into another by reversals. In the worst case, we need two reversals to put each marker of the first genome in the position that it occupies in the second genome (one reversal to put the marker in the proper position and eventually a second reversal to inverse its orientation). Thus, if the two considered genomes has n homologous markers, in the worst case we need $2n$ reversals to sort one genome into the other. We will see later in this chapter that in general at most n reversals are sufficient to sort a genome into another and a fictitious example is given in Figure 6.

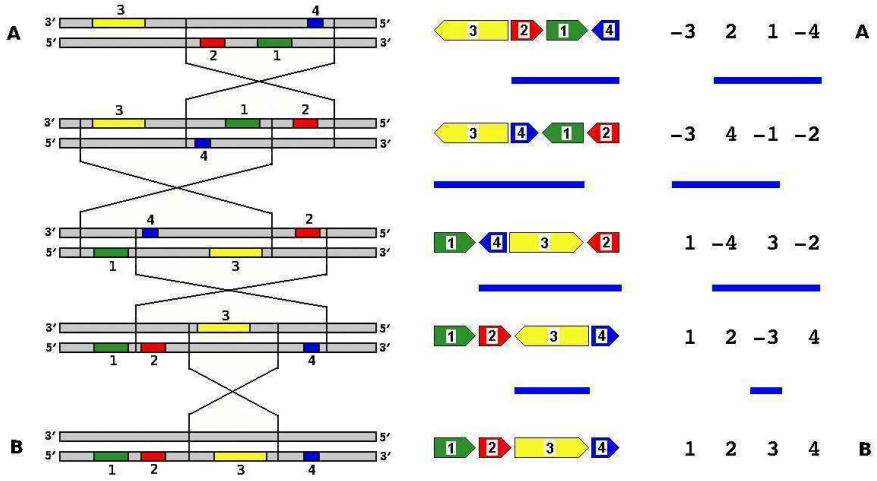

Figure 6: Sorting genome A into genome B by reversals only. Homologous markers (usually genes) are identified by the same numbers and colours. Signs indicate the DNA strand the markers lie on.

Computing the reversal distance, that is, the minimum number of reversals that are required to transform one genome into the other, and finding an optimal sorting sequence can be solved in polynomial time [32, 33]. These two problems have been the topic of several works. The fastest

algorithm to compute the distance takes $O(n)$ time [4] and the fastest way to find an optimal sorting sequence is subquadratic [11, 31, 63]. It is possible that this mathematical notion of reversal distance and the method of searching optimal sequences can underestimate the actual number of steps that occurred biologically. However, the solutions of these two problems are still valuable tools that help to analyze and to understand evolutionary scenarios. Currently, there are at least two available softwares to solve these problems. One is the package GRAPPA[3], that is discussed in more detail in [45] and contains the fastest algorithm to compute the reversal distance (mentioned above). The other is the software GRIMM[4], that is described in [64] and contains one of the most used programs to sort a genome into another by reversals. These programs were used in particular by Ross *et al.* [55] in the analysis of the human sexual chromosomes X and Y and by Blanc *et al.* [13] in the analysis of the *Rickettsia* bacteria.

Observe that with reversals we can simulate a transposition, that is another possible rearrangement event in unichromosomal genomes. A transposition is said to happen when two consecutive markers of a genome exchange their positions. It is always possible to produce the same result as a transposition with a sequence of three reversals (see Figure 7). Thus a sequence of m transpositions can always be transformed in a sequence of $3m$ reversals. However, this does not mean that there is a clear relation between the reversal distance and the transposition distance. Eventually a sequence of m transpositions can be replaced by a sequence with less than $3m$ reversals. Moreover, although the reversal distance can be obtained in polynomial time, the complexity of computing the transposition distance is still an open problem in the algorithmics of genome rearrangements [5].

In the rest of this chapter we will introduce our notation and explain the classical approach of Hannenhali and Pevzner [32, 33, 53] for the sorting by reversals problem.

[3]The package GRAPPA (Genome Rearrangements Analysis under Parsimony and other Phylogenetic Algorithms) contains several programs to deal with genome rearrangements and can be downloaded at http://www.cs.unm.edu/~moret/GRAPPA/.

[4]The software GRIMM contains also algorithms for multichromosomal genome rearrangements and is available online at http://grimm.ucsd.edu/GRIMM/.

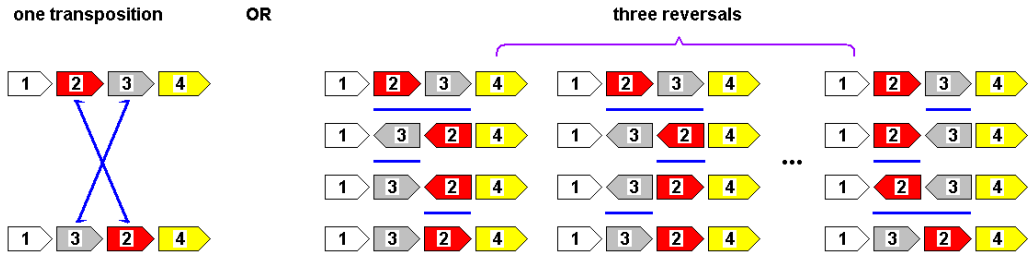

Figure 7: A transposition or a sequence of three reversals may produce the same rearrangement in a genome. Observe that the three reversals can be applied in different orders.

3.1 Permutations, intervals and reversals

We represent the studied genomes by the list of homologous markers (usually genes or blocks of contiguous genes) between them. These homologous genomic markers are represented by the integers $1, 2, \ldots, n$, with a plus or minus sign to indicate the strand they lie on. The order and orientation of the markers of one genome in relation to the other is represented by a *signed permutation* $\pi = (\pi_1, \pi_2, \ldots, \pi_{n-1}, \pi_n)$ of size n over $\{-n, \ldots, -1, 1, \ldots, n\}$, such that, for each value i from 1 to n, either i or $-i$ is mandatorily represented, but not both. The *identity permutation* $(1, 2, 3, \ldots, n)$ is denoted by $\mathcal{I}_n$.

