Exploring the set of all minimal sequences of reversals - An application to test the replication-directed reversal hypothesis

Yasmine Ajana[1], Jean-François Lefebvre[2], Elisabeth R.M. Tillier[3], and Nadia El-Mabrouk[4*]

[1] Département d'informatique et de recherche opérationnelle (DIRO), Université de Montréal, Québec, Canada, ajanay@iro.umontreal.ca.
[2] DIRO, lefebvrj@iro.umontreal.ca.
[3] Clinical Genomics Centre, University Health Network, 101 College street, MBRC 5-501, Toronto, Ontario M5G1L7, Canada, e.tillier@utoronto.ca.
[4] DIRO, Université de Montréal, CP 6128 Succursale Centre-ville, Montréal, Québec H3C3J7, Canada, mabrouk@iro.umontreal.ca.

Abstract. Given two genomes, the problem of sorting by reversals is to explain the evolution of these genomes from a common ancestor by a minimal sequence of reversals. The Hannenhalli and Pevzner (HP) algorithm [HP95a] gives the reversal distance and outputs one possible sequence of reversals. However, there is usually a very large set of such minimal solutions. To really understand the mechanism of reversals, it is important to have access to that set of minimal solutions. In this paper, we develop a new method that allows the user to choose one or several solutions, based on different criteria. In particular, it can be used to sort genomes by weighted reversals. This requires a characterization of all "safe" reversals, as defined in the HP theory [HP95a]. We describe a procedure that outputs the set of all safe reversals at each step of the sorting procedure in time $O(n^3)$, and we show how to characterize a large set of such reversals in a more efficient way. We also describe a linear algorithm allowing to generate a random genome of a given reversal distance. We use our methods to verify the hypothesis that, in bacteria, most reversals act on segments surrounding one of the two endpoints of the replication axis [TC00].

1 Introduction

The availability of complete genome sequence data from a multitude of bacterial and archaeal species now enables evolutionary relationships to be inferred from entire genetic material. The algorithmic study of comparative genomics has focused on inferring the minimal number of genomic mutations (genome rearrangements) necessary to explain the observed differences in gene orders. The rearrangement operations that have been mostly considered by authors are inversion (reversal) of a chromosomal segment [KS95,HP95a,KST97,BMY01], transposition of a segment from one site to another in one chromosome [BP98,WDM98], and reciprocal translocation in the case of multi-chromosomal genomes [Han95]. In particular, the exact polynomial algorithm of Hannenhalli and Pevzner (hereafter "HP") for sorting signed permutations by reversals [HP95a] was a breakthrough for the formal analysis of genome rearrangement. Moreover, they were able to extend their approach to the analysis of both reversals and reciprocal translocations at the same time [HP95b], and we further generalized the method to account for insertions and losses of fragments of chromosomes [EM00].

The HP algorithm allows to compute the reversal distance between two genomes and output one minimal sequence of reversals. However, there is usually a very large set of such minimal solutions. To really understand the mechanism of reversals, it is important to have access, not only to one, but to a large set of such solutions. This allows to choose one or several solutions on specific criteria, as for example the length of the segments involved in the reversals, the cut-sites, or the position of the reversals with respect to the replication axis. So

* to whom correspondence should be addressed

far, the only existing algorithm that allows to weight the different rearrangement operations and to choose one solution of minimal cost has been the *Derange* heuristic [BKS96]. It approximates the optimal distance by minimizing a weighted sum of reversals, transpositions and transversions (inversion and transposition of a segment from one site to another in the genome). As only some specific reversals are allowed (those that reduce the number of breakpoints), sorting by reversals only is not always possible with *Derange*.

In this paper, we develop a new method for sorting signed permutations by reversals, that allows the user to choose one or several solutions, based on different criteria. In particular, it can be used to sort genomes by weighted reversals. The only constraint of the method is minimality: the solutions are all minimal sequences of reversals. This minimality constraint is justified by the fact that the complete set of solutions can be huge and impossible to explore in its entirety. To give an example, in the case of the genomes *C.pneumoniae/C.trachomatis* of reversal distance $d = 422$, there are an average of 661 safe reversals (reversals leading to the reversal distance), at each step of the sorting procedure (422 steps). In *Derange*, the solution space explored was also restricted to a very small number of reversals.

Our new algorithm is based on the HP method. After clearing the hurdles, each step of the HP algorithm consists in choosing one safe reversal [HP95a]. The problem of finding a safe reversal has been considered by many authors [Ber01,KST97,HP95a]. In particular, Bergeron [Ber01] formulated an elementary criterion for a reversal to be safe. Recently, Siepel [Sie02] enumerated all sorting reversals, that is, in addition to safe reversals, all reversals that clear the hurdles. As the number of hurdles is usually close to 0, we do not focus on hurdles in this paper. The key idea of our new method is to begin by clearing the hurdles, and then, at each step, generate all safe reversals, sort them with the user's criteria, and choose the first one in that order.

The paper is organized as follows. In Section 2, we begin by introducing the problem and the HP graph and results. In Section 3, we characterize a large set of safe reversals, and describe an efficient algorithm allowing to generate them. We then show, in Section 4, how to generate all safe reversals at each step of the sorting procedure in time $O(n^3)$. We then develop a new method for generating all minimal sequences of reversals. This is used to describe a branch-and-bound method for sorting permutations by weighted reversals. Section 5 gives an algorithm that generates a random genome of a given reversal distance. These algorithms are finally used, in Section 6, to test the replication-directed reversal hypothesis in bacteria. By comparing orthologous gene positions in several pairs of bacterial genomes, Tillier and Collins [TC00] provided evidence that genomes have been rearranged by reversals, and that a substantial proportion of these reversals results from recombination sites that are determined by the positions of the replication forks. Our results are in agreement with this hypothesis for reasonably shuffled genomes.

2 Preliminaries

This work will be situated in the context of circular genomes, as is most frequently the case for single chromosomal genomes such as organelles and bacteria. Formally, given a set $\mathcal{B}$ of genes, a genome is a circular string of signed ($+$ or $-$) elements of $\mathcal{B}$, no gene appearing more than once. Though we will write a genome as a linear string, it is understood that the first gene is adjacent to the right of the last gene. For a string $X = x_1 x_2 \cdots x_r$, denote by $-X$ the **reverse** string $-x_r - x_{r-1} \cdots - x_1$. A **reversal** (or inversion) transforms some proper substring of a genome into its reverse. By the size of a reversal, we mean the length of the reversed segment.

