Sorting signed permutations by reversals and insertions/deletions of contiguous segments

Nadia El-Mabrouk
mabrouk@iro.umontreal.ca

Département d'informatique et de recherche opérationnelle, Université de Montréal, CP6128 succursale Centre-Ville, Montréal, Québec H3C 3J7

Abstract

Analysis of genome rearrangements allows to compare molecular data from species that diverged a very long time ago. Results and complexities are tightly related to the type of data and genome-level mutations considered. For sorted and signed data, Hannenhalli and Pevzner (HP) developed the first polynomial algorithm in the field. This algorithm solves the problem of sorting by reversals. In this paper, we show how to extend the HP approach to include insertions and deletions of gene segments, allowing to compare genomes containing different genes. We provide an exact algorithm for the asymmetric case, as applies in organellar genome evolution, and a heuristic for the symmetric case, with bounds and a diagnostic for determining whether the output is optimal.

1 Introduction

The polynomial algorithm of Hannenhalli and Pevzner (hereafter "HP") for sorting by reversals [HP95a] was a breakthough for the formal analysis of evolutionary genome rearrangement. Moreover, they were able to extend their approach to include the analysis of interchromosomal translocations [HP95b, Han95]. The applicability of this theory, however, has been limited by the difficulty in extending it to other rearrangement mechanisms, such as gene duplication, or the transposition, duplication or insertion of genomic fragments. In this paper, I show how to extend the HP approach to include insertions and deletions of gene segments, providing an exact algorithm for the asymmetric case, as applies in organellar genome evolution, and a heuristic for the symmetric case, with bounds and a diagnostic for determining whether the output is optimal.

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 prokaryotes. We assume that the direction of transcription is known. Formally, given a set $\mathcal{A}$ of genes, a genome is a circular string of signed ($+$ or $-$) elements of $\mathcal{A}$, no gene appearing more than once. A **gene segment** is a subsequence of consecutive genes in a genome. 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.

Let G and H be two genomes with some genes in common, and others specific to each genome. Write $\mathcal{A}$ for the set of genes in both G and H, and $\mathcal{A}_G$ and $\mathcal{A}_H$ for those in G

and H only respectively. Our problem is to find the minimum number of reversals, insertions and deletions of gene segments necessary to transform G into H, with the restriction that deletions and insertions are not applied on genes present in both genomes G and H. Although such manipulations may be meaningful in some circonstances, in this paper we permit insertions/deletions of genes only in $\mathcal{A}_G$ or $\mathcal{A}_H$.

2 The Hannenhalli-Pevzner theory.

Given a set $\mathcal{A}$ of genes and two circular genomes G and H such that each gene of $\mathcal{A}$ appears exactly once in each genome, how can we find the minimum number of reversals necessary to transform G into H? The HP solution [HP95a] is an exact polynomial algorithm, based on a bicoloured cycle graph constructed from G and H. This graph will be the focus of our extension to the problem to include segment insertion and deletion. We first sketch the HP method.

The HP bicoloured **cycle graph** $\mathcal{G}$ of G with respect to H is 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 $\mathcal{G}$ are just the x^t and the x^h for all x in $\mathcal{A}$. Without confusion, we can also apply the terminology "vertices of G" to mean elements like x^t or x^h. Any two vertices which are adjacent in G, other than x_i^t and x_i^h from the same x, are connected by a black edge, and any two adjacent in H, by a gray edge. Each vertex is incident to exactly one black and one gray edge, so that there is a unique decomposition of $\mathcal{G}$ into c disjoint cycles of alternating edge colours. By the **size of a cycle** we mean the number of black edges it contains. Note that c is maximized when $G = H$, in which case each cycle is of size 1 and $c = |\mathcal{A}|$.

A reversal is determined by the two points where it "cuts" the current genome, which correspond each to a black edge. We say that a **reversal ρ is determined by the two black edges** $e = (a, b)$ and $f = (c, d)$ if ρ reverses the segment of G bounded by vertices b and c. The **reversal determined by a gray edge** g is the reversal determined by the two black edges adjacent to g. A reversal may change the number of cycles, so that minimizing the number of operations can be seen in terms of increasing the number of cycles as fast as possible.

Let e and f be two black edges of a cycle C of length greater than 1. Think of both e and f as being oriented clockwise in the graph. Each one of them induces an orientation of C. If the orientation induced by e and f coincide, we say that they are **convergent**; otherwise we say that they are **divergent**. Let ρ be a reversal determined by the two black edges e and f, and $\Delta(c)$ the increase in the number of cycles of the graph after the reversal ρ.

1. If e and f belong to different cycles, then $\Delta(c) = -1$.
2. If e and f belong to the same cycle and converge, then $\Delta(c) = 0$.
3. If e and f belong to the same cycle and diverge, then $\Delta(c) = 1$.

The goal is to perform as many reversals of the third type as possible. A key concept in the method is the **unoriented components**. A gray edge g in a cycle of size > 1 is **oriented** if the two black edges adjacent to g diverge. Otherwise, the edge is **unoriented**. A cycle C is **oriented** if it contains at least one oriented gray edge. Otherwise it is **unoriented**. We say that two gray edges g_1 and g_2 **cross** if g_1 links vertices a and c, g_2 links vertices b and d, but these vertices are ordered a, b, c, d in G. Two cycles containing gray edges that cross are **connected**. A (connected) **component** of $\mathcal{G}$ is a subset of the cycles, built recursively from one cycle, at each step adding all the remaining cycles connected to any of those already in the construction. A **good component** has at least one oriented cycle and thus at least one oriented gray edge. Otherwise it is a **bad component**.

HP showed that an oriented component can be **solved**, i.e transformed to a set of cycles of size 1, by a series of **good reversals**, that is reversals of the third type. However bad components often require **bad reversals** (reversals of the first and second type). The set of bad components is subdivided into subsets, depending on the difficulty to solve them (i.e. transform them into good components). This subdivision is explained below.

We say that component U **separates** two components U' and U'' if any potential edge we tried to draw from a vertex of U' to one of U'' would cut a gray edge of U. A **hurdle** is a bad component which does not separate any pair of bad components. We further distinguish between two types of hurdles. A hurdle $\mathcal{H}$ is a **superhurdle** if the elimination of $\mathcal{H}$ from $\mathcal{G}$ transforms a non-hurdle $\mathcal{H}'$ into a hurdle, and is a **simple hurdle** if not. A cycle graph $\mathcal{G}$ is a **fortress** if it contains an odd number of hurdles, and they are all superhurdles.