A subset of numbers $\rho \subseteq \{1, 2, \ldots, n-1, n\}$ is said to be an *interval* of a permutation π if there exist $i, j \in \{1, \ldots, n\}$, $1 \leq i \leq j \leq n$, such that $\rho = \{|\pi_i|, |\pi_{i+1}|, \ldots, |\pi_{j-1}|, |\pi_j|\}$. Given a permutation π and an interval ρ of π, we can apply a *reversal* on the interval ρ of π, that is, the operation which reverses the order and flips the signs of the elements of ρ, denoted by $\pi \circ \rho$. If $\pi = (\pi_1, \pi_2, \ldots, \pi_{i-1}, \pi_i, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_j, \pi_{j+1}, \ldots, \pi_{n-1}, \pi_n)$ and $\rho = \{|\pi_i|, |\pi_{i+1}|, \ldots, |\pi_{j-1}|, |\pi_j|\}$,

$$\pi \circ \rho = (\pi_1, \pi_2, \ldots, \pi_{i-1}, -\pi_j, -\pi_{j-1}, \ldots, -\pi_{i+1}, -\pi_i, \pi_{j+1}, \ldots, \pi_{n-1}, \pi_n).$$

For example, with the permutation $\pi = (-3, 2, 1, -4)$ and the interval $\rho = \{1, 2, 4\}$ we have $\pi \circ \rho = (-3, 4, -1, -2)$. Due to this, an interval ρ can also be used to denote a reversal.

We say that a permutation is *linear* when it represents a linear chromosome, or *circular* when it represents a circular chromosome. When a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_{n-1}, \pi_n)$ is circular, the circular permutation $\overline{\pi} = (-\pi_n, -\pi_{n-1}, \ldots, -\pi_2, -\pi_1)$ (generated by a reversal over all values of π) and all circular permutations obtained by a *shift* in π or $\overline{\pi}$ are equivalent to π. A shift of i

elements in a circular permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_{n-i}, \pi_{n-i+1}, \pi_{n-i+2}, \ldots, \pi_{n-1}, \pi_n)$ transfers the last i elements of π to the beginning of π. This operation generates the circular permutation $(\pi_{n-i+1}, \pi_{n-i+2}, \ldots, \pi_{n-1}, \pi_n, \pi_1, \pi_2, \ldots, \pi_{n-i})$. Observe, for example, that the circular permutations $\pi = (-3, 2, 1, -4)$ and $\pi' = (-1, -2, 3, 4)$ are equivalent (we can obtain π' by applying a shift of 3 on $\overline{\pi}$).

For a given permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_{n-1}, \pi_n)$, we say that there is a *point* between each pair of consecutive values π_i and π_{i+1} in π. In addition, if π is circular, there is one additional point between π_n and π_1. If π is linear, there are two additional points, one before π_1 and the other after π_n. We denote by $pts(\pi)$ the number of points in a permutation π. Thus, if π is circular, then $pts(\pi) = n$. Otherwise π is linear and $pts(\pi) = n + 1$.

When we analyze a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_{n-1}, \pi_n)$ with respect to another permutation π_T, each point in π can be an *adjacency* or a *breakpoint*. We say that a pair of consecutive values (π_i, π_{i+1}) in π is an adjacency between π and π_T when either the values in the pair (π_i, π_{i+1}) or the values in the pair $(-\pi_{i+1}, -\pi_i)$ are consecutive in π_T. Moreover, if the permutations are circular, we assume that π_n is the last value of π_T[5], and the pair (π_n, π_1) is an adjacency when π_1 is the first value in π_T. If the permutations are linear, we have an adjacency before π_1 if π_1 is also the first value in π_T and an adjacency after π_n if π_n is also the last value of π_T. All points that are not adjacencies between π and π_T are called breakpoints. We denote by $adj(\pi)$ the number of adjacencies and by $brp(\pi)$ the number of breakpoints in a permutation π. It is easy to see that $brp(\pi) = pts(\pi) - adj(\pi)$. Observe that, if π is sorted, that is, $\pi = \pi_T$, then π has only adjacencies and no breakpoints, and, if $\pi \neq \pi_T$, then π has at least one breakpoint.

A *sequence* or $i-$*sequence* of reversals $\rho_1\rho_2\ldots\rho_i$ is valid for a permutation π if ρ_1 is an interval of π, ρ_2 is an interval of $\pi \circ \rho_1$, ρ_3 is an interval of $(\pi \circ \rho_1) \circ \rho_2$, and so on. If $\rho_1\rho_2\ldots\rho_i$ is a valid $i-$sequence of reversals for a permutation π, then $\pi \circ \rho_1\rho_2\ldots\rho_i$ denotes the consecutive application of the reversals ρ_1, ρ_2, $\ldots\rho_i$ in the order in which they appear. We say that an $i-$sequence of reversals $\rho_1\ldots\rho_i$ *sorts* a permutation π into a permutation π_T if $\pi \circ \rho_1\ldots\rho_i = \pi_T$.

The length of a shortest sequence of reversals sorting a permutation π into π_T is called the *reversal distance* of π and π_T, and is denoted by $d(\pi, \pi_T)$. Let $s = \rho_1\rho_2\ldots\rho_i$ be a valid

[5]If the permutations are circular, without loss of generality, we can assume that the last value in π and π_T are the same; if it is not the case, we take as π an equivalent circular permutation with this characteristic.

$i-$sequence of reversals for a permutation π. If $d(\pi \circ s, \pi_T) = d(\pi, \pi_T) - i$, then s is said to be an *optimal $i-$sequence*. Moreover, if s is an optimal $i-$sequence and $i = d(\pi, \pi_T)$, then s is simply called an *optimal sorting sequence* for π and π_T. We also define the $k-$prefix of an optimal sorting sequence s as the sequence composed by the first k reversals of s. Observe that if s' is a $k-$prefix of an optimal sequence s sorting π into π_T, then $d(\pi \circ s', \pi_T) = d(\pi, \pi_T) - k$, that is, s' is an optimal $k-$sequence for π and π_T. For example, if we consider two linear permutations $\pi = (-3, 2, 1, -4)$ and $\pi_T = \mathcal{I}_4$, we have $d(\pi, \pi_T) = 4$ and one optimal sorting sequence is $\{1,2,4\}\{1,3,4\}\{2,3,4\}\{3\}$, whose $1-$, $2-$ and $3-$prefixes are $\{1,2,4\}$, $\{1,2,4\}\{1,3,4\}$ and $\{1,2,4\}\{1,3,4\}\{2,3,4\}$.

Henceforth we will generally use simply the term sequence or $i-$sequence to refer to an optimal sequence or optimal $i-$sequence of reversals. Moreover, for the purposes of our work, the initial and the target permutations π and π_T are either both linear, or both circular. Without loss of generality, we often omit the target permutation π_T. In this case, π_T corresponds to the identity permutation $\mathcal{I}_n = (1, 2, 3, \ldots, n)$, where n is the size of the initial permutation π, and the notation $d(\pi)$ is equivalent to $d(\pi, \mathcal{I}_n)$.