Given two genomes G, H on the same set of genes $\mathcal{B}$, the HP algorithm computes the minimal number of reversals (reversal distance) required to transform G to H, and outputs one possible sequence of reversals satisfying the minimality criterion. Without loss of generality, we assume that H is the identity permutation, i.e. $H = +1 +2 +3 \cdots +n$. We denote by $R(G)$ the reversal distance between G and H.

The HP algorithm is based on a bicoloured **breakpoint graph** $B(G)$, defined as follows. If gene x_i in genome $G = x_1 \cdots x_n$ has positive sign, replace it by the pair $x_i^t x_i^h$, and if it is negative, by $x_i^h x_i^t$. Then the vertices of $B(G)$ are just the x_i^t and the x_i^h for all x_i of $\mathcal{B}$. Any two vertices that are adjacent in G, other than x_i^t and x_i^h from the same x_i, are connected by a black edge, and any two adjacent in H, by a gray edge (Figure 1.a). This graph decomposes into a set of $c(G)$ disjoint color-alternating cycles. By the **size of a cycle** we mean the number of black edges it contains.

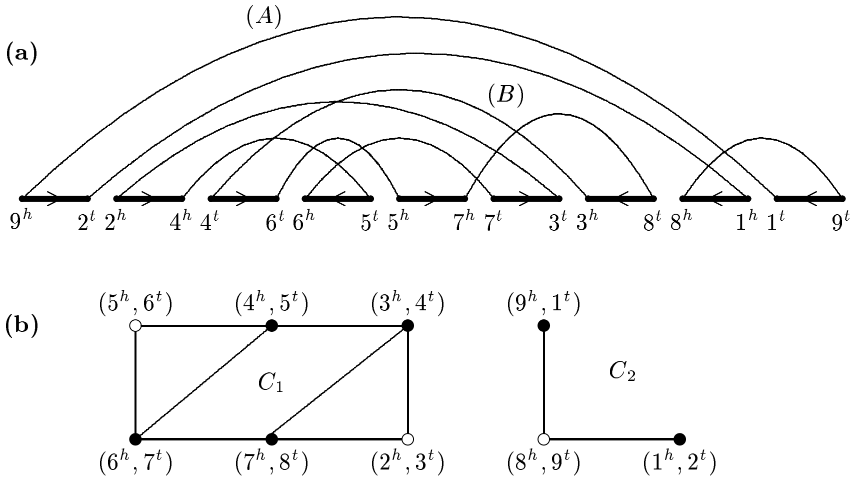

Fig. 1. (a) The breakpoint graph $B(G)$ of the circular genome $G = +2\ -4\ +6\ +5\ -7\ +3\ +8\ -1\ +9$. A possible orientation of the black edges is specified. The graph has three cycles, all of them being oriented cycles, and two oriented components (A) and (B). **(b)** The overlap graph $OV(G)$. The oriented vertices are the black ones. C_1, C_2 are the two components of the graph.

We say that a **reversal** R **is determined by the two black edges** $e = (a, b)$ and $f = (c, d)$ if R reverses the segment of G bounded by the vertices b and c. For example, in Figure 1.a, the reversal determined by the black edges $(4^t, 6^t)$ and $(3^h, 8^t)$ reverses the segment $< +6\ +5\ -7\ +3 >$. Similarly, the **reversal determined by a gray edge** g is the one determined by the two black edges adjacent to g. For example, the reversal determined by the gray edge $(4^h, 5^t)$ reverses the segment $< -4\ +6 >$, i.e. creates the segment $< -6\ +4 >$.

Let e and f be two black edges of the same cycle C of size greater than 1. Think of both e and f as being oriented clockwise in the graph. Each one of them induces an orientation for C. If the orientation induced by e and f coincide, we say that the two edges are **convergent**; otherwise they are **divergent**. For example, in Figure 1.a, the black edges $(4^t, 6^t), (3^h, 8^t)$ diverge, while the black edges $(4^t, 6^t), (5^h, 7^t)$ converge. We also say that a gray edge g is **oriented** if it connects two divergent black edges, and **unoriented** otherwise. For example, the gray edge $(6^h, 7^t)$ is oriented, while $(6^t, 5^h)$ is unoriented.

Lemma 1. (**[HP95a]**) *Let R be a reversal determined by the two black edges e and f.*

1. *If e and f belong to different cycles, then the number of cycles of the breakpoint graph decreases by 1 after performing the reversal R. We say that R is a* **bad reversal**.
2. *If e and f belong to the same cycle and converge, then the number of cycles of the graph remains unchanged after R.*
3. *If e and f belong to the same cycle and diverge, then the number of cycles increases by 1 after R. We say that R is a* **good reversal**.

The number of cycles of the breakpoint graph is maximized when $G = H$. Therefore, the problem of minimizing the number of reversals reduces to the one of increasing the number of cycles as fast as possible, that is of performing as many good reversals as possible. This depends on the decomposition of the graph into its components. To explain this decomposition, we need to introduce some basic definitions. A cycle C is **oriented** if it contains at least one oriented gray edge, and **unoriented** otherwise. A **component** of $B(G)$ is a maximal subset of crossing cycles. An **oriented component** has at least one oriented cycle; otherwise it is an **unoriented component** (see Figure 1.a). Finally, a **safe reversal** is a good reversal that does not create any new unoriented component.

HP proved that an oriented component can be **solved**, i.e transformed to a set of cycles of size 1, by a sequence of safe reversals. As for unoriented components, some of them can still be solved by safe reversals, whereas others, called **hurdles**, require bad reversals to be solved (see [HP95a] for a more detailed description of hurdles).

Finally, the minimal number of reversals necessary to transform G into H is given by:

$$R(G) = n - c(G) + h(G) + fr(G) \qquad \textbf{(HP-formulae)}$$

where n is the size of G (number of genes), $c(G)$ the number of cycles of $B(G)$, $h(G)$ its number of hurdles, and $fr(G) = 1$ or 0 depending on the set of hurdles.

3 Finding safe reversals

The bottleneck of the HP theory is to solve the oriented components by performing exclusively safe reversals. Different methods have been developed [HP95a,BH96,KST97,Ber01], one the most efficient and simple one being that presented in [Ber01]. We use their notations and concepts to generate a set of safe reversals.