The underlying idea is that a bad component U that separates two bad components U' and U'' is automatically solved by solving U' and U'', and thus, can be considered as a good one. On the other hand, a hurdle requires bad reversals to be solved, and a superhurdle is a "bad hurdle", giving rise to additional problems.

To underestand how hurdles are solved, we require the following remark, proved by HP:

Remark 1 :

- A reversal determined by two divergent black edges in a single unoriented cycle C transforms C into an oriented cycle C'. The total number of cycles of the graph remains unchanged.
- A reversal determined by two black edges from two different unoriented cycles C_1 and C_2 transforms C_1 and C_2 into a single oriented cycle C. The total number of cycles in the graph decreases by 1.

In order to solve hurdles, HP perform **merging** of appropriate pairs of hurdles. More precisely, components U' and U'' are solved by performing a particular reversal acting on one edge of an unoriented cycle of U' and one edge of an unoriented cycle of U'', which transforms U' and U'' into one good component. In the case of an odd number of hurdles, some hurdles are **cut**, by performing a reversal of the first type on two edges of the same cycle of a hurdle, which transforms that hurdle into a good component. These operations are more precisely described in the HP algorithm (Figure 1). We require one more notion. Two hurdles $\mathcal{H}_1$ and $\mathcal{H}_2$ are **consecutive** in $\mathcal{G}$ if there is no hurdle $\mathcal{H}$ in $\mathcal{G}$ that has all its vertices between any vertex of $\mathcal{H}_1$ and any vertex of $\mathcal{H}_2$ (clockwise or counterclockwise).

HP showed that the minimum number of reversals necessary to transform G into H is:

$$R(G,H) = b(G,H) - c(G,H) + h(G,H) + fr(G,H) \quad \mathbf{(1)}$$

where $b(G,H)$ is the number of black edges, $c(G,H)$ is the number of cycles (of size ≥ 1), $h(G,H)$ is the number of hurdles of $\mathcal{G}$, and $fr(G,H)$ is 1 if $\mathcal{G}$ is a fortress and 0 otherwise.

The HP algorithm for transforming G into H with a minimum of reversals is described below. Note that $h(G, H)$ changes value during the execution of the algorithm. In [HP95a], it is shown that at each step, a reversal satisfying the conditions of the algorithm can be found. For a more detailed presentation of the HP formula and algorithm, see also [SM-97].

Algorithm HP:
While $\mathcal{G}$ contains a cycle of size > 1
do
    If $\mathcal{G}$ contains a good component $\mathcal{CC}$
        Choose an oriented gray edge g of $\mathcal{CC}$ such that the reversal determined by g does not create a new bad component;
        Execute the reversal determined by g;
    Otherwise
        If $h(G, H)$ is even {hurdle merging}
            Consider two appropriate (non-consecutive, if possible) hurdles $\mathcal{H}_1$ and $\mathcal{H}_2$;
            Arbitrarily choose a black edge e in $\mathcal{H}_1$ and a black edge f in $\mathcal{H}_2$, and execute the reversal determined by e and f;
        Otherwise
            If $h(G, H)$ is odd and there is a simple hurdle $\mathcal{H}$ {hurdle cutting}
                Execute the reversal determined by two arbitrarily chosen edges e and f belonging to a single cycle in $\mathcal{H}$;
            Otherwise {fortress, hurdle merging}
                If $\mathcal{G}$ is a fortress with more than three superhurdles
                    Consider two appropriate (not consecutive) hurdles $\mathcal{H}_1$ and $\mathcal{H}_2$;
                    Arbitrarily choose a black edge e in $\mathcal{H}_1$ and black edge f in $\mathcal{H}_2$, and execute the corresponding reversal;
                Otherwise {3-fortress, hurdle merging}
                    Consider any two hurdles $\mathcal{H}_1$ and $\mathcal{H}_2$;
                    Arbitrarily choose a back edge e in $\mathcal{H}_1$ and a black edge f in $\mathcal{H}_2$, and execute the corresponding reversal;

Figure 1: The HP algorithm for sorting genomes by reversals.

We extend the HP method to take into account deletions in Section 3, insertions in Section 4, and deletions plus insertions in Section 5.

Remark 2 : In the case of an even number $2k$ of hurdles, the algorithm performs $k-1$ merging of pairs of non-consecutive hurdles, followed by a merging of the remaining consecutive two hurdles.

3 Deletions

Returning to the problem at hand, we require the minimum number of **rearrangement operations**, i.e., reversals, insertions and deletions, necessary to transform genome G into genome H. We first consider the case where there are no insertions. All genes in H are also in G, i.e. $\mathcal{A}_H = \emptyset$. The problem is then to transform G into H with a minimum of reversals and deletions. Given that the genes of $\mathcal{A}_G$ are destined to be deleted, their identities and signs are irrelevant (a segment does not need to have all its genes with the same orientation to be deleted), and could be replaced with any symbols different from those used for the genes in $\mathcal{A}$. For any subsequence of vertices of G of form $S = u_1 u_2 \cdots u_{p-1} u_p$, where u_1, u_p correspond to two genes of $\mathcal{A}$, and for all i, $2 \leq i \leq p-1$, u_i corresponds to a gene of $\mathcal{A}_G$, we replace S by the subsequence $S' = u_1 \delta(u_1) u_p$.

Example 1 Consider sets $\mathcal{A} = \{a, b, c, d, e, f\}$, $\mathcal{A}_G = \{g, h, i, j, k, l\}$, and the circular genome G (i.e. the first term follows the last one): $+g\ +a\ -h\ +b\ +c\ +i\ +d\ +l\ +e\ -j\ +k\ +f$. Then G can be rewritten: $\delta(f^h)\ +a\ \delta(a^h)\ +b\ +c\ \delta(c^h)\ +d\ \delta(d^h)\ +e\ \delta(e^h)\ +f$.

We represent genomes G and H by the bicoloured cycle graph $\mathcal{G}(G, H)$. Denote by $\mathbf{V}$ the set of vertices, B the set of black edges and D the set of gray edges of $\mathcal{G}(G, H)$. These three sets defined as follows :

- $\mathbf{V} = \{x^s\}_{x \in \mathcal{A}}^{s \in \{h,t\}}$.
- The black edges pertain to genome G. There are two sorts of black edges: **direct** black edges link two adjacent vertices in G, and **indirect** black edges link two vertices separated by a δ. For an indirect black edge $e = (a, b)$, $\delta(a)$ is the **label of e**.
- Gray edges link adjacent vertices in H.