3.2 The breakpoint graph and the reversal distance

As mentioned, given a permutation π, calculating $d(\pi)$ and finding one optimal sequence of reversals sorting π can be computed in polynomial time. The classical approach for analyzing these two problems was developed by Hannenhalli and Pevzner [8, 32, 33, 53] and is based on a special structure called the *breakpoint graph*, whose edges can be *black* or *gray*.

For a given permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_{n-1}, \pi_n)$, we construct the breakpoint graph of π as follows. If π is linear, we may add the values 0 and $n+1$, that represent the extremities of the chromosome, obtaining the permutation $\pi' = (0, \pi_1, \pi_2, \ldots, \pi_{n-1}, \pi_n, n+1)$. If π is circular (without loss of generality we assume $\pi_n = n$), we may add only the value 0, obtaining the permutation $\pi' = (0, \pi_1, \pi_2, \ldots, \pi_{n-1}, n)$. Then we may link each pair of consecutive values by a horizontal black edge (each black edge represents a point in the permutation). Lastly, we may link with gray edges the first extremity of the black edge that appears after zero or a positive value i (analogously the last extremity of the black edge that appears before a negative value $-i$)

with the last extremity of the black edge that appears before a positive value $i+1$ (analogously the first extremity of the black edge that appears after a negative value $-(i+1)$). Thus, each gray edge links extremities of black edges. At the end, we have a graph with a collection of cycles, and in each cycle black and gray edges alternate. When a cycle contains only one black and one gray edge, it covers an adjacency and is called *trivial cycle*. The cycles that contain four or more edges cover at least two breakpoints and are called *long cycles*. The construction of the breakpoint graph of a linear permutation is illustrated in Figure 8 (A).

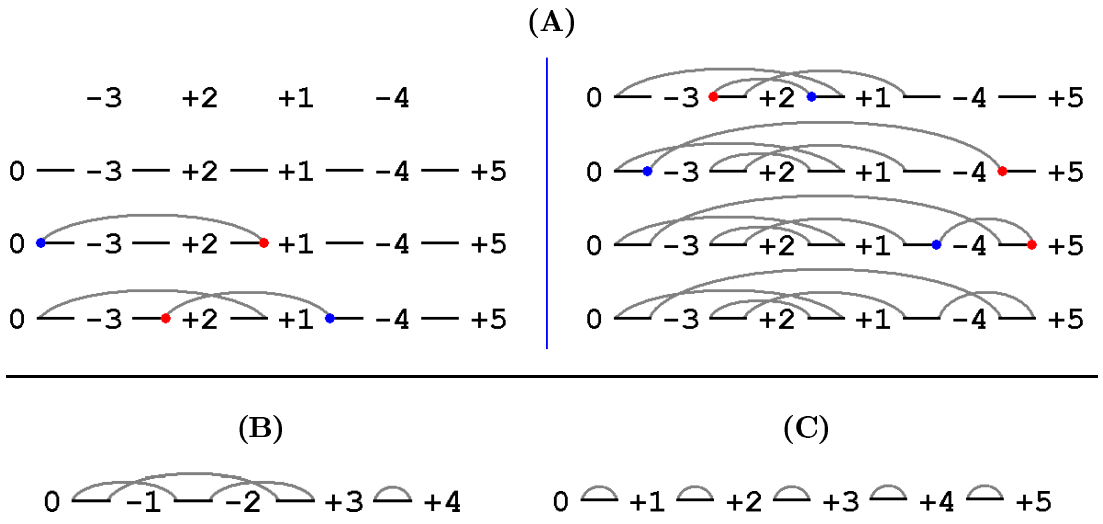

Figure 8: (A) The construction of the breakpoint graph for the linear permutation $\pi = (-3, 2, 1, -4)$ is done by the following steps: 1- add the values **0** *and* $+$**5**, *that represent the extremities of the chromosome; 2- link each pair of consecutive values by a black edge. 3- link with gray edges the first extremity of the black edge that appears after zero or a positive value i (analogously the last extremity of the black edge that appears before a negative value $-i$) with the last extremity of the black edge that appears before a positive value $i+1$ (analogously the first extremity of the black edge that appears after a negative value $-(i+1)$). The obtained breakpoint graph has one long cycle with five breakpoints and no adjacencies. (B) The breakpoint graph for the circular permutation $(-3, 2, 1, -4)$, which is equivalent to the circular permutation $(-1, -2, 3, 4)$. In this case, in the first step we may add only the value* **0** *in the beginning of $(-1, -2, 3, 4)$, henceforth the procedure is identical. This graph has two cycles: one trivial cycle (which correspond to the adjacency between 3 and 4) and one long cycle with three breakpoints. (C) The breakpoint graph for the linear permutation $\mathcal{I}_4 = (1, 2, 3, 4)$. This graph has five trivial cycles (each trivial cycle is an adjacency) and no breakpoints.*

Observe that, for a given permutation π, the breakpoint graph is different depending on whether π is linear or circular, as we can see comparing the graph for the linear permutation $(-3, 2, 1, -4)$ and the circular permutation $(-3, 2, 1, -4)$ (Figure 8 (A) and (B)). However, they can be analyzed exactly in the same way, that is, the only difference between circular and linear

permutation analyses is the breakpoint graph construction. Thus, without loss of generality, henceforth we will often talk about breakpoint graphs, without specifying whether the corresponding permutations are linear or circular. To denote the breakpoint graph of a permutation π, we will use the same symbol π.

If a permutation π is sorted, it has only adjacencies, and the resulting breakpoint graph is a collection of $pts(\pi)$ trivial cycles (see Figure 8 (C)). A breakpoint graph that has only trivial cycles is said to be sorted. Since a long cycle contains at least two breakpoints, if π is unsorted, then π has at most $pts(\pi) - 1$ cycles. This indicates that, in order to sort a permutation, we may induce an increase of the number of cycles in its corresponding breakpoint graph. The number of cycles in the breakpoint graph of a permutation π is denoted by $cyc(\pi)$.