The **overlap graph** $OV(G)$ is defined as follows: the vertices of $OV(G)$ are all the gray edges of $B(G)$, and two vertices v_1, v_2 are connected by an edge iff v_1, v_2 are two intersecting edges in $B(G)$ (Figure 1.b). For any vertex v, the **reversal** v will refer to the one determined by the gray edge v, and the reversal orientation will be the orientation of the vertex v. Bergeron proved that a vertex of $OV(G)$ is oriented iff it has an odd number of adjacent edges. She also proved the following facts (see Figure 2):

- **Fact 1 ([Ber01]):** If one performs the reversal v, the effect on $OV(G)$ is to complement the subgraph formed by the vertex v and its adjacent vertices.

- **Fact 2 ([Ber01]):** If one performs the reversal v, each vertex adjacent to v changes its orientation.

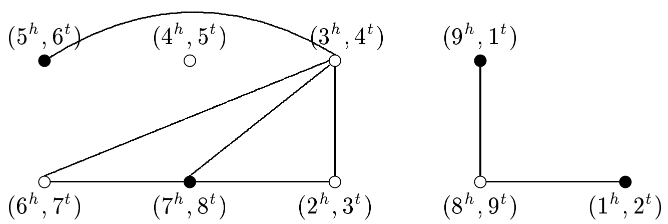

Fig. 2. The effect of the reversal $(4^h, 5^t)$ on the overlap graph $OV(G)$ of Figure 1.b.

Proposition 1. ([Ber01]). *Let the* **score** *of a vertex v of $OV(G)$ be the number of oriented vertices in the overlap graph resulting from the reversal v. Then, an oriented reversal of maximal score is safe.*

From the results given above, an efficient bit vector algorithm for solving each oriented component C of $OV(G)$ has been developed [Ber01]. To do so, each vertex v of C is represented by a bit vector $\mathbf{v}$ whose w^{th} coordinate is 1 iff v is adjacent to vertex w. Then, C is represented by the bit matrix containing as many lines as the number of vertices of the component. Moreover, a vector p is used to store the orientation of each vertex of C (1 if oriented and 0 otherwise), and a vector s is used to store the score for each v. Bergeron showed that the matrix is easily updated after performing one safe reversal.

But proposition 1 usually allows to output only one safe reversal. Here, we give more general criteria allowing to characterize a large subset of safe reversals.

Proposition 2. *Let v be an oriented reversal. If v is unsafe, then there exists an oriented vertex w adjacent to v such that the three following conditions are verified:*

1. *There exists at least one vertex adjacent to w but not to v, or adjacent to v but not to w.*
2. *For any unoriented vertex u adjacent to v, u is adjacent to w.*
3. *For any oriented vertex u adjacent to w, u is adjacent to v.*

Proof. Let OV and OV' be the overlap graph before and after performing the oriented reversal v. Suppose v is an unsafe oriented vertex. Then v creates an unoriented component C in OV', not containing v (because v becomes solved in OV'). As C was not an unoriented component in OV, it should exist an unoriented vertex w belonging to C that was oriented and adjacent to v in OV.

- From Fact 1, the vertices that are adjacent to w in OV' are those that were adjacent to w but not to v in OV, or those that were adjacent to v but not to w in OV. Now, condition 1 is not verified implies that no vertex verifies the above properties. Therefore, in that case, the component C containing w in OV' does not contain any other vertex, and thus C can not be unoriented.
- If there exists u unoriented and adjacent to v but not to w in OV (u contradicts condition 2), then u becomes oriented and adjacent to w in OV', which contradicts the fact that C' containing w is an unoriented component. Thus, condition 2 is verified (see Figure 3).
- If there exists an oriented vertex u adjacent to w but not to v in OV (u contradicts condition 3), then u is still oriented and adjacent to w in OV', and C' is not an unoriented component. Thus, condition 3 is verified (see Figure 3) □

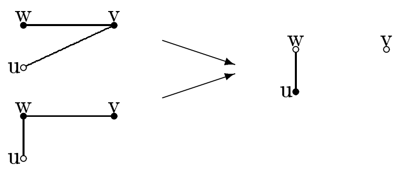

Figure 3: The up left and down left graph contradict condition 2 and 3, respectively. In both cases, the reversal v gives rise to the graph on the right. In this graph, the vertex w belongs to an oriented component.

Sufficient conditions for safe reversals are obtained by taking the negative of Proposition 2. These conditions allow to output a large subset of safe reversals in a very efficient way, by using exclusively bit wise operations.

This is done by *Algorithm safe-reversals (C)*, that outputs the subset of safe reversals for an oriented component C. In this algorithm, $\mathbf{p}$ is the orientation vector defined in [Ber01], & is the *and* bit wise operation, | the *or* bit wise operation and $\neg$ the *not* bit wise operation.

```
Algorithm safe-reversals (C):
For each oriented vertex v of C Do
    v̄ = v & ¬p; bool=true;
    For each oriented vertex w adjacent to v Do
        Z = (w & ¬v)|(v & ¬w);
        V ← v̄ & ¬w;
        w̄ ← w & p;
        W ← w̄ & ¬v;
        If V = 0 and W = 0 and Z ≠ 0 Then
            bool = false; exit;
        End of if
    End of for
    If bool=true Then v is safe Else v is unsafe;
    End of if
End of for
```

Algorithm safe-reversals has at most k^2 steps, where k is the number of vertices of the component C. As the procedure stops as soon as an appropriate w is found, the number of steps is usually lower than k^2. The number of words needed to represent each bit vector is $\lceil \frac{k}{\omega} \rceil$, where ω is the word size. Therefore, the time complexity of the algorithm applied to a component C of k vertices is $O(\lceil \frac{k}{\omega} \rceil k^2)$.

4 Exploring the set of all minimal sequences of reversals

We first consider the problem of generating all minimal sequences of reversals for a breakpoint graph that does not contain any hurdle. We then explore all the possible ways to solve the hurdles with the HP method. Finally, we use these results to develop a new heuristic for sorting permutations by weighted reversals.

4.1 Generating all minimal solutions

Given a graph with only oriented components, a **minimal solution** is a minimal sequence of reversals transforming the graph into a set of cycles of size 1.

Lemma 2. *A sequence of reversals is a minimal solution iff it contains exclusively safe reversals.*

Proof. It follows directly from the HP theory [HP95a] □

From the above lemma, the problem of generating all minimal solutions reduces to the problem of generating all the safe reversals at each step of the sorting procedure. Notice that a reversal on one component of the graph does not affect the other components. Thus, if the order of reversals is not important, we can solve each oriented component independently. Now, given an oriented component C, to find all safe reversals, one should check each pair of divergent black edges of the breakpoint graph. For each divergent pair (e, f), the resulting breakpoint graph should be decomposed into its components. The reversal determined by (e, f) is safe iff all the components are oriented. Checking all the components of a graph takes $O(n)$ time [KST97]. Now, as there are at most $O(n^2)$ pairs of divergent black edges in the breakpoint graph, the total time required to find all safe reversals at each step of the sorting procedure is in $O(n^3)$.