An **indirect cycle** is one containing at least one indirect edge. A hurdle containing at least one indirect cycle is an **indirect hurdle**, otherwise it is a **direct hurdle**.

Example 2 Consider the genome G of Example 1 and the genome $H = +a\ +d\ +c\ +b\ +f\ +e$. The graph $\mathcal{G}(G, H)$ corresponding to these two genomes is depicted in Figure 2. This graph is made up of two cycles, C_1 and C_2, both indirect. These two cycles are unoriented and form a hurdle.

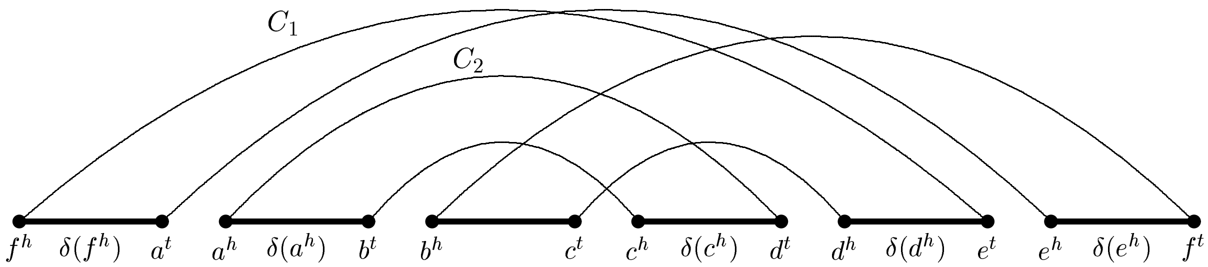

Figure 2: The black edges with a δ are indirect, i.e., there are genes of $\mathcal{A}_G$ between the two vertices that determine each of these edges.

The **reduced genome** $\overline{G}$ induced by G is obtained by deleting from G all the genes in $\mathcal{A}_G$. The cycle graph $\mathcal{G}(\overline{G}, H)$ associated with genomes $\overline{G}$ and H is obtained from $\mathcal{G}(G, H)$ by considering all the black edges of B to be direct.

For a reduced genome, the reversal acting on the two black edges $e = (a, b)$ and $f = (c, d)$ reverses the segment of the genome $\overline{G}$ bounded by vertices b and c. For a genome G containing genes to be deleted, the indirect edges represent, not an adjacency in the genome G, but rather an interval containing only genes to be deleted. We thus have to define what we mean by "the reversal acting on two black edges", of which at least one is indirect.

Let $e = (a, b)$ and $f = (c, d)$ be two indirect edges. The interval $[b, c]$ designates the interval of G bounded by the two vertices b and c; the interval $[b, \delta(c)]$ designates the interval of G bounded on the left by b and on the right by the element of $\mathcal{A}_G$ adjacent to d.

Definition 1 *Consider two black edges $e = (a, b)$ and $f = (c, d)$. The* **reversal ρ determined by e and f** *is defined as follows. If e and f are not both indirect, ρ reverses segment $[b, c]$. Else (e and f are both indirect), ρ reverses segment $[b, \delta(c)]$ (see Figure 3.a).*

Consider an oriented gray edge $g = (b, d)$ adjacent to two black edges $e = (a, b)$ and $f = (c, d)$. The **reversal ρ determined by** *g reverses segment $[b, c]$ if f is direct, and reverses segment $[b, \delta(c)]$ if f is indirect (see Figure 3.b).*

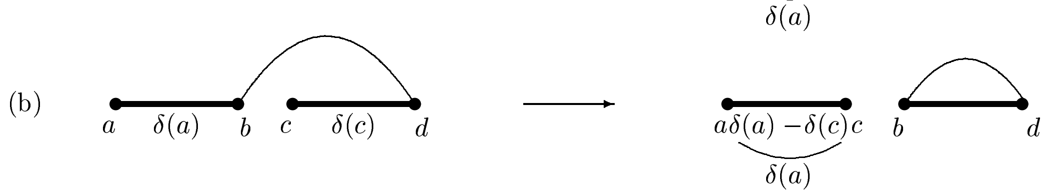

Figure 3: (a) A reversal determined by two indirect black edges. (b) A reversal determined by a gray edge adjacent to two indirect black edges. This creates a direct cycle of size 1.

The HP algorithm can be generalized to graph containing direct and indirect edges by making use of the above definition of a reversal determined by two black edges, or by a gray edge. After each reversal, for each vertex a, $\delta(a)$ represents the sequence of genes of $\mathcal{A}_G$ following a in the resulting genome. These gene sequences change throughout the execution of the algorithm.

Lemma 1 *A reversal determined by an oriented gray edge of a cycle C transforms C into two cycles C_1 and C_2, at least one of which is of size 1. Suppose this is true of C_1. Then the black edge of C_1 is direct.*

Proof : Let $g = (b, d)$ be an oriented gray edge adjacent to two black edges $e = (a, b)$ and $f = (c, d)$. The reversal ρ determined by g reverses the segment bounded by vertices b and c, and in doing so creates a cycle of size 1 containing the black edge $f' = (b, d)$. The fact that f' is direct follows from the definition of a reversal determined by g (see Figure 3.b) □

Corollary 1 *An oriented cycle C of size n is transformed by the HP algorithm, in $n - 1$ reversals, into n cycles of size 1. If C is direct, the n cycles are all direct. If not, only one of these cycles is indirect.*

Proof : The fact that the HP algorithm transforms an oriented cycle C into n cycles of size 1 through the application of $n-1$ reversals is proved in [HP95a]. According to Lemma 1, each reversal creates a direct cycle of size 1. If C is indirect, the genes of $\mathcal{A}_G$ in C all end up in the last size 1 cycle formed □

Corollary 1 states that for each oriented cycle C the HP algorithm gathers all the genes to delete from C into a single segment. At the end of the algorithm, a single deletion is required

Procedure Indirect-hurdle-merging

1. For all i, $1 \leq i \leq p$, do
   While there are at least three hurdles in I_i do
   Perform the merging of any two non-consecutive hurdles in I_i;
   {*At the end of this step, it remains p indirect sets, each of size 1 or 2*}
2. If $p > 1$
   Let $I'_1, \cdots, I'_r$ be the obtained indirect sets of size 1
   For all $j \in \{1, 3, \cdots r - 1 - (r \; mod \; 2)\}$ do
   Merge the hurdle of I'_j with the hurdle of I'_{j+1};
   Let $I''_1, \cdots, I''_s$ be the obtained indirect sets of size 2
   For all j, $1 \leq j \leq s$, do
   Merge one of the two hurdles of I''_j with one of the two hurdles of I''_{j+1};
3. Else
   If $p = 1$, I_1 is of size 2, and there are no direct hurdles
   Merge the two hurdles of I_1.