Hannenhalli and Pevzner [32, 33, 53] described the effects of a reversal ρ over a breakpoint graph π. The authors demonstrated that a reversal ρ is either a *split reversal*, that increases the number of cycles by one, (in this case we have $cyc(\pi \circ \rho) = cyc(\pi) + 1$), or a *joint reversal*, that decreases the number of cycles by one (in this case we have $cyc(\pi \circ \rho) = cyc(\pi) - 1$), or a *neutral reversal*, that maintains the number of cycles unchanged (in this case we have $cyc(\pi \circ \rho) = cyc(\pi)$). In order to characterize these three types of reversals, we assign a direction to each black edge, according to an arbitrary tour in each cycle of the graph. Then, if the extremities of the reversal are in black edges in the same cycle and have opposite directions, we have a split reversal. If the extremities of the reversal are in black edges in different cycles, we have a joint reversal (independently of the directions of the black edges). Finally, if the extremities of the reversal are in black edges in the same cycle and have the same direction, we have a neutral reversal that does not change the number of cycles in the graph. To understand the reasons of these effects, we should investigate how the reversals affect the topology of the graph. In fact, only the two black edges that correspond to the extremities of the reversal are modified. Although some vertices may also have their corresponding values inversed, all the other edges in paths that alternate gray and black edges remain unchanged (consequently, their relative directions remain also unchanged). Figure 9 illustrates the three types of reversals.

In order to sort a permutation, we must maximize the number of split reversals in the sorting sequence. With this information, we can start to conceive the formula for the reversal distance. If we can find a sequence s that has only split reversals for sorting a breakpoint graph π, the

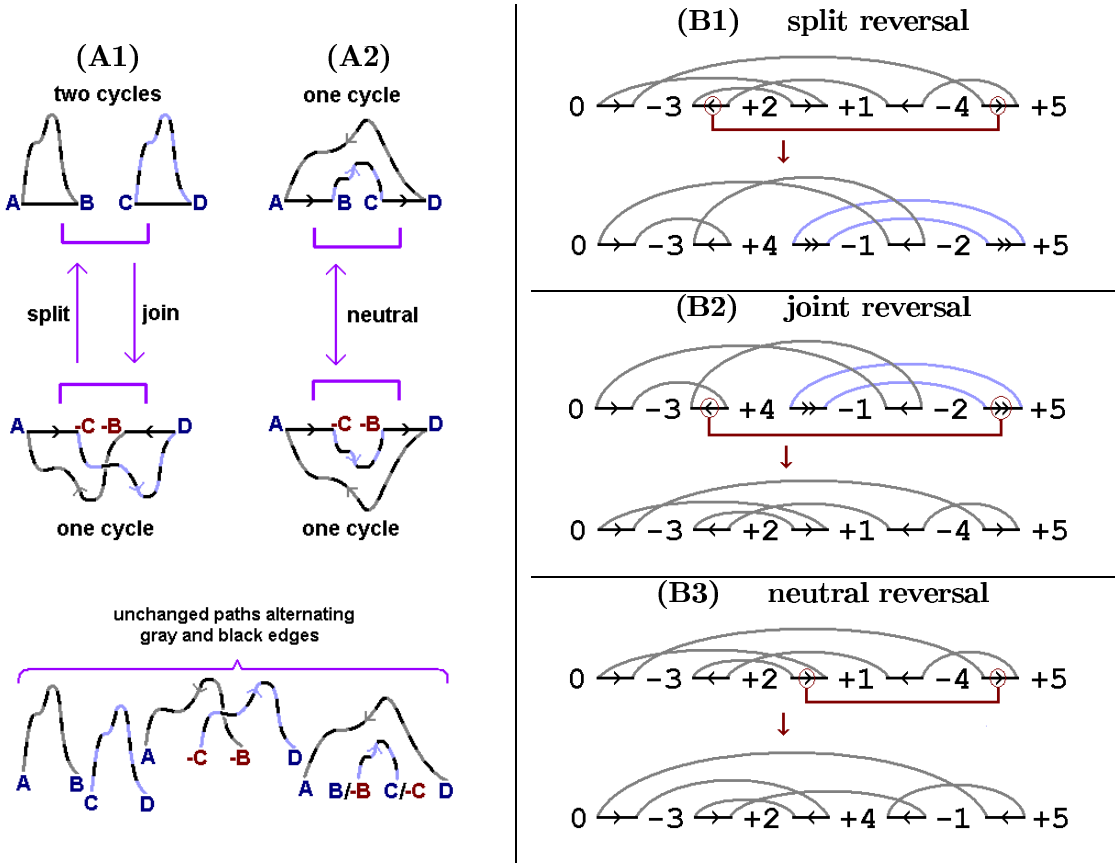

Figure 9: The effects of a reversal over the breakpoint graph. We may assign a direction to each black edge, by an arbitrary tour in each cycle of the graph. The images A1 and A2 illustrate how a reversal affects the topology of the graph. The point A,B (respectively A,-C) appears before the point C,D (respectively -B,D) in the considered permutations. Observe that, with respect to the topology, only the two black edges that correspond to the extremities of the reversal are modified. All the other edges in paths that alternate gray and black edges remain unchanged, although the vertices that are between B and C in the permutation must have their corresponding values inversed. (A1) The two cycles on the top are joined by a reversal whose extremities are in the represented black edges. Inversely, the unique cycle on the botton is split by a reversal whose extremities are in the represented black edges, that have opposite directions. (A2) The number of cycles in the graph is not changed by a reversal whose extremities are in black edges in the same cycle, with the same direction. The images B1, B2 and B3 show the effects over the breakpoint graphs represented in the standard form. (B1) Split reversal: a reversal whose extremities are in black edges in the same cycle and opposite directions may break the cycle in two. (B2) Joint reversal: A reversal whose extremities are in black edges in different cycles may join the two cycles in one (independently of the directions of the black edges). (B3) Neutral reversal: a reversal whose extremities are in black edges in the same cycle and same directions does not change the number of cycles in the graph.

length of s is $pts(\pi) - cyc(\pi)$. However, a split reversal does not always exist. For example, if all black edges of all cycles in the graph have the same direction, we cannot perform a split reversal (Figure 10 (A)). Thus, in some cases, we may need to add some joint and/or neutral reversals in a sorting sequence, and the reversal distance is $d(\pi) \geq pts(\pi) - cyc(\pi)$.

Fortunately, it is always possible to calculate the number of non-split reversals in a sorting sequence. We can define an exact formula to the reversal distance, but first we need to define other properties of the breakpoint graph. When a cycle in the graph has black edges with opposite directions, it is called an *oriented cycle*. Otherwise all black edges in the cycle have the same direction and we have an *unoriented cycle*. A component of the graph is a collection of cycles, such that each cycle of the component has at least one gray edge that overlaps with a gray edge of another cycle in the component. Adjacencies are trivial components, and a non-trivial component contains at least two breakpoints. When a non-trivial component has at least one oriented cycle, it is an *oriented component*. Otherwise it is an *unoriented component*. Figure 10 (B) shows a breakpoint graph with an oriented and an unoriented component.