4.2 Hurdles

If the breakpoint graph contains hurdles, then they have to be transformed into oriented components. This is done by merging and cutting hurdles in a particular way, described by the HP procedure for clearing hurdles [HP95a]. However, this procedure allows some flexibility in choosing hurdles to be merged and cut, and reversals to be performed. More precisely, hurdles can be cleared in several optimal ways:

1. non-consecutive hurdles can be paired in all possible ways;
2. to merge a pair of hurdles, one black edge is arbitrarily chosen in each of the two hurdles;
3. to cut a hurdle, two black edges are arbitrarily chosen from the same cycle.

Therefore, if one is interested in finding all possible sequences of reversals transforming a genome G into a genome H, all ways of clearing hurdles should be considered. To limit the number of solutions, specific constraints on the reversals can be established. For example, if we are interested in finding the shortest reversals, then the chosen black edges for hurdle merging or hurdle cutting are the closest ones, and the only remaining flexibility is in the way to pair the hurdles.

However, notice that the probability for a component to be oriented is higher than its probability to be unoriented. Indeed, with the hypothesis that a vertex has the same probability to be oriented or unoriented, and that the vertices are two-two independent, the probability for a component of size k to be unoriented is $\frac{1}{2^k}$. Moreover, as a hurdle is a special case of an unoriented component, the probability for a component to be a hurdle is even lower. In practice, the unoriented components are almost non-existent (see Table 1).

Bacterial Genomes	Genome size (Nb of common genes)	Number of components	Number of hurdles
C.pneumoniae/C.trachomatis	732	70	0
V.cholera/E.coli	1252	76	1
M.leprae/M.tuberculosis	1295	212	0
C.jejuni/H.pylori	706	9	0
S.pyogenes/S.aureus	438	3	0
U.urealyticum/M.pneumoniae	278	6	0

Table 1. The breakpoint graph structure for various pairwise comparisons of bacterial genomes. Orthologous genes have been identified by Tillier and Collins [TC00]. Column 2 gives the number of orthologous genes considered, column 3 the total number of components of the graph, and column 4 the number of hurdles among these components.

4.3 Sorting signed permutations by weighted reversals

Understanding the mechanism of rearrangements, and in particular when and how reversals occur, is still an open question in biology. In order to answer this question and to test different hypothesis, the HP algorithm giving the reversal distance and one possible solution is not sufficient. So far, the only existing algorithm that explored different solutions has been *Derange* [BKS96]. It approximates the optimal distance by minimizing a weighted sum of reversals, transpositions and transversions. At each step, the algorithm considers only rearrangements that reduce the number of breakpoints. The method is a depth-first search with limited look-ahead and branch-and-bound. As the only operations that are allowed by *Derange* are those that reduce the number of breakpoints, in some cases sorting by reversals only is impossible and transpositions can not be avoided. This is always the case when the overlap graph contains hurdles, as an optimal sequence of reversals should contain at least one bad reversal.

Here, we develop a new depth-first branch-and-bound heuristic for weighted reversals. As for *Derange* algorithm, we do not consider all possible reversals for each node in the search tree, which make the algorithm a heuristic. Rather, we choose a set of reversals leading to the reversal distance.

The general method is as follows: we first clear the hurdles to end up with oriented components. We then solve the oriented components by considering the complete set of safe reversals. In other words, the algorithm has two major steps:

1. If the graph contains hurdles, perform a set of reversals clearing the hurdles.
2. Solve each oriented component independently, by choosing safe reversals.

The depth-first branch-and-bound method is used for the second step. For each component, all its safe reversals are ordered in decreasing order of their weight. Once a first solution is obtained (a sequence of reversals solving the oriented component), the cost of this solution (sum of its reversals weights) becomes the current bound. Then we backtrack, undoing the most recent reversal, and trying the next-best reversal, as long as one exists. If not, we undo the most recent move at the previous level. After each backtrack step, we again proceed until we end up with a partial solution of cost higher than the current best solution, or we completely solve the current component. In that case, if the new solution has a cost lower than the current bound, then the bound is replaced by this cost.

If the components of the overlap graph are small, the search tree is small and the branch-and-bound algorithm is efficient. However, even with our restricted set of reversals, the search becomes huge and unmanageable. For example in the case of large genomes with a high rate of rearrangements, the number of components is close to 1, and the search tree size can be on the order of n^r, where n is the size of the genome, and r is the reversal distance. To avoid such situation, the weighting function should allow a large prune of the search tree. This is not the case for a weighting proportional to the length of the reversed segment. Indeed, a long reversal that is avoided at the beginning of a path, is usually postponed and performed farther in the path. One way to improve the branch-and-bound algorithm is to select a small number of safe reversals at each step. This restriction has been considered in *Derange*.

5 Generating a random genome of a given reversal distance

To test various hypothesis on reversals, one should compare the results obtained for a real genome G with those obtained with a random genome. For example, in the case of the replication-directed reversal hypothesis considered in the next section, we are interested in weighting the reversals according to their position with respect to the replication axis. Now, suppose we find a solution of minimal weight with $R(G)$ reversals. To evaluate the distribution of the obtained reversals, we should compare it with the one obtained with a random genome of the same length and the same reversal distance. The problem is, therefore, to generate a random genome of reversal distance $R(G)$. This can not be done by simply starting with the identity permutation and performing $R(G)$ random reversals. Indeed, we usually end up with a genome that has a reversal distance significantly lower than $R(G)$.

Consider the HP-formulae (Section 2). As the number of hurdles is usually very close to 0, the dominant variable in this formulae is the number of cycles of the graph. In other words, suppose for now that we ignore the hurdles. Then, $R(G) = n - c(G)$, where n is the size of G and $c(G)$ the number of cycles in the breakpoint graph $B(G)$. Therefore, the problem of generating a random genome of size n and reversal distance $R(G)$ can be reduced to that of constructing a random breakpoint graph of size n (number of black edges), having $c(G)$ cycles. To do so, we start with the **partial graph** $B'(G)$, that is the breakpoint graph $B(G)$ without the gray edges, and we complete it with gray edges by *Algorithm construct-cycles* below. In this algorithm, a **free vertex** is a vertex not already adjacent to a gray edge. At the beginning of the algorithm, all vertices are free. At any step of the algorithm, the graph is constituted of a set of alternating paths and cycles. A gray edge is said to **close a path** if it links the two free vertices of the path. We denote by p the number of paths and c the number of cycles of the current graph. At the beginning of the algorithm, $p = n$ and $c = 0$. At the end, we should have $p = 0$ and $c = n - R(G)$.