Figure 4: A suitable order for hurdle merging

for each oriented cycle. Now, consider the case of unoriented cycles forming hurdles. HP solve hurdles by merging and cutting them in a specific way, described in the HP algorithm (Figure 1). The merging of two hurdles $\mathcal{H}_1$ and $\mathcal{H}_2$ is achieved by combining two unoriented cycles C_1 and C_2, one from each of the two hurdles. This gets rid of the two hurdles, and creates an oriented cycle. This cycle is then solved by HP as described above. So that there are as few deletions as possible, the strategy is to carry out as many merging as possible of pairs of indirect cycles. This requires a careful hurdle choice, and cycle choice for each hurdle, within the HP algorithm. Notice that, according to HP algorithm, merging occurs only when the total number of hurdles is even. In order to keep as many indirect hurdles as possible (to be able to perform as many merging as possible of pairs of indirect hurdles), if the initial number of hurdles is odd and one hurdle must be cut, we will prefer to do this to a direct hurdle. In summary:

hurdle merging
Choose two non-consecutive hurdles $\mathcal{H}_1$ and $\mathcal{H}_2$, indirect if possible, in a certain way;
Choose e in $\mathcal{H}_1$ and f in $\mathcal{H}_2$, indirect if possible;
hurdle cutting
Choose a simple hurdle $\mathcal{H}$, direct if possible;
Carry out the reversal determined by any two black edges from the same cycle of $\mathcal{H}$.

It remains to specify precisely how to choose indirect hurdles to merge, when the total number of hurdles is even. Let $\mathcal{SH}$ be the set of hurdles of $\mathcal{G}(G, H)$, and $ih(G, H)$ be the number of indirect hurdles of $\mathcal{G}(G, H)$. We subdivide $\mathcal{SH}$ into subsets I_1, $D_1, \cdots I_p$, D_p, where each I_i is a maximal set made up of consecutive indirect hurdles, bounded at either end by maximal sets D_{i-1} and D_{i+1} of consecutive direct hurdles. The choice of a first set I_1 and a first set D_1 is arbitrary. We set $I_{p+1} = I_1$, and $I_0 = I_p$ (and similarly for D_{p+1} and D_0). The sets I_i, for $1 \leq i \leq p$, are called indirect. The order in which indirect hurdle merging occurs is given by the procedure **Indirect-hurdle-merging** described in Figure 4.

Let $\mu(G,H)$ be the number of merging performed by procedure Indirect-hurdle-merging.

Lemma 2 *Procedure Indirect-hurdle-merging is correct, i.e. it respects the merging constraints of the HP algorithm. Moreover, if $p=1$, $ih(G,H)$ is even and there is at least one direct hurdle, then $\mu(G,H) = \frac{ih(G,H)}{2} - 1$. Otherwise, $\mu(G,H) = \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$.*

Proof : To prove the correctness of the procedure, we should show that, at each step of the algorithm, if more than two hurdles remain in $\mathcal{SH}$, then a merging is operated on two non-consecutive hurdles. The first step of the algorithm clearly respects this constraint. Then, if $p>1$, the algorithm performs a series of merging, each of them operating on two hurdles belonging to two different indirect sets, and for all i and j, I_i and I_j are separated by at least two direct hurdles belonging to D_{i+1} and D_{i-1}.

Notice that, if $p=1$ and there are no direct hurdles, then, it remains exactly two hurdles, and according to remark 2, the merging of these two hurdles is also correct.

If $p=1$ and direct hurdles exist, then the algorithm stops at the end of the first step. The resulting I_1 contains one hurdle if $ih(G,H)$ is odd, and thus $\mu(G,H) = \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$, and two hurdles if $ih(G,H)$ is even, and thus $\mu(G,H) = \frac{ih(G,H)}{2} - 1$.

Suppose now $p=1$, and there are no direct hurdles. If $ih(G,H)$ is odd, then the algorithm stops at the end of the first step, and the resulting I_1 contains one hurdle, and thus $\mu(G,H) = \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$. Else, $ih(G,H)$ is even, then the algorithm proceeds with the third step, and no hurdle is left. Therefore, $\mu(G,H) = \frac{ih(G,H)}{2} = \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$.

If now $p>1$, at the end of the Indirect-hurdle-merging procedure, if $ih(G,H)$ is odd, it remains one indirect hurdle not merged, and thus $\mu(G,H) = \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$, and if $ih(G,H)$ is even, no hurdle is left not merged, and thus $\mu(G,H) = \frac{ih(G,H)}{2} = \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$ □

Lemma 3 *Let μ_P be the number of merging applied to pairs of indirect hurdles of $\mathcal{G}(G,H)$, performed by the HP algorithm, according to any order P. Then $\mu(G,H) \geq \mu_P$.*

Proof : By definition of a merging, we should have $\mu_P \leq \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$. If $\mu(G,H) = \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$, the lemma is proved. The only case for which $\mu(\mathcal{G}) \neq \left\lfloor \frac{ih(G,H)}{2} \right\rfloor$ is when $p=1$, $ih(G,H)$ is even, and at least one direct hurdle exists. Then, $\mu(G,H) = \left\lfloor \frac{ih(G,H)}{2} \right\rfloor - 1$. But in this case, an arbitrary sequence of $\left\lfloor \frac{ih(G,H)}{2} \right\rfloor - 1$ merging acting on pairs of non-consecutive indirect hurdles reduces the set of indirect hurdles to a set of two consecutive hurdles. Given that direct hurdles exist, a valid merging procedure should not merge these two last hurdles. So, in that case, $\mu_P \leq \frac{ih(G,H)}{2} - 1$ □

From now on, when we refer to the HP algorithm, this choice procedure for merging will be assumed. No specific constraints are required to merge hurdles (direct plus at most one indirect) that remain unmerged after the procedure.