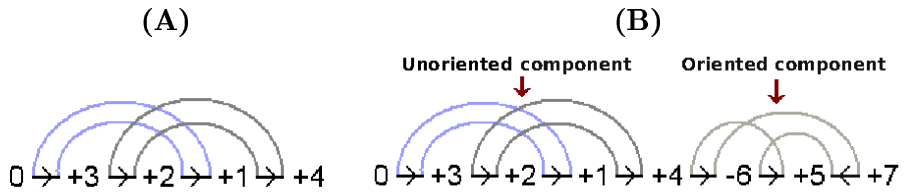

Figure 10: (A) A breakpoint graph in which we cannot perform a split reversal. (B) A breakpoint graph with an oriented and an unoriented component.

A reversal ρ is called *cut reversal* when its extremities are in the same cycle of an unoriented component. A cut reversal is always neutral and transforms an unoriented component into an oriented component (Figure 11 (A)), thus we say that a cut reversal eliminates an unoriented component (observe that a cut reversal does not change the number of cycles in the breakpoint graph). When the breakpoint graph has more than one unoriented component, it is not always necessary to use one cut reversal for each unoriented component. An unoriented component Y separates two other unoriented components X and Z when there is a black edge of Y between any black edge of X and any black edge of Z. In this case, a reversal that has one extremity in X and one extremity in Z will regroup the components X, Y and Z into one oriented component (Figure 11 (B)); this kind of reversal is called *merge reversal*. A merge reversal is always a joint reversal that regroups i unoriented components into one oriented component, for $i \geq 2$, thus we say that a merge reversal eliminates $i \geq 2$ unoriented components (observe that a merge reversal decreases the number of cycles in the breakpoint graph by one).

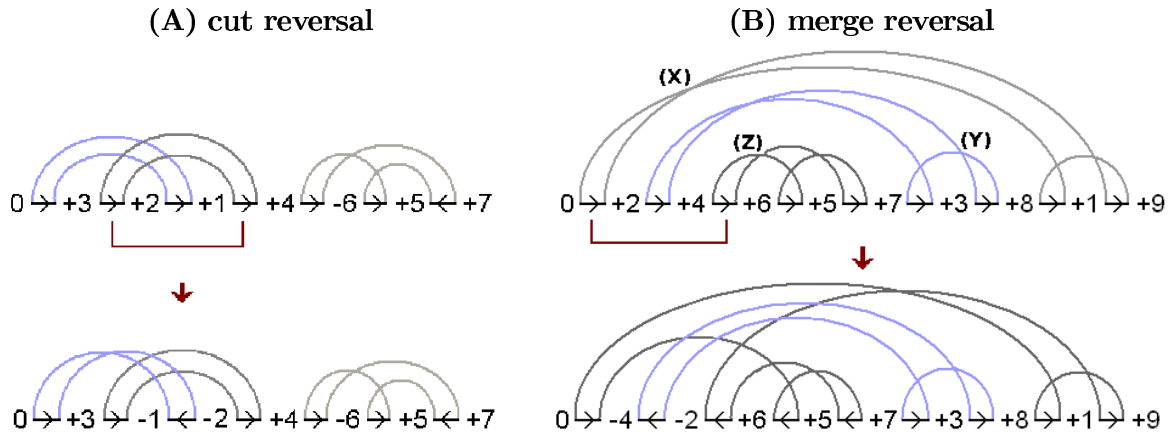

Figure 11: (A) A cut reversal transforms an unoriented into an oriented component and does not change the number of cycles in the breakpoint graph (it is a neutral reversal). (B) The unoriented component Y separates the unoriented components X and Z. A merge reversal regroups the unoriented components X, Y and Z into one oriented component and decreases the number of cycles in the breakpoint graph of one (it is a joint reversal).

An unoriented component that does not separate two other unoriented components is called a *hurdle*. We represent by $hrd(\pi)$ the number of hurdles in a breakpoint graph π. Since a hurdle does not separate unoriented components, each hurdle X can be eliminated either by a cut reversal whose extremities are in points of the same cycle of X (Figure 11 (A)), or together with another hurdle Z by a merge reversal whose extremities are in a point of X and a point of Z (Figure 11 (B)). A cut reversal eliminates one hurdle and does not change the number of cycles in the graph, while a merge reversal eliminates two hurdles at once, and decreases the number of cycles in the graph by one. Thus, each hurdle requires one additional reversal and we can improve the distance formula to $d(\pi) \geq pts(\pi) - cyc(\pi) + hrd(\pi)$. We say that a hurdle Z protects an unoriented component Y that is not a hurdle, if Y becomes a hurdle after the elimination of Z by a cut reversal. In this case, the hurdle Z is called *super-hurdle*. Eliminating a super-hurdle by a cut-reversal does not decrease the number of hurdles in the graph (Figure 12), consequently a super-hurdle may always be eliminated together with another super-hurdle by a merge reversal, that will regroup the two super-hurdles and their corresponding protected unoriented components into one oriented component (Figure 11 (B)).

It remains only one particular case to complete the reversal distance formula. When all the i hurdles of a breakpoint graph are super-hurdles and i is an odd number, the permutation requires an additional effort to be sorted. A breakpoint graph with this characteristic is called a

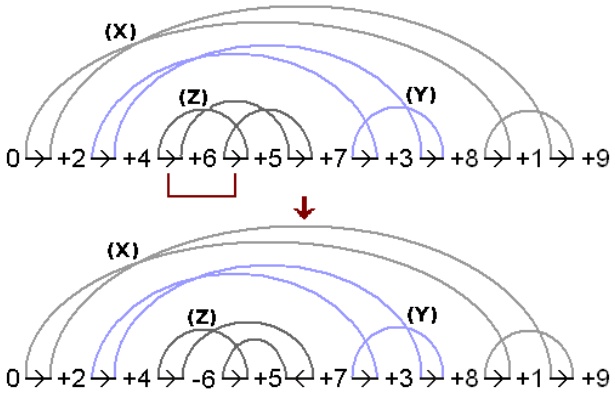

Figure 12: The unoriented component Y separates the super-hurdles X and Z. After eliminating the super-hurdle Z by a cut reversal, the component Y becomes a hurdle, thus the number of hurdles in this graph is not reduced after applying this cut reversal.

fortress. One additional reversal is sufficient to eliminate the fortress (this reversal may be chosen among several possibilities, for example a cut reversal to eliminate a hurdle, or a merge reversal regrouping two hurdles). We denote by $frt(\pi)$ a value that indicates whether the breakpoint graph π is a fortress or not. Thus, if π is a fortress, then $frt(\pi) = 1$, otherwise $frt(\pi) = 0$.