Algorithm construct-cycles:

1. $p = n$, $c = 0$;
2. **While** $c + p \neq n - R(G)$ **Do**
3.   **If** $c = n - R(G) - 1$ and $p > 1$ **Then**
4.   Choose randomly two free vertices belonging to two different paths, and link them with a gray edge;
5.   **Else**
6.   Choose randomly two free vertices and link them with a gray edge;
7.   **End of if**
8.   update p and c;
9. **End of while**
10. Close each path individually;

Lemma 3. Algorithm construct-cycles *constructs a graph with* $R(G)$ *cycles.*

Proof. The algorithm links randomly two free vertices, until we reach one of the two situations:

1. $c = n - R(G) - 1$ and $p > 1$. In that situation, closing one path would create too many cycles. The only possibility is thus to link all the remaining paths together, by choosing successively pairs of free vertices belonging to different paths.
2. $c + p = n - R(G)$. In that case, linking two paths would create less than the required number of cycles. Therefore, the only possibility is to close each remaining path □

The problem is now to use the graph constructed by *Algorithm construct-cycles* to generate a random genome of reversal distance $R(G)$. Notice that the correspondence is not straightforward. Indeed, the gray edges constructed by the algorithm do not necessarily give rise to a unique circular genome containing all the genes of G. Instead, we can end up with a set of circular fragments, where a fragment contains a subset of the genes of G. The problem is then to link these fragments without changing the number of cycles of the graph.

Let F_1, F_2 be two circular fragments obtained with *Algorithm construct-cycles*. Let a_1, b_1 be two adjacent genes in F_1, and a_2, b_2 two adjacent genes in F_2. As a_1, b_1 are connected by a gray edge, they belong to one cycle of the graph. The same holds for a_2, b_2. Therefore, a_1, b_1, a_2, b_2 are in the same cycle or in two different cycles. The (a) part of Figure 4 illustrates one (over two) possible situation with one cycle. The (b) part of Figure 4 illustrates one (over four) possible situation with two cycles. In that case, the other endpoints of the gray edges should be considered. There are different situations depending on the distribution of the 8 vertices in 2, 3 or 4 fragments. Figure 4.b represents the case of 4 fragments. The right part of Figure 4 shows that it is possible to destroy the "bad" gray edges, and construct new ones, in such a way that the total number of cycles is not modified. All the other situations are treated in a similar way, and are not detailed here.

After the correction described above, we end up with a circular genome represented by a breakpoint graph with $c = n - R(G)$ cycles. If the graph does not contain any hurdle, then the genome has the reversal distance $R(G)$. Otherwise, we run the algorithm until we obtain a graph without hurdles. Usually, a unique run of the algorithm is sufficient to obtain an appropriate random genome.

6 Verifying the replication-directed reversal hypothesis

Tillier and Collins [TC00] have determined plots for the relative positions of orthologous pairs of genes in pairwise comparisons of bacterial genomes. In most cases, a high number of orthologous are located at the

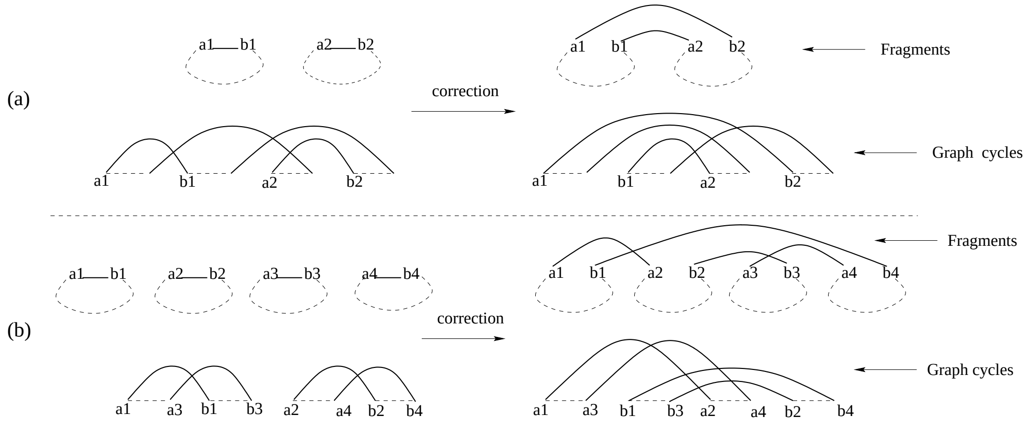

Fig. 4. Thick lines represent gray edges and thin lines are alternating paths of black and gray edges, ending with black edges. Circles represent circular fragments. Left is the situation after Algorithm construct-cycles, and right is the correction to decrease the number of fragments without changing the number of cycles. (a) One possible situation with one cycle. (b) One possible situation with two cycles.

same relative position and orientation in both genomes. But the most interesting fact is that the other genes that have moved are not randomly distributed throughout the genome. Instead, they form a perpendicular diagonal line that intersects the first line at approximately the position of the termination of replication. Moreover, the genes in these orthologous pairs have opposite signs. Of course, the rate of recombination varies from one pairwise comparison to another, and the diagonal is more or less clear depending on the plot. These observations have led to the hypothesis that apparent transpositions are actually resulting from pairs of consecutive reversals at different sites. Moreover, the segments involved in these reversals are not randomly distributed in the genome. Rather, they are surrounding one of the two endpoints of the replication axis, and this endpoint is approximately situated in the middle of the segment. We refer to this hypothesis as the replication-directed reversal hypothesis. Figure 5 is an example of a reversal across the replication axis. The size of a reversal is the length $l = l_1 + l_2$ of the segment involved in the reversal, where l_1 (respectively l_2) is the length of the sub-segment on the left side (respectively right side) of the replication axis endpoint. The lengths are represented as a percentage of the total size of the genome, in term of nucleotide number. Notice that if $l > 50\%$, the complementary reversal, obtained by taking the complementary segment in the genome, gives rise to the same permutation. In particular, a reversal containing the two endpoints of the replication axis can be replaced by the complementary one that does not contain any of these two endpoints. Therefore, we consider only the reversals for which $l \leq 50\%$.