Let $\mathcal{G}_1(G,H)$ be the graph containing only size 1 cycles obtained after applying the HP algorithm to the graph $\mathcal{G}(G,H)$. Consider procedure **Deletion**, described as follow, applied to $\mathcal{G}_1(G,H)$:

Procedure Deletion:
For each indirect edge $e = (a, b)$ of $\mathcal{G}_1(G, H)$
Delete the gene segment between a and b, symbolized by $\delta(a)$.

We will call the HP algorithm augmented by procedure Deletion the **Reversal-deletion algorithm**. This algorithm transforms the genome G into the genome H.

Denote by $ic(G, H)$ the number of indirect cycles of $\mathcal{G}(G, H)$, and by $R(\overline{G}, H)$ the minimum number of reversals necessary to transform $\overline{G}$ into H. This value is obtained by formula (1) of HP (section 2).

Theorem 1 *The Reversal-deletion algorithm performs $R(\overline{G}, H)$ reversals and $D(G, H) = ic(G, H) - \mu(G, H)$ deletions. Therefore: $RD(G, H) = R(\overline{G}, H) + D(G, H)$.*

Moreover, $RD(G, H)$ is the minimum number of rearrangement operations necessary to transform G into H.

Proof : From [HP95a], the HP algorithm transforms $\overline{G}$ to H with a minimum number $R(\overline{G}, H)$ of reversals. From corollary 1, the resolution of a direct cycle does not give rise to any indirect cycle of size 1, and the resolution of an indirect oriented cycle gives rise to exactly one indirect cycle of size 1. Therefore, the number $D(G, H)$ of deletions performed by the reversal-deletion algorithm, is equal to the total number of indirect oriented cycles obtained after solving the hurdles.

A merging of two hurdles consist in applying a reversal determined by two black edges, one from each hurdle. Therefore, from remark 1, a merging transforms two unoriented cycles into one oriented cycle, and thus, transforms two hurdles into one good component.

Now, from lemma 3, at most $\mu(G, H)$ merging can be performed on indirect hurdles. As each of these merging transforms two unoriented indirect cycles into one oriented indirect cycle, we have $D(G, H) \geq ic(G, H) - \mu(G, H)$. But, from lemma 2, $\mu(G, H)$ is exactly the number of merging of indirect hurdles performed by the Reversal-deletion algorithm. Therefore, $D(G, H) = ic(G, H) - \mu(G, H)$, and thus, $RD(G, H) = R(\overline{G}, H) + D(G, H)$.

Suppose that $RD(G, H)$ is not the minimum number of operations necessary to transform G to H. Given that $R(\overline{G}, H)$ is the minimum number of reversals necessary to transform $\overline{G}$ to H, it is clear that we cannot do it with fewer reversals. To reduce the number of deletions, the merging of additional indirect cycles would be necessary. But each such merging would require and additional reversal. The total number of reversals plus deletions would therefore not be less than $RD(G, H)$ $\square$

4 Insertions

The case $\mathcal{A}_G = \emptyset$, where the set of genes of G is a subset of the set of genes of H, i.e. the problem of transforming G into H with a minimum of reversals and insertions, can be solved by the reversals-deletions analysis, where G takes on the role of H, and vice versa. Each deletion in the H-to-G solution becomes an insertion in the G-to-H. Each reversal in one direction corresponds to a reversal of the same segment in the opposite direction. Nevertheless, we will examine the case of insertions in order to introduce some concepts for the subsequent analysis.

Consider then the problem of transforming H to G by a minimum number of reversals and deletions. Let $\mathcal{G}(H, G)$ be the corresponding graph. We will not, however, replace the genes of H to be deleted, by the δ parameters. Let $RD(H, G) = R(\overline{H}, G) + D(H, G)$ be the minimum number of reversals and deletions necessary to transform H into G, where $R(\overline{H}, G)$ is the number of reversals and $D(H, G)$ the corresponding number of deletions. Let $\mathcal{G}_1(H, G)$ be the graph containing only cycles of size 1 obtained by applying the HP algorithm to the graph $\mathcal{G}(H, G)$. The number of indirect cycles of this graph is $D(H, G)$ (with the definition given in theorem 1). Each indirect black edge of $\mathcal{G}_1(H, G)$ is labeled by a sequence of genes of $\mathcal{A}_H$. Consider the genome G^c obtained from G as follows: for each pair of adjacent vertices a and b in G, if (a, b) is an indirect black edge of $\mathcal{G}_1(H, G)$ labeled by the sequence $y_1 \cdots y_p$ of genes of $\mathcal{A}_H$, then insert this gene sequence between a and b in G. We call G^c the **filled genome of** G relative to H.

Lemma 4 *The minimum number of reversals and insertions necessary to transform G into H is $RI(G, H) = R(\overline{H}, G) + D(H, G)$. Moreover, $R(\overline{H}, G)$ is the number of reversals, and $D(H, G)$ is the number of insertions.*

Proof : There exists a sequence $\rho_1, \cdots, \rho_p$ of reversals and deletions that transforms H into G, where $p = R(\overline{H}, G) + D(H, G)$. To find a sequence of reversals and insertions that transforms G into H, the symmetrical operation of ρ_i should be considered for each i, where the symmetrical of a deletion is an insertion. Therefore, $RI(G, H) \leq R(\overline{H}, G) + D(H, G)$.

If there exists a sequence of less than $R(\overline{H}, G) + D(H, G)$ reversals+insertions to transform G into H, then in a symmetrical way, it should exist a sequence of less than $R(\overline{H}, G) + D(H, G)$ reversals and deletions to transform H into G: contradict the fact that $R(\overline{H}, G) + D(H, G)$ is the minimal number of operations required to tranform H into G (theorem 1) $\square$

Example 3 Consider the sets $\mathcal{A} = \{a, b, c, d, e, f, g\}$, $\mathcal{A}_H = \{y_1, y_2, y_3, y_4\}$, and the two genomes $G = +a\ +b\ +c\ +d\ +e\ +f\ +g$ and $H = +y_1\ +d\ +y_2\ +f\ +b\ +y_3\ +c\ +e\ +y_4\ +a\ +g$. Figure 5 depicts the graph $\mathcal{G}(H, G)$.