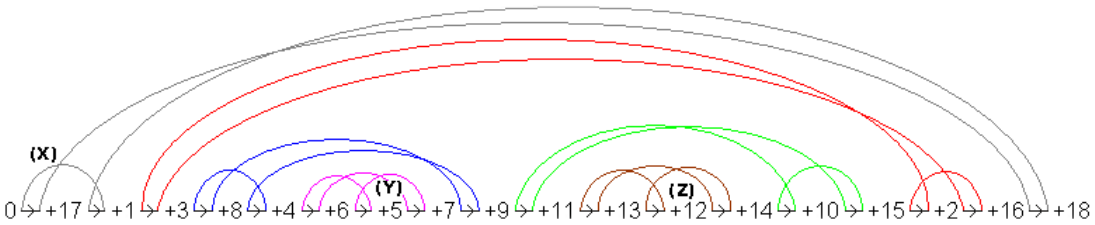

Figure 13: A fortress with 3 super-hurdles (X, Y and Z).

Table 1 summarizes the effects of a reversal that is part of an optimal sorting sequence in a breakpoint graph. The final formula for the reversal distance is:

$$d(\pi) = pts(\pi) - cyc(\pi) + hrd(\pi) + frt(\pi)$$

Remember that if $\pi = (\pi_1, \pi_2, \ldots, \pi_{n-1}, \pi_n)$ is a linear permutation, then $pts(\pi) = n + 1$. Otherwise π is a circular permutation and $pts(\pi) = n$.

Reversal	Type	$\Delta_{cyc(\pi)}$	$\Delta_{hrd(\pi)}$	$\Delta_{frt(\pi)}$
split	split	+1	0	n/A
hurdle cut	neutral	0	−1	n/A
hurdle merge	joint	−1	−2	n/A
fortress elim by unor. comp. cut	neutral	0	0	−1
fortress elim by unor. comp. & hurdle merge	joint	−1	−1	−1

Table 1: The effects of a reversal that is part of an optimal sorting sequence in a breakpoint graph. The columns $\Delta_{cyc(\pi)}$, $\Delta_{hrd(\pi)}$ and $\Delta_{frt(\pi)}$ give, respectively, the variation in the number of cycles, hurdles and fortress of a permutation after applying each reversal.

3.3 Safe and unsafe reversals

If a breakpoint graph does not have unoriented components, it can be sorted with split reversals only. However, if we take no caution to select a split reversal, it may cause the production of new hurdles, which is an undesirable side effect (Figure 14 (A)). A split reversal that produces hurdles is called *unsafe reversal*, while a split reversal that does not produce hurdles is called *safe reversal* (Figure 14 (B)). Fortunately, it has been proven that, for any oriented component, there is always one safe reversal [53].

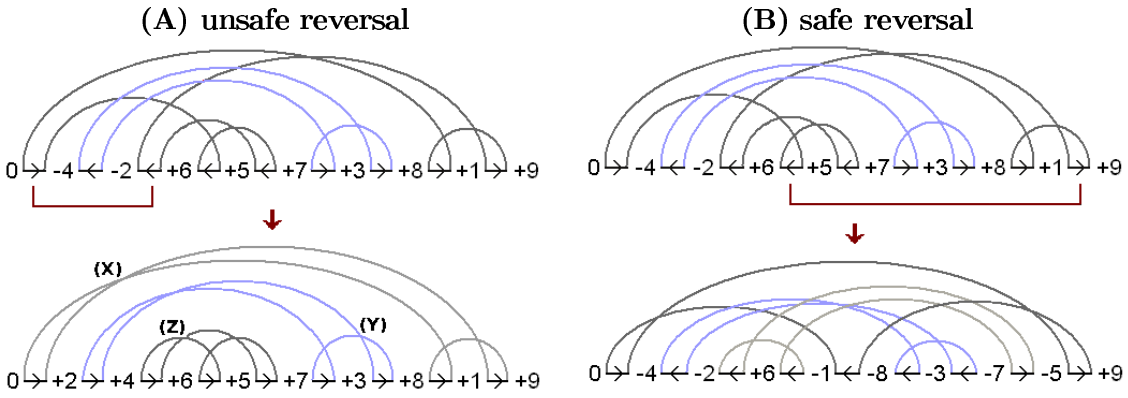

Figure 14: (A) An unsafe reversal breaks a cycle in two, but creates three unoriented component (X, Y and Z). (B) Alternatively, a safe reversal breaks a cycle in two without creating unoriented components.

Hurdles are very rare, and fortresses are even more rare in permutations that represent real genomes [9]. In practice, split reversals are sufficient to sort the majority of the permutations,

and the main challenge is to find safe reversals. A simple way to do that is testing each split reversal to verify whether it is safe or not, until finding a safe reversal. However, there are faster ways to select a safe reversal, and one approach is based on another structure related to the breakpoint graph, that is called *overlap graph* (see more details in [37]).

3.4 Sorting a signed permutation

With the approach described in this chapter, we can obtain a procedure to sort a permutation π by reversals (Algorithm 1).

Algorithm 1 Sorting a signed permutation

Input: A signed permutation π
Output: An optimal sequence of reversals sorting π

construct the breakpoint graph of π
$s \leftarrow \epsilon$ [s is an empty sequence in the beginning]
if $frt(\pi) = 1$ **then**
  choose a reversal ρ to eliminate the fortress
  $\pi \leftarrow \pi \circ \rho$
  $s \leftarrow s \cdot \rho$ [concatenates the reversal ρ to s]
end if
while there is a pair of super-hurdles X and Y in π **do**
  choose a merge reversal ρ to eliminate X and Y
  $\pi \leftarrow \pi \circ \rho$
  $s \leftarrow s \cdot \rho$ [concatenates the reversal ρ to s]
end while
while there is a hurdle Z in π **do**
  choose a cut reversal ρ to eliminate Z
  $\pi \leftarrow \pi \circ \rho$
  $s \leftarrow s \cdot \rho$ [concatenates the reversal ρ to s]
end while
while π is not sorted **do**
  choose a safe split reversal ρ to π
  $\pi \leftarrow \pi \circ \rho$
  $s \leftarrow s \cdot \rho$ [concatenates the reversal ρ to s]
end while
return s [s is an optimal sorting sequence for π]

The theoretical complexity of Algorithm 1 is $O(n^5)$, where n is the size of the input permutation [53]. Further studies improved this theoretical complexity and currently the fastest algorithm to find an optimal sorting sequence is subquadratic [11, 31, 63], while the reversal distance can be computed in $O(n)$ time [4].