We define the asymmetric coefficient $C = \frac{|l_1 - l_2|}{l}$. If $C = 0$, the reversal is perfectly symmetric, and if $C = 1$, the reversed segment does not cross the replication axis.

Our goal here is to test the replication-directed reversal hypothesis in bacterial genomes.

Method: We ran our algorithm on various pairs of bacterial genomes. For one of them, *C.pneumoniae/C.trachomatis*, the two diagonals are clear in the plot (relative positions of orthologous pairs of genes), while for the others, they are hardly visible [TC00]. We used simplified data: the genes present in only one genome have not been considered. This simplification is possible because it does not alter the two diagonals on the plots. However, we considered the real nucleotide distance between pairs of genes.

As no evidence for any particular weighting is established, we first selected, at each step of the algorithm, one random reversal from the set of all safe reversals. In that way, running the algorithm several times gives rise to several possible solutions. Let S be the set of all reversals obtained by running our algorithm t times. To verify the hypothesis, it is not sufficient to count the reversals that have an asymmetric coefficient higher

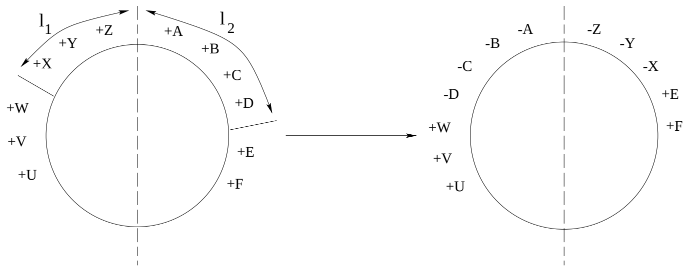

Fig. 5. A reversal surrounding one of the two endpoints of the replication axis. l_1 (respectively l_2) is the length of the sub-segment on the left side (respectively right side) of the replication axis endpoint. A "valid" reversal is a reversal such that $\frac{|l_1-l_2|}{l_1+l_2}$ is close to 0.

than a threshold value, and compare it with the value obtained for a random genome. Indeed, in real data, short reversals are predominant. And as the probability of having a reversal with an asymmetric coefficient close to 0 depends on the reversal size (a short reversal has a high probability of being asymmetric), the size of the reversals should be considered in our experiments. Therefore, we plotted the reversals of the set S depending on their size (Figure 6), and compared them with theoretical results, described in the next paragraph.

Consider a random permutation of n genes, and suppose they are uniformly distributed on the total length of the genome. Then, the size of a reversal can be viewed as the number of genes in its segment. For a reversal R of length l, let X be the random variable whose value is equal to the asymmetric coefficient C of R.

Lemma 4. *The mean and variance of the random variable X are respectively:*

$$E[X] \sim 1 - \frac{l}{n}, \qquad Var[X] \sim 1 - \frac{4(s^2-1)}{3ns} - \frac{(n-s)^2}{n^2}$$

Proof. We want to compute the value of $P\{X = C\}$, for $0 \leq C \leq 1$.

The total number of possible reversals of length l is n if $l < \frac{n}{2}$, and $\frac{n}{2}$ if $l = \frac{n}{2}$. Now let evaluate the number of possible reversals of length l depending on the asymmetric coefficient C:

1. $\mathbf{C = 0}$: The number of possible reversals is 2 if $l < \frac{n}{2}$, and 1 if $l = \frac{n}{2}$. Thus, in both cases, $P\{X = 0\} = \frac{2}{n}$.
2. $\mathbf{0 < C < 1}$: The number of possible reversals is 4 if $l < \frac{n}{2}$, and 2 if $l = \frac{n}{2}$. Thus, in both cases, $P\{X = C\} = \frac{4}{n}$.
3. $\mathbf{C = 1}$: If $l < \frac{n}{2}$, the number of possible reversals is $2(\frac{n}{2} - l + 1) = n - 2l + 2$, and $P\{X = 1\} = 1 - 2\frac{s-1}{n}$; if $l = \frac{n}{2}$, there is only 1 possible reversal, and thus $P\{X = 1\} = \frac{2}{n}$. But for $l = \frac{n}{2}$, $1 - 2\frac{s-1}{n} = \frac{2}{n}$. Thus, in both cases, $P\{X = 1\} = 1 - 2\frac{s-1}{n}$.

The set of all possible values of C other than 0 and 1 is $\{\frac{2i}{l}$, for $1 \leq i \leq \frac{l-2}{2}\}$ if l is even, and it is $\{\frac{2i-1}{l}$, for $1 \leq i \leq \frac{l-1}{2}\}$ if l is odd.

Now, $E[X] = \sum_{0 \leq C \leq 1} CP\{X = C\} = 1 - 2\frac{s-1}{n} + \frac{4}{n}\sum_{0<C<1} C$. By adding the different values of C we obtain:

1. If l is even, $E[X] = 1 - \frac{l}{n}$.
2. If l is odd, $E[X] = 1 - \frac{l}{n} + \frac{1}{nl}$.

Now, $Var(X) = E[X^2] - (E[X])^2$ with $E[X^2] = \sum_{0 \leq C \leq 1} C^2 P\{X = C\}$. By adding the different values of C we find $E[X^2] = 1 - \frac{4(l^2-1)}{3nl}$ for any l. Therefore:

1. If l is even, $Var[X] = 1 - \dfrac{4(l^2-1)}{3nl} - \dfrac{(n-l)^2}{n^2}$.

2. If l is odd, $Var[X] = 1 - \frac{4(l^2-1)}{3nl} - (\frac{n-l}{n} + \frac{1}{nl})^2$

As $\frac{1}{nl}$ is negligible in comparison to the other terms, we omit it in both $E[X]$ and $Var(X)$. The results are then the same for l even and l odd $\square$

From Lemma 4, it follows that the function $f(l) = E[X]$ is a straight line. It is represented by a gray line in the diagrams of Figures 6. Notice that this theoretical line is obtained by assuming that the probability of picking a reversal of size l is the same for any l. However, as our algorithm only chooses sorting reversals (reversals leading to the reversal distance), and as these reversals are chosen on specific cycles of the graph, the assumption of equiprobability on reversals does not hold for the solutions of our algorithm. To have a better idea about this bias, we ran our algorithm several times on random permutations. We plotted the reversals and obtained a second straight line falling below the theoretical line. This is represented by a black line in the diagrams of Figures 6.