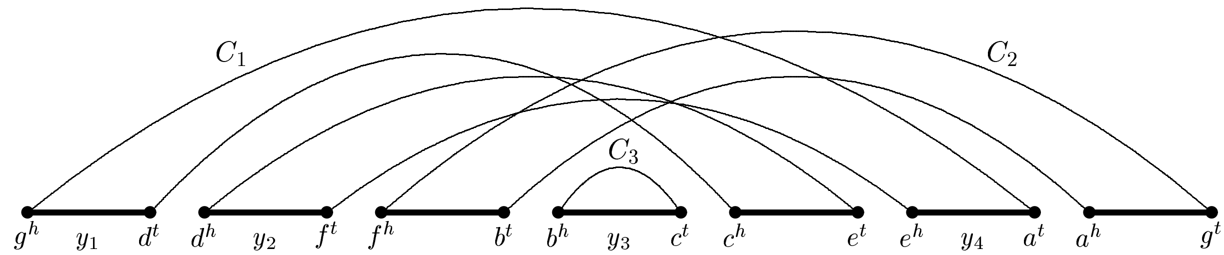

Figure 5: $\mathcal{G}(H, G)$ contains 7 black edges, 3 cycles, of which 2 are indirect, and 1 hurdle.

Given that $\mathcal{G}(H, G)$ contains 7 black edges, 3 cycles and 1 hurdle, $R(\overline{H}, G) = 7 - 3 + 1 = 5$. The HP algorithm does 5 reversals, and produces the following filled genome:

$$G^c = +y_1\ -y_2\ -y_4 + a\ +b\ +y_3\ +c\ +d\ +e\ +f\ +g$$

Two insertions are necessary to transform G into G^c.

5 Insertions and deletions

Consider now the general case where $\mathcal{A}_G \neq \emptyset$ and $\mathcal{A}_H \neq \emptyset$. Let $\overline{G}$ be the reduced genome of G, and $\overline{H}$ the reduced genome of H.

Lemma 5 *Let $RDI(G,H)$ be the minimum number of rearrangement operations necessary to transform G into H. Then $RDI(G,H) \geq R(\overline{G},\overline{H}) + D(H,\overline{G}) + D(G,\overline{H})$.*

Proof : Suppose $RDI(G,H) < R(\overline{G},\overline{H}) + D(G,\overline{H}) + D(H,\overline{G})$, that is, we can transform G into H with less operations.

From lemma 4, the minimal number of reversals+insertions necessary to transform $\overline{G}$ into H is $R(\overline{G},\overline{H}) + D(H,\overline{G})$. As a deletion of a segment of genes in $\mathcal{A}(G)$ can never lower the number of reversals or insertions required to transform G into H, we can not transform G into H with a series of reversals, insertions and deletions, where the total number of the two first operations is less than $R(\overline{G},\overline{H}) + D(H,\overline{G})$.

Therefore, $RDI(G,H) < R(\overline{G},\overline{H}) + D(G,\overline{H}) + D(H,\overline{G})$ means that we can transform G into H with a series of reversals, insertions and deletions, where the number of reversals+insertions is at least $R(\overline{G},\overline{H}) + D(H,\overline{G})$, and the number of deletions is less than $D(G,\overline{H})$. But, as we argued in the proof of theorem 1, to perform one less deletion, we need to perform one more merging of two indirect cycles, and thus one more reversal. The total number of operations necessary to transform G into H can not, therefore, be less than $R(\overline{G},\overline{H}) + D(G,\overline{H}) + D(H,\overline{G})$ $\square$

Let $\overline{G}^c$ be the filled genome of $\overline{G}$ relative to H. Given that $\overline{G}^c$ and H contain the same set of vertices, the edges of $\mathcal{G}(\overline{G}^c, H)$ must all be direct.

We must now take into account the vertices of $\mathcal{A}_G$. Let a and b be two adjacent vertices in $\overline{G}$, separated in G by a segment $X(a)$ of vertices of $\mathcal{A}_G$. If a and b are separated in $\overline{G}^c$ by a segment $Y(a)$ of vertices of $\mathcal{A}_H$, then we have the choice to place $X(a)$ before or after $Y(a)$. This choice is made so that we require as few deletions as possible to transform $\overline{G}^c$ into H. Denote G^c the genome obtained by appropriately adding the vertices of $\mathcal{A}_G$ to $\overline{G}^c$. The graph $\mathcal{G}(G^c,H)$ is obtained from $\mathcal{G}(\overline{G}^c,H)$ by transforming certain black edges of $\mathcal{G}(\overline{G}^c,H)$ into indirect edges. The procedure **Construct-G$^{\mathbf{c}}$** constructs such a graph $\mathcal{G}(G^c,H)$ corresponding to a genome G^c, from the graph $\mathcal{G}(\overline{G}^c,H)$. At each step of the construction, we say that a cycle (or a hurdle) of the obtained graph is indirect, if it contains indirect edges (constructed during previous steps). In addition, note that when an edge of the graph is labeled, it becomes indirect.

Procedure Construct-G^c should not be considered as a procedure that performs insertions of segments of genes of $\mathcal{A}_G$. The rearrangement operations that have actually been performed to obtain G^c, are insertions of segments of genes of $\mathcal{A}_H$. Therefore, the procedure Construct-**G$^{\mathbf{c}}$** should be seen as a procedure that indicates how to insert these ($Y(a)$) segments, i.e. before or after the segments of genes of $\mathcal{A}_G$ ($X(a)$), rather than a procedure that inserts the $X(a)$ segments before or after the $Y(a)$ segments.

Now, after having obtained the genome G^c from G by $D(H,\overline{G})$ suitable insertions, G^c is transformed into H, by applying the Reversal-deletion algorithm to the graph $\mathcal{G}(G^c,H)$. This algorithm performs $R(\overline{G},\overline{H})$ reversals and $D(G^c,H)$ deletions.

We summarize the different steps for transforming G into H as follows:

Procedure Construct-G^c:

For each black edge (a,b) of $\mathcal{G}(\overline{G}^c, H)$ such that a is followed by a segment $X(a)$ in G do
  If a and b are adjacent in $\overline{G}$
    Transform (a,b) into an indirect edge by labeling it by $X(a)$;
  Else, let $y_1 \cdots y_p$ be the sequence of vertices of $\mathcal{A}_H$ located between a and b in $\overline{G}^c$
    If one of the two black edges (a,y_1) or (y_p,b) belongs to an indirect cycle,
      Transform one such edge into an indirect edge by labeling it by $X(a)$;
    Else
      If one of the two black edges (a,y_1) or (y_p,b) belongs to an indirect hurdle
        Transform one such edge into an indirect edge by labeling it by $X(a)$;
      Else
        Label by $X(a)$ either one of the two edges.