References

[1] Adi, S., Braga, M. D. V., Fernandes, C., Ferreira, C., Martinez, F., Sagot, M.-F., Stefanes, M., Tjandraatmadja, C. and Wakabayashi, Y., "Repetition-free longest common subsequence", *Discrete Applied Mathematics*, submitted, 2009 (a preliminary version appeared in Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS), *Eletronic Notes in Discrete Mathematics*, Vol. 30, Pages 243–248, 2008).

[2] Andersson S. G. E. and Kurland C. G., "Reductive evolution of resident genomes", *Trends in Microbiology*, Vol. 6, Number 7, 263–268, 1998.

[3] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M., *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer, 1999.

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

[5] Vineet Bafna, V., Pevzner, P. A., "Sorting by Transpositions", *SIAM Journal on Discrete Mathematics*, vol. 11, issue 2, 224–240, 1998.

[6] Berard S., Bergeron A. and Chauve C., "Conserved structures in evolution scenarios", RCG 2004, *Lecture Notes in Bioinformatics*, vol. 3388, 1–15, 2005.

[7] Berard S., Bergeron A., Chauve C. and Paul C., "Perfect sorting by reversals is not always difficult", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol. 4, No. 1, 4–16, 2007.

[8] Bergeron A., "A very elementary presentation of the Hannenhalli-Pevzner theory", *Discrete Applied Mathematics*, vol. 146, 134–145, 2005.

[9] Bergeron A., Chauve C., Hartmann T. and St-Onge K., "On the properties of sequences of reversals that sort a signed permutation", *JOBIM 2002*, 99–108, 2002.

[10] Bergeron A., Heber S. and Stoye J., "Common intervals and sorting by reversals: a marriage of necessity", *Bioinformatics*, 18 (Suppl. 2): S54–63, 2002.

[11] Bergeron A., Mixtacki J. and Stoye J., "The inversion distance problem", *Mathematics of evolution and phylogeny* (O. Gascuel Ed.) Oxford University Press, 2005.

[12] Bergroth, L., Hakonen, H. and Raita T., "A Survey of Longest Common Subsequence Algorithms", SPIRE 00: pages 39-48, 2000.

[13] Blanc G., Ogata H., Robert C., Audic S., Suhre K., Vestris G., Claverie J.-M. and Raoult D., "Reductive genome evolution from the mother of Rickettsia", *PLoS Genetics*, volume 3, p. 103-114, 2007.

[14] Blin, G., Fertin, G. and Chauve, C., "The breakpoint distance for signed sequences", *Texts in Algorithms*, vol. 3, pages 3-16, CompBioNets 2004.

[15] Bonizzoni P., Della Vedova G., Dondi R., Fertin G. and Vialette S., "Exemplar Longest Common Subsequence", *ACM/IEEE Trans. Computational Biology and Bioinformatics*, Vol. 4, No. 4, pages 535–543, 2007.

[16] Braga M. D. V., "`baobabLuna`: the solution space of sorting by reversals", submitted to

Bioinformatics, 2009.

[17] Braga M. D. V., Gautier C. and Sagot M.-F., "An asymmetric approach to preserve common intervals while sorting by reversals", submitted to *Algorithms for Molecular Biology*, 2009.

[18] Braga M. D. V., Sagot M.-F., Scornavacca C. and Tannier E., "Exploring the solution space of sorting by reversals with experiments and an application to evolution", *Transactions on Computational Biology and Bioinformatics*, volume 5, number 3, 348–356, 2008 (A preliminary version appeared in ISBRA 2007, *Lecture Notes in Bioinformatics* vol. 4463, 293–304).

[19] Brightwell G. and Winkler P., "Counting linear extensions is #P-complete", *STOC '91: Proceedings of the twenty-third annual ACM Symposium on Theory of Computing*, ACM Press, 1991.

[20] Bryant, D., "The complexity of calculating exemplar distances", 2000.

[21] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M., *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley and Sons Ltd, Chichester, UK, 1996.

[22] Caprara, A., "Sorting by reversals is difficult", *RECOMB*, 75–83, 1997.

[23] Cartier, P. and Foata D., "Problèmes combinatoires de commutations et réarrangements", *Lecture Notes in Math*, vol. 85, Springer, Berlin, 1969.

[24] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to algorithms*, MIT Press, 2. edition, 2001.

[25] Diekert V. and Rozenberg G. (eds), *The book of traces*, World Scientific, 1995.

[26] Diekmann Y., Sagot M.F. and Tannier E., "Evolution under reversals: parsimony and conservation of common intervals", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, No. 2, 301–309 (A preliminary version appeared in COCOON 2005, *Lecture Notes in Computer Science*, vol. 3595, 42–51).

[27] Eisen J. A., Heidelberg J. F., White O. and Salzberg S. L., "Evidence for symmetric chromosomal inversions around the replication origin in bacteria", *Genome Biology*, vol. 1, No. 6, 2000.

[28] Fulkerson D. R., "Note on Dilworth's decomposition theorem for partially ordered sets", Proc. Amer. Math. Soc. 7, 701–702, 1956.

[29] Goffeau A. *et al.*, "Life with 6000 genes", Science 274. doi:10.1126/science.274.5287.546, 1996.

[30] Gomez-Valero, L., Rocha, E. P. C., Latorre, A. and Silva, F. J., "Reconstructing the ancestor of *Mycobacyerium leprae*: the dynamics of gene loss and genome reduction", *Genome Research*, vol. 17, 1178–1185, 2007.

[31] Han Y., "Improving the Efficiency of Sorting by Reversals", Proceedings of The 2006 International Conference on Bioinformatics and Computational Biology, CSREA Press, Las Vegas, Nevada, USA, 2006.

[32] Hannenhalli S. and Pevzner P., "Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals)", *Journal of the ACM*, 46:1–27, 1999.

[33] Hannenhalli, S. and Pevzner, P., "Transforming men into mice (polynomial algorithm for genomic distance problem)", In Proceedings of the IEEE 36th Annual Symposium on Foun-

dations of Computer Science, pages 581-592, 1995.