Finally, we weighted the safe reversals by their asymmetric coefficients, and we used our branch-and-bound algorithm to find a solution of minimal weight. Due to the fact that the pairs of genomes tested have large reversal distances and that the weighting function does not permit an extensive pruning of the search tree, we had to limit the search to three safe reversals at most at each step of the algorithm.

Figure 7 shows the results obtained for three pairwise comparisons. To evaluate these results, we compared them with those obtained for a random genome of the same length and reversal distance. These random genomes were generated by the algorithm described in Section 5.

Discussion In [TC00], plots for the relative positions of orthologous pairs of genes in pairwise comparisons of bacterial genomes have been determined. Depending on the rate of recombination for the considered pairwise, the diagonals are more or less visible. For example, the diagonals are clearly identifiable for *C. pneumoniae/C.trachomatis*, hardly visible for *L.lactis/S.pyogenes*, and completely invisible for *C.jejuni/H.pylori*.

The first diagram of Figure 6 shows the curve obtained for *C. pneumoniae/C.trachomatis*. Most points, except the ones for very short reversals, are far below both theoretical lines. In this case, the replication-directed reversal hypothesis is strongly validated. This is further supported by the diagram of Figure 7, where all points are far below the line corresponding to a random genome of the same reversal distance.

Strong support is also provided for the pairwise *S.aureus/S. pyogenes* (second plot of Figure 6 and Figure 7). This is an interesting result, as the diagonals are not clearly represented in the plot of orthologous genes positions. For *V. Cholerae/E. Coli*, we still observe most points below the two theoretical lines. However, nothing can really be deduced from the diagrams of *L.lactis/S.pyogenes* and *U.urealyticum/M. pneumoniae*, and the results for *C.jejuni/H.pylori* fit almost perfectly with the line corresponding to a random genome.

7 Conclusion

Although our new algorithm remains a heuristic, and the branch-and-bound is time and space consuming, it is well suited to test different kinds of hypothesis on rearrangement mechanisms. Indeed, as at each step of the sorting procedure the complete set of safe reversals is known, it is easy to choose, from this set, the reversals that we want to test. This flexibility has been very useful for testing the replication-directed reversal hypothesis. The preliminary results that we have obtained confirm this hypothesis for reasonably shuffled genomes. In the future, we would like to account for other characteristics of reversals to further understand their mechanism in bacterial genomes. In particular, we would like to account for deleted genes and gene families.

The key idea of our method is to take advantage of an exact parsimony approach to develop an efficient general heuristic. This can be extended to other rearrangement operations. In particular, a new heuristic for weighted reversals and transpositions based on the same idea can be envisaged.

References

[Ber01] A. Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. In *12th Annual Symposium on Combinatorial Pattern Matching*, Lecture Notes in Computer Science. Springer Verlag, 2001.

[BH96] P. Berman and S. Hannenhalli. Fast sorting by reversals. In *Seventh Annual Symposium on Combinatorial Pattern Matching*, Lecture Notes in Computer Science, pages 168- 185. Springer-Verlag, 1996.

[BKS96] M. Blanchette, T. Kunisawa, and D. Sankoff. Parametric genome rearrangement. *Gene-Combis*, 172:11-17, 1996.

[BMY01] D.A. Bader, B.M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distances between signed permutations with an experimental study. *J. Comput. Biol.*, 8(5):483-491, 2001.

[BP98] V. Bafna and P.A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, 1998.

[EM00] N. El-Mabrouk. Sorting signed permutations by reversals and insertions/deletions of contiguous segments. *to appear in Journal of Discrete Algorithms, special issue on Matching Patterns*, 2000.

[Han95] S. Hannenhalli. Polynomial-time algorithm for computing translocation distance between genomes. In Z. Galil and E. Ukkonen, editors, *Sixth Annual Symposium on Combinatorial Pattern Matching*, volume 937 of *Lecture Notes in Computer Science*, pages 162–176, Berlin, 1995. Springer.

[HP95a] S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip. (polynomial algorithm for sorting signed permutations by reversals). In *Proceedings of the 27th Annual ACM-SIAM Symposium on the Theory of Computing*, pages 178–189, 1995.

[HP95b] S. Hannenhalli and P.A. Pevzner. Transforming men into mice. In *Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science*, pages 581–592, 1995.

[KS95] J.D. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13:180–210, 1995.

[KST97] H. Kaplan, R. Shamir, and R.E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1997.

[Sie02] Adam C. Siepel. An algorithm to find all sorting reversals. In *Proceedings of the sixth Annual International Conference on Computational Molecular Biology (RECOMB*, Washington, 2002. ACM. to appear.

[TC00] E.R.M. Tillier and R.A. Collins. Genome rearrangement by replication-directed translocation. *Nature Genetics*, 26:195 - 197, October 2000.

[WDM98] M. E. Walter, Z. Dias, and J. Meidanis. Reversal and transposition distance of linear chromosomes. In *String Processing and Information Retrieval: A South American Symposium (SPIRE '98)*, 1998. Submitted to Journal of Computational Biology.