Figure 6: An appropriate transformation of $\overline{G}^c$ into $\overline{G}$, by completing the graph with the genes in $\mathcal{A}_G$

1. Apply the HP algorithm to the graph $\mathcal{G}(H,\overline{G})$. The obtained graph contains $D(H,\overline{G})$ indirect edges.

2. Perform the $D(H,\overline{G})$ insertions (in G) corresponding to these indirect edges. These insertions consequently transform $\overline{G}$ into $\overline{G}^c$.

3. Fill the genome $\overline{G}^c$ to obtain genome G^c, using Procedure Construct-G^c.

4. Apply the Reversal-deletion algorithm to the graph $\mathcal{G}(G^c,H)$. The algorithm performs $R(\overline{G},\overline{H})$ reversals and $D(G^c,H)$ deletions.

Proposition 1 *The number of rearrangement operations occurring during the application of the method is:*

$$RDI_1(G,H) = R(\overline{G},\overline{H}) + D(H,\overline{G}) + D(G^c,H)$$

where $R(\overline{G},\overline{H})$ is the number of reversals, $D(H,\overline{G})$ is the number of insertions and $D(G^c,H)$ is the number of deletions.

Moreover, if RI is the minimal number of reversals and insertions in a series of rearrangement operations transforming G to H, then $RI \geq R(\overline{G},\overline{H}) + D(H,\overline{G})$

Also, if $D(G^c,H) = D(G,\overline{H})$, then $RDI_1(G,H)$ is the minimal number of rearrangement operations required to transform G to H.

Proof : The rearrangement operations performed during the application of the method are $D(H,\overline{G})$ insertions at the second step, and $R(\overline{G},\overline{H})$ reversals and $D(G^c,H)$ deletions at the fourth step.

From lemma 4, $R(\overline{G},\overline{H}) + D(H,\overline{G})$ is the minimal number of rearrangement operations required to transform $\overline{G}$ to H, and thus the minimal number of reversals+insertions in a series of rearrangement operations transforming G to H (as deletions can not lower this number of operations). Moreover, from lemma 5, if $D(G^c,H) = D(G,\overline{H})$, then $RDI_1(G,H)$ should be the minimal number of rearrangement operations required to transform G to H □

Example 4 Consider the sets $\mathcal{A} = \{a, b, c, d, e, f, g\}$, $\mathcal{A}_G = \{x_1, x_2, x_3, x_4, x_5\}$, $\mathcal{A}_H = \{y_1, y_2, y_3, y_4\}$, and the two genomes $G = +x_1 \ +a \ +x_2 \ +b \ +c + x_3 \ +d \ +x_4 \ +e \ +x_5 \ +f \ +g$, and $H = +y_1 \ +d \ +y_2 \ +f \ +b \ +y_3 \ +c \ +e \ +y_4 \ +a \ +g$.

Given that $\overline{G}$ and H are the two genomes in Example 3, Figure 5 represents graph $\mathcal{G}(H, \overline{G})$. Two insertions are necessary to transform $\overline{G}$ into $\overline{G}^c = +y_1 \ -y_2 \ -y_4 + a \ +b \ +y_3 \ +c \ +d \ +e \ +f \ +g$.

Figure 6 depicts $\mathcal{G}(G^c, H)$, which contains two indirect cycles C_1 and C_2, and a single hurdle. Therefore $D(G^c, H) = 2$. Moreover, since $R(\overline{G}, \overline{H}) = 5$ (Example 3), 5 reversals followed by 2 deletions are necessary to convert G^c to H. The total number of rearrangement operations used to transform G to H is thus $2 + 5 + 2 = 9$.

On the other hand, the construction of graph $\mathcal{G}(G, \overline{H})$ reveals that $D(G, \overline{H}) = 2$. Nine is thus the minimal number of rearrangement operations necessary to transform G in H.

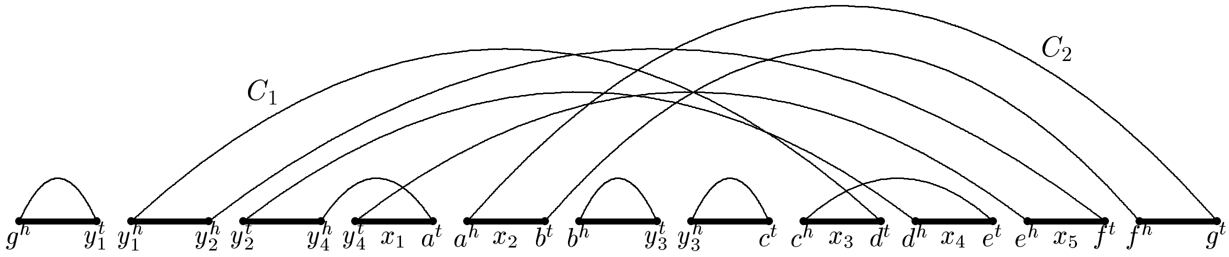

Figure 7: The graph $\mathcal{G}(G^c, H)$.

Theorem 2 *Let $ie(G, \overline{H})$ be the number of indirect edges of $\mathcal{G}(G, \overline{H})$, and let $RDI_1(G, H)$ be the number of rearrangement operations obtained by the method described above. Then:*

$$R(\overline{G}, \overline{H}) + D(H, \overline{G}) + D(G, \overline{H}) \leq RDI_1(G, H) \leq R(\overline{G}, \overline{H}) + D(H, \overline{G}) + ie(G, \overline{H})$$

Proof : The first inequality is deduced from lemma 5 and from the fact that $RDI_1(G, H) \geq RDI(G, H)$.