[34] Heber, S. and Stoye, J., "Finding all common intervals of k permutations", in Combinatorial Pattern Matching, 12th Annual Symposium, *Lecture Notes in Computer Science*, vol. 2089, 207–218, 2001.

[35] IJdo J. W., Baldini A., Ward D. C., Reeders S. T., Wells R. A., "Origin of human chromosome 2: an ancestral telomere-telomere fusion", Proc Natl Acad Sci U S A, 88(20):9051-5, 1991.

[36] Iwase, M., Satta, Y., Hirai, Y., Hirai, H., Imai, H., and Takahata, N., "The amelogenin loci span an ancient pseudoautosomal boundary in diverse mammalian species", *PNAS*, vol. 100, no. 9, 5258–5263, 2003.

[37] Kaplan, H., Shamir, R., and Tarjan, R. E., "Faster and simpler algorithm for sorting signed permutations by reversals", *SIAM J. Comput*, vol. 29, 880-892, 1999.

[38] Kececioglu, J. and Sankoff, D., "Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement", *Algorithmica*, 13:1/2, 180–210, 1995.

[39] Knuth, D., "The Art of Computer Programming, Volume 3: Sorting and Searching", Second Edition, Addison-Wesley, 1998.

[40] Lahn B. T. and Page D. C., "Four evolutionary strata on the human X chromosome", *Science*, vol. 286, 964–967, 1999.

[41] Lemaitre C., Braga M. D. V., Gautier C., Sagot M.-F., Tannier E. and Marais G. A. B., "Footprints of inversions at present and past pseudoautosomal boundaries in human sex chromosomes", submitted to *Genome Biology and Evolution*, 2009[19].

[42] Mackiewicz, P., Mackiewicz, D., Kowalczuk, M. and Cebrat S.,"Flip-flop around the origin and terminus of replication in prokaryotic genomes", *Genome Biology*, vol. 2, No. 12, 2001.

[43] Mazowita, M., Haque, L. and Sankoff, D., "Stability of rearrangement measures in the comparison of genome sequences", *Journal of Computational Biology*, vol. 13, 554–566, 2006.

[44] McLysaght, A., Seoighe, C. and Wolfe K. H., "High frequency of inversions during eukaryote gene order evolution", in *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and the Evolution of Gene Families*, D. Sankoff and J. H. Nadeau (Eds.), 47–55, 2000.

[45] Moret, B.M.E., Wyman, S., Bader, D.A., Warnow, T., and Yan, M., "A new implementation and detailed study of breakpoint analysis", *Proc. 6th Pacific Symp. on Biocomputing* (PSB 2001), Hawaii, World Scientific Pub., 583–594, 2001.

[46] Nakabachi A., Yamashita A., Toh H., Ishikawa H., Dunbar H., Moran N., Hattori M., "The 160-kilobase genome of the bacterial endosymbiont Carsonella", *Science*, 314 (5797): 267, 2006.

[47] Ogata H., Renesto P., Audic S., Robert C., Blanc G., Fournier P.-E., Parinello H., Claverie J.-M. and Raoult D., "Genome sequence of Rickettsia felis identifies the first putative conjugative plasmid in an obligate intracellular parasite", *PLoS Biology*, volume 3, p. 1-12, 2005.

[48] Ogata H., La Scola B., Audic S., Renesto P., Blanc G., Robert C., Fournier P.-E., Claverie J.-M. and Raoult D., "Genome sequence of Rickettsia bellii illuminates the role of amoebae in

[19]The work of Lemaitre *et al.* is attached at the end of this manuscript

gene exchange between intracellular pathogens", *PLoS Genetics*, volume 2, p. 733-744, 2006.

[49] Ohno, S., *Evolution by gene duplication*, Springer-Verlag, New York, 1970.

[50] Ohno, S., *Sex chromosomes and sex-linked genes*, Springer, Berlin, 1967.

[51] Papadimitriou, C. H. and Yannakakis, M., "Optimization, approximation and complexity classes", *Journal of Computer and System Sciences*, 43:425–440, 1991.

[52] Parfrey, L. W., Lahr, D. J. G., Katz, L. A., "The Dynamic Nature of Eukaryotic Genomes", *Molecular Biology and Evolution*, 25 (4): 787, 2008.

[53] Pevzner P., *Computational Molecular Biology - An Algorithmic Approach*, The MIT Press, 2000.

[54] Pradella S., Hans A., Spröer C., Reichenbach H., Gerth K., Beyer S., "Characterisation, genome size and genetic manipulation of the myxobacterium Sorangium cellulosum So ce56". *Arch Microbiol*, 178 (6): 484-92, 2002.

[55] Ross M. T. *et al.*, "The DNA sequence of the human X chromosome", *Nature*, vol. 434, p. 325–337, 2005.

[56] Sankoff, D., "Gene and genome duplication", *Current Opinion in Genetics and Development*, vol. 11, p. 681–684, 2001.

[57] Sankoff, D., "Genome Rearrangement with gene families", *Bioinformatics*, vol. 15, no. 11, pages 909-917, 1999.

[58] Schneiker S. *et al.*, "Complete genome sequence of the myxobacterium Sorangium cellulosum", *Nature Biotechnology*, 25, 1281–1289, 2007.

[59] Siepel A., "An algorithm to enumerate sorting reversals for signed permutations", *J Comput Biol*, 10:575–597, 2003.

[60] Skaletsky H. *et al.*, "The male-specific region of the human Y chromosome is a mosaic of discrete sequence classes", *Nature*, vol. 423, 825–837, 2003.

[61] Steiner G., "An algorithm to generate the ideals of a partial order", *Operations Research Letters*, 5(6):317–320, 1986.

[62] Steiner G., "Polynomial algorithms to count linear extensions in certain posets", *Congressus Numerantium*, 75, 71–90, 1990

[63] Tannier E., Bergeron A. and Sagot M.-F., "Advances on Sorting by Reversals", *Discrete Applied Mathematics*, vol. 155, no. 6-7, 881–888, 2007 (a preliminary version appeared in CPM 2004, *Lecture Notes in Computer Science*, vol. 3595, 42–51).

[64] Tesler, G., "GRIMM: genome rearrangements web server", *Bioinformatics*, vol. 18, no. 3, 492–493, 2002.

[65] Weaver R. F., *Molecular Biology*, Mc Graw Hill, second edition, 2002.

[66] Zheng, C., Lenert, A. and Sankoff, D., "Reversal distance for partially ordered genomes", Bioinformatics 21, i502-i508, 2005.