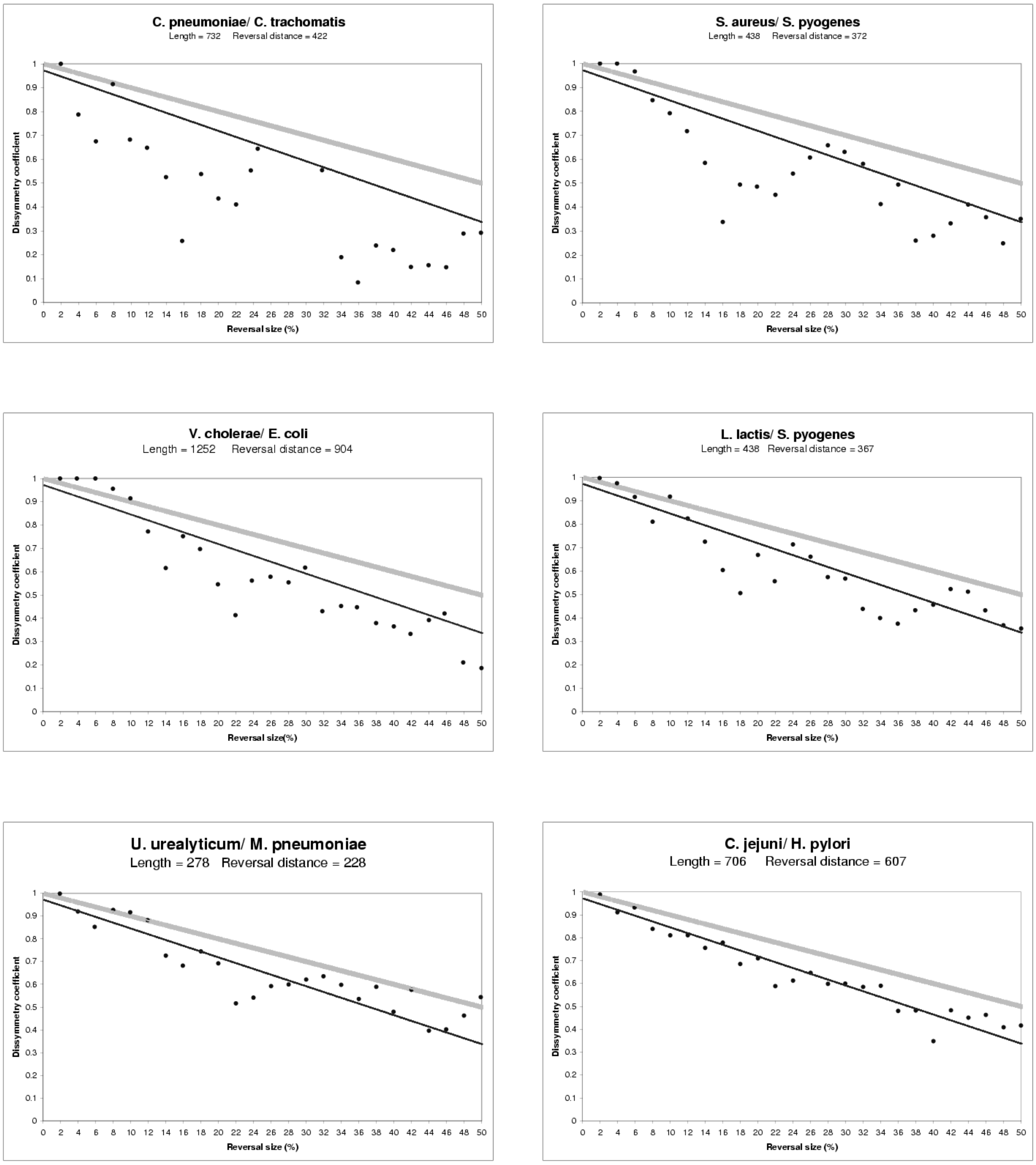

Fig. 6. The diagrams obtained for pairwise comparisons of bacterial genomes. The horizontal axis is the reversal size, written as a percentage of the total length of the genome. The vertical axis it the asymmetric coefficient. Dots represent the reversals obtained by our algorithm. The gray line represents the theoretical mean, and its width is the theoretical variance. The black line is the one obtained by running our algorithm on random permutations. For each pairwise comparison, the reversal distance and the length of the genomes (number of common genes) are given.

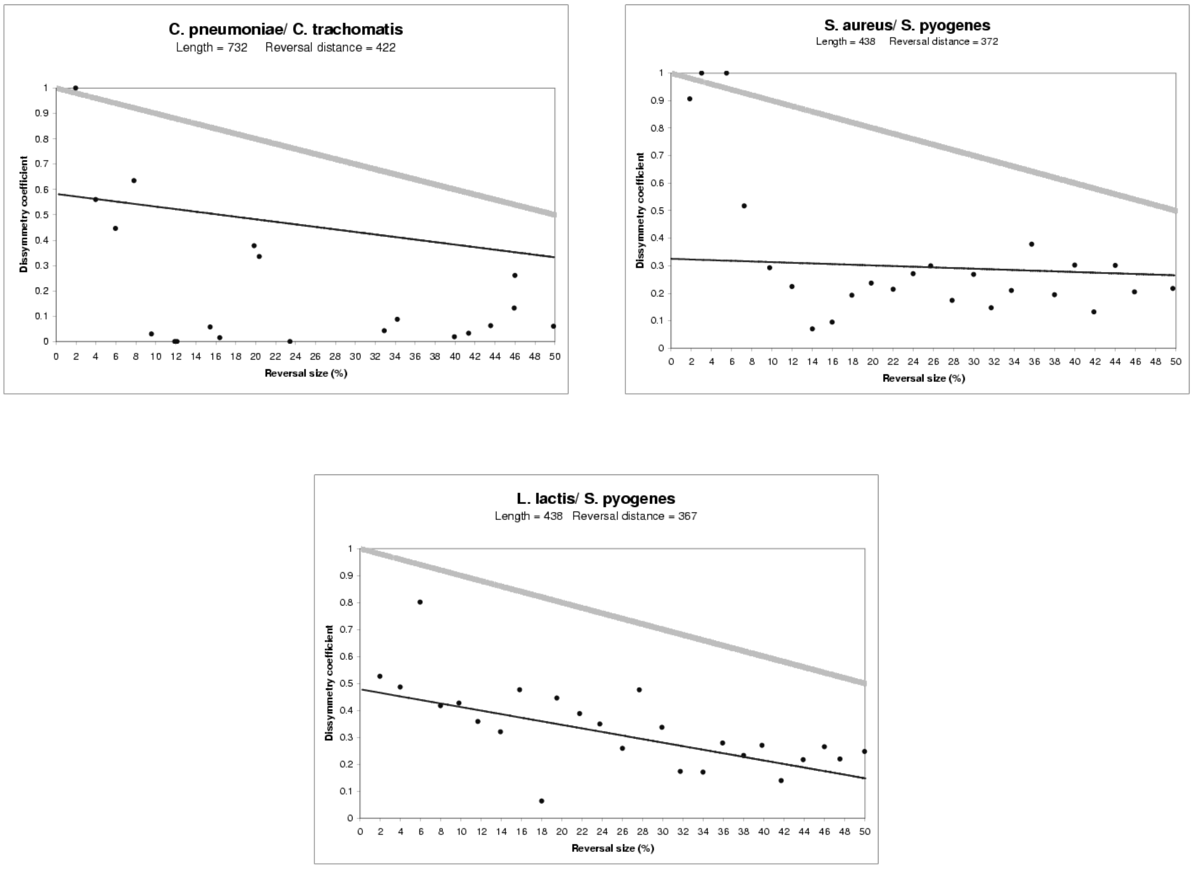

Fig. 7. The diagrams obtained for two pairwise comparisons by our branch-and-bound algorithm. The black thin line is obtained by running the algorithm on a random sequence of the same reversal distance than the tested genomes. The gray line corresponds to the theoretical line of Figure 6.