From proposition 1, to prove the second inequality, we should prove that $D(G^c, H) \leq ie(G, \overline{H})$. From theorem 1, $D(G^c, H) = ic(G^c, H) - \mu(G^c, H)$, and thus, $D(G^c, H) \leq ic(G^c, H)$. But as $\mathcal{G}(G^c, H)$ contains $ie(G, \overline{H})$ indirect edges, the graph contains at most $ie(G, \overline{H})$ indirect cycles. Therefore, $ic(G^c, H) \leq ie(G, \overline{H})$ $\square$

Remark 3 : A better solution will, sometimes, be found by calculating the minimum number of operations necessary to transform H to G. Thus, it is a good idea to always consider $\min(RDI_1(G, H), RDI_1(H, G))$.

The time complexity of our method depends on the complexity of the HP algorithm. If we require only the minimum number of reversals necessary to transform a genome G into a genome H, both of size n, it suffices to construct the cycle graph of G and H, and to count the number of cycles, super-hurdles and simple hurdles within it. As presented in [HP95a], the cycle graph is constructed in $O(n^2)$ time. The complexity becomes $O(n^4)$ if we want to produce an explicit sequence of reversals which transforms G to H. The HP algorithm was sped up by Berman and Hannenhalli [BH96]. The complexities they report are expressed in terms of $\alpha(n)$, the inverse of Ackerman's function. The construction of the cycle graph is $O(n\alpha(n))$ and the actual production of the reversals is $O(n^2\alpha(n))$.

Kaplan, Shamir and Tarjan [KST97] further improved on this complexity. First, they considered a simplified representation of the graph and for efficiency considerations, they avoided generating it explicitly. Secondly, they considered the notion of "happy clique" to simplify the reversal choice procedure to solve good components. Their method does not change hurdles cutting and hurdles merging procedures. With these improvements, the time complexity for the production of a minimal sequence of reversals becomes $O(n^2)$.

As our modifications to take into account insertions and deletions are not affected by the way reversals are chosen to solve good components, they are compatible with Kaplan, Shamir and Tarjan improvements. Therefore, we can assume their complexity for the HP algorithm in analyzing our extension to deletion and insertion.

Theorem 3 *Consider* $n = \max(|\mathcal{A}| + |\mathcal{A}_G|, |\mathcal{A}| + |\mathcal{A}_H|)$*. The transformation of* G *to* H *by the above method is of time complexity* $O(n^2)$*.*

Proof : Step 1. of the method is the application of the HP algorithm to $\mathcal{G}(H, \overline{G})$, and takes time $O(n^2)$. Step 2. consists in performing suitable insertions in $\overline{G}$. Given that $O(n)$ insertions are required, this is done in time $O(n)$. Step 3. is the application of procedure Construction-G^c. Its complexity is in $O(n)$, as it is proportional to the number of black edges of $\mathcal{G}(\overline{G}^c, H)$. The last step is the application of Reversal-deletion algorithm, that is the HP algorithm in $O(n^2)$ time, followed by procedure Deletion in $O(n)$ time $\Box$

Remark 4 : To obtain the parameter $D(G^c, H)$, we should construct genome G^c. To do so, the HP algorithm must be applied to the graph $\mathcal{G}(H, \overline{G})$. The complexity of our method thus remains $O(n^2)$, even if we are only looking for the minimum number of rearrangement operations, and not for the actual operations.

6 An application to the comparison of mitochondrial genomes

To test our method on some realistic data of moderate size, we selected three mitochondrial genomes from GOBASE archives [KLRB+98]. *Reclimonas americana* is thought to most resemble the ancestral mitochondrion, and other mitochondria to have evolved through deletion of segments or individual genes from the set of genes found in *Reclinomonas. Allomyces macrogynus* is a fungus and *Tetrahymena pyriformis* an unrelated protist. To simplify the problem somewhat, we discarded all tRNA genes from the gene orders, since the homological correspondences among tRNAs is not always clear in mitochondria. This left 66 genes for *Reclinomonas*, 17 for *Allomyces* and 23 for *Tetrahymena.*

Reclinomonas contains 49 genes in addition to those in *Allomyces*, and these genes are scattered among 10 segments in the genome. The transformation of the *Reclinomonas* genome to the *Allomyces* one by our algorithm requires 14 reversals but only three segment deletions.

Reclinomonas contains 43 genes in addition to those in *Tetrahymena*, and these genes are scattered among 11 segments in the genome. The transformation of the *Reclinomonas* genome to the *Tetrahymena* one by our algorithm requires 19 reversals but only two segment deletions.

Allomyces and *Tetrahymena* have 12 genes in common. The five supplementary genes in *Allomyces* are each in a separate segment, and the 11 supplementary genes in *Tetrahymena* are found in 6 segments. Converting the *Tetrahymena* genome into the *Allomyces* one by our algorithm requires 9 reversals, 2 insertions et 3 deletions. Moreover, the construction of the graph $\mathcal{G}(G, \overline{H})$ reveals that this number of operations is minimal.

7 Conclusion

The approach taken in this paper could readily be extended to a distance defined on multi-chromosomal genomes, where the rearrangement operations include reciprocal translocation, chromosome fusion and chromosome fission as well as reversal, insertion and deletion. The Hannenhalli and Pevzner approach to translocation and reversal distance [HP95b] makes use of the same kind of cycle graph that we exploit in the present research.

Acknowledgments

Research supported in part by a grant from the Natural Science and Engineering Research Council of Canada. The author is a Scholar in the Evolutionary Biology Program of the Canadian Institute for Advanced Research.

References

[BH96] P. Berman and S. Hannenhalli. Fast sorting by reversals. In D. Hirschberg and G. Myers, editors, *Combinatorial Pattern Matching, Proceedings of the 7th Annual Symposium*, number 1075 in Lecture Notes in Computer Science, pages 168–185. Springer, 1996.

[Cap97] A. Caprara. Sorting by reversals is difficult. In *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB 97)*, pages 75–83, New York, 1997. ACM.

[Cap99] A. Caprara. Formulations and hardness of multiple sorting by reversals. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB 99)*, pages 84–93, New York, 1999. ACM.

[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 (polynomial algorithm for genomic distance problem). In *Proceedings of the IEEE 36th Annual Symposium on Foundations of Computer Science*, pages 581–592, 1995.

[KLRB+98] M. Korab-Laskowska, P. Rioux, N. Brossard, TG. Littlejohn, MW. Gray, BF. Lang, and G. Burger. The organelle genome database project (gobase). *Nucleic Acids Res*, 26:139–146, 1998.

[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*, pages 344–351, New York, 1997.

[SM-97] *Introduction to Computational Molecular Biology.* PWS Publishing Company, 1997.